ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΘΕΣΣΑΛΟΝΙΚΗΣ

ΠΤΥΧΙΑΚΗ

Διαδικτυακή βάση δεδομένων πληροφοριών ταινιών και υποτίτλων

Ονοματεπώνυμο: ΤΖΗΚΑΣ ΜΙΧΑΗΛ
Ετος: 2010
ΑΜ: 2543/04
Εισηγητής: ΑΝΤΩΝΗΣ ΣΙΔΗΡΟΠΟΥΛΟΣ

ΕΛΛΑΔΑ 2008
ΥΠΟΥΡΓΕΙΟ ΕΘΝΙΚΗΣ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ ΕΠΕΑΕΚ
ΕΥΡΩΠΑΪΚΗ ΕΝΩΣΗ
ΣΥΓΧΡΗΜΑΤΟΔΟΤΗΣΗ
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
ΕΥΡΩΠΑΪΚΟ ΤΑΜΕΙΟ ΠΕΡΙΦΕΡΕΙΑΚΗΣ ΑΝΑΠΤΥΞΗΣ
ΠΑΙΔΕΙΑ ΜΠΡΟΣΤΑ
2ο Επιχειρησιακό Πρόγραμμα
Εκπαίδευσης και Αρχικής
Επαγγελματικής Κατάρτισης

Περιεχόμενα

1. ΠΕΡΙΛΗΨΗ

Ο κόσμος του internet τα τελευταία χρόνια αυξάνεται με γρήγορους ρυθμούς, με συνέπεια της ραγδαία αύξηση της πληροφορίας που μπορεί κάποιος να αναζητήσει. Λίγα χρόνια νωρίτερα κατά πλειοψηφία υπήρχαν στατικοί ιστοχώροι τα οποία διαχειρίζονταν μικρή ποσότητα δεδομένων. Τότε που το internet ήταν προνόμιο λίγων δεν υπήρχε άμεση ανάγκη από μια αυτοματοποιημένη διαδικασία όπου κάποιος θα μπορούσε να καταχωρήσει νέες πληροφορίες. Ο κάθε διαχειριστής ήταν και ο προγραμματιστής του ιστοχώρου όπου με βασικές γνώσεις θα μπορούσε να το διαχειριστεί.

Αυτό όμως στις μέρες μας έχει αλλάξει ριζικά. Η ποσότητα δεδομένων που διοχετεύεται στο internet καθημερινά είναι δύσκολο αν όχι αδύνατον να μετρηθεί. Συναντάμε καθημερινά καινούργιους ιστοχώρους που μάλιστα μπορεί να ανανεώνονται και κάθε δευτερόλεπτο.

Όπως γίνεται αντιληπτό είναι αδύνατον να μπορεί να διαχειριστεί ένας ή περισσότεροι άνθρωποι ένα τέτοιο ιστοχώρο δίχως έναν αυτοματοποιημένο μηχανισμό.
Τα τελευταία χρόνια έχουν αναπτυχθεί πολλά συστήματα διαχείρισης περιεχομένου (CMS) όπου είναι ένας δυναμικός τρόπος κατασκευής ιστοχώρου χωρίς να χρειάζονται ιδιαίτερες γνώσεις. Μέσα σε ελάχιστα λεπτά κάποιος χωρίς ιδιαίτερες γνώσεις μπορεί να κατασκευάσει έναν ιστοχώρο και να διοχετεύσει δεδομένα στο κοινό.

Πολλές φορές όμως τα έτοιμα συστήματα διαχείρισης περιεχομένου, που είτε είναι δωρεάν είτε επί πληρωμή, δεν εξυπηρετούν τις ανάγκες μας αφού κάθε ιστοχώρος έχει συγκεκριμένες απαιτήσεις και ιδιαιτερότητες. Αν και υπάρχει η δυνατότητα επέκτασης ενός συστήματος διαχείρισης περιεχομένου με βάση τις ανάγκες μας, στην εργασία αυτή θα ασχοληθούμε με την κατασκευή μιας διαδικτυακής βάση

δεδομένων που θα περιέχει πληροφορίες ταινιών, τους υπότιτλους τους και δεδομένα σχετικά με ταινίες.

