ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ

ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Τεχνικές Ανάκτησης Πληροφορίας σε Περιβάλλον Cloud Computing

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Χρήστος Κ. Πανουτσακόπουλος

Επιβλέπων

Θεοδώρα Βαρβαρίγου

Καθηγήτρια ΕΜΠ

Αθήνα, Οκτώβριος 2015

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ

ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Τεχνικές Ανάκτησης Πληροφορίας σε Περιβάλλον Cloud Computing

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Χρήστος Κ. Πανουτσακόπουλος

Επιβλέπων

Θεοδώρα Βαρβαρίγου

Καθηγήτρια ΕΜΠ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 26η Οκτωβρίου 2015.

Θ. Βαρβαρίγου	Β. Λούμος	Ε. Καγιάφας
Καθηγήτρια ΕΜΠ	Καθηγητής ΕΜΠ	Καθηγητής ΕΜΠ

Αθήνα, Οκτώβριος 2015

Χρήστος Κ. Πανουτσακόπουλος

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π

ΠΕΡΙΛΗΨΗ

Στην σύγχρονη εποχή της πληροφορίας, η ανάκτηση χρήσιμων πληροφοριών από μεγάλους όγκους δεδομένων είναι ένα πρόβλημα που έχει αποκτήσει τεράστια σημασία για πολλές δραστηριότητες. Ιδιαίτερα τα τελευταία χρόνια, η εξέλιξη της τεχνολογίας του υπολογιστικού νέφους έχει οδηγήσει σε μια προσπάθεια να αξιοποιηθούν οι δυνατότητες που παρέχει αυτή η νέα τεχνολογία στον τομέα της ανάκτησης πληροφοριών.

Οι σχεσιακές βάσεις δεδομένων, εξακολουθούν να διαδραματίζουν σημαντικό ρόλο στις αρχιτεκτονικές υπολογιστικού νέφους, παρά την εξέλιξη αξιόλογων εναλλακτικών επιλογών. Βασικός στόχος της παρούσας διπλωματικής εργασίας είναι να εξεταστεί η επίδραση που έχει ο μετασχηματισμός του σχήματος μιας σχεσιακής βάσης δεδομένων σύμφωνα με την θεωρία boolean approach (και συγκεκριμένα ο μετασχηματισμός κατάλληλων κατηγορηματικών μεταβλητών σε σύνολα λογικών μεταβλητών), στον χρόνο εκτέλεσης σχεσιακών ερωτημάτων επί αυτής της βάσης δεδομένων.

Αρχικά, υλοποιήθηκε η σχεσιακή βάση δεδομένων που ορίζεται από το TPCDS Benchmark, το οποίο είναι κατάλληλα σχεδιασμένο για την προσομοίωση συστημάτων υποστήριξης αποφάσεων και περιλαμβάνει μια μεγάλη λίστα σχεσιακών ερωτημάτων διαφόρων τύπων και επιπέδων πολυπλοκότητας. Ακολούθως, δημιουργήθηκε ένα ακριβές αντίγραφο αυτής της βάσης δεδομένων στο οποίο εφαρμόστηκε ο μετασχηματισμός boolean approach. Στη δεύτερη βάση αποθηκεύτηκαν ακριβώς τα ίδια δεδομένα με την πρώτη, κατάλληλα τροποποιημένα ώστε να ταιριάζουν στις μετατροπές που επέφερε στο σχήμα της βάσης δεδομένων ο μετασχηματισμός boolean approach. Στη συνέχεια, αφού επιλέχτηκαν τα καταλληλότερα σχεσιακά ερωτήματα από όσα ορίζει το TPCDS Benchmark, αυτά εκτελέστηκαν επαναληπτικά στις δυο βάσεις δεδομένων και καταγράφηκαν οι αντίστοιχοι χρόνοι εκτέλεσης. Τέλος, συγκρίνοντας τους καταγραμμένους χρόνους εκτέλεσης για τις δυο βάσεις δεδομένων, εξάχθηκαν τα ανάλογα συμπεράσματα για την επίδραση του μετασχηματισμού boolean approach.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ

Ανάκτηση πληροφοριών, υπολογιστικό νέφος, σχεσιακή βάση δεδομένων, Boolean Approach, κατηγορηματική μεταβλητή, λογική μεταβλητή, χρόνος εκτέλεσης, σεχεσιακό σχήμα, σχεσιακό ερώτημα, TPCDS benchmark, βάση δεδομένων TPCDS, σχεσιακά ερωτήματα TPCDS

ABSTRACT

Information retrieval has become a problem of grate significance for many scientific and business domains. Especially now, with the important advances in the field of cloud computing providing increasingly prospectful capabilities, a lot of effort is being made in order to apply these capabilities in the field of information retrieval.

Despite the appearance of successful alternatives, relational databases continue to play an important role in cloud computing architectures. The main goal of this diploma thesis is to examine the influence of transforming the schema of a relational database according to the Boolean Approach theory (specifically, transforming appropriate categorical variables into sets of boolean varibles) on the execution time of relational queries on that database.

To begin with, the database specified by the TPCDS benchmark (which is a benchmark designed to emulate decision support systems and contains a big collection of relational queries) was implemented. After that, an exact copy of this database was created and the Boolean Approach transformation was applied to its scema. The data stored in the second database were logically identical to those stored in the first one; however they were manipulated in order to fit in the transformed database schema. The next step was to choose a subset of the queries defined by TPCDS and execute them repeatedly against the two databases, recording the execution times. Finally, conclusions regarding the effect of the Boolean Approach transformation were reached by compairing the execution times for both databases.

KEYWORDS

Information retrieval, cloud computing, relational database, Boolean Approach, categorical variable, boolean variable, execution time, relational schema, relational query, TPCDS benchmark, TPCDS database, TPCDS queries

ΠΕΡΙΕΧΟΜΕΝΑ

- ΠΑΡΑΡΤΗΜΑ Α: ΚΩΔΙΚΑΣ ΕΡΓΑΣΙΑΣ
  - TpcdsDB.java……ΣΕΛ 164
  - TpcdsBoolDB.java……ΣΕΛ 215
  - SetUp.java……ΣΕΛ 219
  - QueryMaker.java……ΣΕΛ 222
  - Queries.java……ΣΕΛ 225
  - TimeKeeper.java……ΣΕΛ 281
  - GraphMaker.java……ΣΕΛ 282

ΠΡΟΛΟΓΟΣ

Εισαγωγή

Στη σύγχρονη εποχή της πληροφορίας, το πρόβλημα της διαχείρισης μεγάλων όγκων δεδομένων και της ανάκτησης χρήσιμης πληροφορίας από τα δεδομένα αυτά, έχει αποκτήσει τεράστια σημασία. Η συλλογή και αποθήκευση δεδομένων και η αξιοποίηση τους για την ανάκτηση πληροφοριών και την εξαγωγή συμπερασμάτων, αποτελεί αναπόσπαστο κομμάτι δραστηριοτήτων που σχετίζονται με την επιστημονική έρευνα, την οικονομία και τις κοινωνικές επιστήμες. Γενικότερα, σε όλο το φάσμα της ανθρώπινης δραστηριότητας, η ανάκτηση και αξιοποίηση πληροφοριών θεωρείται το πρώτο και ουσιαστικότερο βήμα για την επίλυση ενός προβλήματος.

Επιπλέον, η ολοένα αυξανόμενη χρήση τεχνολογιών πληροφορικής από το σύνολο του παγκόσμιου πληθυσμού, έχει καταστήσει την διαχείριση δεδομένων ιδιαίτερα σημαντική και για την καθημερινότητα των ανθρώπων. Κάθε είδους πληροφορία που αφορά ένα φυσικό πρόσωπο μπορεί να αποθηκευτεί σε ηλεκτρονική μορφή, ώστε να προσπελαστεί και να υποστεί επεξεργασία. Δημογραφικές πληροφορίες, ιατρικά δεδομένα, στοιχεία τραπεζικών λογαριασμών, ακόμα και καταναλωτικές προτιμήσεις , αποτελούν μερικά μόνο παραδείγματα δεδομένων που μπορούν να αποθηκευτούν και να χρησιμοποιηθούν με διάφορους τρόπους. Πληροφορίες που προκύπτουν από την αξιοποίηση αυτών των δεδομένων, μπορούν να χρησιμοποιηθούν τόσο προς όφελος των ίδιων των ανθρώπων που τα δεδομένα αυτά αφορούν, όσο και προς όφελος οργανισμών και εταιριών, που αξιοποιούν τα δεδομένα αυτά για να βελτιώσουν τις υπηρεσίες και τη λειτουργία τους.

Στον τομέα της αποθήκευσης, διαχείρισης και αξιοποίησης δεδομένων, πολύ σημαντικός είναι ο ρόλος που διαδραματίζει η τεχνολογία των βάσεων δεδομένων. Πρόκειται για έναν τεχνολογικό κλάδο που έχει αναπτυχθεί θεαματικά τις τελευταίες δεκαετίες, παράλληλα με τον γενικότερο κλάδο της τεχνολογίας υπολογιστών. Στο σύνολο τους τα προϊόντα βάσεων δεδομένων, τόσο στην παραδοσιακή τους μορφή όσο και στις πιο σύγχρονες εκδοχές τους, παρέχουν μια οικονομική και υπολογιστικά αποδοτική λύση στο πρόβλημα της διαχείρισης δεδομένων. Σε μια βάση δεδομένων, τα δεδομένα αποθηκεύονται με τρόπο δομημένο και αποδοτικό. Επιπλέον μια βάση δεδομένων παρέχει μηχανισμούς που εξασφαλίζουν την ακεραιότητα, την συνέχεια και την ασφάλεια των δεδομένων της. Τέλος, κάθε βάση δεδομένων παρέχει μια διαπροσωπεία μέσω της οποίας οι χρήστες μπορούν να προσπελαύνουν και να ανανεώνουν τα δεδομένα της, καθιστώντας την αξιοποίηση τους εύκολη και αποδοτική.

Τα τελευταία χρόνια, με την ανάπτυξη του τεχνολογικού κλάδου του cloud computing, έχουν ανοίξει νέοι ορίζοντες όσον αφορά το πρόβλημα της διαχείρισης δεδομένων. Όλο και περισσότερο, παραδοσιακές αρχιτεκτονικές αποθήκευσης, διαχείρισης και επεξεργασίας δεδομένων, αντικαθιστώνται από νέα αρχιτεκτονικά μοντέλα, που αξιοποιούν τις δυνατότητες που παρέχει η νέα αυτή τεχνολογία. Ειδικότερα, στον κλάδο των βάσεων δεδομένων, καταβάλλεται κάθε προσπάθεια να αναπτυχθούν νέα προϊόντα, που θα αξιοποιούν τις υποδομές, τα πρωτόκολλα και τις υπηρεσίες του loud computing. Σε μια αρχιτεκτονική cloud computing, ένας χρήστης μπορεί να έχει πρόσβαση σε μια σειρά από υπηρεσίες, που φιλοξενούνται σε απομακρυσμένα μηχανήματα. Αναπόφευκτα, τα τελευταία χρόνια αναπτύσσονται συστήματα βάσεων δεδομένων cloud, γνωστά και ως συστήματα

Database as a Service (DaaS), που σκοπό έχουν να συνδυάσουν της υπηρεσίες μιας βάσης δεδομένων και μιας πλατφόρμας cloud computing, βρίσκοντας την χρυσή τομή όσον αφορά τις διαφορές στη φιλοσοφία της λειτουργίας τους.

Περιγραφή Εργασίας

Στα πλαίσια της παρούσας διπλωματικής εργασίας, εξετάζουμε την επίδραση που μπορεί να επιφέρει η εφαρμογή της θεωρίας boolean approach για την ανάκτηση πληροφοριών από σύνολα δεδομένων, στην αποδοτικότητα της εκτέλεσης ερωτημάτων (queries) σε σχεσιακές βάσεις δεδομένων.

Συγκεκριμένα, η θεωρία boolean approach προτείνει μεταξύ άλλων, τον μετασχηματισμό κατηγορηματικών μεταβλητών με μικρό εύρος τιμών σε σύνολα από δυαδικές μεταβλητές, (με εύρος τιμών το {0,1}). Σε μια σχεσιακή βάση δεδομένων, αυτό μεταφράζεται στην αντικατάσταση attributes των πινάκων της βάσης με σύνολα από δυαδικά attributes, κατά τρόπο που να μην επηρεάζονται τα δεδομένα που αποθηκεύονται σε αυτή.

Αρχικά υλοποιήθηκε μια βάση δεδομένων, και συγκεκριμένα η βάση δεδομένων του TPC benchmark DS, όπως αυτή περιγράφεται στο αντίστοιχο φύλλο προδιαγραφών. Το TPCDS αποτελεί ένα benchmark, ειδικά σχεδιασμένο για την προσομοίωση συστημάτων υποστήριξης αποφάσεων, που αποτελούν ένα τυπικό παράδειγμα δραστηριότητας με έμφυτη την ανάγκη επεξεργασίας τεράστιων όγκων δεδομένων. Στη συνέχεια, δημιουργήθηκε ένα ακριβές αντίγραφο της βάσης δεδομένων TPCDS, στο οποίο εφαρμόστηκε ο μετασχηματισμός boolean approach. Κατόπιν, επιλέχτηκε ένα σύνολο από queries, ένα υποσύνολο από τα σχεσιακά ερωτήματα του TPCDS benchmark, α οποία εμπλέκουν τα attributes των πινάκων της βάσης που μετασχηματίστηκαν κατά boolean approach. Τα ερωτήματα αυτά, αφού υπέστησαν κατάλληλους μετασχηματισμούς ώστε να μπορούν να υποβληθούν στη μετασχηματισμένη βάση δεδομένων, υποβλήθηκαν επαναληπτικά και στις δυο βάσεις και καταγράφηκε ο χρόνος εκτέλεσης τους και στις δύο περιπτώσεις. Τέλος, συγκρίνοντας τις χρονικές επιδόσεις στις δυο περιπτώσεις, προέκυψαν συμπεράσματα σχετικά με την επίδραση που μπορεί να έχει ο μετασχηματισμός boolean approach στην εκτέλεση σχεσιακών queries.

Μετά από αυτή τη σύντομη εισαγωγή, η εργασία περιλαμβάνει:

- Κεφάλαιο 1: Παραθέτει ορισμένες βασικές πληροφορίες σχετικά με τα παραδοσιακά συστήματα διαχείρισης βάσεων δεδομένων.
- Κεφάλαιο 2: Περιγράφει συνοπτικά τις βασικές έννοιες της τεχνολογίας του cloud computing.
- Κεφάλαιο 3: Περιλαμβάνει μια εκτενή περιγραφή του TPCDS benchmark και της αντίστοιχης βάσης δεδομένων.
- Κεφάλαιο 4: Παρουσιάζει το κομμάτι τον μετασχηματισμό boolean approach και εξηγεί πως αυτός εφαρμόστηκε στην TPCDS βάση δεδομένων.
- Κεφάλαιο 5: Αποτελεί το κύριο μέρος της εργασίας, όπου παρουσιάζονται τα queries που εκτελέστηκαν και τα χρονικά αποτελέσματα που καταγράφηκαν.
- Επίλογος: Καταγράφει τα τελικά συμπεράσματα της εργασίας.
- Παράρτημα Α: Ο κώδικας της εργασίας, σε γλώσσα JAVA.

ΚΕΦΑΛΑΙΟ 1: ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

Εισαγωγή

Οι βάσεις δεδομένων, αποτελούν έναν από τους σημαντικότερους τομείς της σύγχρονης πληροφορικής, με το θεωρητικό τους υπόβαθρο αλλά και τις καινοτόμες εφαρμογές τους να αποτελούν πεδίο συνεχούς επιστημονικής έρευνας. Παράλληλα, αποτελούν ένα ισχυρό εργαλείο για μια ευρεία γκάμα δραστηριοτήτων, μεταξύ των οποίων η βιομηχανία, το εμπόριο, οι μεταφορές, οι τηλεπικοινωνίες, η ιατρική και η επιστημονική έρευνα.

Συνειδητά ή όχι, κάθε χρήστης του διαδικτύου χρησιμοποιεί βάσεις δεδομένων καθημερινά για να φέρει σε πέρας βασικές δραστηριότητες του. Κάθε φορά που κάποιος αναζητά μια πληροφορία, πραγματοποιεί μια ηλεκτρονική αγορά, κλείνει ένα αεροπορικό εισιτήριο ή ενημερώνεται για το τραπεζικό του υπόλοιπο μέσω διαδικτύου, στο παρασκήνιο της διεπαφής του με τον αγαπημένο του περιηγητή, συναλλάσσεται με μια σειρά από βάσεις δεδομένων, χωρίς απαραίτητα να το αντιληφθεί.

Πέρα από τον απλό χρήστη του διαδικτύου, οι βάσεις δεδομένων έχουν ευρεία εφαρμογή από εταιρίες και οργανισμούς, που τις χρησιμοποιούν για να αποθηκεύουν τεράστιους όγκους πληροφοριών. Για παράδειγμα, μια εταιρία μπορεί να αποθηκεύει πληροφορίες για τους εργαζόμενους της, του πελάτες της, τα περιουσιακά της στοιχεία και τα προϊόντα της. Αντίστοιχα, ένα εκπαιδευτικό ίδρυμα μπορεί να αποθηκεύει δεδομένα σχετικά με τους σπουδαστές του, τους καθηγητές του, τα μαθήματα που προσφέρει και τα ερευνητικά του προγράμματα.

Μια **βάση δεδομένων**, είναι μια συλλογή από σχετιζόμενα μεταξύ τους δεδομένα, οργανωμένα με έναν δομημένο τρόπο. Ένα **σύστημα διαχείρισης βάσεων δεδομένων**, αποτελείται από ένα σύνολο προγραμμάτων που χρησιμοποιούνται για την πρόσβαση σε μια βάση δεδομένων, με σκοπό την ανάκτηση δεδομένων, την προσθήκη νέων δεδομένων, την τροποποίηση δεδομένων ή τη διαγραφή δεδομένων από τη βάση. Τα συστήματα διαχείρισης βάσεων δεδομένων σχεδιάζονται με τρόπο που να τους επιτρέπει να χειρίζονται τεράστιους όγκους πληροφοριών. Η διαχείριση των δεδομένων περιλαμβάνει τον ορισμό δομών για την αποθήκευση των πληροφοριών , καθώς και την παροχή μηχανισμών που επιτρέπουν τον χειρισμό τους. Επιπλέον, ένα σύστημα διαχείρισης βάσεων δεδομένων οφείλει να διασφαλίζει τόσο την συνέπεια όσο και την ασφάλεια των αποθηκευμένων δεδομένων.

Πλεονεκτήματα των Συστημάτων Βάσεων Δεδομένων

Τα συστήματα διαχείρισης βάσεων δεδομένων αντιμετωπίζουν μια σειρά από μειονεκτήματα που παρουσιάζει η αποθήκευση δεδομένων σε ένα τυπικό σύστημα αρχείων. Μερικά από αυτά τα μειονεκτήματα είναι η επαναληπτικότητα η απομόνωση και η ασυνέπεια των δεδομένων, η δυσκολία στην πρόσβαση των δεδομένων, καθώς και προβλήματα ατομικότητας, ταυτόχρονης πρόσβασης και ασφαλείας του χειρισμού των δεδομένων.

Ένα ακόμη γνώρισμα των συστημάτων βάσεων δεδομένων είναι η **αφαιρετικότητα**. Συγκεκριμένα, οι περίπλοκες δομές δεδομένων που χρησιμοποιούνται για να αναπαριστούν τις αποθηκευμένες πληροφορίες αποκρύπτονται εντελώς από τον χρήστη της βάσης, μέσω διαφόρων επιπέδων αφαιρετικότητας, που σκοπό έχουν την απλοποίηση της διαδικασίας συνδιαλλαγής του χρήστη με το σύστημα. Σε ένα σύστημα διαχείρισης βάσεων δεδομένων υπάρχουν τρία επίπεδα αφαιρετικότητας: Το **φυσικό επίπεδο**, που περιγράφει λεπτομερώς τις περίπλοκες δομές δεδομένων που χρησιμοποιούνται για την αποθήκευση των δεδομένων, **το λογικό επίπεδο**, που περιγράφει ποια δεδομένα αποθηκεύονται στη βάση καθώς και ποιες είναι οι σχέσεις μεταξύ τους, και το **επίπεδο προβολής** που περιλαμβάνει ένα μέρος μόνο ολόκληρης της βάσης δεδομένων.

Στιγμιότυπα και Σχήματα

Οι βάσεις δεδομένων αλλάζουν με την πάροδο του χρόνου, εισάγονται νέα δεδομένα, τροποποιούνται και διαγράφονται παλιά. Το σύνολο των πληροφοριών που περιέχει μια βάση δεδομένων σε μια δεδομένη χρονική στιγμή ονομάζεται **στιγμιότυπο** της βάσης δεδομένων. Αντιθέτως, η καθολική σχεδίαση της βάσης δεδομένων, που περιγράφει τη δομή των δεδομένων που αποθηκεύονται σε αυτή καθώς και τις μεταξύ τους σχέσεις, ονομάζεται **σχήμα** της βάσης δεδομένων. Είναι προφανές ότι μια αλλαγή στα περιεχόμενα της βάσης (πχ η εισαγωγή ενός νέου στοιχείου, ή η τροποποίηση μιας αποθηκευμένης τιμής) επηρεάζει το τρέχον στιγμιότυπο της βάσης, όχι όμως και το σχήμα της, το οποίο παραμένει αναλλοίωτο.

Σχεσιακό Μοντέλο Δεδομένων – Σχεσιακές Βάσεις Δεδομένων

Το πιο συνηθισμένο μοντέλο δεδομένων που χρησιμοποιείται ευρύτατα στα σύγχρονα συστήματα βάσεων δεδομένων και ίσως το καλύτερα θεμελιωμένο θεωρητικά, είναι το **σχεσιακό μοντέλο δεδομένων.**

Το μοντέλο αυτό χρησιμοποιεί ένα σύνολο από πίνακες που αντιπροσωπεύουν τα δεδομένα και τις σχέσεις μεταξύ τους. Οι πίνακες είναι γνωστοί κι ως **σχέσεις (relations)**. Κάθε σχέση έχει ένα σύνολο από **ιδιότητες (attributes)** , που αναπαριστώνται από τις στήλες του αντίστοιχου πίνακα. Κάθε στήλη του πίνακα έχει ένα όνομα, που είναι και το όνομα της αντίστοιχης ιδιότητας. Για κάθε ιδιότητα μιας σχέσης, υπάρχει ένα αντίστοιχο **πεδίο τιμών**, επομένως η αντίστοιχη στήλες παίρνει τιμές από αυτό το πεδίο ή την ειδική τιμή **κενό (null)**, που υποδεικνύει ότι μια συγκεκριμένη τιμή είναι άγνωστη ή δεν υπάρχει. Οι γραμμές των πινάκων αποτελούν τις **εγγραφές (records)** της βάσης δεδομένων, δηλαδή τα δεδομένα που είναι αποθηκευμένα στη βάση σε κάποια χρονική στιγμή. Από μαθηματική άποψη, κάθε γραμμή ενός αποτελεί μια **ν-άδα τιμών**, δηλαδή μια λίστα ν τιμών, όπου ν το πλήθος των στηλών του πίνακα.

Ένα πολύ σημαντικό ζήτημα σε μια σχεσιακή βάση δεδομένων, είναι να μπορούμε να διαφοροποιούμε τις εγγραφές μιας σχέσης, και πιο συγκεκριμένα να μπορούμε να προσδιορίσουμε μονοσήμαντα μια συγκεκριμένη εγγραφή μιας σχέσης, με βάση τις τιμές των ιδιοτήτων της. Δεν επιτρέπεται δυο διαφορετικές εγγραφές να έχουν ίδια τιμή σε όλες τις ιδιότητες. Ένα **υπέρ-κλειδί (superkey)** είναι ένα σύνολο από ιδιότητες μιας σχέσης, που αν ληφθούν υπόψη συνολικά, μας επιτρέπουν να προσδιορίσουμε μοναδικά μια εγγραφή αυτής της σχέσης. Ένα υπέρ-κλειδί μπορεί να περιέχει πολλές

ιδιότητες, και φυσικά μια σχέση μπορεί να έχει πολλά υπέρ-κλειδιά. Ένα ελάχιστο υπέρ-κλειδί, δηλαδή ένα σύνολο ιδιοτήτων που αποτελεί υπέρ-κλειδί μιας σχέσης, αλλά κανένα υποσύνολο του δεν αποτελεί κι αυτό υπέρ-κλειδί της σχέσης, λέγεται **υποψήφιο κλειδί (candidate key)** της σχέσης. Μια σχέση είναι πιθανό να έχει πολλά υποψήφια κλειδιά. Ο σχεδιαστής της βάσης δεδομένων επιλέγει ένα από τα υποψήφια κλειδιά μιας σχέσης ως το κύριο μέσο διαφοροποίησης τον εγγραφών της, δηλαδή ως το **πρωτεύον κλειδί (primary key)** της σχέσης. Φυσικά, απαγορεύεται δυο εγγραφές να έχουν ίδια τιμή για όλες τις ιδιότητες που συνθέτουν το πρωτεύον κλειδί. Η επιλογή του πρωτεύοντος κλειδιού μιας σχέσης πρέπει να γίνεται με μεγάλη προσοχή. Οι τιμές των ιδιοτήτων του, καλό είναι να μην αλλάζουν ποτέ, ή να αλλάζουν σπάνια. Για παράδειγμα, ο ΑΦΜ ενός πολίτη είναι μια καλή ιδιότητα για να συμμετέχει στη συγκρότηση ενός πρωτεύοντος κλειδιού, αφού κανονικά δεν αλλάζει ποτέ. Αντίθετα, το τραπεζικό του υπόλοιπο, το οποίο μπορεί να μεταβάλλεται καθημερινά, δεν αποτελεί καλή επιλογή.

Μια σχέση r1 μπορεί να περιλαμβάνει στις ιδιότητές της το πρωτεύον κλειδί k μιας άλλης σχέσης r2 της βάσης δεδομένων. Στην περίπτωση αυτή, το k αποτελεί **ξένο κλειδί (foreign key)** της r1, η οποία αναφέρεται στην r2. Η σχέση r1 καλείται **σχέση αναφοράς** και η r2 καλείται **αναφερόμενη σχέση** του ξένου κλειδιού.

Ιδιαίτερο ενδιαφέρον παρουσιάζουν μια σειρά από πράξεις που μπορούν να εκτελεστούν επί ή μεταξύ των σχέσεων μιας σχεσιακής βάσης δεδομένων, οι λεγόμενες **σχεσιακές πράξεις.** Βασικό χαρακτηριστικό των σχεσιακών πράξεων είναι ότι πάντα δίνουν ως αποτέλεσμα μια νέα σχέση. Οι πιο συνηθισμένες λειτουργίες επί μιας σχέσης είναι να πάρουμε ένα υποσύνολο των εγγραφών της, δηλαδή μια σχέση με τις ίδιες ιδιότητες με την αρχική αλλά λιγότερες εγγραφές, ή να πάρουμε ένα υποσύνολο των ιδιοτήτων της, δηλαδή μια σχέση με ίδιο πλήθος εγγραφών αλλά μόνο τις επιλεγμένες ιδιότητες. Περισσότερο ενδιαφέρον παρουσιάζουν οι πράξεις μεταξύ δυο σχέσεων. Η πράξη **σύνδεσης (join)** συνδυάζει ζευγάρια εγγραφών ένα από κάθε σχέση, σε μια νέα εγγραφή. Η πιο συνηθισμένη μορφή σύνδεσης είναι η **φυσική σύνδεση (natural join)**, η οποία αντιστοιχεί εγγραφές οι οποίες έχουν ίδια τιμή σε όλες τις κοινές ιδιότητες των δυο σχέσεων. Τέλος, το **καρτεσιανό γινόμενο (Cartesian product)** είναι μια πράξη που συνδέει όλα τα ζεύγη εγγραφών των δύο σχέσεων, ανεξάρτητα από τις τιμές των ιδιοτήτων τους.

Ένα συγκεκριμένο σύνολο εγγραφών που περιέχονται σε μια σχέση σε κάποια χρονική στιγμή αποτελούν το τρέχον **στιγμιότυπο** της σχέσης. Για να ορίσουμε το **σχήμα** μιας σχεσιακής βάσης δεδομένων, αρκεί να παραθέσουμε τις σχέσεις (πίνακες) που περιλαμβάνει, τις ιδιότητες τους (ονόματα στηλών), τα πεδία τιμών των ιδιοτήτων, καθώς και τα κλειδιά και το είδος τους.

Τα περισσότερα σύγχρονα συστήματα διαχείρισης βάσεων δεδομένων βασίζονται στο σχεσιακό μοντέλο δεδομένων. Εναλλακτικά μοντέλα δεδομένων αποτελούν το μοντέλο οντότητας-συσχέτισης, το αντικειμενοστραφές μοντέλο και το ημιδομημένο μοντέλο, ενώ το δικτυακό και το ιεραρχικό μοντέλο θεωρούνται ξεπερασμένα.

Γλώσσες Βάσεων Δεδομένων

Ένα σύστημα διαχείρισης βάσεων δεδομένων παρέχει μια **γλώσσα ορισμού δεδομένων (Data Definition Language - DDL)** με την οποία μπορεί να καθοριστεί το σχήμα μιας βάσης δεδομένων, και μια **γλώσσα χειρισμού δεδομένων (Data Manipulation Language-DML)** με την οποία διατυπώνονται τα

ερωτήματα (queries) και οι **ενημερώσεις (updates)** της βάσης δεδομένων. Συνήθως, DML και DDL δεν αποτελούν ξεχωριστές γλώσσες, αλλά συνιστώσες μιας ενιαίας γλώσσας, κάτι που ισχύει και στην περίπτωση της ευρύτερα χρησιμοποιούμενης γλώσσας βάσεων δεδομένων, την **SQL**.

Με τη γλώσσα χειρισμού δεδομένων δίνεται η δυνατότητα στους χρήστες της βάσης να αποκτούν πρόσβαση στα δεδομένα που είναι αποθηκευμένα στη βάση δεδομένων και να τα χειρίζονται. Ο χειρισμός των δεδομένων περιλαμβάνει την ανάκληση πληροφοριών από τη βάση, την εισαγωγή νέων πληροφοριών στη βάση, τη διαγραφή πληροφοριών από τη βάση και την τροποποίηση πληροφοριών που βρίσκονται αποθηκευμένες στη βάση. Υπάρχουν δυο βασικοί τύποι DML: Οι **διαδικαστικές DML**, που απαιτούν από τον χρήστη να καθορίζει ποια δεδομένα χρειάζονται αλλά και πως αυτά τα δεδομένα να λαμβάνονται, και οι **δηλωτικές (μη διαδικαστικές) DML,** που απαιτούν από το χρήστη να καθορίσει μόνο ποια δεδομένα πρέπει να ληφθούν. Το σημαντικότερο μέρος κάθε DML είναι εκείνο που αφορά την ανάκληση πληροφοριών από μια βάση δεδομένων. Αυτό πραγματοποιείται με προτάσεις που λέγονται **ερωτήματα (queries)**. Το κομμάτι μιας DML που αφορά την ανάκληση πληροφοριών μέσω ερωτημάτων λέγεται **γλώσσα ερωτημάτων (query language)**. Οι όροι *γλώσσα ερωτημάτων* και *γλώσσα χειρισμού δεδομένων* συχνά χρησιμοποιούνται με την ίδια έννοια, χωρίς ωστόσο αυτό να είναι τεχνικά σωστό.

Με την γλώσσα ορισμού δεδομένων εκτός από το σχήμα μιας βάσης δεδομένων μπορούν να καθοριστούν και κάποια επιπλέον χαρακτηριστικά των δεδομένων. Με χρήση εντολών που ανήκουν σε ένα ειδικό μέρος μιας DDL, που λέγεται **γλώσσα αποθήκευσης και ορισμού δεδομένων**, καθορίζονται η δομή αποθήκευσης και οι μέθοδοι πρόσβασης που χρησιμοποιούνται από το σύστημα της βάσης δεδομένων, λεπτομέρειες που φυσικά αποκρύπτονται από τον χρήστη του συστήματος. Επιπλέον, μέσω της DDL καθορίζονται οι **περιορισμοί συνέπειας** που πρέπει να ικανοποιούν τα δεδομένα της βάσης και οι οποίοι ελέγχονται μετά από κάθε τροποποίηση των περιεχομένων της. Μερικές χαρακτηριστικές κατηγορίες περιορισμών συνέπειας των δεδομένων είναι οι ακόλουθες:

- **Περιορισμοί πεδίου τιμών**: Στα πλαίσια του σχεσιακού μοντέλου δεδομένων, κάθε ιδιότητα πρέπει να έχει ένα αντίστοιχο πεδίο επιτρεπτών τιμών (πχ ακέραιοι αριθμοί, χαρακτήρες, συμβολοσειρές, τύπος ημερομηνίας/ώρας). Κάθε φορά που εισάγεται μια νέα εγγραφή σε έναν πίνακα στη βάση δεδομένων, πρέπει να ελέγχεται κατά πόσο οι τιμές της εγγραφής αυτής για τις διάφορες στήλες (ιδιότητες) του πίνακα ανήκουν στα πεδία επιτρεπτών τιμών των αντίστοιχων ιδιοτήτων. Φυσικά ανάλογοι έλεγχοι πρέπει να γίνονται όχι μόνο κατά την εισαγωγή δεδομένων στη βάση, αλλά και κατά την τροποποίηση δεδομένων που ήδη περιέχει.
- **Περιορισμοί Ακεραιότητας αναφορών**: Εξασφαλίζουν ότι όλες οι τιμές μιας στήλης σε έναν πίνακα εμφανίζονται σε μια σημασιολογικά αντίστοιχη στήλη ενός άλλου πίνακα της βάσης. Για παράδειγμα, σε μια βάση δεδομένων ενός εμπορικού καταστήματος, υπάρχει ένας πίνακας *PRODUCTS* που αποθηκεύει πληροφορίες για τα προϊόντα του καταστήματος και ένας πίνακας *SALES* που αποθηκεύει πληροφορίες για τις πωλήσεις του καταστήματος. Ο πίνακας *PRODUCTS* έχει μεταξύ άλλων μία στήλη *product_name* και ο πίνακας *SALES* έχει μεταξύ άλλων μία στήλη *sold_product_name.* Είναι προφανές ότι για να πουληθεί ένα προϊόν πρέπει να είναι καταχωρημένο στον κατάλογο του καταστήματος, επομένως κάθε τιμή που είναι καταχωρημένη στη στήλη *sold_product_name* του πίνακα *SALES* πρέπει να ελέγχεται ότι υπάρχει και στη στήλη *product_name* του πίνακα *PRODUCTS.*

- **Διασφαλίσεις (assertion):** Αποτελούν γενικές συνθήκες που πρέπει πάντα να ικανοποιούν τα περιεχόμενα σε μια βάση δεδομένων. Εφόσον καθοριστεί μια διασφάλιση, η τήρηση της ελέγχεται σε κάθε τροποποίηση των περιεχομένων της βάσης.
- **Εξουσιοδοτήσεις (authorization):** Είναι πολύ συνηθισμένη πρακτική οι χρήστες μιας βάσης δεδομένων να χωρίζονται σε διαφορετικές ομάδες, καθεμία από τις οποίες έχει διαφορετικά δικαιώματα πρόσβασης στα δεδομένα της βάσης. Η διαφοροποιήσεις αυτές μεταξύ των χρηστών προσδιορίζονται από περιορισμούς όπως εξουσιοδότηση ανάγνωσης, εξουσιοδότηση εισαγωγής, εξουσιοδότηση τροποποίησης και εξουσιοδότηση διαγραφής δεδομένων. Σε κάθε χρήστη της βάσης εκχωρείται ένας συνδυασμός αυτών των εξουσιοδοτήσεων.

Πρέπει να τονιστεί ιδιαίτερα ότι οι γλώσσες βάσεων δεδομένων, όπως για παράδειγμα η SQL δεν είναι γλώσσες προγραμματισμού εφαρμογών. Για την ακρίβεια, δεν είναι ισοδύναμες με μια υπολογιστική μηχανή Turing, που σημαίνει ότι υπάρχουν υπολογισμοί που μπορούν να πραγματοποιηθούν με μια γενική γλώσσα προγραμματισμού, όχι όμως με μια γλώσσα όπως η SQL. Στις εφαρμογές που χρησιμοποιούν βάσεις δεδομένων, προγράμματα εφαρμογών γραμμένα σε μια γλώσσα προγραμματισμού υψηλού επιπέδου, όπως η Java, η PHP, η Python και η C++, χρησιμοποιούν ενσωματωμένες εντολές σε μια γλώσσα όπως η SQL ώστε να αλληλεπιδράσουν με τη βάση δεδομένων.

Η γλώσσα ερωτημάτων SQL

Η **γλώσσα SQL** (**Structured Query Language**) αποτελεί την ευρύτερα χρησιμοποιούμενη γλώσσα ερωτημάτων στα σύγχρονα συστήματα διαχείρισης βάσεων δεδομένων. Αναπτύχθηκε στις αρχές της δεκαετίας του ’70 από την IBM, ενώ από το 1986 και μετά, το American National Standards Institute (ANSI) και το International Organization for Standardization (ISO) δημοσιεύουν τυποποιήσεις της γλώσσας, που παίρνουν το όνομα τους από το έτος της δημοσίευσης τους (SQL-86, SQL-89, SQL-99, SQL-2006 κτλ). Η SQL αποτελείται από δυο κύρια μέρη:

- DML (Data Manipulation Language – Γλώσσα Χειρισμού Δεδομένων)
  Το κομμάτι αυτό δίνει τη δυνατότητα σύνταξης ερωτημάτων (queries) για την ανάκτηση συγκεκριμένων εγγραφών της βάσης δεδομένων, εισαγωγή ή διαγραφή εγγραφών από τη βάση και την τροποποίηση εγγραφών της βάσης.

- DDL (Data Definition Language – Γλώσσα Ορισμού Δεδομένων)
  Το κομμάτι αυτό της SQL παρέχει τη δυνατότητα ορισμού, τροποποίησης και διαγραφής σχεσιακών σχημάτων. Επιπλέον, περιλαμβάνει εντολές για τον καθορισμό περιορισμών ακεραιότητας, τον ορισμό προβολών μιας βάσης δεδομένων, τον έλεγχο συναλλαγών και τον καθορισμό δικαιωμάτων σε σχέσεις και προβολές μιας βάσης δεδομένων. Τέλος, παρέχει τις δυνατότητες της ενσωματωμένης και της δυναμικής SQL, που είναι οι δυο τρόποι χρήσης της SQL μέσω μιας γενικής γλώσσας προγραμματισμού.

Παρακάτω, για λόγους πληρότητας, παρουσιάζεται συνοπτικά η σύνταξη των πιο βασικών εντολών της γλώσσας SQL.

Ορισμός Σχέσης

create table relation

(Atr1 Type1, Atr2 Type2, …, AtrN TypeN,

<integrity constraint 1>,…, < integrity constraint K>);

Με μια εντολή αυτής της μορφής ορίζεται μια σχέση με όνομα *relation,* η οποία έχει N τω πλήθος ιδιότητες και K τω πλήθος περιορισμούς ακεραιότητας. Για κάθε ιδιότητα (attribute) ορίζεται το όνομα και ο τύπος του, επομένως και το πεδίο τιμών της. Ορισμένοι βασικοί τύποι είναι:

- **char(n):** Συμβολοσειρά σταθερού μήκους n χαρακτήρων
- **varchar(n):** Συμβολοσειρά μεταβλητού μήκους έως n χαρακτήρων
- **integer:** Ακέραιος αριθμός
- **numeric(a,b):** Αριθμός σταθερής υποδιαστολής με a ψηφία (συν ένα πρόσημο), με τα b από αυτά να είναι δεξιά της υποδιαστολής.
- **float(n):** Αριθμός κινητής υποδιαστολής με ακρίβεια τουλάχιστον n ψηφία.
- **date:** Μια ημερομηνία, που περιλαμβάνει έτος, μήνα και μέρα του μήνα.
- **time:** Μια χρονική στιγμή εκφρασμένη σε ώρες, λεπτά και δευτερόλεπτα.
- **timestamp:** Ένας συνδυασμός date και time.

Επιπλέον, η SQL υποστηρίζει διάφορους περιορισμούς ακεραιότητας, με πιο σημαντικούς τους παρακάτω:

- **primary key(a1,a2,…,ak)**: Ορίζει τον συνδυασμό των ιδιοτήτων a1,…,ak ως πρωτεύον κλειδί της σχέσης. Εξορισμού, οι ιδιότητες a1,…,ak πρέπει να είναι not null και unique.
- **foreign key(a1,a2,…,ak) references r**: Ορίζει τον συνδυασμό των ιδιοτήτων a1,…,ak ως ξένο κλειδί της σχέσης με αναφερόμενη σχέση την r.
- **not null**: Απαγορεύει η ιδιότητα αυτή να παίρνει κενή τιμή.

Εισαγωγή/Διαγραφή εγγραφών από Σχέση

insert into *relation* **values (v1,v2,…vN);**

Μια εντολή σαν κι αυτή εισάγει στη σχέση *relation* μια εγγραφή με τις τιμές v1,…,vN για τις αντίστοιχες ιδιότητες, με τη σειρά που αναγράφονται.

delete from *relation ;*

Μια εντολή σαν κι αυτή διαγράφει όλες τις εγγραφές της σχέσης *relation.*

Τροποποίηση Σχήματος

drop table *relation***;**

Μια εντολή σαν κι αυτή διαγράφει εντελώς τη σχέση *relation* από το σχήμα της βάσης και μαζί, όπως είναι φυσικό, όλες τις εγγραφές της.

alter table *relation* **add** *attribute type;*

Μια εντολή σαν κι αυτή προσθέτει στο σχήμα της σχέσης *relation* την ιδιότητα *attribute* που είναι τύπου *type*.

alter table *relation* **delete** *attribute;*

Μια εντολή σαν κι αυτή διαγράφει από το σχήμα της σχέσης *relation* την ιδιότητα *attribute*.

Βασική Δομή Ερωτημάτων

select *atr1 ,…, atrN*
from *relation*;

Ένα ερώτημα σαν το παραπάνω, επιστρέφει έναν πίνακα που περιέχει όλες τις γραμμές της σχέσης *relation, μόνο όμως τις στήλες που αναφέρονται ρητά, δηλαδή τις στήλες που αντιστοιχούν στις ιδιότητες atr1 ,…, atrN.* Για παράδειγμα, από έναν πίνακα με πληροφορίες σχετικά με τους πελάτες ενός καταστήματος, θα μπορούσαμε με ένα ερώτημα αυτής της μορφής να ανακτήσουμε τα επίθετα και τις διευθύνσεις όλων των πελατών.

select *atr1 ,…, atrN*
from *relation*;
where *conditions*;

Ένα ερώτημα σαν το παραπάνω, επιστρέφει έναν πίνακα που περιέχει μόνο τις στήλες της σχέσης *relation* που αντιστοιχούν στις ιδιότητες *atr1 ,…, atrN,* και μόνο τις γραμμές για τις οποίες ικανοποιείται το κατηγόρημα *conditions,* το οποίο είναι μια λογική συνθήκη που ορίζει περιορισμούς στις εγγραφές που θα ανακτηθούν. . Για παράδειγμα, από έναν πίνακα με πληροφορίες σχετικά με τους πελάτες ενός καταστήματος, με ένα ερώτημα αυτής της μορφής θα μπορούσαμε να ανακτήσουμε τα email όσων πελατών έχουν πραγματοποιήσει αγορές τουλάχιστον 1000€ και μόνο με χρήση πιστωτικής κάρτας.

select *atrA1 ,…, atrAn, atrB1, …, atrBm*
from *relationA, relationB*;
where *conditions*;

Η παραπάνω μορφή ερωτήματος είναι σαφώς πιο χρήσιμη από τις προηγούμενες, αλλά και πολύ πιο απαιτητική υπολογιστικά. Ένα ερώτημα σαν κι αυτό, συνδυάζει τις εγγραφές των σχέσεων *relationA* και *relationB* ανά δύο, κρατώντας μόνο τα ζεύγη εγγραφών για τα οποία είναι αληθές το κατηγόρημα *conditions.* Από την προκύπτουσα σχέση, παίρνουμε σαν αποτέλεσμα του ερωτήματος μόνο τις ζητούμενες στήλες *atrA1 ,…, atrAn* από την σχέση *relationA* και *atrB1, …, atrBm* από την σχέση *relationB.* Ερωτήματα αυτής της μορφής συνδυάζουν πληροφορίες διαφορετικών σχέσεων, παρέχοντας περισσότερες δυνατότητες στον χρήστη της βάσης δεδομένων. Για παράδειγμα, από έναν πίνακα με πληροφορίες σχετικά με τους πελάτες ενός καταστήματος και έναν δεύτερο πίνακα σχετικά με τις πωλήσεις που έχει καταγράψει το κατάστημα, ένα ερώτημα σαν κι αυτό θα συνδυάσει κάθε πελάτη από τον πρώτο πίνακα με τις πωλήσεις στις οποίες αυτός ήταν ο αγοραστής από τον δεύτερο πίνακα και από την προκύπτουσα σχέση θα μπορούσε να επιστρέψει πληροφορίες όπως το ΑΦΜ του πελάτη και πληρωτέο ποσό της πώλησης. Αυτή η διαδικασία συνένωσης των εγγραφών δύο σχέσεων που έχουν ίδιες τιμές στις κοινές ιδιότητες των δυο σχέσεων, λέγεται **φυσικός σύνδεσμος (natural join)**.

<u>Αποθήκευση και Επεξεργασία Δεδομένων - Λειτουργικές Συνιστώσες Συστημάτων Διαχείρισης Βάσεων Δεδομένων</u>

Ένα σύστημα διαχείρισης βάσεων δεδομένων περιλαμβάνει δυο κύρια λειτουργικά συστατικά. Το πρώτο είναι ο **διαχειριστής αποθήκευσης** και το δεύτερο ο **επεξεργαστής ερωτημάτων**. Ο διαχειριστής αποθήκευσης είναι ιδιαίτερα σημαντικός επειδή οι βάσεις δεδομένων απαιτούν γενικά μεγάλους χώρους αποθήκευσης δεδομένων. Είναι υπεύθυνος για την αλληλεπίδραση του συστήματος διαχείρισης βάσεων δεδομένων με το σύστημα αρχείων όπου είναι αποθηκευμένα τα ακατέργαστα δεδομένα, μεταφράζοντας τις DML προτάσεις σε εντολές χαμηλού επιπέδου του συστήματος αρχείων. Συνεπώς, ο διαχειριστής αποθήκευσης είναι υπεύθυνος για την αποθήκευση, την ανάκτηση και τις ενημερώσεις των δεδομένων που βρίσκονται στη βάση δεδομένων. Ο διαχειριστής αποθήκευσης με τη σειρά του αποτελείται από τέσσερα διακριτά συστατικά: Τον **διαχειριστή εξουσιοδότησης και ακεραιότητας**, που διασφαλίζει την ικανοποίηση των περιορισμών ακεραιότητας και ελέγχει τη

δυνατότητα πρόσβασης των χρηστών στα δεδομένα της βάσης, τον **διαχειριστή συναλλαγών**, που ελέγχει την ομαλή εκτέλεση ταυτόχρονων συναλλαγών με τη βάση και τη διατήρηση της συνέπειας των δεδομένων της, τον **διαχειριστή αρχείων**, που διαχειρίζεται τον αποθηκευτικό χώρο και τις δομές δεδομένων που χρησιμοποιούνται για την αποθήκευση των δεδομένων στον δίσκο, και τον **διαχειριστή δεσμευμένης μνήμης (buffer)**, ο οποίος μεταφέρει δεδομένα από τον δίσκο στην κύρια μνήμη και τη μνήμη cache, επιτρέποντας στη βάση να χειρίζεται μεγέθη δεδομένων που υπερβαίνουν συντριπτικά το μέγεθος της κύριας μνήμης. Ο διαχειριστής αποθήκευσης χειρίζεται διάφορες δομές δεδομένων για να επιτελέσει τις λειτουργίες του. Μεταξύ αυτών χειρίζεται **αρχεία δεδομένων**, όπου αποθηκεύονται τα δεδομένα της βάσης, **λεξικά δεδομένων**, που διατηρούν μεταδεδομένα για το σχήμα και γενικότερα τη δομή της βάσης, και **ευρετήρια**, που παρέχουν μηχανισμούς γρήγορης πρόσβασης σε δεδομένα της βάσης. Από τη μεριά του ο επεξεργαστής ερωτημάτων αποτελείται από τρία επί μέρους λειτουργικά στοιχεία: Τον **διερμηνέα DDL,** που ερμηνεύει τις DLL προτάσεις και καταγράφει τους ορισμούς στο λεξικό δεδομένων, τον **μεταγλωττιστή DML,** ο οποίος μεταφράζει τις DML προτάσεις από τη γλώσσα διατύπωσης ερωτημάτων σε εντολές χαμηλού επιπέδου και εκτελεί βελτιστοποίηση ερωτημάτων, και την **μηχανή αξιολόγησης ερωτημάτων,** που εκτελεί τις εντολές που παράγει ο μεταγλωττιστής DML.

Συναλλαγές

Κατά την επεξεργασία των περιεχομένων μιας βάσης δεδομένων, είναι σύνηθες φαινόμενο ένα σύνολο από λειτουργίες να συνθέτουν μια ενιαία λογική μονάδα εργασίας. Ένα τέτοιο σύνολο λειτουργιών ονομάζεται **συναλλαγή (transaction)**. Με άλλα λόγια, μια συναλλαγή είναι μια μονάδα εκτέλεσης ενός προγράμματος που προσπελαύνει και πιθανώς τροποποιεί δεδομένα σε μια βάση δεδομένων.

Οι συναλλαγές αποτελούν έναν δομημένο τρόπο αλληλεπίδρασης με μια βάση δεδομένων. Σε ένα σωστά σχεδιασμένο σύστημα διαχείρισης βάσεων δεδομένων, οι συναλλαγές έχουν κάποιες πολύ σημαντικές ιδιότητες:

- **Ατομικότητα (Atomicity)**: Μια συναλλαγή είτε πραγματοποιείται ολόκληρη είτε δεν πραγματοποιείται καθόλου. Ο χρήστης της βάσης δεδομένων αντιλαμβάνεται μια συναλλαγή ως μια ενιαία λειτουργία, ωστόσο μια αυτή μπορεί να περιλαμβάνει περισσότερες επιμέρους λειτουργίες. Οι λειτουργίες που συνθέτουν μια συναλλαγή επιτρέπεται να εκτελεστούν είτε όλες μαζί είτε καμία.
- **Συνέπεια (Consistency):** Η εκτέλεση μιας συναλλαγής, και χωρίς να λαμβάνεται υπόψη οποιαδήποτε άλλη, διατηρεί πάντα τη συνέπεια των δεδομένων που είναι αποθηκευμένα στη βάση δεδομένων.
- **Απομόνωση (Isolation):** Παρά το γεγονός ότι ενδέχεται να εκτελούνται ταυτόχρονα πολλές συναλλαγές, κάθε συναλλαγή 'αγνοεί' την παρουσία των υπολοίπων, και εκτελείται σαν να ήταν η μοναδική.
- **Ανθεκτικότητα (Durability):** Μετά την ολοκλήρωση (commit) μιας συναλλαγής, οι αλλαγές που πραγματοποιεί στη βάση δεδομένων διατηρούνται, ακόμα κι αν το σύστημα παρουσιάσει πρόβλημα.

Οι παραπάνω ιδιότητες συνήθως ονομάζονται **ιδιότητες ACID**,από το ακρωνύμιο των τεσσάρων όρων στα αγγλικά. Η εξασφάλιση της ιδιότητας της συνέπειας, όταν αναφερόμαστε σε μεμονωμένες

συναλλαγές, αποτελεί ευθύνη του προγραμματιστή που κωδικοποιεί τη συναλλαγή. Αντίθετα, η διατήρηση των ιδιοτήτων της απομόνωσης, της ανθεκτικότητας και της ατομικότητας, αλλά και της συνέπειας στην περίπτωση ταυτόχρονων συναλλαγών, αποτελούν ευθύνη ενός δομικού συστατικού των συστημάτων διαχείρισης βάσεων δεδομένων, που ονομάζεται **διαχειριστής συναλλαγών (transaction manager).** Ο διαχειριστής συναλλαγών με τη σειρά του έχει δυο λειτουργικές συνιστώσες: Τον **διαχειριστή αποκατάστασης (recovery manager)** και τον **διαχειριστή ελέγχου συγχρονισμού.** Ο διαχειριστής αποκατάστασης είναι υπεύθυνος για τη διατήρηση των ιδιοτήτων ACID σε περίπτωση απροσδόκητων διακοπών της λειτουργίας του συστήματος διαχείρισης βάσεων δεδομένων. Ο διαχειριστής ελέγχου συγχρονισμού είναι υπεύθυνος για τη διατήρηση της συνέπειας στην περίπτωση που εκτελούνται πολλές συναλλαγές ταυτόχρονα.

Βιβλιογραφία – Παραπομπές

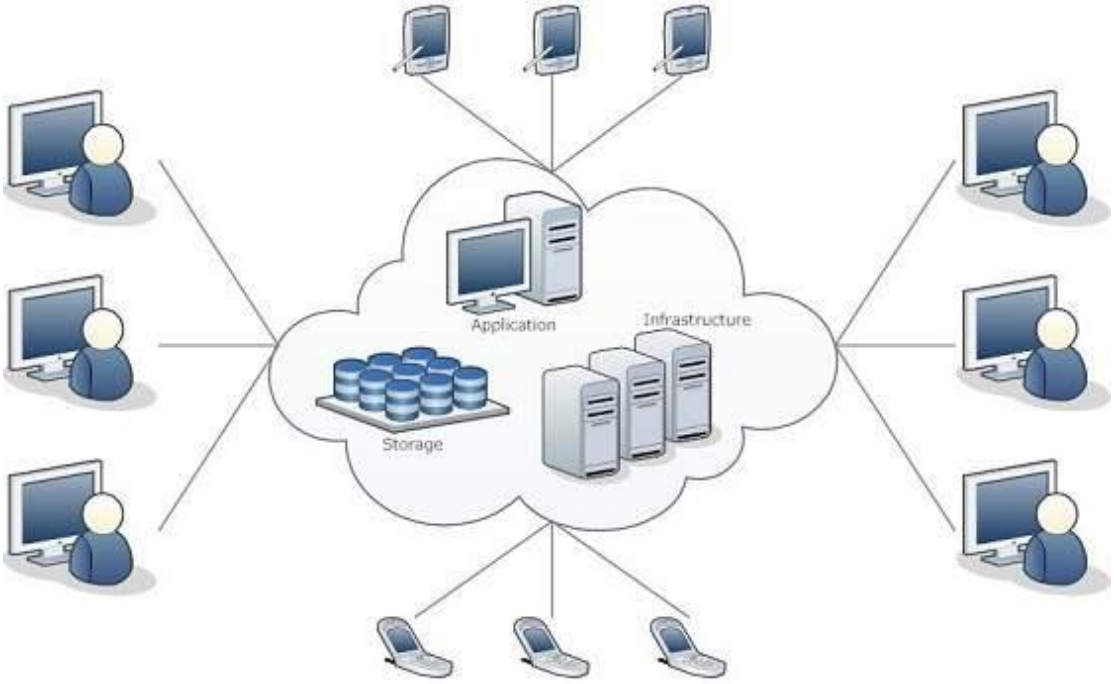
- Database System Concepts (6th ed) - Silberschatz, Korth, Sudarshan
- Database Systems-The Complete Book (2nd ed) – Garcia-Molina,Ullman, Widom
- Database Management Systems (3rd ed) – Ramakrishnan, Gehrke
- MySQL 5.6 Reference Manual (http://dev.mysql.com/doc/refman/5.6/en/)
- SQL Tutorial του εκπαιδευτικού ιστοτόπου TutorialsPoint (http://www.tutorialspoint.com/sql/index.htm)

ΚΕΦΑΛΑΙΟ 2: CLOUD COMPUTING ΚΑΙ ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

Εισαγωγή

Το **cloud computing** είναι μια νέα τεχνολογία που αναμφίβολα οδηγεί τις εξελίξεις στην επιστήμη των υπολογιστών και την αγορά των νέων τεχνολογιών. Πρόκειται για μια τεχνολογία με τεράστιο εύρος εφαρμογών, που έχει αλλάξει ραγδαία τον τρόπο με τον οποίο οι άνθρωποι χρησιμοποιούν δημοφιλείς εφαρμογές της επιστήμης των υπολογιστών. Κλασσικές αρχιτεκτονικές υπολογιστικών συστημάτων, μέρα με τη μέρα αντικαθιστώνται από νέες, που βασίζονται στο cloud computing. Ταυτόχρονα, οικονομικοί κολοσσοί που δραστηριοποιούνται στον τομέα της τεχνολογίας, προσαρμόζουν τα προϊόντα και τις υπηρεσίες τους, ώστε να αξιοποιήσουν τις ολοένα αυξανόμενες δυνατότητες που παρέχει το cloud computing.

Κεντρική ιδέα του cloud computing είναι να **παρέχει στους χρήστες του τη δυνατότητα να χρησιμοποιούν hardware και software που βρίσκεται σε απομακρυσμένες γεωγραφικές τοποθεσίες**, αξιοποιώντας υπάρχουσες δικτυακές υποδομές, όπως του δημόσιου διαδικτύου, αλλά και ιδιωτικών δικτύων WAN, LAN και VPN. Μέσω των παραπάνω δικτυακών υποδομών το cloud computing παρέχει τη δυνατότητα πρόσβασης σε απομακρυσμένα δεδομένα και εφαρμογές. Πολλές δημοφιλείς εφαρμογές, όπως το ηλεκτρονικό ταχυδρομείο, τα κέντρα αποθήκευσης δεδομένων και ο διαμοιρασμός αρχείων, εκτελούνται σε αρχιτεκτονικές cloud computing. Βασικό κέρδος από την επιλογή αυτή, αποτελεί το γεγονός ότι μια εφαρμογή που φιλοξενείται σε μια αρχιτεκτονική cloud είναι ανεξάρτητη του υλικού που χρησιμοποιεί ο χρήστης (**platform independent**), αφού δεν απαιτείται τοπική εγκατάσταση.



Ορισμένα πλεονεκτήματα της χρήσης της τεχνολογίας cloud είναι τα ακόλουθα:

- Η πρόσβαση στους υπολογιστικούς πόρους του cloud δεν απαιτεί οποιουδήποτε είδους διεπαφή με τον πάροχο υπηρεσιών cloud και μπορεί να πραγματοποιηθεί ανά πάσα στιγμή. Οι υπηρεσίες cloud είναι on demand self-service.
- Δίνεται η δυνατότητα διαμοιρασμού υπολογιστικών πόρων από μεγάλο πλήθος χρηστών (resources sharing)
- Δεν απαιτείται η εγκατάσταση λογισμικού τοπικά στα συστήματα των τελικών χρηστών
- Δίνεται η δυνατότητα χρήσης online εργαλείων ανάπτυξης λογισμικού
- Οι υπολογιστικοί πόροι παρέχονται αξιοποιώντας συμβατικές τεχνολογίες δικτύωσης, κάτι που καθιστά την δυνατότητα χρήσης τους ανεξάρτητη του συστήματος του τελικού χρήστη (platform independence)
- Το cloud computing παρέχει μηχανισμούς εξισορρόπησης του υπολογιστικού φορτίου (load balancing), μεγιστοποίησης της αξιοποίησης των υπολογιστικών πόρων (utilization maximization) και βελτίωσης της σχέσης κόστους-αποτελεσματικότητας (cost-effectiveness ratio) που το καθιστούν ιδιαίτερα αξιόπιστο και αποτελεσματικό.

Τα παραπάνω οφέλη της χρήσης cloud computing είναι αναμφισβήτητα, ωστόσο όπως κάθε μεγάλη τεχνολογική εξέλιξη, η χρήση του cloud computing μπορεί να επιφέρει και κάποια προβλήματα. Ορισμένοι κίνδυνοι της χρήσης του cloud computing είναι οι εξής:

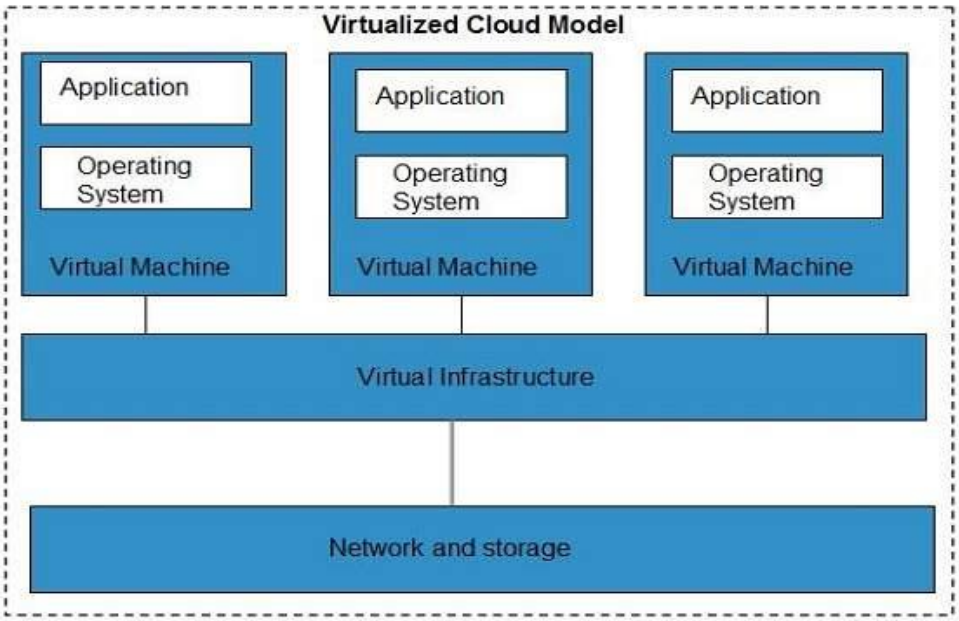
- **Ασφάλεια και Ιδιωτικότητα**: Πρόκειται για το πιο σκοτεινό σημείο αναφορικά με το cloud computing. Η διαχείριση δεδομένων και υποδομών επαφίεται σε κάποιο τρίτο μέρος (third party), που σημαίνει ότι οι χρήστες των υπηρεσιών cloud πρέπει να συμβιβαστούν με την ιδέα ότι τα δεδομένα τους, μεταξύ των οποίων και ευαίσθητα προσωπικά δεδομένα, βρίσκονται στην ευχέρεια του παρόχου υπηρεσιών cloud. Επιπλέον, τα δεδομένα των χρηστών είναι εκτεθειμένα σε διαφόρων τύπων ηλεκτρονικές επιθέσεις κακόβουλων χρηστών του cloud.Εκτός του κινδύνου για τους χρήστες, ελλοχεύει μεγάλος κίνδυνος και για τις εταιρίες παροχής υπηρεσιών cloud, αφού κάθε ένδειξη παραβίασης των μηχανισμών ασφαλείας τους μπορεί να έχει καταστροφικά αποτελέσματα για την εικόνα και την αξιοπιστία τους.
- **Δυσκολία αλλαγής παρόχου υπηρεσιών cloud**: Η μετάβαση από έναν πάροχο υπηρεσιών cloud σε κάποιον άλλο δεν είναι πάντα εύκολη υπόθεση. Μάλιστα σε συγκεκριμένες εφαρμογές είναι εντελώς ανέφικτη. Αυτό δημιουργεί μια ανεπιθύμητη σχέση εξάρτησης μεταξύ χρηστών και παρόχων υπηρεσιών cloud computing, που μπορεί να καταλήξει να είναι προβληματική, ελλείψει εναλλακτικών επιλογών από τη μεριά του χρήστη.
- **Απομόνωση**: Όπως κάθε τεχνολογία που στηρίζεται σε δικτυακές υποδομές, έτσι και το cloud computing κρύβει τον κίνδυνο της απομόνωσης μέρους των διαθέσιμων υπολογιστικών πόρων από τους χρήστες, σε περιπτώσεις προβλημάτων δικτύωσης. Αν για παράδειγμα ένας storage server μείνει offline λόγω προβλήματος στο δίκτυο που τον

συνδέει στο cloud, τα αρχεία που περιέχει δεν είναι προσβάσιμα από τους χρήστες του cloud για όσο χρόνο το πρόβλημα παραμένει.

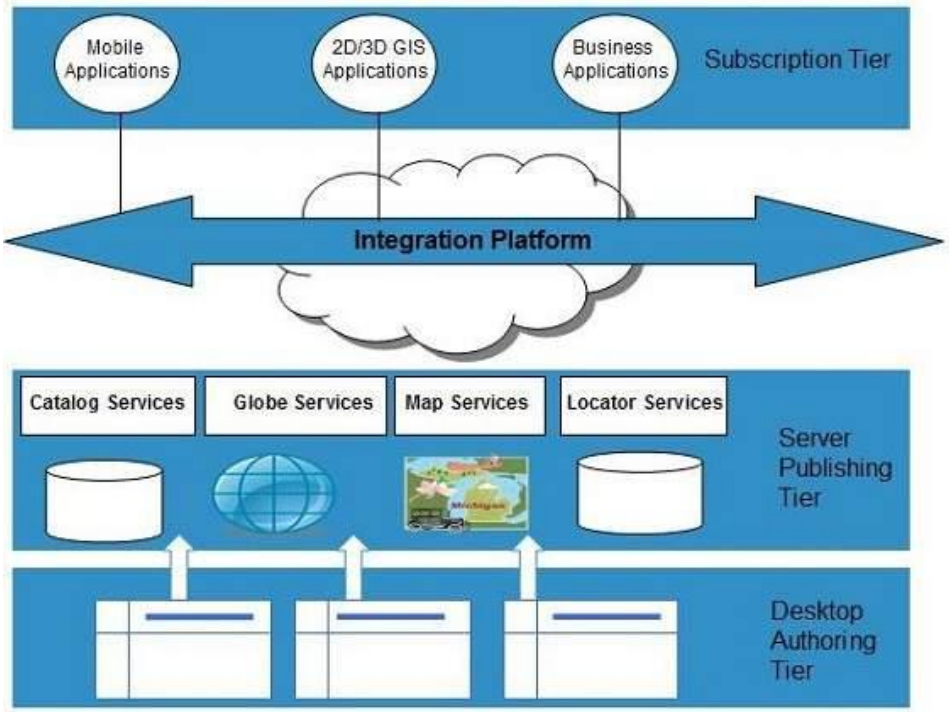
- **Αδυναμία Διαγραφής Δεδομένων**: Σε μια αρχιτεκτονική cloud είναι συνήθης πρακτική τα δεδομένα να αποθηκεύονται σε περισσότερες από μια τοποθεσίες, κάτι που καθιστά δύσκολη ή και αδύνατη την οριστική διαγραφή τους. Αυτό φυσικά αποτελεί μεγάλο πρόβλημα, όταν πρόκειται για ευαίσθητα προσωπικά δεδομένα, που κάποιος χρήστης δεν θα ήθελα να υπάρχουν ες αεί σε κάποιον δημόσια προσβάσιμο χώρο αποθήκευσης.

Το cloud computing βασίζεται σε μεγάλο βαθμό σε μια σειρά από άλλες τεχνολογίες, τις οποίες χρησιμοποιεί στο παρασκήνιο. Μερικές από αυτές τις τεχνολογίες είναι οι εξής:

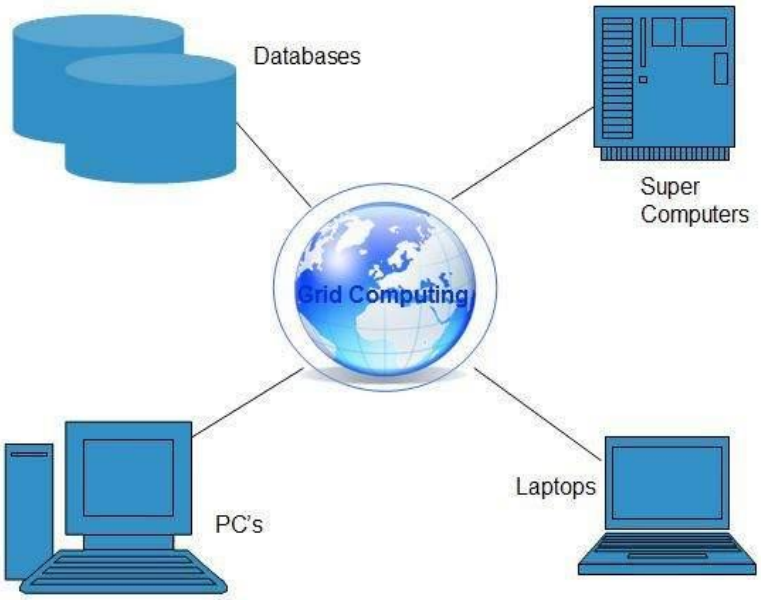
- **Virtualization:** Με την τεχνολογία αυτή επιτυγχάνεται ο αποδοτικός διαμοιρασμός υπολογιστικών πόρων μεταξύ πολλών χρηστών. Αποδίδει λογικές ονομασίες στους φυσικούς υπολογιστικούς πόρους και παραχωρεί στα προγράμματα χρήστη που θέλουν να τους χρησιμοποιήσουν δείκτες που αναφέρονται σε αυτούς. Με αυτόν τον τρόπο τα προγράμματα χρηστών μπορούν να αποκτούν πρόσβαση στους υπολογιστικούς πόρους αυτόνομα, απολαμβάνοντας έτσι ένα καθεστώς εικονικής αποκλειστικότητας, ως προς τη χρήση των υπολογιστικών πόρων του cloud.



- **Service-Oriented Architecture (SOA):** Με την αρχιτεκτονική αυτή, δίνεται έμφαση στην υπηρεσία (service) που παρέχει μια εφαρμογή (application) και διασφαλίζεται ότι κάθε εφαρμογή που φιλοξενείται στο cloud θα μπορεί να ανταλλάσει δεδομένα με εφαρμογές-πελάτες (client applications) κάθε τύπου και κάθε προέλευσης.



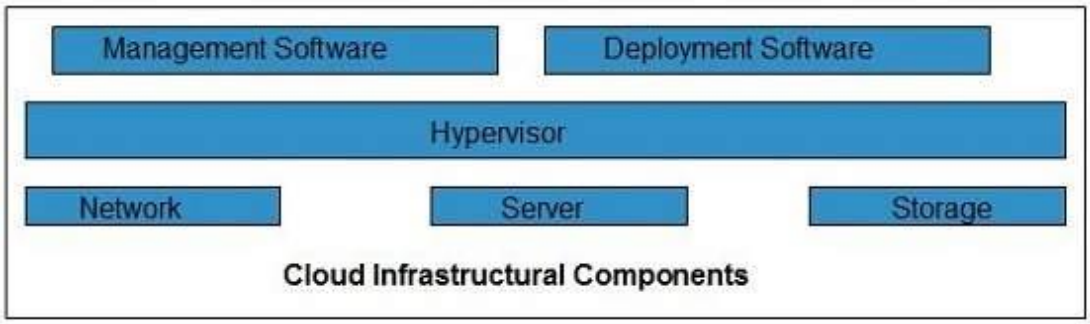
- **Grid (Distributed) Computing:** Σε αυτή την αρχιτεκτονική, υπολογιστές που βρίσκονται σε απομακρυσμένες γεωγραφικά τοποθεσίες μεταξύ τους, διασυνδέονται μέσω δικτυακών υποδομών και λειτουργούν από κοινού για να διεκπεραιώσουν από κοινού υπολογιστικές εργασίες. Στα του grid computing οι υπολογιστικές εργασίες διασπώνται σε μικρότερες επιμέρους εργασίες που διαμοιράζονται στους επεξεργαστικούς πυρήνες του πλέγματος. Με τον τρόπο αυτό το υπολογιστικό φορτίο διαμοιράζεται, γίνεται βέλτιστη χρήση της συνολικής υπολογιστικής ισχύος και η απόδοση του συνολικού συστήματος βελτιώνεται.



Μια τυπική αρχιτεκτονική cloud computing περιλαμβάνει πολυάριθμες λειτουργικές συνιστώσες, οι οποίες εντάσσονται σε δυο ευρείες κατηγορίες:

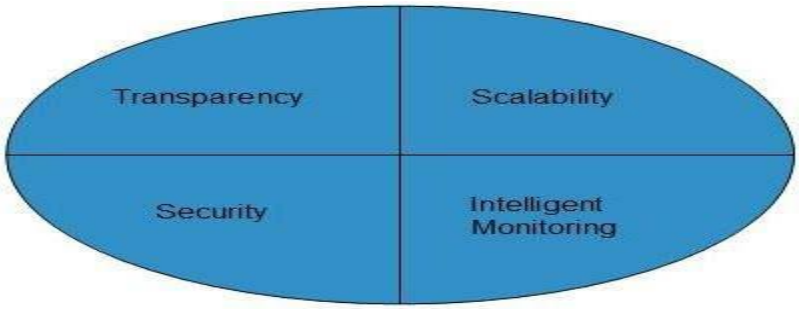
- **Front End**: Το front end αναφέρεται στο κομμάτι του cloud computing που περιλαμβάνει τις εφαρμογές πελάτη με τις οποίες ο χρήστης αλληλεπιδρά με το cloud. Τυπικό παράδειγμα τέτοιας εφαρμογής αποτελούν οι σύγχρονοι web browsers και οι ιστοσελίδας που παρέχουν πρόσβαση στις υπηρεσίες του cloud. Ουσιαστικά το front end αποτελείται από όλες της πύλες πρόσβασης των χρηστών στις υπηρεσίες του cloud.
- **Back End**: Το back end αναφέρεται στο ίδιο το cloud και περιλαμβάνει όλους τους διαθέσιμους υπολογιστικούς πόρους (virtual machine, data storage, servers, applications κτλ). Το back end είναι επίσης υπεύθυνο για την παροχή μηχανισμών ασφαλείας, μηχανισμών ελέγχου δικτυακής συμφόρησης, καθώς και πρωτοκόλλων και middleware επικοινωνίας για την διασύνδεση συσκευών διαφορετικών κατασκευαστών και τύπων.

Η υποδομή μιας τυπικής αρχιτεκτονικής cloud computing αποτελείται από τα εξής δομικά συστατικά: **servers, συσκευές αποθήκευσης δεδομένων, δικτυακή υποδομή, λογισμικό διαχείρισης cloud , λογισμικό deployment και λογισμικό virtualization.**



Μια αρχιτεκτονική cloud computing πρέπει να διασφαλίζει τα ακόλουθα σημαντικά χαρακτηριστικά:

- **Διαφάνεια** ως προς τους διαθέσιμους πόρους
- **Δυνατότητα Κλιμάκωσης** των διατιθέμενων λειτουργιών
- **Ευφυής Επίβλεψη** της ροής των δεδομένων και του υπολογιστικού φόρτου
- **Ασφάλεια** της ιδιωτικότητας και των δεδομένων των χρηστών



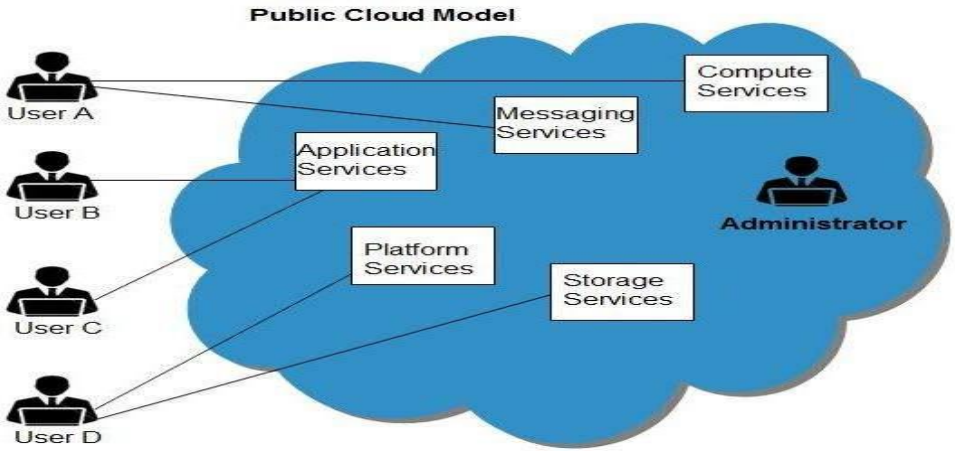
Αρχιτεκτονικά Μοντέλα και Τεχνολογίες Cloud Computing

Τα βασικά μοντέλα αρχιτεκτονικής cloud computing είναι τα εξής:

1. Μοντέλο Δημοσίου Cloud (Public Cloud Model)
2. Μοντέλο Ιδιωτικού Cloud (Private Cloud Model)
3. Υβριδικό μοντέλο Cloud (Hybrid Cloud Model)
4. Μοντέλο cloud κοινότητας (Community Cloud Model)

Public Cloud Model

Σε αυτό το μοντέλο, οι υπολογιστικοί πόροι και οι υπηρεσίες του cloud είναι προσβάσιμα για κάθε ενδιαφερόμενο χρήστη μέσω διαδικτύου. Παραδείγματα δημοσίων cloud αποτελούν τα clouds τεχνολογικών κολοσσών όπως η Amazon, η Microsoft και η Google.



Πλεονεκτήματα:

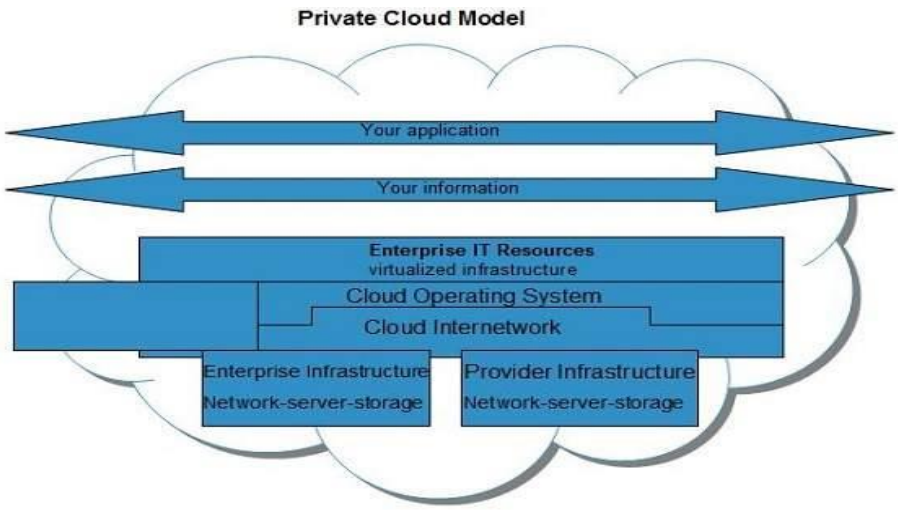
- Ανταγωνιστικό κόστος , λόγω του υψηλού βαθμού διαμοιρασμού των διαθέσιμων πόρων
- Αξιοπιστία, λόγω της παροχής πολλών εναλλακτικών επιλογών για κάθε υπηρεσία ή υπολογιστικό πόρο
- Ευελιξία, καθώς εύκολα μπορεί να ενσωματωθεί σε μια άλλη αρχιτεκτονική cloud
- Γεωγραφική ανεξαρτησία, αφού οι υπολογιστικοί πόροι και οι υπηρεσίες προσπελαύνονται μέσω διαδικτύου.
- Μεγάλη δυνατότητα κλιμάκωσης, λόγω των δυνατοτήτων που παρέχει το διαδίκτυο.

Μειονεκτήματα:

- Κίνδυνοι ασφαλείας πληροφοριών λόγω της έκθεσης σε διαδικτυακές ηλεκτρονικές επιθέσεις.
- Μικρή προσαρμοστικότητα στις συγκεκριμένες ανάγκες κάθε χρήστη, προς χάριν της μαζικής εξυπηρέτησης.

Private Cloud Model

Σε αυτό το μοντέλο, οι υπολογιστικοί πόροι και οι υπηρεσίες του cloud είναι προσβάσιμα μόνο σε χρήστες που δραστηριοποιούνται στα πλαίσια του οργανισμό στον οποίο ανήκει το cloud. Ο οργανισμός αυτός έχει υπό τον πλήρη έλεγχο του τη λειτουργία του cloud και των λειτουργιών που αυτό παρέχει στους χρήστες του.



Πλεονεκτήματα:

- Διατήρηση της ιδιωτικότητας και της ασφάλειας των δεδομένων, λόγω της απουσίας πρόσβασης ξένων προς τον οργανισμό προσώπων.
- Καλύτερος έλεγχος της διαχείρισης των υπολογιστικών πόρων.
- Μεγαλύτερη προσβασιμότητα στους διαθέσιμους πόρους, λόγω του περιορισμένου πλήθους χρηστών.
- Χαμηλότερο κόστος συντήρησης και διαχείρισης των διαθέσιμων πόρων

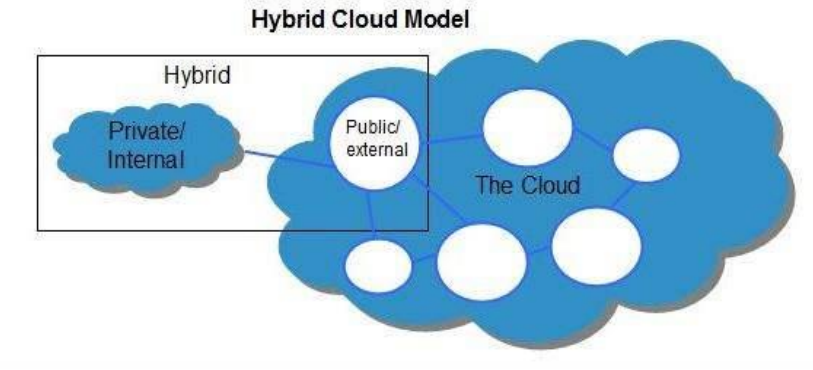
Μειονεκτήματα:

- Υψηλό κόστος κατά την αρχική προσαρμογή των υπολογιστικών συστημάτων ενός οργανισμού στα δεδομένα του cloud computing.
- Περιορισμένη περιοχή δραστηριότητας, αφού η πρόσβαση δεν γίνεται μέσω του δημόσιου διαδικτύου, αλλά είτε μέσω ιδιωτικών δικτύων του οργανισμού είτε μέσω ειδικού λογισμικού.
- Περιορισμένη δυνατότητα κλιμάκωσης των μεγεθών.
- Απαίτηση για εξειδικευμένο προσωπικό.

Hybrid Cloud Model

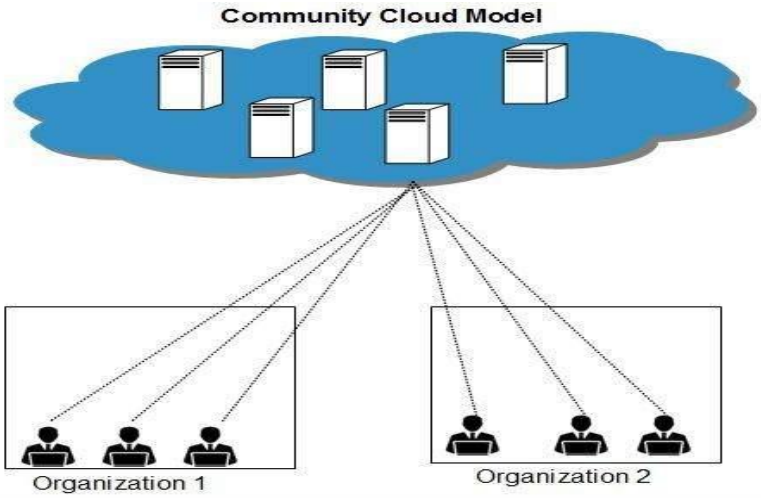
Το μοντέλο αυτό έχει διάφορες μορφές. Σε κάθε περίπτωση αποτελεί έναν συνδυασμό public και private cloud, εφαρμόζοντας πολιτικές που προσεγγίζουν περισσότερο αυτές του public cloud, όταν πρόκειται για απλές υπηρεσίες, και πολιτικές που προσεγγίζουν περισσότερο αυτές του private cloud,

όταν πρόκειται για κρίσιμες υπηρεσίες, που εμπλέκουν ευαίσθητα δεδομένα (κωδικούς πρόσβασης, ιδιωτικές συνομιλίες, αριθμούς πιστωτικών καρτών κτλ)



Community Cloud Model

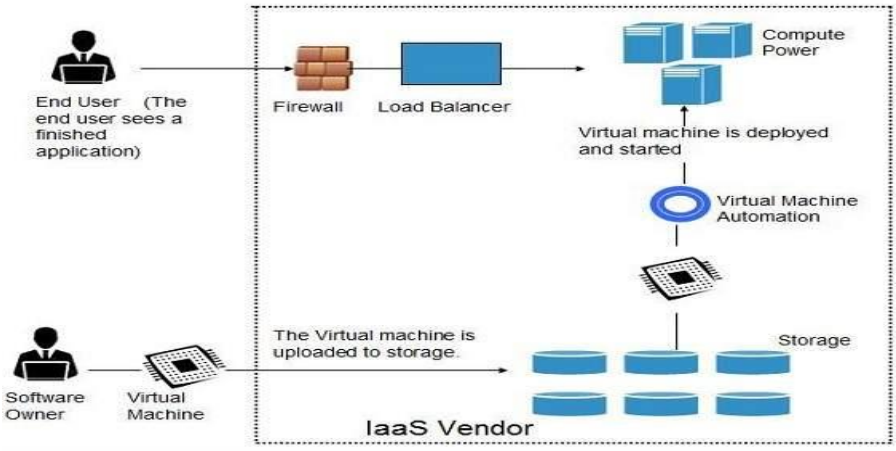
Το μοντέλο αυτό αποτελεί επέκταση του private cloud. Στη περίπτωση αυτή, το cloud Βρίσκεται υπό τον έλεγχο μιας κοινότητας οργανισμών, οι οποίοι παρέχουν δικαιώματα πρόσβασης στους πόρους του cloud σύμφωνα με τα συμφέροντα και τις επιδιώξεις τους. Το community cloud χρησιμοποιεί τις υποδομές όλων των οργανισμών που συναποτελούν την κοινότητα. Σε πολλές περιπτώσεις μάλιστα, η δημιουργία μιας ισχυρής πλατφόρμας cloud computing είναι και η αρχική αιτία του σχηματισμού μιας τέτοιας κοινότητας οργανισμών με κοινά συμφέροντα.



Υποδομή ως Υπηρεσία (Infrastructure as a Service - IaaS)

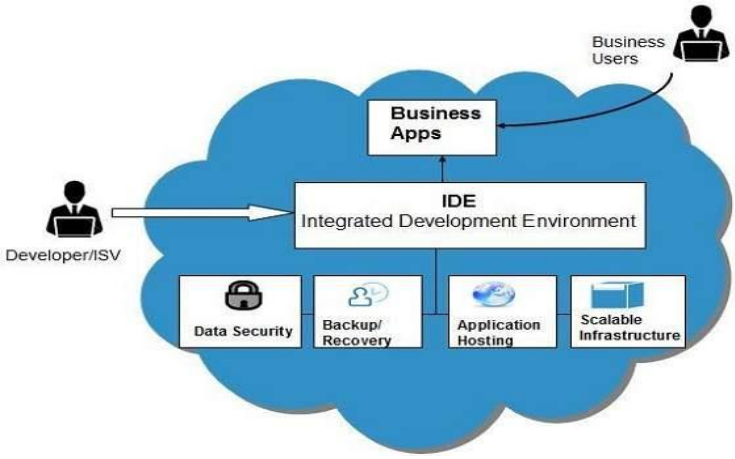
Το cloud computing, μέσω της τεχνολογίας IaaS, παρέχει στους χρήστες πρόσβαση σε θεμελιώδεις υπολογιστικούς πόρους, όπως φυσικές μηχανές, εικονικές μηχανές και αποθήκες δεδομένων. Το μοντέλο IaaS παρέχει επίσης χώρο αποθήκευσης εικονικών μηχανών, εικονικά τοπικά δίκτυα (VLANs), διευθύνσεις IP και πακέτα λογισμικού.

Η παροχή όλων των παραπάνω πόρων στηρίζεται στην τεχνολογία εικονικού εξυπηρετητή (server virtualization). Μέσω αυτής της τεχνολογίας, οι χρήστες χρησιμοποιούν τους διαθέσιμους υπολογιστικούς πόρους σαν να τους έχουν κατά αποκλειστικότητα, χωρίς να αντιλαμβάνονται ότι στην πραγματικότητα τους μοιράζονται με όλους τους υπόλοιπους χρήστες του cloud.



Πλατφόρμα ως Υπηρεσία (Platform as a Service - PaaS)

Το cloud computing, μέσω της τεχνολογίας PaaS, παρέχει το κατάλληλο περιβάλλον χρόνου εκτέλεσης (runtime environment) για τις εφαρμογές που φιλοξενεί, καθώς εργαλεία ανάπτυξης λογισμικού εφαρμογών, μεταξύ των οποίων και απλά point and click εργαλεία που επιτρέπουν σε χρήστες χωρίς ειδικές γνώσεις να δημιουργήσουν απλές web εφαρμογές. Χαρακτηριστικό παράδειγμα αποτελεί το cloud της Google και το ιδιαίτερα διαδεδομένο Google API.



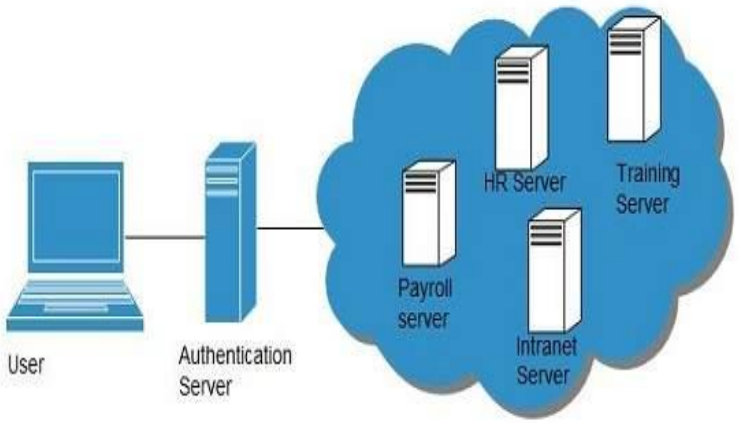
Λογισμικό ως Υπηρεσία (Software as a Service - SaaS)

Το cloud computing, μέσω της τεχνολογίας SaaS, παρέχει στους χρήστες πρόσβαση σε εφαρμογές, τις οποίες μπορούν να χρησιμοποιήσουν απομακρυσμένα, χωρίς τοπική εγκατάσταση στο σύστημα τους. Πρόκειται αναμφισβήτητα για έναν από τους βασικούς λόγους που η τεχνολογία του cloud computing έχει γνωρίσει τόσο μεγάλη επιτυχία και έχει αποδειχτεί τόσο επιδραστική.

