ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Υλοποίηση benchmark για σύγκριση απόδοσης βάσεων δεδομένων

Της φοιτήτριας

Sotirovska Biljana

Αρ. Μητρώου: 04/2720

Επιβλέπων καθηγητής

Αντώνης Σιδηρόπουλος

Θεσσαλονίκη 2012

ΠΡΟΛΟΓΟΣ

Στην επιστήμη των υπολογιστών παρουσιάζεται πολύ συχνά η ανάγκη της επιλογής μίας τεχνολογίας -είτε hardware είτε software- μεταξύ ενός συνόλου παρεμφερών τεχνολογιών. Τα χαρακτηριστικά της κάθε τεχνολογίας δεν είναι απαραιτήτως ευδιάκριτα, με αποτέλεσμα να μην είναι πάντα εύκολη η επιλογή της αποτελεσματικότερης τεχνολογίας. Η έννοια της αποτελεσματικότητας είναι σχετική και εξαρτάται από τις εκάστοτε απαιτήσεις όπως για παράδειγμα ασφάλεια, αξιοπιστία, ευχρηστία κτλ. Ένα επίσης πολύ σημαντικό κριτήριο αποτελεσματικότητας είναι η ταχύτητα εκτέλεσης ενός έργου. Ένα συνηθισμένο παράδειγμα είναι οι επεξεργαστές και οι κάρτες γραφικών. Αυτά τα δύο ήδη υλικού, ενώ ενσωματώνουν αρκετές υπο-τεχνολογίες (OpenGL, Shader Model, AMD PowerNow κτλ) το σημαντικότερο κριτήριο απόδοσης τους (που είναι και το ζητούμενο) είναι η ταχύτητα εκτέλεσης του έργου που καλούνται να υλοποιήσουν.

Πώς όμως ένας υποψήφιος αγοραστής ή χρήστης μπορεί να αποφασίσει το ποιο είναι το γρηγορότερο προϊόν ή υπηρεσία από αυτές που έχει να επιλέξει; Ειδικά αν λάβουμε υπ' όψιν μας τον έντονο ανταγωνισμό που υπάρχει μεταξύ των εταιριών υψηλής τεχνολογίας, αυτή η επιλογή γίνεται ακόμα δυσκολότερη. Η πιο συνηθισμένη λύση είναι η σύγκριση των προϊόντων επί το έργο χωρίς όμως να παρέχει πλήρη και ακριβή αποτελέσματα καθώς, όπως αναφέρθηκε, οι δυνατότητες είναι παρεμφερείς και άρα δυσδιάκριτες. Είναι προφανές ότι απαιτείται η χρήση ενός εργαλείου λήψης αποφάσεων που να εξυπηρετεί το σκοπό αυτό. Τα εργαλεία αυτά ονομάζονται benchmarks, τα οποία μπορούν να μετρήσουν με ακρίβεια τα μεγέθη από τα οποία εξαρτάται η απόδοση των προϊόντων όπως για παράδειγμα τα frames per second, τα flops, τα bit per second κοκ.

Ο όρος benchmark πιθανότατα να προήλθε από τη διαδικασία μέτρησης των διαστάσεων ενός αντικειμένου (marking) πάνω σε πάγκο εργασίας (bench). Υπάρχει η -ανεπιβεβαίωτη- φήμη ότι ο όρος αυτός χρησιμοποιήθηκε για πρώτη φορά από τους τσαγκάρηδες για να πάρουν τα μέτρα για τα παπούτσια των πελατών τους. Τα benchmarks χρησιμοποιούνται για μέτρηση επιδόσεων εξάγοντας έναν -συνήθως αυθαίρετο- δείκτη επίδοσης. Συγκρίνοντας τους δείκτες από το σύνολο των αποτελεσμάτων έχουμε μια σχετική εικόνα για το αποδοτικότερο από τα υποκείμενα της σύγκρισης. Υπάρχουν διαφόρων ειδών benchmarks όπως οικονομικά, ενεργειακά, επιχειρησιακά κα. Στην επιστήμη των ηλεκτρονικών υπολογιστών, το benchmark περιορίζεται στην εκτέλεση ενός προγράμματος ή ενός συνόλου προγραμμάτων, προκειμένου να αξιολογηθεί η σχετική απόδοση μιας εφαρμογής ή ενός εξαρτήματος hardware.

Το benchmark πρέπει να έχει τη δυνατότητα της παραμετροποίησης του τρόπου εκτέλεσης του ώστε να μπορεί να εξάγει αποτελέσματα με ή χωρίς τη χρήση συγκεκριμένων υποσυστημάτων της υπό δοκιμή τεχνολογίας. Για να γίνει πιο κατανοητό αυτό ας πάρουμε ως παράδειγμα τις κάρτες γραφικών οι οποίες έχουν δυνατότητες που μπορούν να απενεργοποιηθούν on demand όπως για παράδειγμα antialiasing, anisotropic filtering, gamma correction κ.α. Οι προαναφερθείσες λειτουργίες μπορούν να επιδράσουν δραματικά την απόδοση της κάρτας και η δυνατότητα απενεργοποίησής τους μπορεί να μας δώσει μια σφαιρικότερη αντίληψη των επιδόσεων της κάρτας.

Για τη μέτρηση επίδοσης των συστημάτων rdbms το κυριότερο και ίσως το μοναδικό μέγεθος ενδιαφέροντος είναι ο χρόνος εκτέλεσης των ερωτημάτων. Μπορεί ο χρόνος αυτός να φαίνεται εξ αρχής αμελητέος για τη σημερινή τεχνολογία αλλά ας φανταστούμε τι θα γινόταν αν σε μια εφορία ή σε ένα κέντρο τεχνικής υποστήριξης μιας εταιρίας, ένα συνηθισμένο query χρειαζόταν 5 λεπτά για να δώσει αποτελέσματα. Θα ήταν καταστροφικό για όλους όσους θα έπρεπε να αναμένουν όλον αυτό το χρόνο χωρίς ουσιαστικά να κάνουν τίποτα. Στις περιπτώσεις αυτές, ακόμα και τα δευτερόλεπτα παίζουν ρόλο. Οι διαχειριστές βάσεων δεδομένων καλούνται, πριν την έναρξη ενός εγχειρήματος να επιλέξουν τον κατάλληλο συνδυασμό rdbms - υπολογιστή. Συνδυασμός που κατά πάσα πιθανότητα θα ακολουθεί την εταιρία ή τον οργανισμό για πολλά χρόνια μετά, ίσως και καθ' όλη τη διάρκεια της ύπαρξής τους. Μια λάθος επιλογή μπορεί να κοστίσει σε χρήματα και χρόνο για τη μετέπειτα μεταφορά των δεδομένων και των ερωτημάτων σε μια νέα τεχνολογία.

Στο σημείο αυτό εισέρχεται η έννοια του benchmark στο χώρο των βάσεων δεδομένων. Ο διαχειριστής της βάσης έχει τη δυνατότητα, πριν την τελική επιλογή, να εγκαταστήσει στο hardware της επιλογής του ένα rdbms υπό δοκιμή και να εισάγει σ αυτό δεδομένα και να εκτελέσει queries στο database schema που προβλέπεται να διαχειριστεί η εταιρία ή ο οργανισμός. Ως αποτέλεσμα δημιουργείται ένα περιβάλλον δοκιμών εξειδικευμένο στις απαιτήσεις των χρηστών του συστήματος με υψηλότερη αξιοπιστία των εξαγόμενων αποτελεσμάτων.

ΠΕΡΙΛΗΨΗ

Στην παρούσα πτυχιακή εργασία αναλύεται η διαδικασία δημιουργίας ενός benchmark για συστήματα σχεσιακών βάσεων δεδομένων σε γλώσσα προγραμματισμού Java. Η εργασία εστιάζει σε όλα τα βήματα της δημιουργίας του benchmark. Αυτό συμπεριλαμβάνει τη δημιουργία του περιβάλλοντος δοκιμών, την ανάπτυξη της εφαρμογής benchmark, την εξαγωγή των αποτελεσμάτων και την απεικόνιση αυτών σε διαγράμματα και τη βελτιστοποίηση των ερωτημάτων με μεγάλους χρόνους εκτέλεσης.

Η δημιουργία του περιβάλλοντος δοκιμών πραγματεύεται τις διαδικασίες του partitioning, της εγκατάστασης των λειτουργικών συστημάτων και των rdbms και της εισαγωγής των δεδομένων. Επίσης μελετάται η δυνατότητα απενεργοποίησης των υπηρεσιών του λειτουργικού συστήματος για την επιτάχυνση των διαδικασιών. Τα υπό δοκιμή rdbms είναι MySql 5, Postgresql 9, IBM DB2 σε λειτουργικά συστήματα Linux και Windows και SqlServer 2008 R2 σε λειτουργικό σύστημα Windows. Η βάση δεδομένων που χρησιμοποιήθηκε είναι αυτή της TPC-H.

Η εφαρμογή benchmark χρησιμοποιεί τεχνολογία JDBC για να συνδεθεί με το εκάστοτε rdbms. Ανάλογα με το configuration που επιλέγει ο χρήστης, η εφαρμογή συνδέεται στον database server και εκτελεί ερωτήματα επαναληπτικά, καταγράφοντας τους χρόνους εκτέλεσης.

Οι χρόνοι που προκύπτουν από την εκτέλεση των δοκιμών, διαβάζονται από βοηθητική εφαρμογή γραμμένη σε Java Swing και απεικονίζονται με τη μορφή ραβδογραμμάτων. Υπάρχει δυνατότητα αποθήκευσης των διαγραμμάτων ως εικόνες για περαιτέρω χρήση και εισαγωγή στη βιβλιογραφία.

Τέλος αναλύεται η διαδικασία βελτιστοποίησης των ερωτημάτων με τη χρήση εργαλείων όπως ο query optimizer και ο query planner που παρέχονται στο κάθε rdbms. Η προσέγγιση και η τεχνική βελτιστοποίησης διαφέρει ανάλογα με τη φύση του ερωτήματος. Η βελτιστοποίηση γίνεται επιλεκτικά για τα ερωτήματα με τη μεγαλύτερη καθυστέρηση και όχι για όλα τα ερωτήματα από τη στιγμή που η βελτιστοποίηση δεν είναι αυτοσκοπός της εργασίας.

ABSTRACT

This thesis analyzes the procedure of creating a benchmark for relational database management systems in Java programming language. The paper gives focus to all the steps of the benchmark creation. These steps include creation of the testing environment, benchmark development, extraction of the testing results and diagrammatic result depiction and finally slow query optimization.

Testing environment creation deals with the processes of partitioning, operating system and RDBMS installation and data insertion. The ability of deactivating the operating system services in order to speed up procedures is also studied. The RDBMS under testing are MySql 5, Postgresql 9, IBM DB2 tested on Linux and Windows and SqlServer 2008 R2 tested on Windows. TPC-H database was used for the testing purposes.

The benchmark application uses JDBC technology to connect on each RDBMS. Depending on the configuration defined by the user, the application connects to the database server, executes queries iteratively and stores the execution times.

The times obtained from the tests execution are read by an aiding application developed in Java Swing and are depicted as bar-charts. The application provides the capability of storing the diagrams as images for further use in documentation.

Finally, the query optimization procedure is analyzed with the use of tools like query optimizer and query planner provided for each RDBMS. The approach and the technique followed varies depending the query's nature. The optimization occurs selectively for the slower queries and not for all of them since optimization is not the subject of the thesis.

Κατάλογος περιεχομένων

ΠΡΟΛΟΓΟΣ....2
ΠΕΡΙΛΗΨΗ....4
ABSTRACT....5
ΕΙΣΑΓΩΓΗ....11
ΚΕΦΑΛΑΙΟ 1: Επιλογή βάσης δεδομένων....12
ΕΙΣΑΓΩΓΗ....12
1.1 Η βάση δεδομένων TPC-H....12
1.2 Η βάση δεδομένων Employees....14
1.3 Η βάση δεδομένων World....15
1.4 Τελική επιλογή....16
ΕΠΙΛΟΓΟΣ....16
ΚΕΦΑΛΑΙΟ 2: Δημιουργία του περιβάλλοντος πειραματισμού....17
ΕΙΣΑΓΩΓΗ....17
2.1 Ο υπολογιστής....17
2.2 Τα λειτουργικά συστήματα και το λογισμικό....18
MySql....18
Postgres....20
DB2....21
SQL Server....21
2.3 Δημιουργία indexes....23
2.4 Απενεργοποίηση περιττών υπηρεσιών στα Windows....25
ΕΠΙΛΟΓΟΣ....26

ΚΕΦΑΛΑΙΟ 3: Εκτέλεση των δοκιμών 27
ΕΙΣΑΓΩΓΗ.....27
3.1 Περιγραφή των ερωτημάτων.....27
Ερμηνεία των queries.....28
Query 1.....28
Query 2.....29
Query 3.....30
Query 4.....31
Query 5.....32
Query 6.....32
Query 7.....33
Query 8.....34
Query 9.....35
Query 10.....36
Query 11.....37
Query 12.....38
Query 13.....39
Query 14.....39
Query 15.....40
Query 16.....41
Query 17.....42
Query 18.....43
Query 19.....44
Query 20.....45

Query 21....46
Query 22....47
Inserts....48
Updates....48
3.2 Εφαρμογή εκτέλεσης των ερωτημάτων....49
3.3 Εξαγωγή και αναπαράσταση των αποτελεσμάτων....51
Εξαγωγή αποτελεσμάτων....51
Δημιουργία διαγραμμάτων....52
ΕΠΙΛΟΓΟΣ....52
ΚΕΦΑΛΑΙΟ 4: Query optimization σε MySql και Postgres....53
ΕΙΣΑΓΩΓΗ....53
4.1 Query optimization στη MySql....53
4.2 Query optimization στην Postgres....58
ΕΠΙΛΟΓΟΣ....62
ΚΕΦΑΛΑΙΟ 5: Εργαλεία κώδικα σε Java....63
ΕΙΣΑΓΩΓΗ....63
5.1 Εργαλεία τροποποίησης δεδομένων....63
5.2 Εργαλείο εξαγωγής διαγραμμάτων....66
ΕΠΙΛΟΓΟΣ....67
ΚΕΦΑΛΑΙΟ 6: Αποτελέσματα δοκιμών....68
ΕΙΣΑΓΩΓΗ....68
6.1 Σύγκριση RDBMS ανά OS και ανά μέγεθος βάσης για το ερώτημα 4....68
6.2 Σύγκριση όλων των RDBMS μεταξύ τους χωρίς indexes....71
6.3 Σύγκριση όλων των RDBMS μεταξύ τους με τη χρήση indexes....73

6.4 Διαγράμματα αποτελεσμάτων για Inserts.....74

6.5 Διαγράμματα αποτελεσμάτων για Updates.....76

ΕΠΙΛΟΓΟΣ.....77

ΣΥΜΠΕΡΑΣΜΑΤΑ.....78

ΒΙΒΛΙΟΓΡΑΦΙΑ.....79

ΟΔΗΓΟΣ ΧΡΗΣΗΣ ΛΟΓΙΣΜΙΚΟΥ.....80

Ευρετήριο πινάκων

Πίνακας 1: Μορφή αρχείου παραμετροποίησης.....49

Πίνακας 2: MySql EXPLAIN EXTENDED.....55

Ευρετήριο σχημάτων

Σχήμα 1: DB2 Q4 σύγκριση μεγεθών και λειτουργικών συστημάτων.....68

Σχήμα 2: MySql Q4 σύγκριση μεγεθών και λειτουργικών συστημάτων.....69

Σχήμα 3: Postgres Q4 σύγκριση μεγεθών και λειτουργικών συστημάτων.....69

Σχήμα 4: SqlServer 2008 Q4 σύγκριση μεγεθών.....70

Σχήμα 5: Σύγκριση όλων των RDBMS για όλα τα OS για τη μικρή βάση.....71

Σχήμα 6: Σύγκριση όλων των RDBMS για όλα τα OS για τη μεσαία βάση.....71

Σχήμα 7: Σύγκριση όλων των RDBMS για όλα τα OS για τη μεγάλη βάση.....72

Σχήμα 8: Σύγκριση όλων των RDBMS για όλα τα OS για τη μικρή βάση (Indexed) 73

Σχήμα 9: Σύγκριση όλων των RDBMS για όλα τα OS για τη μεσαία βάση (Indexed) 73

Σχήμα 10: Σύγκριση όλων των RDBMS για όλα τα OS για τη μεγάλη βάση (Indexed).....73

Σχήμα 11: Αποτελέσματα INSER INTO για τη μικρή βάση για τον πίνακα orders..74

Σχήμα 12: Αποτελέσματα INSER INTO για τη μεσαία βάση για τον πίνακα orders 74

Σχήμα 13: Αποτελέσματα INSER INTO για τη μεγάλη βάση για τον πίνακα orders 75

Σχήμα 14: Αποτελέσματα UPDATE για τη μεγάλη βάση για τον πίνακα lineitem...76

Σχήμα 15: Αποτελέσματα UPDATE για τη μεγάλη βάση για τον πίνακα orders.....76

Σχήμα 16: Αποτελέσματα UPDATE για τη μεγάλη βάση για τον πίνακα part.....76

Σχήμα 17: Αποτελέσματα UPDATE για τη μεγάλη βάση για τον πίνακα supplier...76

Σχήμα 18: Το interface της εφαρμογής εξαγωγής διαγραμμάτων.....80

ΕΙΣΑΓΩΓΗ

Σκοπός της παρούσας πτυχιακής είναι η παρουσίαση της συνολικής διαδικασίας ανάπτυξης, εκτέλεσης και εξαγωγής αποτελεσμάτων (ερμηνεία και αναπαράσταση αυτών) ενός rdbms benchmark, από τη δημιουργία του περιβάλλοντος ανάπτυξης μέχρι και τη βελτιστοποίηση των ερωτημάτων.

Η εκτέλεση δοκιμών πραγματοποιήθηκε πάνω σε τέσσερα rdbms (MySql 5, Postgresql 9, IBM DB2 και SqlServer 2008 R2) στα λειτουργικά συστήματα Windows και Linux, για τρία διαφορετικά μεγέθη βάσεων. Στα κεφάλαια που ακολουθούν θα δούμε αναλυτικά τις διαδικασίες εγκατάστασης του κάθε rdbms στο εκάστοτε λειτουργικό σύστημα, τις βάσεις και τα ερωτήματα που χρησιμοποιήθηκαν καθώς και το software που παράχθηκε για την εκτέλεση του benchmark και την σύγκριση των τελικών αποτελεσμάτων. Αναλυτικότερα:

- Στο πρώτο κεφάλαιο συγκρίνουμε τις υποψήφιες βάσεις και επιλέγουμε την καταλληλότερη.
- Στο δεύτερο κεφάλαιο δημιουργούμε το περιβάλλον δοκιμών (partitioning, εγκατάσταση λειτουργικών συστημάτων, εγκατάσταση rdbms, δημιουργία βάσεων δεδομένων).
- Στο τρίτο κεφάλαιο παραθέτουμε τις λεπτομέρειες εκτέλεσης των δοκιμών και βλέπουμε τα queries που χρησιμοποιήθηκαν αναλυτικά.
- Στο τέταρτο κεφάλαιο περιγράφεται η διαδικασία βελτιστοποίησης ορισμένων ερωτημάτων σε rdbms με πολύ άσχημες επιδώσεις.
- Στο πέμπτο και τελευταίο κεφάλαιο αναλύονται τα εργαλεία σε java που χρησιμοποιήθηκαν για σκοπούς πέραν της εκτέλεσης του benchmark.

ΚΕΦΑΛΑΙΟ 1: Επιλογή βάσης δεδομένων

ΕΙΣΑΓΩΓΗ

Στο πρώτο κεφάλαιο αναλύουμε τρεις βάσεις δεδομένων, οι δημοφιλέστερες που μπορεί να βρει κανείς στο διαδίκτυο. Αυτές είναι: η TPC-H, η οποία είναι του οργανισμού TPC και είναι φτιαγμένη αποκλειστικά για benchmarks. Η Employees την οποία προτείνει η MySql για την εκτέλεση δοκιμών. Τέλος η World η οποία είναι άλλη μια βάση δεδομένων της MySql η οποία συγκεντρώνει τα ονόματα πόλεων και χωρών του κόσμου.

1.1 Η βάση δεδομένων TPC-H

Το Transaction Processing Performance Council (TPC) είναι ένας μη κερδοσκοπικός οργανισμός που ιδρύθηκε το 1988 με σκοπό να καθορίσει την επεξεργασία των συναλλαγών (transaction processing) και των δοκιμών επιδόσεων (benchmarks) και να διαδώσει αξιόπιστα αποτελέσματα στη βιομηχανία. Απαρτίζεται από 20 εταιρίες μεταξύ των οποίων η Cisco, Microsoft, Intel, Oracle και IBM.

Η εταιρία έχει καθορίσει 9 benchmarks από τα οποία μόνο τα 3 είναι σε ισχύ ενώ τα υπόλοιπα έχουν χαρακτηριστεί παρωχημένα. Τα benchmarks που είναι σε ισχύ είναι τα: TPC-C, TPC-E και TPC-H. Τα TPC-C και TPC-E είναι OLTP benchmarks. Το TPC-H benchmark είναι benchmark υποστήριξης αποφάσεων (decision support). Περιλαμβάνει μία πλήρη σουίτα από ερωτήματα, δεδομένα καθώς και βοηθητικά εργαλεία. Η κοινοπραξία που καθόρισε το TPC-H πρότυπο ισχυρίζεται ότι τα ερωτήματα και τα δεδομένα που συμπληρώνουν τη βάση δεδομένων έχουν επιλεγεί για να έχουν σχέση με ευρύ φάσμα της αγοράς, διατηρώντας παράλληλα σε επαρκή βαθμό την ευκολία εφαρμογής. Το benchmark αντικατοπτρίζει τα συστήματα υποστήριξης αποφάσεων που:

- Εξετάζουν μεγάλους όγκους δεδομένων
- Εκτελούν ερωτήματα με υψηλό βαθμό πολυπλοκότητας
- Δίνουν απαντήσεις σε κρίσιμα ερωτήματα των επιχειρήσεων.

Η βάση δεδομένων συμπεριλαμβάνει 8 πίνακες: PART, PARTSUPP, LINEITEM, ORDERS, CUSTOMERS, SUPPLIER, NATION και REGION.