Θα δίνει την δυνατότητα στους χρήστες-μέλη να βρίσκουν πληροφορίες για ταινίες, σκηνοθέτες, ηθοποιούς, κατηγορίες, trailers, reviews, σχόλια, υπότιτλους κ.ά.. Ιδιαίτερη βαρύτητα θα δοθεί στους υπότιτλους και στις νέες κυκλοφορίες αυτών. Επίσης θα δίνεται η δυνατότητα στους χρήστες να καταχωρούν και να διαβάζουν νέα του κινηματογράφου αλλά και να ενημερώνονται για κυκλοφορίες DVD, και πρόγραμμα τηλεόρασης.

Τέλος θα δίνεται η δυνατότητα επικοινωνίας των χρηστών μέσω forum ειδικά διαμορφωμένο για τον τύπο των δεδομένων που υπάρχουν στην βάση. Η βάση θα ενημερώνεται από χρήστες (υπό τον έλεγχο των διαχειριστών) και από διαχειριστές με την χρήση συστήματος διαχείρισης περιεχομένου. Η διαχείριση του ιστοχώρου θα γίνεται μέσω ενός διαμορφωμένου Control Panel από διαχειριστές και μέλη με ειδική πρόσβαση.

Εργαλεία και τεχνολογίες που θα χρησιμοποιηθούν: Apache web server, MySQL, PHP, Javascript, Ajax, Html, XML, CSS, jQuery.

2. ΣΤΟΧΟΙ

Όπως κάθε διαδικασία έτσι και εδώ πρέπει να εφαρμοστούν τεχνικές για να κατασκευαστεί η εφαρμογή χωρίς άσκοπες αναδρομές από πιθανά λάθη ή παραλήψεις.

Βασική ανάγκη ήταν να καταγραφούν οι στόχοι του ιστοχώρου και η σημαντικότητα τους. Έτσι δημιουργήθηκε ο παρακάτω πίνακας που περιέχει τα ανάλογα στοιχεία

Α/Α	Στόχος	Σημαντικότητα
1	Υπότιτλοι	1
2	Ταινίες	1
3	Forum	2
4	Πρόγραμμα Τηλεόρασης	4
5	Box Office	3
6	Σχόλια	2
7	Μέλη	1
8	Διαφημιστικός Χώρος	2
9	Κινηματογραφικά Νέα	3
10	Requests Υπότιτλων	4
11	Επικοινωνία Χρήστη – Admin	1
12	Upload	1

Αυτός ο πίνακας δείχνει με μια σύντομη ματιά τι θα πρέπει να υλοποιηθεί στον ιστοχώρο και πόσο σημαντικό είναι για το όλο project.

Όπως είναι φυσικό κάποια χαρακτηριστικά μπορούν να προστεθούν κατά την διάρκεια ζωής του, και όπως θα δούμε, έτσι έγινε.

Ένα βασικό κομμάτι της ανάλυσης ήταν να αναλυθεί το κομμάτι των ταινιών. Θα έπρεπε να αποφασιστεί τι πληροφορίες θα έπρεπε, να συγκεντρωθούν για κάθε ταινία. Έτσι δόθηκε προσοχή στην γνωστότερη βάση δεδομένων ταινιών του που υπάρχει, το www.imdb.com. Εκεί βρίσκονται όλες οι απαραίτητες πληροφορίες για το κομμάτι αυτό. Το συμπέρασμα λοιπόν είναι ότι χρειάζονται τα εξής χαρακτηριστικά:

1	Ελληνικός Τίτλος
2	Ξένος Τίτλος
3	Είδος Ταινίας
4	Επίσημη Πρεμιέρα
5	Ελληνική Πρεμιέρα
6	Πρεμιέρα DVD
7	Σκηνοθέτης
8	Σεναριογράφος
9	Ηθοποιοί
10	Εικόνα
11	Υπόθεση
12	Γλώσσα
13	Trailer
14	Διάρκεια
15	Rating
16	IMDB κωδικός
17	Διάφορα
18	Τηλεοπτική Σειρά (κατηγοριοποίηση για το αν είναι τηλεοπτική σειρά η ταινία)

Παρόλο που είχε αποφασιστεί πια θα ήταν τα δεδομένα, η κατασκευή του ιστοχώρου δεν θα ήταν τόσο απλή αφού θα έπρεπε να κατασκευαστεί με προϋποθέσεις. Προϋποθέσεις που θα την καθιστούσαν γρήγορη και αποτελεσματική, αφού προβλεπόταν αυξημένη κινητικότητα στον ιστοχώρο. Άρα

η ανάγκη για optimization ήταν πολύ σημαντική αφού πιθανές καθυστερήσεις στο σύστημα μας θα απογοήτευαν τους χρήστες μας.