Ταυτότητα ως Υπηρεσία (Identity as a Service - IaaS)

Το cloud computing, μέσω της τεχνολογίας IaaS, παρέχει στους χρήστες τη δυνατότητα να διαχειρίζονται με εύκολο και ενιαίο τρόπο, όλες εκείνες τις πληροφορίες που τους ταυτοποιούν ως ψηφιακές οντότητες. Γενικότερα, μέσω της τεχνολογίας αυτής ορίζονται οι απαραίτητοι μηχανισμοί για την ταυτοποίηση των επικοινωνούντων οντοτήτων του cloud (χρηστών, οργανισμών, παρόχων υπηρεσιών κτλ)

Ειδικά για την περίπτωση της ταυτοποίησης χρηστών, ειδική αναφορά αξίζει να γίνει στην τεχνολογία **Single Sign On (SSO),** που επιτρέπει στους χρήστες να έχουν πρόσβαση στο σύνολο των συστημάτων που συνθέτουν μια αρχιτεκτονική cloud computing, αν αυτός ταυτοποιηθεί από έναν εξυπηρετητή ταυτοποίησης.



Βάσεις Δεδομένων Cloud

Μια βάση δεδομένων cloud βρίσκεται αποθηκευμένη σε αποθηκευτικούς χώρους ενός cloud και είναι προσβάσιμη μέσω αυτού. Η τεχνολογία βάσεων δεδομένων cloud είναι γνωστή και ως **Database as a Service (DaaS).** Μια βάση δεδομένων cloud μπορεί να είναι μια συμβατική βάση δεδομένων, ωστόσο αξιοποιώντας τις υποδομές του cloud Που τη φιλοξενεί μπορεί να παρουσιάζει αυξημένη προσβασιμότητα και απόδοση. Παρόλα αυτά, οι βάσεις δεδομένων cloud βασίζονται στα ίδια θεωρητικά εργαλεία (αλγορίθμους, δομές δεδομένων, πρωτόκολλα, προδιαγραφές) με τις συμβατικές βάσεις δεδομένων και η μελέτη τους μπορεί να γίνεται παράλληλα, αφού σε γενικές γραμμές, τα όποια συμπεράσματα αφορούν και τις δυο περιπτώσεις.

Μια βάση δεδομένων cloud μπορεί να είναι είτε μια παραδοσιακή βάση δεδομένων (πχ MySQL, Microsoft SQL Server κτλ) είτε μια βάση δεδομένων ειδικά σχεδιασμένη για το cloud computing (Xeround, Azure κτλ).

Μια βάση δεδομένων cloud παρουσιάζει κάποια σαφή πλεονεκτήματα έναντι μιας συμβατικής βάσης δεδομένων, όπως αυξημένη προσβασιμότητα, αποδοτική και αυτοματοποιημένη ανάνηψη από αστοχίες, μεγάλη δυνατότητα κλιμάκωσης μεγέθους με αξιοποίηση των υπολογιστικών πόρων του cloud και πιο αποκεντρωμένη διαχείριση.

Ταυτόχρονα, μια βάση δεδομένων cloud παρουσιάζει και μειονεκτήματα, όπως η έκθεση σε επιθέσεις κακόβουλων χρηστών, η διαχείριση των δεδομένων από τρίτα μέρη και ο αποκλεισμός από την πρόσβαση στα δεδομένα της σε χρονικά διαστήματα που το cloud είναι εκτός λειτουργίας.

Βιβλιογραφία – Παραπομπές

- Cloud Computing - Software Technology and Architectures – Erl, Mahmood, Puttini
- *Architecting the Cloud: Design Decisions for Cloud Computing Service Models – Kavis*
- *Cloud Computing Bible - Sosinsky*
- *Cloud Computing Tutorial , του ιστοτόπου TutorialsPoint (http://www.tutorialspoint.com/cloud_computing/index.htm)*
  Όλες οι εικόνες του παρόντος κεφαλαίου αποτελούν πνευματική ιδιοκτησία του παραπάνω ιστοτόπου.

ΚΕΦΑΛΑΙΟ 3: TPCDS BENCHMARK

ΕΙΣΑΓΩΓΗ

Ένα benchmark, είναι ένα σύνολο από λειτουργίες και προδιαγραφές, που χρησιμεύουν ως σημείο αναφοράς και σύγκρισης, για την εκτίμηση της επίδοσης και της αποτελεσματικότητας πραγματικών υπολογιστικών συστημάτων. Το **TPC Benchmark DS (TPC-DS)** είναι ένα **benchmark** εκτίμησης της απόδοσης **συστημάτων υποστήριξης αποφάσεων (decision support systems)**. Συνεπώς, αποτελεί ένα πολύ ισχυρό εργαλείο για επιχειρήσεις που επιθυμούν να αξιοποιήσουν λειτουργικά δεδομένα (operational data), ώστε να τα μετατρέψουν σε επιχειρηματική πληροφόρηση (business intelligence), που φυσικά θα αξιοποιηθεί στη συνέχεια για τον βραχυπρόθεσμο και μακροπρόθεσμο επιχειρηματικό τους προγραμματισμό.

Το TPC-DS μοντελοποιεί ένα σύνολο από έννοιες που τυγχάνουν ευρείας εφαρμογής στον τομέα της υποστήριξης αποφάσεων, όπως τα ερωτήματα σε βάσεις δεδομένων (database queries) και η διατήρηση δεδομένων (data maintenance). Σκοπός του TPC-DS είναι να απεικονίσει συστήματα υποστήριξης αποφάσεων που μεταξύ άλλων:

- Εξετάζουν μεγάλους όγκους δεδομένων
- Δίνουν απαντήσεις σε πραγματικά προβλήματα των σύγχρονων επιχειρήσεων
- Εκτελούν ερωτήματα διαφόρων λειτουργικών προδιαγραφών και επιπέδων πολυπλοκότητας (ad-hoc, reporting, iterative, OLAP, data mining)
- Χαρακτηρίζονται από υψηλό φόρτο CPU και IO.
- Συγχρονίζονται περιοδικά με OLTP βάσεις δεδομένων μέσω λειτουργιών συντήρησης βάσεων δεδομένων.

Γενικά, το benchmark μετρά τον ρυθμό διεκπεραίωσης ερωτημάτων (query throughput) και την αποδοτικότητα της συντήρησης δεδομένων για ένα δεδομένο συνδυασμό υλικού, λειτουργικού συστήματος και συνόλου ρυθμίσεων του συστήματος διαχείρισης βάσεων δεδομένων (DBMS), υπό την επίδραση ενός ελεγχόμενου και σύνθετου υπολογιστικού φορτίου υποστήριξης αποφάσεων προερχόμενου από πολλούς χρήστες.

Το TPC-DS μοντελοποιεί τα δυο βασικότερα συστατικά κάθε συστήματος υποστήριξης αποφάσεων:

- **Ερωτήματα χρηστών**, που μετατρέπουν λειτουργικά γεγονότα σε επιχειρησιακή πληροφόρηση (business intelligence)
- **Συντήρηση δεδομένων**, που δίνει τη δυνατότητα ευρύτερης αξιοποίησης τους

Ο παραπάνω συνδυασμός καθορίζει την εν γένει αποτελεσματικότητα ενός συστήματος υποστήριξης αποφάσεων και το TPC-DS είναι ειδικά σχεδιασμένο ώστε να αποδίδει με ακρίβεια κατά πόσο ένα σύστημα ανταποκρίνεται αποδοτικά στις δυο του συνιστώσες.

Γενικές Οδηγίες Υλοποίησης

Σκοπός του TPC-DS είναι να παρέχει στις επιχειρήσεις που θα το χρησιμοποιήσουν αντικειμενικά δεδομένα απόδοσης. Για να επιτευχθεί αυτό, απαιτείται το benchmark Να υλοποιηθεί χρησιμοποιώντας συστήματα, προϊόντα και τεχνολογίες που:

- Είναι γενικά διαθέσιμα για τους χρήστες.
- Είναι κατάλληλα για το τμήμα της αγοράς με το οποίο καταπιάνεται συγκεκριμένα το TPC-DS (μεταξύ των άλλων TPC benchmarks), δηλαδή τα περιβάλλοντα υποστήριξης αποφάσεων, που χαρακτηρίζονται από μεγάλους όγκους δεδομένων και υψηλή πολυπλοκότητα.
- Θα μπορούσαν να χρησιμοποιηθούν, για την υλοποίηση του benchmark, από έναν ικανό αριθμό χρηστών.

Λαμβάνοντας υπόψη τα παραπάνω, η βάση δεδομένων του TPC-DS πρέπει να υλοποιείται με εμπορικά διαθέσιμο λογισμικό για συστήματα διαχείρισης βάσεων δεδομένων και τα ερωτήματα του να εκτελούνται μέσω μιας διεπαφής SQL. Γενικά, συνίσταται η χρήση ‘real world’ εργαλείων για την υλοποίηση του benchmark ούτως ώστε να είναι όσο γίνεται πιο αξιόπιστα τα αποτελέσματα. Συνεπώς, απαγορεύεται η χρήση εργαλείων που είναι ειδικά σχεδιασμένα για να βελτιώνουν τα αποτελέσματα του benchmark καθιστώντας τα πλασματικά.

Γενικές Οδηγίες Μετρήσεων

Τα αποτελέσματα του TPC-DS πρέπει να αποτελούν μια όσο το δυνατό πιο πιστή αναπαράσταση της πραγματικής απόδοσης ενός συστήματος με τα χαρακτηριστικά που μοντελοποιεί. Συνεπώς υπάρχουν κάποιες ελάχιστες προδιαγραφές που οφείλει να πληροί οποιαδήποτε μέθοδος επιλεγεί για την καταγραφή τους. Οι βασικότερες από αυτές τις προδιαγραφές είναι:

- Να αποτελεί μια αποδεκτή πρακτική ή standard.
- Να μην βελτιώνει καθόλου τα αποτελέσματα.
- Τα εργαλεία που χρησιμοποιούνται, τόσο σε υλικό όσο και σε λογισμικό, να εντάσσονται σε καθιερωμένα ποιοτικά επίπεδα.
- Να διαφυλάττει, την διακριτικότητα σε περίπτωση καταγραφής λειτουργικών ανωμαλιών, ακόμα και αν αυτό δεν περιλαμβάνεται στις απαιτήσεις του benchmark.

Η χρήση πολλών και ίσως καινοτόμων μεθόδων καταγραφής των αποτελεσμάτων του benchmark ενθαρρύνονται ιδιαίτερα, με την προϋπόθεση ότι ικανοποιούν τις παραπάνω προδιαγραφές.

Στο σημείο αυτό, κρίνεται σκόπιμο να γίνουν ορισμένα σχόλια σχετικά με τις μετρήσεις των αποτελεσμάτων του TPC-DS. Αρχικά, δεν συνίσταται να συγκρίνονται τα αποτελέσματα μιας υλοποίησης του TPC-DS με τα αποτελέσματα οποιουδήποτε άλλου που ανήκει στην οικογένεια των TCP benchmarks, παρά το γεγονός ότι χρησιμοποιούν σε μεγάλο βαθμό κοινή ορολογία και μεθόδους. Επιπλέον, πρέπει να τονιστεί ότι παρά το γεγονός ότι το TPC-DS μοντελοποιεί αρκετά χαρακτηριστικά των συστημάτων υποστήριξης αποφάσεων, σε καμία περίπτωση δεν καλύπτει όλα τα επί μέρους χαρακτηριστικά που μπορεί να έχει ένα τέτοιο σύστημα. Αυτό συμβαίνει γιατί ένα benchmark έχει

γενικά την τάση να μοντελοποιεί κατηγορίες συστημάτων και όχι μεμονωμένα συστήματα. Τέλος, όπως σε κάθε περίπτωση υπολογιστικών μετρήσεων, τα αποτελέσματα του TPC-DS εξαρτώνται άμεσα από τα χαρακτηριστικά του συστήματος στο οποίο διεξάγονται οι μετρήσεις, τα οποία θα πρέπει να καταγράφονται και να συνοδεύουν τα αποτελέσματα του TPC-DS, ώστε να μπορούν αυτά να ερμηνευτούν ορθά.

Επιχειρηματικό περιβάλλον TPC-DS

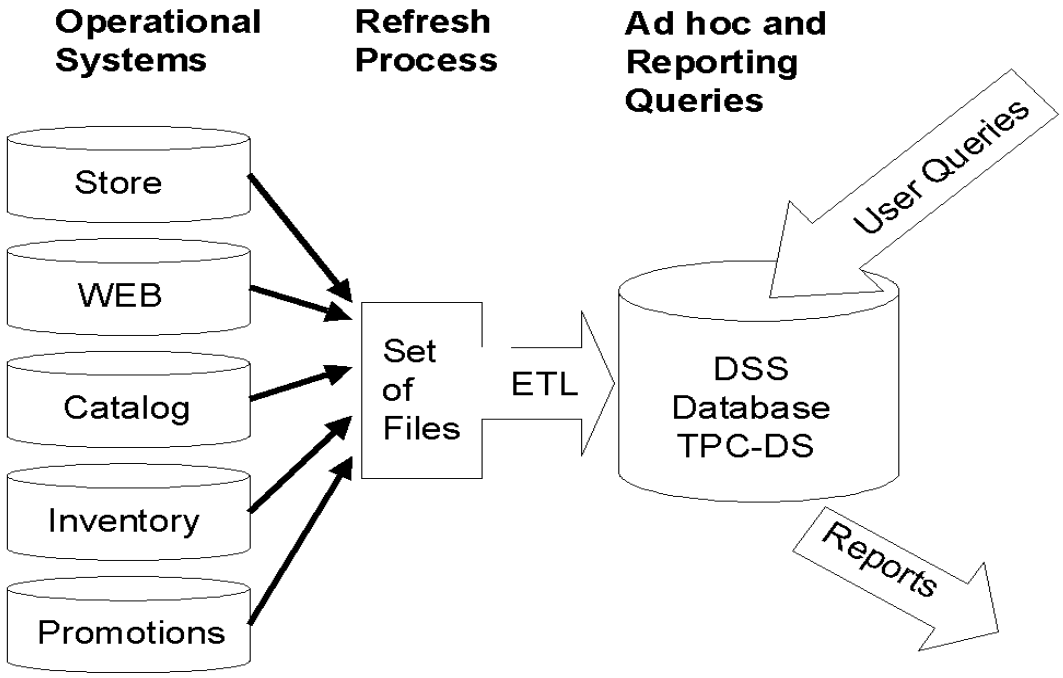
Το TPC-DS έχει χαρακτηριστικά που του επιτρέπουν να αξιολογεί ένα μεγάλο εύρος συστημάτων, διατηρώντας παράλληλα ένα εξίσου μεγάλο εύρος δυνατοτήτων υλοποίησης. Προκειμένου να διευκολυνθεί η διαδικασία εκμάθησης της λειτουργίας του benchmark από τους χρήστες, οι σχεδιαστές του έχουν χρησιμοποιήσει ένα τυπικό επιχειρηματικό περιβάλλον (business environment) που περιλαμβάνει όλα τα τυπικά χαρακτηριστικά ενός συστήματος υποστήριξης αποφάσεων, ως σενάριο εφαρμογής των λειτουργιών του benchmark. Αυτό το επιχειρηματικό περιβάλλον αποτελεί κατά κάποιο τρόπο ένα σημασιολογικό υπόβαθρο (context), μέσα στο οποίο οι διάφορες λειτουργίες του Benchmark αποκτούν νόημα και γίνονται πιο εύκολα κατανοητές, σαν λειτουργίες ενός πραγματικού συστήματος υποστήριξης αποφάσεων.

Το TPC-DS μοντελοποιεί τις λειτουργίες υποστήριξης αποφάσεων μιας υποθετικής επιχείρησης λιανικής διάθεσης προϊόντων, και η βάση δεδομένων του benchmark περιέχει πληροφορίες που αφορούν μια τέτοια επιχείρηση, όπως για παράδειγμα πληροφορίες σχετικά με τους πελάτες, τα προϊόντα και τις αποθήκες της.

Για να γίνει ακόμα πιο ρεαλιστικό το επιχειρηματικό περιβάλλον του TPC-DS, η εμπορική επιχείρηση που χρησιμοποιείται ως μοντέλο θεωρείται ότι δραστηριοποιείται σε τρείς διαφορετικούς τομείς: τις αγορές από το κατάστημα (store sales), τις αγορές καταλόγου (catalog sales) και τις ηλεκτρονικές αγορές (web sales). Συνεπώς το TPC-DS είναι ιδανικό για τη μοντελοποίηση οποιασδήποτε επιχείρησης πρέπει να διαχειριστεί, να πουλήσει και να διανείμει προϊόντα.

Ο σκοπός της επιλογής μιας επιχείρησης λιανικών πωλήσεων ως επιχειρηματικό περιβάλλον του benchmark, είναι τόσο ο χρήστης όσο και ο απλός αναγνώστης των αποτελεσμάτων να έχουν μια άμεση αίσθηση οικειότητας και να μπορούν να κατανοήσουν τις λειτουργίες και τα αποτελέσματα του benchmark μέσα σε ένα εννοιολογικό περιβάλλον που τους είναι οικείο. Φυσικά, αυτό δεν σημαίνει, σε καμία περίπτωση, ότι το TPC-DS δεν μπορεί να χρησιμοποιηθεί από οποιαδήποτε άλλη επιχείρηση αποσκοπεί στη μετατροπή εξωτερικών και λειτουργικών γεγονότων σε επιχειρηματική πληροφόρηση. Μάλιστα, το TCP-DS αφορά επιχειρήσεις που στοχεύουν τόσο στη λήψη βραχυπρόθεσμων αποφάσεων, όσο και στην κατάστρωση μακροπρόθεσμων σχεδίων, πάντα με βάση λειτουργικά δεδομένα που συλλέγουν και επεξεργάζονται.

Τα λειτουργικά συστατικά του TPC-DS, απεικονίζονται στο παρακάτω διάγραμμα.



Η υποθετική επιχείρηση του TPC-DS, έχει όλα τα συνήθη χαρακτηριστικά μιας επιχείρησης λιανικών πωλήσεων, με ένα πλήθος καταστημάτων που εκτείνονται σε εθνικό επίπεδο, ενώ εκτός από το **σύστημα πωλήσεων (sales)** της, περιλαμβάνει ένα **σύστημα προώθησης (promotion)** και ένα σύστημα **απογραφής (inventory)**. Μερικά τυπικά παραδείγματα δραστηριοτήτων που διεκπεραιώνει η επιχείρηση είναι τα εξής:

- Καταγράφει τις αγορές που πραγματοποιούν οι πελάτες της, αλλά και τις επιστροφές που τυχόν προκύπτουν, για όλους τους τομείς πωλήσεων (store, catalog, web).
- Τροποποιεί τις τιμές των προϊόντων της με βάση το σύστημα προώθησης που εφαρμόζει
- Διατηρεί δεδομένα απογραφής για όλες τις αποθήκες της
- Δημιουργεί δυναμικές ιστοσελίδες για την πώληση και την προώθηση των προϊόντων της
- Διατηρεί δεδομένα σχετικά με το καταναλωτικό προφίλ των πελατών της (customer relationship management)

Το TPC-DS δεν μοντελοποιεί όλα τα λειτουργικά μέρη της επιχείρησης. Για παράδειγμα, τα τρία ξεχωριστά κανάλια πώλησης προϊόντων λειτουργούν αυτόνομα και πιθανώς να διατηρούν αυτόνομα δεδομένα, τα οποία δεν λαμβάνονται υπόψη από το benchmark.

Η μοντελοποίηση της επιχείρησης από το TPC-DS συνίσταται στα εξής:

- **Μοντέλο Δεδομένων (Data Model)** και **Πρόσβαση Δεδομένων (Data Access)**
- **Ερωτήματα (Queries)** και **Μοντέλο Χρηστών (User Model)**

- **Συντήρηση Δεδομένων (Data Maintenance)**

Μοντέλο Δεδομένων (Data Model) και Πρόσβαση Δεδομένων (Data Access)

Το TPC-DS μοντελοποιεί μια βάση δεδομένων που είναι ανά πάσα στιγμή διαθέσιμη στους χρήστες, για τροποποιήσεις των περιεχομένων της, αλλά και για διαφόρων τύπων ερωτήματα. Η βάση αυτή μπορεί να προσπελαύνεται και να τροποποιείται ταυτόχρονα από πολλά προγράμματα χρήστη, σε ένα περιβάλλον που επιτρέπει ερωτήματα μεγάλου χρόνου εκτέλεσης, πολλών τμημάτων. Όλα τα ερωτήματα του TPC-DS, καθώς επίσης και οι λειτουργίες συντήρησης δεδομένων που περιλαμβάνει, εφόσον μπορούν να εκτελούνται ταυτόχρονα, οφείλουν να υπόκεινται στις **προδιαγραφές ACID**, δηλαδή να διατηρούν την **ατομικότητα (atomicity)** των συναλλαγών με τη βάση, τη **συνέπεια (consistency)** των δεδομένων, την **απομόνωση (isolation)** των συναλλαγών με τη βάση και την **ανθεκτικότητα (durability)** των δεδομένων. Οι προδιαγραφές ACID πληρούνται από όλες τις κλάσεις ερωτημάτων που περιγράφονται παρακάτω.

Το σχήμα της βάσης δεδομένων του TPC-DS είναι ένα **σχήμα αστέρα (star schema),** αποτελούμενο από **πίνακες γεγονότων (fact tables) και πίνακες διαστάσεων (dimension tables)**. Κάθε πίνακας διαστάσεων έχει ένα **αντιπροσωπευτικό κλειδί (surrogate key, SK)** αποτελούμενο από μια μόνο στήλη. Οι πίνακες γεγονότων συνδέονται με τους πίνακες διαστάσεων μέσω αυτών των αντιπροσωπευτικών κλειδιών.

Οι πίνακες διαστάσεων κατηγοριοποιούνται ως εξής:

- **Στατικοί**: Τα περιεχόμενα τους φορτώνονται μια φόρα, κατά τη φόρτωση της βάσης δεδομένων, και δεν τροποποιούνται με την πάροδο του χρόνου. Παράδειγμα: date_dim
- **Ιστορικοί:** Το ιστορικό των αλλαγών που πραγματοποιούνται στα δεδομένα τους διατηρείται, με τη δημιουργία πολλαπλών γραμμών που αναφέρονται στο ίδιο αντικείμενο. Ειδικές στήλες καθορίζουν τη χρονική περίοδο ισχύος της κάθε γραμμής. Παράδειγμα: Item
- **Μη Ιστορικοί:** Το ιστορικό των αλλαγών που πραγματοποιούνται στα δεδομένα τους δεν διατηρείται και οι τροποποιήσεις στα περιεχόμενα τους απλά αντικαθιστούν παλαιότερες καταχωρήσεις. Παράδειγμα: Customer

Προκειμένου να επιτευχθεί ο βέλτιστος συνδυασμός λειτουργικότητας και απόδοσης, ο διαχειριστής της βάσης μπορεί, μια για πάντα, να ορίσει τα επίπεδα κλειδώματος και τους κανόνες ταυτοχρονισμού για ερωτήματα και συναρτήσεις συντήρησης δεδομένων.

Ερωτήματα (Queries) και Μοντέλο Χρηστών (User Model)

Οι χρήστες και τα ερωτήματα που μοντελοποιούνται από το benchmark παρουσιάζουν τα ακόλουθα χαρακτηριστικά:

- Αντιμετωπίζουν σύνθετα επιχειρηματικά προβλήματα

- Χρησιμοποιούν μια μεγάλη ποικιλία από μεθόδους πρόσβασης (access patterns), τρόπους σύνταξης ερωτημάτων (query phrasings) και περιορισμούς αποτελεσμάτων (answer set constraints).
- Εφαρμόζουν παραμέτρους ερωτημάτων που αλλάζουν μεταξύ των εκτελέσεων

Προκειμένου να ανταποκριθεί στους πολυάριθμους τύπους ερωτημάτων και τις απρόβλεπτες συμπεριφορές χρηστών που παρουσιάζονται σε ένα σύστημα υποστήριξης αποφάσεων, το TPC-DS χρησιμοποιεί ένα αρκετά γενικό μοντέλο ερωτημάτων, που του επιτρέπει να συνδυάζει τα χαρακτηριστικά διαφόρων τύπων ερωτημάτων. Μεταξύ αυτών, διαδραστικά και επαναληπτικά OLAP ερωτήματα, ερωτήματα εξόρυξης δεδομένων (data mining) και ερωτήματα αναφοράς (report).

Το μέγεθος του TPC-DS σχήματος και ειδικά η ύπαρξη των τριών διαφορετικών τομέων πωλήσεων, επιτρέπουν την ενσωμάτωση όλων αυτών των τύπων ερωτημάτων σε ένα benchmark. Ένα υπολογιστικό φορτίο ερωτημάτων ad-hoc, προσομοιώνει ένα περιβάλλον στο οποίο πολλοί χρήστες συνδέονται στη βάση δεδομένων και υποβάλουν ερωτήματα που δεν είναι εκ των προτέρων γνωστά. Ο διαχειριστής της βάσης δεδομένων δεν μπορεί να εφαρμόσει ρυθμίσεις βελτιστοποίησης της απόδοσης τέτοιων ερωτημάτων, με συνέπεια η εκτέλεση τους να απαιτεί συχνά πολύ υπολογιστικό χρόνο. Αντίθετα, τα ερωτήματα reporting είναι συνήθως γνωστά εκ των προτέρων και η εκτέλεση τους μπορεί να βελτιστοποιηθεί με χρήση κατάλληλων μεθόδων τοποθέτησης δεδομένων και βοηθητικών δομών δεδομένων, όπως ευρετήρια. Ο συνδυασμός και των δυο τύπων ερωτημάτων στο ίδιο benchmark δεν είναι απλή υπόθεση, καθώς σε ένα benchmark τα ερωτήματα είναι εξ ορισμού γνωστά από την αρχή. Η λύση που δόθηκε στην περίπτωση του TPC-DS είναι ο λογικός διαχωρισμός του TPC-DS σχήματος σε ad-hoc και reporting τμήματα. Συγκεκριμένα, το κομμάτι των πωλήσεων καταλόγου είναι αφιερωμένο στα reporting queries και τα κομμάτια των πωλήσεων καταστήματος και των ηλεκτρονικών πωλήσεων είναι αφιερωμένα στα ad-hoc queries. Για τα τμήματα του TPC-DS σχήματος που αφορούν το reporting επιτρέπονται σύνθετες βοηθητικές δεδομένων, κάτι που δεν ισχύει για τα τμήματα που αφορούν το add-hoc κομμάτι.

Ένα σύστημα υποστήριξης αποφάσεων οφείλει να υποστηρίζει ένα πολυποίκιλο πλήθος χρηστών, που υποβάλλουν διάφορους τύπους ερωτημάτων. Υπάρχουν πολλοί τρόποι να κατηγοριοποιηθούν οι διάφοροι χρήστες με βάση τα ερωτήματα που υποβάλλουν. Στην περίπτωση του TPC-DS, ορίζονται τέσσερις ευρείες **κλάσεις ερωτημάτων**, στις οποίες εμπίπτουν τα περισσότερα ερωτήματα σε ένα σύστημα υποστήριξης αποφάσεων.

- **Reporting queries:** Περιλαμβάνουν ερωτήματα που εκτελούνται περιοδικά ώστε να δώσουν απάντηση σε γνωστές και προκαθορισμένες ερωτήσεις, σχετικά με την λειτουργική και οικονομική υγεία μιας επιχείρησης. Παρά το γεγονός ότι συνήθως είναι στατικά, συχνά υπόκεινται σε μικρές αλλαγές. Για παράδειγμα, μεταξύ δυο διαδοχικών υποβολών ενός reporting query, μπορούν να γίνουν μικρές αλλαγές που αφορούν ημερομηνίες, γεωγραφικές τοποθεσίες και άλλες τέτοιου είδους λεπτομέρειες.
- **Ad-hoc queries:** Αυτά τα ερωτήματα καταγράφουν τη δυναμική φύση ενός συστήματος υποστήριξης αποφάσεων. Πρόκειται για ερωτήματα που συντάσσονται επί τόπου, για να ανακτηθούν πληροφορίες που οι χρήστες χρειάζονται στιγμιαία. Σημαντικότερη διαφορά σε σχέση με τα reporting queries αποτελεί το γεγονός ότι δεν είναι εκ των προτέρων γνωστά, με

αποτέλεσμα η εκτέλεση τους να μην μπορεί να βελτιστοποιηθεί εκτενώς, με χρήση ειδικών δομών δεδομένων (πχ ευρετήρια).

- **Iterative OLAP queries:** Τα OLAP ερωτήματα επιτρέπουν την σε βάθος ανάλυση των επιχειρησιακών δεδομένων με σκοπό την ανακάλυψη νέων τάσεων και μοτίβων. Συνήθως υποβάλλονται σε ομάδες που αποτελούν σενάρια χρήσης και περιλαμβάνουν ερωτήματα διαφόρων επιπέδων πολυπλοκότητας.
- **Data mining queries:** Η εξόρυξη δεδομένων (data mining) είναι μια διαδικασία επεξεργασίας μεγάλων όγκων δεδομένων και ανάδειξης σχέσεων μεταξύ τους, με σκοπό την πρόβλεψη μελλοντικών μοτίβων, τάσεων ή και συμπεριφορών. Με αυτό τον τρόπο οι επιχειρήσεις μπορούν να πάρουν σωστές αποφάσεις για το μέλλον, βασισμένες στις πληροφορίες που έχουν συλλέξει σε ένα δεδομένο χρονικό διάστημα. Τα σχετικά ερωτήματα συνήθως επιστρέφουν μεγάλα σύνολα αποτελεσμάτων (result sets) ώστε να υποστούν επεξεργασία.

Συντήρηση Δεδομένων (Data Maintenance)

Μια **αποθήκη δεδομένων (data warehouse)** είναι τόσο ακριβής και καίρια, όσο τα λειτουργικά δεδομένα στα οποία βασίζεται. Συνεπώς, το ζήτημα της μεταφοράς δεδομένων (data migration) από λειτουργικά OLTP συστήματα σε αναλυτικά συστήματα υποστήριξης αποφάσεων είναι καθοριστικό και αντιμετωπίζεται με διαφορετικό τρόπο από επιχείρηση σε επιχείρηση και από εφαρμογή σε εφαρμογή.

Η διαδικασία ανανέωσης της βάσης δεδομένων ενός συστήματος υποστήριξης αποφάσεων περιλαμβάνει τρία διακεκριμένα στάδια:

1. **Εξαγωγή Δεδομένων (Data Extraction):**
2. **Μετασχηματισμός Δεδομένων (Data Transformation):**
3. **Φόρτωση Δεδομένων (Data Load):**

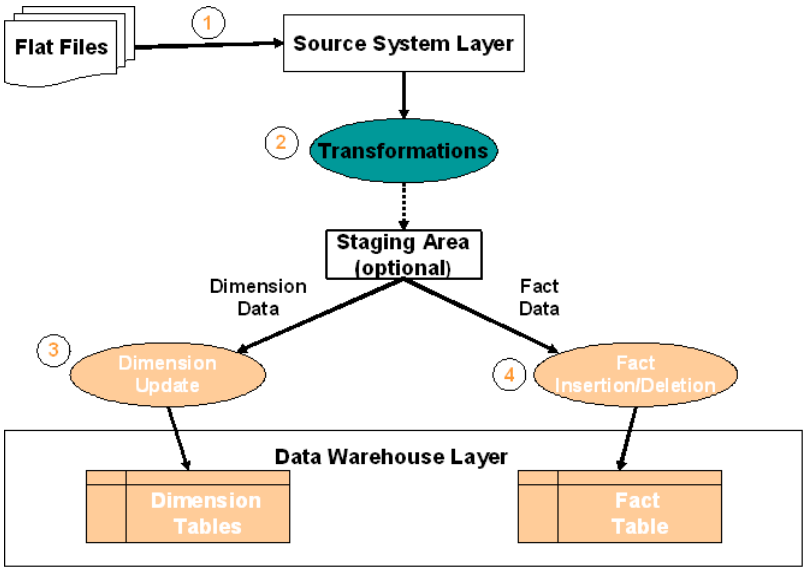
Τα τρία αυτά στάδια είναι γνωστά με το ακρωνύμιο **ETL.** Η μοντελοποίηση τους στο TPC-DS, είναι γνωστή ως **συντήρηση δεδομένων (data maintenance, DM) ή ανανέωση δεδομένων (data refresh).** Οι δύο όροι χρησιμοποιούνται με την ίδια έννοια.

Στα πλαίσια του TPC-DS η διαδικασία της συντήρησης δεδομένων συμπεριλαμβάνει τις ακόλουθες εργασίες:

1. Φόρτωση του συνόλου δεδομένων ανανέωσης (refresh data set), που αποτελείται από νέες, διαγραμμένες και τροποποιημένες εγγραφές.
2. Εφαρμογή μετασχηματισμών δεδομένων στο σύνολο δεδομένων ανανέωσης.
3. Διαχείριση δεδομένων που υπόκεινται σε διαδικασίες ελέγχου έκδοσης και διατήρησης ιστορικού

4. Εισαγωγή νέων εγγραφών στους πίνακες γεγονότων και διαγράφη παλαιών.

Συνοπτικά, η διαδικασία συντήρησης δεδομένων στο TPC-DS απεικονίζεται στο ακόλουθο σχήμα:



ΛΟΓΙΚΗ ΟΡΓΑΝΩΣΗ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ TCPDS

Το σχήμα της βάσης δεδομένων του TPC-DS (στο εξής θα αναφέρεται απλά ως TPC-DS σχήμα) μοντελοποιεί τις πωλήσεις της υποθετικής εμπορικής επιχείρησης που περιγράφεται παραπάνω, η οποία δραστηριοποιείται σε τρείς τομείς πωλήσεων: Τις πωλήσεις καταστήματος (store sales), τις πωλήσεις καταλόγου (catalog sales) και τις πωλήσεις μέσω internet (web sales).

Το TPC-DS σχήμα περιλαμβάνει

- Επτά **πίνακες γεγονότων (fact tables)**: Οι πίνακες αυτοί είναι οι εξής:

  Store Sales (SS)
  Store Returns (SR)
  Catalog Sales (CS)
  Catalog Returns (CR)
  Web Sales (WS)
  Web Returns (WR)
  Inventory (INV)

- Δεκαεπτά **πίνακες διαστάσεων (dimension tables):**

  Store (S)
  Call Center (CC)
  Catalog Page (CP)
  Web Site (WEB)
  Web Page (WP)
  Warehouse (W)
  Customer (C)
  Customer Address (CA)
  Customer Demographics (CD)
  Date Dim (D)
  Household Demographics (HD)
  Item (I)
  Income Band (IB)
  Promotion (P)
  Reason (R)
  Ship Mode (SM)
  Time Dim (T)

Η δομή όλων των πινάκων καταγράφεται αναλυτικά παρακάτω. Ειδικά για τους πίνακες γεγονότων, δίνεται και η σύνδεση τους, μέσω εξωτερικών κλειδιών, με τους πίνακες διαστάσεων της βάσης, μέσω σχετικών διαγραμμάτων οντότητας συσχέτισης.

FACT TABLES

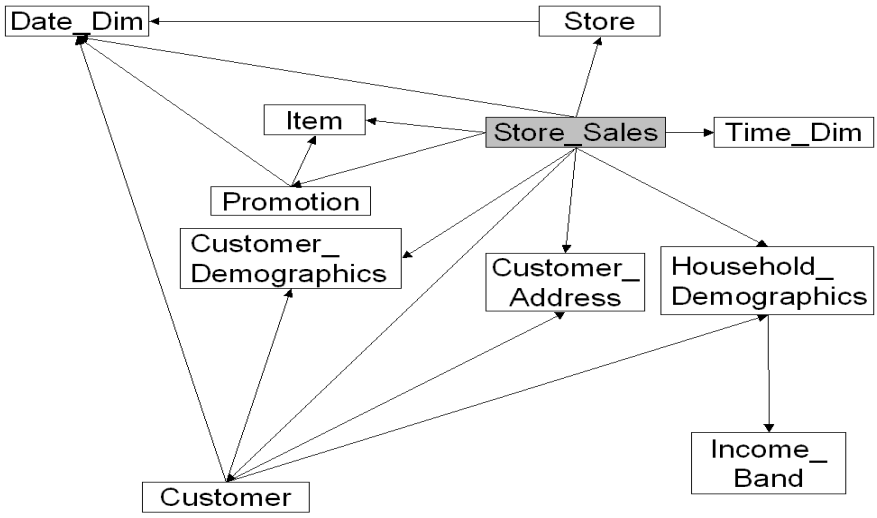
STORE SALES (SS)

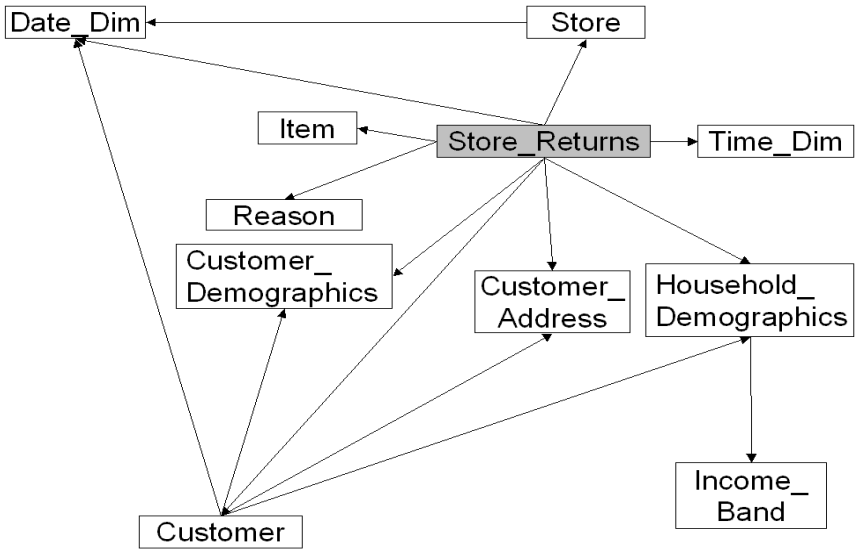
Column	Datatype	NULLs	Primary Key	Foreign Key
ss_sold_date_sk	identifier			d_date_sk
ss_sold_time_sk	identifier			t_time_sk
ss_item_sk (1)	identifier	N	Y	i_item_sk,sr_item_sk
ss_customer_sk	identifier			c_customer_sk
ss_cdemo_sk	identifier			cd_demo_sk
ss_hdemo_sk	identifier			hd_demo_sk
ss_addr_sk	identifier			ca_address_sk
ss_store_sk	identifier			s_store_sk
ss_promo_sk	identifier			p_promo_sk
ss_ticket_number (2)	identifier	N	Y	sr_ticket_number
ss_quantity	integer			
ss_wholesale_cost	decimal(7,2)			
ss_list_price	decimal(7,2)			
ss_sales_price	decimal(7,2)			
ss_ext_discount_amt	decimal(7,2)			
ss_ext_sales_price	decimal(7,2)			
ss_ext_wholesale_cost	decimal(7,2)			
ss_ext_list_price	decimal(7,2)			
ss_ext_tax	decimal(7,2)			
ss_coupon_amt	decimal(7,2)			
ss_net_paid	decimal(7,2)			
ss_net_paid_inc_tax	decimal(7,2)			
ss_net_profit	decimal(7,2)			

STORE RETURNS (SR)

Column	Datatype	NULLs	Primary Key	Foreign Key
sr_returned_date_sk	identifier			d_date_sk
sr_return_time_sk	identifier			t_time_sk
sr_item_sk (1)	identifier	N	Y	i_item_sk,ss_item_sk
sr_customer_sk	identifier			c_customer_sk
sr_cdemo_sk	identifier			cd_demo_sk
sr_hdemo_sk	identifier			hd_demo_sk
sr_addr_sk	identifier			ca_address_sk
sr_store_sk	identifier			s_store_sk
sr_reason_sk	identifier			r_reason_sk
sr_ticket_number (2)	identifier	N	Y	ss_ticket_number
sr_return_quantity	integer			
sr_return_amt	decimal(7,2)			
sr_return_tax	decimal(7,2)			
sr_return_amt_inc_tax	decimal(7,2)			
sr_fee	decimal(7,2)			
sr_return_ship_cost	decimal(7,2)			
sr_refunded_cash	decimal(7,2)			
sr_reversed_charge	decimal(7,2)			
sr_store_credit	decimal(7,2)			
sr_net_loss	decimal(7,2)			

Στα παρακάτω διαγράμματα καταδεικνύεται η σύνδεση, μέσω εξωτερικών κλειδιών, των πινάκων Store Sales και Store Returns, με τους πίνακες διαστάσεων της βάσης.





CATALOG SALES (CS)

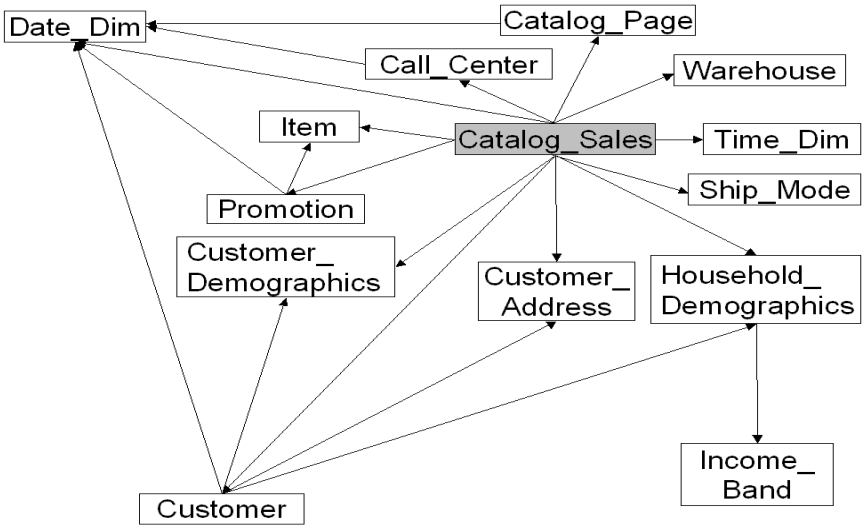
Column	Datatype	NULLs	Primary Key	Foreign Key
cs_sold_date_sk	identifier			d_date_sk
cs_sold_time_sk	identifier			t_time_sk
cs_ship_date_sk	identifier			d_date_sk
cs_bill_customer_sk	identifier			c_customer_sk
cs_bill_cdemo_sk	identifier			cd_demo_sk
cs_bill_hdemo_sk	identifier			hd_demo_sk
cs_bill_addr_sk	identifier			ca_address_sk
cs_ship_customer_sk	identifier			c_customer_sk
cs_ship_cdemo_sk	identifier			cd_demo_sk
cs_ship_hdemo_sk	identifier			hd_demo_sk
cs_ship_addr_sk	identifier			ca_address_sk
cs_call_center_sk	identifier			cc_call_center_sk
cs_catalog_page_sk	identifier			cp_catalog_page_sk
cs_ship_mode_sk	identifier			sm_ship_mode_sk
cs_warehouse_sk	identifier			w_warehouse_sk
cs_item_sk (1)	identifier	N	Y	i_item_sk,cr_item_sk
cs_promo_sk	identifier			p_promo_sk
cs_order_number (2)	identifier	N	Y	cr_order_number
cs_quantity	integer			
cs_wholesale_cost	decimal(7,2)			
cs_list_price	decimal(7,2)			
cs_sales_price	decimal(7,2)			
cs_ext_discount_amt	decimal(7,2)			
cs_ext_sales_price	decimal(7,2)			
cs_ext_wholesale_cost	decimal(7,2)			
cs_ext_list_price	decimal(7,2)			
cs_ext_tax	decimal(7,2)			
cs_coupon_amt	decimal(7,2)			
cs_ext_ship_cost	decimal(7,2)			
cs_net_paid	decimal(7,2)			
cs_net_paid_inc_tax	decimal(7,2)			
cs_net_paid_inc_ship	decimal(7,2)			
cs_net_paid_inc_ship_tax	decimal(7,2)			
cs_net_profit	decimal(7,2)			

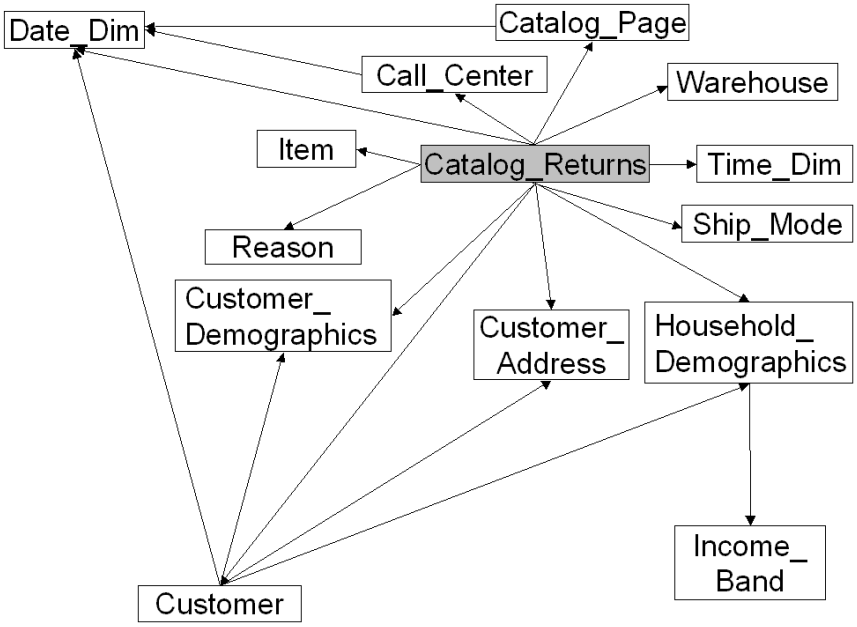
CATALOG RETURNS (CR)

Column	Datatype	NULLs	Primary Key	Foreign Key
cr_returned_date_sk	identifier			d_date_sk
cr_returned_time_sk	identifier			t_time_sk
cr_item_sk (1)	identifier	N	Y	i_item_sk,cs_item_sk
cr_refunded_customer_sk	identifier			c_customer_sk
cr_refunded_cdemo_sk	identifier			cd_demo_sk
cr_refunded_hdemo_sk	identifier			hd_demo_sk
cr_refunded_addr_sk	identifier			ca_address_sk
cr_returning_customer_sk	identifier			c_customer_sk
cr_returning_cdemo_sk	identifier			cd_demo_sk
cr_returning_hdemo_sk	identifier			hd_demo_sk
cr_returning_addr_sk	identifier			ca_address_sk
cr_call_center_sk	identifier			cc_call_center_sk
cr_catalog_page_sk	identifier			cp_catalog_page_sk
cr_ship_mode_sk	identifier			sm_ship_mode_sk
cr_warehouse_sk	identifier			w_warehouse_sk
cr_reason_sk	identifier			r_reason_sk
cr_order_number (2)	identifier	N	Y	cs_order_number
cr_return_quantity	integer			
cr_return_amount	decimal(7,2)			
cr_return_tax	decimal(7,2)			
cr_return_amt_inc_tax	decimal(7,2)			
cr_fee	decimal(7,2)			
cr_return_ship_cost	decimal(7,2)			
cr_refunded_cash	decimal(7,2)			
cr_reversed_charge	decimal(7,2)			

Column	Datatype	NULLs	Primary Key	Foreign Key
cr_store_credit	decimal(7,2)			
cr_net_loss	decimal(7,2)			

Στα παρακάτω διαγράμματα φαίνεται η συσχέτιση, μέσω εξωτερικών κλειδιών, των πινάκων Catalog Sales και Catalog Returns, με τους πίνακες διαστάσεων της βάσης.





WEB SALES (WS)

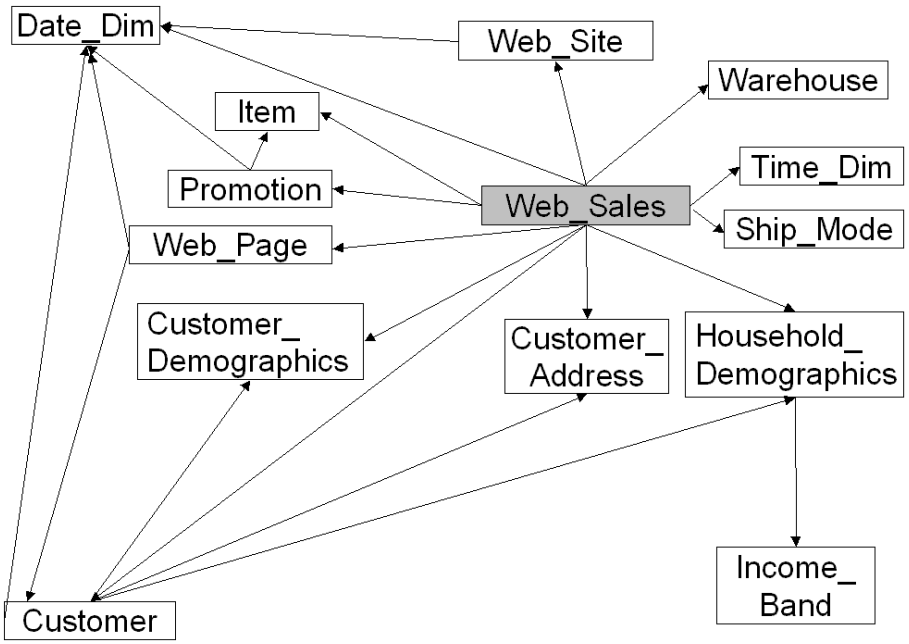
Column	Datatype	NULLs	Primary Key	Foreign Key
ws_sold_date_sk	identifier			d_date_sk
ws_sold_time_sk	identifier			t_time_sk
ws_ship_date_sk	identifier			d_date_sk
ws_item_sk (1)	identifier	N	Y	i_item_sk,wr_item_sk
ws_bill_customer_sk	identifier			c_customer_sk
ws_bill_cdemo_sk	identifier			cd_demo_sk
ws_bill_hdemo_sk	identifier			hd_demo_sk
ws_bill_addr_sk	identifier			ca_address_sk
ws_ship_customer_sk	identifier			c_customer_sk
ws_ship_cdemo_sk	identifier			cd_demo_sk
ws_ship_hdemo_sk	identifier			hd_demo_sk
ws_ship_addr_sk	identifier			ca_address_sk
ws_web_page_sk	identifier			wp_web_page_sk
ws_web_site_sk	identifier			web_site_sk
ws_ship_mode_sk	identifier			sm_ship_mode_sk
ws_warehouse_sk	identifier			w_warehouse_sk
ws_promo_sk	identifier			p_promo_sk
ws_order_number (2)	identifier	N	Y	wr_order_number
ws_quantity	integer			
ws_wholesale_cost	decimal(7,2)			
ws_list_price	decimal(7,2)			
ws_sales_price	decimal(7,2)			
ws_ext_discount_amt	decimal(7,2)			
ws_ext_sales_price	decimal(7,2)			
ws_ext_wholesale_cost	decimal(7,2)			
ws_ext_list_price	decimal(7,2)			
ws_ext_tax	decimal(7,2)			
ws_coupon_amt	decimal(7,2)			
ws_ext_ship_cost	decimal(7,2)			
ws_net_paid	decimal(7,2)			
ws_net_paid_inc_tax	decimal(7,2)			
ws_net_paid_inc_ship	decimal(7,2)			
ws_net_paid_inc_ship_tax	decimal(7,2)			
ws_net_profit	decimal(7,2)			

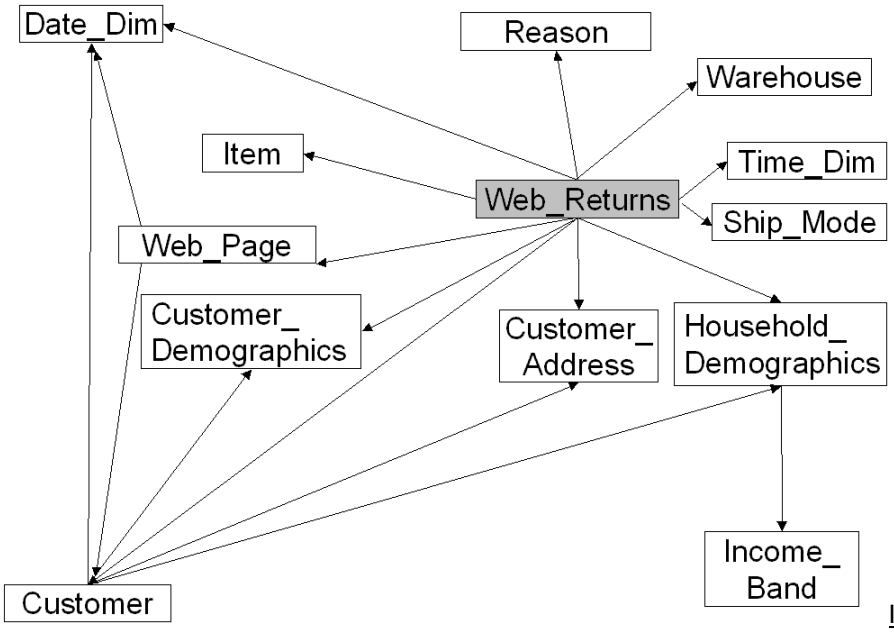
WEB RETURNS (WR)

Column	Datatype	NULLs	Primary Key	Foreign Key
wr_returned_date_sk	identifier			d_date_sk
wr_returned_time_sk	identifier			t_time_sk
wr_item_sk (2)	identifier	N	Y	i_item_sk,ws_item_sk
wr_refunded_customer_sk	identifier			c_customer_sk
wr_refunded_cdemo_sk	identifier			cd_demo_sk
wr_refunded_hdemo_sk	identifier			hd_demo_sk
wr_refunded_addr_sk	identifier			ca_address_sk
wr_returning_customer_sk	identifier			c_customer_sk
wr_returning_cdemo_sk	identifier			cd_demo_sk
wr_returning_hdemo_sk	identifier			hd_demo_sk
wr_returning_addr_sk	identifier			ca_address_sk
wr_web_page_sk	identifier			wp_web_page_sk
wr_reason_sk	identifier			r_reason_sk
wr_order_number (1)	identifier	N	Y	ws_order_number
wr_return_quantity	integer			
wr_return_amt	decimal(7,2)			
wr_return_tax	decimal(7,2)			
wr_return_amt_inc_tax	decimal(7,2)			
wr_fee	decimal(7,2)			
wr_return_ship_cost	decimal(7,2)			
wr_refunded_cash	decimal(7,2)			
wr_reversed_charge	decimal(7,2)			
wr_account_credit	decimal(7,2)			

Column	Datatype	NULLs	Primary Key	Foreign Key
wr_net_loss	decimal(7,2)			

Στα παρακάτω διαγράμματα φαίνεται η συσχέτιση, μέσω εξωτερικών κλειδιών, των πινάκων Catalog Sales και Catalog Returns, με τους πίνακες διαστάσεων της βάσης.

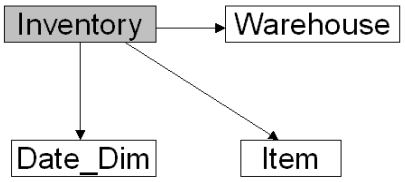




INVENTORY (I)

Column	Datatype	NULLs	Primary Key	Foreign Key
inv_date_sk (1)	identifier	N	Y	d_date_sk
inv_item_sk (2)	identifier	N	Y	i_item_sk
inv_warehouse_sk (3)	identifier	N	Y	w_warehouse_sk
inv_quantity_on_hand	integer			

Στο παρακάτω διάγραμμα φαίνεται η συσχέτιση, μέσω εξωτερικών κλειδιών, του πίνακα Inventory με τους πίνακες διαστάσεων της βάσης.



DIMENSION TABLES

STORE (S)

Column	Datatype	NULLs	Primary Key	Foreign Key
s_store_sk	identifier	N	Y	
s_store_id (B)	char(16)	N		
s_rec_start_date	date			
s_rec_end_date	date			
s_closed_date_sk	identifier			d_date_sk
s_store_name	varchar(50)			
s_number_employees	integer			
s_floor_space	integer			
s_hours	char(20)			
S_manager	varchar(40)			
S_market_id	integer			
S_geography_class	varchar(100)			
S_market_desc	varchar(100)			
s_market_manager	varchar(40)			
s_division_id	integer			
s_division_name	varchar(50)			
s_company_id	integer			
s_company_name	varchar(50)			
s_street_number	varchar(10)			
s_street_name	varchar(60)			
s_street_type	char(15)			
s_suite_number	char(10)			
s_city	varchar(60)			
s_county	varchar(30)			
s_state	char(2)			
s_zip	char(10)			
s_country	varchar(20)			
s_gmt_offset	decimal(5,2)			
s_tax_percentage	decimal(5,2)			

CATALOG PAGE (CP)

Column	Datatype	NULLs	Primary Key	Foreign Key
cp_catalog_page_sk	integer	N	Y	
cp_catalog_page_id (B)	char(16)	N		
cp_start_date_sk	integer			d_date_sk
cp_end_date_sk	integer,			d_date_sk
cp_department	varchar(50)			
cp_catalog_number	integer,			
cp_catalog_page_number	integer,			
cp_description	varchar(100)			
cp_type	varchar(100)			

CALL CENTER (CC)

Column	**Datatype**	**NULLs**	**Primary Key**	**Foreign Key**
cc_call_center_sk	integer	N	Y	
cc_call_center_id (B)	char(16)	N		
cc_rec_start_date	date			
cc_rec_end_date	date			
cc_closed_date_sk	integer			d_date_sk
cc_open_date_sk	integer			d_date_sk
cc_name	varchar(50)			
cc_class	varchar(50)			
cc_employees	integer			
cc_sq_ft	integer			
cc_hours	char(20)			
cc_manager	varchar(40)			
cc_mkt_id	integer			
cc_mkt_class	char(50)			
cc_mkt_desc	varchar(100)			
cc_market_manager	varchar(40)			
cc_division	integer			
cc_division_name	varchar(50)			
cc_company	integer			
cc_company_name	char(50)			
cc_street_number	char(10)			
cc_street_name	varchar(60)			
cc_street_type	char(15)			
cc_suite_number	char(10)			
cc_city	varchar(60)			
cc_county	varchar(30)			
cc_state	char(2)			
cc_zip	char(10)			
cc_country	varchar(20)			
cc_gmt_offset	decimal(5,2)			
cc_tax_percentage	decimal(5,2)			

WEB SITE (WEB)

Column	Datatype	NULLs	Primary Key	Foreign Key
web_site_sk	identifier	N	Y	
web_site_id (B)	char(16)	N		
web_rec_start_date	date			
web_rec_end_date	date			
web_name	varchar(50)			
web_open_date_sk	identifier			d_date_sk
web_close_date_sk	identifier			d_date_sk
web_class	varchar(50)			
web_manager	varchar(40)			
web_mkt_id	integer			
web_mkt_class	varchar(50)			
web_mkt_desc	varchar(100)			
web_market_manager	varchar(40)			
web_company_id	integer			
web_company_name	char(50)			
web_street_number	char(10)			
web_street_name	varchar(60)			
web_street_type	char(15)			
web_suite_number	char(10)			
web_city	varchar(60)			
web_county	varchar(30)			
web_state	char(2)			
web_zip	char(10)			
web_country	varchar(20)			
web_gmt_offset	decimal(5,2)			
web_tax_percentage	decimal(5,2)			

WEB PAGE (WP)

Column	Datatype	NULLs	Primary Key	Foreign Key
wp_web_page_sk	identifier	N	Y	
wp_web_page_id (B)	char(16)	N		
wp_rec_start_date	date			
wp_rec_end_date	date			
wp_creation_date_sk	identifier			d_date_sk
wp_access_date_sk	identifier			d_date_sk
wp_autogen_flag	char(1)			
wp_customer_sk	identifier			c_customer_sk
wp_url	varchar(100)			
wp_type	char(50)			
wp_char_count	integer			
wp_link_count	integer			
wp_image_count	integer			
wp_max_ad_count	integer			

WAREHOUSE (W)

Column	Datatype	NULLs	Primary Key	Foreign Key
w_warehouse_sk	identifier	N	Y	
w_warehouse_id (B)	char(16)	N		
w_warehouse_name	varchar(20)			
w_warehouse_sq_ft	integer			
w_street_number	char(10)			
w_street_name	varchar(60)			
w_street_type	char(15)			
w_suite_number	char(10)			
w_city	varchar(60)			
w_county	varchar(30)			
w_state	char(2)			
w_zip	char(10)			
w_country	varchar(20)			
w_gmt_offset	decimal(5,2)			

CUSTOMER (C)

Column	**Datatype**	**NULLs**	**Primary Key**	**Foreign Key**
c_customer_sk	identifier	N	**Y**	
c_customer_id (B)	char(16)	N		
c_current_cdemo_sk	identifier			cd_demo_sk
c_current_hdemo_sk	identifier			hd_demo_sk
c_current_addr_sk	identifier			ca_addres_sk
c_first_shipto_date_sk	identifier			d_date_sk
c_first_sales_date_sk	identifier			d_date_sk
c_salutation	char(10)			
c_first_name	char(20)			
c_last_name	char(30)			
c_preferred_cust_flag	char(3)			
c_birth_day	integer			
c_birth_month	integer			
c_birth_year	integer			
c_birth_country	varchar(20)			
c_login	char(13)			
c_email_address	char(50)			
c_last_review_date_sk	identifier			

CUSTOMER ADDRESS (CA)

Column	**Datatype**	**NULLs**	**Primary Key**	**Foreign Key**
ca_address_sk	identifier	N	Y	
ca_address_id (B)	char(16)	N		
ca_street_number	char(10)			
ca_street_name	varchar(60)			
ca_street_type	char(15)			
ca_suite_number	char(10)			
ca_city	varchar(60)			
ca_county	varchar(30)			
ca_state	char(2)			
ca_zip	char(10)			
ca_country	varchar(20)			
ca_gmt_offset	decimal(5,2)			
ca_location_type	char(20)			

CUSTOMER DEMOGRAPHICS (CD)

Column	**Datatype**	**NULLs**	**Primary Key**	**Foreign Key**
cd_demo_sk	identifier	N	Y	
cd_gender	char(1)			
cd_marital_status	char(1)			
cd_education_status	char(20)			
cd_purchase_estimate	integer			
cd_credit_rating	char(10)			
cd_dep_count	integer			
cd_dep_employed_count	integer			
cd_dep_college_count	integer			

HOUSEHOLD DEMOGRAPHICS (HD)

Column	Datatype	NULLs	Primary Key	Foreign Key
hd_demo_sk	identifier	N	Y	
hd_income_band_sk	identifier			ib_income_band_sk
hd_buy_potential	char(15)			
hd_dep_count	integer			
hd_vehicle_count	integer			

ITEM (I)

Column	Datatype	NULLs	Primary Key	Foreign Key
i_item_sk	identifier	N	Y	
i_item_id (B)	char(16)	N		
i_rec_start_date	date			
i_rec_end_date	date			
i_item_desc	varchar(200)			
i_current_price	decimal(7,2)			
i_wholesale_cost	decimal(7,2)			
i_brand_id	integer			
i_brand	char(50)			
i_class_id	integer			
i_class	char(50)			
i_category_id	integer			
i_category	char(50)			
i_manufact_id	integer			
i_manufact	char(50)			
i_size	char(20)			
i_formulation	char(20)			
i_color	char(20)			
i_units	char(10)			
i_container	char(10)			
i_manager_id	integer			
i_product_name	char(50)			

DATE DIM (D)

Column	**Datatype**	**NULLs**	**Primary Key**	**Foreign Key**
d_date_sk	identifier	N	Y	
d_date_id (B)	char(16)	N		
d_date	date			
d_month_seq	integer			
d_week_seq	integer			
d_quarter_seq	integer			
d_year	integer			
d_dow	integer			
d_moy	integer			
d_dom	integer			
d_qoy	integer			
d_fy_year	integer			
d_fy_quarter_seq	integer			
d_fy_week_seq	integer			
d_day_name	char(9)			
d_month_name	char(15)			
d_quarter_name	char(6)			
d_holiday	char(1)			
d_weekend	char(1)			
d_following_holiday	char(1)			
d_first_dom	integer			
d_last_dom	integer			
d_same_day_ly	integer			
d_same_day_lq	integer			
d_current_day	char(1)			
d_current_week	char(1)			
d_current_month	char(1)			
d_current_quarter	char(1)			
d_current_year	char(1)			

INCOME BAND (IB)

Column	Datatype	NULLs	Primary Key	Foreign Key
ib_income_band_sk	identifier	N	Y	
ib_lower_bound	integer			
ib_upper_bound	integer			

PROMOTION (P)

Column	Datatype	NULLs	Primary Key	Foreign Key
p_promo_sk	identifier	N	Y	
p_promo_id (B)	char(16)	N		
p_start_date_sk	identifier			d_date_sk
p_end_date_sk	identifier			d_date_sk
p_item_sk	identifier			i_item_sk
p_cost	decimal(15,2)			
p_response_target	integer			
p_promo_name	char(50)			
p_channel_dmail	char(3)			
p_channel_email	char(3)			
p_channel_catalog	char(3)			
p_channel_tv	char(3)			
p_channel_radio	char(3)			
p_channel_press	char(3)			
p_channel_event	char(3)			
p_channel_demo	char(3)			
p_channel_details	varchar(100)			
p_purpose	char(15)			
p_discount_active	char(1)			

REASON (R)

Column	Datatype	NULLs	Primary Key	Foreign Key
r_reason_sk	identifier	N	Y	
r_reason_id (B)	char(16)	N		
r_reason_desc	char(100)			

SHIP MODE (SM)

Column	Datatype	NULLs	Primary Key	Foreign Key
sm_ship_mode_sk	identifier	N	Y	
sm_ship_mode_id (B)	char(16)	N		
sm_type	char(30)			
sm_code	char(10)			
sm_carrier	char(20)			
sm_contract	char(20)			

TIME DIM (T)

Column	Datatype	NULLs	Primary Key	Foreign Key
t_time_sk	Identifier	N	Y	
t_time_id (B)	char(16)	N		
t_time	Integer			
t_hour	Integer			
t_minute	Integer			
t_second	Integer			
t_am_pm	char(2)			
t_shift	char(20)			
t_sub_shift	char(20)			
t_meal_time	char(20)			

Βιβλιογραφία – Παραπομπές

- Ιστότοπος TPCDS benchmark
  http://www.tpc.org/tpcds/
  Όλες οι εικόνες του παρόντος κεφαλαίου αποτελούν πνευματική ιδιοκτησία του παραπάνω ιστοτόπου.

- Έγγραφο προδιαγραφών TPCDS benchmark
  http://www.tpc.org/tpc_documents_current_versions/pdf/tpcds_1.4.pdf

ΚΕΦΑΛΑΙΟ 4: BOOLEAN APPROACH ΚΑΙ ΕΦΑΡΜΟΓΗ ΣΤΗΝ ΒΔ TPCDS

Εισαγωγή στη Θεωρία του Boolean Aproach

Η διαχείριση μεγάλων όγκων δεδομένων έχει καταστεί απολύτως απαραίτητη στον σύγχρονο κόσμο, δεν αποτελεί ωστόσο εύκολη υπόθεση. Η ανάκτηση χρήσιμης πληροφορίας από μεγάλους όγκους δεδομένων αποτελεί αντικείμενο συνεχούς έρευνας και καταβάλλεται κάθε δυνατή προσπάθεια για την εύρεση αποδοτικών μεθόδων για την πραγματοποίηση της. Βασικός στόχος της παρούσας εργασίας είναι να μελετηθεί κατά πόσο η εφαρμογή των ιδεών του **boolean approach** στον σχεδιασμό μιας σχεσιακής βάσης δεδομένων, μπορεί να επηρεάσει την απόδοση εκτέλεσης πολύπλοκων ερωτημάτων στη βάση αυτή.

Το boolean approach αποτελεί μια θεωρητική προσέγγιση του προβλήματος της ανακάλυψης γνώσης σε βάσεις δεδομένων (**knowledge discovery in databases**) και στηρίζεται σε μια απλή βασική ιδέα: Την μετατροπή κατηγορηματικών μεταβλητών (**categorical variables**) σε σύνολα δυαδικών μεταβλητών (**binary variables**) .

Σε όρους σχεσιακών βάσεων δεδομένων, μια μεταβλητή είναι ένα χαρακτηριστικό (attribute) μιας σχέσης. Κατηγορηματική μεταβλητή είναι ένα χαρακτηριστικό, το οποίο κινείται σε ένα μικρό πεδίο τιμών με τουλάχιστον δύο τιμές. Δυαδική μεταβλητή είναι ένα χαρακτηριστικό που κινείται στο πεδίο τιμών {0,1}.

Η μετατροπή μιας κατηγορηματικής μεταβλητής σε ένα σύνολο δυαδικών μεταβλητών γίνεται με τον εξής μηχανισμό:

Μηχανισμός Boolean Approach

Έστω η κατηγορηματική μεταβλητή X, η οποία κινείται στο πεδίο τιμών Π = {v1,…,vm} με M≥2 πιθανές τιμές.

Για κάθε πιθανή τιμή vi της κατηγορηματικής μεταβλητής X ($1 \le i \le M$) ορίζουμε μια δυαδική μεταβλητή Xvi ως εξής:

$$Xvi = 1, \text{ αν } X = vi$$

$$Xvi = 0, \text{ αν } X \neq vi$$

Έτσι, για τις M πιθανές τιμές της κατηγορηματικής μεταβλητής X, προκύπτουν M δυαδικές μεταβλητές Xv1,…,Xvm

Στη συνέχεια της παρούσας εργασίας, θα χρησιμοποιείται ο όρος **booleanized** για να περιγράψει:

- Ένα **χαρακτηριστικό** μιας σχέσης που έχει μετασχηματιστεί κατά boolean approach.

- Μια **σχέση** που περιέχει κάποιο ή κάποια booleanized χαρακτηριστικά.
- Μια **βάση δεδομένων** που περιέχει μία ή περισσότερες booleanized σχέσεις.

Για παράδειγμα, έστω μια σχέση Person με πληροφορίες σχετικά με φυσικά πρόσωπα, που περιλαμβάνει το χαρακτηριστικό 'marital status', με πιθανές τιμές {'single', 'married', 'divorced' }. Έστω ένα μικρό στιγμιότυπο αυτής της σχέσης:

Person	Marital Status
xxx	Single
yyy	Married
zzz	Divorced

Εφαρμόζοντας τον μηχανισμό που περιγράφεται παραπάνω, το ίδιο στιγμιότυπο διαμορφώνεται ως εξής:

Person	MS_Single	MS_Married	MS_Divorced
xxx	1	0	0
yyy	0	1	0
zzz	0	0	1

Όπως φαίνεται και στο παράδειγμα, οι δυαδικές μεταβλητές που αντικαθιστούν μια κατηγορηματική μεταβλητή έχουν άμεση συσχέτιση, αφού σε κάθε εγγραφή μόνο μία από αυτές μπορεί να έχει την τιμή 1 και όλες οι άλλες έχουν την τιμή 0.

Εφαρμογή του Boolean Approach στην Βάση Δεδομένων του TPCDS

Βασικός σκοπός της παρούσας εργασίας είναι να μελετήσει κατά πόσο η τροποποίηση του σχήματος μιας βάσης δεδομένων, με εφαρμογή του μετασχηματισμού boolean approach σε επιλεγμένα χαρακτηριστικά των σχέσεων της, επιφέρει βελτίωση της απόδοσης στην εκτέλεση ερωτημάτων στη βάση αυτή.

Για το σκοπό αυτό, πραγματοποιήθηκαν οι εξής ενέργειες:

1. Υλοποιήθηκε η βάση δεδομένων του TPCDS . (**βάση δεδομένων tpcds**)
2. Δημιουργήθηκε ένα αντίγραφό της, τόσο ως προς το σχήμα όσο και ως προς τα περιεχόμενα, στο οποίο εφαρμόστηκε ο μετασχηματισμός boolean approach σε επιλεγμένα attributes. (**βάση δεδομένων bool_tpcds**)
3. Από το σύνολο των queries που περιλαμβάνει το φύλλο προδιαγραφών του TPCDS, επιλέχθηκαν εκείνα που εμπλέκουν τα παραπάνω επιλεγμένα attributes. Κάθε query γράφτηκε σε SQL σε δυο εκδοχές, μια για την αρχική και μια για την booleanized βάση.
4. Τα queries εκτελέστηκαν επαναληπτικά και στις δυο βάσεις και έγινε καταγραφή και σύγκριση των αντίστοιχων χρόνων εκτέλεσης.

Στο σημείο αυτό καταγράφονται τα attributes της TPCDS βάσης δεδομένων στα οποία εφαρμόστηκε ο μετασχηματισμός boolean approach και η νέα μορφή των αντίστοιχων σχέσεων στη νέα βάση δεδομένων bool_tpcds.

Σχέσεις (πίνακες) και χαρακτηριστικά που έγιναν booleanized:

ΠΙΝΑΚΑΣ	ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ
date_dim	d_day_name d_month_name d_quarter_name
customer_demographics	cd_gender cd_marital_status cd_education_status
customer	c_preferred_cust_flag
time_dim	t_am_pm
promotion	p_channel_dmail p_channel_email p_channel_catalog p_channel_tv p_channel_radio p_channel_press p_channel_event p_channel_demo

date_dim

- **Attribute**: d_day_name
- **Πεδίο τιμών**: {Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday}
- **Booleanized attributes**: d_day_name_Monday, d_day_name_Tuesday, d_day_name_Wednesday, d_day_name_Thursday, d_day_name_Friday, d_day_name_Saturday, d_day_name_Sunday

- **Attribute**: d_month_name
- **Πεδίο τιμών**: {January, February, March, April, May, June, July, August, September, October, November, December}
- **Booleanized attributes**: d_month_name_January, d_month_name_February, d_month_name_March, d_month_name_April, d_month_name_May, d_month_name_June, d_month_name_July, d_month_name_August, d_month_name_September, d_month_name_October, d_month_name_November, d_month_name_December

- **Attribute:** d_quarter_name
- **Πεδίο τιμών**: {first,second,third,fourth }
- **Booleanized attributes**: d_quarter_name_first, d_ quarter _name_second, d_ quarter _name_third, d_ quarter _name_fourth

BOOLEANIZED DATE_DIM

Column	Datatype	NULLs	Primary Key	Foreign Key
d_date_sk	identifier	N	Y	
d_date_id (B)	char(16)	N		
d_date	date			
d_month_seq	integer			
d_week_seq	integer			
d_quarter_seq	integer			
d_year	integer			
d_dow	integer			
d_moy	integer			
d_dom	integer			
d_qoy	integer			
d_fy_year	integer			
d_fy_quarter_seq	integer			
d_fy_week_seq	integer			
d_day_name_Monday	int(1)			
d_day_name_Tuesday	int(1)			
d_day_name_Wednesday	int(1)			
d_day_name_Thursday	int(1)			
d_day_name_Friday	int(1)			
d_day_name_Saturday	int(1)			
d_day_name_Sunday	int(1)			
d_month_name_January	int(1)			
d_month_name_February	int(1)			
d_month_name_March	int(1)			
d_month_name_April	int(1)			
d_month_name_May	int(1)			
d_month_name_June	int(1)			
d_month_name_July	int(1)			
d_month_name_August	int(1)			
d_month_name_September	int(1)			
d_month_name_October	int(1)			
d_month_name_November	int(1)			
d_month_name_December	int(1)			
d_quarter_name_first	int(1)			
d_quarter_name_second	int(1)			
d_quarter_name_third	int(1)			
d_quarter_name_fourth	int(1)			
d_holiday	char(1)			
d_weekend	char(1)			
d_following_holiday	char(1)			
d_first_dom	integer			
d_last_dom	integer			
d_same_day_ly	integer			
d_same_day_lq	integer			
d_current_day	char(1)			
d_current_week	char(1)			
d_current_month	char(1)			
d_current_quarter	char(1)			
d_current_year	char(1)			

customer_demographics

- **Attribute**: cd_marital_status
- **Πεδίο τιμών**: {single, married, divorced}
- **Booleanized attributes**: cd_marital_status_single, cd_marital_status_married, cd_marital_status_divorced

- **Attribute**: cd_education_status
- **Πεδίο τιμών**: {primary, secondary, university, postgraduate}
- **Booleanized attributes**: cd_education_status_primary, cd_education_status_secondary, cd_education_status_university, cd_education_status_postgraduate

- **Attribute**: cd_gender
- **Πεδίο τιμών**: {male,female}
- **Booleanized attributes**: cd_gender_male, cd_gender_female

BOOLEANIZED CUSTOMER_DEMOGRAPHICS

Column	Datatype	NULLs	Primary Key	Foreign Key
cd_demo_sk	identifier	N	Y	
cd_gender_male	int(1)			
cd_gender_female	int(1)			
cd_marital_status_single	int(1)			
cd_marital_status_married	int(1)			
cd_marital_status_divorced	int(1)			
cd_education_status_primary	int(1)			
cd_education_status_secondary	int(1)			
cd_education_status_university	int(1)			
cd_education_status_postgraduate	int(1)			
cd_purchase_estimate	integer			
cd_credit_rating	char(10)			
cd_dep_count	integer			
cd_dep_employed_count	integer			
cd_dep_college_count	integer			

customer

- **Attribute**: c_preferred_cust_flag
- **Πεδίο τιμών**: {yes,no}
- **Booleanized attributes**:
  c_preferred_cust_flag_yes, c_preferred_cust_flag_no

BOOLEANIZED CUSTOMER

Column	Datatype	NULLs	Primary Key	Foreign Key
c_customer_sk	identifier	N	**Y**	
c_customer_id (B)	char(16)	N		
c_current_cdemo_sk	identifier			cd_demo_sk
c_current_hdemo_sk	identifier			hd_demo_sk
c_current_addr_sk	identifier			ca_addres_sk
c_first_shipto_date_sk	identifier			d_date_sk
c_first_sales_date_sk	identifier			d_date_sk
c_salutation	char(10)			
c_first_name	char(20)			
c_last_name	char(30)			
c_preferred_cust_flag_yes	int(1)			
c_preferred_cust_flag_no	int(1)			
c_birth_day	integer			
c_birth_month	integer			
c_birth_year	integer			
c_birth_country	varchar(20)			
c_login	char(13)			
c_email_address	char(50)			
c_last_review_date_sk	identifier			

time_dim

- **Attribute**: t_am_pm
- **Πεδίο τιμών**: {AM,PM}
- **Booleanized attributes**:
  t_am_pm_AM, t_am_pm_PM

BOOLEANIZED TIME_DIM

Column	Datatype	NULLs	Primary Key	Foreign Key
t_time_sk	Identifier	N	Y	
t_time_id (B)	char(16)	N		
t_time	Integer			
t_hour	Integer			
t_minute	Integer			
t_second	Integer			
t_am_pm_AM	int(1)			
t_am_pm_PM	int(1)			
t_shift	char(20)			
t_sub_shift	char(20)			
t_meal_time	char(20)			

PROMOTION

- **Attribute**: p_channel_dmail
- **Πεδίο τιμών**: {yes,no}
- **Booleanized attributes**:
  p_channel_dmail_yes, p_channel_dmail_no

- **Attribute**: p_channel_email
- **Πεδίο τιμών**: {yes,no}
- **Booleanized attributes**:
  p_channel_email_yes, p_channel_email_no

- **Attribute**: p_channel_catalog
- **Πεδίο τιμών**: {yes,no}
- **Booleanized attributes**:
  p_channel_catalog_yes, p_channel_catalog_no

- **Attribute**: p_channel_tv
- **Πεδίο τιμών**: {yes,no}
- **Booleanized attributes**:
  p_channel_tv_yes, p_channel_tv_no

- **Attribute**: p_channel_radio
- **Πεδίο τιμών**: {yes,no}
- **Booleanized attributes**:
  p_channel_radio_yes, p_channel_radio_no

- **Attribute**: p_channel_event
- **Πεδίο τιμών**: {yes,no}
- **Booleanized attributes**:
  p_channel_event_yes, p_channel_event_no

- **Attribute**: p_channel_demo
- **Πεδίο τιμών**: {yes,no}
- **Booleanized attributes**:
  p_channel_demo_yes, p_channel_demo_no

- **Attribute**: p_channel_press
- **Πεδίο τιμών**: {yes,no}
- **Booleanized attributes**:
  p_channel_press_yes, p_channel_press_no

BOOLEANIZED PROMOTION

Column	Datatype	NULLs	Primary Key	Foreign Key
p_promo_sk	identifier	N	Y	
p_promo_id (B)	char(16)	N		
p_start_date_sk	identifier			d_date_sk
p_end_date_sk	identifier			d_date_sk
p_item_sk	identifier			i_item_sk
p_cost	decimal(15,2)			
p_response_target	integer			
p_promo_name	char(50)			
p_channel_dmail_yes	int(1)			
p_channel_dmail_no	int(1)			
p_channel_email_yes	int(1)			
p_channel_email_no	int(1)			
p_channel_catalog_yes	int(1)			
p_channel_catalog_no	int(1)			
p_channel_tv_yes	int(1)			
p_channel_tv_no	int(1)			
p_channel_radio_yes	int(1)			
p_channel_radio_no	int(1)			
p_channel_press_yes	int(1)			
p_channel_press_no	int(1)			
p_channel_event_yes	int(1)			
p_channel_event_no	int(1)			
p_channel_demo_yes	int(1)			
p_channel_demo_no	int(1)			
p_channel_details	varchar(100)			
p_purpose	char(15)			
p_discount_active	char(1)			

Βιβλιογραφία - Παραπομπές

- Ιστότοπος TPCDS benchmark
  http://www.tpc.org/tpcds/
  Όλες οι εικόνες του παρόντος κεφαλαίου αποτελούν πνευματική ιδιοκτησία του παραπάνω ιστοτόπου.
- Έγγραφο προδιαγραφών TPCDS benchmark
  http://www.tpc.org/tpc_documents_current_versions/pdf/tpcds_1.4.pdf
- Secured Disclosure Of Sensitive Data In Data Mining Techniques –
  Gurusamy, Chakrapani (2012)
- Boolean and Cluster Analysis for Knowledge Discovery in Databases –
  Kono, Sakurai, Yamaguchi (2000)

ΚΕΦΑΛΑΙΟ 5: TPCDS QUERIES

Εισαγωγή

Το TPCDS benchmark περιλαμβάνει μια πολύ μεγάλη συλλογή από σχεσιακά ερωτήματα (**TPCDS queries**), διαφόρων τύπων και επιπέδων πολυπλοκότητας. Σκοπός των ερωτημάτων, είναι να υποβληθούν επαναληπτικά στη βάση δεδομένων του TPCDS ώστε να καταγραφεί η αποδοτικότητα της εκτέλεσης τους και με βάση αυτή να προκύψουν χρήσιμα συμπεράσματα για το υπό εξέταση σύστημα.

Όπως όλα τα επιμέρους λειτουργικά συστατικά του TPCDS benchmark, έτσι και τα TPCDS queries, δεν είναι σχεδιασμένα για ένα συγκεκριμένο σύστημα διαχείρισης βάσεων δεδομένων, ούτε καν για μία συγκεκριμένη γλώσσα βάσεων δεδομένων (πχ SQL). Αντίθετα, διατίθενται υπό μορφή templates που ο χρήστης του benchmark οφείλει να μεταφράσει σε πραγματικά ερωτήματα, γραμμένα στην γλώσσα βάσεων δεδομένων που θα επιλέξει. Στα πλαίσια της εργασίας, τα templates που χρησιμοποιήθηκαν μεταφράστηκαν με τέτοιο τρόπο, ώστε τα αντίστοιχα queries είναι συμβατά με την διάλεκτο SQL που χρησιμοποιεί το σύστημα διαχείρισης βάσεων δεδομένων που χρησιμοποιήθηκε (**MySQL Server**).

Κάθε ένα από τα TPCDS queries έχει μια σειρά από χαρακτηριστικά, που όπως έχει ήδη αναφερθεί, μπορούν να το εντάξουν στις εξής κατηγορίες: Reporting queries, Ad-hoc queries, Iterative OLAP queries, Data mining queries. Σε πολλές περιπτώσεις, η κατηγοριοποίηση των TPCDS queries δεν είναι απόλυτη, με την έννοια ότι κάθε query μπορεί να έχει χαρακτηριστικά ικανά να το εντάξουν σε περισσότερες από μια κατηγορίες.

Στα πλαίσια αυτής της εργασίας χρησιμοποιήθηκε μόνο ένα υποσύνολο των TPCDS queries. Για την ακρίβεια, **επιλέχτηκαν εκείνα τα queries στα οποία υπάρχει εμπλοκή των booleanized attributes** της TPCDS βάσης δεδομένων. Καθένα από τα επιλεγμένα queries, αντιπροσωπεύει όλα τα queries με παρεμφερή δομικά χαρακτηριστικά, που θα μπορούσαν να υποβληθούν στη βάση δεδομένων.

Τα queries που επιλέχθηκαν, υποβλήθηκαν επαναληπτικά τόσο στην TPCDS βάση δεδομένων όσο και στην booleanized TPCDS βάση δεδομένων (αφού πρώτα πραγματοποιηθούν οι κατάλληλοι μετασχηματισμοί). Με τη μέθοδο αυτή, παρατηρήθηκε κατά πόσο και σε ποιο βαθμό διαφοροποιείται η απόδοση της εκτέλεσης των ερωτημάτων στις δυο βάσεις. Εφόσον η όλη διαδικασία πραγματοποιήθηκε κάτω από τις ίδιες ακριβώς συνθήκες για τις δυο βάσεις, είναι σαφές ότι οποιαδήποτε διαφορά στην απόδοση οφείλεται στην εφαρμογή του Boolean approach στην TPCDS βάση δεδομένων.

Το βασικό χαρακτηριστικό ως προς το οποίο συγκρίνονται οι δυο υλοποιήσεις της TPCDS βάσης δεδομένων (basic και booleanized) είναι ο χρόνος εκτέλεσης των queries. Η διαφορά απόδοσης μεταξύ τους, καταδεικνύεται χρησιμοποιώντας ως μέτρο την ποσοστιαία χρονική απόκλιση της εκτελέσεων κάθε query στις δυο διαφορετικές υλοποιήσεις.

Η χρήση ποσοστών και όχι απόλυτων τιμών, γίνεται για δυο λόγους:

- προκειμένου τα αποτελέσματα του πειράματος να είναι ποιοτικά, και να μπορούν να γενικευτούν σε βάσεις δεδομένων διαφόρων μεγεθών.

- προκειμένου να είναι πιο εύκολα συγκρίσιμη η επίδραση που έχει η εφαρμογή του boolean approach από query σε query.

Ο υπολογισμός της ποσοστιαίας χρονικής απόκλισης των δύο υλοποιήσεων, σε κάθε επανάληψη κάθε query γίνεται με βάση τον ακόλουθο τύπο :

Ποσοστιαία Χρονική Απόκλιση (ATD%)

- Query: q

- Loop number: j

- Χρόνος εκτέλεσης στην βασική tpcds ΒΔ: t(q,L)

- Χρόνος εκτέλεσης στην booleanized tpcds ΒΔ: bt(q,L)

$$ATD(q,j) = \left(\frac{bt(q,j) - t(q,j)}{t(q,j)}\right) * 100$$

Από την μορφή του παραπάνω τύπου προκύπτουν τα ακόλουθα:

- Αρνητική τιμή του $ATD(q,j)$σημαίνει ότι για τη για την επανάληψη j του query q, η booleanized ΒΔ υπερτερεί της βασικής ΒΔ.
- Θετική τιμή του $ATD(q,j)$σημαίνει ότι για τη για την επανάληψη j του query q, η booleanized ΒΔ υστερεί της βασικής ΒΔ.
- Όσο μεγαλύτερη είναι κατά απόλυτη τιμή το $ATD(q,j)$, τόσο μεγαλύτερη είναι η διαφορά στο χρόνο εκτέλεσης μεταξύ booleanized ΒΔ και βασικής ΒΔ για την εκτέλεση j του query q.

Στην παρακάτω καταγραφή των TPCDS queries που χρησιμοποιήθηκαν, για κάθε TPCDS query συμπεριλαμβάνονται τα εξής:

1. Το **βασικό query σε γλώσσα SQL**, όπως υποβλήθηκε στην TPCDS βάση δεδομένων.
2. Το **booleanized query σε γλώσσα SQL**, όπως υποβλήθηκε στην booleanized TPCDS βάση δεδομένων.
3. Συνοπτική **περιγραφή των SQL μετασχηματισμών** που πραγματοποιήθηκαν, ώστε το αρχικό query, που απευθύνεται στην TPCDS βάση δεδομένων, να μεταφραστεί στο booleanized query που απευθύνεται στην booleanized TPCDS βάση δεδομένων.
4. **Διαγράμματα ποσοστιαίας χρονικής απόκλισης** ανά επανάληψη εκτέλεσης του query και στις δυο υλοποιήσεις της TPCDS βάσης δεδομένων, και συνοπτικός σχολιασμός.