Τα πεδία που περιλαμβάνονται στους πίνακες είναι:

NATION: n_nationkey(**primary key**), n_name, n_regionkey, n_comment

REGION: r_regionkey(**primary key**), r_name, r_comment

SUPPLIER: s_suppkey(**primary key**), s_name, s_address, s_nationkey(**foreign key to NATION**), s_phone, s_acctbal, s_comment

CUSTOMER: c_custkey(**primary key**), c_name, c_address, c_nationkey(**foreign key to NATION**), c_phone, c_acctbal, c_mktsegment, c_comment

PART: p_partkey(**primary key**), p_name, p_mfgr, p_brand, p_type, p_size, p_container, p_retailprice, p_comment

PARTSUPP: ps_partkey(**foreign key to PARTKEY**), ps_suppkey(**foreign key to SUPPLIER**), ps_availqty, ps_supplycost, ps_comment, ps_partkey & ps_suppkey(**primary key**)

ORDERS: o_orderkey(**primary key**), o_custkey(**foreign key to CUSTOMER**), o_orderstatus, o_totalprice, o_orderdate, o_orderpriority, o_clerk, o_shippriority, o_comment

LINEITEM: l_orderkey(**primary key & foreign key to ORDERS**), l_ partkey(**foreign key to PARTSUPP**), l_suppkey(**foreign key to PARTSUPP**), l_linenumber(**primary key**), l_quantity, l_extendedprice, l_discount, l_tax, l_returnflag, l_linestatus, l_shipdate, l_commitdate, l_receiptdate, l_shipinstruct, l_shipmode, l_comment

Η καταχώρηση στη βάση δεδομένων γίνεται με τη χρήση δύο εργαλείων που παρέχονται απ το TPC-H των DBGEN και QGen. Τα προγράμματα αυτά είναι γραμμένα σε ANSI C για λόγους φορητότητας. Από το DBGEN δημιουργούνται οι πίνακες των οποίων το μέγεθος εξαρτάται από παράμετρο στη γραμμή εντολών. Υποστηρίζονται μεγέθη 1GB, 10GB, 30GB, 100GB, 300GB, 1000GB, 3000GB, 10000GB, 30000GB, 100000GB. Το QGen χρησιμοποιείται για την εξαγωγή των ερωτημάτων. Τα ερωτήματα πρέπει να είναι γραμμένα σε μία εμπορικά διαθέσιμη υλοποίηση της SQL. Επειδή το τελευταίο ISO SQL πρότυπο (ISO / IEC 9075:1992), δεν έχει ακόμη εφαρμοστεί πλήρως από τους περισσότερους κατασκευαστές και δεδομένου ότι η SQL γλώσσα συνεχώς εξελίσσεται, η προδιαγραφή του TPC-H benchmark περιλαμβάνει έναν αριθμό των επιτρεπόμενων αποκλίσεων από τους επίσημους ορισμούς των ερωτημάτων.

1.2 Η βάση δεδομένων Employees

Η βάση δεδομένων Employees αναπτύχθηκε από τους Patrick Crews και Giuseppe Maxia και παρέχει βάση δεδομένων μεγέθους περίπου 160MB, κατανεμημένη σε έξι ξεχωριστούς πίνακες και αποτελείται από 4 εκατομμύρια εγγραφές συνολικά. Η βάση είναι διαθέσιμη στην ιστοσελίδα https://launchpad.net/test-db/ και διανέμεται σε μορφή tar.bz2. Περιέχονται οι εντολές δημιουργίας πινάκων για mysql, καθώς και τα δεδομένα σε μορφή εντολών INSERT INTO. Επιπλέον περιέχονται τροποποιημένες εντολές δημιουργίας πινάκων οι οποίες περιλαμβάνουν partitioning των πινάκων titles και salaries βάσει των ημερομηνιών έναρξης. Τέλος περιλαμβάνονται και αρχεία εντολές για την εξακρίβωση της ακεραιότητας των δεδομένων με αλγορίθμους md5 και sha.

Στο διανεμόμενο πακέτο δεν περιέχεται κάποιο βοηθητικό εργαλείο. Το μέγεθος της βάσης είναι 167 MB και περιέχει 300,000 εγγραφές υπαλλήλων με 2.8 εκατομμύρια εγγραφές μισθοδοσίας. Η τεκμηρίωση της βάσης περιορίζεται στα στοιχειώδη με ένα pdf 3 σελίδων διαθέσιμο από την τοποθεσία http://downloads.mysql.com/docs/employee-en.a4.pdf Η βάση διατίθεται υπό την άδεια Creative Commons Attribution-Share Alike 3.0 Unported License.

Οι πίνακες της βάσης είναι οι εξής:

Employees: emp_no(**primary key**), birth_date, first_name, last_name, gender, hire_date

Salaries: emp_no(**primary key & foreign key to Employees**), salary, from_date(**primary key**), to_date

Titles: emp_no(**primary key & foreign key to Employee**), title(**primary key**), from_date(**primary key**), to_date

Departments: dept_no(**primary key**), dept_name

Dept_Emp: emp_no(**primary key & foreign key to Employee**), dept_no(**primary key & foreign key to Departments**), from_date, to_date

Dept_Manager: dept_no(**primary key & foreign key to Departments**), emp_no(**primary key & foreign key to Employee**), from_date, to_date

1.3 Η βάση δεδομένων World

Η world database είναι μια ακόμα δοκιμαστική βάση δεδομένων για mysql διαθέσιμη στην ιστοσελίδα http://dev.mysql.com/doc/index-other.html Τα δεδομένα είναι διαθέσιμα ως ένα σύνολο από τρεις πίνακες και περιγράφουν το σύνολο των χωρών, των πόλεων και τον γλωσσών που ομιλούνται στον κόσμο:

- Country: Πληροφορίες για τις χώρες του κόσμου.
- City: Πληροφορίες για μερικές από τις πόλεις στις χώρες αυτές.
- CountryLanguage: γλώσσες που ομιλούνται σε κάθε χώρα.

Τα αρχεία εγκατάστασης για τη δημιουργία της βάσης δεδομένων World είναι διαθέσιμα για την μηχανή αποθήκευσης MyISAM ή InnoDB:

- world.sql: περιέχει προτάσεις SQL για να δημιουργηθεί η βάση δεδομένων που χρησιμοποιεί MyISAM πίνακες.
- world_innodb.sql: οι προτάσεις είναι παρόμοιες αλλά δημιουργούν πίνακες InnoDB. Οι στήλες είναι ίδιες με εκείνες των πινάκων MyISAM, αλλά οι InnoDB πίνακες εφαρμόζουν επίσης ξένα κλειδιά μεταξύ των πινάκων.

City: id(**primary key**), name, country_code, district, population

Country: code(**primary key**), name, continent, region, surfaceArea, indepYear, population, lifeExpectancy, GNP, GNPOld, localName, governmentForm, headOfState, capital, code2

CountryLanguage: country_code(**primary key & foreign key to Country**), language, isOfficial, percentage

1.4 Τελική επιλογή

Η επιλογή έγινε βάσει του υλικού που διέθετε η κάθε βάση. Από αυτή την άποψη η TPC-H είναι η πληρέστερη καθώς διαθέτει αρχείο με το σχήμα της βάσης, δεδομένα, ερωτήματα και δεδομένα για insert into και ως εκ τούτου κρίθηκε ότι είναι η καταλληλότερη από τις προαναφερθείσες βάσεις δεδομένων. Καθ ότι τα πειράματα έπρεπε να εκτελεστούν σε 3 διαφορετικά μεγέθη το γεγονός ότι η TPC-H έχει τη δυνατότητα να εξάγει δεδομένα μεγέθους (μεταξύ άλλων) 1, 10 και 30 GB, την έκανε πιο ελκυστική υποψήφια. Μειονέκτημά της είναι ότι το μικρότερο μέγεθος δεδομένων που μπορεί να εξάγει το dbgen είναι 1GB, μειονέκτημα που αποδείχθηκε εκ των υστέρων, καθώς τα περισσότερα από τα δοκιμαζόμενα rdbms αντιμετώπισαν σοβαρότατο πρόβλημα επιδόσεων με το εν λόγω μέγεθος.

Αρχικά είχε επιλεγεί ως μέγεθος δεδομένων για τη μικρή βάση αυτό του ενός GB, rdbms όπως η MySql και η Postgres παρουσίασαν πολύ άσχημες επιδόσεις σε ορισμένα ερωτήματα με χρόνους που να αγγίζουν και να ξεπερνούν τις 2 ημέρες για μία επανάληψη. Για το λόγο αυτό έπρεπε είτε να τροποποιηθεί το dbgen ώστε να μπορεί να εξάγει μικρότερα μεγέθη, είτε να τροποποιηθούν τα δεδομένα του ενός GB ώστε να προκύψουν μικρότερες βάσεις δεδομένων χωρίς να παραβιάζεται η αναφορική ακεραιότητα των τελικών δεδομένων. Επειδή και οι δύο τεχνικές καταλήγουν στο ίδιο αποτέλεσμα, δλδ την απώλεια του TPC-H compliance, και επειδή η τροποποίηση ενός προγράμματος σε C ξεφεύγει από τους σκοπούς της πτυχιακής, επιλέχθηκε η δεύτερη λύση. Τα τελικά μεγέθη μετά από τη σμίκρυνση έχουν ως εξής: περίπου 300MB για τη μεγάλη βάση, 135MB για τη μεσαία και 65MB για τη μικρή.

ΕΠΙΛΟΓΟΣ

Στο κεφάλαιο αυτό αναλύθηκαν τα χαρακτηριστικά των τριών υποψηφίων βάσεων και επιλέχθηκε η πληρέστερη βάση, η οποία είναι η TPC-H. Στο επόμενο κεφάλαιο θα αναλύεται η διαδικασία δημιουργίας του περιβάλλοντος δοκιμών και εγκατάστασης τη βάση που μόλις επιλέχθηκε.

ΚΕΦΑΛΑΙΟ 2: Δημιουργία του περιβάλλοντος πειραματισμού

ΕΙΣΑΓΩΓΗ

Στο κεφάλαιο αυτό παρακολουθούμε διεξοδικά τη διαδικασία εγκατάστασης των λειτουργικών συστημάτων σε εφτά partitions (τέσσερα Windows και τρία Linux), την εγκατάσταση του κάθε rdbms και τη δημιουργία της βάσης για τρία διαφορετικά μεγέθη.

2.1 Ο υπολογιστής

Για την εκτέλεση των πειραμάτων χρησιμοποιήθηκε ένας φορητός υπολογιστής Fujitsu-Siemens κατασκευής 2004 με τα εξής τεχνικά χαρακτηριστικά:

Επεξεργαστής: AMD Athlon 64 3400+
Μνήμη 1 GB Single channel DDR
Σκληρός δίσκος WD 250 GB PATA

Ο λόγος για τον οποίο χρησιμοποιήθηκε υπολογιστής παλιάς τεχνολογίας είναι για να προκύψουν μεγαλύτεροι χρόνοι από τις δοκιμές ώστε να είναι ευκολότερη η μέτρηση των χρόνων και να είναι πιο ευδιάκριτη η διαφορά των επιδόσεων μεταξύ των rdbms.

Στο σημείο αυτό αξίζει να αναφερθεί ότι η αρχική επιλογή ήταν ένας φορητός υπολογιστής μάρκας Acer στον οποίο είχαν εγκατασταθεί όλα τα λειτουργικά συστήματα και τα απαραίτητα εργαλεία, και ενώ είχε ξεκινήσει η διαδικασία εξαγωγής των αποτελεσμάτων, παρουσιάστηκε βλάβη στη μητρική πλακέτα. Το εξουσιοδοτημένο συνεργείο για να αντικαταστήσει τη μητρική πλακέτα ζητούσε 300 ευρώ. Η επισκευή ενός τόσο παλιού μηχανήματος κρίθηκε οικονομικά ασύμφορη οπότε χρειάστηκε η ανεύρεση ενός άλλου μηχανήματος - του προαναφερθέντος στην περίπτωσή μας. Στο μηχάνημα αυτό όπως ήταν φυσικό δεν αρκούσε η απλή μεταφορά του δίσκου οπότε έπρεπε να γίνει εγκατάσταση όλων των λειτουργικών συστημάτων, των προγραμμάτων και των οδηγών και να ξεκινήσει η εκτέλεση των δοκιμών από την αρχή. Όλα τα παραπάνω φυσικά έγιναν με το αντίστοιχο κόστος σε χρόνο, το οποίο δεν είναι ευκαταφρόνητο.

2.2 Τα λειτουργικά συστήματα και το λογισμικό

Στον υπολογιστή δοκιμών εγκαταστάθηκαν εφτά (7) λειτουργικά συστήματα σε εφτά partitions. Με τον τρόπο αυτό η επίδοση του κάθε rdbms δεν επηρεάζεται από τις τυχών ενεργές υπηρεσίες των υπολοίπων rdbms. Για το partitioning δημιουργήθηκε ένα primary partition μεγέθους 35 GB και ένα extended στον υπόλοιπο χώρο. Στο extended partition δημιουργήθηκαν όλα τα υπόλοιπα λογικά διαμερίσματα. Τα τέσσερα partitions των windows (NTFS) τοποθετήθηκαν πρώτα και έχουν μέγεθος 35GB. Τα partitions για Linux (ext3) τοποθετήθηκαν τελευταία και έχουν μέγεθος 33GB τα δύο πρώτα και 25GB το τελευταίο. Στο τέλος δεσμεύτηκαν 1000MB για Linux swap.

Όλα τα συστήματα βάσεων δεδομένων εγκαταστάθηκαν σε 2 λειτουργικά, Windows XP SP2 και Ubuntu Linux 11.10 με εξαίρεση τον SQL Server ο οποίος όπως ήταν αναμενόμενο δεν εγκαθίσταται σε Linux. Η δημιουργία των βάσεων δεδομένων καθώς και η συμπλήρωση με δεδομένα έγινε με τη χρήση αρχείων δέσμης (bat) για windows και shell scripts για Linux. Το κάθε αρχείο περιλάμβανε τις εντολές για την κατασκευή των πινάκων, την εισαγωγή των δεδομένων και την κατασκευή των κύριων και ξένων κλειδιών. Λόγω των διαφορών που υπάρχουν στο κάθε rdbms, το κάθε αρχείο δέσμης έπρεπε να περιλαμβάνει διαφορετικές εντολές, παρόλο που το σχήμα της βάσης ήταν σε κάθε περίπτωση ίδιο. Παρακάτω ακολουθεί αναλυτική περιγραφή για το τι χρειάστηκε να γίνει προκειμένου να δημιουργηθεί το περιβάλλον πειραματισμού για κάθε rdbms για κάθε λειτουργικό σύστημα.

MySql

Είναι το πιο διαδεδομένο σύστημα βάσεων δεδομένων στον κόσμο, το οποίο λειτουργεί ως διακομιστής που παρέχει πρόσβαση σε πολλούς χρήστες σε μια σειρά από βάσεις δεδομένων. Το όνομά της το πήρε από την κόρη του συνιδρυτή Michael Widenius, My. Το μεγαλύτερο μέρος της MySql είναι open source με τον πηγαίο κώδικα να είναι διαθέσιμος υπό την άδεια χρήσης GPL v2. Τα κλειστά κομμάτια του κώδικα είναι υπό καθεστώς EULA. Οι εφαρμογές που χρησιμοποιούν βάσεις δεδομένων MySQL περιλαμβάνουν: Joomla, WordPress, phpBB, Drupal και άλλο λογισμικό χτισμένο πάνω στη στοίβα λογισμικού LAMP[1]. MySQL χρησιμοποιείται επίσης σε πολλές υψηλού προφίλ, μεγάλης κλίμακας δικτυακές υπηρεσίες, συμπεριλαμβανομένης της Wikipedia, του Google, του Facebook και του Twitter. Είναι γραμμένη σε C και C++.

1 Το τετράπτυχο Linux, Apache, MySql και PHP είναι γνωστό με τα αρχικά LAMP

Στη MySql υπάρχει η επιλογή μεταξύ δύο συστημάτων αποθήκευσης δεδομένων των innodb και myisam. Η βασική τους διαφορά είναι ότι ενώ το innodb είναι πολύ αργότερο σε σχέση με το myisam, το innodb παρέχει constraint check και είναι transactional αντίθετα με το myisam που δεν έχει αυτές τις δυνατότητες. Τα transactions διασφαλίζουν την εκτέλεση queries τα οποία είναι συνεχόμενα και το ένα εξαρτάται από το άλλο. Ένα άλλο χρήσιμο χαρακτηριστικό του innodb είναι το row-level locking με το οποίο αποφεύγονται προβλήματα που δημιουργούνται όταν δύο queries τύπου insert ή update εκτελούνται την ίδια χρονική στιγμή και είναι χρήσιμο σε high traffic enviroment. Για επαγγελματική χρήση οι εν λόγω δυνατότητες του innodb θεωρούνται απαραίτητες και γι αυτό η χρήση του γρηγορότερου myisam αποκλείστηκε από την πρώτη στιγμή.

Κατά τη διάρκεια της εγκατάστασης παρατηρήθηκαν διαφορές μεταξύ των δύο λειτουργικών συστημάτων οι οποίες συνοψίζονται παρακάτω:

1. Σύστημα αποθήκευσης των δεδομένων: Στα Windows το προεπιλεγμένο σύστημα είναι το innodb ενώ στο Linux το myisam.
2. Το προεπιλεγμένο μέγεθος της cache στα Windows είναι 0 ενώ στο Linux 16MB.
3. Στο Linux τα ονόματα των πινάκων και των πεδίων είναι case sensitive
4. Για την εισαγωγή των δεδομένων σε Linux από command line απαιτείται η χρήση του προσδιοριστή local αμέσως μετά την εντολή Load data. Για παράδειγμα

```
Load data local infile <file name> into <table name> fields
                terminated by "<delimiter>"
```

Για να καταστεί δυνατή η απομακρυσμένη σύνδεση στη MySql απαιτείται η εκτέλεση της παρακάτω εντολής η οποία είναι κοινή και στα 2 λειτουργικά συστήματα.

```
Grant all privileges on *.* to '<username>'@'<address>' identified
                        by "<password>"
```

Όπου:

το *.* σημαίνει ότι η εντολή έχει εφαρμογή σε όλες τις βάσεις
το username είναι το όνομα χρήστη που πρέπει να χρησιμοποιήσει ο απομακρυσμένος υπολογιστής για να συνδεθεί (καθορίζεται εκείνη τη στιγμή, δεν προϋπάρχει)
το address είναι είτε η ip διεύθυνση είτε το DNS του απομακρυσμένου υπολογιστή

Postgres

Η PostgreSQL, είναι ένα αντικείμενο-σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων (ORDBMS) διαθέσιμο για πολλές πλατφόρμες, όπως Linux, FreeBSD, Solaris, Windows και Mac OS X. Διατίθεται βάσει της άδειας PostgreSQL και είναι ελεύθερο και ανοικτού κώδικα λογισμικό. Η PostgreSQL έχει αναπτυχθεί από την Παγκόσμια Ομάδα Ανάπτυξης PostgreSQL, που αποτελείται από ορισμένους εθελοντές που απασχολούνται και εποπτεύονται από εταιρείες όπως η Red Hat και η EnterpriseDB. Το Mac OS X, αρχίζοντας με την έκδοση LION, έχει το διακομιστή της PostgreSQL ως προεπιλεγμένη βάση δεδομένων στην έκδοση του διακομιστή και εργαλεία πελάτη PostgreSQL στην Desktop έκδοση. Η PostgreSQL είναι διαθέσιμη για τα ακόλουθα λειτουργικά συστήματα: Linux (όλες οι πρόσφατες διανομές), το Windows (Win2000 SP4 και νεότερα), FreeBSD, OpenBSD, το NetBSD, το Mac OS X, AIX, BSD / OS, IRIX, OpenSolaris, Solaris και πολλά άλλα ακόμα.
Η διαδικασία εγκατάστασης του συστήματος περιγράφεται στην επίσημη ιστοσελίδα www.postgresql.org/ .

Οι εντολές για τη δημιουργία των βάσεων δεδομένων στα Windows είναι:

```
SET PGDATABASE=<όνομα της βάσης>
SET PGUSER=<user>
SET PGPASSWORD=<password>
"C:\Program Files\PostgreSQL\9.1\bin\psql" -X –variable =
     ON_ERROR_STOP = -1 -w -f <όνομα αρχείου>
```

Οι εντολές για τη δημιουργία των βάσεων δεδομένων σε Linux:

```
export PGDATABASE=<όνομα της βάσης>
export PGUSER=<user>
export PGPASSWORD=<password>
/opt/PostgreSQL/9.1/bin/psql -X --variable=ON_ERROR_STOP= -1
     -w -f <όνομα αρχείου>/
```

Για να επιτραπεί απομακρυσμένη σύνδεση στην Postgres πρέπει να τροποποιήσουμε το pg_hba.conf και να προσθέσουμε τη γραμμή host all all 0.0.0.0/0 trust ώστε να επιτρέπεται σύνδεση από οποιαδήποτε διεύθυνση IP χωρίς password. Στις παλαιότερες εκδόσεις της postgres χρειαζόταν τροποποίηση και το αρχείο postgresql.conf ώστε να αφαιρεθεί το σχόλιο μπροστά από τη γραμμή listen_addresses = '*'. Στην τελευταία έκδοση το αρχείο είχε εκ προεπιλογής τη γραμμή εκτός σχολίων.

DB2

Η IBM DB2 είναι μια σχεσιακή βάση δεδομένων που αναπτύχθηκε από την IBM. Υπάρχουν τρία προϊόντα DB2 που είναι πολύ παρόμοια, αλλά όχι ταυτόσημα: DB2 για LUW (Linux, Unix, και Windows), DB2 για z / OS (mainframe), και DB2 για iSeries. Το προϊόν DB2 LUW τρέχει σε πολλαπλές διανομές Linux και UNIX, όπως το Red Hat Linux, SUSE Linux, AIX, HP / UX, και Solaris, και τα περισσότερα συστήματα Windows.

Εντολές για τη δημιουργία των βάσεων δεδομένων στα windows:

```
db2cmd db2 -s -n -t -f <όνομα αρχείου>
```

Εντολές για τη δημιουργία των βάσεων δεδομένων στο linux:

```
/opt/ibm/db2/V9.7/bin/db2 -s -n -t -f <όνομα αρχείου>
```

-s: η εκτέλεση θα τερματιστεί αν προκύψει σφάλμα

-n: αντικαθιστά το χαρακτήρα νέας γραμμής με το χαρακτήρα που ορίζεται από την παράμετρο -t. Χωρίς αυτή την παράμετρο το αρχείο πρέπει είτε να είναι όλο σε μία γραμμή ή η κάθε γραμμή να τελειώνει με backslahsh (\).