3. ΕΙΣΑΓΩΓΗ ΣΤΙΣ ΤΕΧΝΟΛΟΓΙΕΣ

Ξεκινώντας, μια βασική ανάγκη ήταν να αποφασιστούν οι τεχνολογίες/γλώσσες που θα χρησιμοποιηθούν. Μετά από προσεκτική μελέτη αποφασίστηκε πως θα χρησιμοποιηθούν οι εξής:

HTTP : Hypertext Transfer Protocol, όπου είναι το βασικό πρωτόκολλο μεταφοράς δεδομένων

HTML: Η HTML είναι το ακρωνύμιο των λέξεων HyperText Markup Language, δηλ. Γλώσσα Χαρακτηρισμού Υπέρ-Κειμένου και βασίζεται στη γλώσσα SGML, Standard Generalized Markup Language, που είναι ένα πολύ μεγαλύτερο σύστημα επεξεργασίας εγγράφων. Η HTML ορίζει ένα σύνολο κοινών στυλ για τις Web σελίδες, όπως τίτλοι (titles), επικεφαλίδες (headings), παράγραφοι (paragraphs), λίστες (lists) και πίνακες (tables). Ορίζει επίσης στυλ χαρακτήρων, όπως η έντονη γραφή (boldface) και οι ενότητες κώδικα. Κάθε στοιχείο έχει ένα όνομα και περιέχεται μέσα στα σύμβολα <>, που αποκαλούνται tags (ετικέτες). Όταν γράφουμε μια Web σελίδα με την HTML, στην ουσία δίνουμε τίτλους στα διάφορα στοιχεία της σελίδας μ' αυτά τα tags. Οι φυλλομετρητές, μαζί με τη δυνατότητά τους να ανακτούν σελίδες από το Web, λειτουργούν επίσης και σαν μορφοποιητές για την HTML. Όταν διαβάζουμε μια σελίδα γραμμένη με την HTML σ' έναν φυλλομετρητή, ο φυλλομετρητής διαβάζει (διερμηνεύει) τα tags της HTML και μορφοποιεί το κείμενο και τις εικόνες στην οθόνη. Διαφορετικοί φυλλομετρητές, οι οποίοι τρέχουν σε διαφορετικούς υπολογιστές, μπορεί να αντιστοιχίζουν διαφορετικά στυλ σε κάθε στοιχείο μιας σελίδας. Αυτό σημαίνει ότι οι σελίδες που δημιουργούμε με την HTML μπορεί να δείχνουν εντελώς διαφορετικές από σύστημα σε σύστημα και από φυλλομετρητή σε φυλλομετρητή. Δηλαδή, οι πραγματικές πληροφορίες και οι σύνδεσμοι που περιέχουν οι σελίδες

μας θα είναι πάντα εκεί, αλλά η εμφάνιση των σελίδων στην οθόνη θα είναι διαφορετική.

PHP : Η PHP είναι μια γλώσσα προγραμματισμού για τη δημιουργία σελίδων web με δυναμικό περιεχόμενο. Μια σελίδα PHP περνά από επεξεργασία από ένα συμβατό διακομιστή του Παγκόσμιου Ιστού (π.χ. Apache), ώστε να παραχθεί σε πραγματικό χρόνο το τελικό περιεχόμενο, που θα σταλεί στο πρόγραμμα περιήγησης των επισκεπτών σε μορφή κώδικα HTML.. Ανταγωνιστικές της τεχνολογίας PHP είναι οι εξής γλώσσες προγραμματισμού : ASP (Active Server Pages) της εταιρείας Microsoft, CFML (ColdFusion Markup Language) της εταιρείας Allaire και JSP (JavaServer Pages) της εταιρείας Sun. Το μεγαλύτερο μέρος της σύνταξής της, η PHP το έχει δανειστεί από την C, την Java και την Perl και διαθέτει και μερικά δικά της μοναδικά χαρακτηριστικά. Ο σκοπός της γλώσσας είναι να δώσει τη δυνατότητα στους web developers να δημιουργούν δυναμικά παραγόμενες ιστοσελίδες.