QUERY 1

Basic SQL

query1.sql Raw

```sql
select d_week_seq1,round(sun_sales1/sun_sales2,2)
               ,round(mon_sales1/mon_sales2,2),round(tue_sales1/tue_sales2,2)
             ,round(wed_sales1/wed_sales2,2),round(thu_sales1/thu_sales2,2)
               ,round(fri_sales1/fri_sales2,2),round(sat_sales1/sat_sales2,2)

from
               (select wswscs.d_week_seq d_week_seq1
        ,sun_sales sun_sales1
        ,mon_sales mon_sales1
        ,tue_sales tue_sales1
        ,wed_sales wed_sales1
        ,thu_sales thu_sales1
        ,fri_sales fri_sales1
        ,sat_sales sat_sales1
                from (select d_week_seq,
        sum(case when (d_day_name='Sunday') then sales_price else null end) as sun_sales,
        sum(case when (d_day_name='Monday') then sales_price else null end) as mon_sales,
        sum(case when (d_day_name='Tuesday') then sales_price else  null end) as tue_sales,
        sum(case when (d_day_name='Wednesday') then sales_price else null end) as wed_sales,
        sum(case when (d_day_name='Thursday') then sales_price else null end) as thu_sales,
        sum(case when (d_day_name='Friday') then sales_price else null end) as fri_sales,
        sum(case when (d_day_name='Saturday') then sales_price else null end) as sat_sales
 from (select sold_date_sk
        ,sales_price
  from (select ws_sold_date_sk sold_date_sk
              ,ws_ext_sales_price sales_price
        from web_sales) x
        union all
       (select cs_sold_date_sk sold_date_sk
              ,cs_ext_sales_price sales_price
        from catalog_sales)) as wscs
    ,date_dim
 where d_date_sk = sold_date_sk
 group by d_week_seq) as wswscs,date_dim
               where date_dim.d_week_seq = wswscs.d_week_seq
               and d_year = 2000) y,

               (select wswscs.d_week_seq d_week_seq2
        ,sun_sales sun_sales2
        ,mon_sales mon_sales2
        ,tue_sales tue_sales2
        ,wed_sales wed_sales2
        ,thu_sales thu_sales2
        ,fri_sales fri_sales2
        ,sat_sales sat_sales2
                from (select d_week_seq,
        sum(case when (d_day_name='Sunday') then sales_price else null end) as sun_sales,
        sum(case when (d_day_name='Monday') then sales_price else null end) as mon_sales,
        sum(case when (d_day_name='Tuesday') then sales_price else  null end) as tue_sales,
        sum(case when (d_day_name='Wednesday') then sales_price else null end) as wed_sales,
        sum(case when (d_day_name='Thursday') then sales_price else null end) as thu_sales,
        sum(case when (d_day_name='Friday') then sales_price else null end) as fri_sales,
        sum(case when (d_day_name='Saturday') then sales_price else null end) as sat_sales
 from (select sold_date_sk
        ,sales_price
  from (select ws_sold_date_sk sold_date_sk
              ,ws_ext_sales_price sales_price
        from web_sales) x
        union all
       (select cs_sold_date_sk sold_date_sk
              ,cs_ext_sales_price sales_price
        from catalog_sales)) as wscs
    ,date_dim
 where d_date_sk = sold_date_sk
 group by d_week_seq) as wswscs,date_dim
               where date_dim.d_week_seq = wswscs.d_week_seq
               and     d_year = 2001) z

where d_week_seq1 < d_week_seq2
order by d_week_seq1;
```

Booleanized SQL

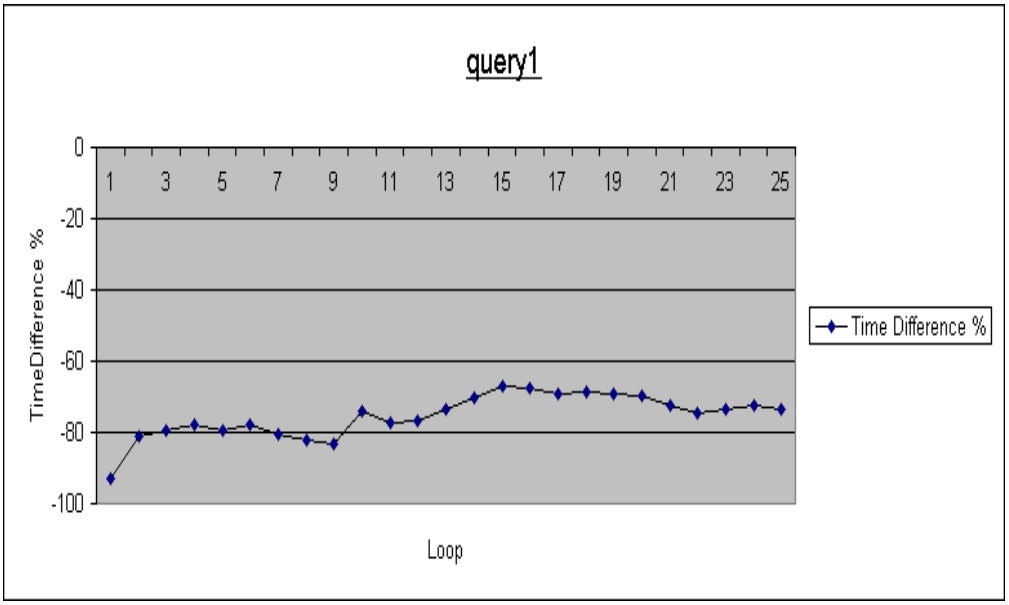
bquery1.sql Raw

```
select d_week_seq1,round(sun_sales1/sun_sales2,2)
                      ,round(mon_sales1/mon_sales2,2),round(tue_sales1/tue_sales2,2)
                         ,round(wed_sales1/wed_sales2,2),round(thu_sales1/thu_sales2,2)
                      ,round(fri_sales1/fri_sales2,2),round(sat_sales1/sat_sales2,2)

                    from
                      (select wswscs.d_week_seq d_week_seq1
                            ,sun_sales sun_sales1
                            ,mon_sales mon_sales1
                            ,tue_sales tue_sales1
                            ,wed_sales wed_sales1
                            ,thu_sales thu_sales1
                            ,fri_sales fri_sales1
                            ,sat_sales sat_sales1
                       from (select d_week_seq,
                            sum(case when (d_day_name_Sunday=1) then sales_price else null end)
                            as sun_sales,
                            sum(case when (d_day_name_Monday=1) then sales_price else null end)
                            as mon_sales,
                            sum(case when (d_day_name_Tuesday=1) then sales_price else  null end)
                            as tue_sales,
                            sum(case when (d_day_name_Wednesday=1) then sales_price else null end)
                            as wed_sales,
                            sum(case when (d_day_name_Thursday=1) then sales_price else null end)
                            as thu_sales,
                            sum(case when (d_day_name_Friday=1) then sales_price else null end)
                            as fri_sales,
                            sum(case when (d_day_name_Saturday=1) then sales_price else null end)
                            as sat_sales
                     from (select sold_date_sk
                            ,sales_price
                      from (select ws_sold_date_sk sold_date_sk
                                  ,ws_ext_sales_price sales_price
                            from web_sales) x
                            union all
                           (select cs_sold_date_sk sold_date_sk
                                  ,cs_ext_sales_price sales_price
                            from catalog_sales)) as wscs
                        ,date_dim
                     where d_date_sk = sold_date_sk
                     group by d_week_seq) as wswscs,date_dim
                      where date_dim.d_week_seq = wswscs.d_week_seq
                      and d_year = 2000) y,

                      (select wswscs.d_week_seq d_week_seq2
                            ,sun_sales sun_sales2
                            ,mon_sales mon_sales2
                            ,tue_sales tue_sales2
                            ,wed_sales wed_sales2
                            ,thu_sales thu_sales2
                            ,fri_sales fri_sales2
                            ,sat_sales sat_sales2
                       from (select d_week_seq,
                            sum(case when (d_day_name_Sunday=1) then sales_price else null end)
                            as sun_sales,
                            sum(case when (d_day_name_Monday=1) then sales_price else null end)
                            as mon_sales,
                            sum(case when (d_day_name_Tuesday=1) then sales_price else  null end)
                            as tue_sales,
                            sum(case when (d_day_name_Wednesday=1) then sales_price else null end)
                            as wed_sales,
                            sum(case when (d_day_name_Thursday=1) then sales_price else null end)
                            as thu_sales,
                            sum(case when (d_day_name_Friday=1) then sales_price else null end)
                            as fri_sales,
                            sum(case when (d_day_name_Saturday=1) then sales_price else null end)
                            as sat_sales
                     from (select sold_date_sk
                            ,sales_price
                      from (select ws_sold_date_sk sold_date_sk
                                  ,ws_ext_sales_price sales_price
                            from web_sales) x
                            union all
                           (select cs_sold_date_sk sold_date_sk
                                  ,cs_ext_sales_price sales_price
                            from catalog_sales)) as wscs
                        ,date_dim
                     where d_date_sk = sold_date_sk
                     group by d_week_seq) as wswscs,date_dim
                      where date_dim.d_week_seq = wswscs.d_week_seq
                      and d_year = 2001) z

                    where d_week_seq1 < d_week_seq2
                    order by d_week_seq1;  ;
```

Μετατροπές SQL

d_day_name: d_day_name = 'Value' <-> d_day_name_Value = 1

Διάγραμμα Ποσοστιαίας Χρονικής Απόκλισης



Σχόλια

Όπως φαίνεται και στο διάγραμμα, σε όλες τις επαναλήψεις εκτέλεσης του query ο χρόνος εκτέλεσης στην booleanized βάση δεδομένων είναι αισθητά μικρότερος.

Είναι σαφές ότι η εφαρμογή του μετασχηματισμού boolean approach επιδρά θετικά σε ερωτήματα τέτοιου είδους. Μπορεί ο μετασχηματισμός να εφαρμόστηκε σε ένα μόνο attribute, ωστόσο αυτό εμπλέκεται σε πολλά εμφωλευμένα queries, και μάλιστα με όλες τις δυνατές τιμές του. Το γεγονός αυτό αυξάνει, τόσο τη συνολική πολυπλοκότητα του query όσο και τον βαθμό επίδρασης του attribute στην εκτέλεση του query.

QUERY 2

Basic SQL

query2.sql Raw

```sql
 SELECT a.ca_state AS state,
       Count(*)   AS cnt
FROM   customer_address a,
       customer c,
       store_sales s,
       date_dim d,
       item i
WHERE  a.ca_address_sk = c.c_current_addr_sk
       AND c.c_customer_sk = s.ss_customer_sk
       AND s.ss_sold_date_sk = d.d_date_sk
       AND s.ss_item_sk = i.i_item_sk
       AND d.d_month_seq = (SELECT DISTINCT ( d_month_seq )
                            FROM   date_dim
                            WHERE  d_year = 2000
                                   AND d_month_name = 'November'
                            LIMIT  1)
       AND i.i_current_price > 1.2 * (SELECT Avg(j.i_current_price)
                                      FROM   item j
                                      WHERE  j.i_category = i.i_category)
GROUP  BY a.ca_state
HAVING Count(*) >= 10
ORDER  BY cnt;
```

Booleanized SQL

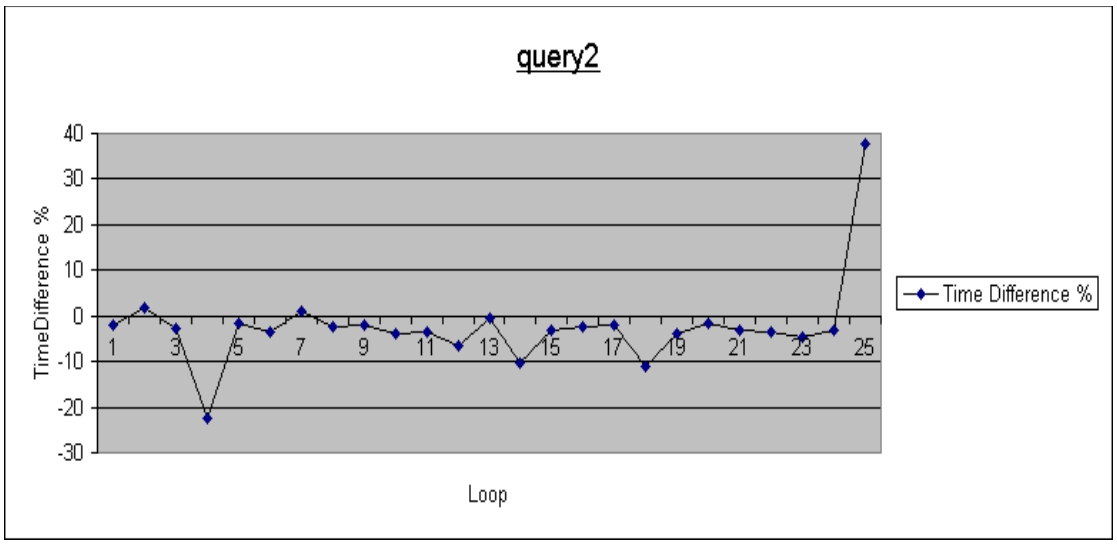
bquery2.sql Raw

```sql
 SELECT a.ca_state AS state,
       Count(*)   AS cnt
FROM   customer_address a,
       customer c,
       store_sales s,
       date_dim d,
       item i
WHERE  a.ca_address_sk = c.c_current_addr_sk
       AND c.c_customer_sk = s.ss_customer_sk
       AND s.ss_sold_date_sk = d.d_date_sk
       AND s.ss_item_sk = i.i_item_sk
       AND d.d_month_seq = (SELECT DISTINCT ( d_month_seq )
                            FROM   date_dim
                            WHERE  d_year = 2000
                                   AND d_month_name_november = 1
                            LIMIT  1)
       AND i.i_current_price > 1.2 * (SELECT Avg(j.i_current_price)
                                      FROM   item j
                                      WHERE  j.i_category = i.i_category)
GROUP  BY a.ca_state
HAVING Count(*) >= 10
ORDER  BY cnt;
```

Μετατροπές SQL

d_day_name: d_day_name = 'Value' <-> d_day_name_Value = 1

Διάγραμμα Ποσοστιαίας Χρονικής Απόκλισης



Σχόλια

Ο χρόνος εκτέλεσης του query είναι και εδώ μικρότερος στις περισσότερες επαναλήψεις. Η διαφορά ωστόσο είναι αισθητά μικρότερη σε σχέση με το πρώτο query. Αυτό συμβαίνει γιατί αν και μετασχηματίστηκε το ίδιο attribute, στο query αυτό η εμπλοκή του είναι πολύ μικρότερη, κι έτσι επιδρά πολύ λιγότερο στην εκτέλεση του query.

QUERY 3

Basic SQL

query3.sql Raw

```sql
SELECT i_item_id,
       Avg(ss_quantity)    AS agg1,
       Avg(ss_list_price)  AS agg2,
       Avg(ss_coupon_amt)  AS agg3,
       Avg(ss_sales_price) AS agg4
FROM   store_sales,
       customer_demographics,
       date_dim,
       item,
       promotion
WHERE  ss_sold_date_sk = d_date_sk
       AND ss_item_sk = i_item_sk
       AND ss_cdemo_sk = cd_demo_sk
       AND ss_promo_sk = p_promo_sk
       AND cd_gender = 'female'
       AND cd_marital_status = 'married'
       AND cd_education_status = 'university'
       AND ( p_channel_email = 'no'
             OR p_channel_event = 'no' )
       AND d_year > 2000
GROUP  BY i_item_id
ORDER  BY i_item_id;
```

Booleanized SQL

bquery3.sql Raw

```sql
SELECT i_item_id,
       Avg(ss_quantity)    AS agg1,
       Avg(ss_list_price)  AS agg2,
       Avg(ss_coupon_amt)  AS agg3,
       Avg(ss_sales_price) AS agg4
FROM   store_sales,
       customer_demographics,
       date_dim,
       item,
       promotion
WHERE  ss_sold_date_sk = d_date_sk
       AND ss_item_sk = i_item_sk
       AND ss_cdemo_sk = cd_demo_sk
       AND ss_promo_sk = p_promo_sk
       AND cd_gender_female = 1
       AND cd_marital_status_married = 1
       AND cd_education_status_university = 1
       AND ( p_channel_email_no = 1
             OR p_channel_event_no = 1 )
       AND d_year > 2000
GROUP  BY i_item_id
ORDER  BY i_item_id;
```

Μετατροπές SQL

cd_gender: cd_gender = ‘Value’ <-> cd_gender_Value = 1

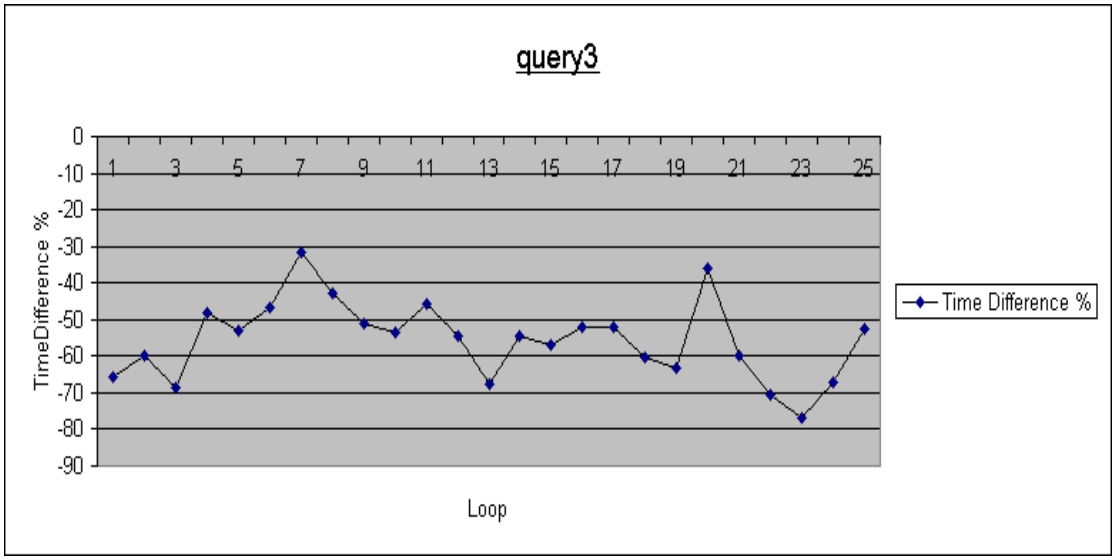
cd_marital_status: cd_marital_status = 'Value' <-> cd_marital_status_Value = 1

cd_education_status: cd_education_status = 'Value' <-> cd_education_status_Value = 1

p_channel_email: p_channel_email = 'Value' <-> p_channel_email_Value = 1

p_channel_event: p_channel_event = 'Value' <-> p_channel_event_Value = 1

Διάγραμμα Ποσοστιαίας Χρονικής Απόκλισης



Σχόλια

Σε όλες τις επαναλήψεις, η εκτέλεση του query στην booleanized βάση δεδομένων απαιτεί αισθητά λιγότερο χρόνο. Φαίνεται ότι και σε αυτή την περίπτωση ο μετασχηματισμός boolean approach έχει θετική επίδραση στον χρόνο εκτέλεσης του query.

Αυτή τη φορά, έχουν μετασχηματιστεί πέντε διαφορετικά attributes, από δυο πίνακες, τα οποία μάλιστα εμπλέκονται σε εμφωλευμένα queries. Όλα τα παραπάνω αυξάνουν την επίδραση που έχει ο μετασχηματισμός στον χρόνο εκτέλεσης και δικαιολογούν την μορφή του διαγράμματος.

QUERY 4

Basic SQL

query4.sql Raw

```sql
SELECT cd_purchase_estimate,
       cd_credit_rating,
       cd_dep_count,
       cd_dep_employed_count,
       cd_dep_college_count
FROM   customer c,
       customer_address ca,
       customer_demographics
WHERE  c.c_current_addr_sk = ca.ca_address_sk
       AND cd_demo_sk = c.c_current_cdemo_sk
       AND EXISTS (SELECT *
                   FROM   store_sales,
                          date_dim
                   WHERE  c.c_customer_sk = ss_customer_sk
                          AND ss_sold_date_sk = d_date_sk
                          AND d_year > 2000
                          AND d_month_name IN ( 'March', 'April', 'May', 'June',
                                                'July', 'August' ))
       AND ( EXISTS (SELECT *
                     FROM   web_sales,
                            date_dim
                     WHERE  c.c_customer_sk = ws_bill_customer_sk
                            AND ws_sold_date_sk = d_date_sk
                            AND d_year > 2000
                            AND ( d_month_name = 'March'
                                  OR d_month_name = 'April'
                                  OR d_month_name = 'May'
                                  OR d_month_name = 'June'
                                  OR d_month_name = 'July'
                                  OR d_month_name = 'August' ))
             OR EXISTS (SELECT *
                        FROM   catalog_sales,
                               date_dim
                        WHERE  c.c_customer_sk = cs_ship_customer_sk
                               AND cs_sold_date_sk = d_date_sk
                               AND d_year > 2000
                               AND ( d_month_name = 'March'
                                     OR d_month_name = 'April'
                                     OR d_month_name = 'May'
                                     OR d_month_name = 'June'
                                     OR d_month_name = 'July'
                                     OR d_month_name = 'August' )) )
GROUP  BY cd_purchase_estimate,
          cd_credit_rating,
          cd_dep_count,
          cd_dep_employed_count,
          cd_dep_college_count
ORDER  BY cd_purchase_estimate,
          cd_credit_rating,
          cd_dep_count,
          cd_dep_employed_count,
          cd_dep_college_count;
```

Booleanized SQL

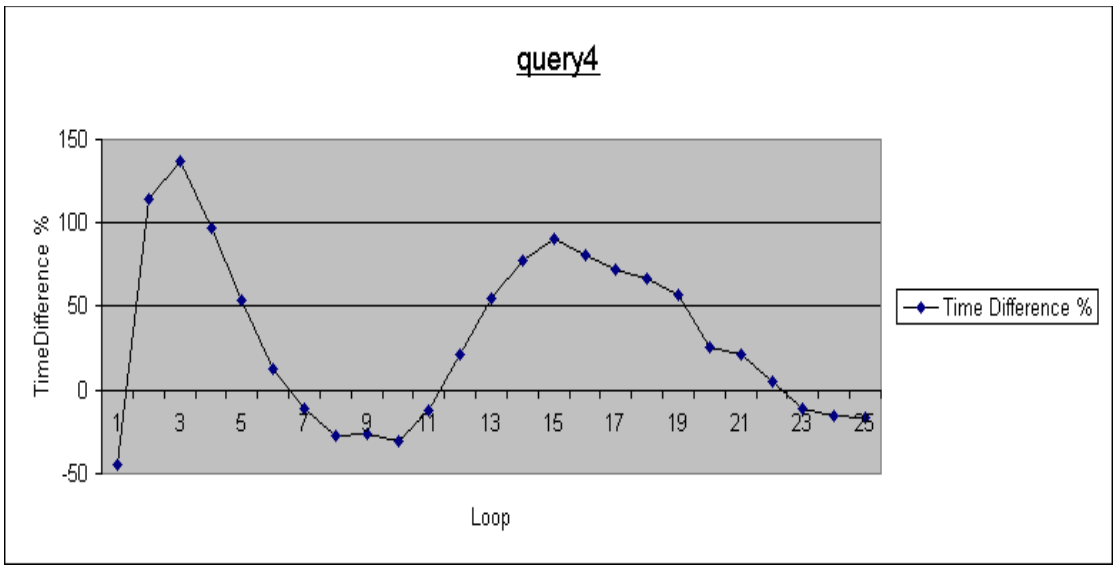
bquery4.sql Raw

```
 SELECT cd_purchase_estimate,
       cd_credit_rating,
       cd_dep_count,
       cd_dep_employed_count,
       cd_dep_college_count
FROM   customer c,
       customer_address ca,
       customer_demographics
WHERE  c.c_current_addr_sk = ca.ca_address_sk
       AND cd_demo_sk = c.c_current_cdemo_sk
       AND EXISTS (SELECT *
                   FROM   store_sales,
                          date_dim
                   WHERE  c.c_customer_sk = ss_customer_sk
                          AND ss_sold_date_sk = d_date_sk
                          AND d_year > 2000
                          AND ( d_month_name_March=1
                                OR d_month_name_April=1
                                OR d_month_name_May=1
                                OR d_month_name_June=1
                                OR d_month_name_July=1
                                OR d_month_name_August=1 ))
       AND ( EXISTS (SELECT *
                     FROM   web_sales,
                            date_dim
                     WHERE  c.c_customer_sk = ws_bill_customer_sk
                            AND ws_sold_date_sk = d_date_sk
                            AND d_year > 2000
                            AND (  d_month_name_March=1
                                 OR d_month_name_April=1
                                 OR d_month_name_May=1
                                 OR d_month_name_June=1
                                 OR d_month_name_July=1
                                 OR d_month_name_August=1 ))
             OR EXISTS (SELECT *
                        FROM   catalog_sales,
                               date_dim
                        WHERE  c.c_customer_sk = cs_ship_customer_sk
                               AND cs_sold_date_sk = d_date_sk
                               AND d_year > 2000
                               AND (  d_month_name_March=1
                                OR d_month_name_April=1
                                OR d_month_name_May=1
                                OR d_month_name_June=1
                                OR d_month_name_July=1
                                OR d_month_name_August=1  )) )
GROUP  BY cd_purchase_estimate,
          cd_credit_rating,
          cd_dep_count,
          cd_dep_employed_count,
          cd_dep_college_count
ORDER  BY cd_purchase_estimate,
          cd_credit_rating,
          cd_dep_count,
          cd_dep_employed_count,
          cd_dep_college_count;
```

Μετατροπές SQL

d_month_name: d_month_name = 'Value' <-> d_month_name_Value = 1

Διάγραμμα Ποσοστιαίας Χρονικής Απόκλισης



Σχόλια

Στην περίπτωση αυτή, όπως φαίνεται και στο διάγραμμα, η συμπεριφορά του χρόνου εκτέλεσης είναι αρκετά άναρχη, με την ποσοστιαία χρονική απόκλιση να εναλλάσσεται μεταξύ θετικών και αρνητικών τιμών. Συνεπώς δεν φαίνεται να υπάρχει άμεση επίδραση του μετασχηματισμού boolean approach στην επίδοση του συγκεκριμένου query.

QUERY 5

Basic SQL

query5.sql Raw

```sql
SELECT cd_gender,
       cd_marital_status,
       cd_education_status,
       cd_purchase_estimate,
       cd_credit_rating,
       cd_dep_count,
       cd_dep_employed_count,
       cd_dep_college_count
FROM   customer c,
       customer_address ca,
       customer_demographics
WHERE  c.c_current_addr_sk = ca.ca_address_sk
       AND cd_demo_sk = c.c_current_cdemo_sk
       AND EXISTS (SELECT *
                   FROM   store_sales,
                          date_dim
                   WHERE  c.c_customer_sk = ss_customer_sk
                          AND ss_sold_date_sk = d_date_sk
                          AND d_year > 2000
                          AND  d_month_name IN ('March','April','May','June','July','August') )
       AND ( EXISTS (SELECT *
                     FROM   web_sales,
                            date_dim
                     WHERE  c.c_customer_sk = ws_bill_customer_sk
                            AND ws_sold_date_sk = d_date_sk
                            AND d_year > 2000
                            AND ( d_month_name = 'March'
                                  OR d_month_name = 'April'
                                  OR d_month_name = 'May'
                                  OR d_month_name = 'June'
                                  OR d_month_name = 'July'
                                  OR d_month_name = 'August' ))
             OR EXISTS (SELECT *
                        FROM   catalog_sales,
                               date_dim
                        WHERE  c.c_customer_sk = cs_ship_customer_sk
                               AND cs_sold_date_sk = d_date_sk
                               AND d_year > 2000
                               AND ( d_month_name = 'March'
                                     OR d_month_name = 'April'
                                     OR d_month_name = 'May'
                                     OR d_month_name = 'June'
                                     OR d_month_name = 'July'
                                     OR d_month_name = 'August' )) )
GROUP  BY cd_gender,
          cd_marital_status,
          cd_education_status,
          cd_purchase_estimate,
          cd_credit_rating,
          cd_dep_count,
          cd_dep_employed_count,
          cd_dep_college_count
ORDER  BY cd_gender,
          cd_marital_status,
          cd_education_status,
          cd_purchase_estimate,
          cd_credit_rating,
          cd_dep_count,
          cd_dep_employed_count,
          cd_dep_college_count;
```

Booleanized SQL

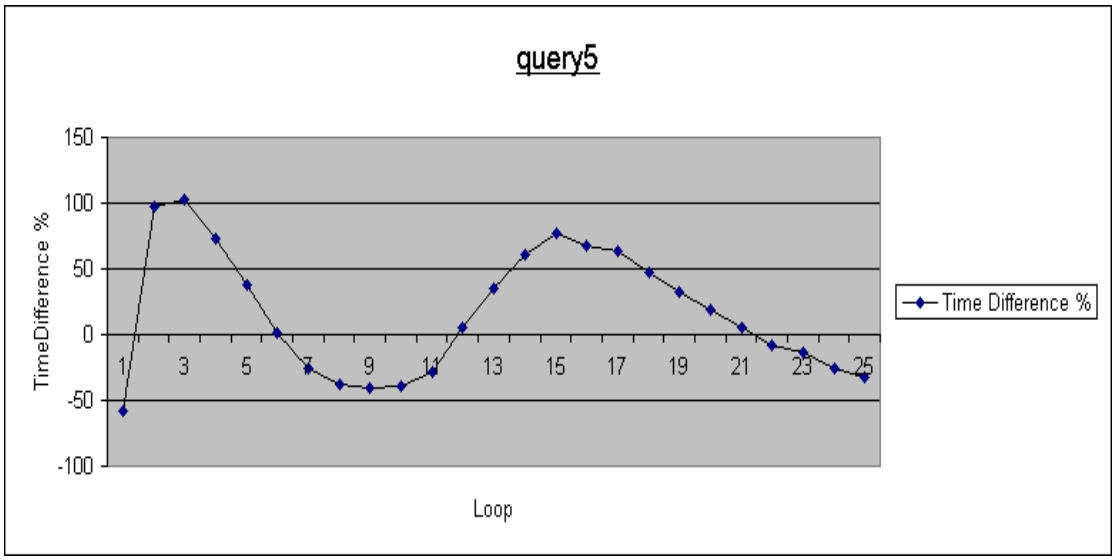
bquery5.sql Raw

```
SELECT cd_gender_male,cd_gender_female,
       cd_marital_status_single,cd_marital_status_married,cd_marital_status_divorced,
       cd_education_status_primary,cd_education_status_secondary,
           cd_education_status_university,cd_education_status_postgraduate,
       cd_purchase_estimate,
       cd_credit_rating,
       cd_dep_count,
       cd_dep_employed_count,
       cd_dep_college_count
FROM   customer c,
       customer_address ca,
       customer_demographics
WHERE  c.c_current_addr_sk = ca.ca_address_sk
       AND cd_demo_sk = c.c_current_cdemo_sk
       AND EXISTS (SELECT *
                   FROM   store_sales,
                          date_dim
                   WHERE  c.c_customer_sk = ss_customer_sk
                          AND ss_sold_date_sk = d_date_sk
                          AND d_year > 2000
                          AND ( d_month_name_March=1
                                OR d_month_name_April=1
                                OR d_month_name_May=1
                                OR d_month_name_June=1
                                OR d_month_name_July=1
                                OR d_month_name_August=1   ))
       AND ( EXISTS (SELECT *
                     FROM   web_sales,
                            date_dim
                     WHERE  c.c_customer_sk = ws_bill_customer_sk
                            AND ws_sold_date_sk = d_date_sk
                            AND d_year > 2000
                            AND ( d_month_name_March=1
                                  OR d_month_name_April=1
                                  OR d_month_name_May=1
                                  OR d_month_name_June=1
                                  OR d_month_name_July=1
                                  OR d_month_name_August=1   ))
             OR EXISTS (SELECT *
                        FROM   catalog_sales,
                               date_dim
                        WHERE  c.c_customer_sk = cs_ship_customer_sk
                               AND cs_sold_date_sk = d_date_sk
                               AND d_year > 2000
                               AND ( d_month_name_March=1
                                     OR d_month_name_April=1
                                     OR d_month_name_May=1
                                     OR d_month_name_June=1
                                     OR d_month_name_July=1
                                     OR d_month_name_August=1   )) )
GROUP  BY cd_gender_male,cd_gender_female,
          cd_marital_status_single,cd_marital_status_married,cd_marital_status_divorced,
          cd_education_status_primary,cd_education_status_secondary,
          cd_education_status_university,cd_education_status_postgraduate,
          cd_purchase_estimate,
          cd_credit_rating,
          cd_dep_count,
          cd_dep_employed_count,
          cd_dep_college_count
ORDER  BY cd_gender_male,cd_gender_female,
          cd_marital_status_single,cd_marital_status_married,cd_marital_status_divorced,
          cd_education_status_primary,cd_education_status_secondary,
          cd_education_status_university,cd_education_status_postgraduate,
          cd_purchase_estimate,
          cd_credit_rating,
          cd_dep_count,
          cd_dep_employed_count,
          cd_dep_college_count;
```

Μετατροπές SQL

d_month_name:	d_month_name = 'Value' <-> d_month_name_Value = 1
cd_gender:	cd_gender = = 'Value' <-> cd_gender_Value = 1
cd_marital_status:	cd_marital_status = 'Value' <-> cd_marital_status_Value = 1
cd_education_status:	cd_education_status = 'Value' <-> cd_education_status_Value = 1

Διάγραμμα Ποσοστιαίας Χρονικής Απόκλισης



Σχόλια

Στην περίπτωση αυτή, όπως φαίνεται και στο διάγραμμα, η συμπεριφορά του χρόνου εκτέλεσης είναι αρκετά άναρχη, με την ποσοστιαία χρονική απόκλιση να εναλλάσσεται μεταξύ θετικών και αρνητικών τιμών. Συνεπώς δεν φαίνεται να υπάρχει άμεση επίδραση του μετασχηματισμού boolean approach στην επίδοση του συγκεκριμένου query.

QUERY 6

Basic SQL

query6.sql Raw

```sql
SELECT c_customer_id                           customer_id,
       c_first_name                            customer_first_name,
       c_last_name                             customer_last_name,
       c_preferred_cust_flag                   customer_preferred_cust_flag
       ,
       c_birth_country
       customer_birth_country,
       c_login                                 customer_login,
       c_email_address                         customer_email_address,
       d_year                                  dyear,
       Sum(ss_ext_list_price - ss_ext_discount_amt) AS year_total,
       's'                                     sale_type
FROM   customer,
       store_sales,
       date_dim
WHERE  c_customer_sk = ss_customer_sk
       AND ss_sold_date_sk = d_date_sk
GROUP  BY c_customer_id,
          c_first_name,
          c_last_name,
          d_year,
          c_preferred_cust_flag,
          c_birth_country,
          c_login,
          c_email_address,
          d_year
UNION ALL
SELECT c_customer_id                           customer_id,
       c_first_name                            customer_first_name,
       c_last_name                             customer_last_name,
       c_preferred_cust_flag                   customer_preferred_cust_flag
       ,
       c_birth_country
       customer_birth_country,
       c_login                                 customer_login,
       c_email_address                         customer_email_address,
       d_year                                  dyear,
       Sum(ws_ext_list_price - ws_ext_discount_amt) AS year_total,
       'w'                                     sale_type
FROM   customer,
       web_sales,
       date_dim
WHERE  c_customer_sk = ws_bill_customer_sk
       AND ws_sold_date_sk = d_date_sk
GROUP  BY c_customer_id,
          c_first_name,
          c_last_name,
          c_preferred_cust_flag,
          c_birth_country,
          c_login,
          c_email_address,
          d_year;
```

Booleanized SQL

bquery6.sql Raw

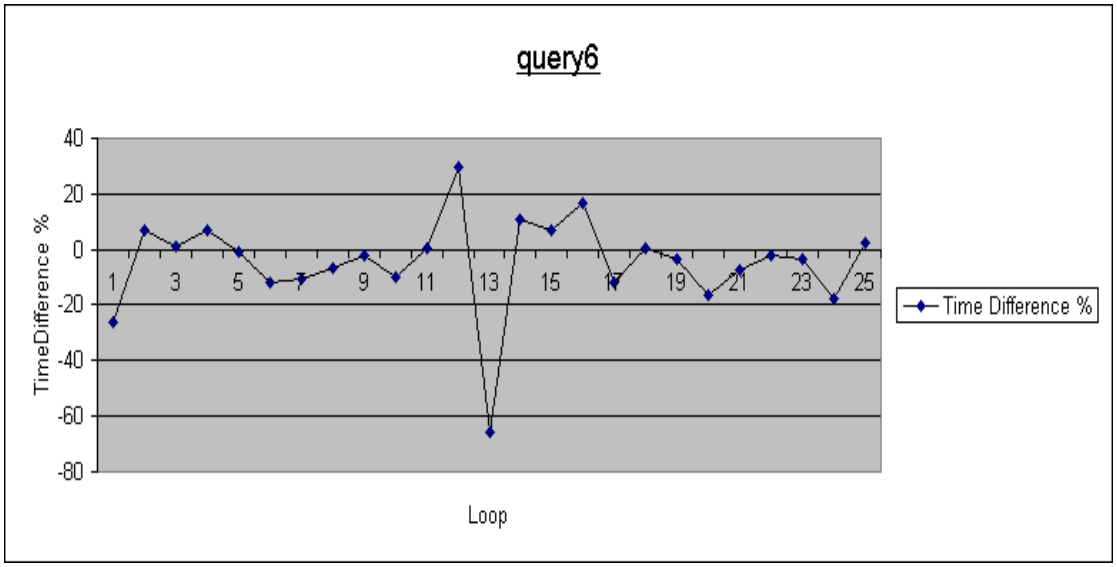
```
 SELECT c_customer_id                              customer_id,
       c_first_name                                customer_first_name,
       c_last_name                                 customer_last_name,
       c_preferred_cust_flag_yes
       customer_preferred_cust_flag_yes,
       c_preferred_cust_flag_no
       customer_preferred_cust_flag_no,
       c_birth_country                             customer_birth_country,
       c_login                                     customer_login,
       c_email_address                             customer_email_address,
       d_year                                      dyear,
       Sum(ss_ext_list_price - ss_ext_discount_amt) AS year_total,
       's'                                         sale_type
FROM   customer,
       store_sales,
       date_dim
WHERE  c_customer_sk = ss_customer_sk
       AND ss_sold_date_sk = d_date_sk
GROUP  BY c_customer_id,
          c_first_name,
          c_last_name,
          d_year,
          c_preferred_cust_flag_yes,
          c_preferred_cust_flag_no,
          c_birth_country,
          c_login,
          c_email_address,
          d_year
UNION ALL
SELECT c_customer_id                              customer_id,
       c_first_name                                customer_first_name,
       c_last_name                                 customer_last_name,
       c_preferred_cust_flag_yes,
       c_preferred_cust_flag_no                    customer_preferred_cust_flag
       ,
       c_birth_country
       customer_birth_country,
       c_login                                     customer_login,
       c_email_address                             customer_email_address,
       d_year                                      dyear,
       Sum(ws_ext_list_price - ws_ext_discount_amt) AS year_total,
       'w'                                         sale_type
FROM   customer,
       web_sales,
       date_dim
WHERE  c_customer_sk = ws_bill_customer_sk
       AND ws_sold_date_sk = d_date_sk
GROUP  BY c_customer_id,
          c_first_name,
          c_last_name,
          c_preferred_cust_flag_yes,
          c_preferred_cust_flag_no,
          c_birth_country,
          c_login,
          c_email_address,
          d_year;
```

Μετατροπές SQL

c_preferred_customer_flag:

c_preferred_customer_flag = 'Value' <-> c_preferred_customer_flag_Value = 1

Διάγραμμα Ποσοστιαίας Χρονικής Απόκλισης



Σχόλια

Στις περισσότερες επαναλήψεις ο χρόνος εκτέλεσης είναι οριακά μικρότερος στην booleanized βάση δεδομένων. Παρόλα αυτά οι χρονικές διαφορές είναι στη συντριπτική πλειοψηφία των περιπτώσεων οριακές και υπάρχουν αρκετές εναλλαγές στο πρόσημο, συνεπώς δεν φαίνεται να υπάρχει ουσιαστική επίδραση του boolean approach Μετασχηματισμού στο χρόνο εκτέλεσης του query.

QUERY 7

Basic SQL

query7.sql Raw

```sql
SELECT Avg(ss_quantity),
       Avg(ss_ext_sales_price),
       Avg(ss_ext_wholesale_cost),
       Sum(ss_ext_wholesale_cost)
FROM   store_sales,
       store,
       customer_demographics,
       household_demographics,
       customer_address,
       date_dim
WHERE  s_store_sk = ss_store_sk
       AND ss_sold_date_sk = d_date_sk
       AND d_year = 2001
       AND ( ( ss_hdemo_sk = hd_demo_sk
               AND cd_demo_sk = ss_cdemo_sk
               AND cd_marital_status = 'married'
               AND cd_education_status = 'university'
               AND ss_sales_price BETWEEN 10000.00 AND 50000.00
               AND hd_dep_count > 5 )
              OR ( ss_hdemo_sk = hd_demo_sk
                   AND cd_demo_sk = ss_cdemo_sk
                   AND cd_marital_status = 'divorced'
                   AND cd_education_status = 'secondary'
                   AND ss_sales_price BETWEEN 20000.00 AND 60000.00
                   AND hd_dep_count < 5 )
              OR ( ss_hdemo_sk = hd_demo_sk
                   AND cd_demo_sk = ss_cdemo_sk
                   AND cd_marital_status = 'single'
                   AND cd_education_status = 'postgraduate'
                   AND ss_sales_price BETWEEN 30000.00 AND 70000.00
                   AND hd_dep_count < 7 ) );
```

Booleanized SQL

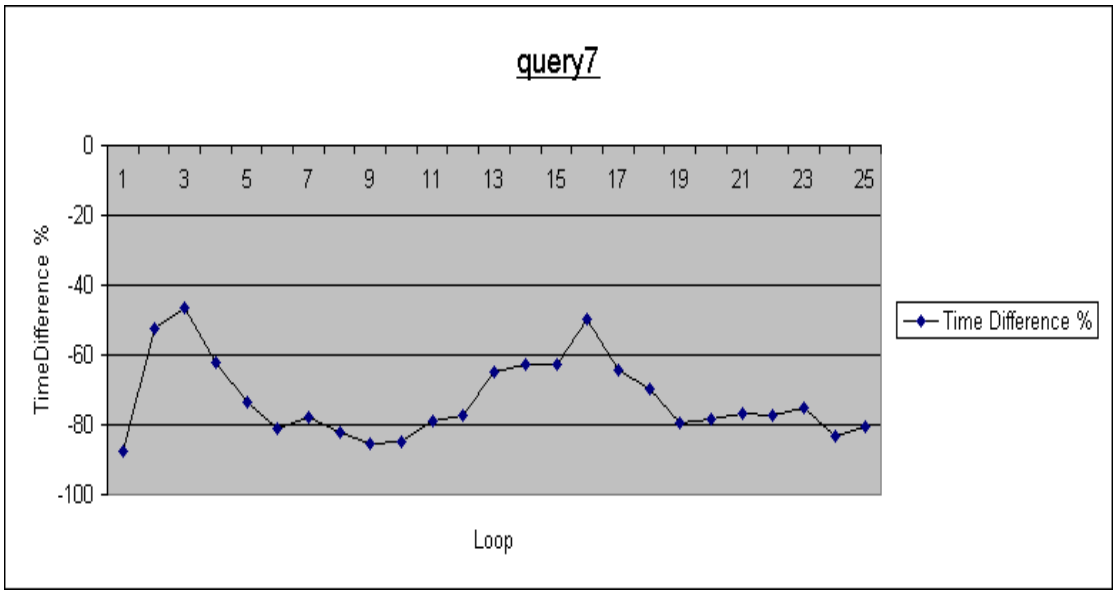
bquery7.sql Raw

```sql
SELECT Avg(ss_quantity),Avg(ss_ext_sales_price),Avg(ss_ext_wholesale_cost),Sum(
       ss_ext_wholesale_cost)
FROM   store_sales,store,customer_demographics,household_demographics,
       customer_address,
       date_dim
WHERE  s_store_sk = ss_store_sk
       AND ss_sold_date_sk = d_date_sk
       AND d_year = 2001
       AND ( ( ss_hdemo_sk = hd_demo_sk
               AND cd_demo_sk = ss_cdemo_sk
               AND cd_marital_status_married=1
               AND cd_education_status_university=1
               AND ss_sales_price BETWEEN 10000.00 AND 50000.00
               AND hd_dep_count > 5 )
              OR ( ss_hdemo_sk = hd_demo_sk
                   AND cd_demo_sk = ss_cdemo_sk
                   AND cd_marital_status_divorced=1
                   AND cd_education_status_secondary=1
                   AND ss_sales_price BETWEEN 20000.00 AND 60000.00
                   AND hd_dep_count < 5 )
              OR ( ss_hdemo_sk = hd_demo_sk
                   AND cd_demo_sk = ss_cdemo_sk
                   AND cd_marital_status_single=1
                   AND cd_education_status_postgraduate=1
                   AND ss_sales_price BETWEEN 30000.00 AND 70000.00
                   AND hd_dep_count < 7 ) );
```

Μετατροπές SQL

cd_gender:	cd_gender = 'Value' <-> cd_gender_Value = 1
cd_marital_status:	cd_marital_status = 'Value' <-> cd_marital_status_Value = 1
cd_education_status:	cd_education_status = 'Value' <-> cd_education_status_Value = 1

Διάγραμμα Ποσοστιαίας Χρονικής Απόκλισης



Σχόλια

Στην περίπτωση αυτή φαίνεται να υπάρχει μεγάλη χρονική διαφορά υπέρ της booleanized βάσης δεδομένων. Μετασχηματίστηκαν τρία attributes τα οποία συμμετέχουν σε πολλά εμφωλευμένα queries και αυτό είχε θεαματική επίδραση στο χρόνο εκτέλεσης του query.

QUERY 8

Basic SQL

query8.sql Raw

```sql
SELECT ca_zip,Sum(cs_sales_price)
FROM   catalog_sales,customer,customer_address,date_dim
WHERE  cs_bill_customer_sk = c_customer_sk
       AND c_current_addr_sk = ca_address_sk
       AND cs_sales_price > 500
       AND cs_sold_date_sk = d_date_sk
       AND d_quarter_name = 'first'
       AND d_year > 2000
GROUP  BY ca_zip
ORDER  BY ca_zip;
```

Booleanized SQL

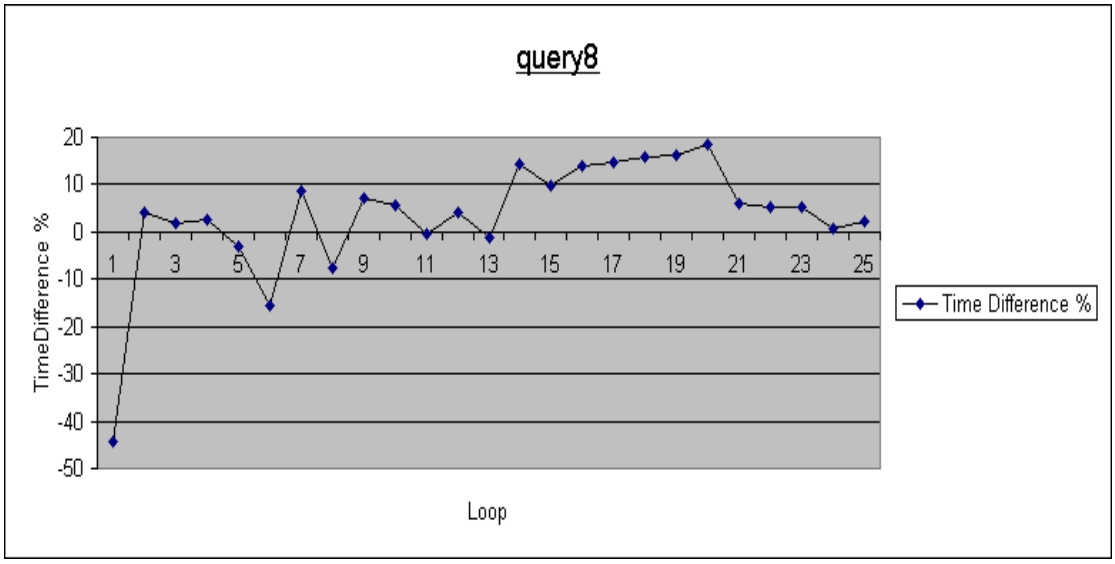
bquery8.sql Raw

```sql
SELECT ca_zip,Sum(cs_sales_price)
FROM   catalog_sales,customer,customer_address,date_dim
WHERE  cs_bill_customer_sk = c_customer_sk
       AND c_current_addr_sk = ca_address_sk
       AND cs_sales_price > 500
       AND cs_sold_date_sk = d_date_sk
       AND d_quarter_name_first=1
       AND d_year > 2000
GROUP  BY ca_zip
ORDER  BY ca_zip;
```

Μετατροπές SQL

d_quarter_name: d_quarter_name = 'Value' <-> d_quarter_name_Value = 1

Διάγραμμα Ποσοστιαίας Χρονικής Απόκλισης



Σχόλια

Στην περίπτωση αυτή παρουσιάζονται μικρές χρονικές διαφορές, (στην πλειοψηφία τους υπέρ της αρχικής βάσης δεδομένων) κι έτσι δεν φαίνεται ο μετασχηματισμός να είχε μεγάλη επίδραση στην εκτέλεση του query. Φυσικά αυτό οφείλεται και στο γεγονός ότι πρόκειται για ένα ιδιαίτερα απλό query, ενώ μετασχηματίστηκε μόλις ένα attribute και το οποίο εμπλέκεται μόνο σε ένα WHERE clause. Συνεπώς, η επίδραση του μετασχηματισμού boolean approach δεν αναμενόταν να είναι εντυπωσιακή.

QUERY 9

Basic SQL

query9.sql Raw

```sql
SELECT i_item_id,i_item_desc,s_state,
       Count(ss_quantity) AS store_sales_quantitycount,
       Avg(ss_quantity) AS store_sales_quantityave,
       Stddev_samp(ss_quantity) AS store_sales_quantitystdev,
       Stddev_samp(ss_quantity) / Avg(ss_quantity) AS store_sales_quantitycov,
       Count(sr_return_quantity) AS as_store_returns_quantitycount,Avg(
       sr_return_quantity) AS as_store_returns_quantityave,
       Stddev_samp(sr_return_quantity) AS as_store_returns_quantitystdev,
              Stddev_samp(sr_return_quantity) / Avg(sr_return_quantity) AS
              store_returns_quantitycov,
       Count(cs_quantity) AS catalog_sales_quantitycount,
       Avg(cs_quantity) AS catalog_sales_quantityave,
       Stddev_samp(cs_quantity) / Avg(cs_quantity) AS
       catalog_sales_quantitystdev,Stddev_samp(cs_quantity) / Avg(cs_quantity)
                                   AS catalog_sales_quantitycov
FROM   store_sales,store_returns,catalog_sales,date_dim d1,date_dim d2,date_dim
       d3,
       store,item
WHERE  d1.d_quarter_name = 'first'
       AND d1.d_date_sk = ss_sold_date_sk
       AND i_item_sk = ss_item_sk
       AND s_store_sk = ss_store_sk
       AND ss_customer_sk = sr_customer_sk
       AND ss_item_sk = sr_item_sk
       AND ss_ticket_number = sr_ticket_number
       AND sr_returned_date_sk = d2.d_date_sk
       AND d2.d_quarter_name IN ( 'first', 'second', 'third' )
       AND sr_customer_sk = cs_bill_customer_sk
       AND sr_item_sk = cs_item_sk
       AND cs_sold_date_sk = d3.d_date_sk
       AND d3.d_quarter_name IN ( 'first', 'second', 'third' )
GROUP  BY i_item_id,i_item_desc,s_state
ORDER  BY i_item_id,i_item_desc,s_state;
```

Booleanized SQL

bquery9.sql Raw

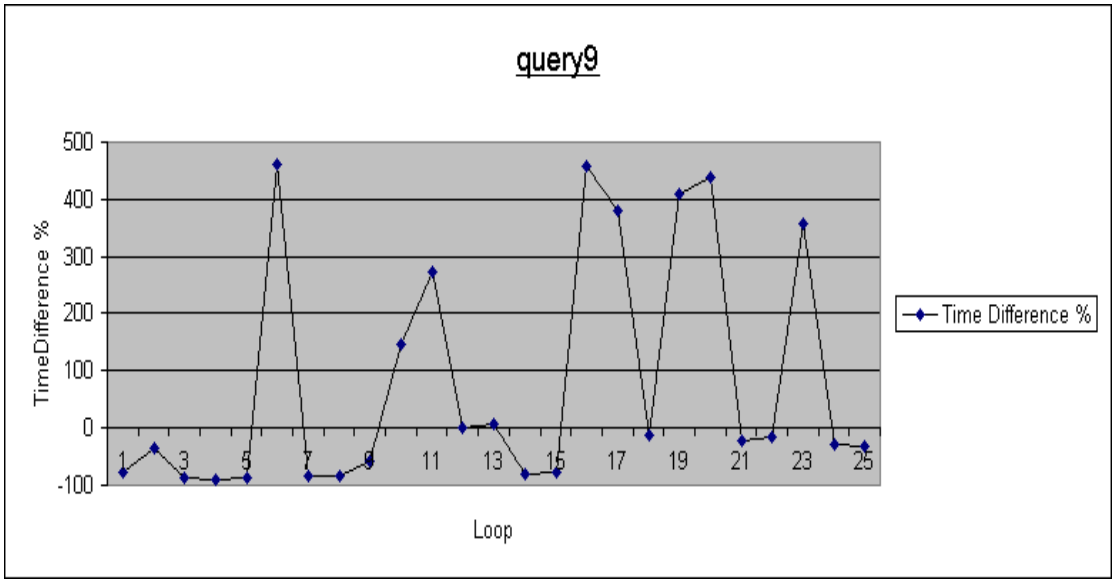
```sql
 SELECT i_item_id,i_item_desc,s_state,
       Count(ss_quantity) AS store_sales_quantitycount,
       Avg(ss_quantity) AS store_sales_quantityave,
       Stddev_samp(ss_quantity) AS store_sales_quantitystdev,
       Stddev_samp(ss_quantity) / Avg(ss_quantity) AS store_sales_quantitycov,
       Count(sr_return_quantity) AS as_store_returns_quantitycount,Avg(
       sr_return_quantity) AS as_store_returns_quantityave,
       Stddev_samp(sr_return_quantity) AS as_store_returns_quantitystdev,
              Stddev_samp(sr_return_quantity) / Avg(sr_return_quantity) AS
              store_returns_quantitycov,
       Count(cs_quantity) AS catalog_sales_quantitycount,
       Avg(cs_quantity) AS catalog_sales_quantityave,
       Stddev_samp(cs_quantity) / Avg(cs_quantity) AS
       catalog_sales_quantitystdev,Stddev_samp(cs_quantity) / Avg(cs_quantity)
                                   AS catalog_sales_quantitycov
FROM   store_sales,store_returns,catalog_sales,date_dim d1,date_dim d2,date_dim
       d3,
       store,item
WHERE  d1.d_quarter_name_first=1
       AND d1.d_date_sk = ss_sold_date_sk
       AND i_item_sk = ss_item_sk
       AND s_store_sk = ss_store_sk
       AND ss_customer_sk = sr_customer_sk
       AND ss_item_sk = sr_item_sk
       AND ss_ticket_number = sr_ticket_number
       AND sr_returned_date_sk = d2.d_date_sk
       AND (d2.d_quarter_name_first=1 OR d2.d_quarter_name_second=1
            OR d2.d_quarter_name_third=1 )
       AND sr_customer_sk = cs_bill_customer_sk
       AND sr_item_sk = cs_item_sk
       AND cs_sold_date_sk = d3.d_date_sk
       AND (d3.d_quarter_name_first=1 OR d3.d_quarter_name_second=1
            OR d3.d_quarter_name_third=1 )
GROUP  BY i_item_id,i_item_desc,s_state
ORDER  BY i_item_id,i_item_desc,s_state;
```

Μετατροπές SQL

d_quarter_name: d_quarter_name = 'Value' <-> d_quarter_name_Value = 1

d_quarter_name: d_quarter_name IN ('Value1', 'Value2', 'Value3') <-> d_quarter_name_Value1=1 OR d_quarter_name_Value2=1 OR d_quarter_name_Value3=1

Διάγραμμα Ποσοστιαίας Χρονικής Απόκλισης



Σχόλια

Παρατηρούνται συνεχείς εναλλαγές στο πρόσημο της χρονικής διαφοράς αλλά και εκατέρωθεν μεγάλες χρονικές διαφορές, ιδιαίτερα εις βάρος της booleanized βάσης δεδομένων, γεγονός που δείχνει ότι ο μετασχηματισμός boolean approach δεν είχε κάποια σταθερής μορφής επίδραση στον χρόνο εκτέλεσης του query στις δυο βάσεις δεδομένων.

QUERY 10

Basic SQL

query10.sql Raw

```sql
SELECT i_brand_id              AS brand_id,
       i_brand                 AS brand,
       i_manufact_id,
       i_manufact,
       Sum(ss_ext_sales_price) AS ext_price
FROM   date_dim,
       store_sales,
       item,
       customer,
       customer_address,
       store
WHERE  d_date_sk = ss_sold_date_sk
       AND ss_item_sk = i_item_sk
       AND d_month_name = 'May'
       AND d_year > 2000
       AND ss_customer_sk = c_customer_sk
       AND c_current_addr_sk = ca_address_sk
       AND ss_store_sk = s_store_sk
GROUP  BY i_brand,
          i_brand_id,
          i_manufact_id,
          i_manufact
ORDER  BY ext_price DESC,
          i_brand,
          i_brand_id,
          i_manufact_id,
          i_manufact
```

Booleanized SQL

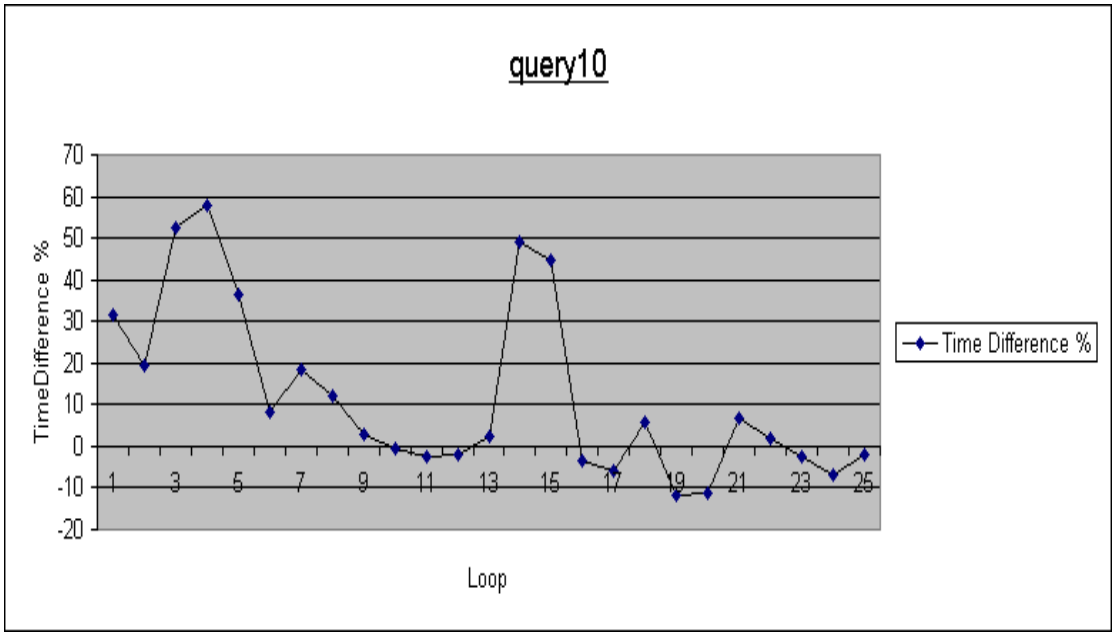
bquery10.sql Raw

```sql
SELECT i_brand_id              AS brand_id,
       i_brand                 AS brand,
       i_manufact_id,
       i_manufact,
       Sum(ss_ext_sales_price) AS ext_price
FROM   date_dim,
       store_sales,
       item,
       customer,
       customer_address,
       store
WHERE  d_date_sk = ss_sold_date_sk
       AND ss_item_sk = i_item_sk
       AND d_month_name_may = 1
       AND d_year > 2000
       AND ss_customer_sk = c_customer_sk
       AND c_current_addr_sk = ca_address_sk
       AND ss_store_sk = s_store_sk
GROUP  BY i_brand,
          i_brand_id,
          i_manufact_id,
          i_manufact
ORDER  BY ext_price DESC,
          i_brand,
          i_brand_id,
          i_manufact_id,
          i_manufact
```

Μετατροπές SQL

d_month_name: d_month_name = 'Value' <-> d_month_name_Value = 1

Διάγραμμα Ποσοστιαίας Χρονικής Απόκλισης



Σχόλια

Παρατηρούνται συνεχείς εναλλαγές στο πρόσημο της χρονικής διαφοράς αλλά και μεγάλες χρονικές διαφορές (κυρίως όταν αυτές είναι υπέρ της αρχικής βάσης δεδομένων), πράγμα που δείχνει ότι ο μετασχηματισμός boolean approach δεν είχε κάποια σταθερής μορφής επίδραση στον χρόνο εκτέλεσης του query στις δυο βάσεις δεδομένων.

QUERY 11

Basic SQL

query11.sql Raw

```sql
SELECT i_item_id,Avg(cs_quantity) AS agg1,Avg(cs_list_price) AS agg2,
       Avg(cs_coupon_amt) AS agg3,Avg(cs_sales_price) AS agg4
FROM   catalog_sales,customer_demographics,date_dim,item,promotion
WHERE  cs_sold_date_sk = d_date_sk
       AND cs_item_sk = i_item_sk
       AND cs_bill_cdemo_sk = cd_demo_sk
       AND cs_promo_sk = p_promo_sk
       AND cd_gender = 'female'
       AND cd_marital_status = 'single'
       AND cd_education_status = 'university'
       AND ( p_channel_email = 'no'
             OR p_channel_event = 'no' )
       AND d_year < 2000
GROUP  BY i_item_id
ORDER  BY i_item_id;
```

Booleanized SQL

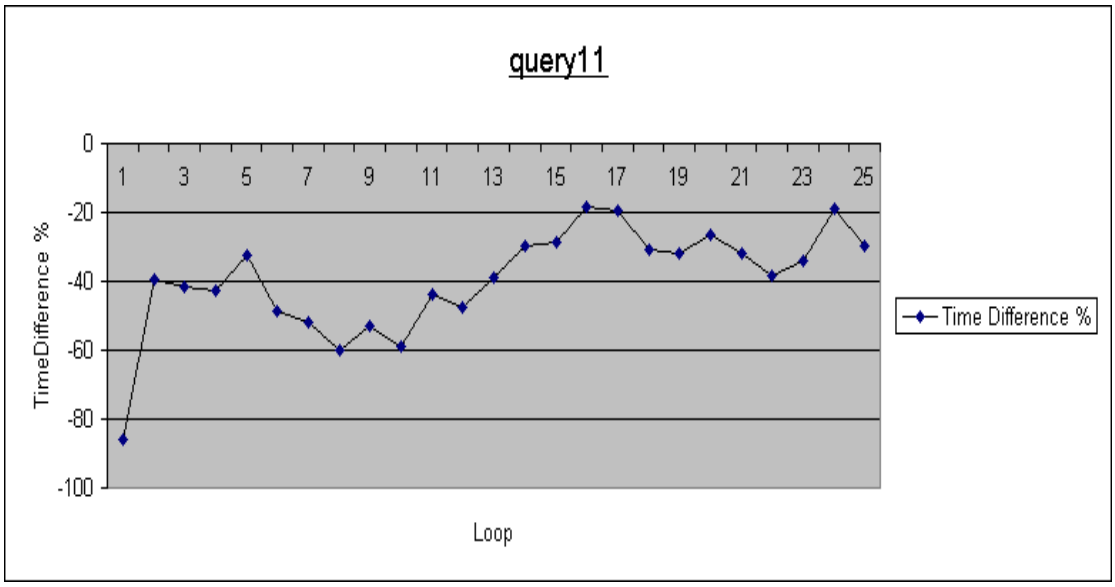
bquery11.sql Raw

```sql
 SELECT i_item_id,Avg(cs_quantity) AS agg1,Avg(cs_list_price) AS agg2,
       Avg(cs_coupon_amt) AS agg3,Avg(cs_sales_price) AS agg4
FROM   catalog_sales,customer_demographics,date_dim,item,promotion
WHERE  cs_sold_date_sk = d_date_sk
       AND cs_item_sk = i_item_sk
       AND cs_bill_cdemo_sk = cd_demo_sk
       AND cs_promo_sk = p_promo_sk
       AND cd_gender_female=1
       AND cd_marital_status_single=1
       AND cd_education_status_university=1
       AND ( p_channel_email_no=1
             OR p_channel_event_no=1 )
       AND d_year < 2000
GROUP  BY i_item_id
ORDER  BY i_item_id;
```

Μετατροπές SQL

cd_gender:	cd_gender = = 'Value' <-> cd_gender_Value = 1
cd_marital_status:	cd_marital_status = 'Value' <-> cd_marital_status_Value = 1
cd_education_status:	cd_education_status = 'Value' <-> cd_education_status_Value = 1
p_channel_email:	p_channel_email = 'Value' <-> p_channel_email_Value = 1
p_channel_event:	p_channel_event = 'Value' <-> p_channel_event_Value = 1

Διάγραμμα Ποσοστιαίας Χρονικής Απόκλισης



Σχόλια

Στην περίπτωση αυτή φαίνεται πως ο ταυτόχρονος μετασχηματισμός πέντε attributes από δυο διαφορετικούς πίνακες επέφερε ορατή διαφορά στο χρόνο εκτέλεσης του query υπέρ της booleanized βάση δεδομένων, στην οποία το query ολοκληρώνεται γρηγορότερα σχεδόν σε όλες τις επαναλήψεις.

QUERY 12

Basic SQL

query12.sql Raw

```
SELECT c_customer_id,
       c_salutation,
       c_first_name,
       c_last_name,
       c_preferred_cust_flag,
       c_birth_day,
       c_birth_month,
       c_birth_year,
       c_birth_country,
       c_login,
       c_email_address,
       c_last_review_date_sk,
       ctr_total_return
FROM   (SELECT wr_returning_customer_sk AS ctr_customer_sk,
               ca_state                 AS ctr_state,
               Sum(wr_return_amt)       AS ctr_total_return
        FROM   web_returns,
               date_dim,
               customer_address
        WHERE  wr_returned_date_sk = d_date_sk
               AND d_year > 2000
               AND wr_returning_addr_sk = ca_address_sk
        GROUP  BY wr_returning_customer_sk,
                  ca_state) AS ctr1,
       customer_address,
       customer
WHERE  ctr1.ctr_total_return > (SELECT Avg(ctr_total_return) * 1.2
                                FROM   (SELECT
              wr_returning_customer_sk AS ctr_customer_sk,
              ca_state                 AS ctr_state,
              Sum(wr_return_amt)       AS ctr_total_return
                                        FROM   web_returns,
                                               date_dim,
                                               customer_address
                                        WHERE  wr_returned_date_sk = d_date_sk
                                               AND d_year > 2000
                                               AND wr_returning_addr_sk =
                                                   ca_address_sk
                                        GROUP  BY wr_returning_customer_sk,
                                                  ca_state) AS ctr2
                                WHERE  ctr1.ctr_state = ctr2.ctr_state)
       AND ca_address_sk = c_current_addr_sk
       AND ctr1.ctr_customer_sk = c_customer_sk
ORDER  BY c_customer_id,
          c_salutation,
          c_first_name,
          c_last_name,
          c_preferred_cust_flag,
          c_birth_day,
          c_birth_month,
          c_birth_year,
          c_birth_country,
          c_login,
          c_email_address,
          c_last_review_date_sk,
          ctr_total_return
```

Booleanized SQL

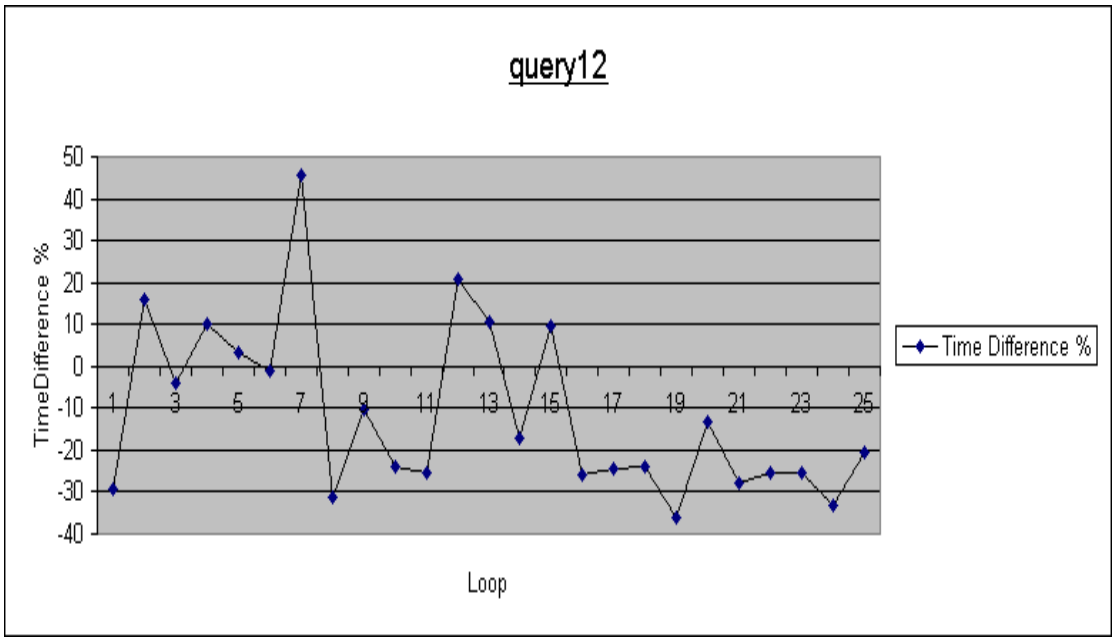
bquery12.sql Raw

```
SELECT c_customer_id,
       c_salutation,
       c_first_name,
       c_last_name,
       c_preferred_cust_flag_yes,
       c_preferred_cust_flag_no,
       c_birth_day,
       c_birth_month,
       c_birth_year,
       c_birth_country,
       c_login,
       c_email_address,
       c_last_review_date_sk,
       ctr_total_return
FROM   (SELECT wr_returning_customer_sk AS ctr_customer_sk,
               ca_state                 AS ctr_state,
               Sum(wr_return_amt)       AS ctr_total_return
        FROM   web_returns,
               date_dim,
               customer_address
        WHERE  wr_returned_date_sk = d_date_sk
               AND d_year > 2000
               AND wr_returning_addr_sk = ca_address_sk
        GROUP  BY wr_returning_customer_sk,
                  ca_state) AS ctr1,
       customer_address,
       customer
WHERE  ctr1.ctr_total_return > (SELECT Avg(ctr_total_return) * 1.2
                                FROM   (SELECT
              wr_returning_customer_sk AS ctr_customer_sk,
              ca_state                 AS ctr_state,
              Sum(wr_return_amt)       AS ctr_total_return
                                        FROM   web_returns,
                                               date_dim,
                                               customer_address
                                        WHERE  wr_returned_date_sk = d_date_sk
                                               AND d_year > 2000
                                               AND wr_returning_addr_sk =
                                                   ca_address_sk
                                        GROUP  BY wr_returning_customer_sk,
                                                  ca_state) AS ctr2
                                WHERE  ctr1.ctr_state = ctr2.ctr_state)
       AND ca_address_sk = c_current_addr_sk
       AND ctr1.ctr_customer_sk = c_customer_sk
ORDER  BY c_customer_id,
          c_salutation,
          c_first_name,
          c_last_name,
          c_preferred_cust_flag_yes,
          c_preferred_cust_flag_yes,
          c_birth_day,
          c_birth_month,
          c_birth_year,
          c_birth_country,
          c_login,
          c_email_address,
          c_last_review_date_sk,
          ctr_total_return
```

Μετατροπές SQL

c_preferred_customer_flag:
c_preferred_customer_flag <->
c_preferred_customer_flag_yes , c_preferred_customer_flag_no

Διάγραμμα Ποσοστιαίας Χρονικής Απόκλισης



Σχόλια

Μετασχηματίστηκε ένα μόνο attribute, το οποίο μάλιστα δεν εμπλέκεται σε κάποιο WHERE clause του query. Παρά το γεγονός ότι οι περισσότερες επαναλήψεις κατέγραψαν μειωμένο χρόνο εκτέλεσης για την booleanized βάση δεδομένων, οι συχνές εναλλαγές προσήμου μαρτυρούν ότι η επίδραση του μετασχηματισμού boolean approach δεν είναι σταθερή για queries ατής της μορφής.

QUERY 13

Basic SQL

query13.sql Raw

```sql
SELECT c_last_name,
       c_first_name,
       c_salutation,
       c_preferred_cust_flag,
       ss_ticket_number,
       cnt
FROM   (SELECT ss_ticket_number,
               ss_customer_sk,
               Count(*) AS cnt
        FROM   store_sales,
               date_dim,
               store,
               household_demographics
        WHERE  store_sales.ss_sold_date_sk = date_dim.d_date_sk
               AND store_sales.ss_store_sk = store.s_store_sk
               AND store_sales.ss_hdemo_sk = household_demographics.hd_demo_sk
               AND ( date_dim.d_day_name = 'Monday'
                      OR date_dim.d_day_name = 'Wednesday'
                      OR date_dim.d_day_name = 'Friday' )
               AND household_demographics.hd_vehicle_count > 0
               AND ( CASE
                       WHEN household_demographics.hd_vehicle_count > 0 THEN
                       household_demographics.hd_dep_count /
                       household_demographics.hd_vehicle_count
                       ELSE NULL
                     end ) > 1.2
               AND date_dim.d_year IN ( 1999, 2000, 2001 )
        GROUP  BY ss_ticket_number,
                  ss_customer_sk) dn,
       customer
WHERE  ss_customer_sk = c_customer_sk
       AND cnt BETWEEN 0 AND 20
ORDER  BY c_last_name,
          c_first_name,
          c_salutation,
          c_preferred_cust_flag DESC
```

Booleanized SQL

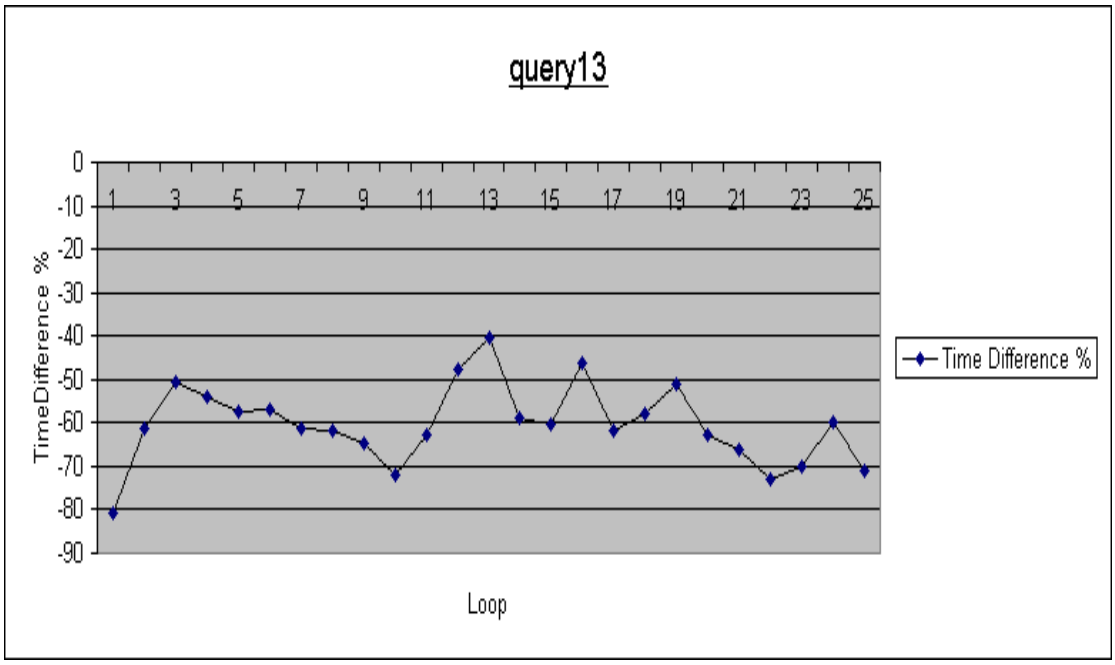
bquery13.sql Raw

```sql
SELECT c_last_name,
       c_first_name,
       c_salutation,
       c_preferred_cust_flag_yes,
       c_preferred_cust_flag_no,
       ss_ticket_number,
       cnt
FROM   (SELECT ss_ticket_number,
               ss_customer_sk,
               Count(*) AS cnt
        FROM   store_sales,
               date_dim,
               store,
               household_demographics
        WHERE  store_sales.ss_sold_date_sk = date_dim.d_date_sk
               AND store_sales.ss_store_sk = store.s_store_sk
               AND store_sales.ss_hdemo_sk = household_demographics.hd_demo_sk
               AND ( date_dim.d_day_name_monday = 1
                      OR date_dim.d_day_name_wednesday = 1
                      OR date_dim.d_day_name_friday = 1 )
               AND household_demographics.hd_vehicle_count > 0
               AND ( CASE
                       WHEN household_demographics.hd_vehicle_count > 0 THEN
                       household_demographics.hd_dep_count /
                       household_demographics.hd_vehicle_count
                       ELSE NULL
                     end ) > 1.2
               AND date_dim.d_year IN ( 1999, 2000, 2001 )
        GROUP  BY ss_ticket_number,
                  ss_customer_sk) dn,
       customer
WHERE  ss_customer_sk = c_customer_sk
       AND cnt BETWEEN 0 AND 20
ORDER  BY c_last_name,
          c_first_name,
          c_salutation,
          c_preferred_cust_flag_yes,
          c_preferred_cust_flag_no DESC
```

Μετατροπές SQL

c_preferred_customer_flag:
c_preferred_customer_flag = 'Value' <-> c_preferred_customer_flag_Value = 1
d_day_name: d_day_name = 'Value' <-> d_day_name_Value = 1

Διάγραμμα Ποσοστιαίας Χρονικής Απόκλισης



Σχόλια

Μετασχηματίστηκαν δυο χαρακτηριστικά, από δυο διαφορετικούς πίνακες. Μάλιστα ένα εξ αυτών εμπλέκεται σε ένα εμφωλευμένο query. Τα αποτελέσματα δείχνουν σαφή υπεροχή της booleanized βάσης δεδομένων, αφού στο σύνολο τον επαναλήψεων ο χρόνος εκτέλεσης του query σε αυτή ήταν αισθητά μικρότερος.

QUERY 14

Basic SQL

query14.sql Raw

```sql
SELECT inv1.w_warehouse_sk,
       inv1.i_item_sk,
       inv1.d_month_name,
       inv1.mean,
       inv1.cov,
       inv2.w_warehouse_sk,
       inv2.i_item_sk,
       inv2.d_month_name,
       inv2.mean,
       inv2.cov
FROM   (SELECT w_warehouse_name,
               w_warehouse_sk,
               i_item_sk,
               d_month_name,
               stdev,
               mean,
               CASE mean
                 WHEN 0 THEN NULL
                 ELSE stdev / mean
               end cov
        FROM  (SELECT w_warehouse_name,
                      w_warehouse_sk,
                      i_item_sk,
                      d_month_name,
                      Stddev_samp(inv_quantity_on_hand) stdev,
                      Avg(inv_quantity_on_hand)         AS mean
               FROM   inventory,
                      item,
                      warehouse,
                      date_dim
               WHERE  inv_item_sk = i_item_sk
                      AND inv_warehouse_sk = w_warehouse_sk
                      AND inv_date_sk = d_date_sk
                      AND d_year > 2000
               GROUP  BY w_warehouse_name,
                         w_warehouse_sk,
                         i_item_sk,
                         d_month_name) foo
        WHERE  CASE mean
                 WHEN 0 THEN 0
                 ELSE stdev / mean
               end > 1) AS inv1,
       (SELECT w_warehouse_name,
               w_warehouse_sk,
               i_item_sk,
               d_month_name,
               stdev,
               mean,
               CASE mean
                 WHEN 0 THEN NULL
                 ELSE stdev / mean
               end cov
        FROM  (SELECT w_warehouse_name,
                      w_warehouse_sk,
                      i_item_sk,
                      d_month_name,
                      Stddev_samp(inv_quantity_on_hand) stdev,
                      Avg(inv_quantity_on_hand)         AS mean
               FROM   inventory,
                      item,
                      warehouse,
                      date_dim
               WHERE  inv_item_sk = i_item_sk
                      AND inv_warehouse_sk = w_warehouse_sk
                      AND inv_date_sk = d_date_sk
                      AND d_year > 2000
               GROUP  BY w_warehouse_name,
                         w_warehouse_sk,
                         i_item_sk,
                         d_month_name) foo
        WHERE  CASE mean
                 WHEN 0 THEN 0
                 ELSE stdev / mean
               end > 1) AS inv2
WHERE  inv1.i_item_sk = inv2.i_item_sk
       AND inv1.w_warehouse_sk = inv2.w_warehouse_sk
       AND inv1.d_month_name = 'January'
       AND inv2.d_month_name = 'February'
ORDER  BY inv1.w_warehouse_sk,
          inv1.i_item_sk,
          inv1.d_month_name,
          inv1.mean,
          inv1.cov,
          inv2.d_month_name,
          inv2.mean,
          inv2.cov;
```

Booleanized SQL

bquery14.sql Raw

```
SELECT inv1.w_warehouse_sk,
       inv1.i_item_sk,
       inv1.d_month_name_january,
       inv1.d_month_name_february,
       inv1.d_month_name_march,
       inv1.d_month_name_april,
       inv1.d_month_name_may,
       inv1.d_month_name_june,
       inv1.d_month_name_july,
       inv1.d_month_name_august,
       inv1.d_month_name_september,
       inv1.d_month_name_october,
       inv1.d_month_name_november,
       inv1.d_month_name_december,
       inv1.mean,
       inv1.cov,
       inv2.w_warehouse_sk,
       inv2.i_item_sk,
       inv2.d_month_name_january,
       inv2.d_month_name_february,
       inv2.d_month_name_march,
       inv2.d_month_name_april,
       inv2.d_month_name_may,
       inv2.d_month_name_june,
       inv2.d_month_name_july,
       inv2.d_month_name_august,
       inv2.d_month_name_september,
       inv2.d_month_name_october,
       inv2.d_month_name_november,
       inv2.d_month_name_december,
       inv2.mean,
       inv2.cov
FROM   (SELECT w_warehouse_name,
               w_warehouse_sk,
               i_item_sk,
               d_month_name_january,
               d_month_name_february,
               d_month_name_march,
               d_month_name_april,
               d_month_name_may,
               d_month_name_june,
               d_month_name_july,
               d_month_name_august,
               d_month_name_september,
               d_month_name_october,
               d_month_name_november,
               d_month_name_december,
               stdev,
               mean,
               CASE mean
                 WHEN 0 THEN NULL
                 ELSE stdev / mean
               end cov
        FROM  (SELECT w_warehouse_name,
                      w_warehouse_sk,
                      i_item_sk,
                      d_month_name_january,
                      d_month_name_february,
                      d_month_name_march,
                      d_month_name_april,
                      d_month_name_may,
                      d_month_name_june,
                      d_month_name_july,
                      d_month_name_august,
                      d_month_name_september,
                      d_month_name_october,
                      d_month_name_november,
                      d_month_name_december,
                      Stddev_samp(inv_quantity_on_hand) stdev,
                      Avg(inv_quantity_on_hand)         AS mean
               FROM   inventory,
                      item,
                      warehouse,
                      date_dim
               WHERE  inv_item_sk = i_item_sk
                      AND inv_warehouse_sk = w_warehouse_sk
                      AND inv_date_sk = d_date_sk
                      AND d_year > 2000
               GROUP  BY w_warehouse_name,
                         w_warehouse_sk,
                         i_item_sk,
                         d_month_name_january,
                         d_month_name_february,
                         d_month_name_march,
                         d_month_name_april,
                         d_month_name_may,
                         d_month_name_june,
                         d_month_name_july,
                         d_month_name_august,
                         d_month_name_september,
                         d_month_name_october,
                         d_month_name_november,
                         d_month_name_december) foo
        WHERE  CASE mean
                 WHEN 0 THEN 0
                 ELSE stdev / mean
               end > 1) AS inv1,
```

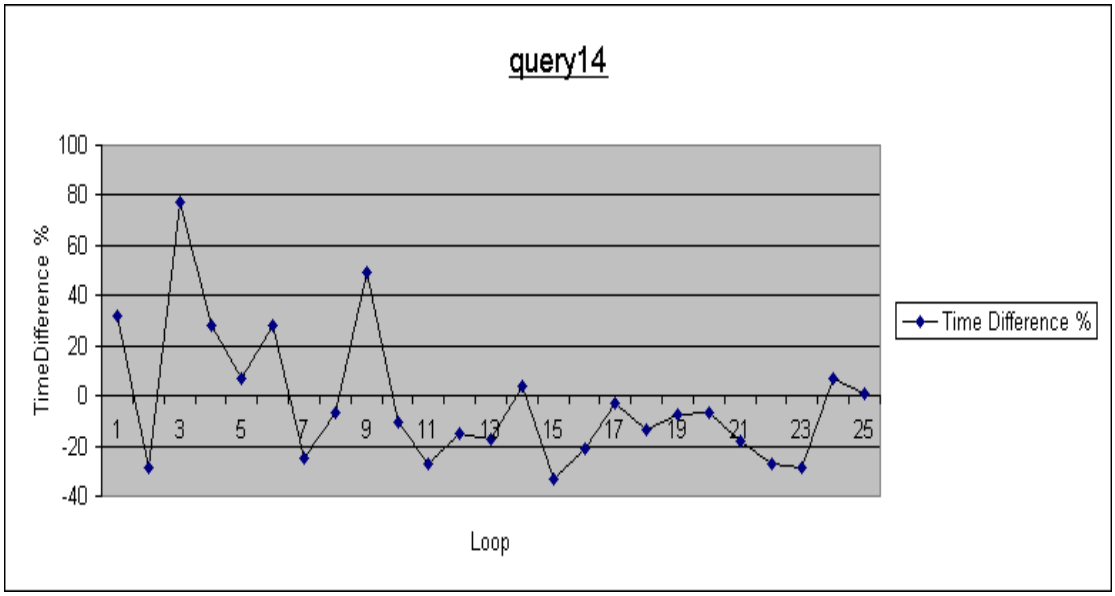
```
        (SELECT w_warehouse_name,
                w_warehouse_sk,
                i_item_sk,
                d_month_name_january,
                d_month_name_february,
                d_month_name_march,
                d_month_name_april,
                d_month_name_may,
                d_month_name_june,
                d_month_name_july,
                d_month_name_august,
                d_month_name_september,
                d_month_name_october,
                d_month_name_november,
                d_month_name_december,
                stdev,
                mean,
                CASE mean
                  WHEN 0 THEN NULL
                  ELSE stdev / mean
                end cov
         FROM  (SELECT w_warehouse_name,
                       w_warehouse_sk,
                       i_item_sk,
                       d_month_name_january,
                       d_month_name_february,
                       d_month_name_march,
                       d_month_name_april,
                       d_month_name_may,
                       d_month_name_june,
                       d_month_name_july,
                       d_month_name_august,
                       d_month_name_september,
                       d_month_name_october,
                       d_month_name_november,
                       d_month_name_december,
                       Stddev_samp(inv_quantity_on_hand) stdev,
                       Avg(inv_quantity_on_hand)         AS mean
                FROM   inventory,
                       item,
                       warehouse,
                       date_dim
                WHERE  inv_item_sk = i_item_sk
                       AND inv_warehouse_sk = w_warehouse_sk
                       AND inv_date_sk = d_date_sk
                       AND d_year > 2000
                GROUP  BY w_warehouse_name,
                          w_warehouse_sk,
                          i_item_sk,
                          d_month_name_january,
                          d_month_name_february,
                          d_month_name_march,
                          d_month_name_april,
                          d_month_name_may,
                          d_month_name_june,
                          d_month_name_july,
                          d_month_name_august,
                          d_month_name_september,
                          d_month_name_october,
                          d_month_name_november,
                          d_month_name_december) foo
         WHERE  CASE mean
                  WHEN 0 THEN 0
                  ELSE stdev / mean
                end > 1) AS inv2
WHERE  inv1.i_item_sk = inv2.i_item_sk
       AND inv1.w_warehouse_sk = inv2.w_warehouse_sk
       AND inv1 .d_month_name_january = 1
       AND inv2 .d_month_name_february = 1
ORDER  BY inv1.w_warehouse_sk,
          inv1.i_item_sk,
          inv1.d_month_name_january,
          inv1.d_month_name_february,
          inv1.d_month_name_march,
          inv1.d_month_name_april,
          inv1.d_month_name_may,
          inv1.d_month_name_june,
          inv1.d_month_name_july,
          inv1.d_month_name_august,
          inv1.d_month_name_september,
          inv1.d_month_name_october,
          inv1.d_month_name_november,
          inv1.d_month_name_december,
          inv1.mean,
          inv1.cov,
          inv2.d_month_name_january,
          inv2.d_month_name_february,
          inv2.d_month_name_march,
          inv2.d_month_name_april,
          inv2.d_month_name_may,
          inv2.d_month_name_june,
          inv2.d_month_name_july,
          inv2.d_month_name_august,
          inv2.d_month_name_september,
          inv2.d_month_name_october,
          inv2.d_month_name_november,
          inv2.d_month_name_december,
          inv2.mean,
          inv2.cov;
```

Μετατροπές SQL

d_month_name:

d_month_ name <-> d_month_name_January, …, d_month_name_December

Διάγραμμα Ποσοστιαίας Χρονικής Απόκλισης



Σχόλια

Μετασχηματίστηκε ένα attribute, το οποίο εμπλέκεται σε SELECT, GROUP BY και ORDER BY clauses. Παρόλο που οι περισσότερες επαναλήψεις κατέγραψαν ελαφρώς μικρότερο χρόνο εκτέλεσης στην booleanized βάση δεδομένων, οι εναλλαγές προσήμου στο διάγραμμα δείχνουν ότι η επίδραση του μετασχηματισμού είναι απρόβλεπτη για queries αυτής της μορφής.