-t τοποθετεί στο τέλος κάθε γραμμής το semicolon (;) ως χαρακτήρα τερματισμού κάθε εντολής

-f καθορίζει από ποιο αρχείο θα διαβάσει το rdbms τις εντολές.

SQL Server

Ο SqlServer 2008 είναι μια σχεσιακή βάση δεδομένων, που αναπτύχθηκε από τη Microsoft. Υπάρχουν τουλάχιστον δώδεκα διαφορετικές εκδόσεις του Microsoft SQL Server που στοχεύουν σε διαφορετικά ακροατήρια και για διάφορα φορτία εργασίας (που κυμαίνονται από μικρές εφαρμογές που αποθηκεύουν και ανακτούν τα δεδομένα στον ίδιο υπολογιστή μέχρι εκατομμύρια χρήστες και υπολογιστές που έχουν πρόσβαση σε τεράστιες ποσότητες δεδομένων από το Internet την ίδια στιγμή).

Για την εγκατάσταση του SQL Server χρειάζεται .NET framework 3.5 ή νεότερο. Η εντολή MS-DOS για την ανάγνωση ενός αρχείου sql είναι:

```
sqlcmd –S <myServer>\<instance name> -i <όνομα αρχείου>
```

όπου –i αντιστοιχεί στο input δλδ. αρχείο εισόδου.

Τα βήματα που απαιτούνται ώστε να πραγματοποιηθεί η απομακρυσμένη σύνδεση είναι:

- Από το Management Studio του SqlServer, στον Object Explorer κάνουμε δεξί κλικ στον κόμβο της σύνδεσης στον SQL Server και επιλέγουμε

Properties. Επιλέγουμε τη σελίδα Security και ενεργοποιούμε την επιλογή “SQL Server authentication mode”.

- Επεκτείνουμε τους κόμβους που βρίσκονται κάτω από τον κόμβο της σύνδεσης στον SQL Server και επιλέγουμε από τον κόμβο Security τον κόμβο Logins και με δεξί κλικ πάνω στον κόμβο του χρήστη sa επιλέγουμε Properties.
- αποεπιλέγουμε το "Enforce password policy"
- στο ίδιο παράθυρο δίνουμε το password που θα χρησιμοποιηθεί για τις μετέπειτα συνδέσεις.
- στο παράθυρο των ιδιοτήτων του sa, αριστερά επιλέγουμε τη σελίδα “Status” και στο Login επιλέγουμε το “Enable”

Από το Configuration Manager

- επεκτείνουμε τον κόμβο SQL Server Network Configuration και επιλέγουμε το Protocols for SQLEXPRESS
- δεξί κλικ στο TCP/IP και επιλογή Enable
- δεξί κλικ στο TCP/IP και επιλογή Properties, καρτέλα IP Addresses
- στο IP1, επιλέγουμε “Yes” στο πεδίο Enabled και στο πεδίο IP Address βάζουμε τη διεύθυνση του localhost (127.0.0.1)
- στο IPAll, στο πεδίο TCP Port βάζουμε 1433

Για να ενεργοποιηθούν οι αλλαγές που κάναμε, κάνουμε επανεκκίνηση του server. Για να γίνει αυτό, επιλέγουμε τον κόμβο SQL Server Services και απ’ τις υπηρεσίες που εμφανίζονται στη δεξιά λίστα, κάνουμε δεξί κλικ στο SQL Server(SQLEXPRESS) και επιλέγουμε Restart.

2.3 Δημιουργία indexes

Το ευρετήριο (index) σε μια βάση δεδομένων είναι μια δομή δεδομένων που βελτιώνει την ταχύτητα των λειτουργιών ανάκτησης δεδομένων σε έναν πίνακα με το κόστος της βραδύτερης εγγραφής και του αυξημένου χώρου αποθήκευσης. Τα indexes μπορούν να δημιουργηθούν χρησιμοποιώντας μία ή περισσότερες στήλες σε έναν πίνακα, παρέχοντας τη δυνατότητα για γρήγορη τυχαία αναζήτηση και αποτελεσματική πρόσβαση σε ταξινομημένα δεδομένα.

Σε μια σχεσιακή βάση δεδομένων, το index είναι αντίγραφο ενός μέρους του πίνακα. Τα indexes μπορούν να δηλωθούν ως unique, exclusion, primary key και foreign key. Τα περισσότερα συστήματα βάσεων δεδομένων αυτόματα δημιουργούν indexes στις στήλες με τα κύρια κλειδιά.

Indexes είναι καλύτερο να ορίζονται σε στήλες που χρησιμοποιούνται συχνά σε where και σε order by. Αν η βάση είναι πιο πολύπλοκη, καλύτερο είναι να μην ορίζονται indexes σε στήλες που θα ενημερώνονται συχνά.

1. Part
    - p_brand
    - p_container
    - p_name
    - p_size
    - p_type
2. Partsupp
    - ps_availqty
    - ps_supplycost
3. Supplier
    - s_comment
4. Customer
    - c_acctbal
    - c_mktsegment
5. Nation
    - n_name

6. Region
    - r_name
7. Orders
    - o_comment
    - o_orderstatus
8. Lineitem
    - l_discount
    - l_extendedprice
    - l_quantity
    - l_returnflag
    - l_shipinstruct
    - l_shipmode

2.4 Απενεργοποίηση περιττών υπηρεσιών στα Windows

Μπορούμε να απενεργοποιήσουμε κάποιες υπηρεσίες όσες πραγματικά δεν πρόκειται ποτέ να χρειαστούμε, απελευθερώνοντας σεβαστή ποσότητα πόρων από το σύστημά μας έτσι ώστε να μην οδηγήσουν σε προβλήματα οποιαδήποτε απαίτηση κάθε χρήστη. Για να γίνει αυτό, ανοίγουμε το μενού από το Start κουμπί και πληκτρολογούμε services.msc στο Run. Κάνουμε δεξί κλικ στο όνομα μιας υπηρεσίας, επιλέγουμε properties (ιδιότητες), και μετά το αντίστοιχο σημείο από το startup type (τύπος εκκίνησης). Οι τιμές είναι τρεις: Automatic (Αυτόματη), Manual (Μη Αυτόματη) και Disabled (Απενεργοποίηση). Μερικές από αυτές τις υπηρεσίες είναι:

-Alerter (Υπηρεσία ειδοποίησης)

Αν ο υπολογιστής δεν ανήκει σε κάποιο δίκτυο ή δεν παίρνει ειδοποιήσεις από τον διαχειριστή, η υπηρεσία αυτή μπορεί να απενεργοποιηθεί.

-Automatic Updates (Αυτόματες ενημερώσεις)

Μπορεί να απενεργοποιηθεί μετά από μια πρώτη χειροκίνητη εκτέλεση του windows updater ώστε να ενημερωθεί το σύστημα με όλα τα critical updates.

-Clipbook (Πρόχειρες σελίδες)

Υπηρεσία που με την εγκατάσταση του SP2 από manual γίνεται disabled

-Error Reporting Service (Υπηρεσία αναφοράς σφαλμάτων)

Η υπηρεσία που αναλαμβάνει την αποστολή αναφοράς σφαλμάτων στη Microsoft. Μας είναι άχρηστη για την εκτέλεση των πειραμάτων.

-FastUser Switching Compatibility (Συμβατότητα FastUser Switching)

Αυτή η υπηρεσία είναι χρήσιμη όταν υπάρχουν περισσότεροι του ενός χρήστες που χρησιμοποιούν τον υπολογιστή. Από τη στιγμή που ο χρήστης είναι ένας, αυτή την υπηρεσία μπορούμε να την απενεργοποιήσουμε.

-Fax και Print Spooler (Fax και Ουρά εκτύπωσης)

Αυτές οι δύο υπηρεσίες είναι χρήσιμες αν στέλνουμε φαξ μέσω του μόντεμ ή έχουμε εκτυπωτή. Αν τίποτα απ τα δύο δεν μας ενδιαφέρει, μπορούμε να τις κλείσουμε.

-IMAPI CD-Burning COM (Υπηρεσία IMAPI CD Burning COM)

Εάν δεν χρησιμοποιούμε τη λειτουργία εγγραφής CD μέσω του Windows Explorer, μπορούμε να την απενεργοποιήσουμε. Δεν απενεργοποιούνται οι ιδιότητες εγγραφής που γίνονται μέσω άλλων προγραμμάτων.

-Indexing Service (Υπηρεσία ευρετηρίου)

Είναι η γνωστή υπηρεσία η οποία βελτιώνει τις ταχύτητες αναζήτησης όταν ψάχνουμε για κάποιο αρχείο ή φάκελο με το όνομά του. Καταναλώνει μεγάλους πόρους όμως ακόμα και όταν δεν την χρησιμοποιούμε. Για τους σκοπούς της πτυχιακής, αυτή η υπηρεσία απενεργοποιήθηκε από τη στιγμή που σε κάθε partiton υπάρχουν μόνο τα αρχεία με τα δεδομένα των τριών βάσεων δεδομένων. Απενεργοποιώντας αυτή την υπηρεσία δεν απενεργοποιείται η αναζήτηση στα Windows.

-Portable Media Serial Number (Υπηρεσία αριθμού σειράς φορητού στοιχείου πολυμέσων)

Σχετίζεται με πολυμέσα και DRM. Όπως είναι φυσικό, δεν χρειάστηκε στην πτυχιακή.

-Remote Desktop Help Session Manager και Remote Registry

Υπηρεσίες που σχετίζονται με απομακρυσμένη διαχείριση. Στον υπολογιστή που ανέλαβε το ρόλο του server υπήρχε φυσική πρόσβαση και για το λόγο αυτό οι δύο αυτές υπηρεσίες απενεργοποιήθηκαν.

-Smart Card

Διαχειρίζεται και ελέγχει την πρόσβαση σε μια έξυπνη κάρτα. Από τη στιγμή που δε χρησιμοποιούμε έξυπνες κάρτες για την πτυχιακή, η υπηρεσία είναι απενεργοποιημένη.

-Windows Image Acquisition (Λήψη εικόνας Windows)

Χρησιμοποιείται αν υπάρχει κάμερα ή σκάνερ αλλιώς μπορεί να την απενεργοποιήσουμε.

-Wireless Zero Configuration (Αρχική ρύθμιση παραμέτρων ασύρματης επικοινωνίας)

Χρησιμοποιείται για τη ρύθμιση συσκευών ασύρματου δικτύου. Δεν υπήρξε ανάγκη για ασύρματο δίκτυο οπότε την απενεργοποιήθηκε.

ΕΠΙΛΟΓΟΣ

Στο κεφάλαιο αυτό περιγράφηκε η διαδικασία εγκατάστασης των λειτουργικών συστημάτων και των απαραίτητων εφαρμογών. Τώρα είμαστε έτοιμοι να εκτελέσουμε τις δοκιμές, κάτι που θα παρουσιάσουμε στο επόμενο κεφάλαιο.

ΚΕΦΑΛΑΙΟ 3: Εκτέλεση των δοκιμών

ΕΙΣΑΓΩΓΗ

Σε αυτό το κεφάλαιο θα εξετάσουμε αναλυτικά τα 22 ερωτήματα που εξάγει το qgen της TPC-H, θα αναλύσουμε το λογισμικό java που αναπτύχθηκε για την εκτέλεση των δοκιμών και τέλος θα εξηγήσουμε τη μέθοδο εξαγωγής των διαγραμμάτων.

3.1 Περιγραφή των ερωτημάτων

Παρακάτω ακολουθεί η ερμηνεία των queries όπως αυτή δίνεται από το manual της TPC-H. Για ορισμένα από τα πεδία που συμμετέχουν στα queries δίνεται από την TPC-H η δυνατότητα επιλογής εύρους τιμών. Το εύρος αυτό ορίζεται στο manual. Όλα τα κύρια κλειδιά έχουν indexes. Για το ερώτημα 15 χρειάστηκε η δημιουργία view.

```
create view revenue (supplier_no, total_revenue) as
select
	l_suppkey,
	sum(l_extendedprice * (1 - l_discount))
from
	lineitem
where
	l_shipdate >= date ':1'
	and l_shipdate < date ':1' + interval '3' month
group by
	l_suppkey;
```

Ερμηνεία των queries

Query 1

Δίνει αναφορά για το εμπόρευμα που τιμολογήθηκε, εστάλη και επιστράφηκε:

```
select
    l_returnflag,
    l_linestatus,
    sum(l_quantity) as sum_qty,
    sum(l_extendedprice) as sum_base_price,
    sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
    sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge,
    avg(l_quantity) as avg_qty,
    avg(l_extendedprice) as avg_price,
    avg(l_discount) as avg_disc,
    count(*) as count_order
from
    lineitem
where
    l_shipdate <= date '1998-12-01' - interval ':1' day
group by
    l_returnflag,
    l_linestatus
order by
    l_returnflag,
    l_linestatus;
```

Η αναζήτηση γίνεται πάνω σε έναν πίνακα (lineitem), σε πεδίο που δεν περιέχει index. Γίνεται εκτεταμένη χρήση aggregation functions πάνω σε πεδία του πίνακα που είναι τύπου decimal. Τα αποτελέσματα ομαδοποιούνται και ταξινομούνται σύμφωνα με τα πεδία l_returnflag και l_linestatus.

Query 2

Βρίσκει ποιος προμηθευτής θα έπρεπε να επιλεγεί για την πραγματοποίηση παραγγελιών για ένα συγκεκριμένο εξάρτημα σε μια συγκεκριμένη περιοχή:

```
select
	s_acctbal,
	s_name,
	n_name,
	p_partkey,
	p_mfgr,
	s_address,
	s_phone,
	s_comment
from
	part,
	supplier,
	partsupp,
	nation,
	region
where
	p_partkey = ps_partkey
	and s_suppkey = ps_suppkey
	and p_size = :1
	and p_type like '%:2'
	and s_nationkey = n_nationkey
	and n_regionkey = r_regionkey
	and r_name = ':3'
	and ps_supplycost = (
		select
			min(ps_supplycost)
		from
			partsupp,
			supplier,
			nation,
			region
		where
			p_partkey = ps_partkey
			and s_suppkey = ps_suppkey
			and s_nationkey = n_nationkey
			and n_regionkey = r_regionkey
			and r_name = ':3'
	)
order by
	s_acctbal desc,
	n_name,
	s_name,
	p_partkey;
```

Η αναζήτηση περιλαμβάνει join σε πέντε πίνακες και κυρίως σε πεδία τύπου integer. Το ερώτημα περιέχει correlated εμφωλευμένο select το οποίο χρησιμοποιεί aggregation function και κάνει join στους ίδιους πίνακες όπως και το εξωτερικό select. Τα αποτελέσματα ταξινομούνται βάσει τεσσάρων πεδίων.

Query 3

Ανακτά τις πρώτες 10 παραγγελίες που δεν εστάλησαν σε μια συγκεκριμένη ημερομηνία, ταξινομημένες κατά φθίνουσα τιμή κόστους:

```
select
      l_orderkey,
      sum(l_extendedprice * (1 - l_discount)) as revenue,
      o_orderdate,
      o_shippriority
from
      customer,
      orders,
      lineitem
where
      c_mktsegment = ':1'
      and c_custkey = o_custkey
      and l_orderkey = o_orderkey
      and o_orderdate < date ':2'
      and l_shipdate > date ':2'
group by
      l_orderkey,
      o_orderdate,
      o_shippriority
order by
      revenue desc,
      o_orderdate;
```

Join σε 3 πίνακες σε πεδία τύπου integer και 2 πεδία date. Συνάθροιση ομαδοποιημένη στα πεδία l_orderkey, o_orderdate και o_shippriority.

Query 4

Δίνει μια εκτίμηση του πόσο καλά δουλεύει το σύστημα προτεραιότητας παραγγελιών και του πόσο ικανοποιημένοι είναι οι πελάτες.

```
select
	o_orderpriority,
	count(*) as order_count
from
	orders
where
	o_orderdate >= date ':1'
	and o_orderdate < date ':1' + interval '3' month
	and exists (
		select
			*
		from
			lineitem
		where
			l_orderkey = o_orderkey
			and l_commitdate < l_receiptdate
	)
group by
	o_orderpriority
order by
	o_orderpriority;
```

Χρησιμοποιεί έναν πίνακα (orders) και η αναζήτηση γίνεται βάσει ενός πεδίου τύπου date. Χρησιμοποιείται correlated εμφωλευμένο select το οποίο κάνει αναζήτηση στον πίνακα lineitem με τη χρήση πεδίων ημερομηνίας και join στον πίνακα orders του εξωτερικού ερωτήματος.

Query 5

Απαριθμεί τον όγκο των εσόδων που έγιναν μέσω των τοπικών προμηθευτών, μέσα σε ένα συγκεκριμένο έτος:

```
select
      n_name,
      sum(l_extendedprice * (1 - l_discount)) as revenue
from
      customer,
      orders,
      lineitem,
      supplier,
      nation,
      region
where
      c_custkey = o_custkey
      and l_orderkey = o_orderkey
      and l_suppkey = s_suppkey
      and c_nationkey = s_nationkey
      and s_nationkey = n_nationkey
      and n_regionkey = r_regionkey
      and r_name = ':1'
      and o_orderdate >= date ':2'
      and o_orderdate < date ':2' + interval '1' year
group by
      n_name
order by
      revenue desc;
```

Join σε 6 πίνακες, φιλτράρισμα του orders με βάση ημερομηνία και του region με βάση το κλειδί του. Εξαγωγή αθροισμάτων ομαδοποιημένα σύμφωνα με το όνομα έθνους. Ταξινόμηση βάση του αθροίσματος.

Query 6

Απεικονίζει το ποσό της αύξησης των εσόδων που θα προέκυπτε από την κατάργηση ορισμένων εκπτώσεων σε επίπεδο εταιρείας, σε ένα δεδομένο έτος. Αυτό το ερώτημα του τύπου "τι θα γινόταν αν" μπορεί να χρησιμοποιηθεί για να εξετάσει τρόπους για να αυξηθούν τα έσοδα:

```
select
      sum(l_extendedprice * l_discount) as revenue
from
      lineitem
where
      l_shipdate >= date ':1'
      and l_shipdate < date ':1' + interval '1' year
      and l_discount between :2 - 0.01 and :2 + 0.01
      and l_quantity < :3;
```

Φιλτράρισμα του πίνακα lineitem σε 3 πεδία (ένα ημερομηνίας και δύο float). Εξαγωγή αθροίσματος.

Query 7

Καθορίζει την αξία των εμπορευμάτων που μεταφέρονται μεταξύ ορισμένων κρατών για να βοηθήσουν στην επαναδιαπραγμάτευση των συμβάσεων αποστολής:

```
select
	supp_nation,
	cust_nation,
	l_year,
	sum(volume) as revenue
from
	(
		select
			n1.n_name as supp_nation,
			n2.n_name as cust_nation,
			extract(year from l_shipdate) as l_year,
			l_extendedprice * (1 - l_discount) as volume
		from
			supplier,
			lineitem,
			orders,
			customer,
			nation n1,
			nation n2
		where
			s_suppkey = l_suppkey
			and o_orderkey = l_orderkey
			and c_custkey = o_custkey
			and s_nationkey = n1.n_nationkey
			and c_nationkey = n2.n_nationkey
			and (
				(n1.n_name = ':1' and n2.n_name = ':2')
				or (n1.n_name = ':2' and n2.n_name = ':1')
			)
			and l_shipdate between date '1995-01-01' and date '1996-
12-31'
	) as shipping
group by
	supp_nation,
	cust_nation,
	l_year
order by
	supp_nation,
	cust_nation,
	l_year;
```

Στο εμφωλευμένο select γίνονται 5 συζεύξεις σε 6 πίνακες ο ένας εκ των οποίων (nation) χρησιμοποιείται δύο φορές. Το nation φιλτράρεται σύμφωνα με το n_name και το lineitem σύμφωνα με την ημερομηνία αποστολής (l_shipdate). Στο εξωτερικό ερώτημα χρησιμοποιήθηκαν αποκλειστικά τα αποτελέσματα του εσωτερικού και υπολογίζεται το άθροισμα ομαδοποιημένο και ταξινομημένο σύμφωνα με τα πεδία supp_nation, cust_nation και l_year.

Query 8

Καθορίζει το πόσο έχει αλλάξει το μερίδιο αγοράς ενός συγκεκριμένου έθνους σε μια συγκεκριμένη περιοχή μέσα σε δύο χρόνια, για ένα δεδομένο τύπο ανταλλακτικών:

```
select
     o_year,
     sum(case
          when nation = ':1' then volume
          else 0
     end) / sum(volume) as mkt_share
from
     (
          select
               extract(year from o_orderdate) as o_year,
               l_extendedprice * (1 - l_discount) as volume,
               n2.n_name as nation
          from
               part,
               supplier,
               lineitem,
               orders,
               customer,
               nation n1,
               nation n2,
               region
          where
               p_partkey = l_partkey
               and s_suppkey = l_suppkey
               and l_orderkey = o_orderkey
               and o_custkey = c_custkey
               and c_nationkey = n1.n_nationkey
               and n1.n_regionkey = r_regionkey
               and r_name = ':2'
               and s_nationkey = n2.n_nationkey
               and o_orderdate between date '1995-01-01'
                                   and date '1996-12-31'
               and p_type = ':3'
     ) as all_nations
group by
     o_year
order by
     o_year;
```

Στο εσωτερικό ερώτημα γίνεται join 8 πινάκων (το nation συμμετέχει δύο φορές). Ο πίνακας orders φιλτράρεται σύμφωνα με το o_orderdate, το part σύμφωνα με τον τύπο του (string) και το region σύμφωνα με το r_name. Το εξωτερικό ερώτημα χρησιμοποιεί το εσωτερικό στο from και πάνω στον πίνακα που προκύπτει κάνει συνάθροιση ομαδοποιημένη και ταξινομημένη πάνω στο o_year.

Query 9

Καθορίζει το κέρδος σε μια συγκεκριμένη σειρά εξαρτημάτων, χωρισμένο ανά έθνος, προμηθευτή και έτος:

```
select
    nation,
    o_year,
    sum(amount) as sum_profit
from
    (
    select
        n_name as nation,
        extract(year from o_orderdate) as o_year,
        l_extendedprice * (1 - l_discount) - ps_supplycost * l_quantity
as amount
    from
        part,
        supplier,
        lineitem,
        partsupp,
        orders,
        nation
    where
        s_suppkey = l_suppkey
        and ps_suppkey = l_suppkey
        and ps_partkey = l_partkey
        and p_partkey = l_partkey
        and o_orderkey = l_orderkey
        and s_nationkey = n_nationkey
        and p_name like '%:1%'
    ) as profit
group by
    nation,
    o_year
order by
    nation,
    o_year desc;
```

Στο εσωτερικό ερώτημα γίνεται join σε 6 πίνακες με τον πίνακα part να φιλτράρεται στο πεδίο p_name. Το εξωτερικό ερώτημα χρησιμοποιεί το εσωτερικό στο from και πάνω στον πίνακα που προκύπτει κάνει συνάθροιση ομαδοποιημένη και ταξινομημένη πάνω στα πεδία nation (από τον πίνακα profit) και o_year.