MySQL, PostgreSQL : Βάσεις δεδομένων. Θεμέλιο λίθος στην ανάπτυξη δυναμικών ιστοχώρων. Η SQL αποτελεί την standard γλώσσα για αλληλεπίδραση με τις περισσότερες βάσεις δεδομένων. Δεν πρέπει να συγχέουμε την SQL με την MySQL. Η MySQL είναι το λογισμικό του διακομιστή βάσεων δεδομένων (database server software) που χρησιμοποιούμε, ενώ η SQL είναι η γλώσσα που χρησιμοποιούμε για να αλληλεπιδράσουμε με την βάση δεδομένων. Μια Βάση Δεδομένων (DataBase) είναι ένας οργανωμένος τρόπος αποθήκευσης πληροφοριών και πρόσβασής τους με πολλούς τρόπους με διάφορα προγράμματα. Μια βάση δεδομένων είναι κάτι παραπάνω από μια απλή συλλογή αποθηκευμένων στοιχείων. Ένας άλλος ορισμός είναι ότι μια βάση δεδομένων είναι ένα ολοκληρωμένο σύστημα που αποτελείται από δεδομένα (data) και από το κατάλληλο λογισμικό (software), τα οποία χρησιμοποιώντας το υλικό (hardware) βοηθούν στην ενημέρωση και πληροφόρηση των χρηστών (users). Ένα πρόγραμμα που διαχειρίζεται βάσεις δεδομένων αποκαλείται Σύστημα Διαχείρισης Βάσεων Δεδομένων (DBMS, DataBase Management System) και με την βοήθειά του μπορούμε να αποθηκεύσουμε, προσθέσουμε, τροποποιήσουμε,

εμφανίσουμε ή και διαγράψουμε τα αποθηκευμένα δεδομένα. Τα δεδομένα που υπάρχουν στις βάσεις δεδομένων πρέπει να είναι : * Ολοκληρωμένα (Integrated), δηλ. τα δεδομένα πρέπει να είναι αποθηκευμένα σε ομοιόμορφα οργανωμένα σύνολα αρχείων όπου δεν πρέπει να υπάρχει επανάληψη ή πλεονασμός (redundancy) των ίδιων στοιχείων. * Καταμεριζόμενα (Shared), δηλ. να μπορούν περισσότεροι του ενός χρήστες να βλέπουν και να μοιράζονται τα ίδια δεδομένα την ίδια χρονική στιγμή.

Javascript: Η JavaScript είναι μια γλώσσα συγγραφής σεναρίων (scripting language) που χρησιμοποιείται για να προσθέσει εφέ και διαλογικότητα (αλληλεπίδραση, διαδραστικότητα, interactivity) στις ιστοσελίδες μας και είναι ανταγωνιστική της γλώσσας προγραμματισμού VBScript. Δημιουργήθηκε από την εταιρεία Netscape και το αρχικό της όνομα ήταν LiveScript. Ο κώδικας της JavaScript γράφεται σε καθαρό κείμενο (ASCII μορφή) και ενσωματώνεται μέσα στον κώδικα της HTML, μπορεί να εκτελεστεί αμέσως ή όταν λαμβάνει χώρα ένα συμβάν (event). Δεν γίνεται μεταγλώττιση (compilation) του κώδικα της JavaScript, αρκεί μόνο ο φυλλομετρητής (browser) να υποστηρίζει την JavaScript.

jQuery: Βιβλιοθήκη της Javascript που έχει ως σκοπό την διευκόληνση στην ανάπτυξη του κώδικα μέ έτοιμες διαδικασίες έυκολης σύνταξης.

4. Η ΑΝΑΓΚΗ ΓΙΑ ΕΝΑ DATABASE-DRIVEN WEB SITE

Σήμερα στο Web, το περιεχόμενο (content) είναι αυτό που κυριαρχεί. Με την χρήση HTML, JavaScript και τη Δυναμική (Dynamic) HTML, μπορούν να δημιουργηθούν ιστοχώροι (web sites) με εντυπωσιακή εμφάνιση. Έρχεται, όμως, η ώρα που πρέπει να γεμίσουν οι σελίδες με πραγματικές πληροφορίες (real information). Όποιος ιστοχώρος τραβάει το ενδιαφέρον των επισκεπτών επανειλημμένα, θα πρέπει να περιέχει καινούργιο και συνεχώς ανανεωμένο περιεχόμενο.