QUERY15

Basic SQL

query15.sql Raw

```sql
SELECT inv1.w_warehouse_sk,
       inv1.i_item_sk,
       inv1.d_month_name,
       inv1.mean,
       inv1.cov,
       inv2.w_warehouse_sk,
       inv2.i_item_sk,
       inv2.d_month_name,
       inv2.mean,
       inv2.cov
FROM   (SELECT w_warehouse_name,
               w_warehouse_sk,
               i_item_sk,
               d_month_name,
               stdev,
               mean,
               CASE mean
                 WHEN 0 THEN NULL
                 ELSE stdev / mean
               end cov
        FROM  (SELECT w_warehouse_name,
                      w_warehouse_sk,
                      i_item_sk,
                      d_month_name,
                      Stddev_samp(inv_quantity_on_hand) stdev,
                      Avg(inv_quantity_on_hand)         AS mean
               FROM   inventory,
                      item,
                      warehouse,
                      date_dim
               WHERE  inv_item_sk = i_item_sk
                      AND inv_warehouse_sk = w_warehouse_sk
                      AND inv_date_sk = d_date_sk
                      AND d_year > 2000
               GROUP  BY w_warehouse_name,
                         w_warehouse_sk,
                         i_item_sk,
                         d_month_name) foo
        WHERE  CASE mean
                 WHEN 0 THEN 0
                 ELSE stdev / mean
               end > 1) AS inv1,
       (SELECT w_warehouse_name,
               w_warehouse_sk,
               i_item_sk,
               d_month_name,
               stdev,
               mean,
               CASE mean
                 WHEN 0 THEN NULL
                 ELSE stdev / mean
               end cov
        FROM  (SELECT w_warehouse_name,
                      w_warehouse_sk,
                      i_item_sk,
                      d_month_name,
                      Stddev_samp(inv_quantity_on_hand) stdev,
                      Avg(inv_quantity_on_hand)         AS mean
               FROM   inventory,
                      item,
                      warehouse,
                      date_dim
               WHERE  inv_item_sk = i_item_sk
                      AND inv_warehouse_sk = w_warehouse_sk
                      AND inv_date_sk = d_date_sk
                      AND d_year > 2000
               GROUP  BY w_warehouse_name,
                         w_warehouse_sk,
                         i_item_sk,
                         d_month_name) foo
        WHERE  CASE mean
                 WHEN 0 THEN 0
                 ELSE stdev / mean
               end > 1) AS inv2
WHERE  inv1.i_item_sk = inv2.i_item_sk
       AND inv1.w_warehouse_sk = inv2.w_warehouse_sk
       AND inv1.d_month_name = 'October'
       AND inv2.d_month_name = 'September'
       AND inv1.cov > 1.5
ORDER  BY inv1.w_warehouse_sk,
          inv1.i_item_sk,
          inv1.d_month_name,
          inv1.mean,
          inv1.cov,
          inv2.d_month_name,
          inv2.mean,
          inv2.cov
```

Booleanized SQL

bquery15.sql Raw

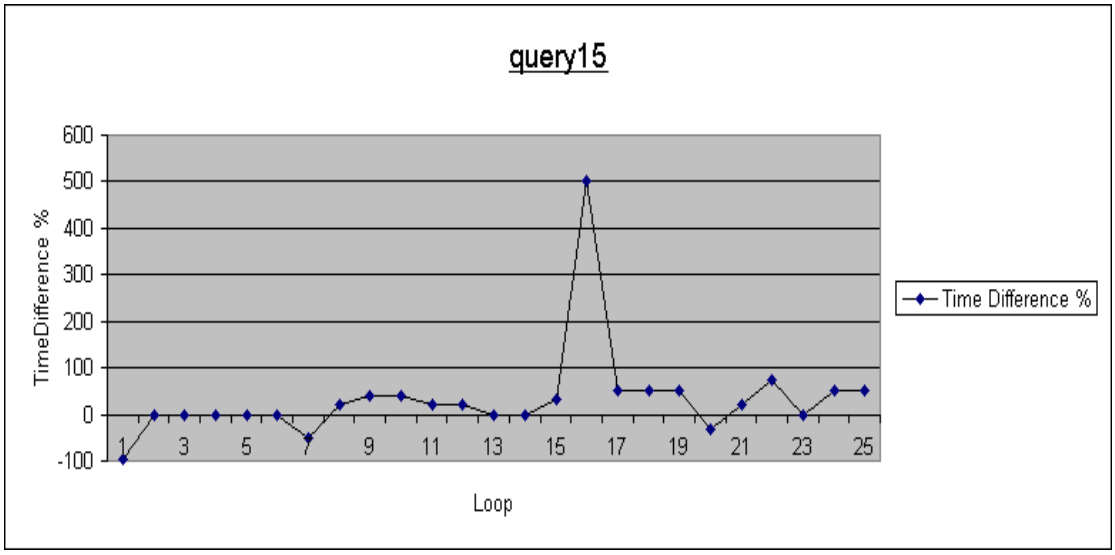
```sql
SELECT inv1.w_warehouse_sk,
       inv1.i_item_sk,
       inv1.d_month_name_january,
       inv1.d_month_name_february,
       inv1.d_month_name_march,
       inv1.d_month_name_april,
       inv1.d_month_name_may,
       inv1.d_month_name_june,
       inv1.d_month_name_july,
       inv1.d_month_name_august,
       inv1.d_month_name_september,
       inv1.d_month_name_october,
       inv1.d_month_name_november,
       inv1.d_month_name_december,
       inv1.mean,
       inv1.cov,
       inv2.w_warehouse_sk,
       inv2.i_item_sk,
       inv2.d_month_name_january,
       inv2.d_month_name_february,
       inv2.d_month_name_march,
       inv2.d_month_name_april,
       inv2.d_month_name_may,
       inv2.d_month_name_june,
       inv2.d_month_name_july,
       inv2.d_month_name_august,
       inv2.d_month_name_september,
       inv2.d_month_name_october,
       inv2.d_month_name_november,
       inv2.d_month_name_december,
       inv2.mean,
       inv2.cov
FROM   (SELECT w_warehouse_name,
               w_warehouse_sk,
               i_item_sk,
               d_month_name_january,
               d_month_name_february,
               d_month_name_march,
               d_month_name_april,
               d_month_name_may,
               d_month_name_june,
               d_month_name_july,
               d_month_name_august,
               d_month_name_september,
               d_month_name_october,
               d_month_name_november,
               d_month_name_december,
               stdev,
               mean,
               CASE mean
                 WHEN 0 THEN NULL
                 ELSE stdev / mean
               end cov
        FROM  (SELECT w_warehouse_name,
                      w_warehouse_sk,
                      i_item_sk,
                      d_month_name_january,
                      d_month_name_february,
                      d_month_name_march,
                      d_month_name_april,
                      d_month_name_may,
                      d_month_name_june,
                      d_month_name_july,
                      d_month_name_august,
                      d_month_name_september,
                      d_month_name_october,
                      d_month_name_november,
                      d_month_name_december,
                      Stddev_samp(inv_quantity_on_hand) stdev,
                      Avg(inv_quantity_on_hand)         AS mean
               FROM   inventory,
                      item,
                      warehouse,
                      date_dim
               WHERE  inv_item_sk = i_item_sk
                      AND inv_warehouse_sk = w_warehouse_sk
                      AND inv_date_sk = d_date_sk
                      AND d_year > 2000
               GROUP  BY w_warehouse_name,
                         w_warehouse_sk,
                         i_item_sk,
                         d_month_name_january,
                         d_month_name_february,
                         d_month_name_march,
                         d_month_name_april,
                         d_month_name_may,
                         d_month_name_june,
                         d_month_name_july,
                         d_month_name_august,
                         d_month_name_september,
                         d_month_name_october,
                         d_month_name_november,
                         d_month_name_december) foo
        WHERE  CASE mean
                 WHEN 0 THEN 0
                 ELSE stdev / mean
               end > 1) AS inv1,
      (SELECT
```

```
        (SELECT w_warehouse_name,
                w_warehouse_sk,
                i_item_sk,
                d_month_name_january,
                d_month_name_february,
                d_month_name_march,
                d_month_name_april,
                d_month_name_may,
                d_month_name_june,
                d_month_name_july,
                d_month_name_august,
                d_month_name_september,
                d_month_name_october,
                d_month_name_november,
                d_month_name_december,
                stdev,
                mean,
                CASE mean
                  WHEN 0 THEN NULL
                  ELSE stdev / mean
                end cov
         FROM  (SELECT w_warehouse_name,
                       w_warehouse_sk,
                       i_item_sk,
                       d_month_name_january,
                       d_month_name_february,
                       d_month_name_march,
                       d_month_name_april,
                       d_month_name_may,
                       d_month_name_june,
                       d_month_name_july,
                       d_month_name_august,
                       d_month_name_september,
                       d_month_name_october,
                       d_month_name_november,
                       d_month_name_december,
                       Stddev_samp(inv_quantity_on_hand) stdev,
                       Avg(inv_quantity_on_hand)         AS mean
                FROM   inventory,
                       item,
                       warehouse,
                       date_dim
                WHERE  inv_item_sk = i_item_sk
                       AND inv_warehouse_sk = w_warehouse_sk
                       AND inv_date_sk = d_date_sk
                       AND d_year > 2000
                GROUP  BY w_warehouse_name,
                          w_warehouse_sk,
                          i_item_sk,
                          d_month_name_january,
                          d_month_name_february,
                          d_month_name_march,
                          d_month_name_april,
                          d_month_name_may,
                          d_month_name_june,
                          d_month_name_july,
                          d_month_name_august,
                          d_month_name_september,
                          d_month_name_october,
                          d_month_name_november,
                          d_month_name_december) foo
         WHERE  CASE mean
                  WHEN 0 THEN 0
                  ELSE stdev / mean
                end > 1) AS inv2
WHERE  inv1.i_item_sk = inv2.i_item_sk
       AND inv1.w_warehouse_sk = inv2.w_warehouse_sk
       AND inv1 .d_month_name_october = 1
       AND inv2 .d_month_name_september = 1
       AND inv1.cov > 1.5
ORDER  BY inv1.w_warehouse_sk,
          inv1.i_item_sk,
          inv1.d_month_name_january,
          inv1.d_month_name_february,
          inv1.d_month_name_march,
          inv1.d_month_name_april,
          inv1.d_month_name_may,
          inv1.d_month_name_june,
          inv1.d_month_name_july,
          inv1.d_month_name_august,
          inv1.d_month_name_september,
          inv1.d_month_name_october,
          inv1.d_month_name_november,
          inv1.d_month_name_december,
          inv1.mean,
          inv1.cov,
          inv2.d_month_name_january,
          inv2.d_month_name_february,
          inv2.d_month_name_march,
          inv2.d_month_name_april,
          inv2.d_month_name_may,
          inv2.d_month_name_june,
          inv2.d_month_name_july,
          inv2.d_month_name_august,
          inv2.d_month_name_september,
          inv2.d_month_name_october,
          inv2.d_month_name_november,
          inv2.d_month_name_december,
          inv2.mean,
          inv2.cov
```

Μετατροπές SQL

d_month_name:
d_month_ name <-> d_month_name_January, …, d_month_name_December

Διάγραμμα Ποσοστιαίας Χρονικής Απόκλισης



Σχόλια

Μετασχηματίστηκε ένα attribute, το οποίο εμπλέκεται σε SELECT, GROUP BY και ORDER BY clauses. Παρόλο που οι περισσότερες επαναλήψεις κατέγραψαν ελαφρώς μικρότερο χρόνο εκτέλεσης στην booleanized βάση δεδομένων, οι εναλλαγές προσήμου στο διάγραμμα δείχνουν ότι η επίδραση του μετασχηματισμού είναι απρόβλεπτη για queries αυτής της μορφής.

QUERY16

Basic SQL

query16.sql Raw

```sql
SELECT dt.d_year,
       item.i_category_id,
       item.i_category,
       Sum(ss_ext_sales_price)
FROM   date_dim dt,
       store_sales,
       item
WHERE  dt.d_date_sk = store_sales.ss_sold_date_sk
       AND store_sales.ss_item_sk = item.i_item_sk
       AND dt.d_month_name = 'April'
       AND dt.d_year > 2000
GROUP  BY dt.d_year,
          item.i_category_id,
          item.i_category
ORDER  BY Sum(ss_ext_sales_price) DESC,
          dt.d_year,
          item.i_category_id,
          item.i_category
```

Booleanized SQL

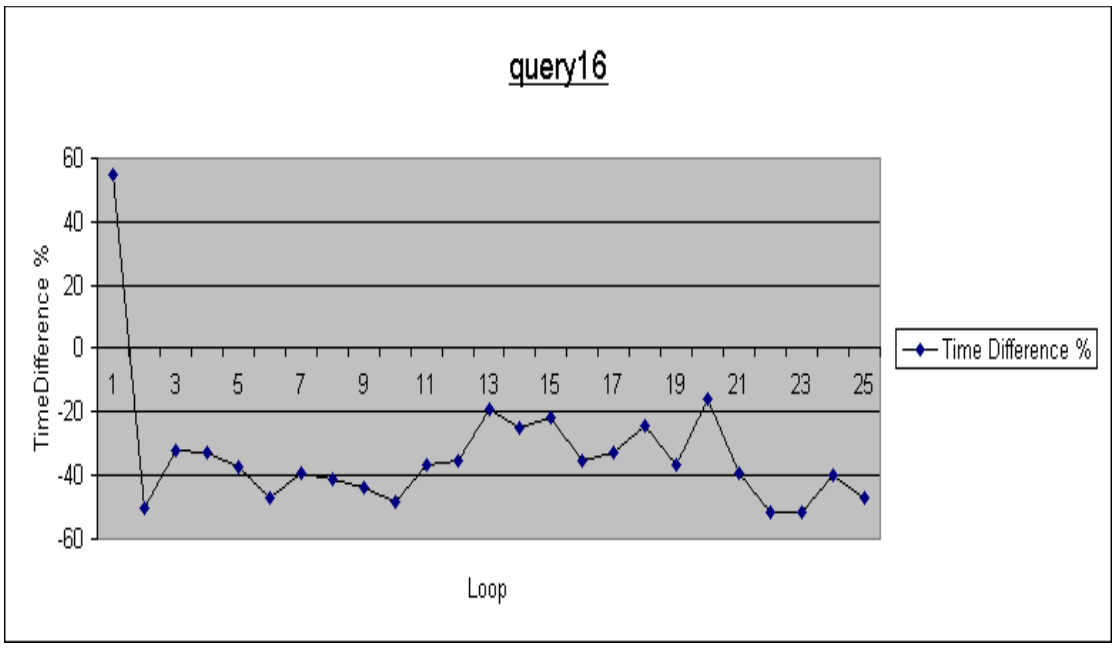
bquery16.sql Raw

```sql
SELECT dt.d_year,
       item.i_category_id,
       item.i_category,
       Sum(ss_ext_sales_price)
FROM   date_dim dt,
       store_sales,
       item
WHERE  dt.d_date_sk = store_sales.ss_sold_date_sk
       AND store_sales.ss_item_sk = item.i_item_sk
       AND dt.d_month_name_April=1
       AND dt.d_year > 2000
GROUP  BY dt.d_year,
          item.i_category_id,
          item.i_category
ORDER  BY Sum(ss_ext_sales_price) DESC,
          dt.d_year,
          item.i_category_id,
          item.i_category
```

Μετατροπές SQL

d_month_name: d_month_name = 'Value' <-> d_month_name_Value = 1

Διάγραμμα Ποσοστιαίας Χρονικής Απόκλισης



Σχόλια

Μετασχηματίστηκε ένα attribute, με σχετικά μεγάλο εύρος τιμών και το οποίο συμμετέχει σε WHERE clause. Στο σύνολο των επαναλήψεων, πλην μιας και μοναδικής αποκλίνουσας μέτρησης στην πρώτη επανάληψη, η booleanized βάση δεδομένων υπερτερεί έναντι της αρχικής, αφού καταγράφει μικρότερους χρόνους εκτέλεσης του query.

QUERY 17

Basic SQL

query17.sql Raw

```sql
SELECT s_store_name,
       s_store_id,
       Sum(CASE
             WHEN ( d_day_name = 'Sunday' ) THEN ss_sales_price
             ELSE NULL
           end) AS sun_sales,
       Sum(CASE
             WHEN ( d_day_name = 'Monday' ) THEN ss_sales_price
             ELSE NULL
           end) AS mon_sales,
       Sum(CASE
             WHEN ( d_day_name = 'Tuesday' ) THEN ss_sales_price
             ELSE NULL
           end) AS tue_sales,
       Sum(CASE
             WHEN ( d_day_name = 'Wednesday' ) THEN ss_sales_price
             ELSE NULL
           end) AS wed_sales,
       Sum(CASE
             WHEN ( d_day_name = 'Thursday' ) THEN ss_sales_price
             ELSE NULL
           end) AS thu_sales,
       Sum(CASE
             WHEN ( d_day_name = 'Friday' ) THEN ss_sales_price
             ELSE NULL
           end) AS fri_sales,
       Sum(CASE
             WHEN ( d_day_name = 'Saturday' ) THEN ss_sales_price
             ELSE NULL
           end) AS sat_sales
FROM   date_dim,
       store_sales,
       store
WHERE  d_date_sk = ss_sold_date_sk
       AND s_store_sk = ss_store_sk
       AND d_year > 2000
GROUP  BY s_store_name,
          s_store_id
ORDER  BY s_store_name,
          s_store_id,
          sun_sales,
          mon_sales,
          tue_sales,
          wed_sales,
          thu_sales,
          fri_sales,
          sat_sales
```

Booleanized SQL

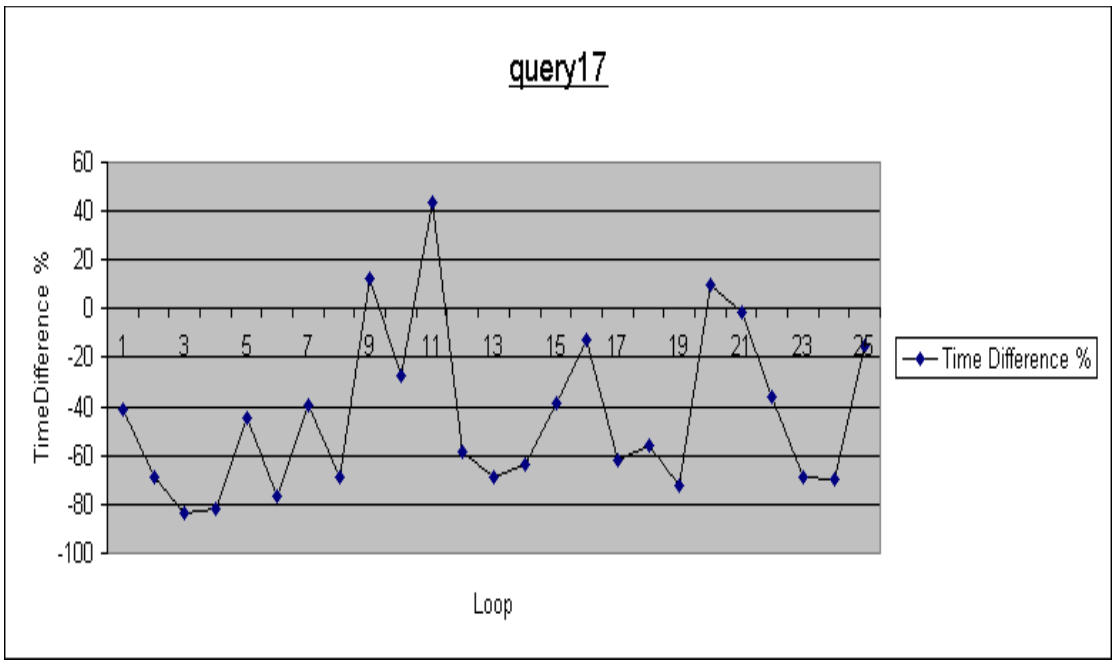
bquery17.sql Raw

```
SELECT s_store_name,
       s_store_id,
       Sum(CASE
             WHEN ( d_day_name_Sunday=1 ) THEN ss_sales_price
             ELSE NULL
           end) AS sun_sales,
       Sum(CASE
             WHEN ( d_day_name_Monday=1 ) THEN ss_sales_price
             ELSE NULL
           end) AS mon_sales,
       Sum(CASE
             WHEN ( d_day_name_Tuesday=1 ) THEN ss_sales_price
             ELSE NULL
           end) AS tue_sales,
       Sum(CASE
             WHEN ( d_day_name_Wednesday=1 ) THEN ss_sales_price
             ELSE NULL
           end) AS wed_sales,
       Sum(CASE
             WHEN ( d_day_name_Thursday=1 ) THEN ss_sales_price
             ELSE NULL
           end) AS thu_sales,
       Sum(CASE
             WHEN ( d_day_name_Friday=1 ) THEN ss_sales_price
             ELSE NULL
           end) AS fri_sales,
       Sum(CASE
             WHEN ( d_day_name_Saturday=1 ) THEN ss_sales_price
             ELSE NULL
           end) AS sat_sales
FROM   date_dim,
       store_sales,
       store
WHERE  d_date_sk = ss_sold_date_sk
       AND s_store_sk = ss_store_sk
       AND d_year > 2000
GROUP  BY s_store_name,
          s_store_id
ORDER  BY s_store_name,
          s_store_id,
          sun_sales,
          mon_sales,
          tue_sales,
          wed_sales,
          thu_sales,
          fri_sales,
          sat_sales
```

Μετατροπές SQL

d_day_name: d_day_name = 'Value' <-> d_day_name_Value = 1

Διάγραμμα Ποσοστιαίας Χρονικής Απόκλισης



Σχόλια

Μετασχηματίστηκε ένα attribute το οποίο συμμετέχει στο σημαντικότερο ως προς το υπολογιστικό κόστος μέρος του query (CASE clause). Παρά το γεγονός ότι υπάρχουν λίγες εξαιρέσεις, στη συντριπτική πλειοψηφία των επαναλήψεων η booleanized βάση δεδομένων καταγράφει αισθητά μικρότερους χρόνους εκτέλεσης, κάτι που δείχνει ότι και σε αυτή την περίπτωση, ο μετασχηματισμός boolean approach είχε θετικό αντίκτυπο.

QUERY 18

Basic SQL

query18.sql Raw

```sql
SELECT ca_zip,
       ca_city,
       ca_county,
       ca_state,
       Sum(ws_sales_price)
FROM   web_sales,
       customer,
       customer_address,
       date_dim,
       item
WHERE  ws_bill_customer_sk = c_customer_sk
       AND c_current_addr_sk = ca_address_sk
       AND ws_item_sk = i_item_sk
       AND i_item_id IN (SELECT i_item_id
                         FROM   item
                         WHERE  i_item_sk < 5000)
       AND ws_sold_date_sk = d_date_sk
       AND d_quarter_name = 'fourth'
       AND d_year > 2000
GROUP  BY ca_zip,
          ca_city,
          ca_county,
          ca_state
ORDER  BY ca_zip,
          ca_city,
          ca_county,
          ca_state
```

Booleanized SQL

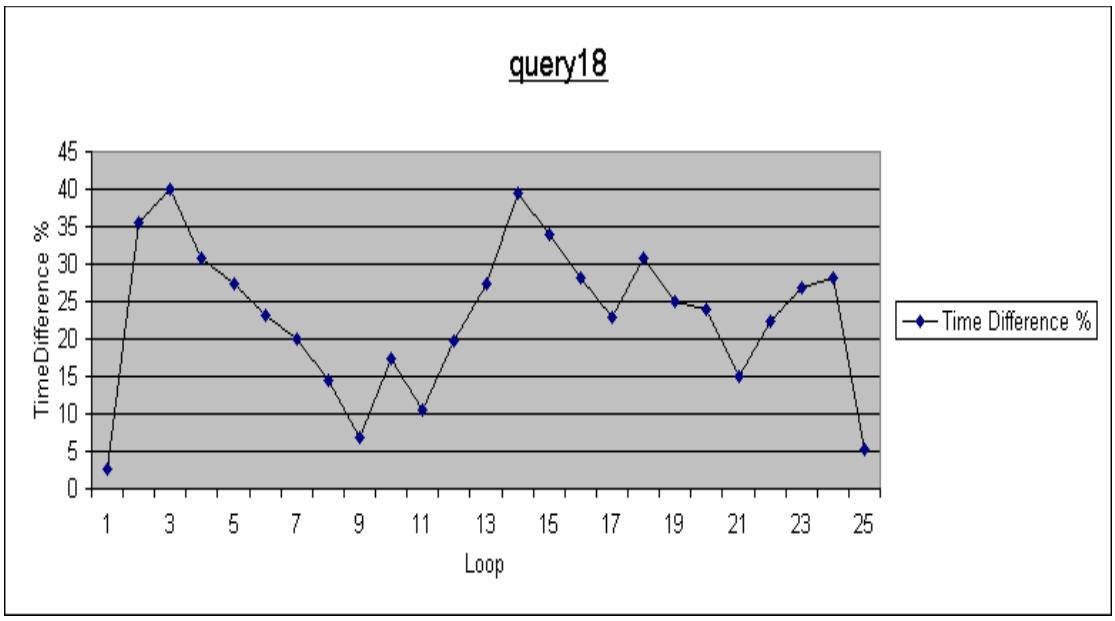
bquery18.sql Raw

```sql
SELECT ca_zip,
       ca_city,
       ca_county,
       ca_state,
       Sum(ws_sales_price)
FROM   web_sales,
       customer,
       customer_address,
       date_dim,
       item
WHERE  ws_bill_customer_sk = c_customer_sk
       AND c_current_addr_sk = ca_address_sk
       AND ws_item_sk = i_item_sk
       AND i_item_id IN (SELECT i_item_id
                         FROM   item
                         WHERE  i_item_sk < 5000)
       AND ws_sold_date_sk = d_date_sk
       AND d_quarter_name_fourth=1
       AND d_year > 2000
GROUP  BY ca_zip,
          ca_city,
          ca_county,
          ca_state
ORDER  BY ca_zip,
          ca_city,
          ca_county,
          ca_state
```

Μετατροπές SQL

d_quarter_name: d_quarter_name = 'Value' <-> d_quarter_name_Value = 1

Διάγραμμα Ποσοστιαίας Χρονικής Απόκλισης



Σχόλια

Στην περίπτωση αυτή μετασχηματίστηκε ένα attribute που εμπλέκεται σε ένα WHERE clause. Τα αποτελέσματα κατέγραψαν υψηλότερους χρόνους για την booleanized βάση δεδομένων, στο σύνολο των επαναλήψεων. Αυτό επιβεβαιώνει την απρόβλεπτη επίδραση του μετασχηματισμού στην απόδοση των queries.

QUERY 19

Basic SQL

query19.sql Raw

```
SELECT c_last_name,
       c_first_name,
       ca_city,
       bought_city,
       ss_ticket_number,
       amt,
       profit
FROM   (SELECT ss_ticket_number,
               ss_customer_sk,
               ca_city            AS bought_city,
               Sum(ss_coupon_amt) AS amt,
               Sum(ss_net_profit) AS profit
        FROM   store_sales,
               date_dim,
               store,
               household_demographics,
               customer_address
        WHERE  store_sales.ss_sold_date_sk = date_dim.d_date_sk
               AND store_sales.ss_store_sk = store.s_store_sk
               AND store_sales.ss_hdemo_sk = household_demographics.hd_demo_sk
               AND store_sales.ss_addr_sk = customer_address.ca_address_sk
               AND ( household_demographics.hd_dep_count < 5
                      OR household_demographics.hd_vehicle_count < 2 )
               AND date_dim.d_day_name IN ( 'Monday', 'Tuesday', 'Wednesday',
                                            'Thursday' )
               AND date_dim.d_year IN ( 1999, 2000, 2001 )
        GROUP  BY ss_ticket_number,
                  ss_customer_sk,
                  ss_addr_sk,
                  ca_city) dn,
       customer,
       customer_address current_addr
WHERE  ss_customer_sk = c_customer_sk
       AND customer.c_current_addr_sk = current_addr.ca_address_sk
       AND current_addr.ca_city <> bought_city
ORDER  BY c_last_name,
          c_first_name,
          ca_city,
          bought_city,
          ss_ticket_number
```

bquery19.sql Raw

```
SELECT c_last_name,
       c_first_name,
       ca_city,
       bought_city,
       ss_ticket_number,
       amt,
       profit
FROM   (SELECT ss_ticket_number,
               ss_customer_sk,
               ca_city            AS bought_city,
               Sum(ss_coupon_amt) AS amt,
               Sum(ss_net_profit) AS profit
        FROM   store_sales,
               date_dim,
               store,
               household_demographics,
               customer_address
        WHERE  store_sales.ss_sold_date_sk = date_dim.d_date_sk
               AND store_sales.ss_store_sk = store.s_store_sk
               AND store_sales.ss_hdemo_sk = household_demographics.hd_demo_sk
               AND store_sales.ss_addr_sk = customer_address.ca_address_sk
               AND ( household_demographics.hd_dep_count < 5
                      OR household_demographics.hd_vehicle_count < 2 )
               AND ( date_dim.d_day_name_Monday=1 OR
                                    date_dim.d_day_name_Tuesday=1 OR
                       date_dim.d_day_name_Wednesday=1 OR
                        date_dim.d_day_name_Thursday=1)
               AND date_dim.d_year IN ( 1999, 2000, 2001 )
        GROUP  BY ss_ticket_number,
                  ss_customer_sk,
                  ss_addr_sk,
                  ca_city) dn,
       customer,
       customer_address current_addr
WHERE  ss_customer_sk = c_customer_sk
       AND customer.c_current_addr_sk = current_addr.ca_address_sk
       AND current_addr.ca_city <> bought_city
ORDER  BY c_last_name,
          c_first_name,
          ca_city,
          bought_city,
          ss_ticket_number
```

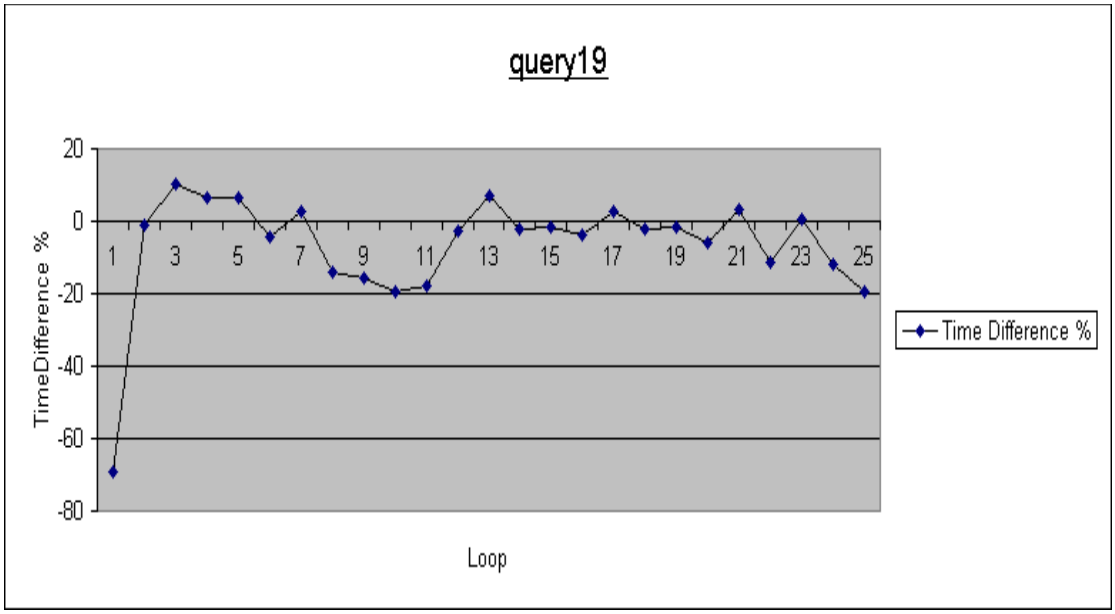
Μετατροπές SQL

d_day_name:

d_day_name IN (v1, v2,v3) <->

d_day_name_v1 = 1 OR d_day_name_v2 = 1 OR d_day_name_v3 = 1

Διάγραμμα Ποσοστιαίας Χρονικής Απόκλισης



Σχόλια

Στην περίπτωση αυτή το attribute που μετασχηματίστηκε εμπλέκεται σε ένα εμφωλευμένο IN clause. Οι χρόνοι που καταγράφηκαν είναι πολύ κοντά μεταξύ τους και υπάρχουν αρκετές εναλλαγές προσήμου. Παρά το γεγονός ότι οι περιπτώσεις που η booleanized βάση υπερτερεί είναι οριακά περισσότερες, δεν μπορούμε να ισχυριστούμε ότι ο μετασχηματισμός είχε ορατή θετική επίδραση στην επίδοση του query.

QUERY 20

Basic SQL

query20.sql Raw

```sql
 SELECT Sum(ss_quantity)
FROM   store_sales,
       store,
       customer_demographics,
       customer_address,
       date_dim
WHERE  s_store_sk = ss_store_sk
       AND ss_sold_date_sk = d_date_sk
       AND d_year > 2000
       AND ( ( cd_demo_sk = ss_cdemo_sk
               AND cd_marital_status = 'married'
               AND cd_education_status = 'university'
               AND ss_sales_price BETWEEN 25000.00 AND 50000.00 )
              OR ( cd_demo_sk = ss_cdemo_sk
                   AND cd_marital_status = 'married'
                   AND cd_education_status = 'university'
                   AND ss_sales_price BETWEEN 0.00 AND 25000.00 )
              OR ( cd_demo_sk = ss_cdemo_sk
                   AND cd_marital_status = 'married'
                   AND cd_education_status = 'university'
                   AND ss_sales_price BETWEEN 75000.00 AND 100000.00 ) );
```

Booleanized SQL

bquery20.sql Raw

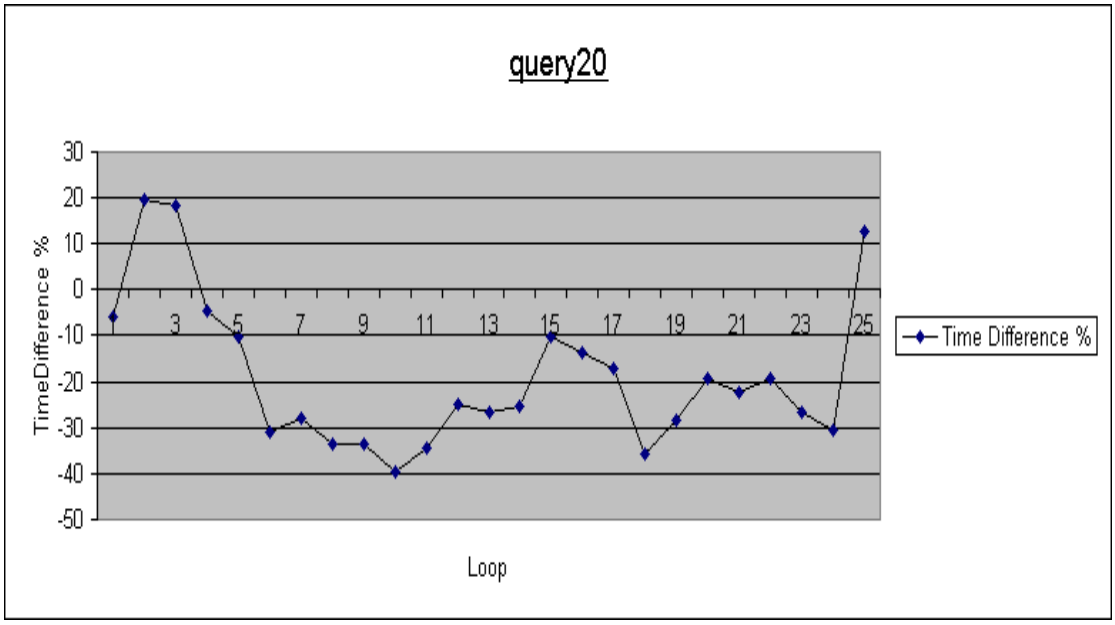
```sql
 SELECT Sum(ss_quantity)
FROM   store_sales,
       store,
       customer_demographics,
       customer_address,
       date_dim
WHERE  s_store_sk = ss_store_sk
       AND ss_sold_date_sk = d_date_sk
       AND d_year > 2000
       AND ( ( cd_demo_sk = ss_cdemo_sk
               AND cd_marital_status_married=1
               AND cd_education_status_university=1
               AND ss_sales_price BETWEEN 25000.00 AND 50000.00 )
              OR ( cd_demo_sk = ss_cdemo_sk
                   AND cd_marital_status_married=1
                   AND cd_education_status_university=1
                   AND ss_sales_price BETWEEN 0.00 AND 25000.00 )
              OR ( cd_demo_sk = ss_cdemo_sk
                   AND cd_marital_status_married=1
                   AND cd_education_status_university=1
                   AND ss_sales_price BETWEEN 75000.00 AND 100000.00 ) );
```

Μετατροπές SQL

cd_marital_status: cd_marital_status = 'Value' <-> cd_marital_status_Value = 1

cd_education_status: cd_education_status = 'Value' <-> cd_education_status_Value = 1

Διάγραμμα Ποσοστιαίας Χρονικής Απόκλισης



Σχόλια

Στην περίπτωση αυτή, ο μετασχηματισμός δυο attributes που εμπλέκονται σε ένα σύνθετο WHERE clause φαίνεται πως επέδρασε θετικά στην εκτέλεση του query, αφού σχεδόν στο σύνολο των επαναλήψεων ο χρόνος εκτέλεσης του query στην booleanized βάση δεδομένων ήταν μικρότερος.

QUERY 21

Basic SQL

query21.sql Raw

```
select
 'web' as channel
 ,web.item
 ,web.return_ratio
 ,web.return_rank
 ,web.currency_rank
 from (
   select
    item
   ,return_ratio
   ,currency_ratio
   ,@return_rank := @return_rank + 1 AS return_rank
   ,@currency_rank := @currency_rank + 1 AS currency_rank
   from
   (SELECT @return_rank := 0 ) r1,
   (SELECT @currency_rank := 0 ) r2,

   (  select ws.ws_item_sk as item
     ,(cast(sum(coalesce(wr.wr_return_quantity,0)) as dec(15,4))/
     cast(sum(coalesce(ws.ws_quantity,0)) as dec(15,4) )) as return_ratio
     ,(cast(sum(coalesce(wr.wr_return_amt,0)) as dec(15,4))/
     cast(sum(coalesce(ws.ws_net_paid,0)) as dec(15,4) )) as currency_ratio
     from
      web_sales ws left outer join web_returns wr
       on (ws.ws_order_number = wr.wr_order_number and
       ws.ws_item_sk = wr.wr_item_sk)
                 ,date_dim
     where
       wr.wr_return_amt > 10000
       and ws.ws_net_profit > 1
                         and ws.ws_net_paid > 0
                         and ws.ws_quantity > 0
                         and ws_sold_date_sk = d_date_sk
                         and d_year > 2000
                         and d_month_name = 'December'
     group by ws.ws_item_sk
   ) in_web
 ) web
 where
 (
 web.return_rank <= 10
 or
 web.currency_rank <= 10
 )
 union
 select
 'catalog' as channel
 ,catalog.item
 ,catalog.return_ratio
 ,catalog.return_rank
 ,catalog.currency_rank
 from (
   select
    item
   ,return_ratio
   ,currency_ratio
   ,@return_rank := @return_rank + 1 AS return_rank
   ,@currency_rank := @currency_rank + 1 AS currency_rank
   from
   (SELECT @return_rank := 0 ) r1,
   (SELECT @currency_rank := 0 ) r2,
   (  select
     cs.cs_item_sk as item
     ,(cast(sum(coalesce(cr.cr_return_quantity,0)) as dec(15,4))/
     cast(sum(coalesce(cs.cs_quantity,0)) as dec(15,4) )) as return_ratio
```

```
    ,(cast(sum(coalesce(cr.cr_return_amount,0)) as dec(15,4))/
    cast(sum(coalesce(cs.cs_net_paid,0)) as dec(15,4) )) as currency_ratio
    from
    catalog_sales cs left outer join catalog_returns cr
      on (cs.cs_order_number = cr.cr_order_number and
      cs.cs_item_sk = cr.cr_item_sk)
               ,date_dim
    where
      cr.cr_return_amount > 10000
      and cs.cs_net_profit > 1
                      and cs.cs_net_paid > 0
                      and cs.cs_quantity > 0
                      and cs_sold_date_sk = d_date_sk
                      and d_year > 2000
                      and d_month_name = 'December'
                group by cs.cs_item_sk
  ) in_cat
) catalog
where
(
catalog.return_rank <= 10
or
catalog.currency_rank <=10
)
union
select
'store' as channel
,store.item
,store.return_ratio
,store.return_rank
,store.currency_rank
from (
  select
   item
  ,return_ratio
  ,currency_ratio
  ,@return_rank := @return_rank + 1 AS return_rank
  ,@currency_rank := @currency_rank + 1 AS currency_rank
  from
  (SELECT @return_rank := 0 ) r1,
  (SELECT @currency_rank := 0 ) r2,
  (  select sts.ss_item_sk as item
    ,(cast(sum(coalesce(sr.sr_return_quantity,0)) as dec(15,4))/cast(sum(coalesce(sts.ss_quantity,0))
    as dec(15,4))) as return_ratio
    ,(cast(sum(coalesce(sr.sr_return_amt,0)) as dec(15,4))/cast(sum(coalesce(sts.ss_net_paid,0))
    as dec(15,4))) as currency_ratio
    from
    store_sales sts left outer join store_returns sr
      on (sts.ss_ticket_number = sr.sr_ticket_number and sts.ss_item_sk = sr.sr_item_sk)
               ,date_dim
    where
      sr.sr_return_amt > 10000
      and sts.ss_net_profit > 1
                      and sts.ss_net_paid > 0
                      and sts.ss_quantity > 0
                      and ss_sold_date_sk = d_date_sk
                      and d_year > 2000
                      and d_month_name = 'December'
    group by sts.ss_item_sk
  ) in_store
) store
where  (
store.return_rank <= 10
or
store.currency_rank <= 10
)
order by 1,4,5
```

Booleanized SQL

bquery21.sql Raw

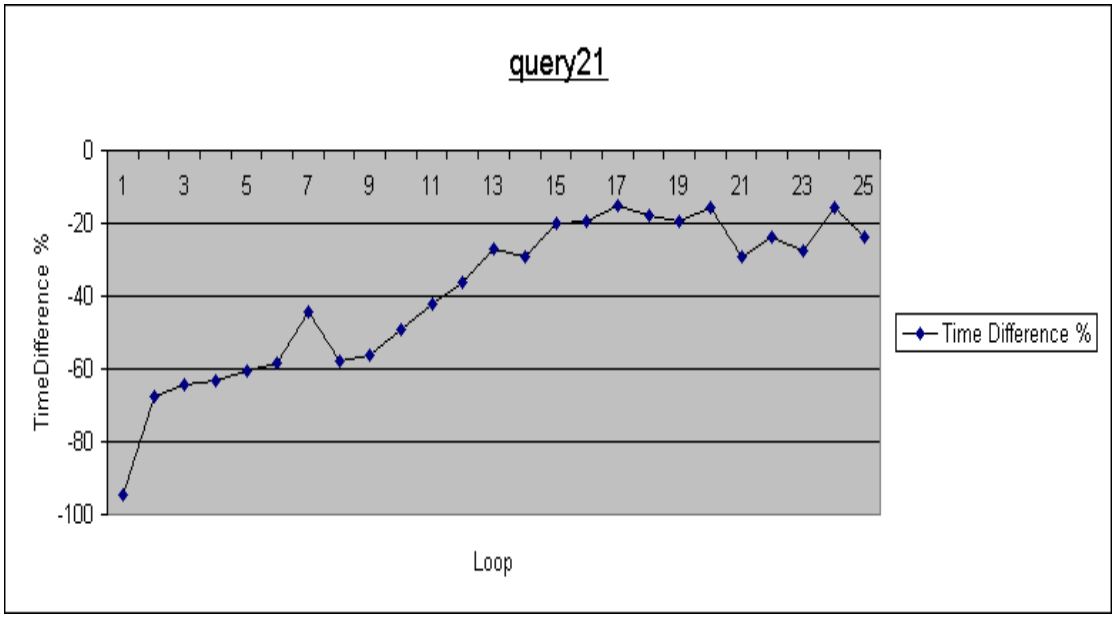
```
select
 'web' as channel
 ,web.item
 ,web.return_ratio
 ,web.return_rank
 ,web.currency_rank
 from (
   select
    item
   ,return_ratio
   ,currency_ratio
   ,@return_rank := @return_rank + 1 AS return_rank
   ,@currency_rank := @currency_rank + 1 AS currency_rank
   from
   (SELECT @return_rank := 0 ) r1,
   (SELECT @currency_rank := 0 ) r2,

   (  select ws.ws_item_sk as item
     ,(cast(sum(coalesce(wr.wr_return_quantity,0)) as dec(15,4))/
     cast(sum(coalesce(ws.ws_quantity,0)) as dec(15,4) )) as return_ratio
     ,(cast(sum(coalesce(wr.wr_return_amt,0)) as dec(15,4))/
     cast(sum(coalesce(ws.ws_net_paid,0)) as dec(15,4) )) as currency_ratio
     from
      web_sales ws left outer join web_returns wr
       on (ws.ws_order_number = wr.wr_order_number and
       ws.ws_item_sk = wr.wr_item_sk)
                 ,date_dim
     where
       wr.wr_return_amt > 10000
       and ws.ws_net_profit > 1
                         and ws.ws_net_paid > 0
                         and ws.ws_quantity > 0
                         and ws_sold_date_sk = d_date_sk
                         and d_year > 2000
                         and d_month_name_December=1
     group by ws.ws_item_sk
   ) in_web
 ) web
 where
 (
 web.return_rank <= 10
 or
 web.currency_rank <= 10
 )
 union
 select
 'catalog' as channel
 ,catalog.item
 ,catalog.return_ratio
 ,catalog.return_rank
 ,catalog.currency_rank
 from (
   select
    item
   ,return_ratio
   ,currency_ratio
   ,@return_rank := @return_rank + 1 AS return_rank
   ,@currency_rank := @currency_rank + 1 AS currency_rank
   from
   (SELECT @return_rank := 0 ) r1,
   (SELECT @currency_rank := 0 ) r2,
   (  select
     cs.cs_item_sk as item
     ,(cast(sum(coalesce(cr.cr_return_quantity,0)) as dec(15,4))/
     cast(sum(coalesce(cs.cs_quantity,0)) as dec(15,4) )) as return_ratio
     ,(cast(sum(coalesce(cr.cr_return_amount,0)) as dec(15,4))/
```

```
    cast(sum(coalesce(cs.cs_net_paid,0)) as dec(15,4) )) as currency_ratio
    from
    catalog_sales cs left outer join catalog_returns cr
      on (cs.cs_order_number = cr.cr_order_number and
      cs.cs_item_sk = cr.cr_item_sk)
               ,date_dim
    where
      cr.cr_return_amount > 10000
      and cs.cs_net_profit > 1
                        and cs.cs_net_paid > 0
                        and cs.cs_quantity > 0
                        and cs_sold_date_sk = d_date_sk
                        and d_year > 2000
                        and d_month_name_December=1
                group by cs.cs_item_sk
  ) in_cat
) catalog
where
(
catalog.return_rank <= 10
or
catalog.currency_rank <=10
)
union
select
'store' as channel
,store.item
,store.return_ratio
,store.return_rank
,store.currency_rank
from (
  select
   item
  ,return_ratio
  ,currency_ratio
  ,@return_rank := @return_rank + 1 AS return_rank
  ,@currency_rank := @currency_rank + 1 AS currency_rank
  from
  (SELECT @return_rank := 0 ) r1,
  (SELECT @currency_rank := 0 ) r2,
  (  select sts.ss_item_sk as item
    ,(cast(sum(coalesce(sr.sr_return_quantity,0)) as dec(15,4))/cast(sum(coalesce(sts.ss_quantity,0))
               as dec(15,4) )) as return_ratio
    ,(cast(sum(coalesce(sr.sr_return_amt,0)) as dec(15,4))/cast(sum(coalesce(sts.ss_net_paid,0))
               as dec(15,4) )) as currency_ratio
    from
    store_sales sts left outer join store_returns sr
      on (sts.ss_ticket_number = sr.sr_ticket_number and sts.ss_item_sk = sr.sr_item_sk)
               ,date_dim
    where
      sr.sr_return_amt > 10000
      and sts.ss_net_profit > 1
                        and sts.ss_net_paid > 0
                        and sts.ss_quantity > 0
                        and ss_sold_date_sk = d_date_sk
                        and d_year > 2000
                        and d_month_name_December=1
    group by sts.ss_item_sk
  ) in_store
) store
where  (
store.return_rank <= 10
or
store.currency_rank <= 10
)
order by 1,4,5
```

Μετατροπές SQL

d_month_name: d_month_name = 'Value' <-> d_month_name_Value = 1

Διάγραμμα Ποσοστιαίας Χρονικής Απόκλισης



Σχόλια

Όπως φαίνεται εύκολα στο διάγραμμα, το query αυτό αποτελεί ένα ακόμα παράδειγμα στο οποίο ο μετασχηματισμός επέδρασε θετικά στον χρόνο εκτέλεσης, αφού στο σύνολο των επαναλήψεων, το query εκτελέστηκε γρηγορότερα στην booleanized βάση δεδομένων. Για άλλη μια φορά το booleanized attribute εμπλέκεται σε εμφωλευμένο (και μάλλιστα επαναλαμβανόμενο) WHERE clause.

QUERY 22

Basic SQL

query22.sql Raw

```sql
SELECT s_store_name,
       s_company_id,
       s_street_number,
       s_street_name,
       s_street_type,
       s_suite_number,
       s_city,
       s_county,
       s_state,
       s_zip,
       Sum(CASE
             WHEN ( sr_returned_date_sk - ss_sold_date_sk <= 30 ) THEN 1
             ELSE 0
           END) AS '30 days',
       Sum(CASE
             WHEN ( sr_returned_date_sk - ss_sold_date_sk > 30 )
                  AND ( sr_returned_date_sk - ss_sold_date_sk <= 60 ) THEN 1
             ELSE 0
           END) AS '31-60 days',
       Sum(CASE
             WHEN ( sr_returned_date_sk - ss_sold_date_sk > 60 )
                  AND ( sr_returned_date_sk - ss_sold_date_sk <= 90 ) THEN 1
             ELSE 0
           END) AS '61-90 days',
       Sum(CASE
             WHEN ( sr_returned_date_sk - ss_sold_date_sk > 90 )
                  AND ( sr_returned_date_sk - ss_sold_date_sk <= 120 ) THEN 1
             ELSE 0
           END) AS '91-120 days',
       Sum(CASE
             WHEN ( sr_returned_date_sk - ss_sold_date_sk > 120 ) THEN 1
             ELSE 0
           END) AS '>120 days'
FROM   store_sales,
       store_returns,
       store,
       date_dim d1,
       date_dim d2
WHERE  d2.d_year > 2000
       AND d2.d_month_name = 'May'
       AND ss_ticket_number = sr_ticket_number
       AND ss_item_sk = sr_item_sk
       AND ss_sold_date_sk = d1.d_date_sk
       AND sr_returned_date_sk = d2.d_date_sk
       AND ss_customer_sk = sr_customer_sk
       AND ss_store_sk = s_store_sk
GROUP  BY s_store_name,
          s_company_id,
          s_street_number,
          s_street_name,
          s_street_type,
          s_suite_number,
          s_city,
          s_county,
          s_state,
          s_zip
ORDER  BY s_store_name,
          s_company_id,
          s_street_number,
          s_street_name,
          s_street_type,
          s_suite_number,
          s_city,
          s_county,
          s_state,
          s_zip;
```

Booleanized SQL

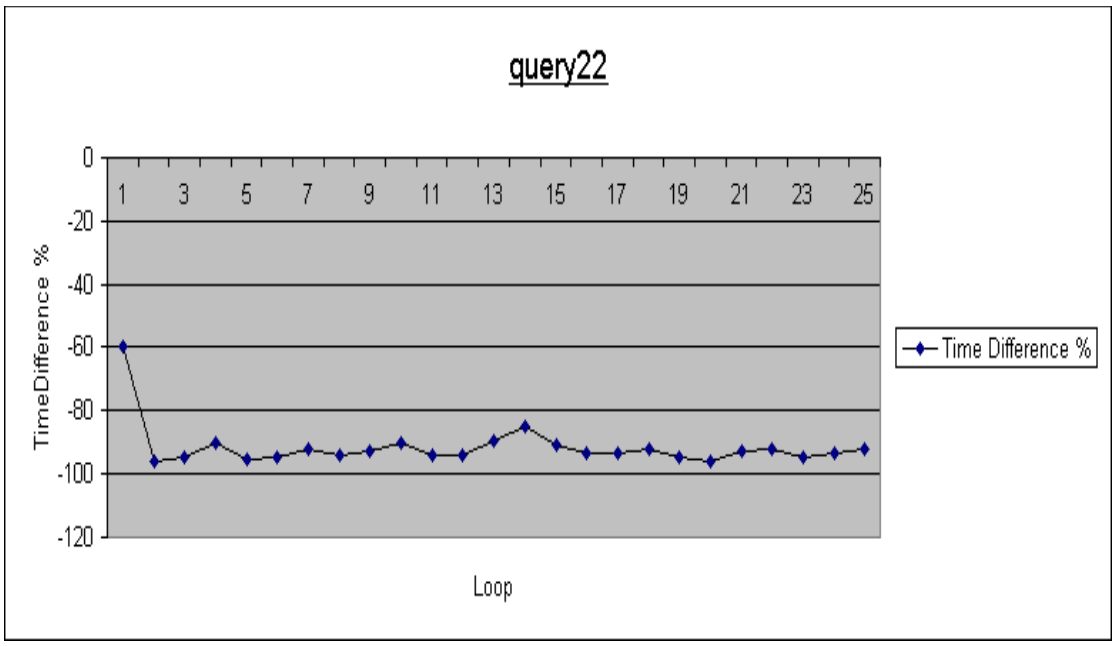
bquery22.sql Raw

```
SELECT s_store_name,
       s_company_id,
       s_street_number,
       s_street_name,
       s_street_type,
       s_suite_number,
       s_city,
       s_county,
       s_state,
       s_zip,
       Sum(CASE
             WHEN ( sr_returned_date_sk - ss_sold_date_sk <= 30 ) THEN 1
             ELSE 0
           END) AS '30 days',
       Sum(CASE
             WHEN ( sr_returned_date_sk - ss_sold_date_sk > 30 )
                  AND ( sr_returned_date_sk - ss_sold_date_sk <= 60 ) THEN 1
             ELSE 0
           END) AS '31-60 days',
       Sum(CASE
             WHEN ( sr_returned_date_sk - ss_sold_date_sk > 60 )
                  AND ( sr_returned_date_sk - ss_sold_date_sk <= 90 ) THEN 1
             ELSE 0
           END) AS '61-90 days',
       Sum(CASE
             WHEN ( sr_returned_date_sk - ss_sold_date_sk > 90 )
                  AND ( sr_returned_date_sk - ss_sold_date_sk <= 120 ) THEN 1
             ELSE 0
           END) AS '91-120 days',
       Sum(CASE
             WHEN ( sr_returned_date_sk - ss_sold_date_sk > 120 ) THEN 1
             ELSE 0
           END) AS '>120 days'
FROM   store_sales,
       store_returns,
       store,
       date_dim d1,
       date_dim d2
WHERE  d2.d_year > 2000
       AND d2.d_month_name_May=1
       AND ss_ticket_number = sr_ticket_number
       AND ss_item_sk = sr_item_sk
       AND ss_sold_date_sk = d1.d_date_sk
       AND sr_returned_date_sk = d2.d_date_sk
       AND ss_customer_sk = sr_customer_sk
       AND ss_store_sk = s_store_sk
GROUP  BY s_store_name,
          s_company_id,
          s_street_number,
          s_street_name,
          s_street_type,
          s_suite_number,
          s_city,
          s_county,
          s_state,
          s_zip
ORDER  BY s_store_name,
          s_company_id,
          s_street_number,
          s_street_name,
          s_street_type,
          s_suite_number,
          s_city,
          s_county,
          s_state,
          s_zip;
```

Μετατροπές SQL

d_month_name: d_month_name = 'Value' <-> d_month_name_Value = 1

Διάγραμμα Ποσοστιαίας Χρονικής Απόκλισης



Σχόλια

Όπως και στο προηγούμενο query, ο μετασχηματισμός του attribute d_month_name που έχει δώδεκα πιθανές τιμές και εμπλέκεται στο where clause του query, φαίνεται να επηρέασε πολύ θετικά την εκτέλεση του query, αφού σε όλες τις επαναλήψεις ο χρόνος εκτέλεσης στην booleanized βάση δεδομένων είναι αισθητά μικρότερος.

QUERY 23

Basic SQL

query23.sql Raw

```sql
SELECT dt.d_year,
       item.i_brand_id          AS brand_id,
       item.i_brand             AS brand,
       Sum(ss_ext_sales_price) AS ext_price
FROM   date_dim dt,
       store_sales,
       item
WHERE  dt.d_date_sk = store_sales.ss_sold_date_sk
       AND store_sales.ss_item_sk = item.i_item_sk
       AND dt.d_month_name = 'September'
       AND dt.d_year > 2000
GROUP  BY dt.d_year,
          item.i_brand,
          item.i_brand_id
ORDER  BY dt.d_year,
          ext_price DESC,
          brand_id;
```

Booleanized SQL

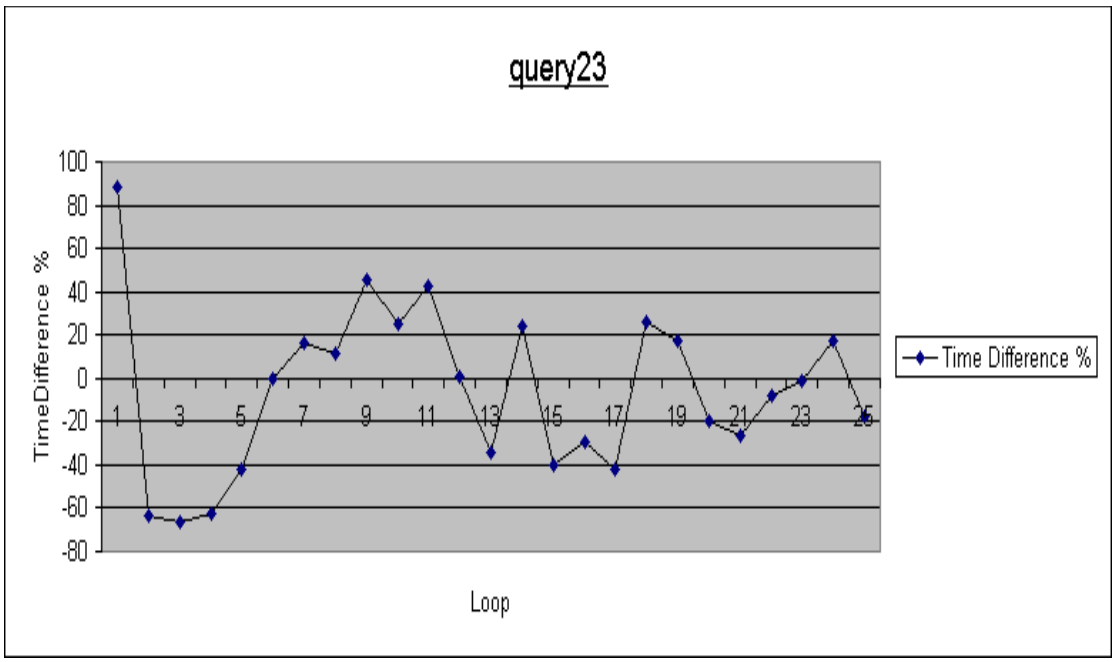
bquery23.sql Raw

```sql
SELECT dt.d_year,
       item.i_brand_id          AS brand_id,
       item.i_brand             AS brand,
       Sum(ss_ext_sales_price) AS ext_price
FROM   date_dim dt,
       store_sales,
       item
WHERE  dt.d_date_sk = store_sales.ss_sold_date_sk
       AND store_sales.ss_item_sk = item.i_item_sk
       AND dt.d_month_name_September=1
       AND dt.d_year > 2000
GROUP  BY dt.d_year,
          item.i_brand,
          item.i_brand_id
ORDER  BY dt.d_year,
          ext_price DESC,
          brand_id;
```

Μετατροπές SQL

d_month_name: d_month_name = 'Value' <-> d_month_name_Value = 1

Διάγραμμα Ποσοστιαίας Χρονικής Απόκλισης



Σχόλια

Πρόκειται για ένα query που καταδεικνύει την απρόβλεπτη επίδραση του μετασχηματισμού στην επίδοση. Πολλές εναλλαγές προσήμου, μαρτυρούν ότι ο μετασχηματισμός δεν επέδρασε προς τη μια ή την άλλη κατεύθυνση.

QUERY 24

Basic SQL

query24.sql Raw

```
 SELECT i_brand_id              AS brand_id,
       i_brand                AS brand,
       Sum(ss_ext_sales_price) AS ext_price
FROM   date_dim,
       store_sales,
       item
WHERE  d_date_sk = ss_sold_date_sk
       AND ss_item_sk = i_item_sk
       AND d_month_name = 'July'
       AND d_year > 2000
GROUP  BY i_brand,
          i_brand_id
ORDER  BY ext_price DESC,
          i_brand_id
```

Booleanized SQL

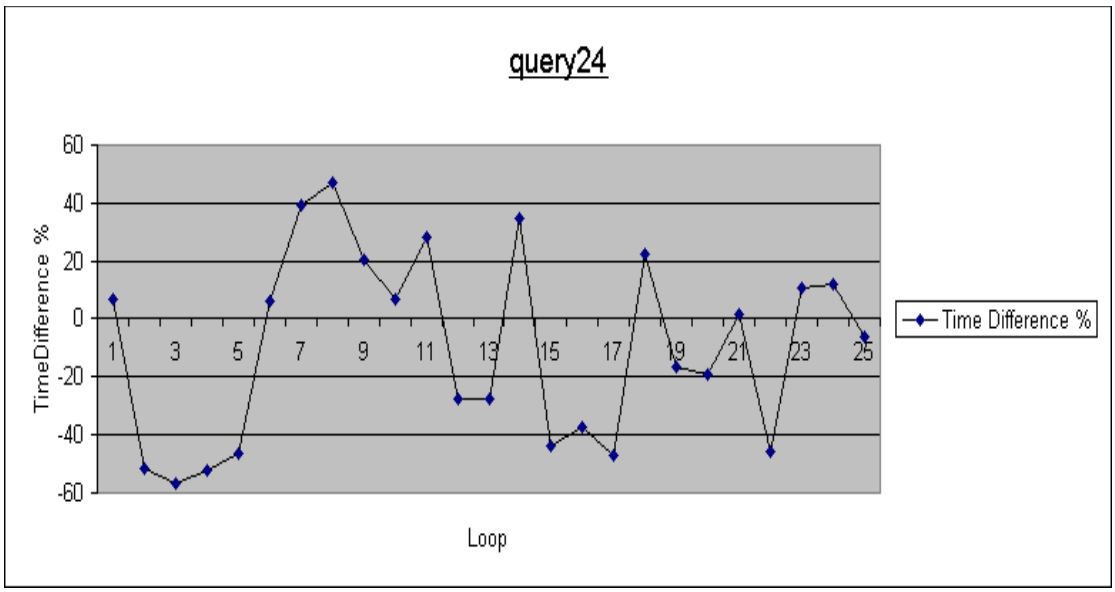
bquery24.sql Raw

```
 SELECT i_brand_id              AS brand_id,
       i_brand                AS brand,
       Sum(ss_ext_sales_price) AS ext_price
FROM   date_dim,
       store_sales,
       item
WHERE  d_date_sk = ss_sold_date_sk
       AND ss_item_sk = i_item_sk
       AND d_month_name_July=1
       AND d_year > 2000
GROUP  BY i_brand,
          i_brand_id
ORDER  BY ext_price DESC,
          i_brand_id
```

Μετατροπές SQL

d_month_name: d_month_name = 'Value' <-> d_month_name_Value = 1

Διάγραμμα Ποσοστιαίας Χρονικής Απόκλισης



Σχόλια

Το συγκεκριμένο είναι χαμηλής υπολογιστικής πολυπλοκότητας και η αλλαγή που επέφερε ο μετασχηματισμός στη δομή του ανεπαίσθητη. Οι συχνές εναλλαγές προσήμου στο διάγραμμα δείχνουν ότι ο μετασχηματισμός δεν επέδρασε προς τη μια ή την άλλη κατεύθυνση, ούτε σε αυτό το query.

QUERY 25

Basic SQL

query25.sql Raw

```sql
select cs1.product_name,cs1.store_name
     ,cs1.store_zip,cs1.b_street_number
     ,cs1.b_streen_name,cs1.b_city
     ,cs1.b_zip,cs1.c_street_number
     ,cs1.c_street_name,cs1.c_city
     ,cs1.c_zip,cs1.syear
     ,cs1.cnt,cs1.s1,cs1.s2,cs1.s3
     ,cs2.s1,cs2.s2,cs2.s3,cs2.syear,cs2.cnt

from
                (select i_product_name product_name
     ,i_item_sk item_sk
     ,s_store_name store_name
     ,s_zip store_zip
     ,ad1.ca_street_number b_street_number
     ,ad1.ca_street_name b_streen_name
     ,ad1.ca_city b_city
     ,ad1.ca_zip b_zip
     ,ad2.ca_street_number c_street_number
     ,ad2.ca_street_name c_street_name
     ,ad2.ca_city c_city
     ,ad2.ca_zip c_zip
     ,d1.d_year as syear
     ,d2.d_year as fsyear
     ,d3.d_year s2year
     ,count(*) as cnt
     ,sum(ss_wholesale_cost) as s1
     ,sum(ss_list_price) as s2
     ,sum(ss_coupon_amt) as s3
  FROM   store_sales
        ,store_returns
        , (select cs_item_sk
        ,sum(cs_ext_list_price) as sale,sum(cr_refunded_cash+cr_reversed_charge+cr_store_credit) as refund
  from catalog_sales
      ,catalog_returns
  where cs_item_sk = cr_item_sk
    and cs_order_number = cr_order_number
  group by cs_item_sk
  having sum(cs_ext_list_price)>2*sum(cr_refunded_cash+cr_reversed_charge+cr_store_credit)) as cs_ui
        ,date_dim d1
        ,date_dim d2
        ,date_dim d3
        ,store
        ,customer
        ,customer_demographics cd1
        ,customer_demographics cd2
        ,promotion
        ,household_demographics hd1
        ,household_demographics hd2
        ,customer_address ad1
        ,customer_address ad2
        ,income_band ib1
        ,income_band ib2
        ,item
  WHERE  ss_store_sk = s_store_sk AND
         ss_sold_date_sk = d1.d_date_sk AND
         ss_customer_sk = c_customer_sk AND
         ss_cdemo_sk= cd1.cd_demo_sk AND
         ss_hdemo_sk = hd1.hd_demo_sk AND
         ss_addr_sk = ad1.ca_address_sk and
         ss_item_sk = i_item_sk and
         ss_item_sk = sr_item_sk and
         ss_ticket_number = sr_ticket_number and
         ss_item_sk = cs_ui.cs_item_sk and
         c_current_cdemo_sk = cd2.cd_demo_sk AND
         c_current_hdemo_sk = hd2.hd_demo_sk AND
         c_current_addr_sk = ad2.ca_address_sk and
         c_first_sales_date_sk = d2.d_date_sk and
         c_first_shipto_date_sk = d3.d_date_sk and
         ss_promo_sk = p_promo_sk and
         hd1.hd_income_band_sk = ib1.ib_income_band_sk and
         hd2.hd_income_band_sk = ib2.ib_income_band_sk and
         cd1.cd_marital_status != cd2.cd_marital_status
group by i_product_name
       ,i_item_sk
       ,s_store_name
       ,s_zip
       ,ad1.ca_street_number
       ,ad1.ca_street_name
       ,ad1.ca_city
       ,ad1.ca_zip
       ,ad2.ca_street_number
       ,ad2.ca_street_name
       ,ad2.ca_city
       ,ad2.ca_zip
       ,d1.d_year
       ,d2.d_year
```

```
        ,d3.d_year) as cs1,

                (select i_product_name product_name
     ,i_item_sk item_sk
     ,s_store_name store_name
     ,s_zip store_zip
     ,ad1.ca_street_number b_street_number
     ,ad1.ca_street_name b_streen_name
     ,ad1.ca_city b_city
     ,ad1.ca_zip b_zip
     ,ad2.ca_street_number c_street_number
     ,ad2.ca_street_name c_street_name
     ,ad2.ca_city c_city
     ,ad2.ca_zip c_zip
     ,d1.d_year as syear
     ,d2.d_year as fsyear
     ,d3.d_year s2year
     ,count(*) as cnt
     ,sum(ss_wholesale_cost) as s1
     ,sum(ss_list_price) as s2
     ,sum(ss_coupon_amt) as s3
  FROM   store_sales
        ,store_returns
        , (select cs_item_sk
        ,sum(cs_ext_list_price) as sale,sum(cr_refunded_cash+cr_reversed_charge+cr_store_credit) as refund
  from catalog_sales
      ,catalog_returns
  where cs_item_sk = cr_item_sk
    and cs_order_number = cr_order_number
  group by cs_item_sk
  having sum(cs_ext_list_price)>2*sum(cr_refunded_cash+cr_reversed_charge+cr_store_credit)) as cs_ui
        ,date_dim d1
        ,date_dim d2
        ,date_dim d3
        ,store
        ,customer
        ,customer_demographics cd1
        ,customer_demographics cd2
        ,promotion
        ,household_demographics hd1
        ,household_demographics hd2
        ,customer_address ad1
        ,customer_address ad2
        ,income_band ib1
        ,income_band ib2
        ,item
  WHERE  ss_store_sk = s_store_sk AND
         ss_sold_date_sk = d1.d_date_sk AND
         ss_customer_sk = c_customer_sk AND
         ss_cdemo_sk= cd1.cd_demo_sk AND
         ss_hdemo_sk = hd1.hd_demo_sk AND
         ss_addr_sk = ad1.ca_address_sk and
         ss_item_sk = i_item_sk and
         ss_item_sk = sr_item_sk and
         ss_ticket_number = sr_ticket_number and
         ss_item_sk = cs_ui.cs_item_sk and
         c_current_cdemo_sk = cd2.cd_demo_sk AND
         c_current_hdemo_sk = hd2.hd_demo_sk AND
         c_current_addr_sk = ad2.ca_address_sk and
         c_first_sales_date_sk = d2.d_date_sk and
         c_first_shipto_date_sk = d3.d_date_sk and
         ss_promo_sk = p_promo_sk and
         hd1.hd_income_band_sk = ib1.ib_income_band_sk and
         hd2.hd_income_band_sk = ib2.ib_income_band_sk and
         cd1.cd_marital_status != cd2.cd_marital_status
group by i_product_name
       ,i_item_sk
       ,s_store_name
       ,s_zip
       ,ad1.ca_street_number
       ,ad1.ca_street_name
       ,ad1.ca_city
       ,ad1.ca_zip
       ,ad2.ca_street_number
       ,ad2.ca_street_name
       ,ad2.ca_city
       ,ad2.ca_zip
       ,d1.d_year
       ,d2.d_year
       ,d3.d_year) as cs2

where cs1.item_sk=cs2.item_sk and
     cs1.syear > 2000 and
     cs2.syear > 2000 and
     cs2.syear != cs1.syear and
     cs2.cnt <= cs1.cnt

order by cs1.product_name,cs1.store_name,cs2.cnt
```

Booleanized SQL

bquery25.sql Raw

```
select cs1.product_name,cs1.store_name
     ,cs1.store_zip,cs1.b_street_number
     ,cs1.b_streen_name,cs1.b_city
     ,cs1.b_zip,cs1.c_street_number
     ,cs1.c_street_name,cs1.c_city
     ,cs1.c_zip,cs1.syear
     ,cs1.cnt,cs1.s1,cs1.s2,cs1.s3
     ,cs2.s1,cs2.s2,cs2.s3,cs2.syear,cs2.cnt

from
                (select i_product_name product_name
     ,i_item_sk item_sk
     ,s_store_name store_name
     ,s_zip store_zip
     ,ad1.ca_street_number b_street_number
     ,ad1.ca_street_name b_streen_name
     ,ad1.ca_city b_city
     ,ad1.ca_zip b_zip
     ,ad2.ca_street_number c_street_number
     ,ad2.ca_street_name c_street_name
     ,ad2.ca_city c_city
     ,ad2.ca_zip c_zip
     ,d1.d_year as syear
     ,d2.d_year as fsyear
     ,d3.d_year s2year
     ,count(*) as cnt
     ,sum(ss_wholesale_cost) as s1
     ,sum(ss_list_price) as s2
     ,sum(ss_coupon_amt) as s3
  FROM   store_sales
       ,store_returns
       , (select cs_item_sk
       ,sum(cs_ext_list_price) as sale,sum(cr_refunded_cash+cr_reversed_charge+cr_store_credit)
                        as refund
  from catalog_sales
      ,catalog_returns
  where cs_item_sk = cr_item_sk
    and cs_order_number = cr_order_number
  group by cs_item_sk
  having sum(cs_ext_list_price)>2*sum(cr_refunded_cash+cr_reversed_charge+cr_store_credit))
       as cs_ui
       ,date_dim d1
       ,date_dim d2
       ,date_dim d3
       ,store
       ,customer
       ,customer_demographics cd1
       ,customer_demographics cd2
       ,promotion
       ,household_demographics hd1
       ,household_demographics hd2
       ,customer_address ad1
       ,customer_address ad2
       ,income_band ib1
       ,income_band ib2
       ,item
  WHERE  ss_store_sk = s_store_sk AND
         ss_sold_date_sk = d1.d_date_sk AND
         ss_customer_sk = c_customer_sk AND
         ss_cdemo_sk= cd1.cd_demo_sk AND
         ss_hdemo_sk = hd1.hd_demo_sk AND
         ss_addr_sk = ad1.ca_address_sk and
         ss_item_sk = i_item_sk and
         ss_item_sk = sr_item_sk and
         ss_ticket_number = sr_ticket_number and
         ss_item_sk = cs_ui.cs_item_sk and
         c_current_cdemo_sk = cd2.cd_demo_sk AND
         c_current_hdemo_sk = hd2.hd_demo_sk AND
         c_current_addr_sk = ad2.ca_address_sk and
         c_first_sales_date_sk = d2.d_date_sk and
         c_first_shipto_date_sk = d3.d_date_sk and
         ss_promo_sk = p_promo_sk and
         hd1.hd_income_band_sk = ib1.ib_income_band_sk and
         hd2.hd_income_band_sk = ib2.ib_income_band_sk and
         ((cd1.cd_marital_status_single = 1 AND cd2.cd_marital_status_single = 0)
                        OR (cd1.cd_marital_status_married = 1 AND cd2.cd_marital_status_married = 0)
                        OR (cd1.cd_marital_status_divorced = 1 AND cd2.cd_marital_status_divorced = 0))
group by i_product_name
       ,i_item_sk
       ,s_store_name
       ,s_zip
       ,ad1.ca_street_number
       ,ad1.ca_street_name
       ,ad1.ca_city
       ,ad1.ca_zip
       ,ad2.ca_street_number
       ,ad2.ca_street_name
       ,ad2.ca_city
       ,ad2.ca_zip
       ,d1.d_year
```

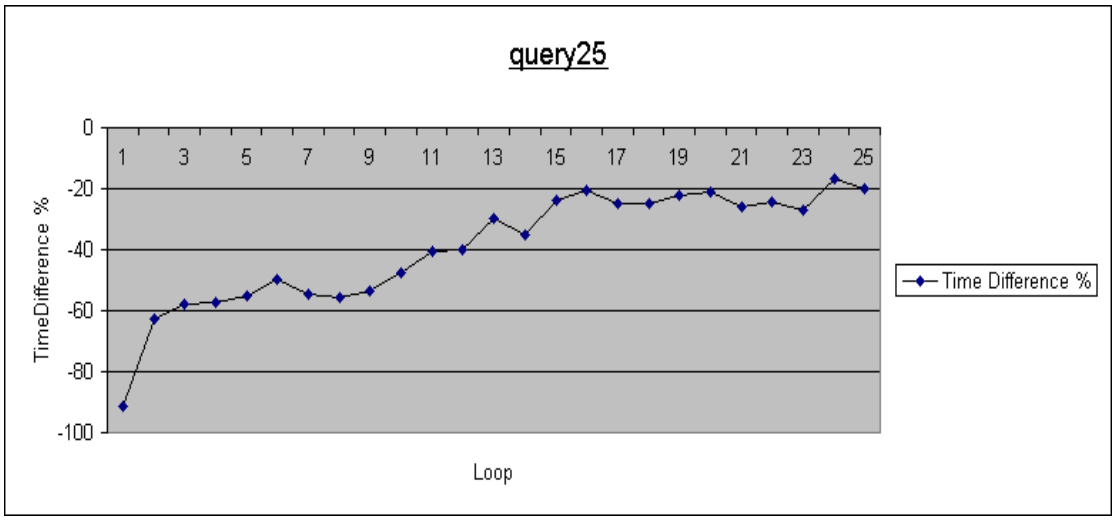
```
        ,d1.d_year
        ,d2.d_year
        ,d3.d_year) as cs1,

                (select i_product_name product_name
     ,i_item_sk item_sk
     ,s_store_name store_name
     ,s_zip store_zip
     ,ad1.ca_street_number b_street_number
     ,ad1.ca_street_name b_streen_name
     ,ad1.ca_city b_city
     ,ad1.ca_zip b_zip
     ,ad2.ca_street_number c_street_number
     ,ad2.ca_street_name c_street_name
     ,ad2.ca_city c_city
     ,ad2.ca_zip c_zip
     ,d1.d_year as syear
     ,d2.d_year as fsyear
     ,d3.d_year s2year
     ,count(*) as cnt
     ,sum(ss_wholesale_cost) as s1
     ,sum(ss_list_price) as s2
     ,sum(ss_coupon_amt) as s3
  FROM   store_sales
        ,store_returns
        , (select cs_item_sk
        ,sum(cs_ext_list_price) as sale,sum(cr_refunded_cash+cr_reversed_charge+cr_store_credit) as refund
  from catalog_sales
      ,catalog_returns
  where cs_item_sk = cr_item_sk
    and cs_order_number = cr_order_number
  group by cs_item_sk
  having sum(cs_ext_list_price)>2*sum(cr_refunded_cash+cr_reversed_charge+cr_store_credit)) as cs_ui
        ,date_dim d1
        ,date_dim d2
        ,date_dim d3
        ,store
        ,customer
        ,customer_demographics cd1
        ,customer_demographics cd2
        ,promotion
        ,household_demographics hd1
        ,household_demographics hd2
        ,customer_address ad1
        ,customer_address ad2
        ,income_band ib1
        ,income_band ib2
        ,item
  WHERE  ss_store_sk = s_store_sk AND
         ss_sold_date_sk = d1.d_date_sk AND
         ss_customer_sk = c_customer_sk AND
         ss_cdemo_sk= cd1.cd_demo_sk AND
         ss_hdemo_sk = hd1.hd_demo_sk AND
         ss_addr_sk = ad1.ca_address_sk and
         ss_item_sk = i_item_sk and
         ss_item_sk = sr_item_sk and
         ss_ticket_number = sr_ticket_number and
         ss_item_sk = cs_ui.cs_item_sk and
         c_current_cdemo_sk = cd2.cd_demo_sk AND
         c_current_hdemo_sk = hd2.hd_demo_sk AND
         c_current_addr_sk = ad2.ca_address_sk and
         c_first_sales_date_sk = d2.d_date_sk and
         c_first_shipto_date_sk = d3.d_date_sk and
         ss_promo_sk = p_promo_sk and
         hd1.hd_income_band_sk = ib1.ib_income_band_sk and
         hd2.hd_income_band_sk = ib2.ib_income_band_sk and
         ((cd1.cd_marital_status_single = 1 AND cd2.cd_marital_status_single = 0)
                     OR (cd1.cd_marital_status_married = 1 AND cd2.cd_marital_status_married = 0)
                     OR (cd1.cd_marital_status_divorced = 1 AND cd2.cd_marital_status_divorced = 0))
group by i_product_name
       ,i_item_sk
       ,s_store_name
       ,s_zip
       ,ad1.ca_street_number
       ,ad1.ca_street_name
       ,ad1.ca_city
       ,ad1.ca_zip
       ,ad2.ca_street_number
       ,ad2.ca_street_name
       ,ad2.ca_city
       ,ad2.ca_zip
       ,d1.d_year
       ,d2.d_year
       ,d3.d_year) as cs2

where cs1.item_sk=cs2.item_sk and
     cs1.syear > 2000 and
     cs2.syear > 2000 and
     cs2.syear != cs1.syear and
     cs2.cnt <= cs1.cnt

order by cs1.product_name,cs1.store_name,cs2.cnt
```

Μετατροπές SQL

cd_marital_status:

```
        cd1.cd_marital_status  !=  cd2.cd_marital_status   <->
        (cd1.cd_marital_status_Value1 = 1  AND  cd2.cd_marital_status_Value1 = 0)
OR      (cd1.cd_marital_status_Value2 = 1  AND  cd2.cd_marital_status_Value2 = 0)
OR      (cd1.cd_marital_status_Value3 = 1  AND  cd2.cd_marital_status_Value3 = 0)
```

Διάγραμμα Ποσοστιαίας Χρονικής Απόκλισης



Σχόλια

Ο μετασχηματισμός του συγκεκριμένου query ήταν ιδιαίτερος, λόγω του λογικού συνδέσμου '!=' στο αρχικό query. Τα αποτελέσματα ήταν ιδιαίτερα θετικά, αφού η Booleanized βάση δεδομένων υπερτερεί σε όλες τις επαναλήψεις.