Query 10

Βρίσκει ποιοι πελάτες έχουν πρόβλημα με τα εξαρτήματα που τους εστάλησαν:

```
select
      c_custkey,
      c_name,
      sum(l_extendedprice * (1 - l_discount)) as revenue,
      c_acctbal,
      n_name,
      c_address,
      c_phone,
      c_comment
from
      customer,
      orders,
      lineitem,
      nation
where
      c_custkey = o_custkey
      and l_orderkey = o_orderkey
      and o_orderdate >= date ':1'
      and o_orderdate < date ':1' + interval '3' month
      and l_returnflag = 'R'
      and c_nationkey = n_nationkey
group by
      c_custkey,
      c_name,
      c_acctbal,
      c_phone,
      n_name,
      c_address,
      c_comment
order by
      revenue desc;
```

Join σε 4 πίνακες και φιλτράρισμα του orders σύμφωνα με την ημερομηνία παραγγελίας και του lineitem σύμφωνα με το l_returnflag. Υπολογίζεται συνάθροιση και γίνεται ομαδοποίηση πάνω στα πεδία c_custkey, c_name, c_acctbal, c_phone, n_name, c_address και c_comment. Τα αποτελέσματα ταξινομούνται με βάση το revenue που είναι το αποτέλεσμα της συνάθροισης.

Query 11

Βρίσκει το πιο σημαντικό υποσύνολο των αποθεμάτων των προμηθευτών σε ένα συγκεκριμένο έθνος

```
select
     ps_partkey,
     sum(ps_supplycost * ps_availqty) as value
from
     partsupp,
     supplier,
     nation
where
     ps_suppkey = s_suppkey
     and s_nationkey = n_nationkey
     and n_name = ':1'
group by
     ps_partkey having
          sum(ps_supplycost * ps_availqty) > (
               select
                    sum(ps_supplycost * ps_availqty) * :2
               from
                    partsupp,
                    supplier,
                    nation
               where
                    ps_suppkey = s_suppkey
                    and s_nationkey = n_nationkey
                    and n_name = ':1'
          )
order by
     value desc;
```

Στο εξωτερικό select γίνεται join σε 3 πίνακες. Ο πίνακας nation φιλτράρεται βάσει του n_name. Η ομαδοποίηση της συνάθροισης εξαρτάται από μια παράσταση στη συνιστώσα HAVING. Το εσωτερικό select εντός του HAVING, συντακτικά, κάνει τον ίδιο υπολογισμό που κάνει και το εξωτερικό ερώτημα (με τη διαφορά ότι πολλαπλασιάζει το αποτέλεσμα με ένα συντελεστή). Τα αποτελέσματα ταξινομούνται βάσει του value.

Query 12

Καθορίζει κατά πόσο η επιλογή λιγότερο ακριβών μέσων αποστολής επηρεάζει αρνητικά τις παραγγελίες με κρίσιμη προτεραιότητα, προκαλώντας καθυστερήσεις στις παραδόσεις:

```
select
	l_shipmode,
	sum(case
		when o_orderpriority = '1-URGENT'
			or o_orderpriority = '2-HIGH'
			then 1
		else 0
	end) as high_line_count,
	sum(case
		when o_orderpriority <> '1-URGENT'
			and o_orderpriority <> '2-HIGH'
			then 1
		else 0
	end) as low_line_count
from
	orders,
	lineitem
where
	o_orderkey = l_orderkey
	and l_shipmode in (':1', ':2')
	and l_commitdate < l_receiptdate
	and l_shipdate < l_commitdate
	and l_receiptdate >= date ':3'
	and l_receiptdate < date ':3' + interval '1' year
group by
	l_shipmode
order by
	l_shipmode;
```

Join 2 πινάκων, φιλτράρισμα του lineitem με βάση τα πεδία l_commitdate, l_receiptdate, l_shipmode και l_shipdate. Υπολογισμός 2 υπό συνθήκη συναθροίσεων και ομαδοποίηση και ταξινόμηση αυτών βάσει του l_shipmode.

Query 13

Ψάχνει τη σχέση μεταξύ των πελατών και του συνόλου των παραγγελιών τους:

```
select
     c_count,
     count(*) as custdist
from
     (
      select
           c_custkey,
           count(o_orderkey)
      from
           customer left outer join orders on c_custkey = o_custkey and
                                               o_comment not like '%:1%:2%'
      group by
           c_custkey
     ) as c_orders (c_custkey, c_count)
group by
     c_count
order by
     custdist desc,
     c_count desc;
```

Η αναζήτηση βασίζεται στο εσωτερικό select, το οποίο πραγματοποιεί left outer join σε 2 πίνακες. Το εξωτερικό ερώτημα χρησιμοποιεί το αποτέλεσμα της εσωτερικής απαρίθμησης για να ομαδοποιήσει τη δική του απαρίθμηση. Τα αποτελέσματα ταξινομούνται βάσει των πεδίων custdist και c_count.

Query 14

Παρακολουθεί την ανταπόκριση της αγοράς σε μια προώθηση όπως μια διαφήμιση ή ειδική εκστρατεία. Καθορίζει τι ποσοστό από τα έσοδα ενός έτους και μήνα έχουν προέλθει από εξαρτήματα προώθησης:

```
select
     100.00 * sum(case
          when p_type like 'PROMO%'
               then l_extendedprice * (1 - l_discount)
          else 0
     end) / sum(l_extendedprice * (1 - l_discount)) as promo_revenue
from
     lineitem,
     part
where
     l_partkey = p_partkey
     and l_shipdate >= date ':1'
     and l_shipdate < date ':1' + interval '1' month;
```

Join σε δύο πίνακες και φιλτράρισμα του lineitem βάσει του l_shipdate. Υπολογισμός δύο αθροισμάτων εκ των οποίων το ένα είναι υπό συνθήκη.

Query 15

Καθορίζει τον καλύτερο προμηθευτή ώστε να βραβευτεί, να πάρει περισσότερη δουλειά ή να επισημανθεί για ειδική αναγνώριση:

```
select
     s_suppkey,
     s_name,
     s_address,
     s_phone,
     total_revenue
from
     supplier,
     revenue
where
     s_suppkey = supplier_no
     and total_revenue = (
          select
               max(total_revenue)
          from
               revenue
     )
order by
     s_suppkey;
```

Join ενός πίνακα και του view revenue. Φιλτράρισμα βάσει του total_revenue. Η τιμή του φίλτρου υπολογίζεται από εσωτερικό ερώτημα το οποίο αναζητά το μέγιστο του revenue.total_revenue.

Query 16

Βρίσκει πόσοι προμηθευτές μπορούν να προμηθεύονται εξαρτήματα με συγκεκριμένα χαρακτηριστικά, π.χ. αν υπάρχει επαρκής αριθμός προμηθευτών για βαριά εξαρτήματα:

```
select
     p_brand,
     p_type,
     p_size,
     count(distinct ps_suppkey) as supplier_cnt
from
     partsupp,
     part
where
     p_partkey = ps_partkey
     and p_brand <> ':1'
     and p_type not like ':2%'
     and p_size in (:3, :4, :5, :6, :7, :8, :9, :10)
     and ps_suppkey not in (
          select
               s_suppkey
          from
               supplier
          where
               s_comment like '%Customer%Complaints%'
     )
group by
     p_brand,
     p_type,
     p_size
order by
     supplier_cnt desc,
     p_brand,
     p_type,
     p_size;
```

Join 2 πινάκων, φιλτράρισμα του part βάσει των πεδίων p_brand, p_type και p_size, και φιλτράρισμα του partsupp με anti-join σε εσωτερικό ερώτημα πάνω στον πίνακα supplier. Ο πίνακας supplier στο εσωτερικό ερώτημα φιλτράρεται σύμφωνα με το πεδίο s_comment. Στο εξωτερικό ερώτημα υπολογίζεται απαρίθμηση ομαδοποιημένη στα πεδία p_brand, p_type και p_size.

Query 17

Καθορίζει πόσα, κατά μέσο όρο, από τα ετήσια έσοδα θα χαθούν αν δεν πραγματοποιούνται πια παραγγελίες με μικρές ποσότητες συγκεκριμένων εξαρτημάτων:

```
select
     sum(l_extendedprice) / 7.0 as avg_yearly
from
     lineitem,
     part
where
     p_partkey = l_partkey
     and p_brand = ':1'
     and p_container = ':2'
     and l_quantity < (
          select
               0.2 * avg(l_quantity)
          from
               lineitem
          where
               l_partkey = p_partkey
     );
```

Το εξωτερικό ερώτημα κάνει join σε 2 πίνακες και φιλτράρει το part σύμφωνα με τα πεδία p_brand και p_container και το lineitem σύμφωνα με τα αποτελέσματα correlated εσωτερικού ερωτήματος. Το εσωτερικό ερώτημα συγκρίνει για κάθε γραμμή του lineitem του εξωτερικού ερωτήματος τη μέση τιμή του l_quantity για τις πλειάδες του lineitem για τις οποίες ισχύει η συνθήκη l_partkey = p_partkey.

Query 18

Ταξινομεί τους πελάτες βάσει του αν έχουν κάνει παραγγελία με μεγάλη ποσότητα:

```
select
      c_name,
      c_custkey,
      o_orderkey,
      o_orderdate,
      o_totalprice,
      sum(l_quantity)
from
      customer,
      orders,
      lineitem
where
      o_orderkey in (
            select
                  l_orderkey
            from
                  lineitem
            group by
                  l_orderkey having
                        sum(l_quantity) > :1
      )
      and c_custkey = o_custkey
      and o_orderkey = l_orderkey
group by
      c_name,
      c_custkey,
      o_orderkey,
      o_orderdate,
      o_totalprice
order by
      o_totalprice desc,
      o_orderdate;
```

Join σε 3 πίνακες. Το order φιλτράρεται με semi-join που προκύπτει από εσωτερικό (non-correlated) ερώτημα. Το εσωτερικό ερώτημα δημιουργεί μια λίστα από τα κλειδιά l_orderkey του πίνακα lineitem ομαδοποιημένα κατά το l_orderkey κατά τέτοιο τρόπο ώστε το άθροισμα l_quantity της κάθε ομάδας να ξεπερνά έναν αριθμό. Στο εξωτερικό ερώτημα υπολογίζεται άθροισμα ομαδοποιημένο βάσει των πεδίων c_name, c_custkey, o_orderkey, o_orderdate και o_totalprice. Τα αποτελέσματα ταξινομούνται κατά τα o_totalprice και o_orderdate.

Query 19

Αναφέρει τα μεικτά μειωμένα έσοδα που προέρχονται από την πώληση επιλεγμένων εξαρτημάτων, διεκπεραιωμένα με έναν συγκεκριμένο τρόπο:

```
select
     sum(l_extendedprice* (1 - l_discount)) as revenue
from
     lineitem,
     part
where
     (
          p_partkey = l_partkey
          and p_brand = ':1'
          and p_container in ('SM CASE', 'SM BOX', 'SM PACK', 'SM PKG')
          and l_quantity >= :4 and l_quantity <= :4 + 10
          and p_size between 1 and 5
          and l_shipmode in ('AIR', 'AIR REG')
          and l_shipinstruct = 'DELIVER IN PERSON'
     )
     or
     (
          p_partkey = l_partkey
          and p_brand = ':2'
          and p_container in ('MED BAG', 'MED BOX', 'MED PKG', 'MED
PACK')
          and l_quantity >= :5 and l_quantity <= :5 + 10
          and p_size between 1 and 10
          and l_shipmode in ('AIR', 'AIR REG')
          and l_shipinstruct = 'DELIVER IN PERSON'
     )
     or
     (
          p_partkey = l_partkey
          and p_brand = ':3'
          and p_container in ('LG CASE', 'LG BOX', 'LG PACK', 'LG PKG')
          and l_quantity >= :6 and l_quantity <= :6 + 10
          and p_size between 1 and 15
          and l_shipmode in ('AIR', 'AIR REG')
          and l_shipinstruct = 'DELIVER IN PERSON'
     );
```

Join σε δύο πίνακες και δημιουργία τριών ομάδων φιλτραρίσματος. Και στις τρεις ομάδες το φιλτράρισμα γίνεται στα ίδια πεδία τύπου string, integer και float.

Query 20

Βρίσκει τους προμηθευτές σε μια συγκεκριμένη χώρα που έχουν επιλεγμένα εξαρτήματα που μπορούν να δοθούν με προσφορά:

```
select
     s_name,
     s_address
from
     supplier,
     nation
where
     s_suppkey in (
          select
               ps_suppkey
          from
               partsupp
          where
               ps_partkey in (
                    select
                         p_partkey
                    from
                         part
                    where
                         p_name like ':1%'
               )
               and ps_availqty > (
                    select
                         0.5 * sum(l_quantity)
                    from
                         lineitem
                    where
                         l_partkey = ps_partkey
                         and l_suppkey = ps_suppkey
                         and l_shipdate >= date ':2'
                         and l_shipdate < date ':2' + interval '1' year
               )
     )
     and s_nationkey = n_nationkey
     and n_name = ':3'
order by
     s_name;
```

Στο εξωτερικό ερώτημα γίνεται join δύο πινάκων, ο πίνακας nation φιλτράρεται με βάσει το n_name. Ο πίνακας supplier φιλτράρεται με βάσει semi-join σε εμφωλευμένο ερώτημα. Το εμφωλευμένο ερώτημα φιλτράρει τον πίνακα partsupp βάσει των πεδίων ps_partkey και ps_availqty. Το ps_partkey φιλτράρεται με semi-join στον πίνακα part ο οποίος φιλτράρεται με το p_name. Το ps_availqty συγκρίνεται με την τιμή που επιστρέφει correlated εμφωλευμένο ερώτημα στο οποίο υπολογίζεται άθροισμα του l_quantity του πίνακα lineitem, ο οποίος φιλτράρεται με το πεδίο l_shipdate.

Query 21

Προσδιορίζει μερικούς προμηθευτές οι οποίοι δεν ήταν σε θέση να μεταφέρουν τα απαραίτητα εξαρτήματα σε εύθετο χρόνο:

```
select
	s_name,
	count(*) as numwait
from
	supplier,
	lineitem l1,
	orders,
	nation
where
	s_suppkey = l1.l_suppkey
	and o_orderkey = l1.l_orderkey
	and o_orderstatus = 'F'
	and l1.l_receiptdate > l1.l_commitdate
	and exists (
		select
			*
		from
			lineitem l2
		where
			l2.l_orderkey = l1.l_orderkey
			and l2.l_suppkey <> l1.l_suppkey
	)
	and not exists (
		select
			*
		from
			lineitem l3
		where
			l3.l_orderkey = l1.l_orderkey
			and l3.l_suppkey <> l1.l_suppkey
			and l3.l_receiptdate > l3.l_commitdate
	)
	and s_nationkey = n_nationkey
	and n_name = ':1'
group by
	s_name
order by
	numwait desc,
	s_name;
```

Join σε 4 πίνακες. Φιλτράρισμα του πίνακα orders με το πεδίο o_orderstatus και του nation με το πεδίο n_name. Ο πίνακας lineitem φιλτράρεται σύμφωνα με τα πεδία l_receiptdate και l_commitdate. Το αποτέλεσμα από το join των πινάκων nation, supplier και lineitem γίνεται semi-join με τον πίνακα lineitem l2. Το αποτέλεσμα του semi-join γίνεται anti-join με τον πίνακα lineitem l3. Τέλος το αποτέλεσμα του anti-join γίνεται join με τον πίνακα orders. Η απαρίθμηση των

πλειάδων γίνεται με βάση το πεδίο s_name. Τα αποτελέσματα ταξινομούνται βάσει του αποτελέσματος της απαρίθμησης (numwait) και του πεδίου s_name.

Query 22

Προσδιορίζει γεωγραφικές περιοχές όπου υπάρχουν πελάτες, οι οποίοι είναι πιθανό να κάνουν μια αγορά:

```
select
	cntrycode,
	count(*) as numcust,
	sum(c_acctbal) as totacctbal
from
	(
		select
			substring(c_phone from 1 for 2) as cntrycode,
			c_acctbal
		from
			customer
		where
			substring(c_phone from 1 for 2) in
				(':1', ':2', ':3', ':4', ':5', ':6', ':7')
			and c_acctbal > (
				select
					avg(c_acctbal)
				from
					customer
				where
					c_acctbal > 0.00
					and substring(c_phone from 1 for 2) in
						(':1', ':2', ':3', ':4', ':5', ':6', ':7')
			)
			and not exists (
				select
					*
				from
					orders
				where
					o_custkey = c_custkey
			)
	) as custsale
group by
	cntrycode
order by
	cntrycode;
```

Το εσωτερικό ερώτημα χρησιμοποιεί τον πίνακα customer φιλτράρεται με βάση τα πεδία c_phone και c_acctbal. Από το πεδίο c_phone εξάγονται οι δύο πρώτοι χαρακτήρες με τη χρήση substring οι οποίοι ανήκουν σε συγκεκριμένη λίστα τιμών. Το c_acctbal συγκρίνεται με την τιμή εσωτερικού ερωτήματος το οποίο επιστρέφει τη μέση τιμή του πεδίου c_acctbal για όλους τους customers με c_acctbal > 0 και κωδικό χώρας που ανήκει στην ίδια λίστα τιμών. Στο φιλτραρισμένο πίνακα

customer γίνεται anti-join με τον πίνακα orders ώστε να εξαιρεθούν οι πελάτες που δεν έχουν παραγγελίες. Τα αποτελέσματα του εσωτερικού ερωτήματος χρησιμοποιούνται από το εξωτερικό για υπολογισμό αθροίσματος και απαρίθμησης, ομαδοποιημένα και ταξινομημένα σύμφωνα με το πεδίο custsale.cntrycode.

Inserts

Τα δεδομένα για τα inserts εξάγονται από το dbgen.exe εκτελώντας την παρακάτω εντολή:

dbgen.exe –s 1 –U 4

Η εντολή αυτή εξάγει 4 αρχεία που περιέχουν τα δεδομένα για τα insert και ισάριθμα αρχεία με τα δεδομένα για τα delete, τα οποία περιέχουν μόνο το primary key των εγγραφών που εισήχθησαν από τα insert.

Τα αρχεία δεν περιέχουν τη σύνταξη για τα INSERT INTO αλλά μόνο τα δεδομένα τα οποία είναι γραμμογραφημένα και χωρισμένα με pipe(|). Για το λόγο αυτό απαιτήθηκε επεξεργασία ώστε να δημιουργηθούν νέα αρχεία στη μορφή του συντακτικού της SQL. Για τη διαδικασία αυτή δημιουργήθηκε εργαλείο σε java το οποίο να διεκπεραιώνει αυτή τη διαδικασία, αν και μετέπειτα πειραματισμός απέδειξε ότι και ένας text editor με προηγμένες λειτουργίες όπως το notepad++ μπορεί να χρησιμοποιηθεί για να επιτευχθεί το ίδιο αποτέλεσμα.

Από το σύνολο των περίπου πέντε χιλιάδων insert εκτελέστηκαν 50, μόνο για τον πίνακα orders. Το dbgen εξάγει αρχεία για τους πίνακες orders και lineitem αλλά λόγω του ότι έπρεπε να μειωθεί το μέγεθος των βάσεων για τους προαναφερθέντες λόγους, η διαδικασία επιλογής των κατάλληλων εγγραφών ώστε να μην παραβιάζεται η ακεραιότητα των δεδομένων ήταν αρκετά επίπονη και χρειάστηκε να γίνει ξανά με custom εργαλείο σε java.

Updates

Έτοιμες εντολές για update δεν παρέχονται από το πρότυπο και για το λόγο αυτό κατασκευάστηκαν μέσω κώδικα, 25 updates για τον καθέναν από τους πίνακες part, supplier, orders και lineitem. Οι εντολές χρησιμοποιήθηκαν περισσότερο στα πεδία που περιέχουν indexes. Για την εξαγωγή των δεδομένων χρησιμοποιήθηκε τυχαιοποίηση.

Για όλα τα java εργαλεία που χρησιμοποιήθηκαν για την εξαγωγή ή τη διαλογή των παραπάνω δεδομένων υπάρχει εκτενής επεξήγηση στο κεφάλαιο 5.1.

3.2 Εφαρμογή εκτέλεσης των ερωτημάτων

Η εκτέλεση των δοκιμών έγινε με τη χρήση προγράμματος γραμμής εντολών γραμμένο σε γλώσσα java. Το πρόγραμμα διαβάζει τα queries που ο χρήστης του παρέχει και τα εκτελεί επαναληπτικά με τη σειρά που αυτά έχουν δηλωθεί. Ο αριθμός των επαναλήψεων μπορεί να καθοριστεί απ το χρήστη. Για της ανάγκες της πτυχιακής το πρόγραμμα φτιάχτηκε ώστε να πραγματοποιεί benchmark στα συγκεκριμένα 4 δοκιμαζόμενα rdbms. Τα χαρακτηριστικά των rdbms (username, password, drivers κτλ) είναι hardcoded. Με πολύ μικρές αλλαγές το πρόγραμμα μπορεί να τροποποιηθεί ώστε να γίνει database agnostic και ο χρήστης να μπορεί να το χρησιμοποιήσει και σε άλλα rdbms της επιλογής του. Τα αποτελέσματα αποθηκεύονται σε αρχείο κειμένου το οποίο επιλέγει ο χρήστης. Σε μία γραμμή τοποθετούνται οι χρόνοι από την κάθε επανάληψη ενός query.

Η παραμετροποίηση της εκτέλεσης του προγράμματος γίνεται με τη χρήση αρχείων παραμετροποίησης (configuration files) λόγω του μεγάλου πλήθους πληροφοριών που χρειάζεται το πρόγραμμα προκειμένου να εκτελεστεί. Η χρήση παραμέτρων γραμμής εντολών θα έκανε δυσκολότερη τη χρήση του. Η εκτέλεση του προγράμματος χωρίς παραμέτρους γραμμής εντολών ξεκινά το benchmark για queries. Για την εκτέλεση των insert/update πρέπει να χρησιμοποιηθεί ο διακόπτης –i. Το αρχείο παραμετροποίησης πρέπει να έχει σε κάθε γραμμή ένα ζεύγος παραμέτρου – τιμής, χωρισμένα με ':'. Τα properties που πρέπει να έχει το αρχείο είναι τα εξής:

host	προσδιορίζει τον υπολογιστή που τρέχει το db server
rdbms	μια από τις παρακάτω τιμές: MySql, PostgreSQL, IBM DB2 και sqlServer
results	προσδιορίζει το αρχείο στο οποίο θα αποθηκευθούν τα αποτελέσματα. Αν δεν υπάρχει δημιουργείται ενώ αν υπάρχει αντικαθίσταται άνευ προειδοποίησης.
queries	προσδιορίζει το αρχείο από το οποίο θα διαβάζονται τα queries προς εκτέλεση. Το κάθε query πρέπει να έχει μια κενή γραμμή από το επόμενο ενώ τα inserts και τα updates τοποθετούνται ένα σε κάθε γραμμή.
repetitions	προαιρετικό. Προσδιορίζει τον αριθμό των επαναλήψεων που θα εκτελεστεί ένα query. Αν η τιμή δεν είναι αριθμητική ή αν η παράμετρος δεν προσδιορίζεται τότε λαμβάνεται η προεπιλεγμένη τιμή (5). Η τιμή αγνοείται για την περίπτωση των updates/inserts τα οποία εκτελούνται μία φορά.

Πίνακας 1: Μορφή αρχείου παραμετροποίησης

Παράδειγμα:

```
host:192.168.1.67
rdbms:postgres
results:Postgres linux results.txt
queries:./queries/Postgres.txt
repetitions:10
```

Οι δοκιμές έγιναν με τη χρήση 2 υπολογιστών συνδεδεμένων σε Ethernet LAN. Το firewall για λόγους απλούστευσης αλλά και επιτάχυνσης των διαδικασιών απενεργοποιήθηκε. Ο παλαιότερος υπολογιστής ανέλαβε το ρόλο του εξυπηρετητή των βάσεων δεδομένων ενώ ο νεότερης τεχνολογίας ανέλαβε την εκτέλεση του προγράμματος. Η επιλογή αυτή έγινε ώστε ο παλαιότερης τεχνολογίας υπολογιστής να προκαλέσει αποτελέσματα μεγαλύτερης διάρκειας και να γίνει η σύγκριση ευκολότερη. Φυσικά αυτό στην πράξη αποδείχτηκε μάταιο λόγω του πολύ μεγάλου χρόνου εκτέλεσης των ερωτημάτων για ορισμένα rdbms. Για τον ίδιο λόγο κρίθηκε αναγκαίο να εμφανίζεται η πρόοδος του benchmark στη γραμμή εντολών. Πιο συγκεκριμένα στην αρχή κάθε επανάληψης εμφανίζεται η ώρα έναρξης ο αύξων αριθμός του ερωτήματος και η επανάληψη, και στο τέλος της επανάληψης, εμφανίζεται η ώρα τερματισμού και η χρονική διάρκεια σε millisecond.

3.3 Εξαγωγή και αναπαράσταση των αποτελεσμάτων

Εξαγωγή αποτελεσμάτων

Τα αποτελέσματα των πειραμάτων καταγράφονται σε αρχείο κειμένου. Στην αρχική εκδοχή του προγράμματος τα αποτελέσματα καταγράφονταν μετά το τέλος των πειραμάτων. Η αυτή απόφαση ελήφθη με τη λογική ότι τα rdbms θα χρειαστούν ένα λογικό χρόνο για να εκτελέσουν τα ερωτήματα και δε θα χρειάζεται ο υπολογιστής να είναι ανοιχτός για μέρες ώστε να κινδυνεύει από διακοπές ρεύματος και απώλεια των αποτελεσμάτων του πειράματος. Σύντομα η ελπίδα των λογικών χρόνων εκτέλεσης αντικαταστάθηκε από την πραγματικότητα των δυόμισι ημερών που χρειάστηκε η mysql να εκτελέσει μία επανάληψη του 18ου query. Μετά και τη μεσολάβηση διακοπής ρεύματος, κρίθηκε αναγκαία η αποθήκευση των αποτελεσμάτων μετά την εκτέλεση των επαναλήψεων ενός query και όχι στο τέλος του πειράματος. Κατ αυτόν τον τρόπο αν χαθούν δεδομένα θα χαθούν μόνο από το τελευταίο query και όχι απ όλα τα προηγούμενα.

Το κάθε αρχείο περιλαμβάνει στις γραμμές τα αποτελέσματα για το κάθε ερώτημα και στις στήλες το χρόνο εκτέλεσης της κάθε επανάληψης. Οι στήλες είναι χωρισμένες με έναν κενό χαρακτήρα. Για την περίπτωση των εντολών insert ή update, η κάθε γραμμή αντιπροσωπεύει τον πίνακα πάνω στον οποίον αυτές εκτελέστηκαν. Στην αρχή της κάθε γραμμής ο χρήστης μπορεί να εισάγει το όνομα του ερωτήματος/πίνακα. Το όνομα πρέπει να είναι διαχωρισμένο με ':' με την υπόλοιπη γραμμή. Το όνομα χρησιμοποιείται από τον εξαγωγέα διαγραμμάτων. Στην κορυφή του αρχείου περιλαμβάνονται ετικέτες (tags) οι οποίες προσδιορίζουν τη ταυτότητα του αρχείου αποτελεσμάτων. Οι ετικέτες τις οποίες αναγνωρίζει το πρόγραμμα είναι οι εξής:

rdbms:
os:
dbsize:
type:
indexed:

Αυτές οι ετικέτες όπως είναι φυσικό δε μπορούν να δημιουργηθούν από το πρόγραμμα, καθώς στην πλειονότητα των περιπτώσεων το πρόγραμμα δε δύναται να γνωρίζει τις τιμές αυτές με εξαίρεση την περίπτωση του rdbms και του os. Για το λόγω αυτό καλείται ο χρήστης να εισάγει τις τιμές χειροκίνητα. Η ύπαρξη των ετικετών είναι απαραίτητη για την επιτυχή δημιουργία των συγκριτικών διαγραμμάτων για τ' αποτελέσματα.

Παράδειγμα:

```
rdbms:db2
os:linux
dbsize:large
type:queries
indexed:no
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
....
0 0 0 0 0 0 0 0 0 0
```

Δημιουργία διαγραμμάτων

Για τη δημιουργία των διαγραμμάτων χρησιμοποιήθηκε η βιβλιοθήκη JFreeChart η οποία είναι ελεύθερου λογισμικού, διανέμεται υπό την άδεια LGPL και είναι διαθέσιμο στην ιστοσελίδα http://www.jfree.org/jfreechart/. Διαθέτει πλούσιο ρεπερτόριο διαγραμμάτων καθώς και demo εφαρμογή η οποία διαθέτει παραδείγματα για κάθε ένα από τα είδη διαγραμμάτων που υποστηρίζει η βιβλιοθήκη. Ο πηγαίος κώδικας για τα διαγράμματα διατίθεται μόνο για τις πολύ απλές περιπτώσεις ενώ για τις υπόλοιπες διατίθεται μαζί με τον οδηγό χρήσης για προγραμματιστές που έχει γράψει ο δημιουργός της βιβλιοθήκης. Το κόστος του οδηγού ξεκινά από €56 και φτάνει μέχρι τα €1268.

Η βιβλιοθήκη δίνει τη δυνατότητα παραμετροποίησης των διαγραμμάτων αν και όπως είναι αναμενόμενο χρειάζεται μια σχετικά καλή γνώση του API της βιβλιοθήκης. Ένα πολύ σημαντικό χαρακτηριστικό της είναι ότι στο panel που απεικονίζεται ένα διάγραμμα, μπορούμε με drag του ποντικιού να κάνουμε zoom σε συγκεκριμένο εύρος τιμών. Αυτό αποδείχθηκε εξαιρετικά χρήσιμο για την περίπτωσή μας λόγω των πάρα πολύ μεγάλων αποκλίσεων των χρόνων που εμφάνισε το κάθε rdbms.

Ο μηχανισμός εξαγωγής των διαγραμμάτων εξετάζεται αναλυτικά στο κεφάλαιο 5.2 και οι οδηγίες χρήσης της εφαρμογής στον οδηγό χρήσης λογισμικού

ΕΠΙΛΟΓΟΣ

Με αυτό το κεφάλαιο ολοκληρώθηκε η διαδικασία εκτέλεσης του benchmark. Συγκεκριμένα σε αυτό το σημείο έγινε η εκτέλεση των ερωτημάτων, η εξαγωγή των αποτελεσμάτων και η απεικόνιση τους σε διαγράμματα. Στο επόμενο κεφάλαιο εξετάζεται το πως μπορούν να βελτιωθούν ορισμένα ερωτήματα που χρειάστηκαν μεγάλο χρονικό διάστημα να εκτελεστούν.

ΚΕΦΑΛΑΙΟ 4: Query optimization σε MySql και Postgres

ΕΙΣΑΓΩΓΗ

Αναμφίβολα τα αποτελέσματα που εξήχθησαν από το ανωτέρω πείραμα προκαλούν 2 σημαντικά ερωτηματικά. Το πρώτο είναι οι εξαιρετικά μεγάλοι χρόνοι εκτέλεσης των ερωτημάτων στα Open Source rdbms και ειδικά στη MySql σε σημείο που να αναρωτιέται κανείς το πώς έγινε επιτυχημένο αυτό το rdbms. Το δεύτερο είναι οι εξαιρετικά μικροί χρόνοι εκτέλεσης στον SqlServer 2008 σε σημείο που να υποψιάζεται κανείς ότι η Microsoft “έστησε” το rdbms της ώστε να έχει την καλύτερη δυνατή απόδοση στα 22 αυτά queries.

Οι επιδώσεις του SqlServer αν και είναι άξιες απορίας, δεν αποτελούν αντικείμενο μελέτης. Είναι άλλωστε στην φύση του ανθρώπου να μην αναρωτιέται όταν κάτι πάει καλύτερα από το αναμενόμενο. Η περίπτωση όμως των Open Source rdbms χρήζει σίγουρα περαιτέρω ανάλυσης, ειδικά αν λάβουμε υπ' όψη μας το γεγονός ότι και τα 2 συστήματα έχουν από 1 ερώτημα που δεν έδωσε αποτελέσματα. Στο κεφάλαιο αυτό θα εξετάσουμε τις βελτιώσεις που δύνανται να υποστούν τα ερωτήματα που χρειάστηκαν πολύ χρόνο για να εκτελεστούν ή που δεν έβγαλαν καθόλου αποτελέσματα.

4.1 Query optimization στη MySql

Στη MySql το ερώτημα 18 δεν έβγαλε καθόλου αποτελέσματα μετά από δύο ημέρες εκτέλεσης. Μοναδική εξαίρεση η μικρή βάση δεδομένων στα Windows που έβγαλε αποτελέσματα μετά από 30254456 millisecond (8,4 ώρες). Αν και τα rdbms παρέχουν έναν βαθμό παραμετροποίησης επιδόσεων, η προσοχή στράφηκε πρώτα στα ερωτήματα κυρίως λόγω του ότι το benchmark συγκρίνει out of the box συστήματα. Κατόπιν ενδελεχούς αναζήτησης στο διαδίκτυο, εντοπίστηκαν άρθρα τα οποία ασχολήθηκαν με τη βελτιστοποίηση των ερωτημάτων της TPC-H. Συγκεκριμένα τα άρθρα με τίτλο “Improving TPC-H-like Queries – Q2” και “Improving TPC-H-like queries – Q17” του www.tokutek.com ήταν διαφωτιστικά.

Ο συντάκτης του άρθρου αναφέρει ότι το ερώτημα 2 (Q2) σε βάση δεδομένων μεγέθους 10GB χρειάστηκε 1830 δευτερόλεπτα για να εκτελεστεί. Παρακάτω στο ίδιο άρθρο εξηγεί το πώς κατάφερε να μειώσει το χρόνο εκτέλεσης περίπου 160 φορές, με μέσο χρόνο εκτέλεσης τα 11 δευτερόλεπτα.

Το πρώτο βήμα είναι η “χαρτογράφηση” του query με την εντολή EXPLAIN. Η εντολή αυτή όταν τοποθετείται μπροστά από ένα ερώτημα, αντί για την εκτέλεση

του ερωτήματος, εμφανίζει έναν πίνακα με το query plan του βελτιστοποιητή ερωτημάτων (query optimizer). Εκεί παρατηρούμε ότι η εκτέλεση του query ξεκινά από έναν μικρό πίνακα (region) ο οποίος γίνεται μετά join σε μεγαλύτερους πίνακες. Καλύτερο θα ήταν να ξεκινούσε πρώτα από τον πίνακα part ώστε να αποκλειστούν οι γραμμές που δεν ικανοποιούν τη συνθήκη (p_size = 35, p_type like '%STEEL') και μετά να γίνει join στους μικρότερους πίνακες. Αυτό επετεύχθη με τη χρήση της εντολής STRAIGHT_JOIN στο κυρίως και στο εμφωλευμένο query. Το STRAIGHT_JOIN δίνει στο συντάκτη του query τον έλεγχο του πλάνου ερωτήματος, με αποτέλεσμα ο optimizer να διαβάζει τους πίνακες με τη σειρά που αναγράφονται στο FROM. Επιπλέον χρειάστηκε για το κυρίως query η αλλαγή της σειράς των supplier και partsupp. Η σειρά των πινάκων στο FROM από part, **supplier, partsupp**, nation, region έγινε part, **partsupp, supplier**, nation, region.

Η εκτέλεση του τροποποιημένου ερωτήματος στον υπολογιστή δοκιμών πραγματοποιήθηκε σε 11,38 δευτερόλεπτα έναντι των περίπου 17 λεπτών του πρωτότυπου ερωτήματος. Για βάση 300 MB, η βελτίωση του χρόνου εκτέλεσης είναι 90 φορές γρηγορότερα.

Για το ερώτημα 17 χρησιμοποιήθηκε η ίδια προσέγγιση με το STRAIGHT_JOIN με τη λεπτομέρεια ότι προστέθηκε clustered index στα πεδία l_partkey και l_suppkey του πίνακα lineitem. Τη δυνατότητα αυτή δεν τη διαθέτει η μηχανή innodb. Οι συντάκτες του άρθρου χρησιμοποίησαν τη δική τους μηχανή TokuDB που έχει αυτή τη δυνατότητα.

Η ανακάλυψη του άρθρου για το ερώτημα 17, προέκυψε από την αναζήτηση ενός τρόπου βελτιστοποίησης του ερωτήματος 18 το οποίο είναι το μοναδικό ερώτημα που δεν τερμάτισε ποτέ, ακόμα και μετά από 2 μέρες εκτέλεσης. Η πρόκληση είναι μεγάλη. Το να κάνεις ένα ερώτημα να τελειώσει σε λίγα δευτερόλεπτα, ενώ φαινόταν να μη θέλει να τελειώσει ποτέ, είναι δύσκολο έργο.

Για το ερώτημα 18 ακολουθήθηκε η ίδια προσέγγιση. Το query εκτελέστηκε με το πρόθεμα EXPLAIN EXTENDED. Η έξοδος της εντολής ήταν η ακόλουθη:

id	select_type	table	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	customer	ALL	PRIMARY	NULL	NULL	NULL	151597	100.00	Using temporary; Using filesort
1	PRIMARY	orders	ref	PRIMARY,O_CUSTKEY	O_CUSTKEY	4	tpc_h.customer.C_CUSTKEY	1	100.00	Using where
1	PRIMARY	lineitem	ref	PRIMARY	PRIMARY	4	tpc_h.orders.O_ORDERKEY	1	100.00	
2	DEPENDENT SUBQUERY	lineitem	index	NULL	PRIMARY	8	NULL	1	74788000.00	

Πίνακας 2: MySql EXPLAIN EXTENDED

Παρατηρούμε ότι ο πίνακας ο οποίος διαβάζεται πρώτος είναι ο customer. Η πρώτη σκέψη ήταν η αλλαγή της σειράς εμφάνισης των πινάκων στο FROM. Τοποθετήθηκε ο πίνακας orders πρώτος έτσι ώστε να αποκλειστούν οι γραμμές που δεν υπάρχουν στα αποτελέσματα του εμφωλευμένου ερωτήματος που είναι μέσα στο IN. Η εκτέλεση του τροποποιημένου ερωτήματος δεν τερμάτισε σε εύλογο χρονικό διάστημα.

Το αμέσως επόμενο εύρημα που προκαλεί εντύπωση είναι η τιμή 74788000.00 στη στήλη filtered του πίνακα αποτελεσμάτων. Βάσει της on-line τεκμηρίωσης, η συγκεκριμένη τιμή δείχνει ποσοστό επί τοις εκατό του συνόλου των γραμμών του πίνακα που θα φιλτραριστούν κατά την εκτέλεση του ερωτήματος. Οποιαδήποτε τιμή μεγαλύτερη του 100% είναι ύποπτη πόσο μάλλον μια τιμή που ξεπερνά τα 74 εκατομμύρια. Οι υποψίες στρέφονται στο εμφωλευμένο ερώτημα το οποίο και εκτελέστηκε ξεχωριστά ώστε να γίνει επιμέρους μέτρηση της επίδοσής του. Κατά την πρώτη εκτέλεση του ερωτήματος, ο χρόνος εκτέλεσής του ήταν 1,2 δευτερόλεπτα. Πολλαπλασιαζόμενο με το 747.880 (ο συνολικός αριθμός φορών που θα εκτελεστεί το εμφωλευμένο) μας δίνει σύνολο 897.456 δευτερόλεπτα δηλαδή κάτι παραπάνω από δέκα μέρες. Οι δύο επόμενες εκτελέσεις (με την cache απενεργοποιημένη) έδωσαν 0,38 δευτερόλεπτα. Άρα στην καλύτερη περίπτωση, το ερώτημα θα χρειαστεί περίπου 3,5 μέρες να εκτελέσει μόνο το εμφωλευμένο query.

Η υπόθεση ότι το εμφωλευμένο query εκτελείται κάθε φορά (είναι φανερό στο ερώτημα ότι η τιμή του δεν αλλάζει κατά τη διάρκεια της εκτέλεσης του κυρίως ερωτήματος) είναι υπόθεση και όχι βεβαιότητα. Μένει να αποδειχθεί στην πράξη η ορθότητα της υπόθεσης. Το εμφωλευμένο query αντικαταστάθηκε από τις τιμές που αυτό επιστρέφει και το κυρίως query ξαναεκτελέστηκε. Το αποτέλεσμα ήταν θεαματικό: μόλις 3,47 δευτερόλεπτα. Η εκτέλεση του ίδιου ερωτήματος με το

πρόθεμα EXPLAIN EXTENDED έρχεται για να επισφραγίσει την ορθότητα της υπόθεσης δίνοντας τιμή 100.00 αντί 74788000.00.

Η λύση στο πρόβλημα αυτό είναι η απομάκρυνση του εμφωλευμένου query από το WHERE ώστε να εκτελεστεί μία φορά. Το query μετακινήθηκε στο FROM ως ενδιάμεσος πίνακας. Η νέα σύνταξη του ερωτήματος έχει ως εξής:

```
select
     c_name,
     c_custkey,
     o_orderkey,
     o_orderdate,
     o_totalprice,
     sum(l_quantity)
from
     (select
          l_orderkey as t_orderkey
     from
          lineitem
     group by
          t_orderkey having
               sum(l_quantity) > :1) as derived,
     customer,
     orders,
     lineitem
where
     o_orderkey in (derived.t_orderkey)
     and c_custkey = o_custkey
     and o_orderkey = l_orderkey
group by
     c_name,
     c_custkey,
     o_orderkey,
     o_orderdate,
     o_totalprice
order by
     o_totalprice desc,
     o_orderdate;
```

Η εκτέλεση του ερωτήματος έδωσε τα ίδια αποτελέσματα σχεδόν στον ίδιο χρόνο με την hard-coded εκδοχή του ερωτήματος (4.64 δευτερόλεπτα). Η ορθότητα των αποτελεσμάτων διασταυρώθηκε με τα αποτελέσματα που έδωσε ο SqlServer για το αρχικό query.

Μία σκέψη του γιατί συμπεριφέρεται κατ' αυτόν τον τρόπο είναι το γεγονός ότι η εκτέλεση των ερωτημάτων έγινε χωρίς την χρήση cache (query_cache_size=0 στο

αρχείο my.ini). Η MySql θα μπορούσε να βάζει στη cache οποιοδήποτε ερώτημα ακόμα και εμφωλευμένο. Απενεργοποιώντας τη cache είναι σα να υποχρεώνουμε τη MySql να εκτελεί το εμφωλευμένο ερώτημα συνέχεια. Η θεωρία αυτή έχει λογική εξήγηση, έμενε να αποδειχτεί η ορθότητά της στην πράξη, κάτι που τελικά δεν έγινε. Η MySql έβαλε στη cache το εμφωλευμένο ερώτημα όταν εκτελέστηκε αυτόνομα αλλά όχι όταν εκτελέστηκε μέσα στο υπόλοιπο query.

4.2 Query optimization στην Postgres

Στην Postgres το ερώτημα που δεν επέστρεψε αποτέλεσμα μετά από 1 μέρα εκτέλεσης στη μεγάλη βάση ήταν το 20ο. Επιπλέον το ερώτημα 17 για το ίδιο μέγεθος χρειάστηκε σχεδόν 3 ώρες (170 λεπτά). Κατόπιν αναζήτησης στο διαδίκτυο κάποιος χρήστης στο προσωπικό του ιστολόγιο παρέθεσε ένα εναλλακτικό ερώτημα για το ερώτημα 17 με σκοπό την δραματική μείωση του χρόνου εκτέλεσης. Σύμφωνα με το συντάκτη του άρθρου, δύο διαδοχικά join πάνω σε δύο πίνακες αποδίδουν πολύ καλύτερα από ότι ένα join πάνω σε τρεις πίνακες.

Παρόλο που το ερώτημα δεν ήταν ολοκληρωμένο και χρειάστηκε επεξεργασία ώστε να έρθει σε εκτελέσιμη μορφή, το αποτέλεσμα ήταν θεαματικό: 38 δευτερόλεπτα από 170 λεπτά, βελτίωση περίπου 270 φορές. Το νέο query έχει ως εξής:

```
select
    sum(l_extendedprice) / 7.0 as avg_yearly
from
    lineitem,
    part,
    (select
        l_partkey,
        0.2 * avg(l_quantity) as part_avg
    from
        lineitem
    group by l_partkey) as avg_quantity
where
    p_partkey = lineitem.l_partkey
    and p_brand = 'Brand#22'
    and p_container = 'LG CAN'
    and avg_quantity.l_partkey = p_partkey
    and l_quantity < part_avg;
```

Βλέποντας το αρχικό query παρατηρούμε ότι το εσωτερικό ερώτημα δεν είναι κοινό εμφωλευμένο αλλά συσχετισμένο (ή συγχρονισμένο) εσωτερικό ερώτημα (correlated or synchronized inner query). Στο συμπέρασμα αυτό οδηγούμαστε από την αναφορά στον πίνακα part του εξωτερικού query από το εσωτερικό. Αυτό σημαίνει ότι το εσωτερικό ερώτημα εκτελείται για κάθε πλειάδα (ή ομάδα πλειάδων) του εξωτερικού ερωτήματος.