QUERY26

Basic SQL

query26.sql Raw

```
SELECT w_warehouse_name,
       w_warehouse_sq_ft,
       w_city,
       w_county,
       w_state,
       w_country,
       year,
       Sum(jan_sales)                    AS jan_sales,
       Sum(feb_sales)                    AS feb_sales,
       Sum(mar_sales)                    AS mar_sales,
       Sum(apr_sales)                    AS apr_sales,
       Sum(may_sales)                    AS may_sales,
       Sum(jun_sales)                    AS jun_sales,
       Sum(jul_sales)                    AS jul_sales,
       Sum(aug_sales)                    AS aug_sales,
       Sum(sep_sales)                    AS sep_sales,
       Sum(oct_sales)                    AS oct_sales,
       Sum(nov_sales)                    AS nov_sales,
       Sum(dec_sales)                    AS dec_sales,
       Sum(jan_sales / w_warehouse_sq_ft) AS jan_sales_per_sq_foot,
       Sum(feb_sales / w_warehouse_sq_ft) AS feb_sales_per_sq_foot,
       Sum(mar_sales / w_warehouse_sq_ft) AS mar_sales_per_sq_foot,
       Sum(apr_sales / w_warehouse_sq_ft) AS apr_sales_per_sq_foot,
       Sum(may_sales / w_warehouse_sq_ft) AS may_sales_per_sq_foot,
       Sum(jun_sales / w_warehouse_sq_ft) AS jun_sales_per_sq_foot,
       Sum(jul_sales / w_warehouse_sq_ft) AS jul_sales_per_sq_foot,
       Sum(aug_sales / w_warehouse_sq_ft) AS aug_sales_per_sq_foot,
       Sum(sep_sales / w_warehouse_sq_ft) AS sep_sales_per_sq_foot,
       Sum(oct_sales / w_warehouse_sq_ft) AS oct_sales_per_sq_foot,
       Sum(nov_sales / w_warehouse_sq_ft) AS nov_sales_per_sq_foot,
       Sum(dec_sales / w_warehouse_sq_ft) AS dec_sales_per_sq_foot,
       Sum(jan_net)                      AS jan_net,
       Sum(feb_net)                      AS feb_net,
       Sum(mar_net)                      AS mar_net,
       Sum(apr_net)                      AS apr_net,
       Sum(may_net)                      AS may_net,
       Sum(jun_net)                      AS jun_net,
       Sum(jul_net)                      AS jul_net,
       Sum(aug_net)                      AS aug_net,
       Sum(sep_net)                      AS sep_net,
       Sum(oct_net)                      AS oct_net,
       Sum(nov_net)                      AS nov_net,
       Sum(dec_net)                      AS dec_net
FROM   ((SELECT w_warehouse_name,
                w_warehouse_sq_ft,
                w_city,
                w_county,
                w_state,
                w_country,
                d_year   AS year,
                Sum(CASE
                      WHEN d_month_name = 'January' THEN
                      ws_sales_price * ws_quantity
                      ELSE 0
                    end) AS jan_sales,
                Sum(CASE
                      WHEN d_month_name = 'February' THEN
                      ws_sales_price * ws_quantity
                      ELSE 0
                    end) AS feb_sales,
                Sum(CASE
                      WHEN d_month_name = 'March' THEN
                      ws_sales_price * ws_quantity
                      ELSE 0
                    end) AS mar_sales,
                Sum(CASE
                      WHEN d_month_name = 'April' THEN
                      ws_sales_price * ws_quantity
                      ELSE 0
                    end) AS apr_sales,
                Sum(CASE
                      WHEN d_month_name = 'May' THEN
                      ws_sales_price * ws_quantity
                      ELSE 0
                    end) AS may_sales,
                Sum(CASE
```

```
                WHEN d_month_name = 'June' THEN
                ws_sales_price * ws_quantity
                ELSE 0
            end) AS jun_sales,
        Sum(CASE
                WHEN d_month_name = 'July' THEN
                ws_sales_price * ws_quantity
                ELSE 0
            end) AS jul_sales,
        Sum(CASE
                WHEN d_month_name = 'August' THEN
                ws_sales_price * ws_quantity
                ELSE 0
            end) AS aug_sales,
        Sum(CASE
                WHEN d_month_name = 'September' THEN
                ws_sales_price * ws_quantity
                ELSE 0
            end) AS sep_sales,
        Sum(CASE
                WHEN d_month_name = 'October' THEN
                ws_sales_price * ws_quantity
                ELSE 0
            end) AS oct_sales,
        Sum(CASE
                WHEN d_month_name = 'November' THEN
                ws_sales_price * ws_quantity
                ELSE 0
            end) AS nov_sales,
        Sum(CASE
                WHEN d_month_name = 'December' THEN
                ws_sales_price * ws_quantity
                ELSE 0
            end) AS dec_sales,
        Sum(CASE
                WHEN d_month_name = 'January' THEN
                ws_net_paid * ws_quantity
                ELSE 0
            end) AS jan_net,
        Sum(CASE
                WHEN d_month_name = 'February' THEN
                ws_net_paid * ws_quantity
                ELSE 0
            end) AS feb_net,
        Sum(CASE
                WHEN d_month_name = 'March' THEN ws_net_paid * ws_quantity
                ELSE 0
            end) AS mar_net,
        Sum(CASE
                WHEN d_month_name = 'April' THEN ws_net_paid * ws_quantity
                ELSE 0
            end) AS apr_net,
        Sum(CASE
                WHEN d_month_name = 'May' THEN ws_net_paid * ws_quantity
                ELSE 0
            end) AS may_net,
        Sum(CASE
                WHEN d_month_name = 'June' THEN ws_net_paid * ws_quantity
                ELSE 0
            end) AS jun_net,
        Sum(CASE
                WHEN d_month_name = 'July' THEN ws_net_paid * ws_quantity
                ELSE 0
            end) AS jul_net,
        Sum(CASE
                WHEN d_month_name = 'August' THEN
                ws_net_paid * ws_quantity
                ELSE 0
            end) AS aug_net,
        Sum(CASE
                WHEN d_month_name = 'September' THEN
                ws_net_paid * ws_quantity
                ELSE 0
            end) AS sep_net,
        Sum(CASE
                WHEN d_month_name = 'October' THEN
                ws_net_paid * ws_quantity
                ELSE 0
            end) AS oct_net
```

```
                end) AS oct_net,
            Sum(CASE
                  WHEN d_month_name = 'November' THEN
                  ws_net_paid * ws_quantity
                  ELSE 0
                end) AS nov_net,
            Sum(CASE
                  WHEN d_month_name = 'December' THEN
                  ws_net_paid * ws_quantity
                  ELSE 0
                end) AS dec_net
     FROM   web_sales,
            warehouse,
            date_dim,
            time_dim,
            ship_mode
     WHERE  ws_warehouse_sk = w_warehouse_sk
            AND ws_sold_date_sk = d_date_sk
            AND ws_sold_time_sk = t_time_sk
            AND ws_ship_mode_sk = sm_ship_mode_sk
            AND d_year = 2000
     GROUP  BY w_warehouse_name,
               w_warehouse_sq_ft,
               w_city,
               w_county,
               w_state,
               w_country,
               d_year)
    UNION ALL
    (SELECT w_warehouse_name,
            w_warehouse_sq_ft,
            w_city,
            w_county,
            w_state,
            w_country,
            d_year   AS year,
            Sum(CASE
                  WHEN d_month_name = 'January' THEN cs_sales_price * cs_quantity
                  ELSE 0
                end) AS jan_sales,
            Sum(CASE
                  WHEN d_month_name = 'February' THEN cs_sales_price * cs_quantity
                  ELSE 0
                end) AS feb_sales,
            Sum(CASE
                  WHEN d_month_name = 'March' THEN cs_sales_price * cs_quantity
                  ELSE 0
                end) AS mar_sales,
            Sum(CASE
                  WHEN d_month_name = 'April' THEN cs_sales_price * cs_quantity
                  ELSE 0
                end) AS apr_sales,
            Sum(CASE
                  WHEN d_month_name = 'May' THEN cs_sales_price * cs_quantity
                  ELSE 0
                end) AS may_sales,
            Sum(CASE
                  WHEN d_month_name = 'June' THEN cs_sales_price * cs_quantity
                  ELSE 0
                end) AS jun_sales,
            Sum(CASE
                  WHEN d_month_name = 'July' THEN cs_sales_price * cs_quantity
                  ELSE 0
                end) AS jul_sales,
            Sum(CASE
                  WHEN d_month_name = 'August' THEN cs_sales_price * cs_quantity
                  ELSE 0
                end) AS aug_sales,
            Sum(CASE
                  WHEN d_month_name = 'September' THEN cs_sales_price * cs_quantity
                  ELSE 0
                end) AS sep_sales,
            Sum(CASE
                  WHEN d_month_name = 'October' THEN cs_sales_price * cs_quantity
                  ELSE 0
                end) AS oct_sales,
            Sum(CASE
                  WHEN d_month_name = 'November' THEN cs_sales_price * cs_quantity
                  ELSE 0
```

```
                    end) AS nov_sales,
                Sum(CASE
                      WHEN d_month_name = 'December' THEN cs_sales_price * cs_quantity
                      ELSE 0
                    end) AS dec_sales,
                Sum(CASE
                      WHEN d_month_name = 'January' THEN cs_net_paid * cs_quantity
                      ELSE 0
                    end) AS jan_net,
                Sum(CASE
                      WHEN d_month_name = 'February' THEN cs_net_paid * cs_quantity
                      ELSE 0
                    end) AS feb_net,
                Sum(CASE
                      WHEN d_month_name = 'March' THEN cs_net_paid * cs_quantity
                      ELSE 0
                    end) AS mar_net,
                Sum(CASE
                      WHEN d_month_name = 'April' THEN cs_net_paid * cs_quantity
                      ELSE 0
                    end) AS apr_net,
                Sum(CASE
                      WHEN d_month_name = 'May' THEN cs_net_paid * cs_quantity
                      ELSE 0
                    end) AS may_net,
                Sum(CASE
                      WHEN d_month_name = 'June' THEN cs_net_paid * cs_quantity
                      ELSE 0
                    end) AS jun_net,
                Sum(CASE
                      WHEN d_month_name = 'July' THEN cs_net_paid * cs_quantity
                      ELSE 0
                    end) AS jul_net,
                Sum(CASE
                      WHEN d_month_name = 'August' THEN cs_net_paid * cs_quantity
                      ELSE 0
                    end) AS aug_net,
                Sum(CASE
                      WHEN d_month_name = 'September' THEN cs_net_paid * cs_quantity
                      ELSE 0
                    end) AS sep_net,
                Sum(CASE
                      WHEN d_month_name = 'October' THEN cs_net_paid * cs_quantity
                      ELSE 0
                    end) AS oct_net,
                Sum(CASE
                      WHEN d_month_name = 'November' THEN cs_net_paid * cs_quantity
                      ELSE 0
                    end) AS nov_net,
                Sum(CASE
                      WHEN d_month_name = 'December' THEN cs_net_paid * cs_quantity
                      ELSE 0
                    end) AS dec_net
         FROM   catalog_sales,
                warehouse,
                date_dim,
                time_dim,
                ship_mode
         WHERE  cs_warehouse_sk = w_warehouse_sk
                AND cs_sold_date_sk = d_date_sk
                AND cs_sold_time_sk = t_time_sk
                AND cs_ship_mode_sk = sm_ship_mode_sk
                AND d_year = 2000
         GROUP  BY w_warehouse_name,
                   w_warehouse_sq_ft,
                   w_city,
                   w_county,
                   w_state,
                   w_country,
                   d_year)) x
GROUP  BY w_warehouse_name,
          w_warehouse_sq_ft,
          w_city,
          w_county,
          w_state,
          w_country,
          year
ORDER  BY w_warehouse_name
```

Booleanized SQL

bquery26.sql Raw

```
SELECT w_warehouse_name,
       w_warehouse_sq_ft,
       w_city,
       w_county,
       w_state,
       w_country,
       year,
       Sum(jan_sales)                    AS jan_sales,
       Sum(feb_sales)                    AS feb_sales,
       Sum(mar_sales)                    AS mar_sales,
       Sum(apr_sales)                    AS apr_sales,
       Sum(may_sales)                    AS may_sales,
       Sum(jun_sales)                    AS jun_sales,
       Sum(jul_sales)                    AS jul_sales,
       Sum(aug_sales)                    AS aug_sales,
       Sum(sep_sales)                    AS sep_sales,
       Sum(oct_sales)                    AS oct_sales,
       Sum(nov_sales)                    AS nov_sales,
       Sum(dec_sales)                    AS dec_sales,
       Sum(jan_sales / w_warehouse_sq_ft) AS jan_sales_per_sq_foot,
       Sum(feb_sales / w_warehouse_sq_ft) AS feb_sales_per_sq_foot,
       Sum(mar_sales / w_warehouse_sq_ft) AS mar_sales_per_sq_foot,
       Sum(apr_sales / w_warehouse_sq_ft) AS apr_sales_per_sq_foot,
       Sum(may_sales / w_warehouse_sq_ft) AS may_sales_per_sq_foot,
       Sum(jun_sales / w_warehouse_sq_ft) AS jun_sales_per_sq_foot,
       Sum(jul_sales / w_warehouse_sq_ft) AS jul_sales_per_sq_foot,
       Sum(aug_sales / w_warehouse_sq_ft) AS aug_sales_per_sq_foot,
       Sum(sep_sales / w_warehouse_sq_ft) AS sep_sales_per_sq_foot,
       Sum(oct_sales / w_warehouse_sq_ft) AS oct_sales_per_sq_foot,
       Sum(nov_sales / w_warehouse_sq_ft) AS nov_sales_per_sq_foot,
       Sum(dec_sales / w_warehouse_sq_ft) AS dec_sales_per_sq_foot,
       Sum(jan_net)                      AS jan_net,
       Sum(feb_net)                      AS feb_net,
       Sum(mar_net)                      AS mar_net,
       Sum(apr_net)                      AS apr_net,
       Sum(may_net)                      AS may_net,
       Sum(jun_net)                      AS jun_net,
       Sum(jul_net)                      AS jul_net,
       Sum(aug_net)                      AS aug_net,
       Sum(sep_net)                      AS sep_net,
       Sum(oct_net)                      AS oct_net,
       Sum(nov_net)                      AS nov_net,
       Sum(dec_net)                      AS dec_net
FROM   ((SELECT w_warehouse_name,
                w_warehouse_sq_ft,
                w_city,
                w_county,
                w_state,
                w_country,
                d_year   AS year,
                Sum(CASE
                      WHEN d_month_name_January = 1 THEN
                      ws_sales_price * ws_quantity
                      ELSE 0
                    end) AS jan_sales,
                Sum(CASE
                      WHEN d_month_name_February = 1 THEN
                      ws_sales_price * ws_quantity
                      ELSE 0
                    end) AS feb_sales,
                Sum(CASE
                      WHEN d_month_name_March = 1 THEN
                      ws_sales_price * ws_quantity
                      ELSE 0
                    end) AS mar_sales,
                Sum(CASE
                      WHEN d_month_name_April = 1 THEN
                      ws_sales_price * ws_quantity
                      ELSE 0
                    end) AS apr_sales,
                Sum(CASE
                      WHEN d_month_name_May = 1 THEN
                      ws_sales_price * ws_quantity
                      ELSE 0
                    end) AS may_sales,
                Sum(CASE
```

```
77                      WHEN d_month_name_June = 1 THEN
78                      ws_sales_price * ws_quantity
79                      ELSE 0
80                  end) AS jun_sales,
81              Sum(CASE
82                      WHEN d_month_name_July = 1 THEN
83                      ws_sales_price * ws_quantity
84                      ELSE 0
85                  end) AS jul_sales,
86              Sum(CASE
87                      WHEN d_month_name_August = 1 THEN
88                      ws_sales_price * ws_quantity
89                      ELSE 0
90                  end) AS aug_sales,
91              Sum(CASE
92                      WHEN d_month_name_September = 1 THEN
93                      ws_sales_price * ws_quantity
94                      ELSE 0
95                  end) AS sep_sales,
96              Sum(CASE
97                      WHEN d_month_name_October = 1 THEN
98                      ws_sales_price * ws_quantity
99                      ELSE 0
100                 end) AS oct_sales,
101             Sum(CASE
102                     WHEN d_month_name_November = 1 THEN
103                     ws_sales_price * ws_quantity
104                     ELSE 0
105                 end) AS nov_sales,
106             Sum(CASE
107                     WHEN d_month_name_December = 1 THEN
108                     ws_sales_price * ws_quantity
109                     ELSE 0
110                 end) AS dec_sales,
111             Sum(CASE
112                     WHEN d_month_name_January = 1 THEN
113                     ws_net_paid * ws_quantity
114                     ELSE 0
115                 end) AS jan_net,
116             Sum(CASE
117                     WHEN d_month_name_February = 1 THEN
118                     ws_net_paid * ws_quantity
119                     ELSE 0
120                 end) AS feb_net,
121             Sum(CASE
122                     WHEN d_month_name_March = 1 THEN ws_net_paid * ws_quantity
123                     ELSE 0
124                 end) AS mar_net,
125             Sum(CASE
126                     WHEN d_month_name_April = 1 THEN ws_net_paid * ws_quantity
127                     ELSE 0
128                 end) AS apr_net,
129             Sum(CASE
130                     WHEN d_month_name_May = 1 THEN ws_net_paid * ws_quantity
131                     ELSE 0
132                 end) AS may_net,
133             Sum(CASE
134                     WHEN d_month_name_June = 1 THEN ws_net_paid * ws_quantity
135                     ELSE 0
136                 end) AS jun_net,
137             Sum(CASE
138                     WHEN d_month_name_July = 1 THEN ws_net_paid * ws_quantity
139                     ELSE 0
140                 end) AS jul_net,
141             Sum(CASE
142                     WHEN d_month_name_August = 1 THEN
143                     ws_net_paid * ws_quantity
144                     ELSE 0
145                 end) AS aug_net,
146             Sum(CASE
147                     WHEN d_month_name_September = 1 THEN
148                     ws_net_paid * ws_quantity
149                     ELSE 0
150                 end) AS sep_net,
151             Sum(CASE
152                     WHEN d_month_name_October = 1 THEN
153                     ws_net_paid * ws_quantity
154                     ELSE 0
```

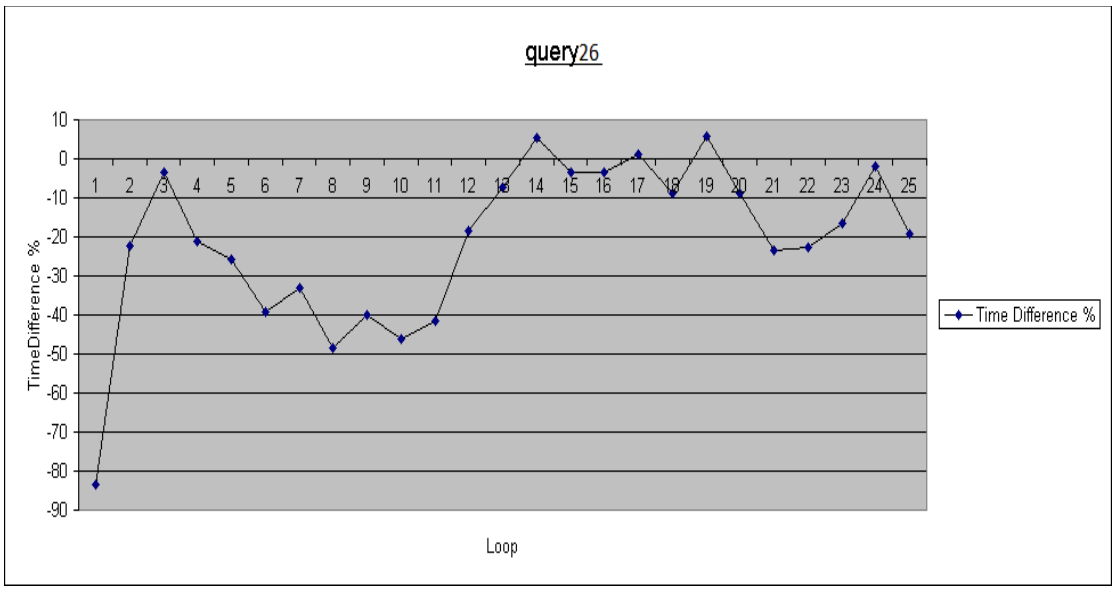
```
                end) AS oct_net,
            Sum(CASE
                  WHEN d_month_name_November = 1 THEN
                  ws_net_paid * ws_quantity
                  ELSE 0
                end) AS nov_net,
            Sum(CASE
                  WHEN d_month_name_December = 1 THEN
                  ws_net_paid * ws_quantity
                  ELSE 0
                end) AS dec_net
     FROM   web_sales,
            warehouse,
            date_dim,
            time_dim,
            ship_mode
     WHERE  ws_warehouse_sk = w_warehouse_sk
            AND ws_sold_date_sk = d_date_sk
            AND ws_sold_time_sk = t_time_sk
            AND ws_ship_mode_sk = sm_ship_mode_sk
            AND d_year = 2000
     GROUP  BY w_warehouse_name,
               w_warehouse_sq_ft,
               w_city,
               w_county,
               w_state,
               w_country,
               d_year)
    UNION ALL
    (SELECT w_warehouse_name,
            w_warehouse_sq_ft,
            w_city,
            w_county,
            w_state,
            w_country,
            d_year   AS year,
            Sum(CASE
                  WHEN d_month_name_January = 1 THEN cs_sales_price * cs_quantity
                  ELSE 0
                end) AS jan_sales,
            Sum(CASE
                  WHEN d_month_name_February = 1 THEN cs_sales_price * cs_quantity
                  ELSE 0
                end) AS feb_sales,
            Sum(CASE
                  WHEN d_month_name_March = 1 THEN cs_sales_price * cs_quantity
                  ELSE 0
                end) AS mar_sales,
            Sum(CASE
                  WHEN d_month_name_April = 1 THEN cs_sales_price * cs_quantity
                  ELSE 0
                end) AS apr_sales,
            Sum(CASE
                  WHEN d_month_name_May = 1 THEN cs_sales_price * cs_quantity
                  ELSE 0
                end) AS may_sales,
            Sum(CASE
                  WHEN d_month_name_June = 1 THEN cs_sales_price * cs_quantity
                  ELSE 0
                end) AS jun_sales,
            Sum(CASE
                  WHEN d_month_name_July = 1 THEN cs_sales_price * cs_quantity
                  ELSE 0
                end) AS jul_sales,
            Sum(CASE
                  WHEN d_month_name_August = 1 THEN cs_sales_price * cs_quantity
                  ELSE 0
                end) AS aug_sales,
            Sum(CASE
                  WHEN d_month_name_September = 1 THEN cs_sales_price * cs_quantity
                  ELSE 0
                end) AS sep_sales,
            Sum(CASE
                  WHEN d_month_name_October = 1 THEN cs_sales_price * cs_quantity
                  ELSE 0
                end) AS oct_sales,
            Sum(CASE
                  WHEN d_month_name_November = 1 THEN cs_sales_price * cs_quantity
                  ELSE 0
```

```
                    end) AS nov_sales,
                Sum(CASE
                      WHEN d_month_name_December = 1 THEN cs_sales_price * cs_quantity
                      ELSE 0
                    end) AS dec_sales,
                Sum(CASE
                      WHEN d_month_name_January = 1 THEN cs_net_paid * cs_quantity
                      ELSE 0
                    end) AS jan_net,
                Sum(CASE
                      WHEN d_month_name_February = 1 THEN cs_net_paid * cs_quantity
                      ELSE 0
                    end) AS feb_net,
                Sum(CASE
                      WHEN d_month_name_March = 1 THEN cs_net_paid * cs_quantity
                      ELSE 0
                    end) AS mar_net,
                Sum(CASE
                      WHEN d_month_name_April = 1 THEN cs_net_paid * cs_quantity
                      ELSE 0
                    end) AS apr_net,
                Sum(CASE
                      WHEN d_month_name_May = 1 THEN cs_net_paid * cs_quantity
                      ELSE 0
                    end) AS may_net,
                Sum(CASE
                      WHEN d_month_name_June = 1 THEN cs_net_paid * cs_quantity
                      ELSE 0
                    end) AS jun_net,
                Sum(CASE
                      WHEN d_month_name_July = 1 THEN cs_net_paid * cs_quantity
                      ELSE 0
                    end) AS jul_net,
                Sum(CASE
                      WHEN d_month_name_August = 1 THEN cs_net_paid * cs_quantity
                      ELSE 0
                    end) AS aug_net,
                Sum(CASE
                      WHEN d_month_name_September = 1 THEN cs_net_paid * cs_quantity
                      ELSE 0
                    end) AS sep_net,
                Sum(CASE
                      WHEN d_month_name_October = 1 THEN cs_net_paid * cs_quantity
                      ELSE 0
                    end) AS oct_net,
                Sum(CASE
                      WHEN d_month_name_November = 1 THEN cs_net_paid * cs_quantity
                      ELSE 0
                    end) AS nov_net,
                Sum(CASE
                      WHEN d_month_name_December = 1 THEN cs_net_paid * cs_quantity
                      ELSE 0
                    end) AS dec_net
         FROM   catalog_sales,
                warehouse,
                date_dim,
                time_dim,
                ship_mode
         WHERE  cs_warehouse_sk = w_warehouse_sk
                AND cs_sold_date_sk = d_date_sk
                AND cs_sold_time_sk = t_time_sk
                AND cs_ship_mode_sk = sm_ship_mode_sk
                AND d_year = 2000
         GROUP  BY w_warehouse_name,
                   w_warehouse_sq_ft,
                   w_city,
                   w_county,
                   w_state,
                   w_country,
                   d_year)) x
GROUP  BY w_warehouse_name,
          w_warehouse_sq_ft,
          w_city,
          w_county,
          w_state,
          w_country,
          year
ORDER  BY w_warehouse_name
```

Μετατροπές SQL

d_month_name: d_month_name = 'Value' <-> d_month_name_Value = 1

Διάγραμμα Ποσοστιαίας Χρονικής Απόκλισης



Σχόλια

Όπως φαίνεται και στο διάγραμμα, ο χρόνος εκτέλεσης του συγκεκριμένου query ήταν αισθητά μικρότερος στην booleanized βάση δεδομένων, στο σύνολο σχεδόν των επαναλήψεων. Το attribute που μετασχηματίστηκε έχει μεγάλο εύρος τιμών και εμπλέκεται σε ένα ιδιαίτερα σύνθετο εμφωλευμένο query.

QUERY27

Basic SQL

query27.sql Raw

```sql
 SELECT
       Count(*) AS cnt1,
       cd_purchase_estimate,
       Count(*) AS cnt2,
       cd_credit_rating,
       Count(*) AS cnt3
FROM   customer c,
       customer_address ca,
       customer_demographics
WHERE  c.c_current_addr_sk = ca.ca_address_sk
       AND cd_demo_sk = c.c_current_cdemo_sk
       AND EXISTS (SELECT *
                   FROM   store_sales,
                          date_dim
                   WHERE  c.c_customer_sk = ss_customer_sk
                          AND ss_sold_date_sk = d_date_sk
                          AND d_year > 2000
                          AND (d_month_name='January' OR d_month_name='February'
                               OR d_month_name='March' OR d_month_name='April'
                               OR d_month_name='May' OR d_month_name='June'))
       AND ( NOT EXISTS (SELECT *
                         FROM   web_sales,
                                date_dim
                         WHERE  c.c_customer_sk = ws_bill_customer_sk
                                AND ws_sold_date_sk = d_date_sk
                                AND d_year > 2000
                                AND (d_month_name='January'  OR d_month_name='February'
                                     OR d_month_name='March' OR d_month_name='April'
                                     OR d_month_name='May' OR d_month_name='June'))
             AND NOT EXISTS (SELECT *
                             FROM   catalog_sales,
                                    date_dim
                             WHERE  c.c_customer_sk = cs_ship_customer_sk
                                    AND cs_sold_date_sk = d_date_sk
                                    AND d_year > 2000
                                    AND (d_month_name='January' OR d_month_name='February'
                                        OR d_month_name='March' OR d_month_name='April'
                                        OR d_month_name='May' OR d_month_name='June')) )
GROUP  BY
          cd_purchase_estimate,
          cd_credit_rating
ORDER  BY
          cd_purchase_estimate,
          cd_credit_rating;
```

Booleanized SQL

bquery27.sql Raw

```
 SELECT Count(*) AS cnt1,
       cd_purchase_estimate,
       Count(*) AS cnt2,
       cd_credit_rating,
       Count(*) AS cnt3
FROM   customer c,
       customer_address ca,
       customer_demographics
WHERE  c.c_current_addr_sk = ca.ca_address_sk
       AND cd_demo_sk = c.c_current_cdemo_sk
       AND EXISTS (SELECT *
                   FROM   store_sales,
                          date_dim
                   WHERE  c.c_customer_sk = ss_customer_sk
                          AND ss_sold_date_sk = d_date_sk
                          AND d_year > 2000
                          AND ( d_month_name_january = 1
                                 OR d_month_name_february = 1
                                 OR d_month_name_march = 1
                                 OR d_month_name_april = 1
                                 OR d_month_name_may = 1
                                 OR d_month_name_june = 1 ))
       AND ( NOT EXISTS (SELECT *
                         FROM   web_sales,
                                date_dim
                         WHERE  c.c_customer_sk = ws_bill_customer_sk
                                AND ws_sold_date_sk = d_date_sk
                                AND d_year > 2000
                                AND ( d_month_name_january = 1
                                       OR d_month_name_february = 1
                                       OR d_month_name_march = 1
                                       OR d_month_name_april = 1
                                       OR d_month_name_may = 1
                                       OR d_month_name_june = 1 ))
             AND NOT EXISTS (SELECT *
                             FROM   catalog_sales,
                                    date_dim
                             WHERE  c.c_customer_sk = cs_ship_customer_sk
                                    AND cs_sold_date_sk = d_date_sk
                                    AND d_year > 2000
                                    AND ( d_month_name_january = 1
                                           OR d_month_name_february = 1
                                           OR d_month_name_march = 1
                                           OR d_month_name_april = 1
                                           OR d_month_name_may = 1
                                           OR d_month_name_june = 1 )) )
GROUP  BY cd_purchase_estimate,
          cd_credit_rating
ORDER  BY cd_purchase_estimate,
          cd_credit_rating;
```

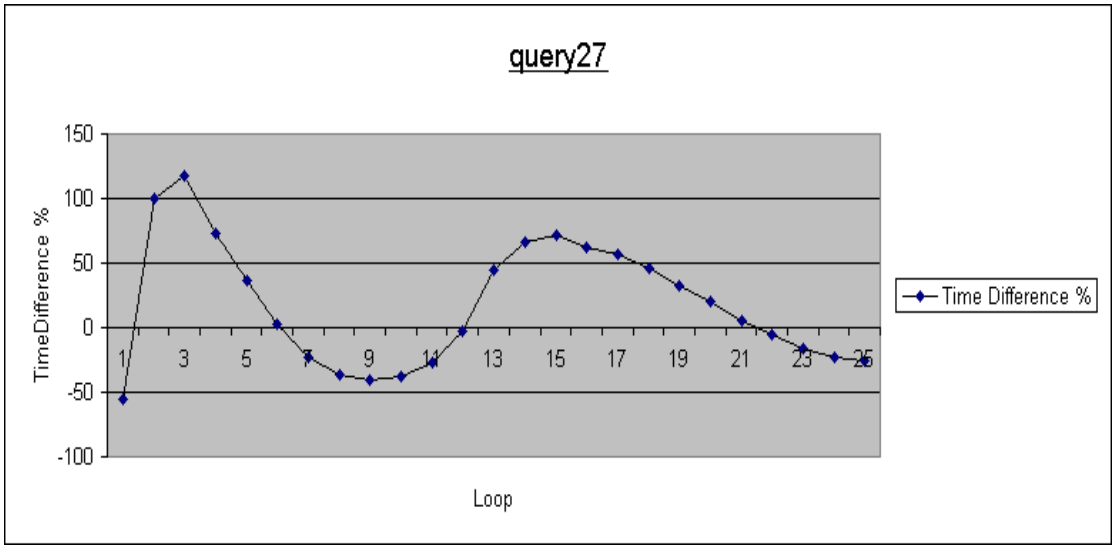
Μετατροπές SQL

d_month_name:

d_month_name IN (v1,v2,v3) <->

d_month_name_v1 = 1 OR d_month_name_v2 = 1 OR d_month_name_v3 = 1

Διάγραμμα Ποσοστιαίας Χρονικής Απόκλισης



Σχόλια

Ένα IN clause στο αρχικό query μετατράπηκε σε έναν συνδυασμό OR clauses επί του ίδιου (booleanized) attribute στο μετασχηματισμένο query. Φαίνεται πως ο μετασχηματισμός δεν επηρέασε με ευθύ τρόπο την επίδοση του συγκεκριμένου query, όπως φαίνεται και από τις συχνές εναλλαγές προσήμου στο διάγραμμα.

QUERY 28

Basic SQL

query28.sql Raw

```sql
SELECT cd_gender,
       cd_marital_status,
       cd_education_status,
       Count(*) AS cnt1,
       cd_purchase_estimate,
       Count(*) AS cnt2,
       cd_credit_rating,
       Count(*) AS cnt3
FROM   customer c,
       customer_address ca,
       customer_demographics
WHERE  c.c_current_addr_sk = ca.ca_address_sk
       AND cd_demo_sk = c.c_current_cdemo_sk
       AND EXISTS (SELECT *
                   FROM   store_sales,
                          date_dim
                   WHERE  c.c_customer_sk = ss_customer_sk
                          AND ss_sold_date_sk = d_date_sk
                          AND d_year > 2000
                          AND (d_month_name='January' OR d_month_name='February'
                          OR d_month_name='March' OR d_month_name='April'
                          OR d_month_name='May' OR d_month_name='June'))
       AND ( NOT EXISTS (SELECT *
                         FROM   web_sales,
                                date_dim
                         WHERE  c.c_customer_sk = ws_bill_customer_sk
                                AND ws_sold_date_sk = d_date_sk
                                AND d_year > 2000
                                AND (d_month_name='January' OR d_month_name='February'
                                OR d_month_name='March' OR d_month_name='April'
                                OR d_month_name='May' OR d_month_name='June'))
             AND NOT EXISTS (SELECT *
                             FROM   catalog_sales,
                                    date_dim
                             WHERE  c.c_customer_sk = cs_ship_customer_sk
                                    AND cs_sold_date_sk = d_date_sk
                                    AND d_year > 2000
                                    AND (d_month_name='January' OR d_month_name='February'
                                    OR d_month_name='March' OR d_month_name='April'
                                    OR d_month_name='May' OR d_month_name='June')) )
GROUP  BY cd_gender,
          cd_marital_status,
          cd_education_status,
          cd_purchase_estimate,
          cd_credit_rating
ORDER  BY cd_gender,
          cd_marital_status,
          cd_education_status,
          cd_purchase_estimate,
          cd_credit_rating;
```

Booleanized SQL

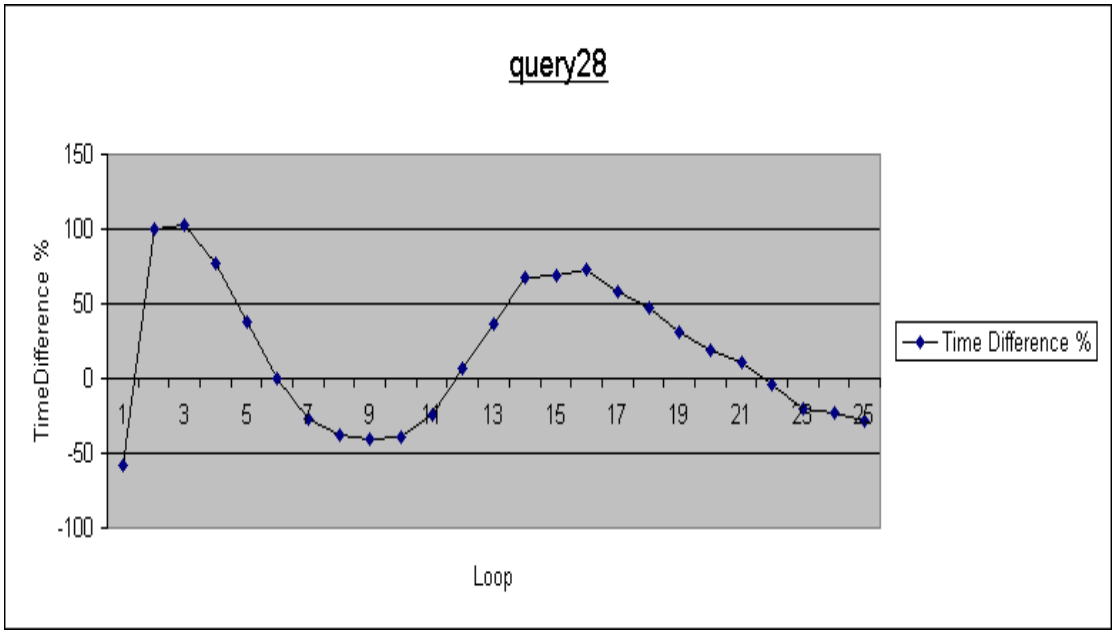
bquery28.sql Raw

```
SELECT cd_gender_male,
       cd_gender_female,
       cd_marital_status_single,
       cd_marital_status_married,
       cd_marital_status_divorced,
       cd_education_status_primary,
       cd_education_status_secondary,
       cd_education_status_university,
       cd_education_status_postgraduate,
       Count(*) AS cnt1,
       cd_purchase_estimate,
       Count(*) AS cnt2,
       cd_credit_rating,
       Count(*) AS cnt3
FROM   customer c,
       customer_address ca,
       customer_demographics
WHERE  c.c_current_addr_sk = ca.ca_address_sk
       AND cd_demo_sk = c.c_current_cdemo_sk
       AND EXISTS (SELECT *
                   FROM   store_sales,
                          date_dim
                   WHERE  c.c_customer_sk = ss_customer_sk
                          AND ss_sold_date_sk = d_date_sk
                          AND d_year > 2000
                          AND ( d_month_name_january = 1
                                 OR d_month_name_february = 1
                                 OR d_month_name_march = 1
                                 OR d_month_name_april = 1
                                 OR d_month_name_may = 1
                                 OR d_month_name_june = 1 ))
       AND ( NOT EXISTS (SELECT *
                         FROM   web_sales,
                                date_dim
                         WHERE  c.c_customer_sk = ws_bill_customer_sk
                                AND ws_sold_date_sk = d_date_sk
                                AND d_year > 2000
                                AND ( d_month_name_january = 1
                                       OR d_month_name_february = 1
                                       OR d_month_name_march = 1
                                       OR d_month_name_april = 1
                                       OR d_month_name_may = 1
                                       OR d_month_name_june = 1 ))
             AND NOT EXISTS (SELECT *
                             FROM   catalog_sales,
                                    date_dim
                             WHERE  c.c_customer_sk = cs_ship_customer_sk
                                    AND cs_sold_date_sk = d_date_sk
                                    AND d_year > 2000
                                    AND ( d_month_name_january = 1
                                           OR d_month_name_february = 1
                                           OR d_month_name_march = 1
                                           OR d_month_name_april = 1
                                           OR d_month_name_may = 1
                                           OR d_month_name_june = 1 )) )
GROUP  BY cd_gender_male,
          cd_gender_female,
          cd_marital_status_single,
          cd_marital_status_married,
          cd_marital_status_divorced,
          cd_education_status_primary,
          cd_education_status_secondary,
          cd_education_status_university,
          cd_education_status_postgraduate,
          cd_purchase_estimate,
          cd_credit_rating
ORDER  BY cd_gender_male,
          cd_gender_female,
          cd_marital_status_single,
          cd_marital_status_married,
          cd_marital_status_divorced,
          cd_education_status_primary,
          cd_education_status_secondary,
          cd_education_status_university,
          cd_education_status_postgraduate,
          cd_purchase_estimate,
          cd_credit_rating;
```

Μετατροπές SQL

d_month_name: d_month_name = 'Value' <-> d_month_name_Value = 1
cd_gender: cd_gender <-> cd_gender_male, cd_gender_female
cd_marital_status:
cd_marital_status <->
cd_marital_single, cd_marital_married, cd_marital_divorced
cd_education_status:
cd_education_status <->
cd_education_status_primary, cd_education_status_secondary,
cd_education_status_university, cd_education_status_postgraduate

Διάγραμμα Ποσοστιαίας Χρονικής Απόκλισης



Σχόλια

Στο query αυτό εφαρμόστηκαν πολλές μετατροπές, ωστόσο ο χρόνος εκτέλεσης δεν επηρεάστηκε με σαφή τρόπο. Οι συχνές εναλλαγές προσήμου στη γραφική παράσταση είναι ενδεικτικές . Καμία από τις δύο υλοποιήσεις της βάσης δεν υπερτερεί ορατά έναντι της άλλης.

QUERY 29

Basic SQL

query29.sql Raw

```sql
SELECT i_brand_id     AS brand_id,
       i_brand        AS brand,
       t_hour,
       t_minute,
       Sum(ext_price) AS ext_price
FROM   item,
       (SELECT ws_ext_sales_price AS ext_price,
               ws_sold_date_sk    AS sold_date_sk,
               ws_item_sk         AS sold_item_sk,
               ws_sold_time_sk    AS time_sk
        FROM   web_sales,
               date_dim
        WHERE  d_date_sk = ws_sold_date_sk
               AND d_month_name = 'February'
               AND d_year > 2000
        UNION ALL
        SELECT cs_ext_sales_price AS ext_price,
               cs_sold_date_sk    AS sold_date_sk,
               cs_item_sk         AS sold_item_sk,
               cs_sold_time_sk    AS time_sk
        FROM   catalog_sales,
               date_dim
        WHERE  d_date_sk = cs_sold_date_sk
               AND d_month_name = 'February'
               AND d_year > 2000
        UNION ALL
        SELECT ss_ext_sales_price AS ext_price,
               ss_sold_date_sk    AS sold_date_sk,
               ss_item_sk         AS sold_item_sk,
               ss_sold_time_sk    AS time_sk
        FROM   store_sales,
               date_dim
        WHERE  d_date_sk = ss_sold_date_sk
               AND d_month_name = 'February'
               AND d_year > 2000) AS tmp,
       time_dim
WHERE  sold_item_sk = i_item_sk
       AND time_sk = t_time_sk
GROUP  BY i_brand,
          i_brand_id,
          t_hour,
          t_minute
ORDER  BY ext_price DESC,
          i_brand_id;
```

Booleanized SQL

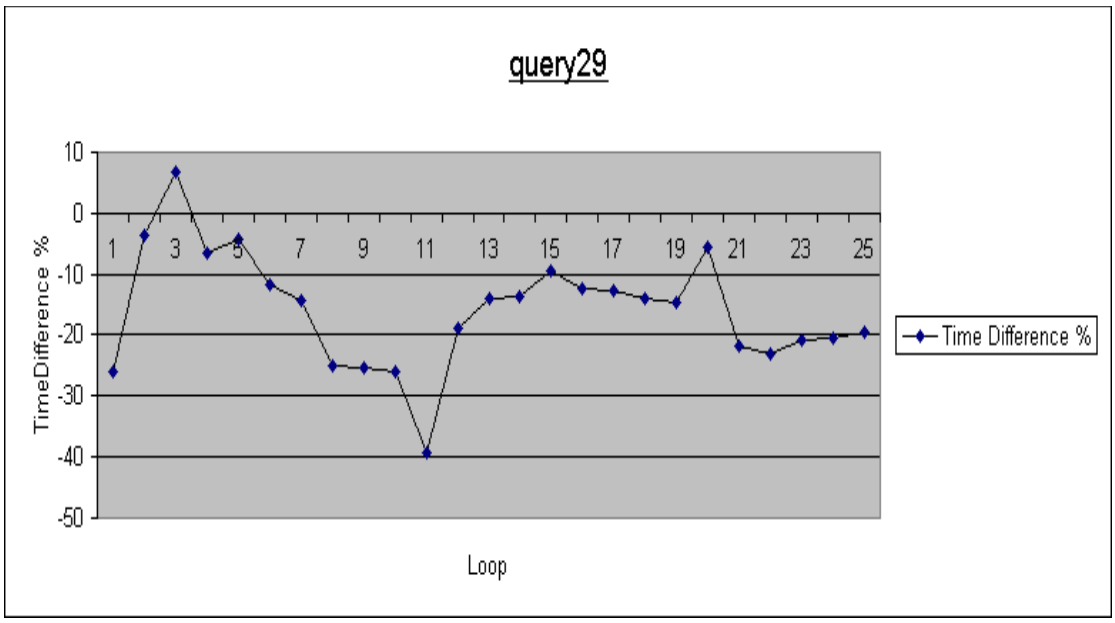
bquery29.sql Raw

```
 SELECT i_brand_id     AS brand_id,
       i_brand        AS brand,
       t_hour,
       t_minute,
       Sum(ext_price) AS ext_price
FROM   item,
       (SELECT ws_ext_sales_price AS ext_price,
               ws_sold_date_sk    AS sold_date_sk,
               ws_item_sk         AS sold_item_sk,
               ws_sold_time_sk    AS time_sk
        FROM   web_sales,
               date_dim
        WHERE  d_date_sk = ws_sold_date_sk
               AND d_month_name_february = 1
               AND d_year > 2000
        UNION ALL
        SELECT cs_ext_sales_price AS ext_price,
               cs_sold_date_sk    AS sold_date_sk,
               cs_item_sk         AS sold_item_sk,
               cs_sold_time_sk    AS time_sk
        FROM   catalog_sales,
               date_dim
        WHERE  d_date_sk = cs_sold_date_sk
               AND d_month_name_february = 1
               AND d_year > 2000
        UNION ALL
        SELECT ss_ext_sales_price AS ext_price,
               ss_sold_date_sk    AS sold_date_sk,
               ss_item_sk         AS sold_item_sk,
               ss_sold_time_sk    AS time_sk
        FROM   store_sales,
               date_dim
        WHERE  d_date_sk = ss_sold_date_sk
               AND d_month_name_february = 1
               AND d_year > 2000) AS tmp,
       time_dim
WHERE  sold_item_sk = i_item_sk
       AND time_sk = t_time_sk
GROUP  BY i_brand,
          i_brand_id,
          t_hour,
          t_minute
ORDER  BY ext_price DESC,
          i_brand_id
```

Μετατροπές SQL

d_month_name: d_month_name = 'Value' <-> d_month_name_Value = 1

Διάγραμμα Ποσοστιαίας Χρονικής Απόκλισης



Σχόλια

Στο query αυτό ο boolean approach μετασχηματισμός είχε σαφή θετικά αποτελέσματα για τον χρόνο εκτέλεσης, αφού στο σύνολο σχεδόν των επαναλήψεων η booleanized βάση δεδομένων υπερτερεί της βασικής. Και σε αυτή την περίπτωση το attribute Που μετασχηματίστηκε εμπλέκεται σε WHERE CLASE εμφωλευμένου query.

QUERY 30

Basic SQL

query30.sql Raw

```sql
SELECT i_item_desc,
       w_warehouse_name,
       d1.d_week_seq,
       Count(CASE
               WHEN p_promo_sk IS NULL THEN 1
               ELSE 0
             END) AS no_promo,
       Count(CASE
               WHEN p_promo_sk IS NOT NULL THEN 1
               ELSE 0
             END) AS promo,
       Count(*)   AS total_cnt
FROM   catalog_sales
       JOIN inventory
         ON ( cs_item_sk = inv_item_sk )
       JOIN warehouse
         ON ( w_warehouse_sk = inv_warehouse_sk )
       JOIN item
         ON ( i_item_sk = cs_item_sk )
       JOIN customer_demographics
         ON ( cs_bill_cdemo_sk = cd_demo_sk )
       JOIN household_demographics
         ON ( cs_bill_hdemo_sk = hd_demo_sk )
       JOIN date_dim d1
         ON ( cs_sold_date_sk = d1.d_date_sk )
       JOIN date_dim d2
         ON ( inv_date_sk = d2.d_date_sk )
       JOIN date_dim d3
         ON ( cs_ship_date_sk = d3.d_date_sk )
       LEFT OUTER JOIN promotion
                    ON ( cs_promo_sk = p_promo_sk )
       LEFT OUTER JOIN catalog_returns
                    ON ( cr_item_sk = cs_item_sk
                         AND cr_order_number = cs_order_number )
WHERE  d3.d_date > d1.d_date
       AND d1.d_year > 2000
       AND cd_marital_status = 'single'
GROUP  BY i_item_desc,
          w_warehouse_name,
          d1.d_week_seq
ORDER  BY total_cnt DESC,
          i_item_desc,
          w_warehouse_name,
          d_week_seq;
```

Booleanized SQL

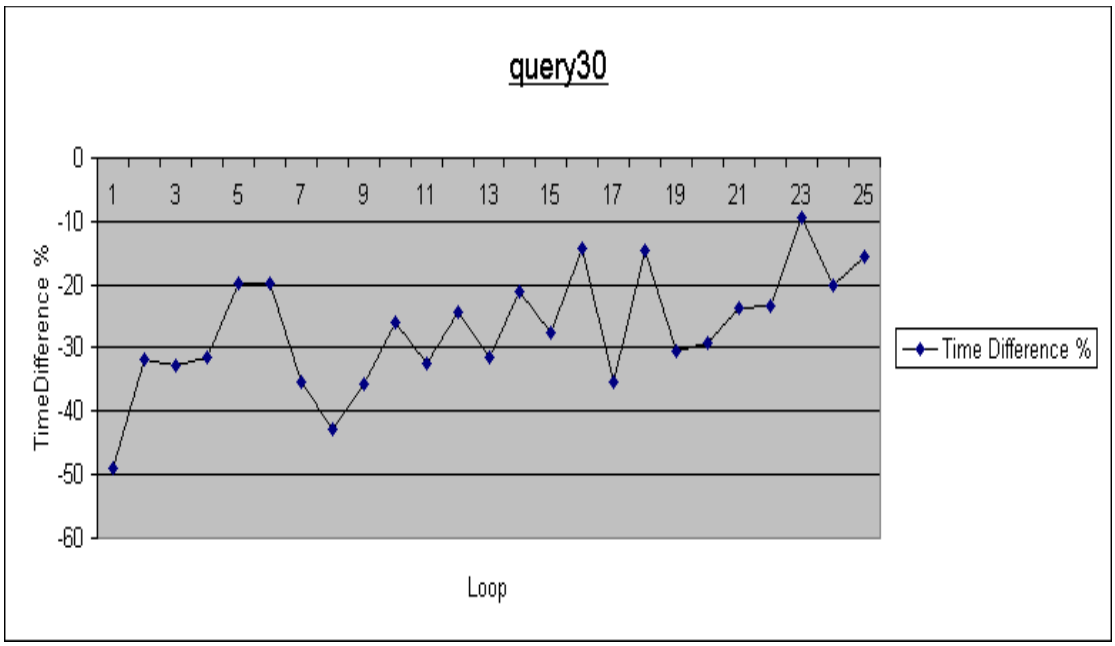
bquery30.sql Raw

```
SELECT i_item_desc,
       w_warehouse_name,
       d1.d_week_seq,
       Count(CASE
               WHEN p_promo_sk IS NULL THEN 1
               ELSE 0
             END) AS no_promo,
       Count(CASE
               WHEN p_promo_sk IS NOT NULL THEN 1
               ELSE 0
             END) AS promo,
       Count(*)   AS total_cnt
FROM   catalog_sales
       JOIN inventory
         ON ( cs_item_sk = inv_item_sk )
       JOIN warehouse
         ON ( w_warehouse_sk = inv_warehouse_sk )
       JOIN item
         ON ( i_item_sk = cs_item_sk )
       JOIN customer_demographics
         ON ( cs_bill_cdemo_sk = cd_demo_sk )
       JOIN household_demographics
         ON ( cs_bill_hdemo_sk = hd_demo_sk )
       JOIN date_dim d1
         ON ( cs_sold_date_sk = d1.d_date_sk )
       JOIN date_dim d2
         ON ( inv_date_sk = d2.d_date_sk )
       JOIN date_dim d3
         ON ( cs_ship_date_sk = d3.d_date_sk )
       LEFT OUTER JOIN promotion
                    ON ( cs_promo_sk = p_promo_sk )
       LEFT OUTER JOIN catalog_returns
                    ON ( cr_item_sk = cs_item_sk
                         AND cr_order_number = cs_order_number )
WHERE  d3.d_date > d1.d_date
       AND d1.d_year > 2000
       AND cd_marital_status_single=1
GROUP  BY i_item_desc,
          w_warehouse_name,
          d1.d_week_seq
ORDER  BY total_cnt DESC,
          i_item_desc,
          w_warehouse_name,
          d_week_seq;
```

Μετατροπές SQL

cd_marital_status: cd_marital_status = 'Value' <-> cd_marital_status_Value = 1

Διάγραμμα Ποσοστιαίας Χρονικής Απόκλισης



Σχόλια

Ακόμα μια περίπτωση στην οποία ο μετασχηματισμός ενός attribute που εμπλέκεται σε WHERE clause επιδρά θετικά στον χρόνο εκτέλεσης του query. Σε όλες τις επαναλήψεις η booleanized βάση δεδομένων υπερτερεί της αρχικής.

QUERY 31

Basic SQL

query31.sql Raw

```sql
SELECT c_last_name,
       c_first_name,
       Substr(s_city, 1, 30),
       ss_ticket_number,
       amt,
       profit
FROM   (SELECT ss_ticket_number,
               ss_customer_sk,
               store.s_city,
               Sum(ss_coupon_amt) AS amt,
               Sum(ss_net_profit) AS profit
        FROM   store_sales,
               date_dim,
               store,
               household_demographics
        WHERE  store_sales.ss_sold_date_sk = date_dim.d_date_sk
               AND store_sales.ss_store_sk = store.s_store_sk
               AND store_sales.ss_hdemo_sk = household_demographics.hd_demo_sk
               AND ( household_demographics.hd_dep_count < 3
                      OR household_demographics.hd_vehicle_count > 2 )
               AND date_dim.d_day_name = 'Monday'
               AND date_dim.d_year IN ( 1999, 2000, 2001 )
               AND store.s_number_employees BETWEEN 200 AND 1000
        GROUP  BY ss_ticket_number,
                  ss_customer_sk,
                  ss_addr_sk,
                  store.s_city) ms,
       customer
WHERE  ss_customer_sk = c_customer_sk
ORDER  BY c_last_name,
          c_first_name,
          Substr(s_city, 1, 30),
          profit;
```

Booleanized SQL

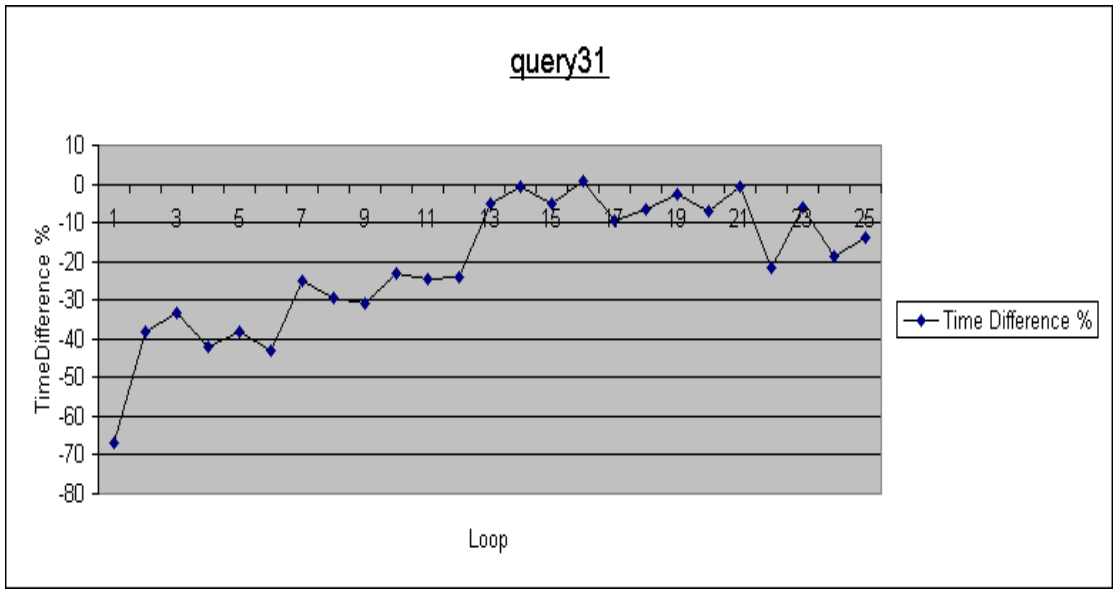
bquery31.sql Raw

```sql
SELECT c_last_name,
       c_first_name,
       Substr(s_city, 1, 30),
       ss_ticket_number,
       amt,
       profit
FROM   (SELECT ss_ticket_number,
               ss_customer_sk,
               store.s_city,
               Sum(ss_coupon_amt) AS amt,
               Sum(ss_net_profit) AS profit
        FROM   store_sales,
               date_dim,
               store,
               household_demographics
        WHERE  store_sales.ss_sold_date_sk = date_dim.d_date_sk
               AND store_sales.ss_store_sk = store.s_store_sk
               AND store_sales.ss_hdemo_sk = household_demographics.hd_demo_sk
               AND ( household_demographics.hd_dep_count < 3
                      OR household_demographics.hd_vehicle_count > 2 )
               AND date_dim.d_day_name_Monday = 1
               AND date_dim.d_year IN ( 1999, 2000, 2001 )
               AND store.s_number_employees BETWEEN 200 AND 1000
        GROUP  BY ss_ticket_number,
                  ss_customer_sk,
                  ss_addr_sk,
                  store.s_city) ms,
       customer
WHERE  ss_customer_sk = c_customer_sk
ORDER  BY c_last_name,
          c_first_name,
          Substr(s_city, 1, 30),
          profit;
```

Μετατροπές SQL

d_month_name: d_month_name = 'Value' <-> d_month_name_Value = 1

Διάγραμμα Ποσοστιαίας Χρονικής Απόκλισης



Σχόλια

Ο μετασχηματισμός ενός attribute που εμπλέκεται σε εμφωλευμένο WHERE clause, αυτή τη φορά του d_day_name που έχει επτά πιθανές τιμές, καταγράφει θετικά αποτελέσματα ως προς τον χρόνο εκτέλεσης του query, καθώς και πάλι, η booleanized βάση δεδομένων υπερτερεί της αρχικής, σε όλες τις επαναλήψεις εκτέλεσης του query.

QUERY 32

Basic SQL

query32.sql Raw

```sql
 SELECT Substr(r_reason_desc, 1, 20),
       Avg(ws_quantity),
       Avg(ws_quantity),
       Avg(ws_quantity),
       Avg(wr_refunded_cash),
       Avg(wr_fee)
FROM   web_sales,
       web_returns,
       web_page,
       customer_demographics cd1,
       customer_demographics cd2,
       customer_address,
       date_dim,
       reason
WHERE  ws_web_page_sk = wp_web_page_sk
       AND ws_item_sk = wr_item_sk
       AND ws_order_number = wr_order_number
       AND ws_sold_date_sk = d_date_sk
       AND d_year > 2000
       AND cd1.cd_demo_sk = wr_refunded_cdemo_sk
       AND cd2.cd_demo_sk = wr_returning_cdemo_sk
       AND ca_address_sk = wr_refunded_addr_sk
       AND r_reason_sk = wr_reason_sk
       AND ( ( cd1.cd_marital_status = 'single'
               AND cd1.cd_marital_status = cd2.cd_marital_status
               AND cd1.cd_education_status = 'secondary'
               AND cd1.cd_education_status = cd2.cd_education_status )
              OR ( cd1.cd_marital_status = 'married'
                   AND cd1.cd_marital_status = cd2.cd_marital_status
                   AND cd1.cd_education_status = 'university'
                   AND cd1.cd_education_status = cd2.cd_education_status)
              OR ( cd1.cd_marital_status = 'divorced'
                   AND cd1.cd_marital_status = cd2.cd_marital_status
                   AND cd1.cd_education_status = 'postgraduate'
                   AND cd1.cd_education_status = cd2.cd_education_status ) )
GROUP  BY r_reason_desc
ORDER  BY Substr(r_reason_desc, 1, 20),
          Avg(ws_quantity),
          Avg(wr_refunded_cash),
          Avg(wr_fee);
```

Booleanized SQL

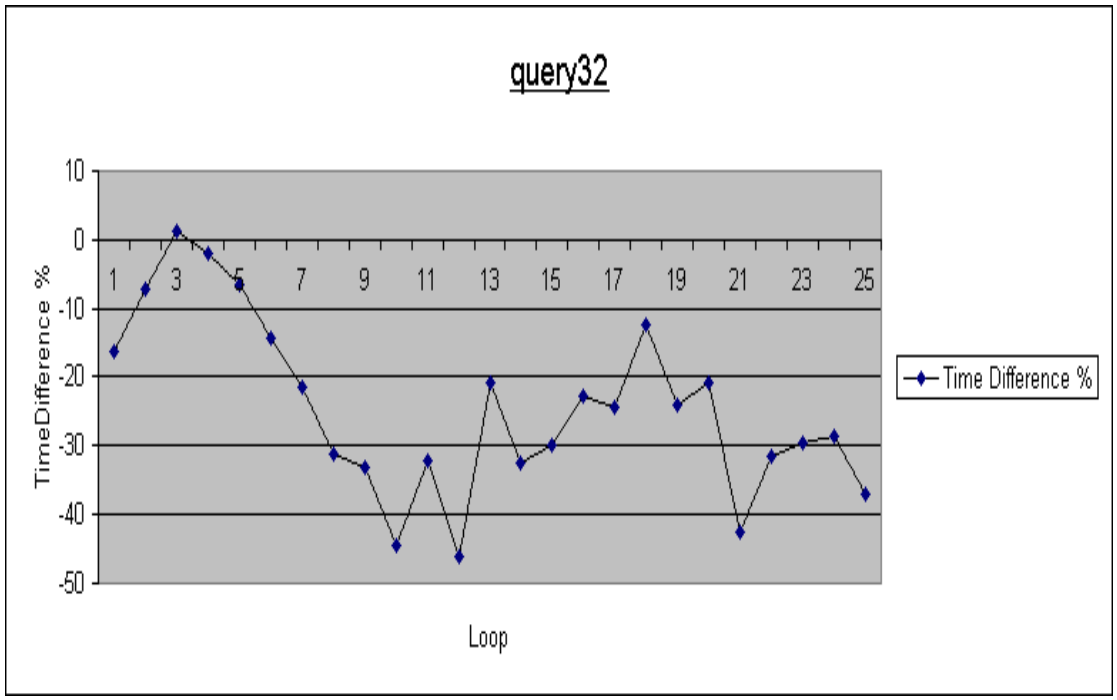
bquery32.sql Raw

```
 SELECT Substr(r_reason_desc, 1, 20),
       Avg(ws_quantity),
       Avg(ws_quantity),
       Avg(ws_quantity),
       Avg(wr_refunded_cash),
       Avg(wr_fee)
FROM   web_sales,
       web_returns,
       web_page,
       customer_demographics cd1,
       customer_demographics cd2,
       customer_address,
       date_dim,
       reason
WHERE  ws_web_page_sk = wp_web_page_sk
       AND ws_item_sk = wr_item_sk
       AND ws_order_number = wr_order_number
       AND ws_sold_date_sk = d_date_sk
       AND d_year > 2000
       AND cd1.cd_demo_sk = wr_refunded_cdemo_sk
       AND cd2.cd_demo_sk = wr_returning_cdemo_sk
       AND ca_address_sk = wr_refunded_addr_sk
       AND r_reason_sk = wr_reason_sk
       AND ( ( cd1.cd_marital_status_single=1
               AND cd2.cd_marital_status_single=1
               AND cd1.cd_education_status_secondary=1
               AND cd2.cd_education_status_secondary=1 )
              OR ( cd1.cd_marital_status_married=1
                   AND cd2.cd_marital_status_married=1
                   AND cd1.cd_education_status_university=1
                   AND cd2.cd_education_status_university=1)
              OR ( cd1.cd_marital_status_divorced=1
                   AND cd2.cd_marital_status_divorced=1
                   AND cd1.cd_education_status_postgraduate=1
                   AND cd2.cd_education_status_postgraduate=1 ) )
GROUP  BY r_reason_desc
ORDER  BY Substr(r_reason_desc, 1, 20),
          Avg(ws_quantity),
          Avg(wr_refunded_cash),
          Avg(wr_fee);
```

Μετατροπές SQL

cd_marital_status: cd_marital_status = 'Value' <-> cd_marital_status_Value = 1
cd_education_status: cd_education_status = 'Value' <-> cd_education_status_Value = 1

Διάγραμμα Ποσοστιαίας Χρονικής Απόκλισης



Σχόλια

Σε αυτό το query μετασχηματίστηκαν δυο attributes τα οποία εμπλέκονται σε ένα σύνθετο WHERE clause. Τα αποτελέσματα είναι εμφανώς θετικά για την Booleanized βάση δεδομένων που υπερτερεί χρονικά σχεδόν στο σύνολο των επαναλήψεων.

QUERY 33

Basic SQL

query33.sql Raw

```sql
 SELECT Cast(amc AS DECIMAL(15, 4)) / Cast(pmc AS DECIMAL(15, 4)) AS am_pm_ratio
FROM   (SELECT Count(*) AS amc
        FROM   web_sales,
               household_demographics,
               time_dim,
               web_page
        WHERE  ws_sold_time_sk = time_dim.t_time_sk
               AND ws_ship_hdemo_sk = household_demographics.hd_demo_sk
               AND ws_web_page_sk = web_page.wp_web_page_sk
               AND time_dim.t_am_pm = 'AM'
               AND household_demographics.hd_dep_count < 5
               AND web_page.wp_char_count BETWEEN 250000 AND 500000) at,
       (SELECT Count(*) AS pmc
        FROM   web_sales,
               household_demographics,
               time_dim,
               web_page
        WHERE  ws_sold_time_sk = time_dim.t_time_sk
               AND ws_ship_hdemo_sk = household_demographics.hd_demo_sk
               AND ws_web_page_sk = web_page.wp_web_page_sk
               AND time_dim.t_am_pm = 'PM'
               AND household_demographics.hd_dep_count < 5
               AND web_page.wp_char_count BETWEEN 250000 AND 500000) pt
ORDER  BY am_pm_ratio;
```

Booleanized SQL

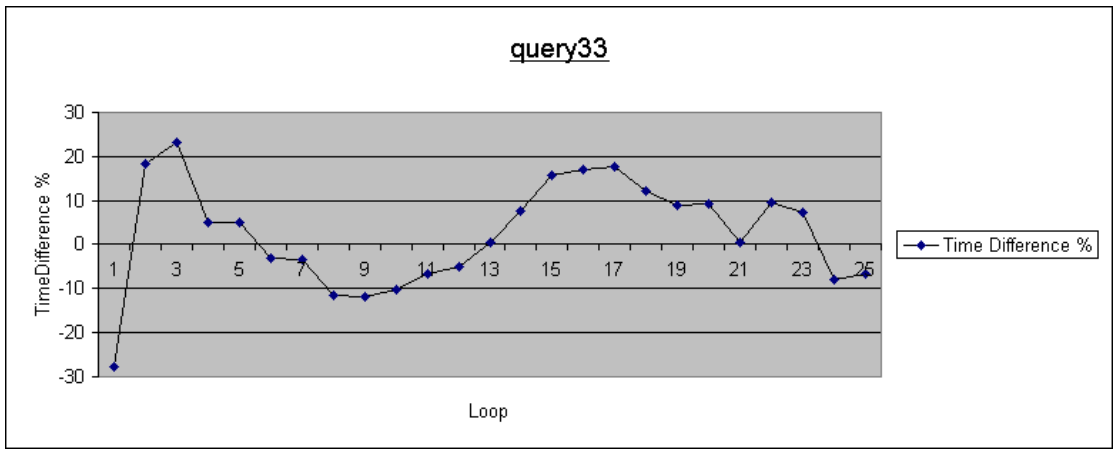
bquery33.sql Raw

```sql
 SELECT Cast(amc AS DECIMAL(15, 4)) / Cast(pmc AS DECIMAL(15, 4)) AS am_pm_ratio
FROM   (SELECT Count(*) AS amc
        FROM   web_sales,
               household_demographics,
               time_dim,
               web_page
        WHERE  ws_sold_time_sk = time_dim.t_time_sk
               AND ws_ship_hdemo_sk = household_demographics.hd_demo_sk
               AND ws_web_page_sk = web_page.wp_web_page_sk
               AND time_dim.t_am_pm_AM=1
               AND household_demographics.hd_dep_count < 5
               AND web_page.wp_char_count BETWEEN 250000 AND 500000) at,
       (SELECT Count(*) AS pmc
        FROM   web_sales,
               household_demographics,
               time_dim,
               web_page
        WHERE  ws_sold_time_sk = time_dim.t_time_sk
               AND ws_ship_hdemo_sk = household_demographics.hd_demo_sk
               AND ws_web_page_sk = web_page.wp_web_page_sk
               AND time_dim.t_am_pm_PM=1
               AND household_demographics.hd_dep_count < 5
               AND web_page.wp_char_count BETWEEN 250000 AND 500000) pt
ORDER  BY am_pm_ratio;
```

Μετατροπές SQL

t_am_pm: t_am_pm = 'value' <-> t_am_pm_value = 1

Διάγραμμα Ποσοστιαίας Χρονικής Απόκλισης



Σχόλια

Το συγκεκριμένο query είναι σχετικά απλό και η αλλαγές στη δομή του από τον μετασχηματισμό ελάχιστες. Τα αποτελέσματα, με τις συνεχείς εναλλαγές προσήμου, δεν δείχνουν σαφή επιρροή του χρόνου εκτέλεσης του query από τον μετασχηματισμό boolean approach.

QUERY 34

Basic SQL

query34.sql Raw

```sql
 SELECT cc_call_center_id AS Call_Center,
       cc_name           AS Call_Center_Name,
       cc_manager        AS Manager,
       Sum(cr_net_loss)  AS Returns_Loss
FROM   call_center,
       catalog_returns,
       date_dim,
       customer,
       customer_address,
       customer_demographics,
       household_demographics
WHERE  cr_call_center_sk = cc_call_center_sk
       AND cr_returned_date_sk = d_date_sk
       AND cr_returning_customer_sk = c_customer_sk
       AND cd_demo_sk = c_current_cdemo_sk
       AND hd_demo_sk = c_current_hdemo_sk
       AND ca_address_sk = c_current_addr_sk
       AND d_year > 2000
       AND d_month_name IN ('January','February','March','April','May','June')
       AND ( ( cd_marital_status = 'single'
               AND cd_education_status = 'secondary' )
              OR ( cd_marital_status = 'married'
                   AND cd_education_status = 'postgraduate' ) )
       AND ca_gmt_offset > 500
GROUP  BY cc_call_center_id,
          cc_name,
          cc_manager,
          cd_marital_status,
          cd_education_status
ORDER  BY Sum(cr_net_loss) DESC;
```

Booleanized SQL

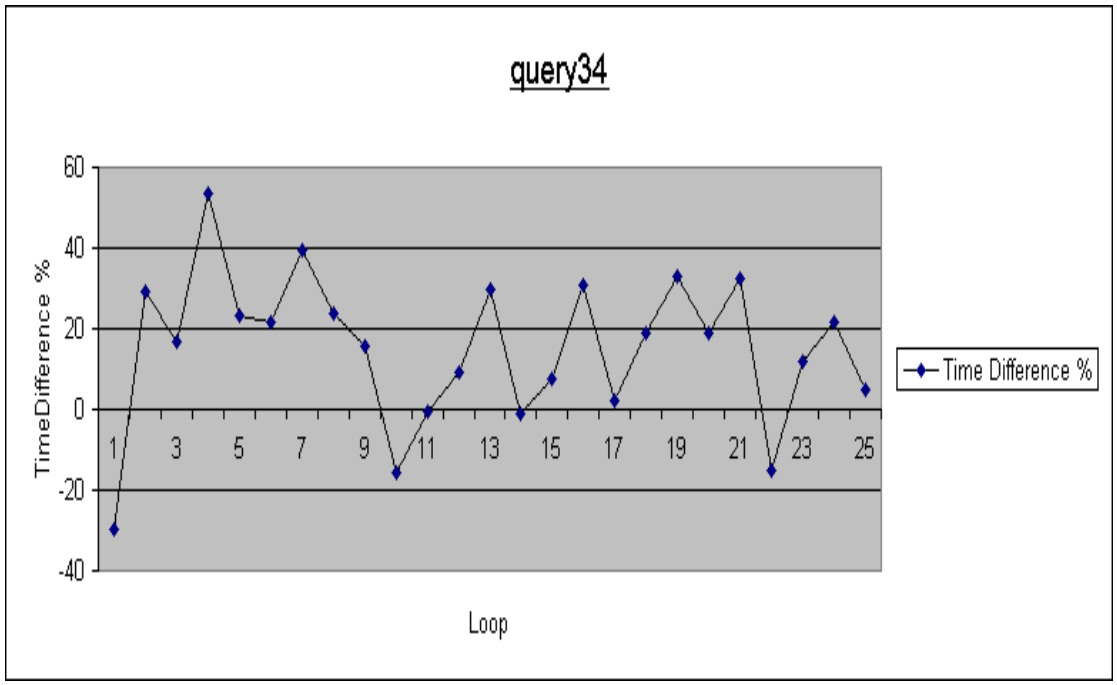
bquery34.sql Raw

```sql
 SELECT cc_call_center_id AS Call_Center,
       cc_name           AS Call_Center_Name,
       cc_manager        AS Manager,
       Sum(cr_net_loss)  AS Returns_Loss
FROM   call_center,
       catalog_returns,
       date_dim,
       customer,
       customer_address,
       customer_demographics,
       household_demographics
WHERE  cr_call_center_sk = cc_call_center_sk
       AND cr_returned_date_sk = d_date_sk
       AND cr_returning_customer_sk = c_customer_sk
       AND cd_demo_sk = c_current_cdemo_sk
       AND hd_demo_sk = c_current_hdemo_sk
       AND ca_address_sk = c_current_addr_sk
       AND d_year > 2000
       AND (d_month_name_January=1 OR d_month_name_February=1 OR
                        d_month_name_March=1 OR d_month_name_April=1 OR
            d_month_name_May=1 OR d_month_name_June=1)
       AND ( ( cd_marital_status_single=1
               AND cd_education_status_secondary=1 )
              OR ( cd_marital_status_married=1
                   AND cd_education_status_postgraduate=1 ) )
       AND ca_gmt_offset > 500
GROUP  BY cc_call_center_id,
          cc_name,
          cc_manager,
          cd_marital_status_single,cd_marital_status_married,cd_marital_status_divorced,
          cd_education_status_primary,cd_education_status_secondary,
          cd_education_status_university,cd_education_status_postgraduate
ORDER  BY Sum(cr_net_loss) DESC;
```

Μετατροπές SQL

d_month_name: d_month_name = 'Value' <-> d_month_name_Value = 1
cd_marital_status: cd_marital_status = 'Value' <-> cd_marital_status_Value = 1
cd_education_status: cd_education_status = 'Value' <-> cd_education_status_Value = 1

Διάγραμμα Ποσοστιαίας Χρονικής Απόκλισης



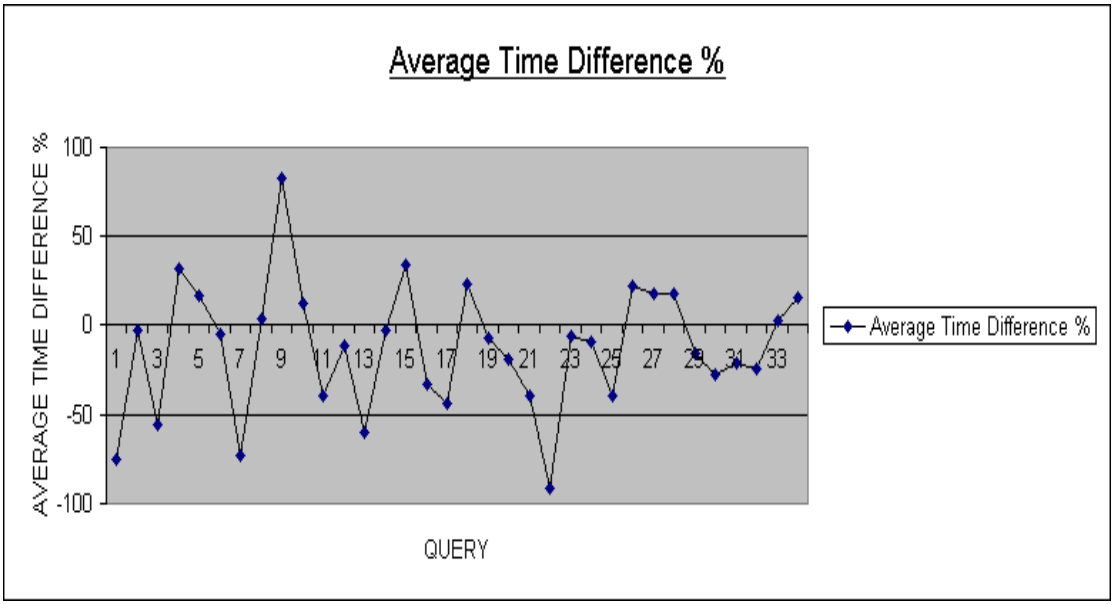
Σχόλια

Το query αυτό, που υπέστη μετασχηματισμό σε τρία attributes, είναι ένα από τα ελάχιστα παραδείγματα στα οποία η αρχική βάση φαίνεται να υπερτερεί χρονικά έναντι της booleanized βάσης, αφού στην πλειοψηφία των επαναλήψεων, καταγράφει μικρότερο χρόνο εκτέλεσης.

ΕΠΙΛΟΓΟΣ

Τα επιμέρους αποτελέσματα των queries καταδεικνύουν ότι η επίδραση του μετασχηματισμού Boolean Approach στην επίδοση της εκτέλεσης των queries, και ειδικότερα στον χρόνο εκτέλεσης, είναι αρκετά απρόβλεπτη.

Στο ακόλουθο διάγραμμα απεικονίζεται η μέση χρονική διαφορά της εκτέλεσης κάθε query στις δυο βάσεις, για όλες τις επαναλήψεις.



Σε πολλές περιπτώσεις η booleanized βάση δεδομένων υπερτερεί της βασικής ως προς το χρόνο εκτέλεσης των ερωτημάτων. Είναι ενδεικτικό ότι καταγράφηκαν queries με χρονικές διαφορές υπέρ της booleanized βάσης δεδομένων, οι οποίες ξεπερνούν κατά μέσο όρο το 50%. Τα αντίστοιχα επί μέρους διαγράμματα των queries, που καταγράφονται στο προηγούμενο κεφάλαιο, καταγράφουν σταθερά μικρότερους χρόνους για την booleanized βάση δεδομένων. Για queries όπως αυτά ο μετασχηματισμός Boolean Approach φαίνεται πως επιδρά θετικά στον χρόνο εκτέλεσης τους.

Παρόλα αυτά, σε εξίσου πολλές περιπτώσεις οι χρονικές διαφορές που παρατηρούνται είναι πολύ μικρές κατά μέσο όρο. Στα queries αυτά καταγράφηκαν χρονικές διαφορές με πολλές εναλλαγές προσήμου, που σημαίνει σε άλλες επαναλήψεις η booleanized βάση υπερτερούσε και σε άλλες υστερούσε. Για queries σαν κι αυτά φαίνεται πως ο μετασχηματισμός Boolean Approach δεν έχει κάποια ιδιαίτερη επίδραση στον χρόνο εκτέλεσης τους.

Τέλος, παρουσιάστηκαν και μεμονωμένες περιπτώσεις queries () στα οποία η booleanized βάση δεδομένων υστερούσε χρονικά, κάτι που καταδεικνύει ακόμα περισσότερο ότι ο μετασχηματισμός Boolean Approach έχει απρόβλεπτες επιπτώσεις, όσον αφορά τον χρόνο εκτέλεσης ενός query.

Σε μια προσπάθεια να καταγραφούν οι παράγοντες που ενισχύουν την επίδραση του μετασχηματισμού Boolean Approach στον χρόνο εκτέλεσης ενός σχεσιακού query, θα εξετάσουμε εκτενέστερα την δομή των queries που κατέγραψαν τις μεγαλύτερες (κατά μέσο όρο) χρονικές διαφορές υπέρ της booleanized βάσης δεδομένων.

- **Query 1**: Μετασχηματίστηκε το attribute d_day_name, το οποίο έχει εύρος επτά διαφορετικών varchar τιμών και εμπλέκεται στον υπολογισμό aggregate function (sum()) εντός εμφωλευμένων queries.
- **Query 3**: Μετασχηματίστηκαν τα attributes cd_gender, cd_marital_status και cd_education_status, του πίνακα customer_demographics, με εύρος δυο, τριών και τεσσάρων varchar τιμών αντίστοιχα, καθώς και τα p_channel_email και p_channel_event του πίνακα promotion με εύρος δυο varchar τιμών το καθένα. Τα attributes συμμετέχουν στον σχηματισμό ενός σύνθετο WHERE clause με συζεύξεις και διαζεύξεις clauses που τα εμπλέκουν.
- **Query 7**: Μετασχηματίστηκαν τα attributes cd_gender, cd_marital_status και cd_education_status, του πίνακα customer_demographics, τα οποία συμμετέχουν σε clauses με συζεύξεις και διαζεύξεις διαφόρων τιμών τους.
- **Query 13**: Μετασχηματίστηκαν τα attributes c_preferred_customer_flag και d_day_name. Το πρώτο έχει εύρος δυο varchar τιμών και συμμετέχει στο ORDER BY clause του query, ενώ το δεύτερο οποίο έχει εύρος επτά διαφορετικών varchar τιμών και εμπλέκεται στον σχηματισμό ενός σύνθετου WHERE clause με διαζεύξεις των τιμών του.
- **Query 22:** Μετασχηματίστηκε το attribute d_month_name, το οποίο έχει εύρος δώδεκα διαφορετικών varchar τιμών και εμπλέκεται σε ένα απλό WHERE clause συζεύξεων.

Παρατηρώντας τη δομή των παραπάνω (και όχι μόνο queries), διαφαίνονται ορισμένες παράμετροι που φαίνεται να αυξάνουν τον βαθμό στον οποίο επιδρά ο μετασχηματισμός Boolean Approach στον χρόνο εκτέλεσης των queries. Παρακάτω καταγράφονται οι σημαντικότερες εξ αυτών:

- Το πλήθος των εμπλεκόμενων booleanized attributes. Φαίνεται πως η συμμετοχή πολλών booleanized attributes αυξάνουν την επίδραση του μετασχηματισμού, ιδιαίτερα όταν εμπλέκονται σε σύνθετα WHERE clauses.
- Το εύρος τιμών των booleanized queries. Φαίνεται πως όσο μεγαλύτερο είναι το εύρος τιμών ενός booleanized attribute τόσο περισσότερο επηρεάζει τον χρόνο εκτέλεσης ο μετασχηματισμός του.
- Τα clauses στα οποία εμπλέκονται booleanized attributes, (WHERE,SELECT, GROUP BY, ORDER BY, WITH IN κτλ). Φαίνεται πως ο μετασχηματισμός έχει μεγάλη επηρρεάζει ιδιαίτερα τα queries που εμπλέκουν τα booleanized attributes σε σύνθετα WHERE CLAUSES, ιδιαίτερα όταν αυτά είναι εμφωλευμένα.

Καταλήγοντας, ο μετασχημάτισμός Boolean Approach ευνοεί τον χρόνο εκτέλεσης σχεσιακών queries σε πολλές περιπτώσεις. Ωστόσο, επειδή η δομή των queries έχει μεγάλη σημασία, πριν από οποιαδήποτε απόφαση για υιοθέτηση του, θα πρέπει να προηγηθούν εκτενείς δοκιμές και μετρήσεις, για queries με την δομή και τα χαρακτηριστικά της εκάστοτε εφαρμογής.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Κεφάλαιο 1

- Database System Concepts (6th ed) - Silberschatz, Korth, Sudarshan
- Database Systems-The Complete Book (2nd ed) – Garcia-Molina,Ullman, Widom
- Database Management Systems (3rd ed) – Ramakrishnan, Gehrke
- MySQL 5.6 Reference Manual
  (http://dev.mysql.com/doc/refman/5.6/en/)
- SQL Tutorial του εκπαιδευτικού ιστοτόπου TutorialsPoint
  (http://www.tutorialspoint.com/sql/index.htm)

Κεφάλαιο 2

- Cloud Computing - Software Technology and Architectures – Erl, Mahmood, Puttini
- *Architecting the Cloud: Design Decisions for Cloud Computing Service Models – Kavis*
- *Cloud Computing Bible - Sosinsky*
- *Cloud Computing Tutorial , του ιστοτόπου TutorialsPoint*
  (http://www.tutorialspoint.com/cloud_computing/index.htm)
  Όλες οι εικόνες του κεφαλαίου αποτελούν πνευματική ιδιοκτησία του παραπάνω ιστοτόπου.

Κεφάλαιο 3

- Ιστότοπος TPCDS benchmark
  http://www.tpc.org/tpcds/
  Όλες οι εικόνες του παρόντος κεφαλαίου αποτελούν πνευματική ιδιοκτησία του παραπάνω ιστοτόπου.
- Έγγραφο προδιαγραφών TPCDS benchmark
  http://www.tpc.org/tpc_documents_current_versions/pdf/tpcds_1.4.pdf

Κεφάλαιο 4

- Ιστότοπος TPCDS benchmark
  http://www.tpc.org/tpcds/
  Όλες οι εικόνες του παρόντος κεφαλαίου αποτελούν πνευματική ιδιοκτησία του παραπάνω ιστοτόπου.
- Έγγραφο προδιαγραφών TPCDS benchmark
  http://www.tpc.org/tpc_documents_current_versions/pdf/tpcds_1.4.pdf
- Secured Disclosure Of Sensitive Data In Data Mining Techniques –
  Gurusamy, Chakrapani (2012)
- Boolean and Cluster Analysis for Knowledge Discovery in Databases –
  Kono, Sakurai, Yamaguchi (2000)