Ο συντάκτης του άρθρου αναφέρει κοντά στο τέλος ότι χρησιμοποίησε την ίδια τεχνική για να βελτιστοποιήσει και το ερώτημα 20. Η πρόκληση ήταν μεγάλη αν λάβουμε υπ όψη μας ότι το Q20 δεν επέστρεψε αποτελέσματα μετά από μία μέρα

εκτέλεσης. Αναλύοντας το ερώτημα παρατηρούμε ότι το τελευταίο εσωτερικό ερώτημα είναι correlated, και άρα το πιθανότερο να προκαλεί την καθυστέρηση.

Σύμφωνα με την προηγούμενη διαδικασία τα βήματα που πρέπει να ακολουθηθούν για τη μετατροπή του συσχετισμένου εσωτερικού ερωτήματος έχουν ως εξής:

1. Μετακινούμε το εσωτερικό ερώτημα από το where στο αντίστοιχο from (οι παρενθέσεις μένουν κενές), δίνοντας ένα όνομα στον πίνακα που προκύπτει (έστω sum_table).

```
select
    ps_suppkey
from
    (select
        0.5 * sum(l_quantity)
     from
        lineitem
     where
        l_partkey = ps_partkey
        and l_suppkey = ps_suppkey
        and l_shipdate >= date ':2'
        and l_shipdate < date ':2' + interval '1' year) AS
sum_table,
    partsupp
where
    ps_partkey in (<ΠΑΡΑΛΕΙΠΕΤΑΙ>)
        and ps_availqty > ()
```

2. Απαλείφουμε όλες τις αναφορές στον εξωτερικό πίνακα (ο partsupp στην περίπτωση του Q20). Οι συνθήκες που φιλτράρουν τον πίνακα lineitem μένουν ανέπαφες.

```
select
    ps_suppkey
from
    (select
        0.5 * sum(l_quantity)
     from
        lineitem
     where
        l_partkey = ps_partkey
        and l_suppkey = ps_suppkey
        and l_shipdate >= date ':2'
        and l_shipdate < date ':2' + interval '1' year) AS
sum_table,
    partsupp
where
```

```
    ps_partkey in (<ΠΑΡΑΛΕΙΠΕΤΑΙ>)
        and ps_availqty > ()
```

3. Το κλειδί που χρησιμοποιήθηκε για το join με τον εξωτερικό πίνακα μετακινείται στη λίστα select του νέου ερωτήματος. Επίσης τοποθετείται σε group by. Το group by είναι απαραίτητο όχι μόνο επειδή επιβάλλεται από το συντακτικό της γλώσσας αλλά για να υπολογιστεί το sum για κάθε ίδιο ζεύγος t_partkey και t_suppkey όπως γίνεται και στο αρχικό ερώτημα.

```
select
    ps_suppkey
from
    (select
        l_partkey as t_partkey,
        l_suppkey as t_suppkey,
        0.5 * sum(l_quantity) AS tm_availqty
     from
         lineitem
     where
         l_shipdate >= date ':2'
         and l_shipdate < date ':2' + interval '1' year
     group by t_partkey, t_suppkey) AS sum_table,
    partsupp
where
    ps_partkey in (<ΠΑΡΑΛΕΙΠΕΤΑΙ>)
    and ps_availqty > ()
```

4. Τα joins που αφαιρέθηκαν στο βήμα 2 πρέπει να τοποθετηθούν στο where του ερωτήματος.

```
select
    ps_suppkey
from
    (select
        l_partkey as t_partkey,
        l_suppkey as t_suppkey,
        0.5 * sum(l_quantity) AS tm_availqty
     from
         lineitem
     where
         l_shipdate >= date ':2'
         and l_shipdate < date ':2' + interval '1' year
     group by t_partkey, t_suppkey) AS sum_table,
    partsupp
where
    ps_partkey in (<ΠΑΡΑΛΕΙΠΕΤΑΙ>)
    and t_partkey = ps_partkey
    and t_suppkey = ps_suppkey
```

```
        and ps_availqty > ()
```

5. Στη θέση που ήταν παλιά το εσωτερικό ερώτημα, σβήνουμε τις κενές παρενθέσεις του ερωτήματος 1 και αντικαθιστούμε με την τιμή του aggregation function του πίνακα sum_table.

```
select
    ps_suppkey
from
    (select
        l_partkey as t_partkey,
        l_suppkey as t_suppkey,
        0.5 * sum(l_quantity) as tm_availqty
    from
        lineitem
    where
        and l_shipdate >= date '1994-01-01'
        and l_shipdate < date '1994-01-01' + interval '1' year
    group by t_partkey, t_suppkey) AS sum_table,
    partsupp
where
    ps_partkey in (<ΠΑΡΑΛΕΙΠΕΤΑΙ>)
        and t_partkey = ps_partkey
        and t_suppkey = ps_suppkey
        and ps_availqty > (sum_table.tm_availqty)
```

Συνοπτικά, αυτό που κάναμε, ήταν να πούμε στον query planer ότι αντί να τρέξει το εμφωλευμένο ερώτημα μία φορά για κάθε γραμμή το εξωτερικού ερωτήματος, θα το τρέξει μία φορά, θα αποθηκεύσει τις τιμές και θα τις χρησιμοποιήσει εκ των υστέρων για να κάνει ελέγχους. Ουσιαστικά θα κάνει caching αντί για on-demand εκτέλεση. Και σ' αυτή την περίπτωση η βελτίωση δεν προκαλεί εντύπωση: 41 δευτερόλεπτα για ένα ερώτημα που δεν ολοκληρώθηκε μετά από μία μέρα εκτέλεσης. Το συνολικό τροποποιημένο ερώτημα που προέκυψε από την ανωτέρω διαδικασία έχει ως εξής:

```
select
    s_name,
    s_address
from
    supplier,
    nation
where
    s_suppkey in (
        select
            ps_suppkey
        from
            (select
                l_partkey as t_partkey,
```

```
                l_suppkey as t_suppkey,
                0.5 * sum(l_quantity) as tm_availqty
            from
                lineitem
            where
                and l_shipdate >= date '1994-01-01'
                and l_shipdate < date '1994-01-01' + interval '1' year
            group by t_partkey, t_suppkey) AS sum_table,
            partsupp
        where
            ps_partkey in (
                select
                    p_partkey
                from
                    part
                where
                    p_name like 'forest%')
                and t_partkey = ps_partkey
                and t_suppkey = ps_suppkey
                and ps_availqty > (sum_table.tm_availqty)
    )
    and s_nationkey = n_nationkey
    and n_name = 'CANADA'
order by
    s_name;
```

ΕΠΙΛΟΓΟΣ

Στο 4ο κεφάλαιο πήραμε τα 2 πιο “προβληματικά” ερωτήματα για την Postgres και τη MySql και κάναμε μια (επιτυχή) προσπάθεια να βελτιώσουμε τους χρόνους εκτέλεσης, με τη βοήθεια και αρκετών διαδικτυακών πηγών. Με το 4ο κεφάλαιο κλείνει ο κύκλος της ενασχόλησης με τα ερωτήματα και την SQL. Στο επόμενο κεφάλαιο θα δούμε το επιμέρους λογισμικό που αναπτύχθηκε ως εργαλείο για την πτυχιακή.

ΚΕΦΑΛΑΙΟ 5: Εργαλεία κώδικα σε Java

ΕΙΣΑΓΩΓΗ

Όπως έχει ήδη αναφερθεί σε προηγούμενα κεφάλαια, και λόγω των απρόοπτων που προέκυψαν, τα δεδομένα που εξάγει η TPC-H κρίθηκαν ακατάλληλα προς άμεση χρήση και απαιτήθηκε η τροποποίηση τους. Η επεξεργασία ενός αρχείου 700MB είναι εφικτή από λίγους επεξεργαστές κειμένου και όταν η επεξεργασία γίνεται πολύπλοκη και οι επεξεργαστές κειμένου κρίνονται ανεπαρκείς. Παρακάτω ακολουθεί μια συνοπτική περιγραφή των εργαλείων που χρησιμοποιήθηκαν για οποιοδήποτε άλλο σκοπό πλην της καθαυτής εκτέλεσης του benchmark.

5.1 Εργαλεία τροποποίησης δεδομένων

FindKey: το FindKey.java χρησιμοποιήθηκε για την αρχική μείωση από τα δεδομένα του ενός GB στα 300MB. Στόχος του είναι να μας πληροφορήσει για το ποια είναι η καταλληλότερη τιμή του l_orderkey για τη διαγραφή των δεδομένων από τους πίνακες lineitem και orders, χωρίς να χαθεί η αναφορική ακεραιότητα. Αφού αποφασίσουμε πόσες γραμμές θέλουμε να μείνουν στο lineitem, αντικαθιστούμε στον κώδικα την τιμή της μεταβλητής linesleft και εκτελούμε. Το πρόγραμμα εκτυπώνει την γραμμή linesleft του αρχείου καθώς και 5 γραμμές επάνω και 5 γραμμές κάτω ώστε να είναι εμφανής η χρήση του l_orderkey το οποίο επαναλαμβάνεται στη lineitem. Το lineitem.tbl που εξάγεται από το dbgen είναι ταξινομημένο βάσει του l_orderkey για το λόγο αυτό η ανωτέρω τεχνική έχει αποτέλεσμα.

Το ίδιο αποτέλεσμα μπορεί να επιτευχθεί και με τη χρήση sql στην περίπτωση που η βάση έχει ήδη εγκατασταθεί με την παρακάτω εντολή:

```
select *
from
      (select l_orderkey
      from lineitem
      order by l_orderkey
      limit 500000 + 5) as temp
order by
      l_orderkey desc
limit 11
```

TruncateDB: για την περαιτέρω μείωση της βάσης δεδομένων από τα 300MB στα 135 και έπειτα στα 65 η FindKey δεν επαρκούσε. Ο πίνακας lineitem είναι ο μεγαλύτερος στη βάση αλλά αυτό δε συνεπάγεται ότι μπορεί να μειώνεται επ' άπειρο. Αυτό θα προκαλούσε δυσανάλογη μείωση του μεγέθους του πίνακα ως προς τους υπόλοιπους. Η εφαρμογή αυτή χρησιμοποιήθηκε για την μείωση των

πινάκων lineitem και partsupp για τη δημιουργία της μεσαίας βάσης δεδομένων και lineitem partsupp και orders για τη δημιουργία της μικρής.

Ως κλειδί αναφοράς για τη μείωση των πινάκων χρησιμοποιούμε αυτό του πίνακα part ο οποίος είναι ο πίνακας από τον οποίο εξαρτώνται οι υπόλοιποι (έμμεσα ή άμεσα). Πειραματιζόμενοι με το πόσες εγγραφές απομένουν σε κάθε πίνακα, επιλέγουμε μια αυθαίρετη τιμή του partKey και τη θέτουμε στο πεδίο partKeyLimit. Βάσει αυτής της τιμής η εφαρμογή αφαιρεί τις εγγραφές από τα αρχεία κειμένου του αρχικού μεγέθους. Τα περιεχόμενα των αρχείων προς μείωση, αποθηκεύονται ανά γραμμή σε λίστες συμβολοσειρών. Για κάθε εγγραφή στη λίστα ανακτάται η τιμή του partKey και αν είναι μικρότερη από αυτή του partKeyLimit, η γραμμή τοποθετείται σε νέα λίστα. Από τη νέα λίστα προκύπτει το αρχείο με τα μειωμένα δεδομένα. Για τη μείωση του πίνακα orders στον οποίο δεν υπάρχει αναφορά στο partKey, απαιτήθηκε η αποθήκευση των κλειδιών (l_orderkey) που δε θα διαγραφούν από τη lineitem και η διατήρηση των εγγραφών στην order που ταιριάζουν με τα κλειδιά αυτά.

CreateInserts: τα δεδομένα που εξάγει το dbgen για τα inserts είναι σε γραμμογραφημένη delimited μορφή με delimiter το pipe (|). Χρειάζονται επεξεργασία ώστε να μετατραπούν σε sql INSERT INTO. Το CreateInserts αναλαμβάνει την μετατροπή αυτή. Στις παραμέτρους γραμμής εντολών περνάμε πρώτο το όνομα του πίνακα και μετά τρία ονόματα αρχείων: πρώτο το αρχείο εισόδου, δεύτερο το αρχείο εξόδου που θα περιέχει τα έτοιμα INSERT INTO και τελευταίο το αρχείο που περιγράφει το schema του πίνακα. Το αρχείο αυτό πρέπει να περιέχει τα ονόματα των πεδίων του πίνακα σε μία γραμμή, χωρισμένα με κόμμα και με τη σειρά που αναγράφονται στο αρχείο εισόδου.

CreateUpdates: η TPC δε δίνει την επιλογή εκτέλεσης updates και για το λόγο αυτό έπρεπε να δημιουργηθούν από την αρχή. Η δημιουργία των updates βασίστηκε στην τυχαιοποίηση. Η εφαρμογή για τους πίνακες order, lineitem, part και supplier δημιουργεί τυχαίες τιμές για συγκεκριμένα πεδία. Τα κλειδιά για τους πίνακες με συνεχόμενες τιμές (part και supplier) βγαίνουν από επιλογή ενός τυχαίου ακεραίου από 1 έως τη μέγιστη τιμή του κλειδιού. Για τους πίνακες orders και lineitem στους οποίους δεν υπάρχει συνεχόμενη τιμή του κλειδιού, απαιτήθηκε η συγκέντρωση σε λίστα όλων των έγκυρων κλειδιών από το αρχείο orders.tbl. Τα πεδία για τα οποία δημιουργήθηκαν random τιμές είναι τύπου integer, float, string, char και date. Ειδική περίπτωση string είναι τα πεδία τηλεφώνου που είναι της μορφής “XXX-XXX-XXXX” και πεδία με συγκεκριμένες τιμές όπως τα l_shipmode, l_shipınstruct και o_orderpriority. Για την περίπτωση αυτή χρησιμοποιήθηκε η εντολή select distinct για τον εντοπισμό των μοναδικών τιμών του κάθε πεδίου και,

αφού καταγράφηκαν στο πρόγραμμα, επιλέχθηκαν με τη χρήση τυχαίου ακέραιου αριθμού.

5.2 Εργαλείο εξαγωγής διαγραμμάτων

Το εργαλείο αυτό αναπτύχθηκε με σκοπό την ad-hoc δημιουργία διαγραμμάτων από τα αποτελέσματα του benchmark. Η χρήση του Microsoft Excel θα είχε περισσότερες δυσκολίες στην υλοποίηση και δε θα έδινε την ίδια ευελιξία επιλογών αναφορικά με το περιεχόμενο των διαγραμμάτων.

Αρχικά να αναφέρουμε ότι η εν λόγω εφαρμογή δεν έχει σκοπό να υποκαταστήσει επαγγελματικά προγράμματα ούτε είναι επαγγελματικού επιπέδου. Γράφτηκε με σκοπό να βοηθηθεί ο χρήστης στο να εξάγει τα διαγράμματα γρήγορα και αποτελεσματικά χωρίς να απαιτείται εκτεταμένη τροποποίηση του κώδικα ή πολύπλοκων φύλων στο excel. Η εφαρμογή, όπως ήδη αναφέρθηκε στο κεφάλαιο 3.3, χρησιμοποιεί τη βιβλιοθήκη ανοιχτού λογισμικού JfreeChart καθώς και τη βιβλιοθήκη γραφικού περιβάλλοντος Swing.

Κατά την εκκίνηση της εφαρμογής, το πρόγραμμα προσπαθεί να διαβάσει όλο το περιεχόμενο του καταλόγου results ο οποίος πρέπει να βρίσκεται στο path εκτέλεσης. Αν ο κατάλογος είτε δεν υπάρχει είτε δεν έχει καθόλου περιεχόμενο η εφαρμογή τερματίζει με ανάλογο μήνυμα σφάλματος. Το πρόγραμμα διαβάζει όλα τα αρχεία που βρίσκονται στον κατάλογο και τα οποία πρέπει να είναι στη μορφή που περιγράφεται στο κεφάλαιο 3.3. Τα δεδομένα του κάθε αρχείου (tags και χρόνοι) αποθηκεύονται σε αντικείμενα. Τα tags ομαδοποιούνται ώστε να μπορεί να τα επιλέξει ο χρήστης. Αφού ο χρήστης επιλέξει τα tags, δηλαδή τις κατηγορίες των διαγραμμάτων που θέλει να εξάγει, ξεκινάει η διαδικασία εξαγωγής των διαγραμμάτων.

Τα διαγράμματα εξάγονται ανά γραμμή αποτελεσμάτων (ερώτημα, πίνακας κοκ). Κάθε αρχείο αποτελεσμάτων φιλτράρεται βάσει των tags που έχει καθορίσει ο χρήστης. Για κάθε γραμμή αποτελεσμάτων εξάγεται ο μέσος όρος της και αποθηκεύεται σε ένα αντικείμενο τύπου CategoryDataset (ορίζεται στο JfreeChart). Από το CategoryDataset δημιουργείται ένα διάγραμμα του οποίου ο τίτλος καθορίζεται από το όνομα που δόθηκε στη γραμμή αποτελεσμάτων. Αν αυτό δεν καθορίζεται τότε χρησιμοποιείται ο αύξων αριθμός της γραμμής με το πρόθεμα “Query”. Στον τίτλο προστίθενται οι επιλογές των tags του χρήστη.

Τα διαγράμματα παραθέτονται σε δύο στήλες, σε ένα panel με δυνατότητα scrolling. Στη βάση της κάθε ράβδου εμφανίζεται η αριθμητική τιμή που εκπροσωπεί η ράβδος. Αξίζει να αναφερθεί ότι η προεπιλεγμένη συμπεριφορά της βιβλιοθήκης είναι να μην εμφανίζει την τιμή. Χρειάστηκαν ορισμένες τροποποιήσεις στον κώδικα ώστε να γίνει αυτό εφικτό. Το προεπιλεγμένο χρώμα των ραβδογραμμάτων είναι το κόκκινο. Ο χρήστης μπορεί να το αλλάξει κάνοντας κλικ

στο κουμπί “Chart Color...”. Τέλος για λόγους ευκολίας, προστέθηκε το κουμπί “Save Images" το οποίο αποθηκεύει όλα τα διαγράμματα στον κατάλογο images του path εκτέλεσης σε μορφή png. Ο κατάλογος πρέπει να προϋπάρχει.

ΕΠΙΛΟΓΟΣ

Στο κεφάλαιο αυτό παρουσιάστηκαν τα εργαλεία που χρησιμοποιήθηκαν για διάφορες εργασίες (πλην της εκτέλεσης του benchmark) όπως η μείωση του μεγέθους της βάσης και η εξαγωγή των διαγραμμάτων. Στο 6ο και τελευταίο κεφάλαιο παρατίθενται διαγράμματα από τα αποτελέσματα των δοκιμών.

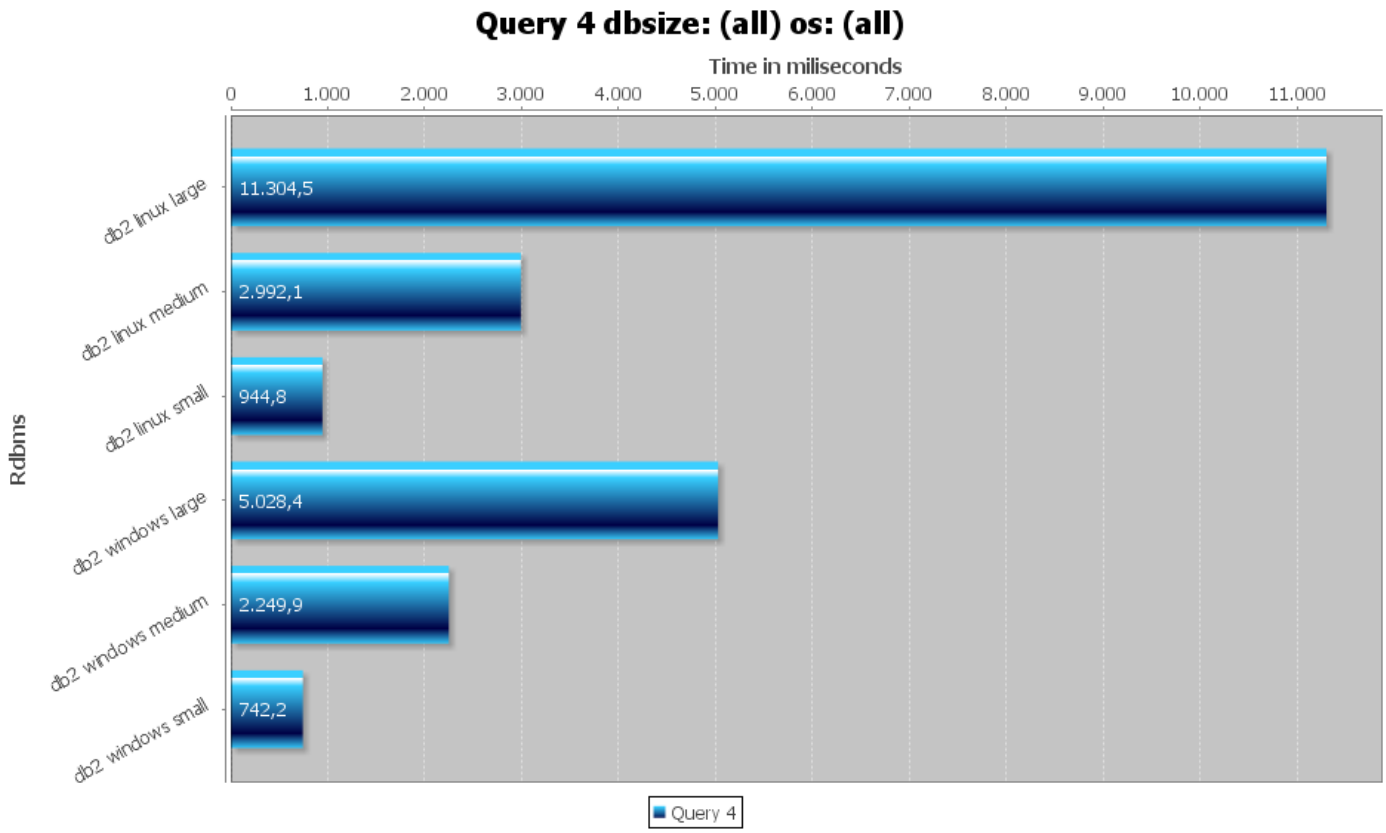
ΚΕΦΑΛΑΙΟ 6: Αποτελέσματα δοκιμών

ΕΙΣΑΓΩΓΗ

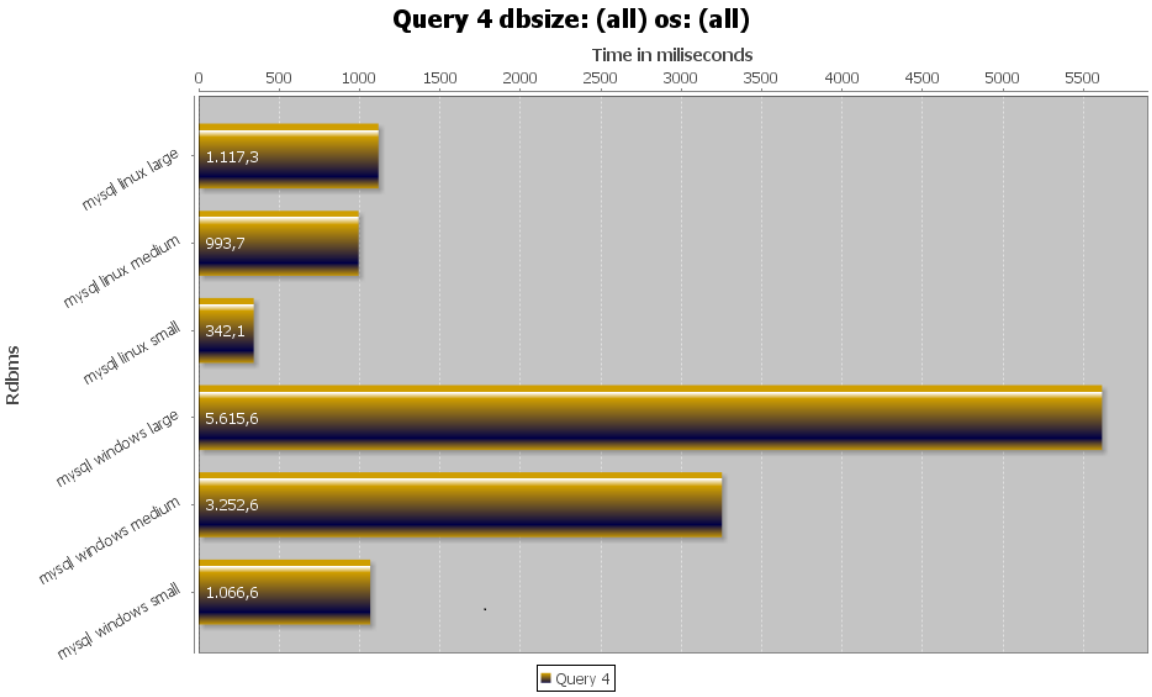
Τα αποτελέσματα των δοκιμών όπως ήδη αναφέρθηκε στο κεφάλαιο 3.3 εξάγονται σε μορφή αρχείου κειμένου ενώ υπάρχει και η δυνατότητα δημιουργίας διαγραμμάτων με τη χρήση του PlotDbChart. Στο 6ο και τελευταίο κεφάλαιο παρατίθενται αντιπροσωπευτικά διαγράμματα από την εκτέλεση των δοκιμών.

Τα τέσσερα πρώτα διαγράμματα είναι συγκριτικά για το κάθε rdbms για το ερώτημα 4 χωρίς τη χρήση indexes. Η σύγκριση γίνεται ανά λειτουργικό σύστημα και ανά μέγεθος βάσης δεδομένων. Στα 3 επόμενα διαγράμματα ακολουθεί σύγκριση όλων των RDBMS ανά λειτουργικό σύστημα χωρίς indexes. Τα διαγράμματα αντιστοιχούν ένα σε κάθε μέγεθος βάσης. Στα σχήματα 8 9 και 10 απεικονίζεται η ίδια σύγκριση με την χρήση indexes.

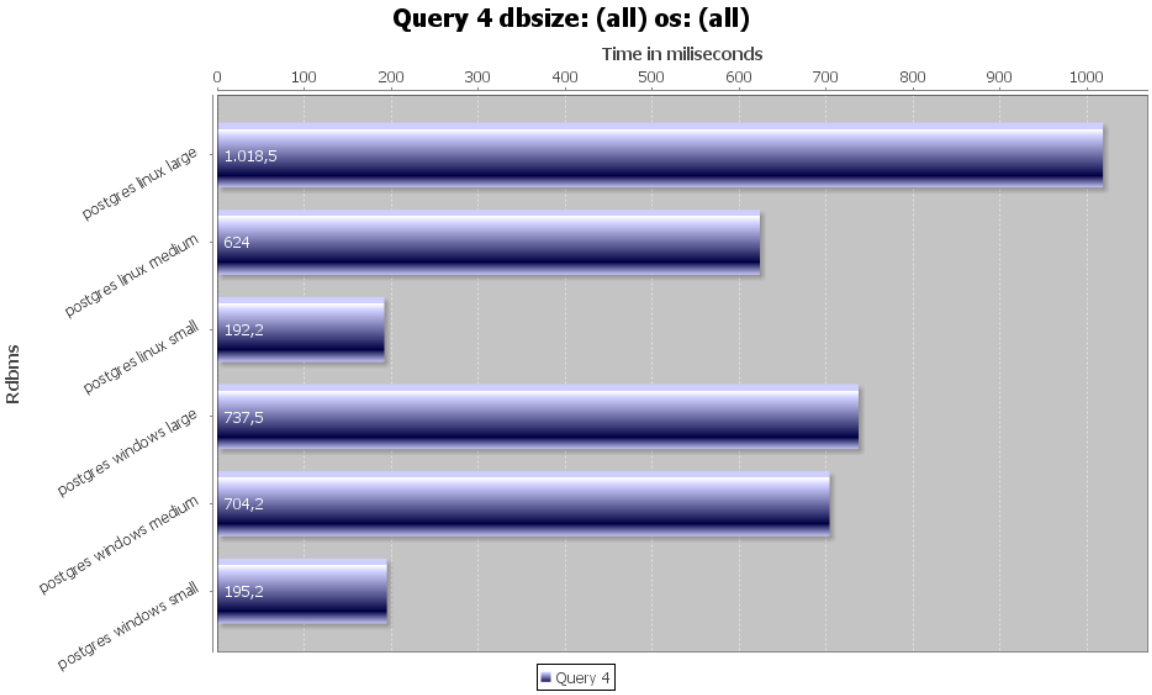
6.1 Σύγκριση RDBMS ανά OS και ανά μέγεθος βάσης για το ερώτημα 4



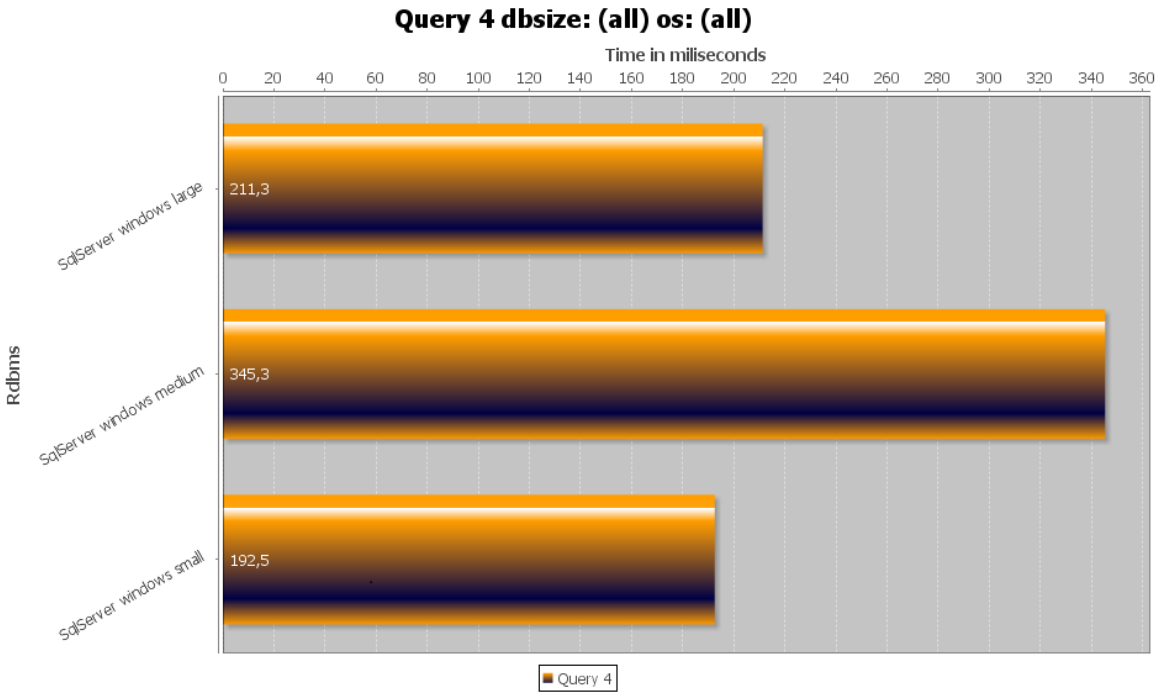
Σχήμα 1: DB2 Q4 σύγκριση μεγεθών και λειτουργικών συστημάτων



Σχήμα 2: MySql Q4 σύγκριση μεγεθών και λειτουργικών συστημάτων

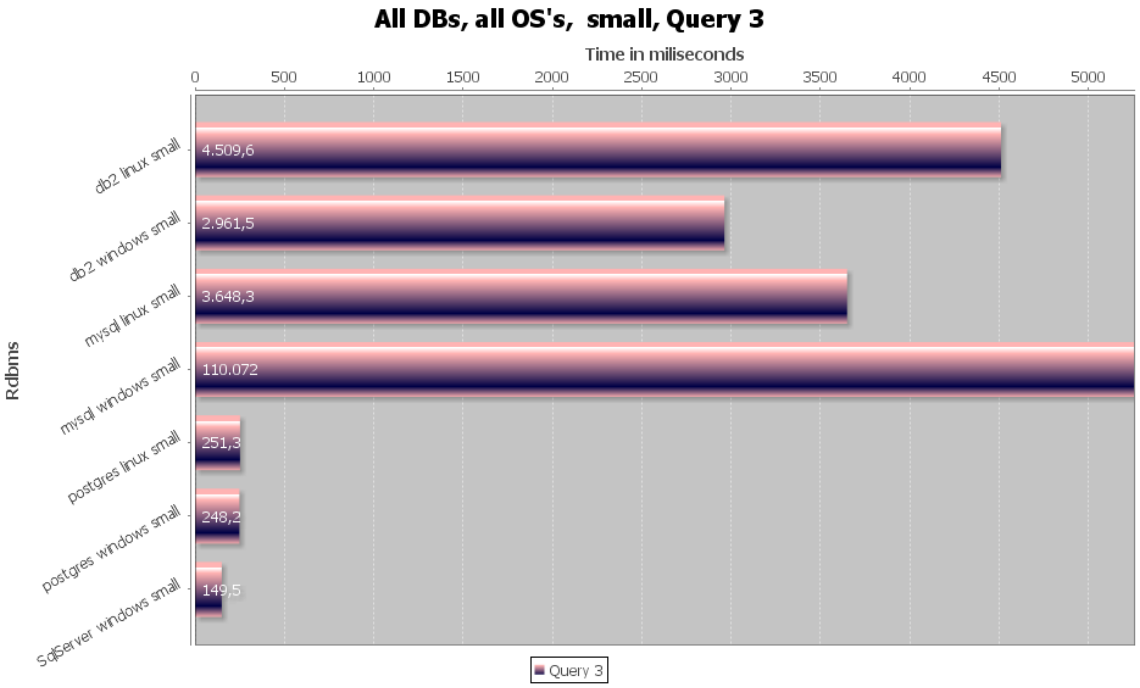


Σχήμα 3: Postgres Q4 σύγκριση μεγεθών και λειτουργικών συστημάτων

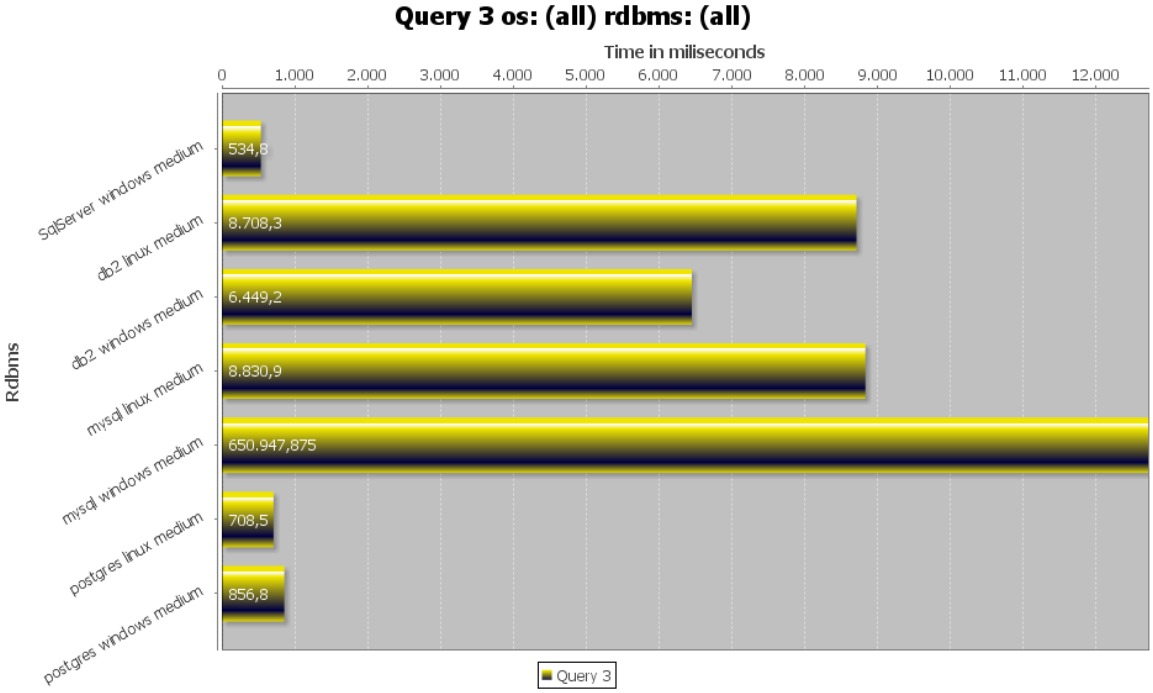


Σχήμα 4: SqlServer 2008 Q4 σύγκριση μεγεθών

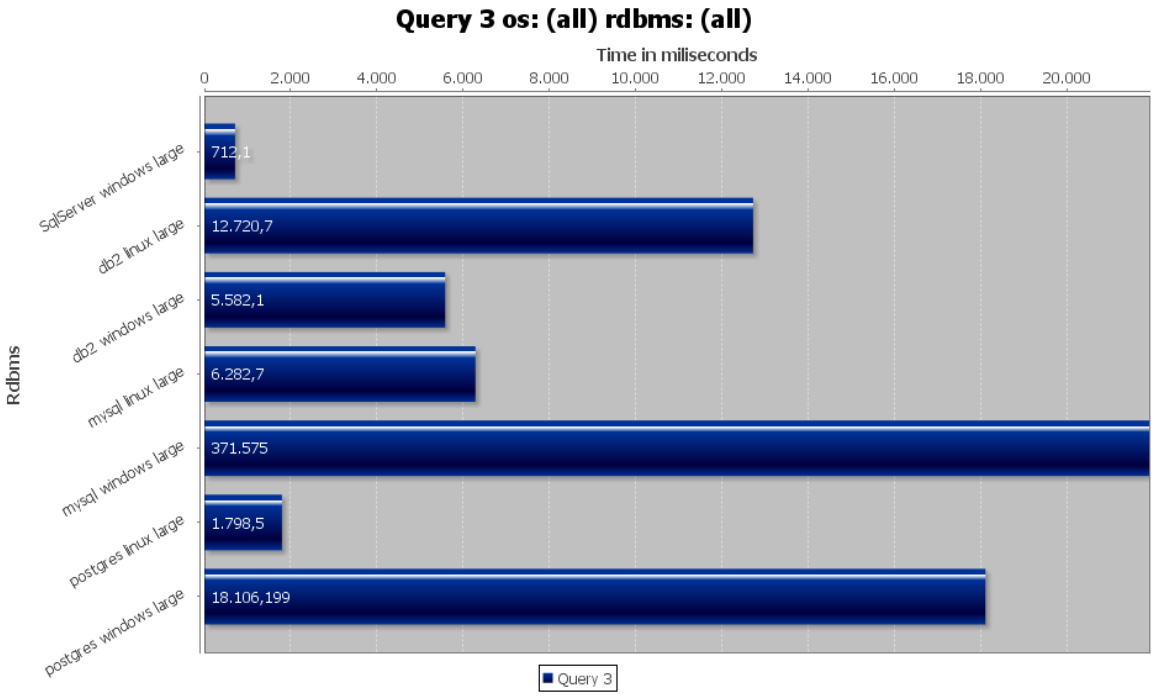
6.2 Σύγκριση όλων των RDBMS μεταξύ τους χωρίς indexes



Σχήμα 5: Σύγκριση όλων των RDBMS για όλα τα OS για τη μικρή βάση

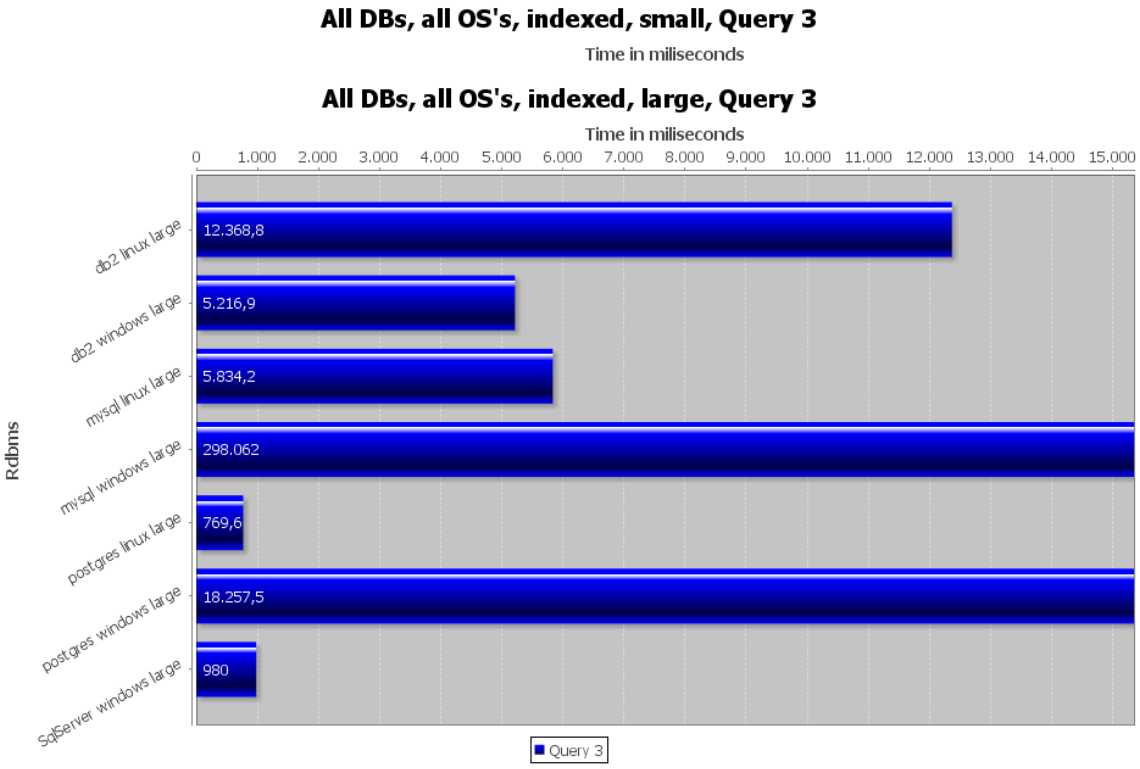


Σχήμα 6: Σύγκριση όλων των RDBMS για όλα τα OS για τη μεσαία βάση

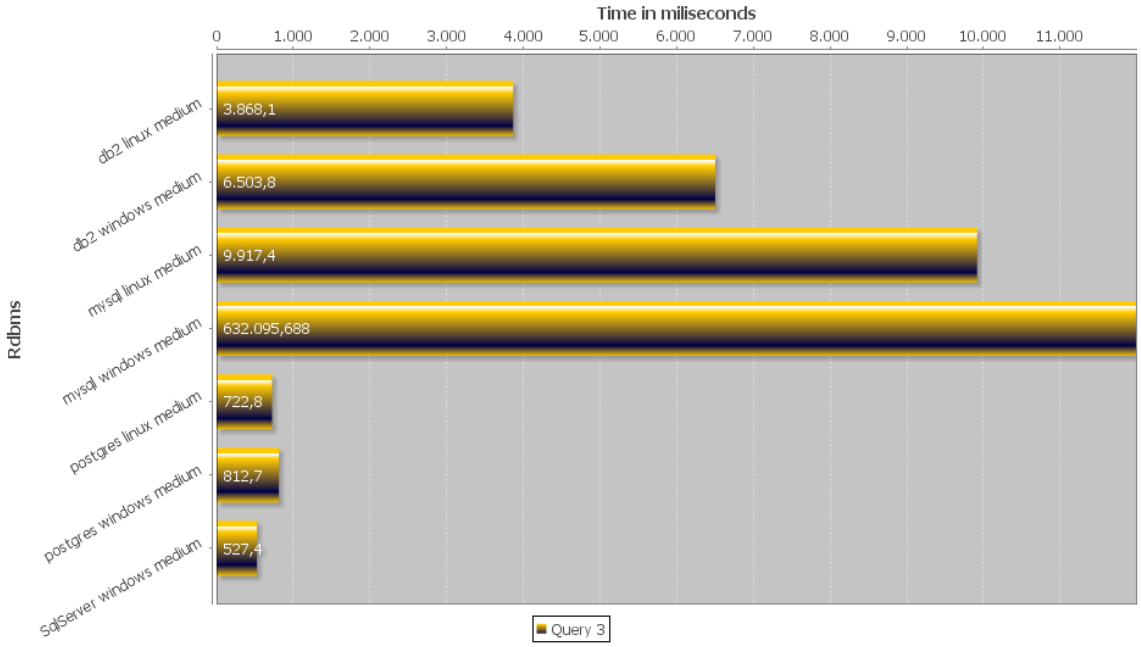


Σχήμα 7: Σύγκριση όλων των RDBMS για όλα τα OS για τη μεγάλη βάση

6.3 Σύγκριση όλων των RDBMS μεταξύ τους με τη χρήση indexes

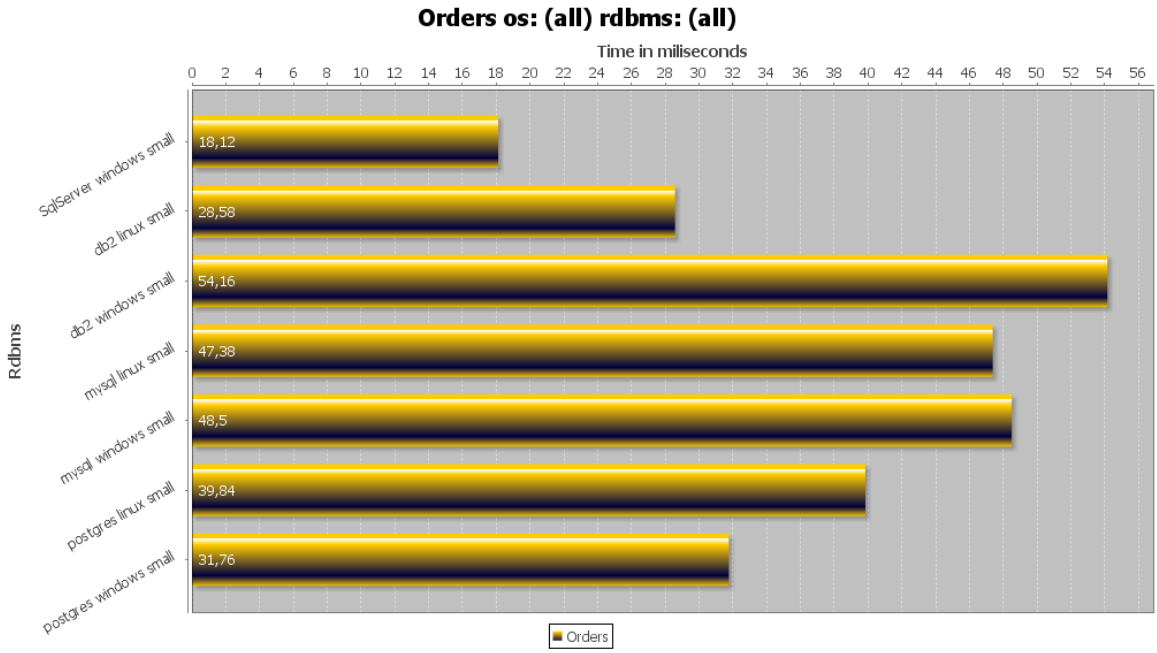


Σχήμα 10: Σύγκριση όλων των RDBMS για όλα τα OS για τη μεγάλη βάση (Indexed)