TpcdsDB.java

```
package thesis;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Random;


public class TpcdsDB {

// JDBC driver name and database URL
final static String JDBC_DRIVER = "com.mysql.jdbc.Driver";
final static String DB_SERVER_URL = "jdbc:mysql://localhost/";

//      Database name
final static String DBNAME1  = "tpcds";
final static String DBNAME2  = "bool_tpcds";

//  Database credentials
final static String USER = "root";
final static String PASS = "";

//Cardinalities of tables.....................................................

/* Note:
*
* card(ss) > card(sr)
* card(cs) > card(cr)
* card(ws) > card(wr)
*
**/

public final int cc_cardinality = 10;
public final int cp_cardinality = 10;
public final int cr_cardinality = 10000;
public final int cs_cardinality = 150000;
public final int c_cardinality = 40000;
public final int ca_cardinality = 1000;
public final int cd_cardinality = 20000;
public final int d_cardinality = 30000;
public final int hd_cardinality = 1000;
public final int ib_cardinality = 1000;
public final int inv_cardinality = 100;
public final int i_cardinality = 50000;
public final int p_cardinality = 1000;
public final int r_cardinality = 300;
public final int sm_cardinality = 50;
public final int s_cardinality = 200;
public final int sr_cardinality = 10000;
public final int ss_cardinality = 150000;
public final int t_cardinality = 50000;
public final int w_cardinality = 100;
public final int wp_cardinality = 1000;
public final int wr_cardinality = 10000;
public final int ws_cardinality = 150000;
```

```
public final int web_cardinality = 200;


//Random data methods

private int rint(int max){
// returns a random integer in the range 1..max
Random generator = new Random();
int rand = generator.nextInt(max) + 1 ;
return rand;
}

private int rintrange(int min,int max){
// returns a random integer in the range min..max
Random r = new Random();
int res = min + r.nextInt((max - min) + 1);
return res;
}

private double rdec(int tot, int dec){
// returns a random decimal with dec digits after comma and at most tot total digits
Random r = new Random();
double max = Math.pow(10, tot-dec) - 1;
double rand =  max * r.nextDouble();
int tmp = (int)(rand*(Math.pow(10, dec)));
rand = (double)(tmp/(Math.pow(10, dec)));
return rand ;
}

private String rvchar(int n){
// returns a random varchar(n) (length n/2 .. n) surrounded by quotes
Random r = new Random();
int l = r.nextInt((n - (n/2)) + 1) + (n/2);
String characters = "abcdefghijklmnopqrstuvwxyz0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";
char[] text = new char[l];
for (int i = 0; i < l; i++)
{
text[i] = characters.charAt(r.nextInt(characters.length()));
}
String s =  new String(text);
return "\'" + s + "\'";
}

private  String rchar(int n){
// returns a random char(n) (length exactly n) surrounded be quotes
Random r = new Random();
String characters = "abcdefghijklmnopqrstuvwxyz0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";
char[] text = new char[n];
for (int i = 0; i < n; i++)
{
text[i] = characters.charAt(r.nextInt(characters.length()));
}
String s =  new String(text);
return "\'" + s + "\'";
}

private String rdate(int minyear,int maxyear){

Random r = new Random();
int day = 1 + r.nextInt((28 - 1) + 1);
int month = 1 + r.nextInt((12 - 1) + 1);
int year = minyear + r.nextInt((maxyear - minyear) + 1);
String d = "\'" + year + "-" + month + "-" + day + "\'";
```

```
return d;
}


private String rvalue(String[] vals){
int j = rintrange(0, vals.length-1);
return "\'" + vals[j] + "\'";
}


// Lists of possible values, for attributes that will be booleanized.....................

public final String[] cd_gender_vals = {"male","female"};

public final String[] cd_marital_status_vals = {"single","married","divorced"};

public final String[] cd_education_status_vals =
{"primary","secondary","university","postgraduate"};

public final String[] d_day_name_vals =
{"Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","Sunday"};

public final String[] d_month_name_vals =
{"January","February","March","April","May","June",
"July","August","September","October","November","December"};

public final String[] d_quarter_name_vals = {"first","second","third","fourth"};

public final String[] t_am_pm_vals = {"AM","PM"};

public final String[] yes_no_vals = {"yes","no"};

//Create table statements.....................................................

public final String cc_ctable =
"CREATE TABLE IF NOT EXISTS call_center ("
+ "cc_call_center_sk int not null auto_increment,"
+ "cc_call_center_id char(16) not null,"
+ "cc_rec_start_date date,"
+ "cc_rec_end_date date,"
+ "cc_closed_date_sk int,"
+ "cc_open_date_sk int,"
+ "cc_name varchar(50),"
+ "cc_class varchar(50),"
+ "cc_employees int,"
+ "cc_sq_ft int,"
+ "cc_hours char(20),"
+ "cc_manager varchar(40),"
+ "cc_mkt_id int,"
+ "cc_mkt_class char(50),"
+ "cc_mkt_desc varchar(100),"
+ "cc_market_manager varchar(40),"
+ "cc_division integer,"
+ "cc_division_name varchar(50),"
+ "cc_company integer,"
+ "cc_company_name char(50),"
+ "cc_street_number char(10),"
+ "cc_street_name varchar(60),"
+ "cc_street_type char(15),"
+ "cc_suite_number char(10),"
+ "cc_city varchar(60),"
+ "cc_county varchar(30),"
+ "cc_state char(2),"
```

```
+ "cc_zip char(10),"
+ "cc_country varchar(20),"
+ "cc_gmt_offset decimal(5,2),"
+ "cc_tax_percentage decimal(5,2),"
+ "primary key (cc_call_center_sk),"
+ "foreign key (cc_closed_date_sk) references date_dim (d_date_sk),"
+ "foreign key (cc_open_date_sk) references date_dim (d_date_sk)"
+ ")";

public final String cp_ctable =
"CREATE TABLE IF NOT EXISTS catalog_page ("
+ "cp_catalog_page_sk int not null auto_increment,"
+ "cp_catalog_page_id char(16) not null,"
+ "cp_start_date_sk int,"
+ "cp_end_date_sk int,"
+ "cp_department varchar(50),"
+ "cp_catalog_number int,"
+ "cp_catalog_page_number int,"
+ "cp_description varchar(100),"
+ "cp_type varchar(100),"
+ "primary key (cp_catalog_page_sk),"
+ "foreign key (cp_start_date_sk) references date_dim (d_date_sk),"
+ "foreign key (cp_end_date_sk) references date_dim (d_date_sk)"
+ ")";

public final String cr_ctable =
"CREATE TABLE IF NOT EXISTS catalog_returns ("
+ "cr_returned_date_sk int,"
+ "cr_returned_time_sk int,"
+ "cr_item_sk int not null,"
+ "cr_refunded_customer_sk int,"
+ "cr_refunded_cdemo_sk int,"
+ "cr_refunded_hdemo_sk int,"
+ "cr_refunded_addr_sk int,"
+ "cr_returning_customer_sk int,"
+ "cr_returning_cdemo_sk int,"
+ "cr_returning_hdemo_sk int,"
+ "cr_returning_addr_sk int,"
+ "cr_call_center_sk int,"
+ "cr_catalog_page_sk int,"
+ "cr_ship_mode_sk int,"
+ "cr_warehouse_sk int,"
+ "cr_reason_sk int,"
+ "cr_order_number int not null,"
+ "cr_return_quantity int,"
+ "cr_return_amount decimal(7,2),"
+ "cr_return_tax decimal(7,2),"
+ "cr_return_amt_inc_tax decimal(7,2),"
+ "cr_fee decimal(7,2),"
+ "cr_return_ship_cost decimal(7,2),"
+ "cr_refunded_cash decimal(7,2),"
+ "cr_reversed_charge decimal(7,2),"
+ "cr_store_credit decimal(7,2),"
+ "cr_net_loss decimal(7,2),"
+ "primary key (cr_item_sk, cr_order_number),"
+ "foreign key (cr_returned_date_sk) references date_dim (d_date_sk),"
+ "foreign key (cr_returned_time_sk) references time_dim (t_time_sk),"
+ "foreign key (cr_item_sk) references item (i_item_sk),"
+ "foreign key (cr_refunded_customer_sk) references customer (c_customer_sk),"
+ "foreign key (cr_refunded_cdemo_sk) references customer_demographics (cd_demo_sk),"
+ "foreign key (cr_refunded_hdemo_sk) references household_demographics (hd_demo_sk),"
+ "foreign key (cr_refunded_addr_sk) references customer_address (ca_address_sk),"
+ "foreign key (cr_returning_customer_sk) references customer (c_customer_sk),"
```

```
+ "foreign key (cr_returning_cdemo_sk) references customer_demographics (cd_demo_sk),"
+ "foreign key (cr_returning_hdemo_sk) references household_demographics (hd_demo_sk),"
+ "foreign key (cr_returning_addr_sk) references customer_address (ca_address_sk),"
+ "foreign key (cr_call_center_sk) references call_center (cc_call_center_sk),"
+ "foreign key (cr_catalog_page_sk) references catalog_page (cp_catalog_page_sk),"
+ "foreign key (cr_ship_mode_sk) references ship_mode (sm_ship_mode_sk),"
+ "foreign key (cr_warehouse_sk) references warehouse (w_warehouse_sk),"
+ "foreign key (cr_reason_sk) references reason (r_reason_sk),"
+ "foreign key (cr_item_sk, cr_order_number) references catalog_sales (cs_item_sk,
cs_order_number) on delete cascade"
+ ")";

public final String cs_ctable =
"CREATE TABLE IF NOT EXISTS catalog_sales ("
+ "cs_sold_date_sk int,"
+ "cs_sold_time_sk int,"
+ "cs_ship_date_sk int,"
+ "cs_bill_customer_sk int,"
+ "cs_bill_cdemo_sk int,"
+ "cs_bill_hdemo_sk int,"
+ "cs_bill_addr_sk int,"
+ "cs_ship_customer_sk int,"
+ "cs_ship_cdemo_sk int,"
+ "cs_ship_hdemo_sk int,"
+ "cs_ship_addr_sk int,"
+ "cs_call_center_sk int,"
+ "cs_catalog_page_sk int,"
+ "cs_ship_mode_sk int,"
+ "cs_warehouse_sk int,"
+ "cs_item_sk int not null,"
+ "cs_promo_sk int,"
+ "cs_order_number int not null,"
+ "cs_quantity int,"
+ "cs_wholesale_cost decimal(7,2),"
+ "cs_list_price decimal(7,2),"
+ "cs_sales_price decimal(7,2),"
+ "cs_ext_discount_amt decimal(7,2),"
+ "cs_ext_sales_price decimal(7,2),"
+ "cs_ext_wholesale_cost decimal(7,2),"
+ "cs_ext_list_price decimal(7,2),"
+ "cs_ext_tax decimal(7,2),"
+ "cs_coupon_amt decimal(7,2),"
+ "cs_ext_ship_cost decimal(7,2) ,"
+ "cs_net_paid decimal(7,2),"
+ "cs_net_paid_inc_tax decimal(7,2),"
+ "cs_net_paid_inc_ship decimal(7,2) ,"
+ "cs_net_paid_inc_ship_tax decimal(7,2) ,"
+ "cs_net_profit decimal(7,2),"
+ "primary key (cs_item_sk,cs_order_number),"
+ "foreign key (cs_sold_date_sk) references date_dim (d_date_sk),"
+ "foreign key (cs_sold_time_sk) references time_dim (t_time_sk),"
+ "foreign key (cs_ship_date_sk) references date_dim (d_date_sk),"
+ "foreign key (cs_bill_customer_sk) references customer (c_customer_sk),"
+ "foreign key (cs_bill_cdemo_sk) references customer_demographics (cd_demo_sk),"
+ "foreign key (cs_bill_hdemo_sk) references household_demographics (hd_demo_sk),"
+ "foreign key (cs_bill_addr_sk) references customer_address (ca_address_sk),"
+ "foreign key (cs_ship_customer_sk) references customer (c_customer_sk),"
+ "foreign key (cs_ship_cdemo_sk) references customer_demographics (cd_demo_sk),"
+ "foreign key (cs_ship_hdemo_sk) references household_demographics (hd_demo_sk),"
+ "foreign key (cs_ship_addr_sk) references customer_address (ca_address_sk),"
+ "foreign key (cs_call_center_sk) references call_center (cc_call_center_sk),"
+ "foreign key (cs_catalog_page_sk) references catalog_page (cp_catalog_page_sk),"
+ "foreign key (cs_ship_mode_sk) references ship_mode (sm_ship_mode_sk),"
```

```
+ "foreign key (cs_warehouse_sk) references warehouse (w_warehouse_sk),"
+ "foreign key (cs_item_sk) references item (i_item_sk),"
+ "foreign key (cs_promo_sk) references promotion (p_promo_sk)"
+ ")";

public final String c_ctable =
"CREATE TABLE IF NOT EXISTS customer ("
+ "c_customer_sk int not null auto_increment,"
+ "c_customer_id char(16) not null,"
+ "c_current_cdemo_sk int,"
+ "c_current_hdemo_sk int,"
+ "c_current_addr_sk int,"
+ "c_first_shipto_date_sk int,"
+ "c_first_sales_date_sk int,"
+ "c_salutation char(10),"
+ "c_first_name char(20),"
+ "c_last_name char(30),"
+ "c_preferred_cust_flag char(3),"
+ "c_birth_day int,"
+ "c_birth_month int,"
+ "c_birth_year int,"
+ "c_birth_country varchar(20),"
+ "c_login char(13),"
+ "c_email_address char(50),"
+ "c_last_review_date_sk int,"
+ "primary key (c_customer_sk),"
+ "foreign key (c_current_cdemo_sk) references customer_demographics (cd_demo_sk),"
+ "foreign key (c_current_hdemo_sk) references household_demographics (hd_demo_sk),"
+ "foreign key (c_current_addr_sk) references customer_address (ca_address_sk),"
+ "foreign key (c_first_shipto_date_sk) references date_dim (d_date_sk),"
+ "foreign key (c_first_sales_date_sk) references date_dim (d_date_sk),"
+ "foreign key (c_last_review_date_sk) references date_dim (d_date_sk)"
+ ")";

public final String ca_ctable =
"CREATE TABLE IF NOT EXISTS customer_address ("
+ "ca_address_sk int not null auto_increment,"
+ "ca_address_id char(16) not null,"
+ "ca_street_number char(10),"
+ "ca_street_name varchar(60),"
+ "ca_street_type char(15),"
+ "ca_suite_number char(10),"
+ "ca_city varchar(60),"
+ "ca_county varchar(30),"
+ "ca_state char(2),"
+ "ca_zip char(10),"
+ "ca_country varchar(20),"
+ "ca_gmt_offset decimal(5,2),"
+ "ca_location_type char(20),"
+ "primary key (ca_address_sk)"
+ ")";

public final String cd_ctable =
"CREATE TABLE IF NOT EXISTS customer_demographics ("
+ "cd_demo_sk int not null auto_increment,"
+ "cd_gender char(6),"
+ "cd_marital_status char(8),"
+ "cd_education_status char(20),"
+ "cd_purchase_estimate int,"
+ "cd_credit_rating char(10),"
+ "cd_dep_count int,"
+ "cd_dep_employed_count int,"
+ "cd_dep_college_count int,"
```

```
+ "primary key (cd_demo_sk)"
+ ")";

public final String d_ctable =
"CREATE TABLE IF NOT EXISTS date_dim ("
+ "d_date_sk int not null auto_increment,"
+ "d_date_id char(16) not null,"
+ "d_date date,"
+ "d_month_seq int,"
+ "d_week_seq int,"
+ "d_quarter_seq int,"
+ "d_year int,"
+ "d_dow int,"
+ "d_moy int,"
+ "d_dom int,"
+ "d_qoy int,"
+ "d_fy_year int,"
+ "d_fy_quarter_seq int,"
+ "d_fy_week_seq int,"
+ "d_day_name char(9),"
+ "d_month_name char(15),"
+ "d_quarter_name char(6),"
+ "d_holiday char(3),"
+ "d_weekend char(3),"
+ "d_following_holiday char(3),"
+ "d_first_dom int,"
+ "d_last_dom int,"
+ "d_same_day_ly int,"
+ "d_same_day_lq int,"
+ "d_current_day char(3),"
+ "d_current_week char(3),"
+ "d_current_month char(3),"
+ "d_current_quarter char(3),"
+ "d_current_year char(3),"
+ "primary key (d_date_sk)"
+ ")";

public final String hd_ctable =
"CREATE TABLE IF NOT EXISTS household_demographics ("
+ "hd_demo_sk int not null auto_increment,"
+ "hd_income_band_sk int,"
+ "hd_buy_potential char(15),"
+ "hd_dep_count int,"
+ "hd_vehicle_count int,"
+ "primary key (hd_demo_sk),"
+ "foreign key (hd_income_band_sk) references income_band (ib_income_band_sk)"
+ ")";

public final String ib_ctable =
"CREATE TABLE IF NOT EXISTS income_band ("
+ "ib_income_band_sk int not null auto_increment,"
+ "ib_lower_bound int,"
+ "ib_upper_bound int,"
+ "primary key (ib_income_band_sk)"
+ ")";

public final String inv_ctable =
"CREATE TABLE IF NOT EXISTS inventory ("
+ "inv_date_sk int not null,"
+ "inv_item_sk int not null,"
+ "inv_warehouse_sk int not null,"
+ "inv_quantity_on_hand int,"
+ "primary key (inv_date_sk,inv_item_sk,inv_warehouse_sk),"
```

```
+ "foreign key (inv_date_sk) references date_dim (d_date_sk),"
+ "foreign key (inv_item_sk) references item (i_item_sk),"
+ "foreign key (inv_warehouse_sk) references warehouse (w_warehouse_sk)"
+ ")";

public final String i_ctable =
"CREATE TABLE IF NOT EXISTS item ("
+ "i_item_sk int not null auto_increment,"
+ "i_item_id char(16) not null,"
+ "i_rec_start_date date,"
+ "i_rec_end_date date,"
+ "i_item_desc varchar(200),"
+ "i_current_price decimal(7,2),"
+ "i_wholesale_cost decimal(7,2),"
+ "i_brand_id int,"
+ "i_brand char(50),"
+ "i_class_id int,"
+ "i_class char(50),"
+ "i_category_id int,"
+ "i_category char(50),"
+ "i_manufact_id int,"
+ "i_manufact char(50),"
+ "i_size char(20),"
+ "i_formulation char(20),"
+ "i_color char(20),"
+ "i_units char(10),"
+ "i_container char(10),"
+ "i_manager_id int,"
+ "i_product_name char(50),"
+ "primary key (i_item_sk)"
+ ")";

public final String p_ctable =
"CREATE TABLE IF NOT EXISTS promotion ("
+ "p_promo_sk int not null auto_increment,"
+ "p_promo_id char(16) not null,"
+ "p_start_date_sk int,"
+ "p_end_date_sk int,"
+ "p_item_sk int,"
+ "p_cost decimal(15,2),"
+ "p_response_target int,"
+ "p_promo_name char(50),"
+ "p_channel_dmail char(3),"
+ "p_channel_email char(3),"
+ "p_channel_catalog char(3),"
+ "p_channel_tv char(3),"
+ "p_channel_radio char(3),"
+ "p_channel_press char(3),"
+ "p_channel_event char(3),"
+ "p_channel_demo char(3),"
+ "p_channel_details varchar(100),"
+ "p_purpose char(15),"
+ "p_discount_active char(3),"
+ "primary key (p_promo_sk),"
+ "foreign key (p_start_date_sk) references date_dim (d_date_sk),"
+ "foreign key (p_end_date_sk) references date_dim (d_date_sk),"
+ "foreign key (p_item_sk) references item (i_item_sk)"
+ ")";

public final String r_ctable =
"CREATE TABLE IF NOT EXISTS reason ("
+ "r_reason_sk int not null auto_increment,"
+ "r_reason_id char(16) not null,"
```

```
+ "r_reason_desc char(100),"
+ "primary key (r_reason_sk)"
+ ")";

public final String sm_ctable =
"CREATE TABLE IF NOT EXISTS ship_mode ("
+ "sm_ship_mode_sk int not null auto_increment,"
+ "sm_ship_mode_id char(16) not null,"
+ "sm_type char(30),"
+ "sm_code char(10),"
+ "sm_carrier char(20),"
+ "sm_contract char(20),"
+ "primary key (sm_ship_mode_sk)"
+ ")";

public final String s_ctable =
"CREATE TABLE IF NOT EXISTS store ("
+ "s_store_sk int not null auto_increment,"
+ "s_store_id char(16) not null,"
+ "s_rec_start_date date,"
+ "s_rec_end_date date,"
+ "s_closed_date_sk int,"
+ "s_store_name varchar(50),"
+ "s_number_employees int,"
+ "s_floor_space int,"
+ "s_hours char(20),"
+ "s_manager varchar(40),"
+ "s_market_id int,"
+ "s_geography_class varchar(100),"
+ "s_market_desc varchar(100),"
+ "s_market_manager varchar(40),"
+ "s_division_id int,"
+ "s_division_name varchar(50),"
+ "s_company_id int,"
+ "s_company_name varchar(50),"
+ "s_street_number varchar(10),"
+ "s_street_name varchar(60),"
+ "s_street_type char(15),"
+ "s_suite_number char(10),"
+ "s_city varchar(60),"
+ "s_county varchar(30),"
+ "s_state char(2),"
+ "s_zip char(10),"
+ "s_country varchar(20),"
+ "s_gmt_offset decimal(5,2),"
+ "s_tax_percentage decimal(5,2),"
+ "primary key (s_store_sk),"
+ "foreign key (s_closed_date_sk) references date_dim (d_date_sk)"
+ ")";

public final String sr_ctable =
"CREATE TABLE IF NOT EXISTS store_returns ("
+ "sr_returned_date_sk int,"
+ "sr_returned_time_sk int,"
+ "sr_item_sk int not null,"
+ "sr_customer_sk int,"
+ "sr_cdemo_sk int,"
+ "sr_hdemo_sk int,"
+ "sr_addr_sk int,"
+ "sr_store_sk int,"
+ "sr_reason_sk int,"
+ "sr_ticket_number int not null,"
+ "sr_return_quantity int,"
```

```
+ "sr_return_amt decimal(7,2),"
+ "sr_return_tax decimal(7,2),"
+ "sr_return_amt_inc_tax decimal(7,2),"
+ "sr_fee decimal(7,2),"
+ "sr_return_ship_cost decimal(7,2),"
+ "sr_refunded_cash decimal(7,2),"
+ "sr_reversed_charge decimal(7,2),"
+ "sr_store_credit decimal(7,2),"
+ "sr_net_loss decimal(7,2),"
+ "primary key (sr_item_sk,sr_ticket_number),"
+ "foreign key (sr_returned_date_sk) references date_dim (d_date_sk),"
+ "foreign key (sr_returned_time_sk) references time_dim (t_time_sk),"
+ "foreign key (sr_item_sk) references item (i_item_sk),"
+ "foreign key (sr_customer_sk) references customer (c_customer_sk),"
+ "foreign key (sr_cdemo_sk) references customer_demographics (cd_demo_sk),"
+ "foreign key (sr_hdemo_sk) references household_demographics (hd_demo_sk),"
+ "foreign key (sr_addr_sk) references customer_address (ca_address_sk),"
+ "foreign key (sr_store_sk) references store (s_store_sk),"
+ "foreign key (sr_reason_sk) references reason (r_reason_sk),"
+ "foreign key (sr_item_sk,sr_ticket_number) references store_sales (ss_item_sk,ss_ticket_number)
on delete cascade"
+ ")";

public final String ss_ctable =
"CREATE TABLE IF NOT EXISTS store_sales ("
+ "ss_sold_date_sk int,"
+ "ss_sold_time_sk int,"
+ "ss_item_sk int not null,"
+ "ss_customer_sk int,"
+ "ss_cdemo_sk int,"
+ "ss_hdemo_sk int,"
+ "ss_addr_sk int,"
+ "ss_store_sk int,"
+ "ss_promo_sk int,"
+ "ss_ticket_number int not null,"
+ "ss_quantity int,"
+ "ss_wholesale_cost decimal(7,2),"
+ "ss_list_price decimal(7,2),"
+ "ss_sales_price decimal(7,2),"
+ "ss_ext_discount_amt decimal(7,2),"
+ "ss_ext_sales_price decimal(7,2),"
+ "ss_ext_wholesale_cost decimal(7,2),"
+ "ss_ext_list_price decimal(7,2),"
+ "ss_ext_tax decimal(7,2),"
+ "ss_coupon_amt decimal(7,2),"
+ "ss_net_paid decimal(7,2),"
+ "ss_net_paid_inc_tax decimal(7,2),"
+ "ss_net_profit decimal(7,2),"
+ "primary key (ss_item_sk,ss_ticket_number),"
+ "foreign key (ss_sold_date_sk) references date_dim (d_date_sk),"
+ "foreign key (ss_sold_time_sk) references time_dim (t_time_sk),"
+ "foreign key (ss_item_sk) references item (i_item_sk),"
+ "foreign key (ss_customer_sk) references customer (c_customer_sk),"
+ "foreign key (ss_cdemo_sk) references customer_demographics (cd_demo_sk),"
+ "foreign key (ss_hdemo_sk) references household_demographics (hd_demo_sk),"
+ "foreign key (ss_addr_sk) references customer_address (ca_address_sk),"
+ "foreign key (ss_store_sk) references store (s_store_sk),"
+ "foreign key (ss_promo_sk) references promotion (p_promo_sk)"
+ ")";

public final String t_ctable =
"CREATE TABLE IF NOT EXISTS time_dim ("
+ "t_time_sk int not null auto_increment,"
```

```
+ "t_time_id char(16) not null,"
+ "t_time int,"
+ "t_hour int,"
+ "t_minute int,"
+ "t_second int,"
+ "t_am_pm char(2),"
+ "t_shift char(20),"
+ "t_sub_shift char(20),"
+ "t_meal_time char(20),"
+ "primary key (t_time_sk)"
+ ")";

public final String w_ctable =
"CREATE TABLE IF NOT EXISTS warehouse ("
+ "w_warehouse_sk int not null auto_increment,"
+ "w_warehouse_id char(16) not null,"
+ "w_warehouse_name varchar(20),"
+ "w_warehouse_sq_ft int,"
+ "w_street_number char(10),"
+ "w_street_name varchar(60),"
+ "w_street_type char(15),"
+ "w_suite_number char(10),"
+ "w_city varchar(60),"
+ "w_county varchar(30),"
+ "w_state char(2),"
+ "w_zip char(10),"
+ "w_country varchar(20),"
+ "w_gmt_offset decimal(5,2),"
+ "primary key (w_warehouse_sk)"
+ ")";

public final String wp_ctable =
"CREATE TABLE IF NOT EXISTS web_page ("
+ "wp_web_page_sk int not null auto_increment,"
+ "wp_web_page_id char(16) not null,"
+ "wp_rec_start_date date,"
+ "wp_rec_end_date date,"
+ "wp_creation_date_sk int,"
+ "wp_access_date_sk int,"
+ "wp_autogen_flag char(3),"
+ "wp_customer_sk int,"
+ "wp_url varchar(100),"
+ "wp_type char(50),"
+ "wp_char_count int,"
+ "wp_link_count int,"
+ "wp_image_count int,"
+ "wp_max_ad_count int,"
+ "primary key (wp_web_page_sk),"
+ "foreign key (wp_creation_date_sk) references date_dim (d_date_sk),"
+ "foreign key (wp_access_date_sk) references date_dim (d_date_sk),"
+ "foreign key (wp_customer_sk) references customer (c_customer_sk)"
+ ")";

public final String wr_ctable =
"CREATE TABLE IF NOT EXISTS web_returns ("
+ "wr_returned_date_sk int,"
+ "wr_returned_time_sk int,"
+ "wr_item_sk int not null,"
+ "wr_refunded_customer_sk int,"
+ "wr_refunded_cdemo_sk int,"
+ "wr_refunded_hdemo_sk int,"
+ "wr_refunded_addr_sk int,"
+ "wr_returning_customer_sk int,"
```

```
+ "wr_returning_cdemo_sk int,"
+ "wr_returning_hdemo_sk int,"
+ "wr_returning_addr_sk int,"
+ "wr_web_page_sk int,"
+ "wr_reason_sk int,"
+ "wr_order_number int not null,"
+ "wr_return_quantity int,"
+ "wr_return_amt decimal(7,2),"
+ "wr_return_tax decimal(7,2),"
+ "wr_return_amt_inc_tax decimal(7,2),"
+ "wr_fee decimal(7,2),"
+ "wr_return_ship_cost decimal(7,2),"
+ "wr_refunded_cash decimal(7,2),"
+ "wr_reversed_charge decimal(7,2),"
+ "wr_account_credit decimal(7,2),"
+ "wr_net_loss decimal(7,2),"
+ "primary key (wr_item_sk,wr_order_number),"
+ "foreign key (wr_returned_date_sk) references date_dim (d_date_sk),"
+ "foreign key (wr_returned_time_sk) references time_dim (t_time_sk),"
+ "foreign key (wr_item_sk) references item (i_item_sk),"
+ "foreign key (wr_refunded_customer_sk) references customer (c_customer_sk),"
+ "foreign key (wr_refunded_cdemo_sk) references customer_demographics (cd_demo_sk),"
+ "foreign key (wr_refunded_hdemo_sk) references household_demographics (hd_demo_sk),"
+ "foreign key (wr_refunded_addr_sk) references customer_address (ca_address_sk),"
+ "foreign key (wr_returning_customer_sk) references customer (c_customer_sk),"
+ "foreign key (wr_returning_cdemo_sk) references customer_demographics (cd_demo_sk),"
+ "foreign key (wr_returning_hdemo_sk) references household_demographics (hd_demo_sk),"
+ "foreign key (wr_returning_addr_sk) references customer_address (ca_address_sk),"
+ "foreign key (wr_web_page_sk) references web_page (wp_web_page_sk),"
+ "foreign key (wr_reason_sk) references reason (r_reason_sk),"
+ "foreign key (wr_item_sk,wr_order_number) references web_sales (ws_item_sk,ws_order_number) on
delete cascade"
+ ")";

public final String ws_ctable =
"CREATE TABLE IF NOT EXISTS web_sales ("
+ "ws_sold_date_sk int,"
+ "ws_sold_time_sk int,"
+ "ws_ship_date_sk int,"
+ "ws_item_sk int not null,"
+ "ws_bill_customer_sk int,"
+ "ws_bill_cdemo_sk int,"
+ "ws_bill_hdemo_sk int,"
+ "ws_bill_addr_sk int,"
+ "ws_ship_customer_sk int,"
+ "ws_ship_cdemo_sk int,"
+ "ws_ship_hdemo_sk int,"
+ "ws_ship_addr_sk int,"
+ "ws_web_page_sk int,"
+ "ws_web_site_sk int,"
+ "ws_ship_mode_sk int,"
+ "ws_warehouse_sk int,"
+ "ws_promo_sk int,"
+ "ws_order_number int not null,"
+ "ws_quantity int,"
+ "ws_wholesale_cost decimal(7,2),"
+ "ws_list_price decimal(7,2),"
+ "ws_sales_price decimal(7,2),"
+ "ws_ext_discount_amt decimal(7,2),"
+ "ws_ext_sales_price decimal(7,2),"
+ "ws_ext_wholesale_cost decimal(7,2),"
+ "ws_ext_list_price decimal(7,2),"
+ "ws_ext_tax decimal(7,2),"
```

```
+ "ws_coupon_amt decimal(7,2),"
+ "ws_ext_ship_cost decimal(7,2),"
+ "ws_net_paid decimal(7,2),"
+ "ws_net_paid_inc_tax decimal(7,2),"
+ "ws_net_paid_inc_ship decimal(7,2),"
+ "ws_net_paid_inc_ship_tax decimal(7,2),"
+ "ws_net_profit decimal(7,2),"
+ "primary key (ws_item_sk, ws_order_number),"
+ "foreign key (ws_sold_date_sk) references date_dim (d_date_sk),"
+ "foreign key (ws_sold_time_sk) references time_dim (t_time_sk),"
+ "foreign key (ws_ship_date_sk) references date_dim (d_date_sk),"
+ "foreign key (ws_item_sk) references item (i_item_sk),"
+ "foreign key (ws_bill_customer_sk) references customer (c_customer_sk),"
+ "foreign key (ws_bill_cdemo_sk) references customer_demographics (cd_demo_sk),"
+ "foreign key (ws_bill_hdemo_sk) references household_demographics (hd_demo_sk),"
+ "foreign key (ws_bill_addr_sk) references customer_address (ca_address_sk),"
+ "foreign key (ws_ship_customer_sk) references customer (c_customer_sk),"
+ "foreign key (ws_ship_cdemo_sk) references customer_demographics (cd_demo_sk),"
+ "foreign key (ws_ship_hdemo_sk) references household_demographics (hd_demo_sk),"
+ "foreign key (ws_ship_addr_sk) references customer_address (ca_address_sk),"
+ "foreign key (ws_web_page_sk) references web_page (wp_web_page_sk),"
+ "foreign key (ws_web_site_sk) references web_site (web_site_sk),"
+ "foreign key (ws_ship_mode_sk) references ship_mode (sm_ship_mode_sk),"
+ "foreign key (ws_warehouse_sk) references warehouse (w_warehouse_sk),"
+ "foreign key (ws_promo_sk) references promotion (p_promo_sk)"
+ ")";

public final String web_ctable =
"CREATE TABLE IF NOT EXISTS web_site ("
+ "web_site_sk int not null auto_increment,"
+ "web_site_id char(16) not null,"
+ "web_rec_start_date date,"
+ "web_rec_end_date date,"
+ "web_name varchar(50),"
+ "web_open_date_sk int,"
+ "web_close_date_sk int,"
+ "web_class varchar(50),"
+ "web_manager varchar(40),"
+ "web_mkt_id int,"
+ "web_mkt_class varchar(50),"
+ "web_mkt_desc varchar(100),"
+ "web_market_manager varchar(40),"
+ "web_company_id int,"
+ "web_company_name char(50),"
+ "web_street_number char(10),"
+ "web_street_name varchar(60),"
+ "web_street_type char(15),"
+ "web_suite_number char(10),"
+ "web_city varchar(60),"
+ "web_county varchar(30),"
+ "web_state char(2),"
+ "web_zip char(10),"
+ "web_country varchar(20),"
+ "web_gmt_offset decimal(5,2),"
+ "web_tax_percentage decimal(5,2),"
+ "primary key (web_site_sk),"
+ "foreign key (web_open_date_sk) references date_dim (d_date_sk),"
+ "foreign key (web_close_date_sk) references date_dim (d_date_sk)"
+ ")";

//#########################################################################################################
###########
```

```
// This method says if a pair (item_sk,order_number) is already used in the table

private static boolean used(int sk,int num,String table,String attrsel,String attrsk,String
attrnum){
boolean result;
Connection conn = null;
Statement stmt = null;
try{
Class.forName("com.mysql.jdbc.Driver");
conn = DriverManager.getConnection(DB_SERVER_URL+DBNAME1, USER, PASS);
stmt = conn.createStatement();

String sql = "SELECT " + attrsel
+ " FROM " + table
+ " WHERE " + attrsk  + " = " + sk
+ " AND "  + attrnum + " = " + num
+ " ;";

ResultSet rs = stmt.executeQuery(sql);

if(rs.next()){
result = true;
}
else{
result = false;
}

rs.close();

return result;

}
catch(SQLException se){
se.printStackTrace();
return false;
}catch(Exception e){
e.printStackTrace();
return false;
}finally{
try{
if(stmt!=null)
conn.close();
}catch(SQLException se){
}// do nothing
try{
if(conn!=null)
conn.close();
}catch(SQLException se){
se.printStackTrace();
}//end finally try
}//end try
}

//This method just (re)creates the tpcds database

public void createBasicDB(){
Connection conn1 = null;
Statement stmt1 = null;
String sql;

try{
Class.forName(JDBC_DRIVER);
```

```
conn1 = DriverManager.getConnection(DB_SERVER_URL,USER,PASS);
stmt1 = conn1.createStatement();

sql = "DROP DATABASE IF EXISTS " + DBNAME1;
stmt1.executeUpdate(sql);

sql = "CREATE DATABASE " + DBNAME1;
stmt1.executeUpdate(sql);

System.out.println("CREATED DATABASE : " + DBNAME1);

stmt1.close();
conn1.close();

}
catch(SQLException se){
//Handle errors for JDBC
se.printStackTrace();
}
catch(Exception e){
//Handle errors for Class.forName
e.printStackTrace();
}
finally{
//finally block used to close resources
try{
if(stmt1!=null)
stmt1.close();
}
catch(SQLException se2){
}// nothing we can do
try{
if(conn1!=null)
conn1.close();
}
catch(SQLException se){
se.printStackTrace();
}
}
}


// One method to create and populate each table, for modularity

public void createTable_cc(){
Connection conn1 = null;
Statement stmt1 = null;
String sql;
int cardinality;


try{
Class.forName(JDBC_DRIVER);
conn1 = DriverManager.getConnection(DB_SERVER_URL + DBNAME1,USER,PASS);
stmt1 = conn1.createStatement();

sql = "DROP TABLE IF EXISTS " + "call_center" ;
stmt1.executeUpdate(sql);

sql = cc_ctable ;
stmt1.executeUpdate(sql);

cardinality = cc_cardinality;
```

```
for(int i=1;i<=cardinality;i++){
sql =
"INSERT INTO call_center "
+ "(cc_call_center_id, cc_rec_start_date, cc_rec_end_date, cc_closed_date_sk,"
+ "cc_open_date_sk, cc_name, cc_class, cc_employees, cc_sq_ft, cc_hours,"
+ "cc_manager, cc_mkt_id, cc_mkt_class, cc_mkt_desc, cc_market_manager, "
+ "cc_division, cc_division_name, cc_company, cc_company_name, cc_street_number,"
+ "cc_street_name, cc_street_type, cc_suite_number, cc_city, cc_county, cc_state,"
+ "cc_zip, cc_country, cc_gmt_offset, cc_tax_percentage) "
+ "VALUES ("
+ rchar(16) + ","
+ rdate(2000,2005) + ","
+ rdate(2005,2014) + ","
+ rint(d_cardinality) + ","
+ rint(d_cardinality) + ","
+ rvchar(50) + ","
+ rvchar(50) + ","
+ rintrange(20, 500) + ","
+ rintrange(100, 2500) + ","
+ rchar(20) + ","
+ rvchar(40) + ","
+ rint(1000) + ","
+ rchar(50) + ","
+ rvchar(100) + ","
+ rvchar(40) + ","
+ rint(1000) + ","
+ rvchar(50) + ","
+ rint(1000) + ","
+ rchar(50) + ","
+ rchar(10) + ","
+ rvchar(60) + ","
+ rchar(15) + ","
+ rchar(10) + ","
+ rvchar(60) + ","
+ rvchar(30) + ","
+ rchar(2) + ","
+ rchar(10) + ","
+ rvchar(20) + ","
+ rdec(5,2) + ","
+ rdec(5,2)
+ ")";

stmt1.executeUpdate(sql);
}

System.out.println("Table \'call_center\' created");

stmt1.close();
conn1.close();

}
catch(SQLException se){
//Handle errors for JDBC
se.printStackTrace();
}
catch(Exception e){
//Handle errors for Class.forName
e.printStackTrace();
}
finally{
//finally block used to close resources
try{
if(stmt1!=null)
```

```
stmt1.close();
}
catch(SQLException se2){
}// nothing we can do
try{
if(conn1!=null)
conn1.close();
}
catch(SQLException se){
se.printStackTrace();
}
}

}


public void createTable_cp(){
Connection conn1 = null;
Statement stmt1 = null;
String sql;
int cardinality;


try{
Class.forName(JDBC_DRIVER);
conn1 = DriverManager.getConnection(DB_SERVER_URL + DBNAME1,USER,PASS);
stmt1 = conn1.createStatement();

sql = "DROP TABLE IF EXISTS " + "catalog_page" ;
stmt1.executeUpdate(sql);

sql = cp_ctable ;
stmt1.executeUpdate(sql);

cardinality = cp_cardinality;
for(int i=1;i<=cardinality;i++){
sql =
"INSERT INTO catalog_page "
+ "(cp_catalog_page_id, cp_start_date_sk, cp_end_date_sk, cp_department,"
+ "cp_catalog_number, cp_catalog_page_number, cp_description, cp_type) "
+ "VALUES ("
+ rchar(16) + ","
+ rint(d_cardinality) + ","
+ rint(d_cardinality) + ","
+ rvchar(50) + ","
+ rint(3000) + ","
+ rint(2000) + ","
+ rvchar(100) + ","
+ rvchar(100)
+ ")";

stmt1.executeUpdate(sql);
}

System.out.println("Table \'catalog_page\' created");

stmt1.close();
conn1.close();

}
catch(SQLException se){
//Handle errors for JDBC
se.printStackTrace();
```

```
}
catch(Exception e){
//Handle errors for Class.forName
e.printStackTrace();
}
finally{
//finally block used to close resources
try{
if(stmt1!=null)
stmt1.close();
}
catch(SQLException se2){
}// nothing we can do
try{
if(conn1!=null)
conn1.close();
}
catch(SQLException se){
se.printStackTrace();
}
}

}


public void createTable_cs_cr(){
Connection conn1 = null;
Statement stmt1 = null;
String sql;
int rand_item_sk,rand_order_num;


try{
Class.forName(JDBC_DRIVER);
conn1 = DriverManager.getConnection(DB_SERVER_URL + DBNAME1,USER,PASS);
stmt1 = conn1.createStatement();

sql = "DROP TABLE IF EXISTS " + "catalog_returns" ;
stmt1.executeUpdate(sql);

sql = "DROP TABLE IF EXISTS " + "catalog_sales" ;
stmt1.executeUpdate(sql);

sql = cs_ctable ;
stmt1.executeUpdate(sql);

sql = cr_ctable ;
stmt1.executeUpdate(sql);

for(int i=1;i<=cs_cardinality;i++){
rand_item_sk = rint(i_cardinality);
rand_order_num = rint(10000);

while(used(rand_item_sk,rand_order_num,"catalog_sales","cs_sold_date_sk","cs_item_sk","cs_order_n
umber"))
{
rand_item_sk = rint(i_cardinality);
rand_order_num = rint(10000);
}
sql =
"INSERT INTO catalog_sales "
+ "(cs_sold_date_sk, cs_sold_time_sk, cs_ship_date_sk, cs_bill_customer_sk,"
+ " cs_bill_cdemo_sk, cs_bill_hdemo_sk, cs_bill_addr_sk, cs_ship_customer_sk,"
```

```
+ " cs_ship_cdemo_sk, cs_ship_hdemo_sk, cs_ship_addr_sk, cs_call_center_sk,"
+ " cs_catalog_page_sk, cs_ship_mode_sk, cs_warehouse_sk, cs_item_sk,"
+ " cs_promo_sk, cs_order_number, cs_quantity, cs_wholesale_cost, cs_list_price,"
+ " cs_sales_price, cs_ext_discount_amt, cs_ext_sales_price, cs_ext_wholesale_cost,"
+ " cs_ext_list_price, cs_ext_tax, cs_coupon_amt, cs_ext_ship_cost, cs_net_paid,"
+ " cs_net_paid_inc_tax, cs_net_paid_inc_ship, cs_net_paid_inc_ship_tax, "
+ " cs_net_profit)"
+ "VALUES ("
+ rint(d_cardinality) + ","
+ rint(t_cardinality) + ","
+ rint(d_cardinality) + ","
+ rint(c_cardinality) + ","
+ rint(cd_cardinality) + ","
+ rint(hd_cardinality) + ","
+ rint(ca_cardinality) + ","
+ rint(c_cardinality) + ","
+ rint(cd_cardinality) + ","
+ rint(hd_cardinality) + ","
+ rint(ca_cardinality) + ","
+ rint(cc_cardinality) + ","
+ rint(cp_cardinality) + ","
+ rint(sm_cardinality) + ","
+ rint(w_cardinality) + ","
+ rand_item_sk + ","
+ rint(p_cardinality) + ","
+ rand_order_num + ","
+ rintrange(20,2000) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2)
+ ")";

stmt1.executeUpdate(sql);

if(i<=cr_cardinality){

sql =
"INSERT INTO catalog_returns "
+ "(cr_returned_date_sk, cr_returned_time_sk, cr_item_sk, cr_refunded_customer_sk,"
+ " cr_refunded_cdemo_sk, cr_refunded_hdemo_sk, cr_refunded_addr_sk,"
+ " cr_returning_customer_sk, cr_returning_cdemo_sk, cr_returning_hdemo_sk,"
+ " cr_returning_addr_sk, cr_call_center_sk, cr_catalog_page_sk, cr_ship_mode_sk,"
+ " cr_warehouse_sk, cr_reason_sk, cr_order_number, cr_return_quantity, "
+ " cr_return_amount, cr_return_tax, cr_return_amt_inc_tax, cr_fee, cr_return_ship_cost,"
+ " cr_refunded_cash, cr_reversed_charge, cr_store_credit,cr_net_loss) "
+ "VALUES ("
+ rint(d_cardinality) + ","
+ rint(t_cardinality) + ","
+ rand_item_sk + ","
+ rint(c_cardinality) + ","
+ rint(cd_cardinality) + ","
```

```
+ rint(hd_cardinality) + ","
+ rint(ca_cardinality) + ","
+ rint(c_cardinality) + ","
+ rint(cd_cardinality) + ","
+ rint(hd_cardinality) + ","
+ rint(ca_cardinality) + ","
+ rint(cc_cardinality) + ","
+ rint(cp_cardinality) + ","
+ rint(sm_cardinality) + ","
+ rint(w_cardinality) + ","
+ rint(r_cardinality) + ","
+ rand_order_num + ","
+ rintrange(20,2000) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2)
+ ")";

stmt1.executeUpdate(sql);
}
}

System.out.println("Table \'catalog_sales\' created");
System.out.println("Table \'catalog_returns\' created");

stmt1.close();
conn1.close();

}
catch(SQLException se){
//Handle errors for JDBC
se.printStackTrace();
}
catch(Exception e){
//Handle errors for Class.forName
e.printStackTrace();
}
finally{
//finally block used to close resources
try{
if(stmt1!=null)
stmt1.close();
}
catch(SQLException se2){
}// nothing we can do
try{
if(conn1!=null)
conn1.close();
}
catch(SQLException se){
se.printStackTrace();
}
}

}
```

```
public void createTable_c(){
Connection conn1 = null;
Statement stmt1 = null;
String sql;
int cardinality;


try{
Class.forName(JDBC_DRIVER);
conn1 = DriverManager.getConnection(DB_SERVER_URL + DBNAME1,USER,PASS);
stmt1 = conn1.createStatement();

sql = "DROP TABLE IF EXISTS " + "customer" ;
stmt1.executeUpdate(sql);

sql = c_ctable ;
stmt1.executeUpdate(sql);

cardinality = c_cardinality;
for(int i=1;i<=cardinality;i++){
sql =
"INSERT INTO customer "
+ "(c_customer_id, c_current_cdemo_sk, c_current_hdemo_sk, c_current_addr_sk, "
+ "c_first_shipto_date_sk, c_first_sales_date_sk, c_salutation, c_first_name,"
+ "c_last_name, c_preferred_cust_flag, c_birth_day, c_birth_month,"
+ "c_birth_year, c_birth_country, c_login, c_email_address, c_last_review_date_sk) "
+ "VALUES ("
+ rchar(16) + ","
+ rint(cd_cardinality) + ","
+ rint(hd_cardinality) + ","
+ rint(ca_cardinality) + ","
+ rint(d_cardinality) + ","
+ rint(d_cardinality) + ","
+ rchar(10) + ","
+ rchar(20) + ","
+ rchar(30) + ","
+ rvalue(yes_no_vals) + ","
+ rintrange(1,28) + ","
+ rintrange(1,12) + ","
+ rintrange(1954,2004) + ","
+ rvchar(20) + ","
+ rchar(13) + ","
+ rchar(50) + ","
+ rint(d_cardinality)
+ ")";

stmt1.executeUpdate(sql);
}

System.out.println("Table \'customer\' created");

stmt1.close();
conn1.close();

}
catch(SQLException se){
//Handle errors for JDBC
se.printStackTrace();
}
catch(Exception e){
//Handle errors for Class.forName
e.printStackTrace();
}
```

```
finally{
//finally block used to close resources
try{
if(stmt1!=null)
stmt1.close();
}
catch(SQLException se2){
}// nothing we can do
try{
if(conn1!=null)
conn1.close();
}
catch(SQLException se){
se.printStackTrace();
}
}

}


public void createTable_ca(){
Connection conn1 = null;
Statement stmt1 = null;
String sql;
int cardinality;


try{
Class.forName(JDBC_DRIVER);
conn1 = DriverManager.getConnection(DB_SERVER_URL + DBNAME1,USER,PASS);
stmt1 = conn1.createStatement();

sql = "DROP TABLE IF EXISTS " + "customer_address" ;
stmt1.executeUpdate(sql);

sql = ca_ctable ;
stmt1.executeUpdate(sql);

cardinality = ca_cardinality;
for(int i=1;i<=cardinality;i++){
sql =
"INSERT INTO customer_address "
+ "(ca_address_id, ca_street_number, ca_street_name, ca_street_type, ca_suite_number,"
+ "ca_city, ca_county, ca_state, ca_zip, ca_country,"
+ "ca_gmt_offset, ca_location_type) "
+ "VALUES ("
+ rchar(16) + ","
+ rchar(10) + ","
+ rvchar(60) + ","
+ rchar(15) + ","
+ rchar(10) + ","
+ rvchar(60) + ","
+ rvchar(30) + ","
+ rchar(2) + ","
+ rchar(10) + ","
+ rvchar(20) + ","
+ rdec(5,2) + ","
+ rchar(20)
+ ")";

stmt1.executeUpdate(sql);
}
```

```
System.out.println("Table \'customer_address\' created");

stmt1.close();
conn1.close();

}
catch(SQLException se){
//Handle errors for JDBC
se.printStackTrace();
}
catch(Exception e){
//Handle errors for Class.forName
e.printStackTrace();
}
finally{
//finally block used to close resources
try{
if(stmt1!=null)
stmt1.close();
}
catch(SQLException se2){
}// nothing we can do
try{
if(conn1!=null)
conn1.close();
}
catch(SQLException se){
se.printStackTrace();
}
}

}


public void createTable_cd(){
Connection conn1 = null;
Statement stmt1 = null;
String sql;
int cardinality;


try{
Class.forName(JDBC_DRIVER);
conn1 = DriverManager.getConnection(DB_SERVER_URL + DBNAME1,USER,PASS);
stmt1 = conn1.createStatement();

sql = "DROP TABLE IF EXISTS " + "customer_demographics" ;
stmt1.executeUpdate(sql);

sql = cd_ctable ;
stmt1.executeUpdate(sql);

cardinality = cd_cardinality;
for(int i=1;i<=cardinality;i++){
sql =
"INSERT INTO customer_demographics "
+ "(cd_gender,cd_marital_status,cd_education_status,cd_purchase_estimate,cd_credit_rating,"
+ "cd_dep_count,cd_dep_employed_count,cd_dep_college_count) "
+ "VALUES ("
+ rvalue(cd_gender_vals) + ","
+ rvalue(cd_marital_status_vals) + ","
+ rvalue(cd_education_status_vals) + ","
+ rintrange(100,5000) + ","
```

```
+ rchar(10) + ","
+ rintrange(5,1000) + ","
+ rintrange(2,1000) + ","
+ rintrange(0,10)
+ ")";

stmt1.executeUpdate(sql);
}

System.out.println("Table \'customer_demographics\' created");

stmt1.close();
conn1.close();

}
catch(SQLException se){
//Handle errors for JDBC
se.printStackTrace();
}
catch(Exception e){
//Handle errors for Class.forName
e.printStackTrace();
}
finally{
//finally block used to close resources
try{
if(stmt1!=null)
stmt1.close();
}
catch(SQLException se2){
}// nothing we can do
try{
if(conn1!=null)
conn1.close();
}
catch(SQLException se){
se.printStackTrace();
}
}

}


public void createTable_d(){
Connection conn1 = null;
Statement stmt1 = null;
String sql;
int cardinality;


try{
Class.forName(JDBC_DRIVER);
conn1 = DriverManager.getConnection(DB_SERVER_URL + DBNAME1,USER,PASS);
stmt1 = conn1.createStatement();

sql = "DROP TABLE IF EXISTS " + "date_dim" ;
stmt1.executeUpdate(sql);

sql = d_ctable ;
stmt1.executeUpdate(sql);

cardinality = d_cardinality;
for(int i=1;i<=cardinality;i++){
```

```
sql =
"INSERT INTO date_dim "
+ "(d_date_id, d_date, d_month_seq, d_week_seq, d_quarter_seq,"
+ "d_year, d_dow, d_moy, d_dom, d_qoy,"
+ "d_fy_year, d_fy_quarter_seq, d_fy_week_seq, d_day_name, d_month_name, d_quarter_name,"
+ "d_holiday, d_weekend, d_following_holiday, d_first_dom, d_last_dom,"
+ "d_same_day_ly, d_same_day_lq, d_current_day, d_current_week, d_current_month,"
+ "d_current_quarter, d_current_year"
+ ") "
+ "VALUES ("
+ rchar(16) + ","
+ rdate(2000,2014) + ","
+ rintrange(1,12) + ","
+ rint(52) + ","
+ rint(4) + ","
+ rintrange(1954,2014) + ","
+ rint(7) + ","
+ rint(12) + ","
+ rint(30) + ","
+ rint(4) + ","
+ rintrange(1954,2014) + ","
+ rint(4) + ","
+ rint(52) + ","
+ rvalue(d_day_name_vals) + ","
+ rvalue(d_month_name_vals) + ","
+ rvalue(d_quarter_name_vals) + ","
+ rvalue(yes_no_vals) + ","
+ rvalue(yes_no_vals) + ","
+ rvalue(yes_no_vals) + ","
+ rint(10) + ","
+ rint(10) + ","
+ rint(10) + ","
+ rint(10) + ","
+ rvalue(yes_no_vals) + ","
+ rvalue(yes_no_vals) + ","
+ rvalue(yes_no_vals) + ","
+ rvalue(yes_no_vals) + ","
+ rvalue(yes_no_vals)
+ ")";

stmt1.executeUpdate(sql);
}

System.out.println("Table \'date_dim\' created");

stmt1.close();
conn1.close();

}
catch(SQLException se){
//Handle errors for JDBC
se.printStackTrace();
}
catch(Exception e){
//Handle errors for Class.forName
e.printStackTrace();
}
finally{
//finally block used to close resources
try{
if(stmt1!=null)
stmt1.close();
}
```

```
catch(SQLException se2){
}// nothing we can do
try{
if(conn1!=null)
conn1.close();
}
catch(SQLException se){
se.printStackTrace();
}
}

}


public void createTable_hd(){
Connection conn1 = null;
Statement stmt1 = null;
String sql;
int cardinality;


try{
Class.forName(JDBC_DRIVER);
conn1 = DriverManager.getConnection(DB_SERVER_URL + DBNAME1,USER,PASS);
stmt1 = conn1.createStatement();

sql = "DROP TABLE IF EXISTS " + "household_demographics" ;
stmt1.executeUpdate(sql);

sql = hd_ctable ;
stmt1.executeUpdate(sql);

cardinality = hd_cardinality;
for(int i=1;i<=cardinality;i++){
sql =
"INSERT INTO household_demographics "
+ "(hd_income_band_sk, hd_buy_potential, hd_dep_count, hd_vehicle_count) "
+ "VALUES ("
+ rint(ib_cardinality) + ","
+ rchar(15) + ","
+ rintrange(2,10) + ","
+ rintrange(0,10)
+ ")";

stmt1.executeUpdate(sql);
}

System.out.println("Table \'household_demographics\' created");

stmt1.close();
conn1.close();

}
catch(SQLException se){
//Handle errors for JDBC
se.printStackTrace();
}
catch(Exception e){
//Handle errors for Class.forName
e.printStackTrace();
}
finally{
//finally block used to close resources
```

```
try{
if(stmt1!=null)
stmt1.close();
}
catch(SQLException se2){
}// nothing we can do
try{
if(conn1!=null)
conn1.close();
}
catch(SQLException se){
se.printStackTrace();
}
}

}


public void createTable_ib(){
Connection conn1 = null;
Statement stmt1 = null;
String sql;
int cardinality;


try{
Class.forName(JDBC_DRIVER);
conn1 = DriverManager.getConnection(DB_SERVER_URL + DBNAME1,USER,PASS);
stmt1 = conn1.createStatement();

sql = "DROP TABLE IF EXISTS " + "income_band" ;
stmt1.executeUpdate(sql);

sql = ib_ctable;
stmt1.executeUpdate(sql);

cardinality = ib_cardinality;
for(int i=1;i<=cardinality;i++){
sql =
"INSERT INTO income_band "
+ "(ib_lower_bound , ib_upper_bound) "
+ "VALUES ("
+ rintrange(1000,10000) + ","
+ rintrange(15000,500000)
+ ")";

stmt1.executeUpdate(sql);
}

System.out.println("Table \'income_band\' created");

stmt1.close();
conn1.close();

}
catch(SQLException se){
//Handle errors for JDBC
se.printStackTrace();
}
catch(Exception e){
//Handle errors for Class.forName
e.printStackTrace();
}
```

```
finally{
//finally block used to close resources
try{
if(stmt1!=null)
stmt1.close();
}
catch(SQLException se2){
}// nothing we can do
try{
if(conn1!=null)
conn1.close();
}
catch(SQLException se){
se.printStackTrace();
}
}

}


public void createTable_inv(){
Connection conn1 = null;
Statement stmt1 = null;
String sql;
int cardinality;


try{
Class.forName(JDBC_DRIVER);
conn1 = DriverManager.getConnection(DB_SERVER_URL + DBNAME1,USER,PASS);
stmt1 = conn1.createStatement();

sql = "DROP TABLE IF EXISTS " + "inventory" ;
stmt1.executeUpdate(sql);

sql = inv_ctable ;
stmt1.executeUpdate(sql);

cardinality = inv_cardinality;
for(int i=1;i<=cardinality;i++){
sql =
"INSERT INTO inventory "
+ "(inv_date_sk, inv_item_sk, inv_warehouse_sk, inv_quantity_on_hand) "
+ "VALUES ("
+ rint(d_cardinality) + ","
+ rint(i_cardinality) + ","
+ rint(w_cardinality) + ","
+ rintrange(0,100000)
+ ")";

stmt1.executeUpdate(sql);
}

System.out.println("Table \'inventory\' created");

stmt1.close();
conn1.close();

}
catch(SQLException se){
//Handle errors for JDBC
se.printStackTrace();
}
```

```
catch(Exception e){
//Handle errors for Class.forName
e.printStackTrace();
}
finally{
//finally block used to close resources
try{
if(stmt1!=null)
stmt1.close();
}
catch(SQLException se2){
}// nothing we can do
try{
if(conn1!=null)
conn1.close();
}
catch(SQLException se){
se.printStackTrace();
}
}

}


public void createTable_i(){
Connection conn1 = null;
Statement stmt1 = null;
String sql;
int cardinality;


try{
Class.forName(JDBC_DRIVER);
conn1 = DriverManager.getConnection(DB_SERVER_URL + DBNAME1,USER,PASS);
stmt1 = conn1.createStatement();

sql = "DROP TABLE IF EXISTS " + "item" ;
stmt1.executeUpdate(sql);

sql = i_ctable ;
stmt1.executeUpdate(sql);

cardinality = i_cardinality;
for(int i=1;i<=cardinality;i++){
sql =
"INSERT INTO ITEM "
+ "( i_item_id, i_rec_start_date, i_rec_end_date, i_item_desc,"
+ " i_current_price, i_wholesale_cost,"
+ " i_brand_id, i_brand, i_class_id,"
+ " i_class, i_category_id, i_category, i_manufact_id, i_manufact, i_size,"
+ " i_formulation, i_color, i_units, i_container , i_manager_id, i_product_name) "
+ "VALUES ("
+ rchar(16) + ","
+ rdate(2000,2005) + ","
+ rdate(2006,2014) + ","
+ rvchar(200) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rint(10000) + ","
+ rchar(50) + ","
+ rint(10000) + ","
+ rchar(50) + ","
+ rint(10000) + ","
```

```
+ rchar(50) + ","
+ rint(10000) + ","
+ rchar(50) + ","
+ rchar(20) + ","
+ rchar(20) + ","
+ rchar(20) + ","
+ rchar(10) + ","
+ rchar(10) + ","
+ rint(5000) + ","
+ rchar(50)
+ ")";

stmt1.executeUpdate(sql);
}

System.out.println("Table \'item\' created");

stmt1.close();
conn1.close();

}
catch(SQLException se){
//Handle errors for JDBC
se.printStackTrace();
}
catch(Exception e){
//Handle errors for Class.forName
e.printStackTrace();
}
finally{
//finally block used to close resources
try{
if(stmt1!=null)
stmt1.close();
}
catch(SQLException se2){
}// nothing we can do
try{
if(conn1!=null)
conn1.close();
}
catch(SQLException se){
se.printStackTrace();
}
}

}


public void createTable_p(){
Connection conn1 = null;
Statement stmt1 = null;
String sql;
int cardinality;


try{
Class.forName(JDBC_DRIVER);
conn1 = DriverManager.getConnection(DB_SERVER_URL + DBNAME1,USER,PASS);
stmt1 = conn1.createStatement();

sql = "DROP TABLE IF EXISTS " + "promotion" ;
stmt1.executeUpdate(sql);
```

```
sql = p_ctable ;
stmt1.executeUpdate(sql);

cardinality = p_cardinality;
for(int i=1;i<=cardinality;i++){
sql =
"INSERT INTO promotion "
+ "(p_promo_id, p_start_date_sk, p_end_date_sk, p_item_sk, p_cost,"
+ "p_response_target, p_promo_name, p_channel_dmail, p_channel_email, p_channel_catalog,"
+ "p_channel_tv, p_channel_radio, p_channel_press, p_channel_event, p_channel_demo,"
+ "p_channel_details, p_purpose, p_discount_active) "
+ "VALUES ("
+ rchar(16) + ","
+ rint(d_cardinality) + ","
+ rint(d_cardinality) + ","
+ rint(i_cardinality) + ","
+ rdec(15,2) + ","
+ rintrange(1000,50000) + ","
+ rchar(50) + ","
+ rvalue(yes_no_vals) + ","
+ rvalue(yes_no_vals) + ","
+ rvalue(yes_no_vals) + ","
+ rvalue(yes_no_vals) + ","
+ rvalue(yes_no_vals) + ","
+ rvalue(yes_no_vals) + ","
+ rvalue(yes_no_vals) + ","
+ rvalue(yes_no_vals) + ","
+ rvchar(100) + ","
+ rchar(15) + ","
+ rvalue(yes_no_vals)
+ ")";

stmt1.executeUpdate(sql);
}

System.out.println("Table \'promotion\' created");

stmt1.close();
conn1.close();

}
catch(SQLException se){
//Handle errors for JDBC
se.printStackTrace();
}
catch(Exception e){
//Handle errors for Class.forName
e.printStackTrace();
}
finally{
//finally block used to close resources
try{
if(stmt1!=null)
stmt1.close();
}
catch(SQLException se2){
}// nothing we can do
try{
if(conn1!=null)
conn1.close();
}
catch(SQLException se){
```

```
se.printStackTrace();
}
}

}


public void createTable_r(){
Connection conn1 = null;
Statement stmt1 = null;
String sql;
int cardinality;


try{
Class.forName(JDBC_DRIVER);
conn1 = DriverManager.getConnection(DB_SERVER_URL + DBNAME1,USER,PASS);
stmt1 = conn1.createStatement();

sql = "DROP TABLE IF EXISTS " + "reason" ;
stmt1.executeUpdate(sql);

sql = r_ctable ;
stmt1.executeUpdate(sql);

cardinality = r_cardinality;
for(int i=1;i<=cardinality;i++){
sql =
"INSERT INTO reason "
+ "(r_reason_id ,r_reason_desc) "
+ "VALUES ("
+ rchar(16) + ","
+ rchar(100)
+ ")";

stmt1.executeUpdate(sql);
}

System.out.println("Table \'reason\' created");

stmt1.close();
conn1.close();

}
catch(SQLException se){
//Handle errors for JDBC
se.printStackTrace();
}
catch(Exception e){
//Handle errors for Class.forName
e.printStackTrace();
}
finally{
//finally block used to close resources
try{
if(stmt1!=null)
stmt1.close();
}
catch(SQLException se2){
}// nothing we can do
try{
if(conn1!=null)
conn1.close();
```

```
}
catch(SQLException se){
se.printStackTrace();
}
}

}


public void createTable_sm(){
Connection conn1 = null;
Statement stmt1 = null;
String sql;
int cardinality;


try{
Class.forName(JDBC_DRIVER);
conn1 = DriverManager.getConnection(DB_SERVER_URL + DBNAME1,USER,PASS);
stmt1 = conn1.createStatement();

sql = "DROP TABLE IF EXISTS " + "ship_mode" ;
stmt1.executeUpdate(sql);

sql = sm_ctable ;
stmt1.executeUpdate(sql);

cardinality = sm_cardinality;
for(int i=1;i<=cardinality;i++){
sql =
"INSERT INTO ship_mode "
+ "(sm_ship_mode_id,sm_type,sm_code,sm_carrier,sm_contract) "
+ "VALUES ("
+ rchar(16) + ","
+ rchar(30) + ","
+ rchar(10) + ","
+ rchar(20) + ","
+ rchar(20)
+ ")";

stmt1.executeUpdate(sql);
}

System.out.println("Table \'ship_mode\' created");

stmt1.close();
conn1.close();

}
catch(SQLException se){
//Handle errors for JDBC
se.printStackTrace();
}
catch(Exception e){
//Handle errors for Class.forName
e.printStackTrace();
}
finally{
//finally block used to close resources
try{
if(stmt1!=null)
stmt1.close();
}
```

```
catch(SQLException se2){
}// nothing we can do
try{
if(conn1!=null)
conn1.close();
}
catch(SQLException se){
se.printStackTrace();
}
}

}


public void createTable_s(){
Connection conn1 = null;
Statement stmt1 = null;
String sql;
int cardinality;


try{
Class.forName(JDBC_DRIVER);
conn1 = DriverManager.getConnection(DB_SERVER_URL + DBNAME1,USER,PASS);
stmt1 = conn1.createStatement();

sql = "DROP TABLE IF EXISTS " + "store" ;
stmt1.executeUpdate(sql);

sql = s_ctable ;
stmt1.executeUpdate(sql);

cardinality = s_cardinality;
for(int i=1;i<=cardinality;i++){
sql =
"INSERT INTO store "
+ "(s_store_id, s_rec_start_date, s_rec_end_date, s_closed_date_sk, s_store_name,"
+ " s_number_employees, s_floor_space, s_hours, s_manager, s_market_id,"
+ " s_geography_class, s_market_desc, s_market_manager, s_division_id, "
+ " s_division_name, s_company_id, s_company_name, s_street_number, s_street_name,"
+ " s_street_type, s_suite_number,s_city,"
+ " s_county,s_state,s_zip,s_country,s_gmt_offset,s_tax_percentage) "
+ "VALUES ("
+ rchar(16) + ","
+ rdate(2000,2005) + ","
+ rdate(2006,2014) + ","
+ rint(d_cardinality) + ","
+ rvchar(50) + ","
+ rintrange(10,500) + ","
+ rintrange(250,1500) + ","
+ rchar(20) + ","
+ rvchar(40) + ","
+ rint(1000) + ","
+ rvchar(100) + ","
+ rvchar(100) + ","
+ rvchar(40) + ","
+ rint(1000) + ","
+ rvchar(50) + ","
+ rint(1000) + ","
+ rvchar(50) + ","
+ rvchar(10) + ","
+ rvchar(60) + ","
+ rchar(15) + ","
```

```
+ rchar(10) + ","
+ rvchar(60)+ ","
+ rvchar(30)+ ","
+ rchar(2) + ","
+ rchar(10) + ","
+ rvchar(20) + ","
+ rdec(5,2) + ","
+ rdec(5,2)
+ ")";

stmt1.executeUpdate(sql);
}

System.out.println("Table \'store\' created");

stmt1.close();
conn1.close();

}
catch(SQLException se){
//Handle errors for JDBC
se.printStackTrace();
}
catch(Exception e){
//Handle errors for Class.forName
e.printStackTrace();
}
finally{
//finally block used to close resources
try{
if(stmt1!=null)
stmt1.close();
}
catch(SQLException se2){
}// nothing we can do
try{
if(conn1!=null)
conn1.close();
}
catch(SQLException se){
se.printStackTrace();
}
}

}


public void createTable_ss_sr(){
Connection conn1 = null;
Statement stmt1 = null;
String sql;
int rand_item_sk,rand_ticket_num;


try{
Class.forName(JDBC_DRIVER);
conn1 = DriverManager.getConnection(DB_SERVER_URL + DBNAME1,USER,PASS);
stmt1 = conn1.createStatement();

sql = "DROP TABLE IF EXISTS " + "store_returns" ;
stmt1.executeUpdate(sql);

sql = "DROP TABLE IF EXISTS " + "store_sales" ;
```

```
stmt1.executeUpdate(sql);

sql = ss_ctable ;
stmt1.executeUpdate(sql);

sql = sr_ctable ;
stmt1.executeUpdate(sql);

for(int i=1;i<=ss_cardinality;i++){
rand_item_sk = rint(i_cardinality);
rand_ticket_num = rint(10000);
while(used(rand_item_sk,rand_ticket_num,"store_sales","ss_sold_date_sk","ss_item_sk","ss_ticket_n
umber")){
rand_item_sk = rint(i_cardinality);
rand_ticket_num = rint(10000);
}
sql =
"INSERT INTO store_sales "
+ "(ss_sold_date_sk, ss_sold_time_sk, ss_item_sk, ss_customer_sk, ss_cdemo_sk,"
+ " ss_hdemo_sk, ss_addr_sk, ss_store_sk, ss_promo_sk, ss_ticket_number,"
+ " ss_quantity, ss_wholesale_cost, ss_list_price, ss_sales_price,"
+ " ss_ext_discount_amt,ss_ext_sales_price,ss_ext_wholesale_cost,ss_ext_list_price,"
+ " ss_ext_tax, ss_coupon_amt, ss_net_paid, ss_net_paid_inc_tax, ss_net_profit) "
+ "VALUES ("
+ rint(d_cardinality) + ","
+ rint(t_cardinality) + ","
+ rand_item_sk + ","
+ rint(c_cardinality) + ","
+ rint(cd_cardinality) + ","
+ rint(hd_cardinality) + ","
+ rint(ca_cardinality) + ","
+ rint(s_cardinality) + ","
+ rint(p_cardinality) + ","
+ rand_ticket_num + ","
+ rintrange(20,200) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2)
+ ")";

stmt1.executeUpdate(sql);

if(i<=sr_cardinality){

sql =
"INSERT INTO store_returns "
+ "(sr_returned_date_sk, sr_returned_time_sk, sr_item_sk, sr_customer_sk,"
+ " sr_cdemo_sk, sr_hdemo_sk, sr_addr_sk, sr_store_sk, sr_reason_sk,"
+ " sr_ticket_number, sr_return_quantity, sr_return_amt, sr_return_tax,"
+ " sr_return_amt_inc_tax, sr_fee, sr_return_ship_cost, sr_refunded_cash,"
+ " sr_reversed_charge, sr_store_credit, sr_net_loss) "
+ "VALUES ("
+ rint(d_cardinality) + ","
+ rint(t_cardinality) + ","
```

```
+ rand_item_sk + ","
+ rint(c_cardinality) + ","
+ rint(cd_cardinality) + ","
+ rint(hd_cardinality) + ","
+ rint(ca_cardinality) + ","
+ rint(s_cardinality) + ","
+ rint(r_cardinality) + ","
+ rand_ticket_num + ","
+ rintrange(20,200) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2)
+ ")";

stmt1.executeUpdate(sql);
}

}

System.out.println("Table \'store_sales\' created");
System.out.println("Table \'store_returns\' created");

stmt1.close();
conn1.close();

}
catch(SQLException se){
//Handle errors for JDBC
se.printStackTrace();
}
catch(Exception e){
//Handle errors for Class.forName
e.printStackTrace();
}
finally{
//finally block used to close resources
try{
if(stmt1!=null)
stmt1.close();
}
catch(SQLException se2){
}// nothing we can do
try{
if(conn1!=null)
conn1.close();
}
catch(SQLException se){
se.printStackTrace();
}
}

}

public void createTable_t(){
Connection conn1 = null;
Statement stmt1 = null;
String sql;
```

```
int cardinality;


try{
Class.forName(JDBC_DRIVER);
conn1 = DriverManager.getConnection(DB_SERVER_URL + DBNAME1,USER,PASS);
stmt1 = conn1.createStatement();

sql = "DROP TABLE IF EXISTS " + "time_dim" ;
stmt1.executeUpdate(sql);

sql = t_ctable ;
stmt1.executeUpdate(sql);

cardinality = t_cardinality;
for(int i=1;i<=cardinality;i++){
sql =
"INSERT INTO time_dim "
+ "(t_time_id,t_time ,t_hour ,t_minute ,"
+ "t_second ,t_am_pm ,t_shift ,t_sub_shift ,t_meal_time) "
+ "VALUES ("
+ rchar(16) + ","
+ rintrange(0,86400)+ ","
+ rintrange(0,23)+ ","
+ rintrange(0,59)+ ","
+ rintrange(0,59)+ ","
+ rvalue(t_am_pm_vals) + ","
+ rchar(20) + ","
+ rchar(20) + ","
+ rchar(20)
+ ")";

stmt1.executeUpdate(sql);
}

System.out.println("Table \'time_dim\' created");

stmt1.close();
conn1.close();

}
catch(SQLException se){
//Handle errors for JDBC
se.printStackTrace();
}
catch(Exception e){
//Handle errors for Class.forName
e.printStackTrace();
}
finally{
//finally block used to close resources
try{
if(stmt1!=null)
stmt1.close();
}
catch(SQLException se2){
}// nothing we can do
try{
if(conn1!=null)
conn1.close();
}
catch(SQLException se){
se.printStackTrace();
```

```
}
}

}

public void createTable_w(){
Connection conn1 = null;
Statement stmt1 = null;
String sql;
int cardinality;


try{
Class.forName(JDBC_DRIVER);
conn1 = DriverManager.getConnection(DB_SERVER_URL + DBNAME1,USER,PASS);
stmt1 = conn1.createStatement();

sql = "DROP TABLE IF EXISTS " + "warehouse" ;
stmt1.executeUpdate(sql);

sql = w_ctable ;
stmt1.executeUpdate(sql);

cardinality = w_cardinality;
for(int i=1;i<=cardinality;i++){
sql =
"INSERT INTO warehouse "
+ "(w_warehouse_id, w_warehouse_name, w_warehouse_sq_ft, w_street_number, w_street_name,"
+ "w_street_type, w_suite_number, w_city, w_county, w_state,"
+ "w_zip, w_country, w_gmt_offset) "
+ "VALUES ("
+ rchar(16) + ","
+ rvchar(20) + ","
+ rintrange(500,4000) + ","
+ rchar(10) + ","
+ rvchar(60) + ","
+ rchar(15) + ","
+ rchar(10) + ","
+ rvchar(60) + ","
+ rvchar(30) + ","
+ rchar(2) + ","
+ rchar(10) + ","
+ rvchar(20) + ","
+ rdec(5,2)
+ ")";

stmt1.executeUpdate(sql);
}

System.out.println("Table \'warehouse\' created");

stmt1.close();
conn1.close();

}
catch(SQLException se){
//Handle errors for JDBC
se.printStackTrace();
}
catch(Exception e){
//Handle errors for Class.forName
e.printStackTrace();
}
```

```
finally{
//finally block used to close resources
try{
if(stmt1!=null)
stmt1.close();
}
catch(SQLException se2){
}// nothing we can do
try{
if(conn1!=null)
conn1.close();
}
catch(SQLException se){
se.printStackTrace();
}
}

}

public void createTable_wp(){
Connection conn1 = null;
Statement stmt1 = null;
String sql;
int cardinality;


try{
Class.forName(JDBC_DRIVER);
conn1 = DriverManager.getConnection(DB_SERVER_URL + DBNAME1,USER,PASS);
stmt1 = conn1.createStatement();

sql = "DROP TABLE IF EXISTS " + "web_page" ;
stmt1.executeUpdate(sql);

sql = wp_ctable ;
stmt1.executeUpdate(sql);

cardinality = wp_cardinality;
for(int i=1;i<=cardinality;i++){
sql =
"INSERT INTO web_page "
+ "(wp_web_page_id, wp_rec_start_date, wp_rec_end_date, wp_creation_date_sk,"
+ "wp_access_date_sk, wp_autogen_flag, wp_customer_sk, wp_url,"
+ "wp_type, wp_char_count, wp_link_count, wp_image_count, wp_max_ad_count) "
+ "VALUES ("
+ rchar(16) + ","
+ rdate(2000,2005) + ","
+ rdate(2006,2014) + ","
+ rint(d_cardinality) + ","
+ rint(d_cardinality) + ","
+ rvalue(yes_no_vals) + ","
+ rint(c_cardinality) + ","
+ rvchar(100) + ","
+ rchar(50) + ","
+ rintrange(1000,1000000) + ","
+ rintrange(10,200) + ","
+ rintrange(10,50) + ","
+ rintrange(5,15)
+ ")";

stmt1.executeUpdate(sql);
}
```

```
System.out.println("Table \'web_page\' created");

stmt1.close();
conn1.close();

}
catch(SQLException se){
//Handle errors for JDBC
se.printStackTrace();
}
catch(Exception e){
//Handle errors for Class.forName
e.printStackTrace();
}
finally{
//finally block used to close resources
try{
if(stmt1!=null)
stmt1.close();
}
catch(SQLException se2){
}// nothing we can do
try{
if(conn1!=null)
conn1.close();
}
catch(SQLException se){
se.printStackTrace();
}
}

}

public void createTable_ws_wr(){
Connection conn1 = null;
Statement stmt1 = null;
String sql;
int rand_item_sk,rand_order_num;


try{
Class.forName(JDBC_DRIVER);
conn1 = DriverManager.getConnection(DB_SERVER_URL + DBNAME1,USER,PASS);
stmt1 = conn1.createStatement();

sql = "DROP TABLE IF EXISTS " + "web_returns" ;
stmt1.executeUpdate(sql);

sql = "DROP TABLE IF EXISTS " + "web_sales" ;
stmt1.executeUpdate(sql);

sql = ws_ctable ;
stmt1.executeUpdate(sql);

sql = wr_ctable ;
stmt1.executeUpdate(sql);

for(int i=1;i<=ws_cardinality;i++){
rand_item_sk = rint(i_cardinality);
rand_order_num = rint(10000);
while(used(rand_item_sk,rand_order_num,"web_sales","ws_sold_date_sk","ws_item_sk","ws_order_numbe
r")){
rand_item_sk = rint(i_cardinality);
```

```
rand_order_num = rint(10000);
}
sql =
"INSERT INTO web_sales "
+ "(ws_sold_date_sk, ws_sold_time_sk, ws_ship_date_sk, ws_item_sk,"
+ " ws_bill_customer_sk, ws_bill_cdemo_sk, ws_bill_hdemo_sk, ws_bill_addr_sk,"
+ " ws_ship_customer_sk, ws_ship_cdemo_sk, ws_ship_hdemo_sk, ws_ship_addr_sk,"
+ " ws_web_page_sk, ws_web_site_sk, ws_ship_mode_sk, ws_warehouse_sk,"
+ " ws_promo_sk, ws_order_number, ws_quantity, ws_wholesale_cost,"
+ " ws_list_price, ws_sales_price, ws_ext_discount_amt, ws_ext_sales_price,"
+ " ws_ext_wholesale_cost, ws_ext_list_price, ws_ext_tax, ws_coupon_amt,"
+ " ws_ext_ship_cost, ws_net_paid, ws_net_paid_inc_tax, ws_net_paid_inc_ship,"
+ " ws_net_paid_inc_ship_tax, ws_net_profit) "
+ "VALUES ("
+ rint(d_cardinality) + ","
+ rint(t_cardinality) + ","
+ rint(d_cardinality) + ","
+ rand_item_sk + ","
+ rint(c_cardinality) + ","
+ rint(cd_cardinality) + ","
+ rint(hd_cardinality) + ","
+ rint(ca_cardinality) + ","
+ rint(c_cardinality) + ","
+ rint(cd_cardinality) + ","
+ rint(hd_cardinality) + ","
+ rint(ca_cardinality) + ","
+ rint(wp_cardinality) + ","
+ rint(web_cardinality) + ","
+ rint(sm_cardinality) + ","
+ rint(w_cardinality) + ","
+ rint(p_cardinality) + ","
+ rand_order_num + ","
+ rintrange(20,200) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2)
+ ")";

stmt1.executeUpdate(sql);

if(i<=wr_cardinality){

sql =
"INSERT INTO web_returns "
+ "(wr_returned_date_sk, wr_returned_time_sk, wr_item_sk, wr_refunded_customer_sk,"
+ " wr_refunded_cdemo_sk, wr_refunded_hdemo_sk, wr_refunded_addr_sk, "
+ " wr_returning_customer_sk, wr_returning_cdemo_sk, wr_returning_hdemo_sk,"
+ " wr_returning_addr_sk, wr_web_page_sk, wr_reason_sk, wr_order_number,"
+ " wr_return_quantity, wr_return_amt, wr_return_tax, wr_return_amt_inc_tax,"
+ " wr_fee,wr_return_ship_cost, wr_refunded_cash, wr_reversed_charge,"
+ " wr_account_credit, wr_net_loss) "
```

```
+ "VALUES ("
+ rint(d_cardinality) + ","
+ rint(t_cardinality) + ","
+ rand_item_sk + ","
+ rint(c_cardinality) + ","
+ rint(cd_cardinality) + ","
+ rint(hd_cardinality) + ","
+ rint(ca_cardinality) + ","
+ rint(c_cardinality) + ","
+ rint(cd_cardinality) + ","
+ rint(hd_cardinality) + ","
+ rint(ca_cardinality) + ","
+ rint(wp_cardinality) + ","
+ rint(r_cardinality) + ","
+ rand_order_num + ","
+ rintrange(20,200) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2) + ","
+ rdec(7,2)
+ ")";

stmt1.executeUpdate(sql);
}
}

System.out.println("Table \'web_sales\' created");
System.out.println("Table \'web_returns\' created");

stmt1.close();
conn1.close();

}
catch(SQLException se){
//Handle errors for JDBC
se.printStackTrace();
}
catch(Exception e){
//Handle errors for Class.forName
e.printStackTrace();
}
finally{
//finally block used to close resources
try{
if(stmt1!=null)
stmt1.close();
}
catch(SQLException se2){
}// nothing we can do
try{
if(conn1!=null)
conn1.close();
}
catch(SQLException se){
se.printStackTrace();
}
}
```

```
}

public void createTable_web(){
Connection conn1 = null;
Statement stmt1 = null;
String sql;
int cardinality;


try{
Class.forName(JDBC_DRIVER);
conn1 = DriverManager.getConnection(DB_SERVER_URL + DBNAME1,USER,PASS);
stmt1 = conn1.createStatement();

sql = "DROP TABLE IF EXISTS " + "web_site" ;
stmt1.executeUpdate(sql);

sql = web_ctable ;
stmt1.executeUpdate(sql);

cardinality = web_cardinality;
for(int i=1;i<=cardinality;i++){
sql =
"INSERT INTO web_site "
+ "(web_site_id, web_rec_start_date, web_rec_end_date, web_name,"
+ "web_open_date_sk, web_close_date_sk, web_class, web_manager,"
+ "web_mkt_id, web_mkt_class, web_mkt_desc, web_market_manager,"
+ "web_company_id, web_company_name, web_street_number, web_street_name,"
+ "web_street_type, web_suite_number, web_city, web_county,"
+ "web_state, web_zip, web_country, web_gmt_offset,web_tax_percentage) "
+ "VALUES ("
+ rchar(16) + ","
+ rdate(2000,2005) + ","
+ rdate(2006,2014) + ","
+ rvchar(50) + ","
+ rint(d_cardinality) + ","
+ rint(d_cardinality) + ","
+ rvchar(50) + ","
+ rvchar(40) + ","
+ rint(10000) + ","
+ rvchar(50) + ","
+ rvchar(100) + ","
+ rvchar(40) + ","
+ rint(10000) + ","
+ rchar(50) + ","
+ rchar(10) + ","
+ rvchar(60) + ","
+ rchar(15) + ","
+ rchar(10) + ","
+ rvchar(60) + ","
+ rvchar(30) + ","
+ rchar(2) + ","
+ rchar(10) + ","
+ rvchar(20) + ","
+ rdec(5,2) + ","
+ rdec(5,2)
+ ")";

stmt1.executeUpdate(sql);
}

System.out.println("Table \'web_site\' created");
```

```
stmt1.close();
conn1.close();

}
catch(SQLException se){
//Handle errors for JDBC
se.printStackTrace();
}
catch(Exception e){
//Handle errors for Class.forName
e.printStackTrace();
}
finally{
//finally block used to close resources
try{
if(stmt1!=null)
stmt1.close();
}
catch(SQLException se2){
}// nothing we can do
try{
if(conn1!=null)
conn1.close();
}
catch(SQLException se){
se.printStackTrace();
}
}

}


// Method to update the fact tables and rerun queries

public void updateFacts(double percent){

//This is how many rows will be replaced from ss,cs,ws
int nss = (int) (percent*ss_cardinality);
int ncs = (int) (percent*cs_cardinality);
int nws = (int) (percent*ws_cardinality);

Connection conn1 = null,conn2 = null;
Statement  stmt1 = null,stmt2 = null;
try{

conn1 = DriverManager.getConnection(DB_SERVER_URL+DBNAME1, USER, PASS);
stmt1 = conn1.createStatement();

conn2 = DriverManager.getConnection(DB_SERVER_URL+DBNAME2, USER, PASS);
stmt2 = conn2.createStatement();

String sql;
ResultSet rs1;

int rand_item_sk,rand_ticket_num,rand_order_num ;

//First, delete n rows from ss,cs,ws and the respective rows from sr,cr,wr (on delete cascade)

sql = "DELETE FROM store_sales LIMIT " + nss;
stmt1.executeUpdate(sql);
stmt2.executeUpdate(sql);

sql = "DELETE FROM catalog_sales LIMIT " + ncs;
```

```
stmt1.executeUpdate(sql);
stmt2.executeUpdate(sql);

sql = "DELETE FROM web_sales LIMIT " + nws;
stmt1.executeUpdate(sql);
stmt2.executeUpdate(sql);

//I need to know the current cardinalities of sr,cr,wr
//cause deleting a row in s doesn't necessarily mean deleting one from r !
//In each table I will add 'original cardinality' - 'current cardinality' rows

int xs,xc,xw;

sql = "SELECT COUNT(sr_item_sk) FROM store_returns";
rs1 = stmt1.executeQuery(sql);
rs1.next();
xs = sr_cardinality - rs1.getInt(1);


sql = "SELECT COUNT(cr_item_sk) FROM catalog_returns";
rs1 = stmt1.executeQuery(sql);
rs1.next();
xc = cr_cardinality - rs1.getInt(1);

sql = "SELECT COUNT(wr_item_sk) FROM web_returns";
rs1 = stmt1.executeQuery(sql);
rs1.next();
xw = wr_cardinality - rs1.getInt(1);

//ss and sr
for(int i=0;i<nss;i++){
//ss and sr
rand_item_sk = rint(i_cardinality);
rand_ticket_num = rint(10000);
while(used(rand_item_sk,rand_ticket_num,"store_sales","ss_sold_date_sk","ss_item_sk","ss_ticket_n
umber")){
rand_item_sk = rint(i_cardinality);
rand_ticket_num = rint(10000);
}
int d = rint(d_cardinality);
int t = rint(t_cardinality);
int c = rint(c_cardinality);
int cd = rint(cd_cardinality);
int hd = rint(hd_cardinality);
int ca = rint(ca_cardinality);
int s = rint(s_cardinality);
int p = rint(p_cardinality);
int ranged = rintrange(20,200);
double dec1 = rdec(7,2);
double dec2 = rdec(7,2);
double dec3 = rdec(7,2);
double dec4 = rdec(7,2);
double dec5 = rdec(7,2);
double dec6 = rdec(7,2);
double dec7 = rdec(7,2);
double dec8 = rdec(7,2);
double dec9 = rdec(7,2);
double dec10 = rdec(7,2);
double dec11 = rdec(7,2);
double dec12 = rdec(7,2);
sql =
"INSERT INTO store_sales "
+ "(ss_sold_date_sk, ss_sold_time_sk, ss_item_sk, ss_customer_sk, ss_cdemo_sk,"
```

```
+ " ss_hdemo_sk, ss_addr_sk, ss_store_sk, ss_promo_sk, ss_ticket_number,"
+ " ss_quantity, ss_wholesale_cost, ss_list_price, ss_sales_price,"
+ " ss_ext_discount_amt,ss_ext_sales_price,ss_ext_wholesale_cost,ss_ext_list_price,"
+ " ss_ext_tax, ss_coupon_amt, ss_net_paid, ss_net_paid_inc_tax, ss_net_profit) "
+ "VALUES ("
+ d + ","+ t + ","+ rand_item_sk + ","+ c + ","+ cd + ","
+ hd + ","+ ca + ","+ s + ","+ p + ","+ rand_ticket_num + ","+ ranged + ","
+ dec1 + ","+ dec2 + ","+ dec3 + ","+ dec4 + ","+ dec5 + ","+ dec6 + ","
+ dec7 + ","+ dec8 + ","+ dec9 + ","+ dec10 + ","+ dec11 + ","+ dec12 + ")";

stmt1.executeUpdate(sql);
stmt2.executeUpdate(sql);

if(i<xs){

int dr = rint(d_cardinality);
int tr= rint(t_cardinality);
int cr = rint(c_cardinality);
int cdr = rint(cd_cardinality);
int hdr = rint(hd_cardinality);
int car = rint(ca_cardinality);
int sr = rint(s_cardinality);
int rr = rint(r_cardinality);
int rangedr = rintrange(20,200);
double dec1r = rdec(7,2);
double dec2r = rdec(7,2);
double dec3r = rdec(7,2);
double dec4r = rdec(7,2);
double dec5r = rdec(7,2);
double dec6r = rdec(7,2);
double dec7r = rdec(7,2);
double dec8r = rdec(7,2);
double dec9r = rdec(7,2);

sql =
"INSERT INTO store_returns "
+ "(sr_returned_date_sk, sr_returned_time_sk, sr_item_sk, sr_customer_sk,"
+ " sr_cdemo_sk, sr_hdemo_sk, sr_addr_sk, sr_store_sk, sr_reason_sk,"
+ " sr_ticket_number, sr_return_quantity, sr_return_amt, sr_return_tax,"
+ " sr_return_amt_inc_tax, sr_fee, sr_return_ship_cost, sr_refunded_cash,"
+ " sr_reversed_charge, sr_store_credit, sr_net_loss) "
+ "VALUES ("
+ dr + ","+ tr + ","+ rand_item_sk + ","+ cr + ","+ cdr + ","
+ hdr + ","+ car + ","+ sr + ","+ rr + ","+ rand_ticket_num + ","+ rangedr + ","
+ dec1r + ","+ dec2r + ","+ dec3r + ","+ dec4r + ","+ dec5r + ","
+ dec6r + ","+ dec7r + ","+ dec8r + ","+ dec9r+ ")";

stmt1.executeUpdate(sql);
stmt2.executeUpdate(sql);
}
}

System.out.println(">>>>>>>>>>>>>>store_sales-returns UPDATED by " + nss);

//cs and cr
for(int i=0;i<ncs;i++){
rand_item_sk = rint(i_cardinality);
rand_order_num = rint(10000);

while(used(rand_item_sk,rand_order_num,"catalog_sales","cs_sold_date_sk","cs_item_sk","cs_order_n
umber"))
{
rand_item_sk = rint(i_cardinality);
```

```
rand_order_num = rint(10000);
}

int d1 = rint(d_cardinality) ;
int t = rint(t_cardinality) ;
int d2 = rint(d_cardinality);
int c1 = rint(c_cardinality);
int cd1 = rint(cd_cardinality);
int hd1 = rint(hd_cardinality);
int ca1 = rint(ca_cardinality) ;
int c2 = rint(c_cardinality);
int cd2 = rint(cd_cardinality);
int hd2 = rint(hd_cardinality);
int ca2 = rint(ca_cardinality) ;
int cc =  rint(cc_cardinality);
int cp = rint(cp_cardinality);
int sm = rint(sm_cardinality);
int w = rint(w_cardinality);
int p = rint(p_cardinality);
int ranged = rintrange(20,2000);
double dec1 = rdec(7,2);
double dec2 = rdec(7,2);
double dec3 = rdec(7,2);
double dec4 = rdec(7,2);
double dec5 = rdec(7,2);
double dec6 = rdec(7,2);
double dec7 = rdec(7,2);
double dec8 = rdec(7,2);
double dec9 = rdec(7,2);
double dec10 = rdec(7,2);
double dec11 = rdec(7,2);
double dec12 = rdec(7,2);
double dec13 = rdec(7,2);
double dec14 = rdec(7,2);
double dec15 = rdec(7,2);

sql =
"INSERT INTO catalog_sales "
+ "(cs_sold_date_sk, cs_sold_time_sk, cs_ship_date_sk, cs_bill_customer_sk,"
+ " cs_bill_cdemo_sk, cs_bill_hdemo_sk, cs_bill_addr_sk, cs_ship_customer_sk,"
+ " cs_ship_cdemo_sk, cs_ship_hdemo_sk, cs_ship_addr_sk, cs_call_center_sk,"
+ " cs_catalog_page_sk, cs_ship_mode_sk, cs_warehouse_sk, cs_item_sk,"
+ " cs_promo_sk, cs_order_number, cs_quantity, cs_wholesale_cost, cs_list_price,"
+ " cs_sales_price, cs_ext_discount_amt, cs_ext_sales_price, cs_ext_wholesale_cost,"
+ " cs_ext_list_price, cs_ext_tax, cs_coupon_amt, cs_ext_ship_cost, cs_net_paid,"
+ " cs_net_paid_inc_tax, cs_net_paid_inc_ship, cs_net_paid_inc_ship_tax, "
+ " cs_net_profit)"
+ "VALUES ("
+ d1 + ","+ t + ","+ d2 + ","+ c1 + ","+ cd1 + ","+ hd1 + ","+ ca1 + ","
+ c2 + ","+ cd2 + ","+ hd2 + ","+ ca2 + ","+ cc + ","+ cp + ","
+ sm + ","+ w + ","+ rand_item_sk + ","+ p + ","+ rand_order_num + ","+ ranged + ","
+ dec1 + ","+ dec2 + ","+ dec3 + ","+ dec4 + ","+ dec5 + ","
+ dec6 + ","+ dec7 + ","+ dec8 + ","+ dec9 + ","+ dec10 + ","
+ dec11 + ","+ dec12 + ","+ dec13 + ","+ dec14 + ","+ dec15+ ")";

stmt1.executeUpdate(sql);
stmt2.executeUpdate(sql);

if(i<xc){

int d1r = rint(d_cardinality) ;
int tr = rint(t_cardinality) ;
int c1r = rint(c_cardinality);
```

```
int cd1r = rint(cd_cardinality);
int hd1r = rint(hd_cardinality);
int ca1r = rint(ca_cardinality) ;
int c2r = rint(c_cardinality);
int cd2r = rint(cd_cardinality);
int hd2r = rint(hd_cardinality);
int ca2r = rint(ca_cardinality) ;
int ccr =  rint(cc_cardinality);
int cpr = rint(cp_cardinality);
int smr = rint(sm_cardinality);
int wr = rint(w_cardinality);
int rr = rint(r_cardinality);
int rangedr = rintrange(20,2000);
double dec1r = rdec(7,2);
double dec2r = rdec(7,2);
double dec3r = rdec(7,2);
double dec4r = rdec(7,2);
double dec5r = rdec(7,2);
double dec6r = rdec(7,2);
double dec7r = rdec(7,2);
double dec8r = rdec(7,2);
double dec9r = rdec(7,2);

sql =
"INSERT INTO catalog_returns "
+ "(cr_returned_date_sk, cr_returned_time_sk, cr_item_sk, cr_refunded_customer_sk,"
+ " cr_refunded_cdemo_sk, cr_refunded_hdemo_sk, cr_refunded_addr_sk,"
+ " cr_returning_customer_sk, cr_returning_cdemo_sk, cr_returning_hdemo_sk,"
+ " cr_returning_addr_sk, cr_call_center_sk, cr_catalog_page_sk, cr_ship_mode_sk,"
+ " cr_warehouse_sk, cr_reason_sk, cr_order_number, cr_return_quantity, "
+ " cr_return_amount, cr_return_tax, cr_return_amt_inc_tax, cr_fee, cr_return_ship_cost,"
+ " cr_refunded_cash, cr_reversed_charge, cr_store_credit,cr_net_loss) "
+ "VALUES ("
+ d1r + ","+ tr + ","+ rand_item_sk + ","+ c1r + ","+ cd1r + ","+ hd1r + ","
+ ca1r + ","+ c2r + ","+ cd2r + ","+ hd2r + ","+ ca2r + ","+ ccr + ","
+ cpr + ","+ smr + ","+ wr + ","+ rr + ","+ rand_order_num + ","+ rangedr + ","
+ dec1r + ","+ dec2r + ","+ dec3r + ","+ dec4r + ","+ dec5r + ","
+ dec6r + ","+ dec7r + ","+ dec8r + ","+ dec9r + ")";

stmt1.executeUpdate(sql);
stmt2.executeUpdate(sql);
}
}

System.out.println(">>>>>>>>>>>>>catalog_sales-returns UPDATED by " + ncs);

//ws and wr
for(int i=0;i<nws;i++){

rand_item_sk = rint(i_cardinality);
rand_order_num = rint(10000);
while(used(rand_item_sk,rand_order_num,"web_sales","ws_sold_date_sk","ws_item_sk","ws_order_numbe
r")){
rand_item_sk = rint(i_cardinality);
rand_order_num = rint(10000);
}

int d1 = rint(d_cardinality) ;
int t = rint(t_cardinality) ;
int d2 = rint(d_cardinality);
int c1 = rint(c_cardinality);
int cd1 = rint(cd_cardinality);
int hd1 = rint(hd_cardinality);
```

```
int ca1 = rint(ca_cardinality) ;
int c2 = rint(c_cardinality);
int cd2 = rint(cd_cardinality);
int hd2 = rint(hd_cardinality);
int ca2 = rint(ca_cardinality) ;
int wp = rint(wp_cardinality);
int web = rint(web_cardinality);
int sm = rint(sm_cardinality);
int w = rint(w_cardinality);
int p = rint(p_cardinality);
int ranged = rintrange(20,2000);
double dec1 = rdec(7,2);
double dec2 = rdec(7,2);
double dec3 = rdec(7,2);
double dec4 = rdec(7,2);
double dec5 = rdec(7,2);
double dec6 = rdec(7,2);
double dec7 = rdec(7,2);
double dec8 = rdec(7,2);
double dec9 = rdec(7,2);
double dec10 = rdec(7,2);
double dec11 = rdec(7,2);
double dec12 = rdec(7,2);
double dec13 = rdec(7,2);
double dec14 = rdec(7,2);
double dec15 = rdec(7,2);
sql =
"INSERT INTO web_sales "
+ "(ws_sold_date_sk, ws_sold_time_sk, ws_ship_date_sk, ws_item_sk,"
+ " ws_bill_customer_sk, ws_bill_cdemo_sk, ws_bill_hdemo_sk, ws_bill_addr_sk,"
+ " ws_ship_customer_sk, ws_ship_cdemo_sk, ws_ship_hdemo_sk, ws_ship_addr_sk,"
+ " ws_web_page_sk, ws_web_site_sk, ws_ship_mode_sk, ws_warehouse_sk,"
+ " ws_promo_sk, ws_order_number, ws_quantity, ws_wholesale_cost,"
+ " ws_list_price, ws_sales_price, ws_ext_discount_amt, ws_ext_sales_price,"
+ " ws_ext_wholesale_cost, ws_ext_list_price, ws_ext_tax, ws_coupon_amt,"
+ " ws_ext_ship_cost, ws_net_paid, ws_net_paid_inc_tax, ws_net_paid_inc_ship,"
+ " ws_net_paid_inc_ship_tax, ws_net_profit) "
+ "VALUES ("
+ d1 + ","+ t + ","+ d2 + ","+ rand_item_sk + ","+ c1 + ","+ cd1 + ","
+ hd1 + ","+ ca1 + ","+ c2 + ","+ cd2 + ","+ hd2 + ","+ ca2 + ","
+ wp + ","+ web + ","+ sm + ","+ w + ","+ p + ","+ rand_order_num + ","+ ranged + ","
+ dec1 + ","+ dec2 + ","+ dec3 + ","+ dec4 + ","+ dec5 + ","
+ dec6 + ","+ dec7 + ","+ dec8 + ","+ dec9 + ","+ dec10 + ","
+ dec11 + ","+ dec12 + ","+ dec13 + ","+ dec14 + ","+ dec15+ ")";

stmt1.executeUpdate(sql);
stmt2.executeUpdate(sql);

if(i<xw){

int d1r = rint(d_cardinality) ;
int tr = rint(t_cardinality) ;
int c1r = rint(c_cardinality);
int cd1r = rint(cd_cardinality);
int hd1r = rint(hd_cardinality);
int ca1r = rint(ca_cardinality) ;
int c2r = rint(c_cardinality);
int cd2r = rint(cd_cardinality);
int hd2r = rint(hd_cardinality);
int ca2r = rint(ca_cardinality) ;
int wpr = rint(wp_cardinality);
int rr = rint(r_cardinality);
int rangedr = rintrange(20,2000);
```

```
double dec1r = rdec(7,2);
double dec2r = rdec(7,2);
double dec3r = rdec(7,2);
double dec4r = rdec(7,2);
double dec5r = rdec(7,2);
double dec6r = rdec(7,2);
double dec7r = rdec(7,2);
double dec8r = rdec(7,2);
double dec9r = rdec(7,2);
sql =
"INSERT INTO web_returns "
+ "(wr_returned_date_sk, wr_returned_time_sk, wr_item_sk, wr_refunded_customer_sk,"
+ " wr_refunded_cdemo_sk, wr_refunded_hdemo_sk, wr_refunded_addr_sk, "
+ " wr_returning_customer_sk, wr_returning_cdemo_sk, wr_returning_hdemo_sk,"
+ " wr_returning_addr_sk, wr_web_page_sk, wr_reason_sk, wr_order_number,"
+ " wr_return_quantity, wr_return_amt, wr_return_tax, wr_return_amt_inc_tax,"
+ " wr_fee,wr_return_ship_cost, wr_refunded_cash, wr_reversed_charge,"
+ " wr_account_credit, wr_net_loss) "
+ "VALUES ("
+ d1r + ","+ tr + ","+ rand_item_sk + ","+ c1r + ","+ cd1r + ","
+ hd1r + ","+ ca1r + ","+ c2r + ","+ cd2r + ","+ hd2r + ","+ ca2r + ","
+ wpr + ","+ rr + ","+ rand_order_num + ","+ rangedr + ","
+ dec1r + ","+ dec2r + ","+ dec3r + ","+ dec4r + ","+ dec5r + ","
+ dec6r + ","+ dec7r + ","+ dec8r + ","+ dec9r + ")";

stmt1.executeUpdate(sql);
stmt2.executeUpdate(sql);
}
}

System.out.println(">>>>>>>>>>>>>>web_sales-returns UPDATED by " + nws);

rs1.close();

}catch(SQLException se){

se.printStackTrace();
}catch(Exception e){

e.printStackTrace();
}finally{

try{
if(stmt1!=null)
stmt1.close();
if(stmt2!=null)
stmt2.close();
}catch(SQLException se){
}
try{
if(conn1!=null)
conn1.close();
if(conn2!=null)
conn2.close();
}catch(SQLException se){
se.printStackTrace();
}
}

}
}
```

TpcdsBoolDB.java

```
package thesis;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;


public class TpcdsBoolDB {


static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
static final String DB_SERVER_URL = "jdbc:mysql://localhost/";

static final String USER = "root";
static final String PASS = "";

static final String BASICDB  = "tpcds";
static final String BOOLDB  = "bool_tpcds";


//This method (re)creates the bool_tpcds database
public void createBoolDB(){
Connection conn1 = null;
Statement stmt1 = null;
String sql;

try{
Class.forName(JDBC_DRIVER);
conn1 = DriverManager.getConnection(DB_SERVER_URL,USER,PASS);
stmt1 = conn1.createStatement();

sql = "DROP DATABASE IF EXISTS " + BOOLDB;
stmt1.executeUpdate(sql);

sql = "CREATE DATABASE " + BOOLDB;
stmt1.executeUpdate(sql);

stmt1.close();
conn1.close();

System.out.println("CREATED DATABASE : " + BOOLDB);

}
catch(SQLException se){

se.printStackTrace();
}
catch(Exception e){

e.printStackTrace();
}
finally{

try{
if(stmt1!=null)
stmt1.close();
}
catch(SQLException se2){
```

```
}
try{
if(conn1!=null)
conn1.close();
}
catch(SQLException se){
se.printStackTrace();
}
}

}

//This method copies all data from basic to boolean db, using mysqldump
public void copyBasicToBool(){
Connection  conn = null;
Statement  stmt = null;
String sql;

try{
Class.forName(JDBC_DRIVER);

conn = DriverManager.getConnection(DB_SERVER_URL + BOOLDB,USER,PASS);
stmt = conn.createStatement();

//Create all tables using the same CREATE TABLE statements
TpcdsDB rory = new TpcdsDB();

String[] cts = new String[]{
rory.t_ctable,rory.sm_ctable,rory.r_ctable,rory.ib_ctable,rory.i_ctable,
rory.d_ctable,rory.cd_ctable,rory.ca_ctable,rory.w_ctable,rory.hd_ctable,
rory.p_ctable,rory.c_ctable,rory.wp_ctable,rory.web_ctable,rory.cp_ctable,
rory.cc_ctable,rory.s_ctable,rory.inv_ctable,rory.ss_ctable,rory.sr_ctable,
rory.cs_ctable,rory.cr_ctable,rory.ws_ctable,rory.wr_ctable
};

String[] tn = new String[]{
"time_dim","ship_mode","reason","income_band","item",
"date_dim","customer_demographics","customer_address","warehouse","household_demographics",
"promotion","customer","web_page","web_site","catalog_page",
"call_center","store","inventory","store_sales","store_returns",
"catalog_sales","catalog_returns","web_sales","web_returns"
};

for(int i=0;i<cts.length;i++){
//First create table
sql = cts[i];
stmt.executeUpdate(sql);

//Then copy data in the new table
String tname = tn[i];
sql = "INSERT INTO " + tname + " SELECT * FROM " + BASICDB + "." + tname;
stmt.executeUpdate(sql);

System.out.println("Table \'" + tname + "\' copied");
}


stmt.close();
conn.close();

System.out.println("COPIED DATA INTO BOOLEAN DATABASE " );

}
```

```
catch(SQLException se){

se.printStackTrace();
}
catch(Exception e){

e.printStackTrace();
}
finally{

try{
if(stmt!=null)
stmt.close();
}
catch(SQLException se2){
}
try{
if(conn!=null)
conn.close();
}
catch(SQLException se){
se.printStackTrace();
}
}
}

//This method substitutes 'attribute' in 'table' with the corresponding boolean attributes

//This method booleanizes the table on one attribute
public void booleanize(String table, String key, String attribute,String[] val){
Connection conn1 = null, conn2 = null;
Statement stmt1 = null, stmt2 = null;
ResultSet rs;
String sql;

try{
Class.forName(JDBC_DRIVER);
conn1 = DriverManager.getConnection(DB_SERVER_URL+BOOLDB,USER,PASS);
stmt1 = conn1.createStatement();
conn2 = DriverManager.getConnection(DB_SERVER_URL+BOOLDB,USER,PASS);
stmt2 = conn1.createStatement();

//First insert the new columns
for(int i=0;i<val.length;i++){
sql = "ALTER TABLE " + table + " ADD " + attribute + "_" + val[i] + " int(1)";
stmt1.executeUpdate(sql);
}

//Then enter the right 0-1 values in the new columns
sql = "SELECT " + key + " , " + attribute + " FROM " + table;
rs = stmt2.executeQuery(sql);

while(rs.next()){

int k = rs.getInt(key);
String a = rs.getString(attribute);

sql = "UPDATE " + table + " SET " ;

for(int j=0;j<val.length;j++){
if(a.equals(val[j])){
sql += attribute + "_" + val[j] + " = 1," ;
}
```

```
else{
sql += attribute + "_" + val[j] + " = 0," ;
}
}
// Remove last unwanted comma
sql = sql.substring(0, sql.length() - 1);

sql += " WHERE " + key + " = " + k;

stmt1.executeUpdate(sql);

}

//Finally, drop the initial column
sql = "ALTER TABLE " + table + " DROP COLUMN  " + attribute;
stmt1.executeUpdate(sql);

System.out.println("Table " + table + " attribute " + attribute + " booleanized.");

stmt1.close();
conn1.close();

}
catch(SQLException se){

se.printStackTrace();
}
catch(Exception e){

e.printStackTrace();
}
finally{

try{
if(stmt1!=null)
stmt1.close();
if(stmt2!=null)
stmt2.close();
}
catch(SQLException se2){
}
try{
if(conn1!=null)
conn1.close();
if(conn2!=null)
conn2.close();
}
catch(SQLException se){
se.printStackTrace();
}
}
}

}
```