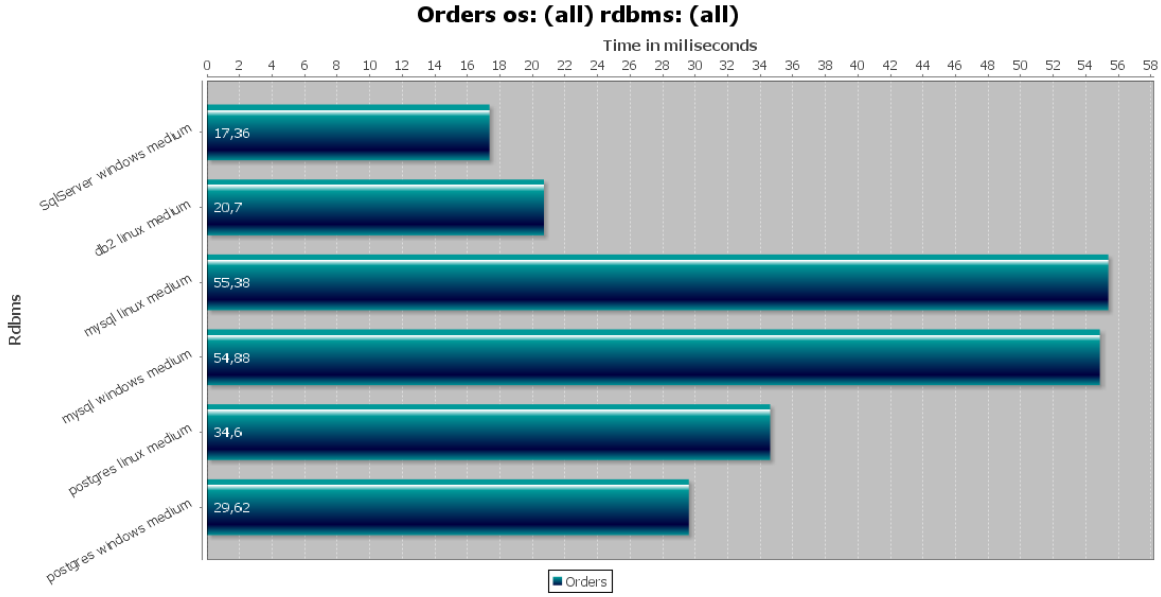


Σχήμα 9: Σύγκριση όλων των RDBMS για όλα τα OS για τη μεσαία βάση (Indexed)

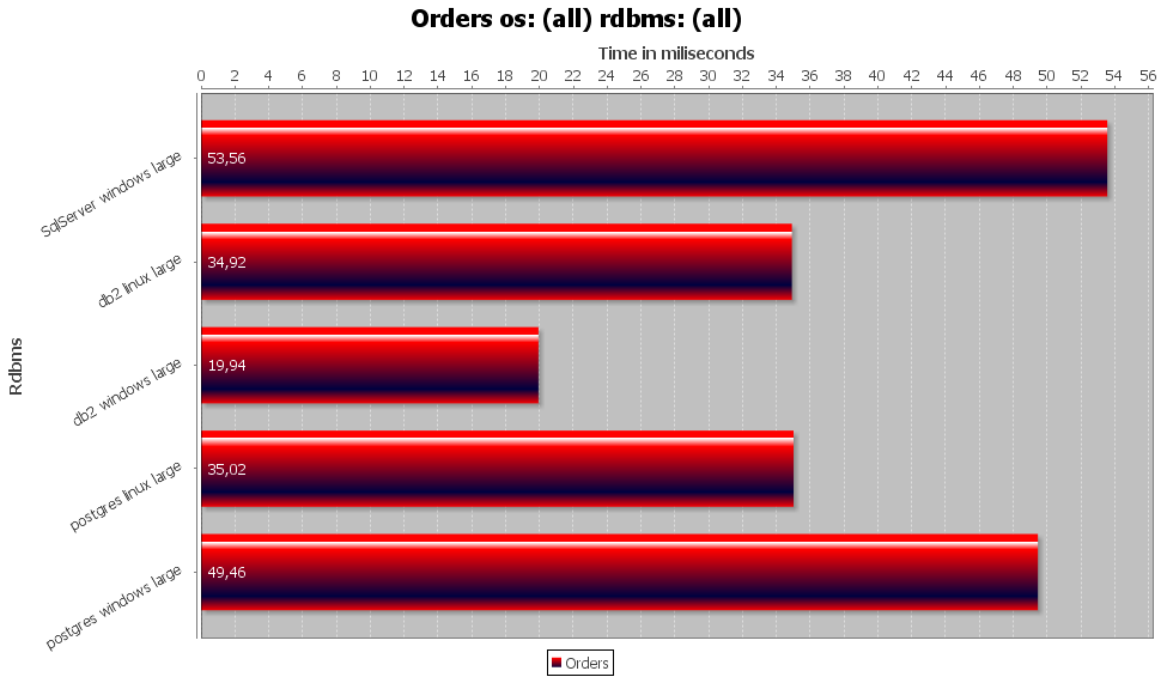
6.4 Διαγράμματα αποτελεσμάτων για Inserts



Σχήμα 11: Αποτελέσματα INSER INTO για τη μικρή βάση για τον πίνακα orders

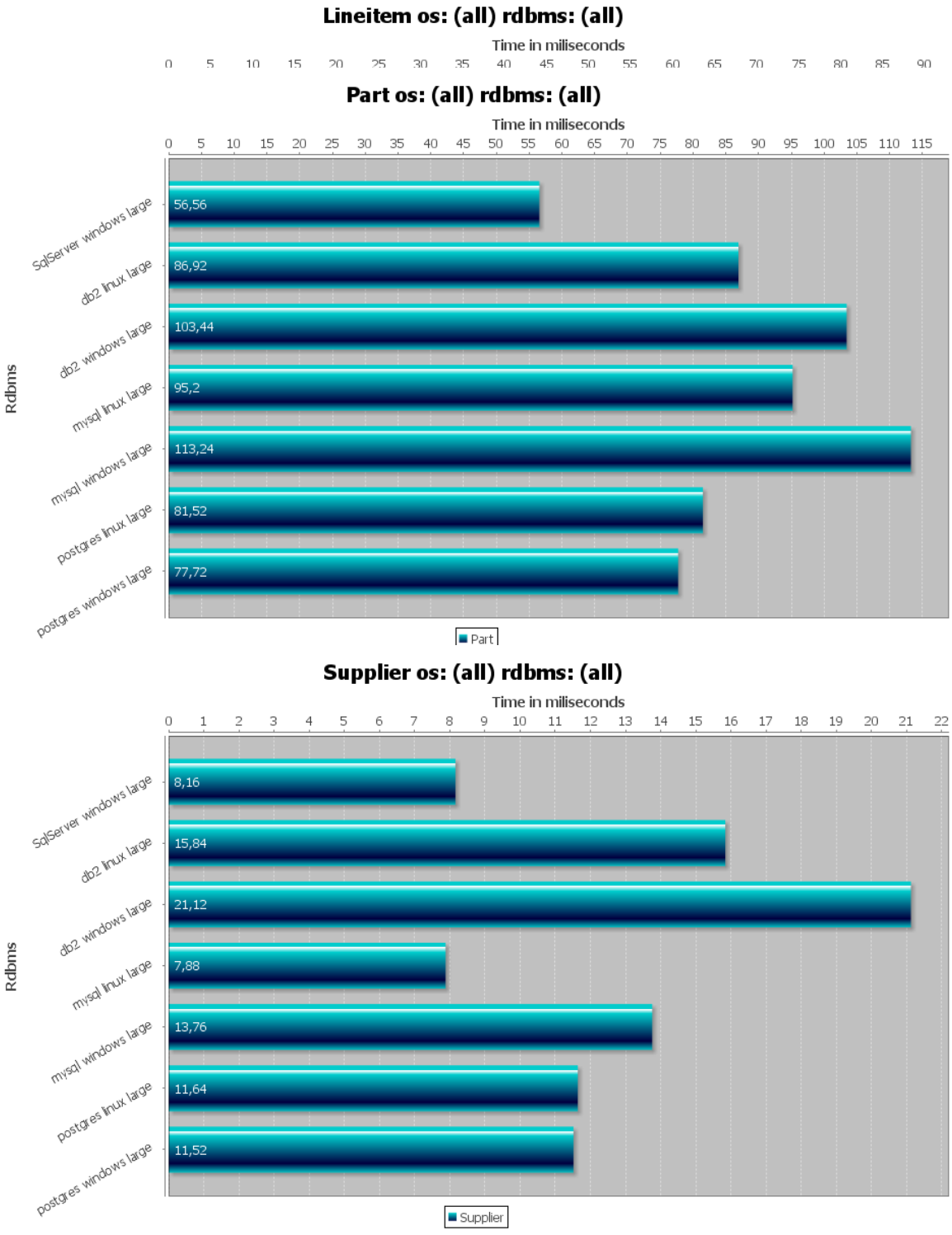


Σχήμα 12: Αποτελέσματα INSER INTO για τη μεσαία βάση για τον πίνακα orders



Σχήμα 13: Αποτελέσματα INSER INTO για τη μεγάλη βάση για τον πίνακα orders

6.5 Διαγράμματα αποτελεσμάτων για Updates



Σχήμα 17: Αποτελέσματα UPDATE για τη μεγάλη βάση για τον πίνακα supplier
Σχήμα 15: Αποτελέσματα UPDATE για τη μεγάλη βάση για τον πίνακα orders

ΕΠΙΛΟΓΟΣ

Με την παρουσίαση ορισμένων διαγραμμάτων ολοκληρώνεται η αναλυτική παρουσίαση της πτυχιακής εργασίας. Ακολουθούν τα συμπεράσματα που προέκυψαν από την εκπόνησή της καθώς και οι βιβλιογραφικές αναφορές πάνω στις οποίες τεκμηριώθηκε.

ΣΥΜΠΕΡΑΣΜΑΤΑ

Συγκρίνοντας τα αποτελέσματα των ερωτημάτων που προέκυψαν από τις δοκιμές στις βάσεις δεδομένων, προέκυψαν τα εξής συμπεράσματα για το κάθε RDBMS.

- Για την DB2 διαπιστώθηκε ότι αποδίδει καλύτερα σε λειτουργικό σύστημα Windows από ότι σε Linux. Η μείωση του χρόνου εκτέλεσης των ίδιων ερωτημάτων από τη μεγαλύτερη προς τη μικρότερη βάση φαίνεται να είναι σχεδόν γραμμική. Τα indexes στην DB2 για την πλειοψηφία των ερωτημάτων συνετέλεσαν σε θεαματική βελτίωση του χρόνου εκτέλεσης με ελάχιστες εξαιρέσεις όπου οι χρόνοι ήταν σχεδόν ίδιοι ή ελαφρώς χειρότεροι. Τα indexes φαίνεται να είναι αποτελεσματικότερα σε λειτουργικό σύστημα Windows και για βάσεις με πολλά δεδομένα.
- Η MySql παρουσίαζε εκθετική αύξηση του χρόνου εκτέλεσης ερωτημάτων όσο αυξάνεται το μέγεθος της βάσης. Όπως ήταν αναμενόμενο οι χρόνοι εκτέλεσης των ερωτημάτων σε Linux ήταν κατά πολύ μικρότεροι από τους αντίστοιχους σε Windows. Τα indexes στην πλειοψηφία των ερωτημάτων έδειξαν να παρέχουν από μικρή έως καμία βελτίωση στους χρόνους εκτέλεσης.
- Για την Postgres η εικόνα δεν είναι πολύ διαφορετική: οι επιδόσεις της σε Linux είναι αισθητά καλύτερες από ότι σε Windows. Τα indexes δε δείχνουν να επηρεάζουν σημαντικά τις επιδόσεις του RDBMS. Στην περίπτωση των Windows που οι χρόνοι εκτέλεσης είναι μεγαλύτεροι η διαφορά επιδόσεων μεταξύ των μεγεθών είναι σχεδόν εκθετική.
- Για τον SQL Server ο οποίος όπως είναι αναμενόμενο τρέχει μόνο σε Windows, είχε τις καλύτερες επιδόσεις από όλα τα υπό δοκιμή RDBMS. Η διαφορά των χρόνων εκτέλεσης μεταξύ των μεγεθών είναι γραμμική. Τα indexes στον SQL Server φάνηκαν να τον επιβράδυναν για τη συντριπτική πλειοψηφία των ερωτημάτων με ελάχιστες εξαιρέσεις.

Αναφορικά με το query optimization τα συμπεράσματα που προκύπτουν από το tweaking του κεφαλαίου 4 είναι ότι ο query optimizer των open source RDBMS της MySql κάνει συχνότερα λάθος επιλογές σε σχέση με τα υπόλοιπα rdbms και χρειάζεται την παρέμβαση του συντάκτη των ερωτημάτων. Παρατηρήθηκε ότι ο optimizer της MySql δεν έχει τη λογική να εκτελέσει μία φορά τα εμφωλευμένα ερωτήματα, όπου αυτό μπορεί να γίνει, αλλά και το cache mechanism είναι αρκετά απλοϊκό ώστε να μη μπορεί να κάνει cache τα εμφωλευμένα ερωτήματα. Από την άλλη ο optimizer της Postgres φάνηκε ότι το αδύναμο σημείο του είναι τα correlated ερωτήματα για τα οποία έδειχνε να μην μπορεί να σχεδιάσει αποτελεσματικά query plans.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Raghu Ramakrishnan, Johannes Gehrke [2000] Συστήματα διαχείρισης βάσεων δεδομένων 2η έκδοση Τόμος Α' pp 216-226, 481-500

Ramez Elmasri, Shamkant B. Navathe [2004] Fun damentals of database systems m, Fourth Edition. Pp 232, 233, 495-533

http://en.wikipedia.org/wiki/Relational_algebra#Semijoin_.28.E2.8B.89.29.28.E2.8B.8A.29

http://www.tokutek.com/2009/06/improving_tpc_h_like_queries_q17/

http://www.tokutek.com/2009/04/improving-tpc-h-like-queries-q2/

http://dbaspects.blogspot.com/2010/03/tpch-on-postgresql.html

http://ss64.com/nt/syntax-services.html

http://www.michael-thomas.com/tech/db2/db2_survival_guide.htm

http://www.interspire.com/content/2006/02/15/introduction-to-database-indexes/

http://msdn.microsoft.com/en-us/library/ms170572.aspx

http://msdn.microsoft.com/en-us/library/ms165702(v=sql.105).aspx

http://blogs.msdn.com/b/craigfr/archive/2006/07/19/671712.aspx

https://help.ubuntu.com/11.10/serverguide/mysql.html

http://dev.mysql.com/doc/refman/5.0/en/explain-output.html

http://dev.mysql.com/doc/refman/5.0/en/explain-extended.html

http://publib.boulder.ibm.com/infocenter/db2luw/v8/index.jsp?topic=/com.ibm.db2.udb.doc/welcome.htm

http://txcowboycoder.wordpress.com/2010/11/11/auto-execute-psql-commands-via-batch-file/

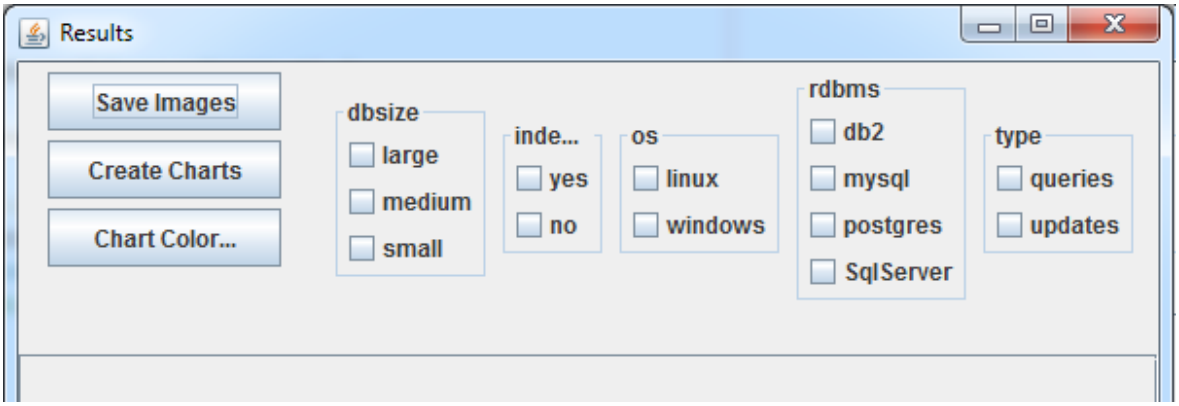
http://etutorials.org/SQL/Postgresql/Part+I+General+PostgreSQL+Use/Chapter+4.+Performance/Understanding+How+PostgreSQL+Executes+a+Query/

http://wiki.postgresql.org/wiki/TPC-H παραθέτει (αλλά δεν τεκμηριώνει) τη θεωρία ότι τα εμπορικά rdbms έχουν optimizations επί τούτου ώστε οι εταιρίες να μπορούν να πουν ότι έχουν άριστες επιδόσεις στο συγκεκριμένο benchmark.

ΟΔΗΓΟΣ ΧΡΗΣΗΣ ΛΟΓΙΣΜΙΚΟΥ

RunBenchmark: από αυτό το αρχείο ξεκινά η εκτέλεση του προγράμματος εκτέλεσης δοκιμών. Όπως ήδη αναφέρθηκε στο κεφάλαιο 3.2, η εφαρμογή διαβάζει το configuration file για να προσδιορίσει σε ποιο υπολογιστή να συνδεθεί, ποιο rdbms, το αρχείο με τα queries, τον αριθμό των επαναλήψεων και το αρχείο όπου θα αποθηκευτούν τα αποτελέσματα. Για την εκτέλεση των εντολών τροποποίησης της βάσης (insert, update και delete) προσθέτουμε στη γραμμή εντολών την παράμετρο -i.

PlotDbChart: το PlotDbChart είναι η Swing εφαρμογή εξαγωγής διαγραμμάτων. Στο σχήμα 5 βλέπουμε ένα στιγμιότυπο της εφαρμογής μόλις αυτή έχει ξεκινήσει. Δεξιά από τα κουμπιά ενεργειών βλέπουμε τις κατηγορίες των tags με τις δυνατές



Σχήμα 18: Το interface της εφαρμογής εξαγωγής διαγραμμάτων

τιμές τους. Τα checkboxes δημιουργούνται δυναμικά ανάλογα με τα αρχεία που υπάρχουν στον κατάλογο results.

Προσοχή πρέπει να δοθεί ώστε όλες οι κατηγορίες να έχουν τουλάχιστον μία τιμή επιλεγμένη, ακόμα και για τις περιπτώσεις που δεν έχουν νόημα όπως για παράδειγμα το tag indexed για την περίπτωση των updates και των inserts. Για την εξαγωγή αποτελεσμάτων για updates και inserts υποχρεωτικά πρέπει να επιλεγεί το “no” της κατηγορίας “indexed”.

Η επιλογή “Chart Color...” ανοίγει ένα modal dialog με έναν JColorChooser για την επιλογή του χρώματος των διαγραμμάτων. Το dialog στερείται κουμπιών “OK” και “Cancel”, το επιλεγμένο χρώμα αποθηκεύεται αμέσως μετά το κλείσιμο του παραθύρου.