SetUp.java

```
package thesis;


public class SetUp {

public static void main(String[] args) {

TimeKeeper clock = new TimeKeeper();
long startTime = System.currentTimeMillis();
long st;

//BASIC DB ========================================================================

TpcdsDB basic = new TpcdsDB();


//Create the basic DB
basic.createBasicDB();

//Create and populate the tables of the basic DB

st = System.currentTimeMillis();
basic.createTable_t() ;
clock.timer(st, System.currentTimeMillis());

st = System.currentTimeMillis();
basic.createTable_sm() ;
clock.timer(st, System.currentTimeMillis());

st = System.currentTimeMillis();
basic.createTable_r() ;
clock.timer(st, System.currentTimeMillis());

st = System.currentTimeMillis();
basic.createTable_ib() ;
clock.timer(st, System.currentTimeMillis());

st = System.currentTimeMillis();
basic.createTable_i() ;
clock.timer(st, System.currentTimeMillis());

st = System.currentTimeMillis();
basic.createTable_d() ;
clock.timer(st, System.currentTimeMillis());

st = System.currentTimeMillis();
basic.createTable_cd() ;
clock.timer(st, System.currentTimeMillis());

st = System.currentTimeMillis();
basic.createTable_ca() ;
clock.timer(st, System.currentTimeMillis());

st = System.currentTimeMillis();
basic.createTable_w() ;
clock.timer(st, System.currentTimeMillis());

st = System.currentTimeMillis();
basic.createTable_hd() ;
clock.timer(st, System.currentTimeMillis());
```

```
st = System.currentTimeMillis();
basic.createTable_p() ;
clock.timer(st, System.currentTimeMillis());

st = System.currentTimeMillis();
basic.createTable_c() ;
clock.timer(st, System.currentTimeMillis());

st = System.currentTimeMillis();
basic.createTable_wp() ;
clock.timer(st, System.currentTimeMillis());

st = System.currentTimeMillis();
basic.createTable_web() ;
clock.timer(st, System.currentTimeMillis());

st = System.currentTimeMillis();
basic.createTable_cp() ;
clock.timer(st, System.currentTimeMillis());

st = System.currentTimeMillis();
basic.createTable_cc() ;
clock.timer(st, System.currentTimeMillis());

st = System.currentTimeMillis();
basic.createTable_s() ;
clock.timer(st, System.currentTimeMillis());

st = System.currentTimeMillis();
basic.createTable_inv() ;
clock.timer(st, System.currentTimeMillis());

st = System.currentTimeMillis();
basic.createTable_ss_sr() ;
clock.timer(st, System.currentTimeMillis());

st = System.currentTimeMillis();
basic.createTable_cs_cr() ;
clock.timer(st, System.currentTimeMillis());

st = System.currentTimeMillis();
basic.createTable_ws_wr() ;
clock.timer(st, System.currentTimeMillis());


//BOOLEAN DB ================================================================


TpcdsBoolDB bool = new TpcdsBoolDB();

//Create the boolean DB
bool.createBoolDB();

//Copy everything from the basic DB to the boolean DB
bool.copyBasicToBool();


//Booleanize selected attributes
bool.booleanize("customer_demographics", "cd_demo_sk", "cd_gender", basic.cd_gender_vals);
bool.booleanize("customer_demographics", "cd_demo_sk", "cd_marital_status",
basic.cd_marital_status_vals);
bool.booleanize("customer_demographics", "cd_demo_sk", "cd_education_status",
basic.cd_education_status_vals);
```

```
bool.booleanize("time_dim", "t_time_sk", "t_am_pm", basic.t_am_pm_vals);
bool.booleanize("customer", "c_customer_sk", "c_preferred_cust_flag", basic.yes_no_vals);

bool.booleanize("promotion", "p_promo_sk", "p_channel_dmail", basic.yes_no_vals);
bool.booleanize("promotion", "p_promo_sk", "p_channel_email", basic.yes_no_vals);
bool.booleanize("promotion", "p_promo_sk", "p_channel_catalog", basic.yes_no_vals);
bool.booleanize("promotion", "p_promo_sk", "p_channel_tv", basic.yes_no_vals);
bool.booleanize("promotion", "p_promo_sk", "p_channel_radio", basic.yes_no_vals);
bool.booleanize("promotion", "p_promo_sk", "p_channel_press", basic.yes_no_vals);
bool.booleanize("promotion", "p_promo_sk", "p_channel_event", basic.yes_no_vals);
bool.booleanize("promotion", "p_promo_sk", "p_channel_demo", basic.yes_no_vals);

bool.booleanize("date_dim", "d_date_sk", "d_day_name", basic.d_day_name_vals);
bool.booleanize("date_dim", "d_date_sk", "d_month_name", basic.d_month_name_vals);
bool.booleanize("date_dim", "d_date_sk", "d_quarter_name", basic.d_quarter_name_vals);


//bool.booleanize("promotion", "p_promo_sk", "p_discount_active", basic.yes_no_vals);

//bool.booleanize("web_page", "wp_web_page_sk", "wp_autogen_flag" , basic.yes_no_vals);

//bool.booleanize("date_dim", "d_date_sk", "d_holiday" , basic.yes_no_vals);
//bool.booleanize("date_dim", "d_date_sk", "d_weekend" , basic.yes_no_vals);
//bool.booleanize("date_dim", "d_date_sk", "d_following_holiday" , basic.yes_no_vals);
//bool.booleanize("date_dim", "d_date_sk", "d_current_day" , basic.yes_no_vals);
//bool.booleanize("date_dim", "d_date_sk", "d_current_week" , basic.yes_no_vals);
//bool.booleanize("date_dim", "d_date_sk", "d_current_month" , basic.yes_no_vals);
//bool.booleanize("date_dim", "d_date_sk", "d_current_quarter" , basic.yes_no_vals);
//bool.booleanize("date_dim", "d_date_sk", "d_current_year" , basic.yes_no_vals);


System.out.println("SETUP COMPLETE");


clock.timer(startTime, System.currentTimeMillis());
}

}
```

QueryMaker.java

```
package thesis;

import java.io.File;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import jxl.Workbook;
import jxl.write.Label;
import jxl.write.Number;
import jxl.write.WritableSheet;
import jxl.write.WritableWorkbook;
import jxl.write.WriteException;


public class QueryMaker {

// JDBC driver name and database URL
static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
static final String DB_SERVER_URL = "jdbc:mysql://localhost/";

//      Database name
static final String DBNAME1  = "tpcds";
static final String DBNAME2  = "bool_tpcds";

//  Database credentials
static final String USER = "root";
static final String PASS = "Rory1908";

//Path to xls output files
static final String XLS_PATH =
"C:\\Users\\Pantso\\Desktop\\Diplo\\excels\\";

//Excel File name
static final String FILE_NAME = "times.xls";


//Number of loops for each query
static final int LOOPS = 25;

//Percentage of data replaced in fact tables after each query
static final double PERCENTAGE = 0.1;

//Get the queries

static Queries queries = new Queries();
static String[] QUERIES1 = queries.getBasic_queries();
static String[] QUERIES2 = queries.getBool_queries();



public static void main(String[] args) throws IOException, WriteException {

TimeKeeper clock = new TimeKeeper();
long start = System.currentTimeMillis();

Connection conn1 = null,conn2 = null;
Statement stmt1 = null,stmt2 = null;
```

```
TpcdsDB rory = new TpcdsDB();

//Create an Excel workbook
WritableWorkbook ww;
String fname = XLS_PATH + FILE_NAME;
ww = Workbook.createWorkbook(new File(fname));


//Create one sheet for each query
for(int i=0;i<QUERIES1.length;i++){
int queryNum = i + 1;
WritableSheet wsheet = ww.createSheet("query" + queryNum , i);
Label label = new Label(0, 0, "basic");
wsheet.addCell(label);
label = new Label(1, 0, "bool");
wsheet.addCell(label);
}

try {

Class.forName(JDBC_DRIVER);

conn1 = DriverManager.getConnection(DB_SERVER_URL + DBNAME1, USER, PASS);
stmt1 = conn1.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);

conn2 = DriverManager.getConnection(DB_SERVER_URL + DBNAME2, USER, PASS);
stmt2 = conn2.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);


for (int j = 0; j < LOOPS; j++) {

//First update the fact tables
if(j>0){
rory.updateFacts(PERCENTAGE);
}
//Then run the queries...
System.out.println("*************************************************************");
System.out.println("LOOP " + Integer.toString(j + 1));
System.out.println("*************************************************************");

for (int i = 0; i < QUERIES1.length; i++) {

long startTime = System.currentTimeMillis();
ResultSet rs1 = stmt1.executeQuery(QUERIES1[i]);
long time1 = TimeKeeper.strictTimer(startTime, System.currentTimeMillis());

Number num1 = new Number(0, j+1, time1);
ww.getSheet(i).addCell(num1);


startTime = System.currentTimeMillis();
ResultSet rs2 = stmt2.executeQuery(QUERIES2[i]);
long time2 = TimeKeeper.strictTimer(startTime, System.currentTimeMillis());

Number num2 = new Number(1, j + 1, time2);
ww.getSheet(i).addCell(num2);

rs1.close();
rs2.close();

System.out.println("Query " + Integer.toString(i + 1));
System.out.println("db1: " + time1 + "ms" + "\t" + "db2: " + time2 + "ms");
System.out.println("*************************************************************");
```

```
}

System.out.println("************************************************************");
System.out.println("LOOP " + Integer.toString(j + 1) + " COMPLETE" );
System.out.println("************************************************************");

}

ww.write();
ww.close();
}
catch(Exception e){

e.printStackTrace();
}
finally{

try{
if(stmt1!=null)
stmt1.close();
if(stmt2!=null)
stmt2.close();
}
catch(SQLException se){
}
try{
if(conn1!=null)
conn1.close();
if(conn2!=null)
conn2.close();
}
catch(SQLException se){
se.printStackTrace();
}
}
clock.timer(start, System.currentTimeMillis());

}

}
```

Queries.java

```
package thesis;


package thesis;

public class Queries {

//BASIC QUERIES.....................................................................


// QUERY 2
final static String q2 =
" select d_week_seq1,round(sun_sales1/sun_sales2,2) " +
"  ,round(mon_sales1/mon_sales2,2),round(tue_sales1/tue_sales2,2) " +
"     ,round(wed_sales1/wed_sales2,2),round(thu_sales1/thu_sales2,2) " +
"  ,round(fri_sales1/fri_sales2,2),round(sat_sales1/sat_sales2,2) " +
"   " +
"from  " +
"  (select wswscs.d_week_seq d_week_seq1 " +
"        ,sun_sales sun_sales1 " +
"        ,mon_sales mon_sales1 " +
"        ,tue_sales tue_sales1 " +
"        ,wed_sales wed_sales1 " +
"        ,thu_sales thu_sales1 " +
"        ,fri_sales fri_sales1 " +
"        ,sat_sales sat_sales1 " +
"  from (select d_week_seq, " +
"        sum(case when (d_day_name='Sunday') then sales_price else null end) as sun_sales, " +
"        sum(case when (d_day_name='Monday') then sales_price else null end) as mon_sales, " +
"        sum(case when (d_day_name='Tuesday') then sales_price else  null end) as tue_sales, " +
"        sum(case when (d_day_name='Wednesday') then sales_price else null end) as wed_sales, " +
"        sum(case when (d_day_name='Thursday') then sales_price else null end) as thu_sales, " +
"        sum(case when (d_day_name='Friday') then sales_price else null end) as fri_sales, " +
"        sum(case when (d_day_name='Saturday') then sales_price else null end) as sat_sales " +
" from (select sold_date_sk " +
"        ,sales_price " +
"  from (select ws_sold_date_sk sold_date_sk " +
"              ,ws_ext_sales_price sales_price " +
"        from web_sales) x " +
"        union all " +
"       (select cs_sold_date_sk sold_date_sk " +
"              ,cs_ext_sales_price sales_price " +
"        from catalog_sales)) as wscs " +
"     ,date_dim " +
" where d_date_sk = sold_date_sk " +
" group by d_week_seq) as wswscs,date_dim  " +
"  where date_dim.d_week_seq = wswscs.d_week_seq  " +
"  and d_year = 2000) y,  " +
"   " +
"  (select wswscs.d_week_seq d_week_seq2 " +
"        ,sun_sales sun_sales2 " +
"        ,mon_sales mon_sales2 " +
"        ,tue_sales tue_sales2 " +
"        ,wed_sales wed_sales2 " +
"        ,thu_sales thu_sales2 " +
"        ,fri_sales fri_sales2 " +
"        ,sat_sales sat_sales2 " +
"  from (select d_week_seq, " +
"        sum(case when (d_day_name='Sunday') then sales_price else null end) as sun_sales, " +
"        sum(case when (d_day_name='Monday') then sales_price else null end) as mon_sales, " +
"        sum(case when (d_day_name='Tuesday') then sales_price else  null end) as tue_sales, " +
```

```
"        sum(case when (d_day_name='Wednesday') then sales_price else null end) as wed_sales, " +
"        sum(case when (d_day_name='Thursday') then sales_price else null end) as thu_sales, " +
"        sum(case when (d_day_name='Friday') then sales_price else null end) as fri_sales, " +
"        sum(case when (d_day_name='Saturday') then sales_price else null end) as sat_sales " +
" from (select sold_date_sk " +
"        ,sales_price " +
"  from (select ws_sold_date_sk sold_date_sk " +
"              ,ws_ext_sales_price sales_price " +
"        from web_sales) x " +
"        union all " +
"       (select cs_sold_date_sk sold_date_sk " +
"              ,cs_ext_sales_price sales_price " +
"        from catalog_sales)) as wscs " +
"     ,date_dim " +
" where d_date_sk = sold_date_sk " +
" group by d_week_seq) as wswscs,date_dim  " +
"  where date_dim.d_week_seq = wswscs.d_week_seq  " +
"  and d_year = 2001) z " +
"   " +
"where d_week_seq1 < d_week_seq2  " +
"order by d_week_seq1; " ;

// QUERY 6
final static String q6 =
" select a.ca_state as state, count(*) as cnt " +
" from customer_address a,customer c,store_sales s,date_dim d,item i " +
" where a.ca_address_sk = c.c_current_addr_sk " +
" and c.c_customer_sk = s.ss_customer_sk and s.ss_sold_date_sk = d.d_date_sk " +
" and s.ss_item_sk = i.i_item_sk " +
" and d.d_month_seq = (select distinct (d_month_seq) " +
" from date_dim where d_year = 2000 " +
" and d_month_name = 'November' limit 1) " +
" and i.i_current_price > 1.2 * " +
" (select avg(j.i_current_price) from item j " +
" where j.i_category = i.i_category) group by a.ca_state " +
" having count(*) >= 10 order by cnt ";

// QUERY 7
final static String q7 =
" select  i_item_id, " +
" avg(ss_quantity) as agg1, " +
" avg(ss_list_price) as agg2, " +
" avg(ss_coupon_amt) as agg3, " +
" avg(ss_sales_price) as agg4 " +
" from " +
" store_sales, customer_demographics, date_dim, item, promotion " +
" where " +
" ss_sold_date_sk = d_date_sk and " +
" ss_item_sk = i_item_sk and " +
" ss_cdemo_sk = cd_demo_sk and " +
" ss_promo_sk = p_promo_sk and " +
" cd_gender = 'female' and  " +
" cd_marital_status = 'married' and " +
" cd_education_status = 'university' and " +
" (p_channel_email = 'no' or p_channel_event = 'no') and " +
" d_year >  2000 " +
" group by i_item_id " +
" order by i_item_id ";

// QUERY 10_1
final static String q10_1 =
"  SELECT cd_purchase_estimate, " +
"       cd_credit_rating, " +
```

```
"       cd_dep_count, " +
"       cd_dep_employed_count, " +
"       cd_dep_college_count " +
"FROM   customer c, " +
"       customer_address ca, " +
"       customer_demographics " +
"WHERE  c.c_current_addr_sk = ca.ca_address_sk " +
"       AND cd_demo_sk = c.c_current_cdemo_sk " +
"       AND EXISTS (SELECT * " +
"                   FROM   store_sales, " +
"                          date_dim " +
"                   WHERE  c.c_customer_sk = ss_customer_sk " +
"                          AND ss_sold_date_sk = d_date_sk " +
"                          AND d_year > 2000 " +
"                          AND  d_month_name IN ('March','April','May','June','July','August') ) 
" +
"       AND ( EXISTS (SELECT * " +
"                     FROM   web_sales, " +
"                            date_dim " +
"                     WHERE  c.c_customer_sk = ws_bill_customer_sk " +
"                            AND ws_sold_date_sk = d_date_sk " +
"                            AND d_year > 2000 " +
"                            AND ( d_month_name = 'March' " +
"                                   OR d_month_name = 'April' " +
"                                   OR d_month_name = 'May' " +
"                                   OR d_month_name = 'June' " +
"                                   OR d_month_name = 'July' " +
"                                   OR d_month_name = 'August' )) " +
"              OR EXISTS (SELECT * " +
"                         FROM   catalog_sales, " +
"                                date_dim " +
"                         WHERE  c.c_customer_sk = cs_ship_customer_sk " +
"                                AND cs_sold_date_sk = d_date_sk " +
"                                AND d_year > 2000 " +
"                                AND ( d_month_name = 'March' " +
"                                       OR d_month_name = 'April' " +
"                                       OR d_month_name = 'May' " +
"                                       OR d_month_name = 'June' " +
"                                       OR d_month_name = 'July' " +
"                                       OR d_month_name = 'August' )) ) " +
"GROUP  BY cd_purchase_estimate, " +
"          cd_credit_rating, " +
"          cd_dep_count, " +
"          cd_dep_employed_count, " +
"          cd_dep_college_count " +
"ORDER  BY cd_purchase_estimate, " +
"          cd_credit_rating, " +
"          cd_dep_count, " +
"          cd_dep_employed_count, " +
"          cd_dep_college_count;   ";

// QUERY 10_2
final static String q10_2 =
"  SELECT cd_gender, " +
"       cd_marital_status, " +
"       cd_education_status, " +
"       cd_purchase_estimate, " +
"       cd_credit_rating, " +
"       cd_dep_count, " +
"       cd_dep_employed_count, " +
"       cd_dep_college_count " +
"FROM   customer c, " +
"       customer_address ca, " +
```

```
"       customer_demographics " +
"WHERE  c.c_current_addr_sk = ca.ca_address_sk " +
"       AND cd_demo_sk = c.c_current_cdemo_sk " +
"       AND EXISTS (SELECT * " +
"                   FROM   store_sales, " +
"                          date_dim " +
"                   WHERE  c.c_customer_sk = ss_customer_sk " +
"                          AND ss_sold_date_sk = d_date_sk " +
"                          AND d_year > 2000 " +
"                          AND  d_month_name IN ('March','April','May','June','July','August') )
" +
"       AND ( EXISTS (SELECT * " +
"                     FROM   web_sales, " +
"                            date_dim " +
"                     WHERE  c.c_customer_sk = ws_bill_customer_sk " +
"                            AND ws_sold_date_sk = d_date_sk " +
"                            AND d_year > 2000 " +
"                            AND ( d_month_name = 'March' " +
"                                   OR d_month_name = 'April' " +
"                                   OR d_month_name = 'May' " +
"                                   OR d_month_name = 'June' " +
"                                   OR d_month_name = 'July' " +
"                                   OR d_month_name = 'August' )) " +
"              OR EXISTS (SELECT * " +
"                         FROM   catalog_sales, " +
"                                date_dim " +
"                         WHERE  c.c_customer_sk = cs_ship_customer_sk " +
"                                AND cs_sold_date_sk = d_date_sk " +
"                                AND d_year > 2000 " +
"                                AND ( d_month_name = 'March' " +
"                                       OR d_month_name = 'April' " +
"                                       OR d_month_name = 'May' " +
"                                       OR d_month_name = 'June' " +
"                                       OR d_month_name = 'July' " +
"                                       OR d_month_name = 'August' )) ) " +
"GROUP  BY cd_gender, " +
"          cd_marital_status, " +
"          cd_education_status, " +
"          cd_purchase_estimate, " +
"          cd_credit_rating, " +
"          cd_dep_count, " +
"          cd_dep_employed_count, " +
"          cd_dep_college_count " +
"ORDER  BY cd_gender, " +
"          cd_marital_status, " +
"          cd_education_status, " +
"          cd_purchase_estimate, " +
"          cd_credit_rating, " +
"          cd_dep_count, " +
"          cd_dep_employed_count, " +
"          cd_dep_college_count;   ";


// QUERY 11
final static String q11 =
" select "
+ " c_customer_id customer_id "
+ " ,c_first_name customer_first_name ,c_last_name customer_last_name "
+ " ,c_preferred_cust_flag customer_preferred_cust_flag "
+ " ,c_birth_country customer_birth_country "
+ " ,c_login customer_login "
+ " ,c_email_address customer_email_address "
+ " ,d_year dyear,sum(ss_ext_list_price-ss_ext_discount_amt) as year_total "
```

```
+ " ,'s' sale_type "
+ " from customer,store_sales,date_dim "
+ " where c_customer_sk = ss_customer_sk and ss_sold_date_sk = d_date_sk "
+ " group by c_customer_id ,c_first_name,c_last_name,d_year,c_preferred_cust_flag "
+ " ,c_birth_country,c_login,c_email_address ,d_year  "
+ " UNION ALL "
+ " select c_customer_id customer_id "
+ " ,c_first_name customer_first_name,c_last_name customer_last_name "
+ " ,c_preferred_cust_flag customer_preferred_cust_flag "
+ " ,c_birth_country customer_birth_country "
+ " ,c_login customer_login ,c_email_address customer_email_address "
+ " ,d_year dyear,sum(ws_ext_list_price-ws_ext_discount_amt) as year_total "
+ " ,'w' sale_type "
+ " from customer ,web_sales ,date_dim "
+ " where c_customer_sk = ws_bill_customer_sk and ws_sold_date_sk = d_date_sk "
+ " group by c_customer_id ,c_first_name ,c_last_name ,c_preferred_cust_flag "
+ " ,c_birth_country ,c_login ,c_email_address,d_year ";

// QUERY 13
final static String q13 =
" select avg(ss_quantity),avg(ss_ext_sales_price),avg(ss_ext_wholesale_cost) "
+ " ,sum(ss_ext_wholesale_cost) "
+ " from store_sales,store,customer_demographics,household_demographics "
+ " ,customer_address ,date_dim "
+ " where s_store_sk = ss_store_sk "
+ " and  ss_sold_date_sk = d_date_sk and d_year = 2001 "
+ " and((ss_hdemo_sk=hd_demo_sk "
+ " and cd_demo_sk = ss_cdemo_sk "
+ " and cd_marital_status = 'married' "
+ " and cd_education_status = 'university' "
+ " and ss_sales_price between 10000.00 and 50000.00 "
+ " and hd_dep_count < 5 "
+ " ) or (ss_hdemo_sk=hd_demo_sk "
+ " and cd_demo_sk = ss_cdemo_sk "
+ " and cd_marital_status = 'divorced' "
+ " and cd_education_status = 'secondary' "
+ " and ss_sales_price between 20000.00 and 60000.00 "
+ " and hd_dep_count > 5 "
+ " ) or (ss_hdemo_sk=hd_demo_sk "
+ " and cd_demo_sk = ss_cdemo_sk "
+ " and cd_marital_status = 'single' "
+ " and cd_education_status = 'postgraduate' "
+ " and ss_sales_price between 30000.00 and 70000.00 "
+ " and hd_dep_count < 7 ) ) ";

// QUERY 15
final static String q15 =
" select ca_zip, sum(cs_sales_price) " +
" from catalog_sales,customer ,customer_address,date_dim " +
" where cs_bill_customer_sk = c_customer_sk and c_current_addr_sk = ca_address_sk " +
" and cs_sales_price > 500 and cs_sold_date_sk = d_date_sk " +
" and d_quarter_name = 'first' and d_year > 2000 " +
" group by ca_zip order by ca_zip ";

// QUERY 17
final static String q17 =
" select i_item_id ,i_item_desc,s_state,count(ss_quantity) as store_sales_quantitycount " +
" ,avg(ss_quantity) as store_sales_quantityave " +
" ,stddev_samp(ss_quantity) as store_sales_quantitystdev " +
" ,stddev_samp(ss_quantity)/avg(ss_quantity) as store_sales_quantitycov " +
" ,count(sr_return_quantity) as as_store_returns_quantitycount " +
" ,avg(sr_return_quantity) as as_store_returns_quantityave " +
" ,stddev_samp(sr_return_quantity) as as_store_returns_quantitystdev " +
```

```
" ,stddev_samp(sr_return_quantity)/avg(sr_return_quantity) as store_returns_quantitycov " +
"  ,count(cs_quantity) as catalog_sales_quantitycount " +
" ,avg(cs_quantity) as catalog_sales_quantityave " +
" ,stddev_samp(cs_quantity)/avg(cs_quantity) as catalog_sales_quantitystdev " +
" ,stddev_samp(cs_quantity)/avg(cs_quantity) as catalog_sales_quantitycov " +
" from store_sales,store_returns,catalog_sales,date_dim d1,date_dim d2 " +
" ,date_dim d3,store,item " +
" where d1.d_quarter_name = 'first' " +
" and d1.d_date_sk = ss_sold_date_sk and i_item_sk = ss_item_sk " +
" and s_store_sk = ss_store_sk and ss_customer_sk = sr_customer_sk " +
" and ss_item_sk = sr_item_sk and ss_ticket_number = sr_ticket_number " +
" and sr_returned_date_sk = d2.d_date_sk " +
" and d2.d_quarter_name in ('first','second','third') " +
" and sr_customer_sk = cs_bill_customer_sk " +
" and sr_item_sk = cs_item_sk " +
" and cs_sold_date_sk = d3.d_date_sk " +
" and d3.d_quarter_name in ('first','second','third') " +
" group by i_item_id " +
" ,i_item_desc ,s_state " +
" order by i_item_id,i_item_desc,s_state ";

// QUERY 19
final static String q19 =
" select i_brand_id as brand_id, i_brand as brand, i_manufact_id, i_manufact, " +
" sum(ss_ext_sales_price) as ext_price " +
" from date_dim, store_sales, item,customer,customer_address,store " +
" where d_date_sk = ss_sold_date_sk and ss_item_sk = i_item_sk " +
" and d_month_name = 'May' " +
" and d_year > 2000 " +
" and ss_customer_sk = c_customer_sk " +
" and c_current_addr_sk = ca_address_sk " +
" and ss_store_sk = s_store_sk " +
" group by i_brand ,i_brand_id,i_manufact_id,i_manufact " +
" order by ext_price desc,i_brand,i_brand_id,i_manufact_id,i_manufact ";

// QUERY 26
final static String q26 =
" select i_item_id, avg(cs_quantity) as agg1, "
+ " avg(cs_list_price) as agg2, avg(cs_coupon_amt) as agg3, "
+ " avg(cs_sales_price) as agg4 "
+ " from catalog_sales, customer_demographics, date_dim, item, promotion "
+ " where cs_sold_date_sk = d_date_sk and "
+ " cs_item_sk = i_item_sk and "
+ " cs_bill_cdemo_sk = cd_demo_sk and "
+ " cs_promo_sk = p_promo_sk and "
+ " cd_gender = 'female' and "
+ " cd_marital_status = 'single' and "
+ " cd_education_status = 'university' and "
+ " (p_channel_email = 'no' or p_channel_event = 'no') and "
+ " d_year < 2000 "
+ " group by i_item_id "
+ " order by i_item_id " ;

// QUERY 30
final static String q30 =
" select  c_customer_id,c_salutation,c_first_name,c_last_name,c_preferred_cust_flag " +
"       ,c_birth_day,c_birth_month,c_birth_year,c_birth_country,c_login,c_email_address " +
"       ,c_last_review_date_sk,ctr_total_return " +
"    " +
"from  " +
" (select wr_returning_customer_sk as ctr_customer_sk " +
"        ,ca_state as ctr_state,  " +
"   sum(wr_return_amt) as ctr_total_return " +
```

```
" from web_returns " +
"     ,date_dim " +
"     ,customer_address " +
" where wr_returned_date_sk = d_date_sk  " +
"   and d_year > 2000 " +
"   and wr_returning_addr_sk = ca_address_sk  " +
" group by wr_returning_customer_sk " +
"         ,ca_state) as ctr1,customer_address,customer " +
"    " +
"where ctr1.ctr_total_return > (select avg(ctr_total_return)*1.2 " +
"         from (select wr_returning_customer_sk as ctr_customer_sk " +
"        ,ca_state as ctr_state,  " +
"   sum(wr_return_amt) as ctr_total_return " +
" from web_returns " +
"     ,date_dim " +
"     ,customer_address " +
" where wr_returned_date_sk = d_date_sk  " +
"   and d_year > 2000 " +
"   and wr_returning_addr_sk = ca_address_sk  " +
" group by wr_returning_customer_sk " +
"         ,ca_state) as ctr2  " +
"                       where ctr1.ctr_state = ctr2.ctr_state) " +
"       and ca_address_sk = c_current_addr_sk " +
"       and ctr1.ctr_customer_sk = c_customer_sk " +
" order by c_customer_id,c_salutation,c_first_name,c_last_name,c_preferred_cust_flag " +
"
,c_birth_day,c_birth_month,c_birth_year,c_birth_country,c_login,c_email_address " +
"                  ,c_last_review_date_sk,ctr_total_return ";

// QUERY 34
final static String q34 =
" SELECT c_last_name,c_first_name,c_salutation,c_preferred_cust_flag,  " +
"       ss_ticket_number,cnt  " +
"FROM   (SELECT ss_ticket_number,ss_customer_sk,Count(*) AS cnt  " +
"        FROM   store_sales,date_dim,store,household_demographics  " +
"        WHERE  store_sales.ss_sold_date_sk = date_dim.d_date_sk  " +
"               AND store_sales.ss_store_sk = store.s_store_sk  " +
"               AND store_sales.ss_hdemo_sk = household_demographics.hd_demo_sk  " +
"               AND ( date_dim.d_day_name='Monday' OR date_dim.d_day_name='Wednesday'  " +
"                      OR date_dim.d_day_name='Friday' )  " +
"               AND household_demographics.hd_vehicle_count > 0  " +
"               AND ( CASE  " +
"                       WHEN household_demographics.hd_vehicle_count > 0 THEN  " +
"                       household_demographics.hd_dep_count /  " +
"                       household_demographics.hd_vehicle_count  " +
"                       ELSE NULL  " +
"                     end ) > 1.2  " +
"               AND date_dim.d_year IN ( 1999, 2000, 2001 )  " +
"        GROUP  BY ss_ticket_number,ss_customer_sk) dn,customer  " +
"WHERE  ss_customer_sk = c_customer_sk  " +
"       AND cnt BETWEEN 0 AND 20  " +
"ORDER  BY c_last_name,c_first_name,c_salutation,c_preferred_cust_flag DESC  ";

// QUERY 39A
final static String q39A =
" select inv1.w_warehouse_sk,inv1.i_item_sk,inv1.d_month_name,inv1.mean, inv1.cov " +
"        ,inv2.w_warehouse_sk,inv2.i_item_sk,inv2.d_month_name,inv2.mean, inv2.cov " +
"   " +
"from  " +
" (select w_warehouse_name,w_warehouse_sk,i_item_sk,d_month_name " +
"       ,stdev,mean, case mean when 0 then null else stdev/mean end cov " +
" from(select w_warehouse_name,w_warehouse_sk,i_item_sk,d_month_name " +
"            ,stddev_samp(inv_quantity_on_hand) stdev,avg(inv_quantity_on_hand) as mean " +
```

```
"      from inventory " +
"          ,item " +
"          ,warehouse " +
"          ,date_dim " +
"      where inv_item_sk = i_item_sk " +
"        and inv_warehouse_sk = w_warehouse_sk " +
"        and inv_date_sk = d_date_sk " +
"        and d_year > 2000 " +
"      group by w_warehouse_name,w_warehouse_sk,i_item_sk,d_month_name) foo " +
" where case mean when 0 then 0 else stdev/mean end > 1) as inv1, " +
"  " +
" (select w_warehouse_name,w_warehouse_sk,i_item_sk,d_month_name " +
"       ,stdev,mean, case mean when 0 then null else stdev/mean end cov " +
" from(select w_warehouse_name,w_warehouse_sk,i_item_sk,d_month_name " +
"            ,stddev_samp(inv_quantity_on_hand) stdev,avg(inv_quantity_on_hand) as mean " +
"      from inventory " +
"          ,item " +
"          ,warehouse " +
"          ,date_dim " +
"      where inv_item_sk = i_item_sk " +
"        and inv_warehouse_sk = w_warehouse_sk " +
"        and inv_date_sk = d_date_sk " +
"        and d_year > 2000  " +
"      group by w_warehouse_name,w_warehouse_sk,i_item_sk,d_month_name) foo " +
" where case mean when 0 then 0 else stdev/mean end > 1) as inv2  " +
"  " +
" where inv1.i_item_sk = inv2.i_item_sk " +
"  and inv1.w_warehouse_sk =  inv2.w_warehouse_sk " +
"  and inv1.d_month_name = 'January' " +
"  and inv2.d_month_name = 'February' " +
"order by inv1.w_warehouse_sk,inv1.i_item_sk,inv1.d_month_name,inv1.mean,inv1.cov " +
"        ,inv2.d_month_name,inv2.mean, inv2.cov ";

// QUERY 39_B
final static String q39B =
" select inv1.w_warehouse_sk,inv1.i_item_sk,inv1.d_month_name,inv1.mean, inv1.cov " +
"        ,inv2.w_warehouse_sk,inv2.i_item_sk,inv2.d_month_name,inv2.mean, inv2.cov " +
"   " +
"from  " +
"  (select w_warehouse_name,w_warehouse_sk,i_item_sk,d_month_name " +
"       ,stdev,mean, case mean when 0 then null else stdev/mean end cov " +
" from(select w_warehouse_name,w_warehouse_sk,i_item_sk,d_month_name " +
"            ,stddev_samp(inv_quantity_on_hand) stdev,avg(inv_quantity_on_hand) as mean " +
"      from inventory " +
"          ,item " +
"          ,warehouse " +
"          ,date_dim " +
"      where inv_item_sk = i_item_sk " +
"        and inv_warehouse_sk = w_warehouse_sk " +
"        and inv_date_sk = d_date_sk " +
"        and d_year > 2000 " +
"      group by w_warehouse_name,w_warehouse_sk,i_item_sk,d_month_name) foo " +
" where case mean when 0 then 0 else stdev/mean end > 1) as inv1, " +
"  (select w_warehouse_name,w_warehouse_sk,i_item_sk,d_month_name " +
"       ,stdev,mean, case mean when 0 then null else stdev/mean end cov " +
" from(select w_warehouse_name,w_warehouse_sk,i_item_sk,d_month_name " +
"            ,stddev_samp(inv_quantity_on_hand) stdev,avg(inv_quantity_on_hand) as mean " +
"      from inventory " +
"          ,item " +
"          ,warehouse " +
"          ,date_dim " +
"      where inv_item_sk = i_item_sk " +
"        and inv_warehouse_sk = w_warehouse_sk " +
```

```
"        and inv_date_sk = d_date_sk " +
"        and d_year > 2000 " +
"      group by w_warehouse_name,w_warehouse_sk,i_item_sk,d_month_name) foo " +
" where case mean when 0 then 0 else stdev/mean end > 1) as inv2 " +
" " +
"where inv1.i_item_sk = inv2.i_item_sk " +
"  and inv1.w_warehouse_sk =  inv2.w_warehouse_sk " +
"  and inv1.d_month_name='October' " +
"  and inv2.d_month_name='September' " +
"  and inv1.cov > 1.5 " +
"order by inv1.w_warehouse_sk,inv1.i_item_sk,inv1.d_month_name,inv1.mean,inv1.cov " +
"        ,inv2.d_month_name,inv2.mean, inv2.cov ";

// QUERY 42
final static String q42 =
" select dt.d_year,item.i_category_id,item.i_category,sum(ss_ext_sales_price) " +
" from   date_dim dt,store_sales,item " +
" where dt.d_date_sk = store_sales.ss_sold_date_sk " +
" and store_sales.ss_item_sk = item.i_item_sk " +
" and dt.d_month_name = 'April' " +
" and dt.d_year > 2000 " +
" group by   dt.d_year,item.i_category_id,item.i_category " +
" order by sum(ss_ext_sales_price) desc,dt.d_year,item.i_category_id,item.i_category ";

// QUERY 43
final static String q43 =
" select s_store_name, s_store_id, " +
" sum(case when (d_day_name='Sunday') then ss_sales_price else null end) as sun_sales, " +
" sum(case when (d_day_name='Monday') then ss_sales_price else null end) as mon_sales, " +
" sum(case when (d_day_name='Tuesday') then ss_sales_price else  null end) as tue_sales, " +
" sum(case when (d_day_name='Wednesday') then ss_sales_price else null end) as wed_sales, " +
" sum(case when (d_day_name='Thursday') then ss_sales_price else null end) as thu_sales, " +
" sum(case when (d_day_name='Friday') then ss_sales_price else null end) as fri_sales, " +
" sum(case when (d_day_name='Saturday') then ss_sales_price else null end) as sat_sales " +
" from date_dim, store_sales, store " +
" where d_date_sk = ss_sold_date_sk and s_store_sk = ss_store_sk and " +
" d_year > 2000 " +
" group by s_store_name, s_store_id " +
" order by s_store_name, s_store_id,sun_sales, " +
" mon_sales,tue_sales,wed_sales,thu_sales,fri_sales,sat_sales ";

// QUERY 45
final static String q45 =
" select ca_zip,ca_city,ca_county,ca_state , sum(ws_sales_price) " +
" from web_sales, customer, customer_address, date_dim, item " +
" where ws_bill_customer_sk = c_customer_sk and c_current_addr_sk = ca_address_sk " +
" and ws_item_sk = i_item_sk and i_item_id in (select i_item_id " +
" from item where i_item_sk < 5000 ) " +
" and ws_sold_date_sk = d_date_sk " +
" and d_quarter_name = 'fourth' and d_year > 2000 " +
" group by ca_zip, ca_city,ca_county,ca_state " +
" order by ca_zip, ca_city,ca_county,ca_state ";

// QUERY 46
final static String q46 =
" select c_last_name ,c_first_name,ca_city,bought_city,ss_ticket_number,amt,profit " +
" from (select ss_ticket_number,ss_customer_sk,ca_city as bought_city " +
" ,sum(ss_coupon_amt) as amt,sum(ss_net_profit) as profit " +
" from store_sales,date_dim,store,household_demographics,customer_address " +
" where store_sales.ss_sold_date_sk = date_dim.d_date_sk " +
" and store_sales.ss_store_sk = store.s_store_sk  " +
" and store_sales.ss_hdemo_sk = household_demographics.hd_demo_sk " +
" and store_sales.ss_addr_sk = customer_address.ca_address_sk " +
```

```
" and (household_demographics.hd_dep_count < 5 or " +
" household_demographics.hd_vehicle_count < 2) " +
" and date_dim.d_day_name in ('Monday','Tuesday','Wednesday','Thursday') " +
" and date_dim.d_year in (1999,2000,2001) " +
" group by ss_ticket_number,ss_customer_sk,ss_addr_sk,ca_city) " +
" dn,customer,customer_address current_addr where ss_customer_sk = c_customer_sk " +
" and customer.c_current_addr_sk = current_addr.ca_address_sk " +
" and current_addr.ca_city <> bought_city " +
" order by c_last_name ,c_first_name,ca_city ,bought_city,ss_ticket_number ";


// QUERY 48
final static String q48 =
" select sum(ss_quantity) "
+ " from store_sales, store, customer_demographics, customer_address, date_dim "
+ " where s_store_sk = ss_store_sk "
+ " and  ss_sold_date_sk = d_date_sk and d_year > 2000 "
+ " and (( "
+ " cd_demo_sk = ss_cdemo_sk and "
+ " cd_marital_status = 'married' and "
+ " cd_education_status = 'university' and "
+ " ss_sales_price between 25000.00 and 50000.00 ) or ( "
+ " cd_demo_sk = ss_cdemo_sk and "
+ " cd_marital_status = 'married' and "
+ " cd_education_status = 'university' and "
+ " ss_sales_price between 0.00 and 25000.00 ) or ( "
+ " cd_demo_sk = ss_cdemo_sk and "
+ " cd_marital_status = 'married' and "
+ " cd_education_status = 'university' and "
+ " ss_sales_price between 75000.00 and 100000.00)) ";

//QUERY 49
final static String q49 =
" select  " +
" 'web' as channel " +
" ,web.item " +
" ,web.return_ratio " +
" ,web.return_rank " +
" ,web.currency_rank " +
" from ( " +
"   select  " +
"    item " +
"   ,return_ratio " +
"   ,currency_ratio " +
"   ,@return_rank := @return_rank + 1 AS return_rank " +
"   ,@currency_rank := @currency_rank + 1 AS currency_rank " +
"   from  " +
"   (SELECT @return_rank := 0 ) r1,  " +
"   (SELECT @currency_rank := 0 ) r2,  " +
"    " +
"   (  select ws.ws_item_sk as item " +
"     ,(cast(sum(coalesce(wr.wr_return_quantity,0)) as dec(15,4))/ " +
"     cast(sum(coalesce(ws.ws_quantity,0)) as dec(15,4) )) as return_ratio " +
"     ,(cast(sum(coalesce(wr.wr_return_amt,0)) as dec(15,4))/ " +
"     cast(sum(coalesce(ws.ws_net_paid,0)) as dec(15,4) )) as currency_ratio " +
"     from  " +
"      web_sales ws left outer join web_returns wr  " +
"       on (ws.ws_order_number = wr.wr_order_number and  " +
"       ws.ws_item_sk = wr.wr_item_sk) " +
"                 ,date_dim " +
"     where  " +
"       wr.wr_return_amt > 10000  " +
"       and ws.ws_net_profit > 1 " +
```

```
"                         and ws.ws_net_paid > 0 " +
"                         and ws.ws_quantity > 0 " +
"                         and ws_sold_date_sk = d_date_sk " +
"                         and d_year > 2000 " +
"                         and d_month_name = 'December' " +
"     group by ws.ws_item_sk " +
"   ) in_web " +
" ) web " +
" where  " +
" ( " +
" web.return_rank <= 10 " +
" or " +
" web.currency_rank <= 10 " +
" ) " +
" union " +
" select  " +
" 'catalog' as channel " +
" ,catalog.item " +
" ,catalog.return_ratio " +
" ,catalog.return_rank " +
" ,catalog.currency_rank " +
" from ( " +
"   select  " +
"    item " +
"   ,return_ratio " +
"   ,currency_ratio " +
"   ,@return_rank := @return_rank + 1 AS return_rank " +
"   ,@currency_rank := @currency_rank + 1 AS currency_rank " +
"   from  " +
"   (SELECT @return_rank := 0 ) r1,  " +
"   (SELECT @currency_rank := 0 ) r2, " +
"   (  select  " +
"     cs.cs_item_sk as item " +
"     ,(cast(sum(coalesce(cr.cr_return_quantity,0)) as dec(15,4))/ " +
"     cast(sum(coalesce(cs.cs_quantity,0)) as dec(15,4) )) as return_ratio " +
"     ,(cast(sum(coalesce(cr.cr_return_amount,0)) as dec(15,4))/ " +
"     cast(sum(coalesce(cs.cs_net_paid,0)) as dec(15,4) )) as currency_ratio " +
"     from  " +
"     catalog_sales cs left outer join catalog_returns cr " +
"       on (cs.cs_order_number = cr.cr_order_number and  " +
"       cs.cs_item_sk = cr.cr_item_sk) " +
"                ,date_dim " +
"     where  " +
"       cr.cr_return_amount > 10000  " +
"       and cs.cs_net_profit > 1 " +
"                         and cs.cs_net_paid > 0 " +
"                         and cs.cs_quantity > 0 " +
"                         and cs_sold_date_sk = d_date_sk " +
"                         and d_year > 2000 " +
"                         and d_month_name = 'December' " +
"                 group by cs.cs_item_sk " +
"   ) in_cat " +
" ) catalog " +
" where  " +
" ( " +
" catalog.return_rank <= 10 " +
" or " +
" catalog.currency_rank <=10 " +
" ) " +
" union " +
" select  " +
" 'store' as channel " +
" ,store.item " +
```

```
" ,store.return_ratio " +
" ,store.return_rank " +
" ,store.currency_rank " +
" from ( " +
"   select  " +
"    item " +
"   ,return_ratio " +
"   ,currency_ratio " +
"   ,@return_rank := @return_rank + 1 AS return_rank " +
"   ,@currency_rank := @currency_rank + 1 AS currency_rank " +
"   from  " +
"   (SELECT @return_rank := 0 ) r1,  " +
"   (SELECT @currency_rank := 0 ) r2, " +
"   (  select sts.ss_item_sk as item " +
"     ,(cast(sum(coalesce(sr.sr_return_quantity,0)) as
dec(15,4))/cast(sum(coalesce(sts.ss_quantity,0)) as dec(15,4) )) as return_ratio " +
"     ,(cast(sum(coalesce(sr.sr_return_amt,0)) as
dec(15,4))/cast(sum(coalesce(sts.ss_net_paid,0)) as dec(15,4) )) as currency_ratio " +
"     from  " +
"     store_sales sts left outer join store_returns sr " +
"       on (sts.ss_ticket_number = sr.sr_ticket_number and sts.ss_item_sk = sr.sr_item_sk) " +
"                ,date_dim " +
"     where  " +
"       sr.sr_return_amt > 10000  " +
"       and sts.ss_net_profit > 1 " +
"                         and sts.ss_net_paid > 0  " +
"                         and sts.ss_quantity > 0 " +
"                         and ss_sold_date_sk = d_date_sk " +
"                         and d_year > 2000 " +
"                         and d_month_name = 'December' " +
"     group by sts.ss_item_sk " +
"   ) in_store " +
" ) store " +
" where  ( " +
" store.return_rank <= 10 " +
" or  " +
" store.currency_rank <= 10 " +
" ) " +
" order by 1,4,5 " ;

// QUERY 50
final static String q50 =
"  SELECT s_store_name, " +
"       s_company_id, " +
"       s_street_number, " +
"       s_street_name, " +
"       s_street_type, " +
"       s_suite_number, " +
"       s_city, " +
"       s_county, " +
"       s_state, " +
"       s_zip, " +
"       Sum(CASE " +
"             WHEN ( sr_returned_date_sk - ss_sold_date_sk <= 30 ) THEN 1 " +
"             ELSE 0 " +
"           END) AS '30 days', " +
"       Sum(CASE " +
"             WHEN ( sr_returned_date_sk - ss_sold_date_sk > 30 ) " +
"                  AND ( sr_returned_date_sk - ss_sold_date_sk <= 60 ) THEN 1 " +
"             ELSE 0 " +
"           END) AS '31-60 days', " +
"       Sum(CASE " +
"             WHEN ( sr_returned_date_sk - ss_sold_date_sk > 60 ) " +
```

```
"                  AND ( sr_returned_date_sk - ss_sold_date_sk <= 90 ) THEN 1 " +
"             ELSE 0 " +
"           END) AS '61-90 days', " +
"       Sum(CASE " +
"             WHEN ( sr_returned_date_sk - ss_sold_date_sk > 90 ) " +
"                  AND ( sr_returned_date_sk - ss_sold_date_sk <= 120 ) THEN 1 " +
"             ELSE 0 " +
"           END) AS '91-120 days', " +
"       Sum(CASE " +
"             WHEN ( sr_returned_date_sk - ss_sold_date_sk > 120 ) THEN 1 " +
"             ELSE 0 " +
"           END) AS '>120 days' " +
"FROM   store_sales, " +
"       store_returns, " +
"       store, " +
"       date_dim d1, " +
"       date_dim d2 " +
"WHERE  d2.d_year > 2000 " +
"       AND d2.d_month_name = 'May' " +
"       AND ss_ticket_number = sr_ticket_number " +
"       AND ss_item_sk = sr_item_sk " +
"       AND ss_sold_date_sk = d1.d_date_sk " +
"       AND sr_returned_date_sk = d2.d_date_sk " +
"       AND ss_customer_sk = sr_customer_sk " +
"       AND ss_store_sk = s_store_sk " +
"GROUP  BY s_store_name, " +
"          s_company_id, " +
"          s_street_number, " +
"          s_street_name, " +
"          s_street_type, " +
"          s_suite_number, " +
"          s_city, " +
"          s_county, " +
"          s_state, " +
"          s_zip " +
"ORDER  BY s_store_name, " +
"          s_company_id, " +
"          s_street_number, " +
"          s_street_name, " +
"          s_street_type, " +
"          s_suite_number, " +
"          s_city, " +
"          s_county, " +
"          s_state, " +
"          s_zip;   ";

// QUERY 52
final static String q52 =
" select dt.d_year,item.i_brand_id as brand_id,item.i_brand as brand " +
" ,sum(ss_ext_sales_price) as ext_price " +
" from date_dim dt,store_sales,item " +
" where dt.d_date_sk = store_sales.ss_sold_date_sk " +
" and store_sales.ss_item_sk = item.i_item_sk " +
" and dt.d_month_name = 'September' " +
" and dt.d_year > 2000 " +
" group by dt.d_year,item.i_brand, " +
" item.i_brand_id order by dt.d_year,ext_price desc,brand_id  ";

// QUERY 55
final static String q55 =
" select i_brand_id as brand_id, i_brand as brand, " +
"  sum(ss_ext_sales_price) as ext_price " +
" from date_dim, store_sales, item " +
```

```
" where d_date_sk = ss_sold_date_sk " +
"   and ss_item_sk = i_item_sk " +
"   and d_month_name = 'July' " +
"   and d_year > 2000 " +
" group by i_brand, i_brand_id " +
" order by ext_price desc, i_brand_id ";

// QUERY 64
final static  String q64 =
" select cs1.product_name,cs1.store_name " +
"     ,cs1.store_zip,cs1.b_street_number " +
"     ,cs1.b_streen_name,cs1.b_city " +
"     ,cs1.b_zip,cs1.c_street_number " +
"     ,cs1.c_street_name,cs1.c_city " +
"     ,cs1.c_zip,cs1.syear " +
"     ,cs1.cnt,cs1.s1,cs1.s2,cs1.s3 " +
"     ,cs2.s1,cs2.s2,cs2.s3,cs2.syear,cs2.cnt " +
" " +
"from  " +
"  (select i_product_name product_name " +
"     ,i_item_sk item_sk " +
"     ,s_store_name store_name " +
"     ,s_zip store_zip " +
"     ,ad1.ca_street_number b_street_number " +
"     ,ad1.ca_street_name b_streen_name " +
"     ,ad1.ca_city b_city " +
"     ,ad1.ca_zip b_zip " +
"     ,ad2.ca_street_number c_street_number " +
"     ,ad2.ca_street_name c_street_name " +
"     ,ad2.ca_city c_city " +
"     ,ad2.ca_zip c_zip " +
"     ,d1.d_year as syear " +
"     ,d2.d_year as fsyear " +
"     ,d3.d_year s2year " +
"     ,count(*) as cnt " +
"     ,sum(ss_wholesale_cost) as s1 " +
"     ,sum(ss_list_price) as s2 " +
"     ,sum(ss_coupon_amt) as s3 " +
"  FROM   store_sales " +
"        ,store_returns " +
"        , (select cs_item_sk " +
"        ,sum(cs_ext_list_price) as sale,sum(cr_refunded_cash+cr_reversed_charge+cr_store_credit)
as refund " +
"  from catalog_sales " +
"      ,catalog_returns " +
"  where cs_item_sk = cr_item_sk " +
"    and cs_order_number = cr_order_number " +
"  group by cs_item_sk " +
"  having sum(cs_ext_list_price)>2*sum(cr_refunded_cash+cr_reversed_charge+cr_store_credit)) as
cs_ui " +
"        ,date_dim d1 " +
"        ,date_dim d2 " +
"        ,date_dim d3 " +
"        ,store " +
"        ,customer " +
"        ,customer_demographics cd1 " +
"        ,customer_demographics cd2 " +
"        ,promotion " +
"        ,household_demographics hd1 " +
"        ,household_demographics hd2 " +
"        ,customer_address ad1 " +
"        ,customer_address ad2 " +
"        ,income_band ib1 " +
```

```
"        ,income_band ib2 " +
"        ,item " +
"  WHERE  ss_store_sk = s_store_sk AND " +
"         ss_sold_date_sk = d1.d_date_sk AND " +
"         ss_customer_sk = c_customer_sk AND " +
"         ss_cdemo_sk= cd1.cd_demo_sk AND " +
"         ss_hdemo_sk = hd1.hd_demo_sk AND " +
"         ss_addr_sk = ad1.ca_address_sk and " +
"         ss_item_sk = i_item_sk and " +
"         ss_item_sk = sr_item_sk and " +
"         ss_ticket_number = sr_ticket_number and " +
"         ss_item_sk = cs_ui.cs_item_sk and " +
"         c_current_cdemo_sk = cd2.cd_demo_sk AND " +
"         c_current_hdemo_sk = hd2.hd_demo_sk AND " +
"         c_current_addr_sk = ad2.ca_address_sk and " +
"         c_first_sales_date_sk = d2.d_date_sk and " +
"         c_first_shipto_date_sk = d3.d_date_sk and " +
"         ss_promo_sk = p_promo_sk and " +
"         hd1.hd_income_band_sk = ib1.ib_income_band_sk and " +
"         hd2.hd_income_band_sk = ib2.ib_income_band_sk and " +
"         cd1.cd_marital_status != cd2.cd_marital_status " +
"group by i_product_name " +
"       ,i_item_sk " +
"       ,s_store_name " +
"       ,s_zip " +
"       ,ad1.ca_street_number " +
"       ,ad1.ca_street_name " +
"       ,ad1.ca_city " +
"       ,ad1.ca_zip " +
"       ,ad2.ca_street_number " +
"       ,ad2.ca_street_name " +
"       ,ad2.ca_city " +
"       ,ad2.ca_zip " +
"       ,d1.d_year " +
"       ,d2.d_year " +
"       ,d3.d_year) as cs1, " +
"     " +
"  (select i_product_name product_name " +
"     ,i_item_sk item_sk " +
"     ,s_store_name store_name " +
"     ,s_zip store_zip " +
"     ,ad1.ca_street_number b_street_number " +
"     ,ad1.ca_street_name b_streen_name " +
"     ,ad1.ca_city b_city " +
"     ,ad1.ca_zip b_zip " +
"     ,ad2.ca_street_number c_street_number " +
"     ,ad2.ca_street_name c_street_name " +
"     ,ad2.ca_city c_city " +
"     ,ad2.ca_zip c_zip " +
"     ,d1.d_year as syear " +
"     ,d2.d_year as fsyear " +
"     ,d3.d_year s2year " +
"     ,count(*) as cnt " +
"     ,sum(ss_wholesale_cost) as s1 " +
"     ,sum(ss_list_price) as s2 " +
"     ,sum(ss_coupon_amt) as s3 " +
"  FROM   store_sales " +
"        ,store_returns " +
"        , (select cs_item_sk " +
"        ,sum(cs_ext_list_price) as sale,sum(cr_refunded_cash+cr_reversed_charge+cr_store_credit)
as refund " +
"  from catalog_sales " +
"      ,catalog_returns " +
```

```
"  where cs_item_sk = cr_item_sk " +
"    and cs_order_number = cr_order_number " +
"  group by cs_item_sk " +
"  having sum(cs_ext_list_price)>2*sum(cr_refunded_cash+cr_reversed_charge+cr_store_credit)) as
cs_ui " +
"        ,date_dim d1 " +
"        ,date_dim d2 " +
"        ,date_dim d3 " +
"        ,store " +
"        ,customer " +
"        ,customer_demographics cd1 " +
"        ,customer_demographics cd2 " +
"        ,promotion " +
"        ,household_demographics hd1 " +
"        ,household_demographics hd2 " +
"        ,customer_address ad1 " +
"        ,customer_address ad2 " +
"        ,income_band ib1 " +
"        ,income_band ib2 " +
"        ,item " +
"  WHERE  ss_store_sk = s_store_sk AND " +
"         ss_sold_date_sk = d1.d_date_sk AND " +
"         ss_customer_sk = c_customer_sk AND " +
"         ss_cdemo_sk= cd1.cd_demo_sk AND " +
"         ss_hdemo_sk = hd1.hd_demo_sk AND " +
"         ss_addr_sk = ad1.ca_address_sk and " +
"         ss_item_sk = i_item_sk and " +
"         ss_item_sk = sr_item_sk and " +
"         ss_ticket_number = sr_ticket_number and " +
"         ss_item_sk = cs_ui.cs_item_sk and " +
"         c_current_cdemo_sk = cd2.cd_demo_sk AND " +
"         c_current_hdemo_sk = hd2.hd_demo_sk AND " +
"         c_current_addr_sk = ad2.ca_address_sk and " +
"         c_first_sales_date_sk = d2.d_date_sk and " +
"         c_first_shipto_date_sk = d3.d_date_sk and " +
"         ss_promo_sk = p_promo_sk and " +
"         hd1.hd_income_band_sk = ib1.ib_income_band_sk and " +
"         hd2.hd_income_band_sk = ib2.ib_income_band_sk and " +
"         cd1.cd_marital_status != cd2.cd_marital_status " +
"group by i_product_name " +
"       ,i_item_sk " +
"       ,s_store_name " +
"       ,s_zip " +
"       ,ad1.ca_street_number " +
"       ,ad1.ca_street_name " +
"       ,ad1.ca_city " +
"       ,ad1.ca_zip " +
"       ,ad2.ca_street_number " +
"       ,ad2.ca_street_name " +
"       ,ad2.ca_city " +
"       ,ad2.ca_zip " +
"       ,d1.d_year " +
"       ,d2.d_year " +
"       ,d3.d_year) as cs2 " +
"   " +
"where cs1.item_sk=cs2.item_sk and " +
"     cs1.syear > 2000 and " +
"     cs2.syear > 2000 and " +
"     cs2.syear != cs1.syear and " +
"     cs2.cnt <= cs1.cnt " +
" " +
"order by cs1.product_name,cs1.store_name,cs2.cnt";
```

```
// QUERY 66
final static String q66 =
"  SELECT w_warehouse_name, " +
"       w_warehouse_sq_ft, " +
"       w_city, " +
"       w_county, " +
"       w_state, " +
"       w_country, " +
"       year, " +
"       Sum(jan_sales)                     AS jan_sales, " +
"       Sum(feb_sales)                     AS feb_sales, " +
"       Sum(mar_sales)                     AS mar_sales, " +
"       Sum(apr_sales)                     AS apr_sales, " +
"       Sum(may_sales)                     AS may_sales, " +
"       Sum(jun_sales)                     AS jun_sales, " +
"       Sum(jul_sales)                     AS jul_sales, " +
"       Sum(aug_sales)                     AS aug_sales, " +
"       Sum(sep_sales)                     AS sep_sales, " +
"       Sum(oct_sales)                     AS oct_sales, " +
"       Sum(nov_sales)                     AS nov_sales, " +
"       Sum(dec_sales)                     AS dec_sales, " +
"       Sum(jan_sales / w_warehouse_sq_ft) AS jan_sales_per_sq_foot, " +
"       Sum(feb_sales / w_warehouse_sq_ft) AS feb_sales_per_sq_foot, " +
"       Sum(mar_sales / w_warehouse_sq_ft) AS mar_sales_per_sq_foot, " +
"       Sum(apr_sales / w_warehouse_sq_ft) AS apr_sales_per_sq_foot, " +
"       Sum(may_sales / w_warehouse_sq_ft) AS may_sales_per_sq_foot, " +
"       Sum(jun_sales / w_warehouse_sq_ft) AS jun_sales_per_sq_foot, " +
"       Sum(jul_sales / w_warehouse_sq_ft) AS jul_sales_per_sq_foot, " +
"       Sum(aug_sales / w_warehouse_sq_ft) AS aug_sales_per_sq_foot, " +
"       Sum(sep_sales / w_warehouse_sq_ft) AS sep_sales_per_sq_foot, " +
"       Sum(oct_sales / w_warehouse_sq_ft) AS oct_sales_per_sq_foot, " +
"       Sum(nov_sales / w_warehouse_sq_ft) AS nov_sales_per_sq_foot, " +
"       Sum(dec_sales / w_warehouse_sq_ft) AS dec_sales_per_sq_foot, " +
"       Sum(jan_net)                       AS jan_net, " +
"       Sum(feb_net)                       AS feb_net, " +
"       Sum(mar_net)                       AS mar_net, " +
"       Sum(apr_net)                       AS apr_net, " +
"       Sum(may_net)                       AS may_net, " +
"       Sum(jun_net)                       AS jun_net, " +
"       Sum(jul_net)                       AS jul_net, " +
"       Sum(aug_net)                       AS aug_net, " +
"       Sum(sep_net)                       AS sep_net, " +
"       Sum(oct_net)                       AS oct_net, " +
"       Sum(nov_net)                       AS nov_net, " +
"       Sum(dec_net)                       AS dec_net " +
" FROM   ((SELECT w_warehouse_name, " +
"                 w_warehouse_sq_ft, " +
"                 w_city, " +
"                 w_county, " +
"                 w_state, " +
"                 w_country, " +
"                 d_year   AS year, " +
"                 Sum(CASE " +
"                       WHEN d_month_name = 'January' THEN " +
"                       ws_sales_price * ws_quantity " +
"                       ELSE 0 " +
"                     end) AS jan_sales, " +
"                 Sum(CASE " +
"                       WHEN d_month_name = 'February' THEN " +
"                       ws_sales_price * ws_quantity " +
"                       ELSE 0 " +
"                     end) AS feb_sales, " +
"                 Sum(CASE " +
```

```
"                      WHEN d_month_name = 'March' THEN " +
"                      ws_sales_price * ws_quantity " +
"                      ELSE 0 " +
"                  end) AS mar_sales, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'April' THEN " +
"                      ws_sales_price * ws_quantity " +
"                      ELSE 0 " +
"                  end) AS apr_sales, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'May' THEN " +
"                      ws_sales_price * ws_quantity " +
"                      ELSE 0 " +
"                  end) AS may_sales, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'June' THEN " +
"                      ws_sales_price * ws_quantity " +
"                      ELSE 0 " +
"                  end) AS jun_sales, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'July' THEN " +
"                      ws_sales_price * ws_quantity " +
"                      ELSE 0 " +
"                  end) AS jul_sales, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'August' THEN " +
"                      ws_sales_price * ws_quantity " +
"                      ELSE 0 " +
"                  end) AS aug_sales, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'September' THEN " +
"                      ws_sales_price * ws_quantity " +
"                      ELSE 0 " +
"                  end) AS sep_sales, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'October' THEN " +
"                      ws_sales_price * ws_quantity " +
"                      ELSE 0 " +
"                  end) AS oct_sales, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'November' THEN " +
"                      ws_sales_price * ws_quantity " +
"                      ELSE 0 " +
"                  end) AS nov_sales, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'December' THEN " +
"                      ws_sales_price * ws_quantity " +
"                      ELSE 0 " +
"                  end) AS dec_sales, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'January' THEN " +
"                      ws_net_paid * ws_quantity " +
"                      ELSE 0 " +
"                  end) AS jan_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'February' THEN " +
"                      ws_net_paid * ws_quantity " +
"                      ELSE 0 " +
"                  end) AS feb_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'March' THEN ws_net_paid * ws_quantity " +
"                      ELSE 0 " +
"                  end) AS mar_net, " +
```

```
"                Sum(CASE " +
"                      WHEN d_month_name = 'April' THEN ws_net_paid * ws_quantity " +
"                      ELSE 0 " +
"                    end) AS apr_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'May' THEN ws_net_paid * ws_quantity " +
"                      ELSE 0 " +
"                    end) AS may_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'June' THEN ws_net_paid * ws_quantity " +
"                      ELSE 0 " +
"                    end) AS jun_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'July' THEN ws_net_paid * ws_quantity " +
"                      ELSE 0 " +
"                    end) AS jul_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'August' THEN " +
"                      ws_net_paid * ws_quantity " +
"                      ELSE 0 " +
"                    end) AS aug_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'September' THEN " +
"                      ws_net_paid * ws_quantity " +
"                      ELSE 0 " +
"                    end) AS sep_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'October' THEN " +
"                      ws_net_paid * ws_quantity " +
"                      ELSE 0 " +
"                    end) AS oct_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'November' THEN " +
"                      ws_net_paid * ws_quantity " +
"                      ELSE 0 " +
"                    end) AS nov_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'December' THEN " +
"                      ws_net_paid * ws_quantity " +
"                      ELSE 0 " +
"                    end) AS dec_net " +
"         FROM   web_sales, " +
"                warehouse, " +
"                date_dim, " +
"                time_dim, " +
"                ship_mode " +
"         WHERE  ws_warehouse_sk = w_warehouse_sk " +
"                AND ws_sold_date_sk = d_date_sk " +
"                AND ws_sold_time_sk = t_time_sk " +
"                AND ws_ship_mode_sk = sm_ship_mode_sk " +
"                AND d_year = 2000 " +
"         GROUP  BY w_warehouse_name, " +
"                   w_warehouse_sq_ft, " +
"                   w_city, " +
"                   w_county, " +
"                   w_state, " +
"                   w_country, " +
"                   d_year) " +
"        UNION ALL " +
"        (SELECT w_warehouse_name, " +
"                w_warehouse_sq_ft, " +
"                w_city, " +
"                w_county, " +
```

```
"                w_state, " +
"                w_country, " +
"                d_year   AS year, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'January' THEN cs_sales_price * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS jan_sales, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'February' THEN cs_sales_price * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS feb_sales, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'March' THEN cs_sales_price * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS mar_sales, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'April' THEN cs_sales_price * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS apr_sales, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'May' THEN cs_sales_price * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS may_sales, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'June' THEN cs_sales_price * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS jun_sales, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'July' THEN cs_sales_price * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS jul_sales, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'August' THEN cs_sales_price * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS aug_sales, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'September' THEN cs_sales_price * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS sep_sales, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'October' THEN cs_sales_price * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS oct_sales, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'November' THEN cs_sales_price * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS nov_sales, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'December' THEN cs_sales_price * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS dec_sales, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'January' THEN cs_net_paid * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS jan_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'February' THEN cs_net_paid * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS feb_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'March' THEN cs_net_paid * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS mar_net, " +
```

```
"                Sum(CASE " +
"                      WHEN d_month_name = 'April' THEN cs_net_paid * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS apr_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'May' THEN cs_net_paid * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS may_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'June' THEN cs_net_paid * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS jun_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'July' THEN cs_net_paid * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS jul_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'August' THEN cs_net_paid * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS aug_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'September' THEN cs_net_paid * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS sep_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'October' THEN cs_net_paid * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS oct_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'November' THEN cs_net_paid * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS nov_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name = 'December' THEN cs_net_paid * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS dec_net " +
"         FROM   catalog_sales, " +
"                warehouse, " +
"                date_dim, " +
"                time_dim, " +
"                ship_mode " +
"         WHERE  cs_warehouse_sk = w_warehouse_sk " +
"                AND cs_sold_date_sk = d_date_sk " +
"                AND cs_sold_time_sk = t_time_sk " +
"                AND cs_ship_mode_sk = sm_ship_mode_sk " +
"                AND d_year = 2000 " +
"         GROUP  BY w_warehouse_name, " +
"                   w_warehouse_sq_ft, " +
"                   w_city, " +
"                   w_county, " +
"                   w_state, " +
"                   w_country, " +
"                   d_year)) x " +
" GROUP  BY w_warehouse_name, " +
"           w_warehouse_sq_ft, " +
"           w_city, " +
"           w_county, " +
"           w_state, " +
"           w_country, " +
"           year " +
" ORDER  BY w_warehouse_name   ";

// QUERY 69_1
```

```
final static String q69_1 =
"  SELECT  " +
"       Count(*) AS cnt1, " +
"       cd_purchase_estimate, " +
"       Count(*) AS cnt2, " +
"       cd_credit_rating, " +
"       Count(*) AS cnt3 " +
" FROM   customer c, " +
"       customer_address ca, " +
"       customer_demographics " +
" WHERE  c.c_current_addr_sk = ca.ca_address_sk " +
"       AND cd_demo_sk = c.c_current_cdemo_sk " +
"       AND EXISTS (SELECT * " +
"                   FROM   store_sales, " +
"                          date_dim " +
"                   WHERE  c.c_customer_sk = ss_customer_sk " +
"                          AND ss_sold_date_sk = d_date_sk " +
"                          AND d_year > 2000 " +
"                          AND (d_month_name='January' OR d_month_name='February' OR
d_month_name='March' OR d_month_name='April' OR d_month_name='May' OR d_month_name='June')) " +
"       AND ( NOT EXISTS (SELECT * " +
"                         FROM   web_sales, " +
"                                date_dim " +
"                         WHERE  c.c_customer_sk = ws_bill_customer_sk " +
"                                AND ws_sold_date_sk = d_date_sk " +
"                                AND d_year > 2000 " +
"                                AND (d_month_name='January' OR d_month_name='February' OR
d_month_name='March' OR d_month_name='April' OR d_month_name='May' OR d_month_name='June')) " +
"             AND NOT EXISTS (SELECT * " +
"                             FROM   catalog_sales, " +
"                                    date_dim " +
"                             WHERE  c.c_customer_sk = cs_ship_customer_sk " +
"                                    AND cs_sold_date_sk = d_date_sk " +
"                                    AND d_year > 2000 " +
"                                    AND (d_month_name='January' OR d_month_name='February' OR
d_month_name='March' OR d_month_name='April' OR d_month_name='May' OR d_month_name='June')) ) " +
" GROUP  BY  " +
"          cd_purchase_estimate, " +
"          cd_credit_rating " +
" ORDER  BY  " +
"          cd_purchase_estimate, " +
"          cd_credit_rating;   ";

// QUERY 69_2
final static String q69_2 =
"  SELECT cd_gender, " +
"       cd_marital_status, " +
"       cd_education_status, " +
"       Count(*) AS cnt1, " +
"       cd_purchase_estimate, " +
"       Count(*) AS cnt2, " +
"       cd_credit_rating, " +
"       Count(*) AS cnt3 " +
" FROM   customer c, " +
"       customer_address ca, " +
"       customer_demographics " +
" WHERE  c.c_current_addr_sk = ca.ca_address_sk " +
"       AND cd_demo_sk = c.c_current_cdemo_sk " +
"       AND EXISTS (SELECT * " +
"                   FROM   store_sales, " +
"                          date_dim " +
"                   WHERE  c.c_customer_sk = ss_customer_sk " +
"                          AND ss_sold_date_sk = d_date_sk " +
```

```
"                          AND d_year > 2000 " +
"                          AND (d_month_name='January' OR d_month_name='February' OR
d_month_name='March' OR d_month_name='April' OR d_month_name='May' OR d_month_name='June')) " +
"       AND ( NOT EXISTS (SELECT * " +
"                         FROM   web_sales, " +
"                                date_dim " +
"                         WHERE  c.c_customer_sk = ws_bill_customer_sk " +
"                                AND ws_sold_date_sk = d_date_sk " +
"                                AND d_year > 2000 " +
"                                AND (d_month_name='January' OR d_month_name='February' OR
d_month_name='March' OR d_month_name='April' OR d_month_name='May' OR d_month_name='June')) " +
"             AND NOT EXISTS (SELECT * " +
"                             FROM   catalog_sales, " +
"                                    date_dim " +
"                             WHERE  c.c_customer_sk = cs_ship_customer_sk " +
"                                    AND cs_sold_date_sk = d_date_sk " +
"                                    AND d_year > 2000 " +
"                                    AND (d_month_name='January' OR d_month_name='February' OR
d_month_name='March' OR d_month_name='April' OR d_month_name='May' OR d_month_name='June')) ) " +
" GROUP  BY cd_gender, " +
"          cd_marital_status, " +
"          cd_education_status, " +
"          cd_purchase_estimate, " +
"          cd_credit_rating " +
" ORDER  BY cd_gender, " +
"          cd_marital_status, " +
"          cd_education_status, " +
"          cd_purchase_estimate, " +
"          cd_credit_rating;   ";

// QUERY 71
final static String q71 =
" select i_brand_id as brand_id, i_brand as brand,t_hour,t_minute, " +
"   sum(ext_price) as ext_price " +
" from item, (select ws_ext_sales_price as ext_price,  " +
"                         ws_sold_date_sk as sold_date_sk, " +
"                         ws_item_sk as sold_item_sk, " +
"                         ws_sold_time_sk as time_sk   " +
"                 from web_sales,date_dim " +
"                 where d_date_sk = ws_sold_date_sk " +
"                   and d_month_name = 'February' " +
"                   and d_year > 2000 " +
"                 union all " +
"                 select cs_ext_sales_price as ext_price, " +
"                         cs_sold_date_sk as sold_date_sk, " +
"                         cs_item_sk as sold_item_sk, " +
"                         cs_sold_time_sk as time_sk " +
"                 from catalog_sales,date_dim " +
"                 where d_date_sk = cs_sold_date_sk " +
"                   and d_month_name='February' " +
"                   and d_year>2000 " +
"                 union all " +
"                 select ss_ext_sales_price as ext_price, " +
"                         ss_sold_date_sk as sold_date_sk, " +
"                         ss_item_sk as sold_item_sk, " +
"                         ss_sold_time_sk as time_sk " +
"                 from store_sales,date_dim " +
"                 where d_date_sk = ss_sold_date_sk " +
"                   and d_month_name='February' " +
"                   and d_year>2000 " +
"                 ) as tmp,time_dim " +
" where " +
"   sold_item_sk = i_item_sk " +
```

```
"   and time_sk = t_time_sk " +
" group by i_brand, i_brand_id,t_hour,t_minute " +
" order by ext_price desc, i_brand_id " +
" ; ";

// QUERY 72
final static String q72 =
" SELECT i_item_desc, " +
"       w_warehouse_name, " +
"       d1.d_week_seq, " +
"       Count(CASE " +
"               WHEN p_promo_sk IS NULL THEN 1 " +
"               ELSE 0 " +
"             END) AS no_promo, " +
"       Count(CASE " +
"               WHEN p_promo_sk IS NOT NULL THEN 1 " +
"               ELSE 0 " +
"             END) AS promo, " +
"       Count(*)   AS total_cnt " +
"FROM   catalog_sales " +
"       JOIN inventory " +
"         ON ( cs_item_sk = inv_item_sk ) " +
"       JOIN warehouse " +
"         ON ( w_warehouse_sk = inv_warehouse_sk ) " +
"       JOIN item " +
"         ON ( i_item_sk = cs_item_sk ) " +
"       JOIN customer_demographics " +
"         ON ( cs_bill_cdemo_sk = cd_demo_sk ) " +
"       JOIN household_demographics " +
"         ON ( cs_bill_hdemo_sk = hd_demo_sk ) " +
"       JOIN date_dim d1 " +
"         ON ( cs_sold_date_sk = d1.d_date_sk ) " +
"       JOIN date_dim d2 " +
"         ON ( inv_date_sk = d2.d_date_sk ) " +
"       JOIN date_dim d3 " +
"         ON ( cs_ship_date_sk = d3.d_date_sk ) " +
"       LEFT OUTER JOIN promotion " +
"                    ON ( cs_promo_sk = p_promo_sk ) " +
"       LEFT OUTER JOIN catalog_returns " +
"                    ON ( cr_item_sk = cs_item_sk " +
"                         AND cr_order_number = cs_order_number ) " +
"WHERE  d3.d_date > d1.d_date " +
"       AND d1.d_year > 2000 " +
"       AND cd_marital_status = 'single' " +
"GROUP  BY i_item_desc, " +
"          w_warehouse_name, " +
"          d1.d_week_seq " +
"ORDER  BY total_cnt DESC, " +
"          i_item_desc, " +
"          w_warehouse_name, " +
"          d_week_seq;   ";


// QUERY 79
final static String q79 =
"  select " +
"  c_last_name,c_first_name,substr(s_city,1,30),ss_ticket_number,amt,profit " +
"  from " +
"   (select ss_ticket_number " +
"          ,ss_customer_sk " +
"          ,store.s_city " +
"          ,sum(ss_coupon_amt) as amt " +
"          ,sum(ss_net_profit) as profit " +
```

```
"    from store_sales,date_dim,store,household_demographics " +
"    where store_sales.ss_sold_date_sk = date_dim.d_date_sk " +
"    and store_sales.ss_store_sk   " +
"    and store_sales.ss_hdemo_sk = household_demographics.hd_demo_sk " +
"    and (household_demographics.hd_dep_count < 3  or " +
"    household_demographics.hd_vehicle_count > 2) " +
"    and date_dim.d_day_name = 'Monday' " +
"    and date_dim.d_year in (1999,2000,2001)  " +
"    and store.s_number_employees between 200 and 1000 " +
"    group by ss_ticket_number,ss_customer_sk,ss_addr_sk,store.s_city) ms,customer " +
"    where ss_customer_sk = c_customer_sk " +
"   order by c_last_name,c_first_name,substr(s_city,1,30), profit ";

// QUERY 85
final static String q85 =
"  SELECT Substr(r_reason_desc, 1, 20), " +
"       Avg(ws_quantity), " +
"       Avg(ws_quantity), " +
"       Avg(ws_quantity), " +
"       Avg(wr_refunded_cash), " +
"       Avg(wr_fee) " +
"FROM   web_sales, " +
"       web_returns, " +
"       web_page, " +
"       customer_demographics cd1, " +
"       customer_demographics cd2, " +
"       customer_address, " +
"       date_dim, " +
"       reason " +
"WHERE  ws_web_page_sk = wp_web_page_sk " +
"       AND ws_item_sk = wr_item_sk " +
"       AND ws_order_number = wr_order_number " +
"       AND ws_sold_date_sk = d_date_sk " +
"       AND d_year > 2000 " +
"       AND cd1.cd_demo_sk = wr_refunded_cdemo_sk " +
"       AND cd2.cd_demo_sk = wr_returning_cdemo_sk " +
"       AND ca_address_sk = wr_refunded_addr_sk " +
"       AND r_reason_sk = wr_reason_sk " +
"       AND ( ( cd1.cd_marital_status = 'single' " +
"               AND cd1.cd_marital_status = cd2.cd_marital_status " +
"               AND cd1.cd_education_status = 'secondary' " +
"               AND cd1.cd_education_status = cd2.cd_education_status ) " +
"              OR ( cd1.cd_marital_status = 'married' " +
"                   AND cd1.cd_marital_status = cd2.cd_marital_status " +
"                   AND cd1.cd_education_status = 'university' " +
"                   AND cd1.cd_education_status = cd2.cd_education_status) " +
"              OR ( cd1.cd_marital_status = 'divorced' " +
"                   AND cd1.cd_marital_status = cd2.cd_marital_status " +
"                   AND cd1.cd_education_status = 'postgraduate' " +
"                   AND cd1.cd_education_status = cd2.cd_education_status ) ) " +
"GROUP  BY r_reason_desc " +
"ORDER  BY Substr(r_reason_desc, 1, 20), " +
"          Avg(ws_quantity), " +
"          Avg(wr_refunded_cash), " +
"          Avg(wr_fee);   ";

// QUERY 90
final static String q90 =
" SELECT Cast(amc AS DECIMAL(15, 4)) / Cast(pmc AS DECIMAL(15, 4)) AS am_pm_ratio " +
"FROM   (SELECT Count(*) AS amc " +
"        FROM   web_sales, " +
"               household_demographics, " +
"               time_dim, " +
```

```
"               web_page " +
"        WHERE  ws_sold_time_sk = time_dim.t_time_sk " +
"               AND ws_ship_hdemo_sk = household_demographics.hd_demo_sk " +
"               AND ws_web_page_sk = web_page.wp_web_page_sk " +
"               AND time_dim.t_am_pm = 'AM' " +
"               AND household_demographics.hd_dep_count < 5 " +
"               AND web_page.wp_char_count BETWEEN 250000 AND 500000) at, " +
"       (SELECT Count(*) AS pmc " +
"        FROM   web_sales, " +
"               household_demographics, " +
"               time_dim, " +
"               web_page " +
"        WHERE  ws_sold_time_sk = time_dim.t_time_sk " +
"               AND ws_ship_hdemo_sk = household_demographics.hd_demo_sk " +
"               AND ws_web_page_sk = web_page.wp_web_page_sk " +
"               AND time_dim.t_am_pm = 'PM' " +
"               AND household_demographics.hd_dep_count < 5 " +
"               AND web_page.wp_char_count BETWEEN 250000 AND 500000) pt " +
" ORDER  BY am_pm_ratio;  ";


// QUERY 91
final static String q91 =
"  SELECT cc_call_center_id AS Call_Center, " +
"       cc_name           AS Call_Center_Name, " +
"       cc_manager        AS Manager, " +
"       Sum(cr_net_loss)  AS Returns_Loss " +
" FROM   call_center, " +
"       catalog_returns, " +
"       date_dim, " +
"       customer, " +
"       customer_address, " +
"       customer_demographics, " +
"       household_demographics " +
" WHERE  cr_call_center_sk = cc_call_center_sk " +
"       AND cr_returned_date_sk = d_date_sk " +
"       AND cr_returning_customer_sk = c_customer_sk " +
"       AND cd_demo_sk = c_current_cdemo_sk " +
"       AND hd_demo_sk = c_current_hdemo_sk " +
"       AND ca_address_sk = c_current_addr_sk " +
"       AND d_year > 2000 " +
"       AND d_month_name IN ('January','February','March','April','May','June') " +
"       AND ( ( cd_marital_status = 'single' " +
"               AND cd_education_status = 'secondary' ) " +
"              OR ( cd_marital_status = 'married' " +
"                   AND cd_education_status = 'postgraduate' ) ) " +
"       AND ca_gmt_offset > 500 " +
" GROUP  BY cc_call_center_id, " +
"          cc_name, " +
"          cc_manager, " +
"          cd_marital_status, " +
"          cd_education_status " +
" ORDER  BY Sum(cr_net_loss) DESC;   ";


//BOOL QUERIES.....................................................................


// BOOL QUERY 2
final static String bq2 =
" select d_week_seq1,round(sun_sales1/sun_sales2,2) " +
"  ,round(mon_sales1/mon_sales2,2),round(tue_sales1/tue_sales2,2) " +
"     ,round(wed_sales1/wed_sales2,2),round(thu_sales1/thu_sales2,2) " +
```

```
"  ,round(fri_sales1/fri_sales2,2),round(sat_sales1/sat_sales2,2) " +
"   " +
"from  " +
"  (select wswscs.d_week_seq d_week_seq1 " +
"         ,sun_sales sun_sales1 " +
"         ,mon_sales mon_sales1 " +
"         ,tue_sales tue_sales1 " +
"         ,wed_sales wed_sales1 " +
"         ,thu_sales thu_sales1 " +
"         ,fri_sales fri_sales1 " +
"         ,sat_sales sat_sales1 " +
"  from (select d_week_seq, " +
"         sum(case when (d_day_name_Sunday=1) then sales_price else null end) as sun_sales, " +
"         sum(case when (d_day_name_Monday=1) then sales_price else null end) as mon_sales, " +
"         sum(case when (d_day_name_Tuesday=1) then sales_price else  null end) as tue_sales, " +
"         sum(case when (d_day_name_Wednesday=1) then sales_price else null end) as wed_sales, " +
"         sum(case when (d_day_name_Thursday=1) then sales_price else null end) as thu_sales, " +
"         sum(case when (d_day_name_Friday=1) then sales_price else null end) as fri_sales, " +
"         sum(case when (d_day_name_Saturday=1) then sales_price else null end) as sat_sales " +
" from (select sold_date_sk " +
"         ,sales_price " +
"  from (select ws_sold_date_sk sold_date_sk " +
"               ,ws_ext_sales_price sales_price " +
"         from web_sales) x " +
"         union all " +
"        (select cs_sold_date_sk sold_date_sk " +
"               ,cs_ext_sales_price sales_price " +
"         from catalog_sales)) as wscs " +
"     ,date_dim " +
" where d_date_sk = sold_date_sk " +
" group by d_week_seq) as wswscs,date_dim  " +
"  where date_dim.d_week_seq = wswscs.d_week_seq  " +
"  and d_year = 2000) y,  " +
"   " +
"  (select wswscs.d_week_seq d_week_seq2 " +
"         ,sun_sales sun_sales2 " +
"         ,mon_sales mon_sales2 " +
"         ,tue_sales tue_sales2 " +
"         ,wed_sales wed_sales2 " +
"         ,thu_sales thu_sales2 " +
"         ,fri_sales fri_sales2 " +
"         ,sat_sales sat_sales2 " +
"  from (select d_week_seq, " +
"         sum(case when (d_day_name_Sunday=1) then sales_price else null end) as sun_sales, " +
"         sum(case when (d_day_name_Monday=1) then sales_price else null end) as mon_sales, " +
"         sum(case when (d_day_name_Tuesday=1) then sales_price else  null end) as tue_sales, " +
"         sum(case when (d_day_name_Wednesday=1) then sales_price else null end) as wed_sales, " +
"         sum(case when (d_day_name_Thursday=1) then sales_price else null end) as thu_sales, " +
"         sum(case when (d_day_name_Friday=1) then sales_price else null end) as fri_sales, " +
"         sum(case when (d_day_name_Saturday=1) then sales_price else null end) as sat_sales " +
" from (select sold_date_sk " +
"         ,sales_price " +
"  from (select ws_sold_date_sk sold_date_sk " +
"               ,ws_ext_sales_price sales_price " +
"         from web_sales) x " +
"         union all " +
"        (select cs_sold_date_sk sold_date_sk " +
"               ,cs_ext_sales_price sales_price " +
"         from catalog_sales)) as wscs " +
"     ,date_dim " +
" where d_date_sk = sold_date_sk " +
" group by d_week_seq) as wswscs,date_dim  " +
"  where date_dim.d_week_seq = wswscs.d_week_seq  " +
```

```
"  and d_year = 2001) z " +
"   " +
"where d_week_seq1 < d_week_seq2  " +
"order by d_week_seq1; " ;

// BOOL QUERY 6
final static String bq6 =
" select a.ca_state as state, count(*) as cnt " +
" from customer_address a,customer c,store_sales s,date_dim d,item i " +
" where a.ca_address_sk = c.c_current_addr_sk " +
" and c.c_customer_sk = s.ss_customer_sk and s.ss_sold_date_sk = d.d_date_sk " +
" and s.ss_item_sk = i.i_item_sk " +
" and d.d_month_seq = (select distinct (d_month_seq) " +
" from date_dim where d_year = 2000 " +
" and d_month_name_November = 1 limit 1) " +
" and i.i_current_price > 1.2 * " +
" (select avg(j.i_current_price) from item j " +
" where j.i_category = i.i_category) group by a.ca_state " +
" having count(*) >= 10 order by cnt ";


// BOOL QUERY 7
final static String bq7 =
" select  i_item_id, " +
" avg(ss_quantity) as agg1, " +
" avg(ss_list_price) as agg2, " +
" avg(ss_coupon_amt) as agg3, " +
" avg(ss_sales_price) as agg4 " +
" from " +
" store_sales, customer_demographics, date_dim, item, promotion " +
" where " +
" ss_sold_date_sk = d_date_sk and " +
" ss_item_sk = i_item_sk and " +
" ss_cdemo_sk = cd_demo_sk and " +
" ss_promo_sk = p_promo_sk and " +
" cd_gender_female=1 and  " +
" cd_marital_status_married=1 and " +
" cd_education_status_university=1 and " +
" (p_channel_email_no=1 or p_channel_event_no=1) and " +
" d_year >  2000 " +
" group by i_item_id " +
" order by i_item_id ";

// BOOL QUERY 10_1
final static String bq10_1 =
" SELECT cd_purchase_estimate," +
"       cd_credit_rating," +
"       cd_dep_count," +
"       cd_dep_employed_count," +
"       cd_dep_college_count" +
" FROM   customer c," +
"       customer_address ca," +
"       customer_demographics" +
" WHERE  c.c_current_addr_sk = ca.ca_address_sk" +
"       AND cd_demo_sk = c.c_current_cdemo_sk" +
"       AND EXISTS (SELECT *" +
"                   FROM   store_sales," +
"                          date_dim" +
"                   WHERE  c.c_customer_sk = ss_customer_sk" +
"                          AND ss_sold_date_sk = d_date_sk" +
"                          AND d_year > 2000" +
"                          AND ( d_month_name_March=1" +
"                                OR d_month_name_April=1" +
```

```
"                                OR d_month_name_May=1" +
"                                OR d_month_name_June=1" +
"                                OR d_month_name_July=1" +
"                                OR d_month_name_August=1 ))" +
"       AND ( EXISTS (SELECT *" +
"                     FROM   web_sales," +
"                            date_dim" +
"                     WHERE  c.c_customer_sk = ws_bill_customer_sk" +
"                            AND ws_sold_date_sk = d_date_sk" +
"                            AND d_year > 2000" +
"                            AND (  d_month_name_March=1" +
"                                 OR d_month_name_April=1" +
"                                 OR d_month_name_May=1" +
"                                 OR d_month_name_June=1" +
"                                 OR d_month_name_July=1" +
"                                 OR d_month_name_August=1 ))" +
"              OR EXISTS (SELECT *" +
"                         FROM   catalog_sales," +
"                                date_dim" +
"                         WHERE  c.c_customer_sk = cs_ship_customer_sk" +
"                                AND cs_sold_date_sk = d_date_sk" +
"                                AND d_year > 2000" +
"                                AND (  d_month_name_March=1" +
"                                 OR d_month_name_April=1" +
"                                 OR d_month_name_May=1" +
"                                 OR d_month_name_June=1" +
"                                 OR d_month_name_July=1" +
"                                 OR d_month_name_August=1  )) )" +
" GROUP  BY cd_purchase_estimate," +
"          cd_credit_rating," +
"          cd_dep_count," +
"          cd_dep_employed_count," +
"          cd_dep_college_count" +
" ORDER  BY cd_purchase_estimate," +
"          cd_credit_rating," +
"          cd_dep_count," +
"          cd_dep_employed_count," +
"          cd_dep_college_count;  ";

// BOOL QUERY 10_2
final static String bq10_2 =
" SELECT cd_gender_male,cd_gender_female," +
"       cd_marital_status_single,cd_marital_status_married,cd_marital_status_divorced," +
"
cd_education_status_primary,cd_education_status_secondary,cd_education_status_university,cd_educa
tion_status_postgraduate," +
"       cd_purchase_estimate," +
"       cd_credit_rating," +
"       cd_dep_count," +
"       cd_dep_employed_count," +
"       cd_dep_college_count" +
" FROM   customer c," +
"       customer_address ca," +
"       customer_demographics" +
" WHERE  c.c_current_addr_sk = ca.ca_address_sk" +
"       AND cd_demo_sk = c.c_current_cdemo_sk" +
"       AND EXISTS (SELECT *" +
"                   FROM   store_sales," +
"                          date_dim" +
"                   WHERE  c.c_customer_sk = ss_customer_sk" +
"                          AND ss_sold_date_sk = d_date_sk" +
"                          AND d_year > 2000" +
"                          AND ( d_month_name_March=1" +
```

```
"                                OR d_month_name_April=1" +
"                                OR d_month_name_May=1" +
"                                OR d_month_name_June=1" +
"                                OR d_month_name_July=1" +
"                                OR d_month_name_August=1   ))" +
"       AND ( EXISTS (SELECT *" +
"                     FROM   web_sales," +
"                            date_dim" +
"                     WHERE  c.c_customer_sk = ws_bill_customer_sk" +
"                            AND ws_sold_date_sk = d_date_sk" +
"                            AND d_year > 2000" +
"                            AND ( d_month_name_March=1" +
"                                 OR d_month_name_April=1" +
"                                 OR d_month_name_May=1" +
"                                 OR d_month_name_June=1" +
"                                 OR d_month_name_July=1" +
"                                 OR d_month_name_August=1   ))" +
"              OR EXISTS (SELECT *" +
"                         FROM   catalog_sales," +
"                                date_dim" +
"                         WHERE  c.c_customer_sk = cs_ship_customer_sk" +
"                                AND cs_sold_date_sk = d_date_sk" +
"                                AND d_year > 2000" +
"                                AND ( d_month_name_March=1" +
"                                 OR d_month_name_April=1" +
"                                 OR d_month_name_May=1" +
"                                 OR d_month_name_June=1" +
"                                 OR d_month_name_July=1" +
"                                 OR d_month_name_August=1   )) )" +
" GROUP  BY cd_gender_male,cd_gender_female," +
"          cd_marital_status_single,cd_marital_status_married,cd_marital_status_divorced," +
"
cd_education_status_primary,cd_education_status_secondary,cd_education_status_university,cd_educa
tion_status_postgraduate," +
"          cd_purchase_estimate," +
"          cd_credit_rating," +
"          cd_dep_count," +
"          cd_dep_employed_count," +
"          cd_dep_college_count" +
" ORDER  BY cd_gender_male,cd_gender_female," +
"          cd_marital_status_single,cd_marital_status_married,cd_marital_status_divorced," +
"
cd_education_status_primary,cd_education_status_secondary,cd_education_status_university,cd_educa
tion_status_postgraduate," +
"          cd_purchase_estimate," +
"          cd_credit_rating," +
"          cd_dep_count," +
"          cd_dep_employed_count," +
"          cd_dep_college_count;  ";


// BOOL QUERY 11
final static String bq11 =
"  SELECT c_customer_id customer_id,c_first_name customer_first_name, " +
"       c_last_name customer_last_name, " +
"       c_preferred_cust_flag_yes customer_preferred_cust_flag_yes, " +
"       c_preferred_cust_flag_no customer_preferred_cust_flag_no, " +
"       c_birth_country customer_birth_country,c_login customer_login, " +
"       c_email_address customer_email_address,d_year dyear,Sum( " +
"       ss_ext_list_price - ss_ext_discount_amt) AS year_total,'s' sale_type " +
"FROM   customer,store_sales,date_dim " +
"WHERE  c_customer_sk = ss_customer_sk " +
"       AND ss_sold_date_sk = d_date_sk " +
```

```
"GROUP  BY
c_customer_id,c_first_name,c_last_name,d_year,c_preferred_cust_flag_yes,c_preferred_cust_flag_no,
" +
"c_birth_country,c_login,c_email_address,d_year " +
"UNION ALL " +
"SELECT c_customer_id customer_id,c_first_name customer_first_name, " +
"       c_last_name customer_last_name, " +
"       c_preferred_cust_flag_yes,c_preferred_cust_flag_no customer_preferred_cust_flag, " +
"       c_birth_country customer_birth_country,c_login customer_login, " +
"       c_email_address customer_email_address,d_year dyear,Sum( " +
"       ws_ext_list_price - ws_ext_discount_amt) AS year_total,'w' sale_type " +
"FROM   customer,web_sales,date_dim " +
"WHERE  c_customer_sk = ws_bill_customer_sk " +
"       AND ws_sold_date_sk = d_date_sk " +
"GROUP  BY
c_customer_id,c_first_name,c_last_name,c_preferred_cust_flag_yes,c_preferred_cust_flag_no, " +
"c_birth_country,c_login,c_email_address,d_year   ";

// BOOL QUERY 13
final static String bq13 =
"  SELECT Avg(ss_quantity),Avg(ss_ext_sales_price),Avg(ss_ext_wholesale_cost),Sum(" +
"       ss_ext_wholesale_cost)" +
" FROM   store_sales,store,customer_demographics,household_demographics," +
"       customer_address," +
"       date_dim" +
" WHERE  s_store_sk = ss_store_sk" +
"       AND ss_sold_date_sk = d_date_sk" +
"       AND d_year = 2001" +
"       AND ( ( ss_hdemo_sk = hd_demo_sk" +
"               AND cd_demo_sk = ss_cdemo_sk" +
"               AND cd_marital_status_married=1" +
"               AND cd_education_status_university=1" +
"               AND ss_sales_price BETWEEN 10000.00 AND 50000.00" +
"               AND hd_dep_count > 5 )" +
"             OR ( ss_hdemo_sk = hd_demo_sk" +
"                  AND cd_demo_sk = ss_cdemo_sk" +
"                  AND cd_marital_status_divorced=1" +
"                  AND cd_education_status_secondary=1" +
"                  AND ss_sales_price BETWEEN 20000.00 AND 60000.00" +
"                  AND hd_dep_count < 5 )" +
"             OR ( ss_hdemo_sk = hd_demo_sk" +
"                  AND cd_demo_sk = ss_cdemo_sk" +
"                  AND cd_marital_status_single=1" +
"                  AND cd_education_status_postgraduate=1" +
"                  AND ss_sales_price BETWEEN 30000.00 AND 70000.00" +
"                  AND hd_dep_count < 7 ) );  ";

// BOOL QUERY 15
final static String bq15 =
" SELECT ca_zip,Sum(cs_sales_price) " +
" FROM   catalog_sales,customer,customer_address,date_dim " +
" WHERE  cs_bill_customer_sk = c_customer_sk " +
"       AND c_current_addr_sk = ca_address_sk " +
"       AND cs_sales_price > 500 " +
"       AND cs_sold_date_sk = d_date_sk " +
"       AND d_quarter_name_first=1 " +
"       AND d_year > 2000 " +
" GROUP  BY ca_zip " +
" ORDER  BY ca_zip;  ";


// BOOL QUERY 17
final static String bq17 =
```

```
" SELECT i_item_id,i_item_desc,s_state,  " +
"       Count(ss_quantity) AS store_sales_quantitycount,  " +
"       Avg(ss_quantity) AS store_sales_quantityave,  " +
"       Stddev_samp(ss_quantity) AS store_sales_quantitystdev,  " +
"       Stddev_samp(ss_quantity) / Avg(ss_quantity) AS store_sales_quantitycov,  " +
"       Count(sr_return_quantity) AS as_store_returns_quantitycount,Avg(  " +
"       sr_return_quantity) AS as_store_returns_quantityave,  " +
"       Stddev_samp(sr_return_quantity) AS as_store_returns_quantitystdev,  " +
"              Stddev_samp(sr_return_quantity) / Avg(sr_return_quantity) AS  " +
"              store_returns_quantitycov,  " +
"       Count(cs_quantity) AS catalog_sales_quantitycount,  " +
"       Avg(cs_quantity) AS catalog_sales_quantityave,  " +
"       Stddev_samp(cs_quantity) / Avg(cs_quantity) AS  " +
"       catalog_sales_quantitystdev,Stddev_samp(cs_quantity) / Avg(cs_quantity)  " +
"                                  AS catalog_sales_quantitycov  " +
"FROM   store_sales,store_returns,catalog_sales,date_dim d1,date_dim d2,date_dim  " +
"       d3,  " +
"       store,item  " +
"WHERE  d1.d_quarter_name_first=1  " +
"       AND d1.d_date_sk = ss_sold_date_sk  " +
"       AND i_item_sk = ss_item_sk  " +
"       AND s_store_sk = ss_store_sk  " +
"       AND ss_customer_sk = sr_customer_sk  " +
"       AND ss_item_sk = sr_item_sk  " +
"       AND ss_ticket_number = sr_ticket_number  " +
"       AND sr_returned_date_sk = d2.d_date_sk  " +
"       AND (d2.d_quarter_name_first=1 OR d2.d_quarter_name_second=1 OR d2.d_quarter_name_third=1
)  " +
"       AND sr_customer_sk = cs_bill_customer_sk  " +
"       AND sr_item_sk = cs_item_sk  " +
"       AND cs_sold_date_sk = d3.d_date_sk  " +
"       AND (d3.d_quarter_name_first=1 OR d3.d_quarter_name_second=1 OR d3.d_quarter_name_third=1
)  " +
"GROUP  BY i_item_id,i_item_desc,s_state  " +
"ORDER  BY i_item_id,i_item_desc,s_state;  ";

// BOOL QUERY 19
final static String bq19 =
" SELECT i_brand_id AS brand_id,i_brand AS brand,i_manufact_id,i_manufact,Sum( " +
"       ss_ext_sales_price) AS ext_price " +
" FROM   date_dim,store_sales,item,customer,customer_address,store " +
" WHERE  d_date_sk = ss_sold_date_sk " +
"       AND ss_item_sk = i_item_sk " +
"       AND d_month_name_May=1 " +
"       AND d_year > 2000 " +
"       AND ss_customer_sk = c_customer_sk " +
"       AND c_current_addr_sk = ca_address_sk " +
"       AND ss_store_sk = s_store_sk " +
" GROUP  BY i_brand,i_brand_id,i_manufact_id,i_manufact " +
" ORDER  BY ext_price DESC,i_brand,i_brand_id,i_manufact_id,i_manufact  ";


// BOOL QUERY 26
final static String bq26 =
" SELECT i_item_id,Avg(cs_quantity) AS agg1,Avg(cs_list_price) AS agg2,  " +
"       Avg(cs_coupon_amt) AS agg3,Avg(cs_sales_price) AS agg4  " +
" FROM   catalog_sales,customer_demographics,date_dim,item,promotion  " +
" WHERE  cs_sold_date_sk = d_date_sk  " +
"       AND cs_item_sk = i_item_sk  " +
"       AND cs_bill_cdemo_sk = cd_demo_sk  " +
"       AND cs_promo_sk = p_promo_sk  " +
"       AND cd_gender_female=1  " +
"       AND cd_marital_status_single=1  " +
```

```
"       AND cd_education_status_university=1  " +
"       AND ( p_channel_email_no=1  " +
"             OR p_channel_event_no=1 )  " +
"       AND d_year < 2000  " +
" GROUP  BY i_item_id  " +
" ORDER  BY i_item_id;  ";

// BOOL QUERY 30
final static String bq30 =
" select  c_customer_id,c_salutation,c_first_name,c_last_name,c_preferred_cust_flag_yes,  " +
"              c_preferred_cust_flag_no,c_birth_day,c_birth_month,c_birth_year,c_birth_country,
" +
"        c_login,c_email_address,c_last_review_date_sk,ctr_total_return  " +
"           " +
" from   " +
"       (select wr_returning_customer_sk as ctr_customer_sk  " +
"        ,ca_state as ctr_state,   " +
"   sum(wr_return_amt) as ctr_total_return  " +
" from web_returns  " +
"     ,date_dim  " +
"     ,customer_address  " +
" where wr_returned_date_sk = d_date_sk   " +
"   and d_year > 2000  " +
"   and wr_returning_addr_sk = ca_address_sk   " +
" group by wr_returning_customer_sk  " +
"         ,ca_state) as ctr1,customer_address,customer  " +
"               " +
" where ctr1.ctr_total_return > (select avg(ctr_total_return)*1.2  " +
"         from (select wr_returning_customer_sk as ctr_customer_sk  " +
"        ,ca_state as ctr_state,   " +
"   sum(wr_return_amt) as ctr_total_return  " +
" from web_returns  " +
"     ,date_dim  " +
"     ,customer_address  " +
" where wr_returned_date_sk = d_date_sk   " +
"   and d_year > 2000  " +
"   and wr_returning_addr_sk = ca_address_sk   " +
" group by wr_returning_customer_sk  " +
"         ,ca_state) as ctr2   " +
"                       where ctr1.ctr_state = ctr2.ctr_state)  " +
"       and ca_address_sk = c_current_addr_sk  " +
"       and ctr1.ctr_customer_sk = c_customer_sk  " +
" order by c_customer_id,c_salutation,c_first_name,c_last_name,c_preferred_cust_flag_yes,  " +
"                              c_preferred_cust_flag_yes,c_birth_day,c_birth_month,c_birth_year,
" +
"                c_birth_country,c_login,c_email_address,c_last_review_date_sk,ctr_total_return";

// BOOL QUERY 34
final static String bq34 =
" SELECT c_last_name,c_first_name,c_salutation,c_preferred_cust_flag_yes,  " +
"        c_preferred_cust_flag_no,ss_ticket_number,cnt  " +
"FROM   (SELECT ss_ticket_number,ss_customer_sk,Count(*) AS cnt  " +
"        FROM   store_sales,date_dim,store,household_demographics  " +
"        WHERE  store_sales.ss_sold_date_sk = date_dim.d_date_sk  " +
"               AND store_sales.ss_store_sk = store.s_store_sk  " +
"               AND store_sales.ss_hdemo_sk = household_demographics.hd_demo_sk  " +
"               AND ( date_dim.d_day_name_Monday=1 OR date_dim.d_day_name_Wednesday=1  " +
"                      OR date_dim.d_day_name_Friday=1 )  " +
"               AND household_demographics.hd_vehicle_count > 0  " +
"               AND ( CASE  " +
"                       WHEN household_demographics.hd_vehicle_count > 0 THEN  " +
"                       household_demographics.hd_dep_count /  " +
"                       household_demographics.hd_vehicle_count  " +
```

```
"                       ELSE NULL  " +
"                     end ) > 1.2  " +
"               AND date_dim.d_year IN ( 1999, 2000, 2001 )  " +
"        GROUP  BY ss_ticket_number,ss_customer_sk) dn,customer  " +
"WHERE  ss_customer_sk = c_customer_sk  " +
"       AND cnt BETWEEN 0 AND 20  " +
"ORDER  BY c_last_name,c_first_name,c_salutation,c_preferred_cust_flag_yes,  " +
"                c_preferred_cust_flag_no    DESC  ";

// BOOL QUERY 39A
final static String bq39A =
"  select
inv1.w_warehouse_sk,inv1.i_item_sk,inv1.d_month_name_January,inv1.d_month_name_February,inv1.d_mo
nth_name_March,inv1.d_month_name_April,inv1.d_month_name_May,inv1.d_month_name_June,inv1.d_month_
name_July,inv1.d_month_name_August,inv1.d_month_name_September,inv1.d_month_name_October,inv1.d_m
onth_name_November,inv1.d_month_name_December,inv1.mean, inv1.cov    " +
"
,inv2.w_warehouse_sk,inv2.i_item_sk,inv2.d_month_name_January,inv2.d_month_name_February,inv2.d_m
onth_name_March,inv2.d_month_name_April,inv2.d_month_name_May,inv2.d_month_name_June,inv2.d_month
_name_July,inv2.d_month_name_August,inv2.d_month_name_September,inv2.d_month_name_October,inv2.d_
month_name_November,inv2.d_month_name_December,inv2.mean, inv2.cov    " +
"                                    " +
"                              from     " +
"                               (select
w_warehouse_name,w_warehouse_sk,i_item_sk,d_month_name_January,d_month_name_February,d_month_name
_March,d_month_name_April,d_month_name_May,d_month_name_June,d_month_name_July,d_month_name_Augus
t,d_month_name_September,d_month_name_October,d_month_name_November,d_month_name_December    " +
"                                     ,stdev,mean, case mean when 0 then null else stdev/mean end
cov    " +
"                               from(select
w_warehouse_name,w_warehouse_sk,i_item_sk,d_month_name_January,d_month_name_February,d_month_name
_March,d_month_name_April,d_month_name_May,d_month_name_June,d_month_name_July,d_month_name_Augus
t,d_month_name_September,d_month_name_October,d_month_name_November,d_month_name_December    " +
"                                          ,stddev_samp(inv_quantity_on_hand)
stdev,avg(inv_quantity_on_hand) as mean    " +
"                                     from inventory    " +
"                                         ,item    " +
"                                         ,warehouse    " +
"                                         ,date_dim    " +
"                                     where inv_item_sk = i_item_sk    " +
"                                       and inv_warehouse_sk = w_warehouse_sk    " +
"                                       and inv_date_sk = d_date_sk    " +
"                                       and d_year > 2000    " +
"                                     group by
w_warehouse_name,w_warehouse_sk,i_item_sk,d_month_name_January,d_month_name_February,d_month_name
_March,d_month_name_April,d_month_name_May,d_month_name_June,d_month_name_July,d_month_name_Augus
t,d_month_name_September,d_month_name_October,d_month_name_November,d_month_name_December) foo
" +
"                              where case mean when 0 then 0 else stdev/mean end > 1) as inv1,
" +
"                                  " +
"                              (select
w_warehouse_name,w_warehouse_sk,i_item_sk,d_month_name_January,d_month_name_February,d_month_name
_March,d_month_name_April,d_month_name_May,d_month_name_June,d_month_name_July,d_month_name_Augus
t,d_month_name_September,d_month_name_October,d_month_name_November,d_month_name_December    " +
"                                     ,stdev,mean, case mean when 0 then null else stdev/mean end
cov    " +
"                               from(select
w_warehouse_name,w_warehouse_sk,i_item_sk,d_month_name_January,d_month_name_February,d_month_name
_March,d_month_name_April,d_month_name_May,d_month_name_June,d_month_name_July,d_month_name_Augus
t,d_month_name_September,d_month_name_October,d_month_name_November,d_month_name_December    " +
"                                          ,stddev_samp(inv_quantity_on_hand)
stdev,avg(inv_quantity_on_hand) as mean    " +
```

```
"                                  from inventory    " +
"                                      ,item    " +
"                                      ,warehouse    " +
"                                      ,date_dim    " +
"                                  where inv_item_sk = i_item_sk    " +
"                                    and inv_warehouse_sk = w_warehouse_sk    " +
"                                    and inv_date_sk = d_date_sk    " +
"                                    and d_year > 2000     " +
"                                  group by
w_warehouse_name,w_warehouse_sk,i_item_sk,d_month_name_January,d_month_name_February,d_month_name
_March,d_month_name_April,d_month_name_May,d_month_name_June,d_month_name_July,d_month_name_Augus
t,d_month_name_September,d_month_name_October,d_month_name_November,d_month_name_December) foo
" +
"                               where case mean when 0 then 0 else stdev/mean end > 1) as inv2
" +
"                                   " +
"                               where inv1.i_item_sk = inv2.i_item_sk    " +
"                                and inv1.w_warehouse_sk =  inv2.w_warehouse_sk    " +
"                                and inv1 .d_month_name_January =1   " +
"                                and inv2 .d_month_name_February  =1  " +
"                              order by
inv1.w_warehouse_sk,inv1.i_item_sk,inv1.d_month_name_January,inv1.d_month_name_February,inv1.d_mo
nth_name_March,inv1.d_month_name_April,inv1.d_month_name_May,inv1.d_month_name_June,inv1.d_month_
name_July,inv1.d_month_name_August,inv1.d_month_name_September,inv1.d_month_name_October,inv1.d_m
onth_name_November,inv1.d_month_name_December,inv1.mean,inv1.cov    " +
"
,inv2.d_month_name_January,inv2.d_month_name_February,inv2.d_month_name_March,inv2.d_month_name_A
pril,inv2.d_month_name_May,inv2.d_month_name_June,inv2.d_month_name_July,inv2.d_month_name_August
,inv2.d_month_name_September,inv2.d_month_name_October,inv2.d_month_name_November,inv2.d_month_na
me_December,inv2.mean, inv2.cov ;";

// BOOL QUERY 39B
final static String bq39B =
"select
inv1.w_warehouse_sk,inv1.i_item_sk,inv1.d_month_name_January,inv1.d_month_name_February,inv1.d_mo
nth_name_March,inv1.d_month_name_April,inv1.d_month_name_May,inv1.d_month_name_June,inv1.d_month_
name_July,inv1.d_month_name_August,inv1.d_month_name_September,inv1.d_month_name_October,inv1.d_m
onth_name_November,inv1.d_month_name_December,inv1.mean, inv1.cov  " +
"
,inv2.w_warehouse_sk,inv2.i_item_sk,inv2.d_month_name_January,inv2.d_month_name_February,inv2.d_m
onth_name_March,inv2.d_month_name_April,inv2.d_month_name_May,inv2.d_month_name_June,inv2.d_month
_name_July,inv2.d_month_name_August,inv2.d_month_name_September,inv2.d_month_name_October,inv2.d_
month_name_November,inv2.d_month_name_December,inv2.mean, inv2.cov  " +
"                " +
"from   " +
"              (select
w_warehouse_name,w_warehouse_sk,i_item_sk,d_month_name_January,d_month_name_February,d_month_name
_March,d_month_name_April,d_month_name_May,d_month_name_June,d_month_name_July,d_month_name_Augus
t,d_month_name_September,d_month_name_October,d_month_name_November,d_month_name_December  " +
"       ,stdev,mean, case mean when 0 then null else stdev/mean end cov  " +
" from(select
w_warehouse_name,w_warehouse_sk,i_item_sk,d_month_name_January,d_month_name_February,d_month_name
_March,d_month_name_April,d_month_name_May,d_month_name_June,d_month_name_July,d_month_name_Augus
t,d_month_name_September,d_month_name_October,d_month_name_November,d_month_name_December  " +
"            ,stddev_samp(inv_quantity_on_hand) stdev,avg(inv_quantity_on_hand) as mean  " +
"      from inventory  " +
"          ,item  " +
"          ,warehouse  " +
"          ,date_dim  " +
"      where inv_item_sk = i_item_sk  " +
"        and inv_warehouse_sk = w_warehouse_sk  " +
"        and inv_date_sk = d_date_sk  " +
"        and d_year > 2000  " +
```

```
"      group by
w_warehouse_name,w_warehouse_sk,i_item_sk,d_month_name_January,d_month_name_February,d_month_name
_March,d_month_name_April,d_month_name_May,d_month_name_June,d_month_name_July,d_month_name_Augus
t,d_month_name_September,d_month_name_October,d_month_name_November,d_month_name_December) foo  "
+
" where case mean when 0 then 0 else stdev/mean end > 1) as inv1,  " +
"              (select
w_warehouse_name,w_warehouse_sk,i_item_sk,d_month_name_January,d_month_name_February,d_month_name
_March,d_month_name_April,d_month_name_May,d_month_name_June,d_month_name_July,d_month_name_Augus
t,d_month_name_September,d_month_name_October,d_month_name_November,d_month_name_December  " +
"       ,stdev,mean, case mean when 0 then null else stdev/mean end cov  " +
" from(select
w_warehouse_name,w_warehouse_sk,i_item_sk,d_month_name_January,d_month_name_February,d_month_name
_March,d_month_name_April,d_month_name_May,d_month_name_June,d_month_name_July,d_month_name_Augus
t,d_month_name_September,d_month_name_October,d_month_name_November,d_month_name_December  " +
"            ,stddev_samp(inv_quantity_on_hand) stdev,avg(inv_quantity_on_hand) as mean  " +
"      from inventory  " +
"          ,item  " +
"          ,warehouse  " +
"          ,date_dim  " +
"      where inv_item_sk = i_item_sk  " +
"        and inv_warehouse_sk = w_warehouse_sk  " +
"        and inv_date_sk = d_date_sk  " +
"        and d_year > 2000  " +
"      group by
w_warehouse_name,w_warehouse_sk,i_item_sk,d_month_name_January,d_month_name_February,d_month_name
_March,d_month_name_April,d_month_name_May,d_month_name_June,d_month_name_July,d_month_name_Augus
t,d_month_name_September,d_month_name_October,d_month_name_November,d_month_name_December) foo  "
+
" where case mean when 0 then 0 else stdev/mean end > 1) as inv2  " +
"  " +
"where inv1.i_item_sk = inv2.i_item_sk  " +
"  and inv1.w_warehouse_sk =  inv2.w_warehouse_sk  " +
"  and inv1 .d_month_name_October=1  " +
"  and inv2 .d_month_name_September=1  " +
"  and inv1.cov > 1.5  " +
"order by
inv1.w_warehouse_sk,inv1.i_item_sk,inv1.d_month_name_January,inv1.d_month_name_February,inv1.d_mo
nth_name_March,inv1.d_month_name_April,inv1.d_month_name_May,inv1.d_month_name_June,inv1.d_month_
name_July,inv1.d_month_name_August,inv1.d_month_name_September,inv1.d_month_name_October,inv1.d_m
onth_name_November,inv1.d_month_name_December,inv1.mean,inv1.cov  " +
"
,inv2.d_month_name_January,inv2.d_month_name_February,inv2.d_month_name_March,inv2.d_month_name_A
pril,inv2.d_month_name_May,inv2.d_month_name_June,inv2.d_month_name_July,inv2.d_month_name_August
,inv2.d_month_name_September,inv2.d_month_name_October,inv2.d_month_name_November,inv2.d_month_na
me_December,inv2.mean, inv2.cov";


// BOOL QUERY 42
final static String bq42 =
"  SELECT dt.d_year, " +
"       item.i_category_id, " +
"       item.i_category, " +
"       Sum(ss_ext_sales_price) " +
" FROM   date_dim dt, " +
"       store_sales, " +
"       item " +
" WHERE  dt.d_date_sk = store_sales.ss_sold_date_sk " +
"       AND store_sales.ss_item_sk = item.i_item_sk " +
"       AND dt.d_month_name_April=1 " +
"       AND dt.d_year > 2000 " +
" GROUP  BY dt.d_year, " +
"          item.i_category_id, " +
```

```
"          item.i_category " +
" ORDER  BY Sum(ss_ext_sales_price) DESC, " +
"          dt.d_year, " +
"          item.i_category_id, " +
"          item.i_category   ";



// BOOL QUERY 43
final static String bq43 =
"  SELECT s_store_name, " +
"       s_store_id, " +
"       Sum(CASE " +
"             WHEN ( d_day_name_Sunday=1 ) THEN ss_sales_price " +
"             ELSE NULL " +
"           end) AS sun_sales, " +
"       Sum(CASE " +
"             WHEN ( d_day_name_Monday=1 ) THEN ss_sales_price " +
"             ELSE NULL " +
"           end) AS mon_sales, " +
"       Sum(CASE " +
"             WHEN ( d_day_name_Tuesday=1 ) THEN ss_sales_price " +
"             ELSE NULL " +
"           end) AS tue_sales, " +
"       Sum(CASE " +
"             WHEN ( d_day_name_Wednesday=1 ) THEN ss_sales_price " +
"             ELSE NULL " +
"           end) AS wed_sales, " +
"       Sum(CASE " +
"             WHEN ( d_day_name_Thursday=1 ) THEN ss_sales_price " +
"             ELSE NULL " +
"           end) AS thu_sales, " +
"       Sum(CASE " +
"             WHEN ( d_day_name_Friday=1 ) THEN ss_sales_price " +
"             ELSE NULL " +
"           end) AS fri_sales, " +
"       Sum(CASE " +
"             WHEN ( d_day_name_Saturday=1 ) THEN ss_sales_price " +
"             ELSE NULL " +
"           end) AS sat_sales " +
"FROM   date_dim, " +
"       store_sales, " +
"       store " +
"WHERE  d_date_sk = ss_sold_date_sk " +
"       AND s_store_sk = ss_store_sk " +
"       AND d_year > 2000 " +
"GROUP  BY s_store_name, " +
"          s_store_id " +
"ORDER  BY s_store_name, " +
"          s_store_id, " +
"          sun_sales, " +
"          mon_sales, " +
"          tue_sales, " +
"          wed_sales, " +
"          thu_sales, " +
"          fri_sales, " +
"          sat_sales   ";


// BOOL QUERY 45
final static String bq45 =
"  SELECT ca_zip, " +
"       ca_city, " +
```

```
"       ca_county, " +
"       ca_state, " +
"       Sum(ws_sales_price) " +
"FROM   web_sales, " +
"       customer, " +
"       customer_address, " +
"       date_dim, " +
"       item " +
"WHERE  ws_bill_customer_sk = c_customer_sk " +
"       AND c_current_addr_sk = ca_address_sk " +
"       AND ws_item_sk = i_item_sk " +
"       AND i_item_id IN (SELECT i_item_id " +
"                         FROM   item " +
"                         WHERE  i_item_sk < 5000) " +
"       AND ws_sold_date_sk = d_date_sk " +
"       AND d_quarter_name_fourth=1 " +
"       AND d_year > 2000 " +
"GROUP  BY ca_zip, " +
"          ca_city, " +
"          ca_county, " +
"          ca_state " +
"ORDER  BY ca_zip, " +
"          ca_city, " +
"          ca_county, " +
"          ca_state   ";

//  BOOL QUERY 46
final static String bq46 =
"  SELECT c_last_name, " +
"       c_first_name, " +
"       ca_city, " +
"       bought_city, " +
"       ss_ticket_number, " +
"       amt, " +
"       profit " +
"FROM   (SELECT ss_ticket_number, " +
"               ss_customer_sk, " +
"               ca_city            AS bought_city, " +
"               Sum(ss_coupon_amt) AS amt, " +
"               Sum(ss_net_profit) AS profit " +
"        FROM   store_sales, " +
"               date_dim, " +
"               store, " +
"               household_demographics, " +
"               customer_address " +
"        WHERE  store_sales.ss_sold_date_sk = date_dim.d_date_sk " +
"               AND store_sales.ss_store_sk = store.s_store_sk " +
"               AND store_sales.ss_hdemo_sk = household_demographics.hd_demo_sk " +
"               AND store_sales.ss_addr_sk = customer_address.ca_address_sk " +
"               AND ( household_demographics.hd_dep_count < 5 " +
"                      OR household_demographics.hd_vehicle_count < 2 ) " +
"               AND ( date_dim.d_day_name_Monday=1 OR " +
"  date_dim.d_day_name_Tuesday=1 OR " +
"                        date_dim.d_day_name_Wednesday=1 OR " +
"                         date_dim.d_day_name_Thursday=1) " +
"               AND date_dim.d_year IN ( 1999, 2000, 2001 ) " +
"        GROUP  BY ss_ticket_number, " +
"                  ss_customer_sk, " +
"                  ss_addr_sk, " +
"                  ca_city) dn, " +
"       customer, " +
"       customer_address current_addr " +
"WHERE  ss_customer_sk = c_customer_sk " +
```

```
"       AND customer.c_current_addr_sk = current_addr.ca_address_sk " +
"       AND current_addr.ca_city <> bought_city " +
"ORDER  BY c_last_name, " +
"          c_first_name, " +
"          ca_city, " +
"          bought_city, " +
"          ss_ticket_number   ";



// BOOL QUERY 48
final static String bq48 =
"  SELECT Sum(ss_quantity) " +
" FROM   store_sales, " +
"       store, " +
"       customer_demographics, " +
"       customer_address, " +
"       date_dim " +
" WHERE  s_store_sk = ss_store_sk " +
"       AND ss_sold_date_sk = d_date_sk " +
"       AND d_year > 2000 " +
"       AND ( ( cd_demo_sk = ss_cdemo_sk " +
"               AND cd_marital_status_married=1 " +
"               AND cd_education_status_university=1 " +
"               AND ss_sales_price BETWEEN 25000.00 AND 50000.00 ) " +
"              OR ( cd_demo_sk = ss_cdemo_sk " +
"                   AND cd_marital_status_married=1 " +
"                   AND cd_education_status_university=1 " +
"                   AND ss_sales_price BETWEEN 0.00 AND 25000.00 ) " +
"              OR ( cd_demo_sk = ss_cdemo_sk " +
"                   AND cd_marital_status_married=1 " +
"                   AND cd_education_status_university=1 " +
"                   AND ss_sales_price BETWEEN 75000.00 AND 100000.00 ) );   ";

//BOOL QUERY 49
final static String bq49 =
" select  " +
" 'web' as channel " +
" ,web.item " +
" ,web.return_ratio " +
" ,web.return_rank " +
" ,web.currency_rank " +
" from ( " +
"   select  " +
"    item " +
"   ,return_ratio " +
"   ,currency_ratio " +
"   ,@return_rank := @return_rank + 1 AS return_rank " +
"   ,@currency_rank := @currency_rank + 1 AS currency_rank " +
"   from  " +
"   (SELECT @return_rank := 0 ) r1,  " +
"   (SELECT @currency_rank := 0 ) r2,  " +
"    " +
"   (  select ws.ws_item_sk as item " +
"     ,(cast(sum(coalesce(wr.wr_return_quantity,0)) as dec(15,4))/ " +
"     cast(sum(coalesce(ws.ws_quantity,0)) as dec(15,4) )) as return_ratio " +
"     ,(cast(sum(coalesce(wr.wr_return_amt,0)) as dec(15,4))/ " +
"     cast(sum(coalesce(ws.ws_net_paid,0)) as dec(15,4) )) as currency_ratio " +
"     from  " +
"      web_sales ws left outer join web_returns wr  " +
"       on (ws.ws_order_number = wr.wr_order_number and  " +
"       ws.ws_item_sk = wr.wr_item_sk) " +
"                 ,date_dim " +
```

```
"     where  " +
"       wr.wr_return_amt > 10000  " +
"       and ws.ws_net_profit > 1 " +
"                         and ws.ws_net_paid > 0 " +
"                         and ws.ws_quantity > 0 " +
"                         and ws_sold_date_sk = d_date_sk " +
"                         and d_year > 2000 " +
"                         and d_month_name_December=1 " +
"     group by ws.ws_item_sk " +
"   ) in_web " +
" ) web " +
" where  " +
" ( " +
" web.return_rank <= 10 " +
" or " +
" web.currency_rank <= 10 " +
" ) " +
" union " +
" select  " +
" 'catalog' as channel " +
" ,catalog.item " +
" ,catalog.return_ratio " +
" ,catalog.return_rank " +
" ,catalog.currency_rank " +
" from ( " +
"   select  " +
"    item " +
"   ,return_ratio " +
"   ,currency_ratio " +
"   ,@return_rank := @return_rank + 1 AS return_rank " +
"   ,@currency_rank := @currency_rank + 1 AS currency_rank " +
"   from  " +
"   (SELECT @return_rank := 0 ) r1,  " +
"   (SELECT @currency_rank := 0 ) r2, " +
"   (  select  " +
"     cs.cs_item_sk as item " +
"     ,(cast(sum(coalesce(cr.cr_return_quantity,0)) as dec(15,4))/ " +
"     cast(sum(coalesce(cs.cs_quantity,0)) as dec(15,4) )) as return_ratio " +
"     ,(cast(sum(coalesce(cr.cr_return_amount,0)) as dec(15,4))/ " +
"     cast(sum(coalesce(cs.cs_net_paid,0)) as dec(15,4) )) as currency_ratio " +
"     from  " +
"     catalog_sales cs left outer join catalog_returns cr " +
"       on (cs.cs_order_number = cr.cr_order_number and  " +
"       cs.cs_item_sk = cr.cr_item_sk) " +
"                ,date_dim " +
"     where  " +
"       cr.cr_return_amount > 10000  " +
"       and cs.cs_net_profit > 1 " +
"                         and cs.cs_net_paid > 0 " +
"                         and cs.cs_quantity > 0 " +
"                         and cs_sold_date_sk = d_date_sk " +
"                         and d_year > 2000 " +
"                         and d_month_name_December=1 " +
"                 group by cs.cs_item_sk " +
"   ) in_cat " +
" ) catalog " +
" where  " +
" ( " +
" catalog.return_rank <= 10 " +
" or " +
" catalog.currency_rank <=10 " +
" ) " +
" union " +
```

```
" select  " +
" 'store' as channel " +
" ,store.item " +
" ,store.return_ratio " +
" ,store.return_rank " +
" ,store.currency_rank " +
" from ( " +
"   select  " +
"    item " +
"   ,return_ratio " +
"   ,currency_ratio " +
"   ,@return_rank := @return_rank + 1 AS return_rank " +
"   ,@currency_rank := @currency_rank + 1 AS currency_rank " +
"   from  " +
"   (SELECT @return_rank := 0 ) r1,  " +
"   (SELECT @currency_rank := 0 ) r2, " +
"   (  select sts.ss_item_sk as item " +
"     ,(cast(sum(coalesce(sr.sr_return_quantity,0)) as
dec(15,4))/cast(sum(coalesce(sts.ss_quantity,0)) as dec(15,4) )) as return_ratio " +
"     ,(cast(sum(coalesce(sr.sr_return_amt,0)) as
dec(15,4))/cast(sum(coalesce(sts.ss_net_paid,0)) as dec(15,4) )) as currency_ratio " +
"     from  " +
"     store_sales sts left outer join store_returns sr " +
"       on (sts.ss_ticket_number = sr.sr_ticket_number and sts.ss_item_sk = sr.sr_item_sk) " +
"                ,date_dim " +
"     where  " +
"       sr.sr_return_amt > 10000  " +
"       and sts.ss_net_profit > 1 " +
"                         and sts.ss_net_paid > 0  " +
"                         and sts.ss_quantity > 0 " +
"                         and ss_sold_date_sk = d_date_sk " +
"                         and d_year > 2000 " +
"                         and d_month_name_December=1 " +
"     group by sts.ss_item_sk " +
"   ) in_store " +
" ) store " +
" where  ( " +
" store.return_rank <= 10 " +
" or  " +
" store.currency_rank <= 10 " +
" ) " +
" order by 1,4,5 " ;

// BOOL QUERY 50
final static String bq50 =
"  SELECT s_store_name, " +
"       s_company_id, " +
"       s_street_number, " +
"       s_street_name, " +
"       s_street_type, " +
"       s_suite_number, " +
"       s_city, " +
"       s_county, " +
"       s_state, " +
"       s_zip, " +
"       Sum(CASE " +
"             WHEN ( sr_returned_date_sk - ss_sold_date_sk <= 30 ) THEN 1 " +
"             ELSE 0 " +
"           END) AS '30 days', " +
"       Sum(CASE " +
"             WHEN ( sr_returned_date_sk - ss_sold_date_sk > 30 ) " +
"                  AND ( sr_returned_date_sk - ss_sold_date_sk <= 60 ) THEN 1 " +
"             ELSE 0 " +
```

```
"           END) AS '31-60 days', " +
"       Sum(CASE " +
"             WHEN ( sr_returned_date_sk - ss_sold_date_sk > 60 ) " +
"                  AND ( sr_returned_date_sk - ss_sold_date_sk <= 90 ) THEN 1 " +
"             ELSE 0 " +
"           END) AS '61-90 days', " +
"       Sum(CASE " +
"             WHEN ( sr_returned_date_sk - ss_sold_date_sk > 90 ) " +
"                  AND ( sr_returned_date_sk - ss_sold_date_sk <= 120 ) THEN 1 " +
"             ELSE 0 " +
"           END) AS '91-120 days', " +
"       Sum(CASE " +
"             WHEN ( sr_returned_date_sk - ss_sold_date_sk > 120 ) THEN 1 " +
"             ELSE 0 " +
"           END) AS '>120 days' " +
" FROM   store_sales, " +
"       store_returns, " +
"       store, " +
"       date_dim d1, " +
"       date_dim d2 " +
" WHERE  d2.d_year > 2000 " +
"       AND d2.d_month_name_May=1 " +
"       AND ss_ticket_number = sr_ticket_number " +
"       AND ss_item_sk = sr_item_sk " +
"       AND ss_sold_date_sk = d1.d_date_sk " +
"       AND sr_returned_date_sk = d2.d_date_sk " +
"       AND ss_customer_sk = sr_customer_sk " +
"       AND ss_store_sk = s_store_sk " +
" GROUP  BY s_store_name, " +
"          s_company_id, " +
"          s_street_number, " +
"          s_street_name, " +
"          s_street_type, " +
"          s_suite_number, " +
"          s_city, " +
"          s_county, " +
"          s_state, " +
"          s_zip " +
" ORDER  BY s_store_name, " +
"          s_company_id, " +
"          s_street_number, " +
"          s_street_name, " +
"          s_street_type, " +
"          s_suite_number, " +
"          s_city, " +
"          s_county, " +
"          s_state, " +
"          s_zip;   ";

// BOOL QUERY 52
final static String bq52 =
"  SELECT dt.d_year, " +
"       item.i_brand_id         AS brand_id, " +
"       item.i_brand            AS brand, " +
"       Sum(ss_ext_sales_price) AS ext_price " +
"FROM   date_dim dt, " +
"       store_sales, " +
"       item " +
"WHERE  dt.d_date_sk = store_sales.ss_sold_date_sk " +
"       AND store_sales.ss_item_sk = item.i_item_sk " +
"       AND dt.d_month_name_September=1 " +
"       AND dt.d_year > 2000 " +
"GROUP  BY dt.d_year, " +
```

```
"          item.i_brand, " +
"          item.i_brand_id " +
"ORDER  BY dt.d_year, " +
"          ext_price DESC, " +
"          brand_id;  ";

// BOOL QUERY 55
final static String bq55 =
"  SELECT i_brand_id              AS brand_id, " +
"       i_brand                 AS brand, " +
"       Sum(ss_ext_sales_price) AS ext_price " +
"FROM   date_dim, " +
"       store_sales, " +
"       item " +
"WHERE  d_date_sk = ss_sold_date_sk " +
"       AND ss_item_sk = i_item_sk " +
"       AND d_month_name_July=1 " +
"       AND d_year > 2000 " +
"GROUP  BY i_brand, " +
"          i_brand_id " +
"ORDER  BY ext_price DESC, " +
"          i_brand_id   ";

// BOOL QUERY 64
final static  String bq64 =
" select cs1.product_name,cs1.store_name " +
"     ,cs1.store_zip,cs1.b_street_number " +
"     ,cs1.b_streen_name,cs1.b_city " +
"     ,cs1.b_zip,cs1.c_street_number " +
"     ,cs1.c_street_name,cs1.c_city " +
"     ,cs1.c_zip,cs1.syear " +
"     ,cs1.cnt,cs1.s1,cs1.s2,cs1.s3 " +
"     ,cs2.s1,cs2.s2,cs2.s3,cs2.syear,cs2.cnt " +
" " +
"from  " +
"\t\t(select i_product_name product_name " +
"     ,i_item_sk item_sk " +
"     ,s_store_name store_name " +
"     ,s_zip store_zip " +
"     ,ad1.ca_street_number b_street_number " +
"     ,ad1.ca_street_name b_streen_name " +
"     ,ad1.ca_city b_city " +
"     ,ad1.ca_zip b_zip " +
"     ,ad2.ca_street_number c_street_number " +
"     ,ad2.ca_street_name c_street_name " +
"     ,ad2.ca_city c_city " +
"     ,ad2.ca_zip c_zip " +
"     ,d1.d_year as syear " +
"     ,d2.d_year as fsyear " +
"     ,d3.d_year s2year " +
"     ,count(*) as cnt " +
"     ,sum(ss_wholesale_cost) as s1 " +
"     ,sum(ss_list_price) as s2 " +
"     ,sum(ss_coupon_amt) as s3 " +
"  FROM   store_sales " +
"        ,store_returns " +
"        , (select cs_item_sk " +
"        ,sum(cs_ext_list_price) as sale,sum(cr_refunded_cash+cr_reversed_charge+cr_store_credit)
as refund " +
"  from catalog_sales " +
"      ,catalog_returns " +
"  where cs_item_sk = cr_item_sk " +
"    and cs_order_number = cr_order_number " +
```

```
"  group by cs_item_sk " +
"  having sum(cs_ext_list_price)>2*sum(cr_refunded_cash+cr_reversed_charge+cr_store_credit)) as
cs_ui " +
"        ,date_dim d1 " +
"        ,date_dim d2 " +
"        ,date_dim d3 " +
"        ,store " +
"        ,customer " +
"        ,customer_demographics cd1 " +
"        ,customer_demographics cd2 " +
"        ,promotion " +
"        ,household_demographics hd1 " +
"        ,household_demographics hd2 " +
"        ,customer_address ad1 " +
"        ,customer_address ad2 " +
"        ,income_band ib1 " +
"        ,income_band ib2 " +
"        ,item " +
"  WHERE  ss_store_sk = s_store_sk AND " +
"         ss_sold_date_sk = d1.d_date_sk AND " +
"         ss_customer_sk = c_customer_sk AND " +
"         ss_cdemo_sk= cd1.cd_demo_sk AND " +
"         ss_hdemo_sk = hd1.hd_demo_sk AND " +
"         ss_addr_sk = ad1.ca_address_sk and " +
"         ss_item_sk = i_item_sk and " +
"         ss_item_sk = sr_item_sk and " +
"         ss_ticket_number = sr_ticket_number and " +
"         ss_item_sk = cs_ui.cs_item_sk and " +
"         c_current_cdemo_sk = cd2.cd_demo_sk AND " +
"         c_current_hdemo_sk = hd2.hd_demo_sk AND " +
"         c_current_addr_sk = ad2.ca_address_sk and " +
"         c_first_sales_date_sk = d2.d_date_sk and " +
"         c_first_shipto_date_sk = d3.d_date_sk and " +
"         ss_promo_sk = p_promo_sk and " +
"         hd1.hd_income_band_sk = ib1.ib_income_band_sk and " +
"         hd2.hd_income_band_sk = ib2.ib_income_band_sk and " +
"         ((cd1.cd_marital_status_single = 1 AND cd2.cd_marital_status_single = 0) OR
(cd1.cd_marital_status_married = 1 AND cd2.cd_marital_status_married = 0) OR
(cd1.cd_marital_status_divorced = 1 AND cd2.cd_marital_status_divorced = 0)) " +
"group by i_product_name " +
"       ,i_item_sk " +
"       ,s_store_name " +
"       ,s_zip " +
"       ,ad1.ca_street_number " +
"       ,ad1.ca_street_name " +
"       ,ad1.ca_city " +
"       ,ad1.ca_zip " +
"       ,ad2.ca_street_number " +
"       ,ad2.ca_street_name " +
"       ,ad2.ca_city " +
"       ,ad2.ca_zip " +
"       ,d1.d_year " +
"       ,d2.d_year " +
"       ,d3.d_year) as cs1, " +
"    " +
" (select i_product_name product_name " +
"     ,i_item_sk item_sk " +
"     ,s_store_name store_name " +
"     ,s_zip store_zip " +
"     ,ad1.ca_street_number b_street_number " +
"     ,ad1.ca_street_name b_streen_name " +
"     ,ad1.ca_city b_city " +
"     ,ad1.ca_zip b_zip " +
```

```
"     ,ad2.ca_street_number c_street_number " +
"     ,ad2.ca_street_name c_street_name " +
"     ,ad2.ca_city c_city " +
"     ,ad2.ca_zip c_zip " +
"     ,d1.d_year as syear " +
"     ,d2.d_year as fsyear " +
"     ,d3.d_year s2year " +
"     ,count(*) as cnt " +
"     ,sum(ss_wholesale_cost) as s1 " +
"     ,sum(ss_list_price) as s2 " +
"     ,sum(ss_coupon_amt) as s3 " +
"  FROM   store_sales " +
"        ,store_returns " +
"        , (select cs_item_sk " +
"        ,sum(cs_ext_list_price) as sale,sum(cr_refunded_cash+cr_reversed_charge+cr_store_credit)
as refund " +
"  from catalog_sales " +
"      ,catalog_returns " +
"  where cs_item_sk = cr_item_sk " +
"    and cs_order_number = cr_order_number " +
"  group by cs_item_sk " +
"  having sum(cs_ext_list_price)>2*sum(cr_refunded_cash+cr_reversed_charge+cr_store_credit)) as
cs_ui " +
"        ,date_dim d1 " +
"        ,date_dim d2 " +
"        ,date_dim d3 " +
"        ,store " +
"        ,customer " +
"        ,customer_demographics cd1 " +
"        ,customer_demographics cd2 " +
"        ,promotion " +
"        ,household_demographics hd1 " +
"        ,household_demographics hd2 " +
"        ,customer_address ad1 " +
"        ,customer_address ad2 " +
"        ,income_band ib1 " +
"        ,income_band ib2 " +
"        ,item " +
"  WHERE  ss_store_sk = s_store_sk AND " +
"         ss_sold_date_sk = d1.d_date_sk AND " +
"         ss_customer_sk = c_customer_sk AND " +
"         ss_cdemo_sk= cd1.cd_demo_sk AND " +
"         ss_hdemo_sk = hd1.hd_demo_sk AND " +
"         ss_addr_sk = ad1.ca_address_sk and " +
"         ss_item_sk = i_item_sk and " +
"         ss_item_sk = sr_item_sk and " +
"         ss_ticket_number = sr_ticket_number and " +
"         ss_item_sk = cs_ui.cs_item_sk and " +
"         c_current_cdemo_sk = cd2.cd_demo_sk AND " +
"         c_current_hdemo_sk = hd2.hd_demo_sk AND " +
"         c_current_addr_sk = ad2.ca_address_sk and " +
"         c_first_sales_date_sk = d2.d_date_sk and " +
"         c_first_shipto_date_sk = d3.d_date_sk and " +
"         ss_promo_sk = p_promo_sk and " +
"         hd1.hd_income_band_sk = ib1.ib_income_band_sk and " +
"         hd2.hd_income_band_sk = ib2.ib_income_band_sk and " +
"         ((cd1.cd_marital_status_single = 1 AND cd2.cd_marital_status_single = 0) OR
(cd1.cd_marital_status_married = 1 AND cd2.cd_marital_status_married = 0) OR
(cd1.cd_marital_status_divorced = 1 AND cd2.cd_marital_status_divorced = 0)) " +
" group by i_product_name " +
"       ,i_item_sk " +
"       ,s_store_name " +
"       ,s_zip " +
```

```
"       ,ad1.ca_street_number " +
"       ,ad1.ca_street_name " +
"       ,ad1.ca_city " +
"       ,ad1.ca_zip " +
"       ,ad2.ca_street_number " +
"       ,ad2.ca_street_name " +
"       ,ad2.ca_city " +
"       ,ad2.ca_zip " +
"       ,d1.d_year " +
"       ,d2.d_year " +
"       ,d3.d_year) as cs2 " +
" " +
" where cs1.item_sk=cs2.item_sk and " +
"     cs1.syear > 2000 and " +
"     cs2.syear > 2000 and " +
"     cs2.syear != cs1.syear and " +
"     cs2.cnt <= cs1.cnt " +
" " +
"order by cs1.product_name,cs1.store_name,cs2.cnt ";

// BOOL QUERY 66
final static String bq66 =
"  SELECT w_warehouse_name, " +
"       w_warehouse_sq_ft, " +
"       w_city, " +
"       w_county, " +
"       w_state, " +
"       w_country, " +
"       year, " +
"       Sum(jan_sales)                     AS jan_sales, " +
"       Sum(feb_sales)                     AS feb_sales, " +
"       Sum(mar_sales)                     AS mar_sales, " +
"       Sum(apr_sales)                     AS apr_sales, " +
"       Sum(may_sales)                     AS may_sales, " +
"       Sum(jun_sales)                     AS jun_sales, " +
"       Sum(jul_sales)                     AS jul_sales, " +
"       Sum(aug_sales)                     AS aug_sales, " +
"       Sum(sep_sales)                     AS sep_sales, " +
"       Sum(oct_sales)                     AS oct_sales, " +
"       Sum(nov_sales)                     AS nov_sales, " +
"       Sum(dec_sales)                     AS dec_sales, " +
"       Sum(jan_sales / w_warehouse_sq_ft) AS jan_sales_per_sq_foot, " +
"       Sum(feb_sales / w_warehouse_sq_ft) AS feb_sales_per_sq_foot, " +
"       Sum(mar_sales / w_warehouse_sq_ft) AS mar_sales_per_sq_foot, " +
"       Sum(apr_sales / w_warehouse_sq_ft) AS apr_sales_per_sq_foot, " +
"       Sum(may_sales / w_warehouse_sq_ft) AS may_sales_per_sq_foot, " +
"       Sum(jun_sales / w_warehouse_sq_ft) AS jun_sales_per_sq_foot, " +
"       Sum(jul_sales / w_warehouse_sq_ft) AS jul_sales_per_sq_foot, " +
"       Sum(aug_sales / w_warehouse_sq_ft) AS aug_sales_per_sq_foot, " +
"       Sum(sep_sales / w_warehouse_sq_ft) AS sep_sales_per_sq_foot, " +
"       Sum(oct_sales / w_warehouse_sq_ft) AS oct_sales_per_sq_foot, " +
"       Sum(nov_sales / w_warehouse_sq_ft) AS nov_sales_per_sq_foot, " +
"       Sum(dec_sales / w_warehouse_sq_ft) AS dec_sales_per_sq_foot, " +
"       Sum(jan_net)                       AS jan_net, " +
"       Sum(feb_net)                       AS feb_net, " +
"       Sum(mar_net)                       AS mar_net, " +
"       Sum(apr_net)                       AS apr_net, " +
"       Sum(may_net)                       AS may_net, " +
"       Sum(jun_net)                       AS jun_net, " +
"       Sum(jul_net)                       AS jul_net, " +
"       Sum(aug_net)                       AS aug_net, " +
"       Sum(sep_net)                       AS sep_net, " +
"       Sum(oct_net)                       AS oct_net, " +
```

```
"       Sum(nov_net)                        AS nov_net, " +
"       Sum(dec_net)                        AS dec_net " +
" FROM   ((SELECT w_warehouse_name, " +
"                 w_warehouse_sq_ft, " +
"                 w_city, " +
"                 w_county, " +
"                 w_state, " +
"                 w_country, " +
"                 d_year   AS year, " +
"                 Sum(CASE " +
"                       WHEN d_month_name_January = 1 THEN " +
"                       ws_sales_price * ws_quantity " +
"                       ELSE 0 " +
"                     end) AS jan_sales, " +
"                 Sum(CASE " +
"                       WHEN d_month_name_February = 1 THEN " +
"                       ws_sales_price * ws_quantity " +
"                       ELSE 0 " +
"                     end) AS feb_sales, " +
"                 Sum(CASE " +
"                       WHEN d_month_name_March = 1 THEN " +
"                       ws_sales_price * ws_quantity " +
"                       ELSE 0 " +
"                     end) AS mar_sales, " +
"                 Sum(CASE " +
"                       WHEN d_month_name_April = 1 THEN " +
"                       ws_sales_price * ws_quantity " +
"                       ELSE 0 " +
"                     end) AS apr_sales, " +
"                 Sum(CASE " +
"                       WHEN d_month_name_May = 1 THEN " +
"                       ws_sales_price * ws_quantity " +
"                       ELSE 0 " +
"                     end) AS may_sales, " +
"                 Sum(CASE " +
"                       WHEN d_month_name_June = 1 THEN " +
"                       ws_sales_price * ws_quantity " +
"                       ELSE 0 " +
"                     end) AS jun_sales, " +
"                 Sum(CASE " +
"                       WHEN d_month_name_July = 1 THEN " +
"                       ws_sales_price * ws_quantity " +
"                       ELSE 0 " +
"                     end) AS jul_sales, " +
"                 Sum(CASE " +
"                       WHEN d_month_name_August = 1 THEN " +
"                       ws_sales_price * ws_quantity " +
"                       ELSE 0 " +
"                     end) AS aug_sales, " +
"                 Sum(CASE " +
"                       WHEN d_month_name_September = 1 THEN " +
"                       ws_sales_price * ws_quantity " +
"                       ELSE 0 " +
"                     end) AS sep_sales, " +
"                 Sum(CASE " +
"                       WHEN d_month_name_October = 1 THEN " +
"                       ws_sales_price * ws_quantity " +
"                       ELSE 0 " +
"                     end) AS oct_sales, " +
"                 Sum(CASE " +
"                       WHEN d_month_name_November = 1 THEN " +
"                       ws_sales_price * ws_quantity " +
"                       ELSE 0 " +
```

```
"                    end) AS nov_sales, " +
"                Sum(CASE " +
"                      WHEN d_month_name_December = 1 THEN " +
"                      ws_sales_price * ws_quantity " +
"                      ELSE 0 " +
"                    end) AS dec_sales, " +
"                Sum(CASE " +
"                      WHEN d_month_name_January = 1 THEN " +
"                      ws_net_paid * ws_quantity " +
"                      ELSE 0 " +
"                    end) AS jan_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name_February = 1 THEN " +
"                      ws_net_paid * ws_quantity " +
"                      ELSE 0 " +
"                    end) AS feb_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name_March = 1 THEN ws_net_paid * ws_quantity " +
"                      ELSE 0 " +
"                    end) AS mar_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name_April = 1 THEN ws_net_paid * ws_quantity " +
"                      ELSE 0 " +
"                    end) AS apr_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name_May = 1 THEN ws_net_paid * ws_quantity " +
"                      ELSE 0 " +
"                    end) AS may_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name_June = 1 THEN ws_net_paid * ws_quantity " +
"                      ELSE 0 " +
"                    end) AS jun_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name_July = 1 THEN ws_net_paid * ws_quantity " +
"                      ELSE 0 " +
"                    end) AS jul_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name_August = 1 THEN " +
"                      ws_net_paid * ws_quantity " +
"                      ELSE 0 " +
"                    end) AS aug_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name_September = 1 THEN " +
"                      ws_net_paid * ws_quantity " +
"                      ELSE 0 " +
"                    end) AS sep_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name_October = 1 THEN " +
"                      ws_net_paid * ws_quantity " +
"                      ELSE 0 " +
"                    end) AS oct_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name_November = 1 THEN " +
"                      ws_net_paid * ws_quantity " +
"                      ELSE 0 " +
"                    end) AS nov_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name_December = 1 THEN " +
"                      ws_net_paid * ws_quantity " +
"                      ELSE 0 " +
"                    end) AS dec_net " +
"         FROM   web_sales, " +
"                warehouse, " +
```

```
"                date_dim, " +
"                time_dim, " +
"                ship_mode " +
"         WHERE  ws_warehouse_sk = w_warehouse_sk " +
"                AND ws_sold_date_sk = d_date_sk " +
"                AND ws_sold_time_sk = t_time_sk " +
"                AND ws_ship_mode_sk = sm_ship_mode_sk " +
"                AND d_year = 2000 " +
"         GROUP  BY w_warehouse_name, " +
"                   w_warehouse_sq_ft, " +
"                   w_city, " +
"                   w_county, " +
"                   w_state, " +
"                   w_country, " +
"                   d_year) " +
"        UNION ALL " +
"        (SELECT w_warehouse_name, " +
"                w_warehouse_sq_ft, " +
"                w_city, " +
"                w_county, " +
"                w_state, " +
"                w_country, " +
"                d_year   AS year, " +
"                Sum(CASE " +
"                      WHEN d_month_name_January = 1 THEN cs_sales_price * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS jan_sales, " +
"                Sum(CASE " +
"                      WHEN d_month_name_February = 1 THEN cs_sales_price * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS feb_sales, " +
"                Sum(CASE " +
"                      WHEN d_month_name_March = 1 THEN cs_sales_price * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS mar_sales, " +
"                Sum(CASE " +
"                      WHEN d_month_name_April = 1 THEN cs_sales_price * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS apr_sales, " +
"                Sum(CASE " +
"                      WHEN d_month_name_May = 1 THEN cs_sales_price * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS may_sales, " +
"                Sum(CASE " +
"                      WHEN d_month_name_June = 1 THEN cs_sales_price * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS jun_sales, " +
"                Sum(CASE " +
"                      WHEN d_month_name_July = 1 THEN cs_sales_price * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS jul_sales, " +
"                Sum(CASE " +
"                      WHEN d_month_name_August = 1 THEN cs_sales_price * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS aug_sales, " +
"                Sum(CASE " +
"                      WHEN d_month_name_September = 1 THEN cs_sales_price * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS sep_sales, " +
"                Sum(CASE " +
"                      WHEN d_month_name_October = 1 THEN cs_sales_price * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS oct_sales, " +
```

```
"                Sum(CASE " +
"                      WHEN d_month_name_November = 1 THEN cs_sales_price * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS nov_sales, " +
"                Sum(CASE " +
"                      WHEN d_month_name_December = 1 THEN cs_sales_price * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS dec_sales, " +
"                Sum(CASE " +
"                      WHEN d_month_name_January = 1 THEN cs_net_paid * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS jan_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name_February = 1 THEN cs_net_paid * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS feb_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name_March = 1 THEN cs_net_paid * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS mar_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name_April = 1 THEN cs_net_paid * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS apr_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name_May = 1 THEN cs_net_paid * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS may_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name_June = 1 THEN cs_net_paid * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS jun_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name_July = 1 THEN cs_net_paid * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS jul_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name_August = 1 THEN cs_net_paid * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS aug_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name_September = 1 THEN cs_net_paid * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS sep_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name_October = 1 THEN cs_net_paid * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS oct_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name_November = 1 THEN cs_net_paid * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS nov_net, " +
"                Sum(CASE " +
"                      WHEN d_month_name_December = 1 THEN cs_net_paid * cs_quantity " +
"                      ELSE 0 " +
"                    end) AS dec_net " +
"         FROM   catalog_sales, " +
"                warehouse, " +
"                date_dim, " +
"                time_dim, " +
"                ship_mode " +
"         WHERE  cs_warehouse_sk = w_warehouse_sk " +
"                AND cs_sold_date_sk = d_date_sk " +
```

```
"                AND cs_sold_time_sk = t_time_sk " +
"                AND cs_ship_mode_sk = sm_ship_mode_sk " +
"                AND d_year = 2000 " +
"         GROUP  BY w_warehouse_name, " +
"                   w_warehouse_sq_ft, " +
"                   w_city, " +
"                   w_county, " +
"                   w_state, " +
"                   w_country, " +
"                   d_year)) x " +
" GROUP  BY w_warehouse_name, " +
"           w_warehouse_sq_ft, " +
"           w_city, " +
"           w_county, " +
"           w_state, " +
"           w_country, " +
"           year " +
" ORDER  BY w_warehouse_name   ";

// BOOL QUERY 69_1
final static String bq69_1 =
"  SELECT  " +
"       Count(*) AS cnt1, " +
"       cd_purchase_estimate, " +
"       Count(*) AS cnt2, " +
"       cd_credit_rating, " +
"       Count(*) AS cnt3 " +
" FROM   customer c, " +
"       customer_address ca, " +
"       customer_demographics " +
" WHERE  c.c_current_addr_sk = ca.ca_address_sk " +
"       AND cd_demo_sk = c.c_current_cdemo_sk " +
"       AND EXISTS (SELECT * " +
"                   FROM   store_sales, " +
"                          date_dim " +
"                   WHERE  c.c_customer_sk = ss_customer_sk " +
"                          AND ss_sold_date_sk = d_date_sk " +
"                          AND d_year > 2000 " +
"                          AND (d_month_name_January=1 OR d_month_name_February=1 OR
d_month_name_March=1 OR d_month_name_April=1 OR d_month_name_May=1 OR d_month_name_June=1)) " +
"       AND ( NOT EXISTS (SELECT * " +
"                         FROM   web_sales, " +
"                                date_dim " +
"                         WHERE  c.c_customer_sk = ws_bill_customer_sk " +
"                                AND ws_sold_date_sk = d_date_sk " +
"                                AND d_year > 2000 " +
"                                AND (d_month_name_January=1 OR d_month_name_February=1 OR
d_month_name_March=1 OR d_month_name_April=1 OR d_month_name_May=1 OR d_month_name_June=1)) " +
"             AND NOT EXISTS (SELECT * " +
"                             FROM   catalog_sales, " +
"                                    date_dim " +
"                             WHERE  c.c_customer_sk = cs_ship_customer_sk " +
"                                    AND cs_sold_date_sk = d_date_sk " +
"                                    AND d_year > 2000 " +
"                                    AND (d_month_name_January=1 OR d_month_name_February=1 OR
d_month_name_March=1 OR d_month_name_April=1 OR d_month_name_May=1 OR d_month_name_June=1)) ) " +
" GROUP  BY  " +
"           cd_purchase_estimate, " +
"           cd_credit_rating " +
" ORDER  BY  " +
"           cd_purchase_estimate, " +
"           cd_credit_rating;   ";
```

```
// BOOL QUERY 69_2
final static String bq69_2 =
"  SELECT cd_gender_male,cd_gender_female, " +
"       cd_marital_status_single,cd_marital_status_married,cd_marital_status_divorced, " +
"       cd_education_status_primary,cd_education_status_secondary, " +
"       cd_education_status_university,cd_education_status_postgraduate, " +
"       Count(*) AS cnt1, " +
"       cd_purchase_estimate, " +
"       Count(*) AS cnt2, " +
"       cd_credit_rating, " +
"       Count(*) AS cnt3 " +
"FROM   customer c, " +
"       customer_address ca, " +
"       customer_demographics " +
"WHERE  c.c_current_addr_sk = ca.ca_address_sk " +
"       AND cd_demo_sk = c.c_current_cdemo_sk " +
"       AND EXISTS (SELECT * " +
"                   FROM   store_sales, " +
"                          date_dim " +
"                   WHERE  c.c_customer_sk = ss_customer_sk " +
"                          AND ss_sold_date_sk = d_date_sk " +
"                          AND d_year > 2000 " +
"                          AND (d_month_name_January=1 OR d_month_name_February=1 OR
d_month_name_March=1 OR d_month_name_April=1 OR d_month_name_May=1 OR d_month_name_June=1)) " +
"       AND ( NOT EXISTS (SELECT * " +
"                         FROM   web_sales, " +
"                                date_dim " +
"                         WHERE  c.c_customer_sk = ws_bill_customer_sk " +
"                                AND ws_sold_date_sk = d_date_sk " +
"                                AND d_year > 2000 " +
"                                AND (d_month_name_January=1 OR d_month_name_February=1 OR
d_month_name_March=1 OR d_month_name_April=1 OR d_month_name_May=1 OR d_month_name_June=1)) " +
"             AND NOT EXISTS (SELECT * " +
"                             FROM   catalog_sales, " +
"                                    date_dim " +
"                             WHERE  c.c_customer_sk = cs_ship_customer_sk " +
"                                    AND cs_sold_date_sk = d_date_sk " +
"                                    AND d_year > 2000 " +
"                                    AND (d_month_name_January=1 OR d_month_name_February=1 OR
d_month_name_March=1 OR d_month_name_April=1 OR d_month_name_May=1 OR d_month_name_June=1)) ) " +
"GROUP  BY cd_gender_male,cd_gender_female, " +
"       cd_marital_status_single,cd_marital_status_married,cd_marital_status_divorced, " +
"       cd_education_status_primary,cd_education_status_secondary, " +
"       cd_education_status_university,cd_education_status_postgraduate, " +
"          cd_purchase_estimate, " +
"          cd_credit_rating " +
"ORDER  BY cd_gender_male,cd_gender_female, " +
"       cd_marital_status_single,cd_marital_status_married,cd_marital_status_divorced, " +
"       cd_education_status_primary,cd_education_status_secondary, " +
"       cd_education_status_university,cd_education_status_postgraduate, " +
"          cd_purchase_estimate, " +
"          cd_credit_rating;   ";

// BOOL QUERY 71
final static String bq71 =
"   select i_brand_id as brand_id, i_brand as brand,t_hour,t_minute,   " +
"                               sum(ext_price) as ext_price   " +
"                             from item, (select ws_ext_sales_price as ext_price,    " +
"                                                ws_sold_date_sk as sold_date_sk,   " +
"                                                ws_item_sk as sold_item_sk,   " +
"                                                ws_sold_time_sk as time_sk     " +
"                                           from web_sales,date_dim   " +
"                                           where d_date_sk = ws_sold_date_sk   " +
```

```
"                                          and d_month_name_February=1   " +
"                                          and d_year > 2000   " +
"                                        union all   " +
"                                        select cs_ext_sales_price as ext_price,   " +
"                                               cs_sold_date_sk as sold_date_sk,   " +
"                                               cs_item_sk as sold_item_sk,   " +
"                                               cs_sold_time_sk as time_sk   " +
"                                        from catalog_sales,date_dim   " +
"                                        where d_date_sk = cs_sold_date_sk   " +
"                                          and d_month_name_February=1  " +
"                                          and d_year>2000   " +
"                                        union all   " +
"                                        select ss_ext_sales_price as ext_price,   " +
"                                               ss_sold_date_sk as sold_date_sk,   " +
"                                               ss_item_sk as sold_item_sk,   " +
"                                               ss_sold_time_sk as time_sk   " +
"                                        from store_sales,date_dim   " +
"                                        where d_date_sk = ss_sold_date_sk   " +
"                                          and d_month_name_February=1  " +
"                                          and d_year>2000   " +
"                                        ) as tmp,time_dim   " +
"                            where   " +
"                              sold_item_sk = i_item_sk   " +
"                              and time_sk = t_time_sk     " +
"                            group by i_brand, i_brand_id,t_hour,t_minute   " +
"                            order by ext_price desc, i_brand_id   ";

// BOOL QUERY 72
final static String bq72 =
" SELECT i_item_desc, " +
"       w_warehouse_name, " +
"       d1.d_week_seq, " +
"       Count(CASE " +
"               WHEN p_promo_sk IS NULL THEN 1 " +
"               ELSE 0 " +
"             END) AS no_promo, " +
"       Count(CASE " +
"               WHEN p_promo_sk IS NOT NULL THEN 1 " +
"               ELSE 0 " +
"             END) AS promo, " +
"       Count(*)   AS total_cnt " +
"FROM   catalog_sales " +
"       JOIN inventory " +
"         ON ( cs_item_sk = inv_item_sk ) " +
"       JOIN warehouse " +
"         ON ( w_warehouse_sk = inv_warehouse_sk ) " +
"       JOIN item " +
"         ON ( i_item_sk = cs_item_sk ) " +
"       JOIN customer_demographics " +
"         ON ( cs_bill_cdemo_sk = cd_demo_sk ) " +
"       JOIN household_demographics " +
"         ON ( cs_bill_hdemo_sk = hd_demo_sk ) " +
"       JOIN date_dim d1 " +
"         ON ( cs_sold_date_sk = d1.d_date_sk ) " +
"       JOIN date_dim d2 " +
"         ON ( inv_date_sk = d2.d_date_sk ) " +
"       JOIN date_dim d3 " +
"         ON ( cs_ship_date_sk = d3.d_date_sk ) " +
"       LEFT OUTER JOIN promotion " +
"                    ON ( cs_promo_sk = p_promo_sk ) " +
"       LEFT OUTER JOIN catalog_returns " +
"                    ON ( cr_item_sk = cs_item_sk " +
"                         AND cr_order_number = cs_order_number ) " +
```

```
"WHERE  d3.d_date > d1.d_date " +
"       AND d1.d_year > 2000 " +
"       AND cd_marital_status_single=1 " +
"GROUP  BY i_item_desc, " +
"          w_warehouse_name, " +
"          d1.d_week_seq " +
"ORDER  BY total_cnt DESC, " +
"          i_item_desc, " +
"          w_warehouse_name, " +
"          d_week_seq;   ";

// BOOL QUERY 79
final static String bq79 =
"  SELECT c_last_name, " +
"       c_first_name, " +
"       Substr(s_city, 1, 30), " +
"       ss_ticket_number, " +
"       amt, " +
"       profit " +
"FROM   (SELECT ss_ticket_number, " +
"               ss_customer_sk, " +
"               store.s_city, " +
"               Sum(ss_coupon_amt) AS amt, " +
"               Sum(ss_net_profit) AS profit " +
"        FROM   store_sales, " +
"               date_dim, " +
"               store, " +
"               household_demographics " +
"        WHERE  store_sales.ss_sold_date_sk = date_dim.d_date_sk " +
"               AND store_sales.ss_store_sk = store.s_store_sk " +
"               AND store_sales.ss_hdemo_sk = household_demographics.hd_demo_sk " +
"               AND ( household_demographics.hd_dep_count < 3 " +
"                      OR household_demographics.hd_vehicle_count > 2 ) " +
"               AND date_dim.d_day_name_Monday = 1 " +
"               AND date_dim.d_year IN ( 1999, 2000, 2001 ) " +
"               AND store.s_number_employees BETWEEN 200 AND 1000 " +
"        GROUP  BY ss_ticket_number, " +
"                  ss_customer_sk, " +
"                  ss_addr_sk, " +
"                  store.s_city) ms, " +
"       customer " +
"WHERE  ss_customer_sk = c_customer_sk " +
"ORDER  BY c_last_name, " +
"          c_first_name, " +
"          Substr(s_city, 1, 30), " +
"          profit;   ";

// BOOL QUERY 85
final static String bq85 =
"  SELECT Substr(r_reason_desc, 1, 20), " +
"       Avg(ws_quantity), " +
"       Avg(ws_quantity), " +
"       Avg(ws_quantity), " +
"       Avg(wr_refunded_cash), " +
"       Avg(wr_fee) " +
"FROM   web_sales, " +
"       web_returns, " +
"       web_page, " +
"       customer_demographics cd1, " +
"       customer_demographics cd2, " +
"       customer_address, " +
"       date_dim, " +
"       reason " +
```

```
"WHERE  ws_web_page_sk = wp_web_page_sk " +
"       AND ws_item_sk = wr_item_sk " +
"       AND ws_order_number = wr_order_number " +
"       AND ws_sold_date_sk = d_date_sk " +
"       AND d_year > 2000 " +
"       AND cd1.cd_demo_sk = wr_refunded_cdemo_sk " +
"       AND cd2.cd_demo_sk = wr_returning_cdemo_sk " +
"       AND ca_address_sk = wr_refunded_addr_sk " +
"       AND r_reason_sk = wr_reason_sk " +
"       AND ( ( cd1.cd_marital_status_single=1 " +
"               AND cd2.cd_marital_status_single=1 " +
"               AND cd1.cd_education_status_secondary=1 " +
"               AND cd2.cd_education_status_secondary=1 ) " +
"              OR ( cd1.cd_marital_status_married=1 " +
"                   AND cd2.cd_marital_status_married=1 " +
"                   AND cd1.cd_education_status_university=1 " +
"                   AND cd2.cd_education_status_university=1) " +
"              OR ( cd1.cd_marital_status_divorced=1 " +
"                   AND cd2.cd_marital_status_divorced=1 " +
"                   AND cd1.cd_education_status_postgraduate=1 " +
"                   AND cd2.cd_education_status_postgraduate=1 ) ) " +
"GROUP  BY r_reason_desc " +
"ORDER  BY Substr(r_reason_desc, 1, 20), " +
"          Avg(ws_quantity), " +
"          Avg(wr_refunded_cash), " +
"          Avg(wr_fee);   ";

// BOOL QUERY 90
final static String bq90 =
"  SELECT Cast(amc AS DECIMAL(15, 4)) / Cast(pmc AS DECIMAL(15, 4)) AS am_pm_ratio " +
"FROM   (SELECT Count(*) AS amc " +
"        FROM   web_sales, " +
"               household_demographics, " +
"               time_dim, " +
"               web_page " +
"        WHERE  ws_sold_time_sk = time_dim.t_time_sk " +
"               AND ws_ship_hdemo_sk = household_demographics.hd_demo_sk " +
"               AND ws_web_page_sk = web_page.wp_web_page_sk " +
"               AND time_dim.t_am_pm_AM=1 " +
"               AND household_demographics.hd_dep_count < 5 " +
"               AND web_page.wp_char_count BETWEEN 250000 AND 500000) at, " +
"       (SELECT Count(*) AS pmc " +
"        FROM   web_sales, " +
"               household_demographics, " +
"               time_dim, " +
"               web_page " +
"        WHERE  ws_sold_time_sk = time_dim.t_time_sk " +
"               AND ws_ship_hdemo_sk = household_demographics.hd_demo_sk " +
"               AND ws_web_page_sk = web_page.wp_web_page_sk " +
"               AND time_dim.t_am_pm_PM=1 " +
"               AND household_demographics.hd_dep_count < 5 " +
"               AND web_page.wp_char_count BETWEEN 250000 AND 500000) pt " +
"ORDER  BY am_pm_ratio;   ";


// BOOL QUERY 91
final static String bq91 =
"  SELECT cc_call_center_id AS Call_Center, " +
"       cc_name           AS Call_Center_Name, " +
"       cc_manager        AS Manager, " +
"       Sum(cr_net_loss)  AS Returns_Loss " +
" FROM   call_center, " +
"       catalog_returns, " +
```

```
"       date_dim, " +
"       customer, " +
"       customer_address, " +
"       customer_demographics, " +
"       household_demographics " +
" WHERE  cr_call_center_sk = cc_call_center_sk " +
"       AND cr_returned_date_sk = d_date_sk " +
"       AND cr_returning_customer_sk = c_customer_sk " +
"       AND cd_demo_sk = c_current_cdemo_sk " +
"       AND hd_demo_sk = c_current_hdemo_sk " +
"       AND ca_address_sk = c_current_addr_sk " +
"       AND d_year > 2000 " +
"       AND (d_month_name_January=1 OR d_month_name_February=1 OR  " +
" d_month_name_March=1 OR d_month_name_April=1 OR " +
"            d_month_name_May=1 OR d_month_name_June=1) " +
"       AND ( ( cd_marital_status_single=1 " +
"               AND cd_education_status_secondary=1 ) " +
"              OR ( cd_marital_status_married=1 " +
"                   AND cd_education_status_postgraduate=1 ) ) " +
"       AND ca_gmt_offset > 500 " +
" GROUP  BY cc_call_center_id, " +
"          cc_name, " +
"          cc_manager, " +
"          cd_marital_status_single,cd_marital_status_married,cd_marital_status_divorced, " +
"          cd_education_status_primary,cd_education_status_secondary, " +
"          cd_education_status_university,cd_education_status_postgraduate " +
" ORDER  BY Sum(cr_net_loss) DESC;   ";

//QUERY TABLES FOR QUERY MAKER.....................................................
private final  String[] basic_queries
= { q2,q6,q7,q10_1,q10_2,
q11,q13,q15,q17,q19,
q26,q30,q34,q39A,q39B,
q42,q43,q45,q46,q48,
q49,q50,q52,q55,q64,
q66,q69_1,q69_2,q71,q72,q79,
q85,q90,q91};

private final String[] bool_queries = {
bq2,bq6,bq7,bq10_1,bq10_2,
bq11,bq13,bq15,bq17,bq19,
bq26,bq30,bq34,bq39A,bq39B,
bq42,bq43,bq45,bq46,bq48,
bq49,bq50,bq52,bq55,bq64,
bq66,bq69_1,bq69_2,bq71,bq72,bq79,
bq85,bq90,bq91
};

public String[] getBasic_queries() {
return basic_queries;
}

public String[] getBool_queries() {
return bool_queries;
}

}
```

TimeKeeper.java

```
package thesis;


public class TimeKeeper {

	// This is just for help, during the build process
	public void timer(long start, long end){
		long totalTime = end - start;
		long hours = totalTime / (60 * 60 * 1000);
		totalTime = totalTime % (60 * 60 * 1000);
		long mins = totalTime / (60 * 1000);
		totalTime = totalTime % (60 * 1000);
		long secs = totalTime / 1000;
		String m,s;
		if(mins<10){
			m = "0" + mins;
		}
		else{
			m = "" + mins;
		}
		if(secs<10){
			s = "0" + secs;
		}
		else{
			s = "" + secs;
		}
		System.out.println("TIME >>>>>>>>>>>>>>>>>: " + hours + ":" + m + ":" + s);
	}

	//This is an actual timer, with strict values and record keeping for the charts
	public static long strictTimer(long start, long end){
		long totalTime = end - start;
		return totalTime;
	}
}
```

GraphMaker.java

```
package thesis;



import com.smartxls.ChartFormat;
import com.smartxls.ChartShape;
import com.smartxls.RangeStyle;
import com.smartxls.WorkBook;

import java.awt.*;

public class GraphMaker {

// BACKUP THE ORIGINAL TIME.XLS FILE BEFORE RUNNING !!!!!!!!!!!!!!
/*
* Before run,
* specify
* 1 root directory (XLS_PATH) where QueryMaker results must be (TIMES_FILE_NAME)
* 4 sub-directories(TIME_PATH, DIFF_PATH, VAL_PATH, AVG_PATH)
*
* Before rerun,
* delete everything from the XLS_PATH except for the BACKUP_FILE_NAME
* and rename BACKUP_FILE_NAME to TIMES_FILE_NAME
*
* */

//The LOOPS parameter from query maker
static final int LOOPS = 25;

//Number of queries
static final int QUERIES = 34;

//Path to root excel directory
static final String XLS_PATH =
"C:\\ excels\\";

//Directory for png files with graphs of times FOR EACH QUERY
static final String TIME_PATH =
XLS_PATH + "time\\";

//Directory for png files with graphs of % time differences FOR EACH QUERY
static final String DIF_PATH =
XLS_PATH + "time-dif\\";

//Directory for png files with the time and time dif values FOR EACH QUERY
static final String VAL_PATH =
XLS_PATH + "values\\";

//Directory for png file with graph of the average % time difference OVER ALL QUERIES
static final String AVG_PATH =
XLS_PATH + "average\\";

//Excel File name
static final String TIMES_FILE_NAME = "times.xls";

//New excel file with time differences
static final String DIF_FILE_NAME = "timedifs.xls";

//New excel file with averages
static final String AVG_FILE_NAME = "average.xls";
```

```
//Backup of times.xls
static final String BACKUP_FILE_NAME = "backup_times.xls";

private static void backupTimes() throws Exception{

WorkBook times = new WorkBook();
times.read(XLS_PATH + TIMES_FILE_NAME);
times.write(XLS_PATH + BACKUP_FILE_NAME);

}

private static void calculateTimeDifs() throws Exception {
// Calculates the time differences FOR EACH QUERY
// [boolTime - basicTime]
// and as a percentage of the basicTime
// ([boolTime - basicTime]/basicTime) * 100

WorkBook wold = new WorkBook();
wold.read(XLS_PATH + TIMES_FILE_NAME);
WorkBook wnew = new WorkBook();
int numSheets = wold.getNumSheets();
wnew.setNumSheets(numSheets);

for(int i=0;i<numSheets;i++){
wnew.CopySheetFromBook(wold, i, i);
wnew.setText(i, 0, 2, "Dif");
wnew.setText(i, 0, 3, "Dif %");

//Calculate time differences
for(int j=1;j<=LOOPS;j++){

long tbool = Integer.valueOf(wold.getText(j,1));
long tbasic = Integer.valueOf(wold.getText(j,0));

long tdif = tbool - tbasic;
double tdifpercent = ((tbool - tbasic)*1.0/tbasic)*100;

wnew.setNumber(i, j, 2, tdif);
wnew.setNumber(i, j, 3, tdifpercent);

}

//Align cells
RangeStyle rangeStyle = wnew.getRangeStyle(0, 0, LOOPS, 3);
rangeStyle.setHorizontalAlignment(RangeStyle.HorizontalAlignmentRight);
rangeStyle.setVerticalAlignment(RangeStyle.VerticalAlignmentCenter);
wnew.setRangeStyle(rangeStyle, 0, 0, LOOPS, 3);
}


wnew.write(XLS_PATH + DIF_FILE_NAME);
}

private static void calculateAverage() throws Exception {
// Calculates the average time difference (percentage) for EACH query
// [SumOverAllLoops(timeDifferencePercentage)/LOOPS ]

WorkBook wold = new WorkBook();
wold.read(XLS_PATH + DIF_FILE_NAME);
WorkBook wnew = new WorkBook();
wnew.setNumSheets(1);
wnew.setText(0, 0, 0, "QUERY");
wnew.setText(0, 0, 1, "AVG.T.DIF");
```

```
WorkBook old = new WorkBook();
old.read(XLS_PATH + DIF_FILE_NAME);


double avg;
for(int i=0;i<QUERIES;i++){
old.setSheet(i);

//Calculation of average value
avg = 0;
for(int j=0;j<LOOPS;j++){
avg += old.getNumber(j+1,3);
}
avg /= LOOPS;

wnew.setText(0, i + 1, 0, "q" + String.valueOf(i+1));
wnew.setNumber(0, i + 1, 1, avg);
}

RangeStyle rangeStyle = wnew.getRangeStyle(0, 0,QUERIES , 2);
rangeStyle.setHorizontalAlignment(RangeStyle.HorizontalAlignmentRight);
rangeStyle.setVerticalAlignment(RangeStyle.VerticalAlignmentCenter);
wnew.setRangeStyle(rangeStyle, 0, 0, QUERIES, 2);

wnew.write(XLS_PATH + AVG_FILE_NAME);
}

private static void takeValuePics() throws Exception {
WorkBook w = new WorkBook();
w.read(XLS_PATH + DIF_FILE_NAME);
int numSheets = w.getNumSheets();
String out;
for(int i=0;i<numSheets;i++) {
w.setSheet(i);
w.setPrintGridLines(true);
w.setPrintScale(100);
out = VAL_PATH + "values_" + w.getSheetName(i) + ".png";
w.sheetRangeToImage(0,0,LOOPS,3,out);
}
}

private static void makeTimeCharts(){
WorkBook workBook = new WorkBook();
try {
workBook.read(XLS_PATH + TIMES_FILE_NAME);
} catch (Exception e) {
e.printStackTrace();
}

int sheetsNum = workBook.getNumSheets();
for(int i=0;i<sheetsNum;i++){
try {
workBook.setSheet(i);
String sheetName = workBook.getSheetName(i);

int left = 5;
int right = 20;
int top = 1;
int bottom = 20;

ChartShape chart = workBook.addChart(left,top,right,bottom);
```

```
chart.setChartType(ChartShape.Column);

String linkRange = sheetName + "!$A$2:$B$" + String.valueOf(LOOPS+1);
chart.setLinkRange(linkRange, false);

chart.setAxisTitle(ChartShape.XAxis, 0, "Loop");
chart.setAxisTitle(ChartShape.YAxis, 0, "Time");

ChartFormat seriesformat = chart.getSeriesFormat(0);
seriesformat.setSolid();
seriesformat.setForeColor(Color.BLUE.getRGB());
chart.setSeriesFormat(0, seriesformat);

seriesformat = chart.getSeriesFormat(1);
seriesformat.setSolid();
seriesformat.setForeColor(Color.RED.getRGB());
chart.setSeriesFormat(1, seriesformat);

chart.setSeriesName(0, "Basic");
chart.setSeriesName(1, "Booleanized");
chart.setTitle(sheetName);

ChartFormat titleformat = chart.getTitleFormat();
titleformat.setFontSize(14 * 20);
titleformat.setFontUnderline(true);
chart.setTitleFormat(titleformat);

workBook.writeXLSX(XLS_PATH + TIMES_FILE_NAME);

java.io.FileOutputStream out =
new java.io.FileOutputStream(TIME_PATH + "image_" + sheetName + ".png");
chart.writeChartAsPNG(workBook, out);
out.close();


} catch (Exception e) {
e.printStackTrace();
}
}
}

private static void makeTimeDifferenceCharts(){
WorkBook workBook = new WorkBook();
try {
workBook.read(XLS_PATH + DIF_FILE_NAME);
} catch (Exception e) {
e.printStackTrace();
}

int sheetsNum = workBook.getNumSheets();
for(int i=0;i<sheetsNum;i++){
try {
workBook.setSheet(i);
String sheetName = workBook.getSheetName(i);

int left = 5;
int right = 20;
int top = 1;
int bottom = 20;
ChartShape chart = workBook.addChart(left, top, right, bottom);

chart.setChartType(ChartShape.Line);
```

```
String linkRange = sheetName + "!$D$2:$D$" + String.valueOf(LOOPS+1);
chart.setLinkRange(linkRange, false);

chart.setAxisTitle(ChartShape.XAxis, 0, "Loop");
chart.setAxisTitle(ChartShape.YAxis, 0, "TimeDifference %");

ChartFormat seriesformat = chart.getSeriesFormat(0);
seriesformat.setLineColor(Color.BLACK.getRGB());
seriesformat.setSolid();
chart.setSeriesFormat(0, seriesformat);


chart.setSeriesName(0, "Time Difference %");
chart.setTitle(sheetName);

ChartFormat titleformat = chart.getTitleFormat();
titleformat.setFontSize(14 * 20);
titleformat.setFontUnderline(true);
chart.setTitleFormat(titleformat);

workBook.writeXLSX(XLS_PATH + DIF_FILE_NAME);

java.io.FileOutputStream out =
new java.io.FileOutputStream(DIF_PATH  + "image_"+ sheetName + ".png");
chart.writeChartAsPNG(workBook, out);
out.close();


} catch (Exception e) {
e.printStackTrace();
}
}
}

private static void makeATDChart(){
WorkBook workBook = new WorkBook();
try {
workBook.read(XLS_PATH + AVG_FILE_NAME);
workBook.setSheet(0);
String sheetName = "Average Time Difference %";
workBook.setSheetName(0,sheetName);
int left = 5;
int right = 20;
int top = 1;
int bottom = 20;
ChartShape chart = workBook.addChart(left, top, right, bottom);
chart.setChartType(ChartShape.Line);

String linkRange = sheetName + "!$B$2:$B$" + String.valueOf(QUERIES+1);
chart.setLinkRange(linkRange, false);

chart.setAxisTitle(ChartShape.XAxis, 0, "QUERY");
chart.setAxisTitle(ChartShape.YAxis, 0, "AVERAGE TIME DIFFERENCE %");

ChartFormat seriesformat = chart.getSeriesFormat(0);
seriesformat.setLineColor(Color.BLACK.getRGB());
seriesformat.setSolid();
chart.setSeriesFormat(0, seriesformat);

chart.setSeriesName(0, "Average Time Difference %");
chart.setTitle(sheetName);

//set chart title's font property
```

```
ChartFormat titleformat = chart.getTitleFormat();
titleformat.setFontSize(14 * 20);
titleformat.setFontUnderline(true);
//titleformat.setTextRotation(90);
chart.setTitleFormat(titleformat);

workBook.writeXLSX(XLS_PATH + AVG_FILE_NAME);

java.io.FileOutputStream out =
new java.io.FileOutputStream(AVG_PATH  + "atd_image.png");
chart.writeChartAsPNG(workBook, out);
out.close();

} catch (Exception e) {
e.printStackTrace();
}
}


public static void main(String[] args) throws Exception {

System.out.println("Backing up QueryMaker results @ " + XLS_PATH + BACKUP_FILE_NAME);
backupTimes();

System.out.println("Calculating time differences @ " + XLS_PATH + DIF_FILE_NAME);
calculateTimeDifs();

System.out.println("Calculating averages @ " + XLS_PATH + AVG_FILE_NAME);
calculateAverage();

System.out.println("Making png images of the time values @ " + VAL_PATH);
takeValuePics();

System.out.println("Making running time charts and png images @ " + TIME_PATH);
makeTimeCharts();

System.out.println("Making time difference charts and png images @ " + DIF_PATH);
makeTimeDifferenceCharts();

System.out.println("Making average time difference charts and png images @ " + AVG_PATH);
makeATDChart();

System.out.println("All Done!");

}

}
```