Το πρόβλημα είναι ότι συχνά, οι άνθρωποι που παρέχουν το περιεχόμενο για ένα ιστοχώρο δεν είναι οι ίδιοι μ' αυτούς που κάνουν και τη σχεδίασή του (design). Αλλά είναι σύνηθες, ο παροχέας του περιεχομένου να μην γνωρίζει καθόλου από HTML. Πώς, όμως, ο παροχέας θα μπορέσει να το εγκαταστήσει στο Web site;

Δεν μπορούν φυσικά όλες οι εταιρείες να απασχολούν πλήρως έναν Webmaster και οι περισσότεροι από τους Webmasters έχουν άλλες δουλειές να κάνουν από το να αντιγράφουν αρχεία του Word μέσα σε tags της HTML. Η λύση σ' αυτό το πρόβλημα είναι το database-driven site design. Πετυχαίνοντας τον πλήρη διαχωρισμό ανάμεσα στη σχεδίαση του ιστοχώρου και το περιεχόμενο που παρουσιάζεται μέσα σ' αυτό, είναι δυνατόν να αξιοποιηθούν ξεχωριστά οι ανάγκες.

Αντί να δημιουργηθεί ένα HTML αρχείο για την κάθε σελίδα του ιστοχώρου, χρειάζεται μόνο να δημιουργηθεί μια σελίδα για το κάθε είδος πληροφορίας που πρέπει να παρουσιαστεί. Αντί να επικολλείται συνέχεια νέο περιεχόμενο στις σελίδες, δημιουργείτε ένα απλό σύστημα διαχείρισης περιεχομένου (content management system) που δίνει τη δυνατότητα στους συγγραφείς να δημοσιεύουν (post) το καινούργιο περιεχόμενό τους οι ίδιοι χωρίς να κάνουν καθόλου χρήση της HTML.

Για να δημιουργηθεί ένα database-driven Web site, θα χρησιμοποιηθεί η γλώσσα συγγραφής σεναρίων στην πλευρά του διακομιστή (server-side scripting language) PHP και η σχεσιακή βάση δεδομένων (relational database) MySQL. Θα πρέπει, όμως, ο πάροχος στον οποίο δημοσιεύονται οι σελίδες να υποστηρίζει τον συνδυασμό PHP/MySQL.

5. Η PHP ΚΑΙ Η MYSQL

Τα εργαλεία που θα χρησιμοποιηθούν γι' αυτή τη δουλειά θα είναι τα προγράμματα PHP και MySQL. Η PHP είναι μια γλώσσα συγγραφής σεναρίων στην πλευρά του διακομιστή (server-side scripting language). Είναι σαν ένα πρόσθετο (plug-in) για τον Web server που του δίνει τη δυνατότητα να κάνει κάτι περισσότερο από το να στέλνει απλά και μόνο απλές ιστοσελίδες (Web pages) όταν τις ζητάνε οι φυλλομετρητές.

Με εγκατεστημένη την PHP, ο Web server θα είναι σε θέση να διαβάζει ένα νέο είδος αρχείων, το αποκαλούμενο PHP script, το οποίο μπορεί να κάνει εργασίες όπως ανάκτηση πληροφοριών της τελευταίας στιγμής (retrieve up-to-the-minute information) από μια βάση δεδομένων και καταχώρησή τους σε μια Web page πριν αυτή αποσταλεί στον φυλλομετρητή που τη ζήτησε.

Για να μπορέσουν να ανακτηθούν οι πληροφορίες από μια βάση δεδομένων, θα πρέπει πρώτα φυσικά να υπάρχει μια βάση δεδομένων (database) και αυτός είναι ο λόγος της παρουσίας της MySQL. Η MySQL είναι ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων (relational database management system) ή RDBMS. Σε γενικές γραμμές είναι ένα πακέτο λογισμικού (software package) που είναι πολύ καλό στην οργάνωση και τη διαχείριση μεγάλων ποσοτήτων πληροφοριών.

Με τη χρήση της MySQL είναι εύκολη η πρόσβαση σ' αυτές τις πληροφορίες χρησιμοποιώντας μια γλώσσα συγγραφής σεναρίων στην πλευρά του διακομιστή (server-side scripting languages), όπως είναι η PHP. Ο κώδικας που θα

χρησιμοποιηθεί εδώ θα μπορεί να εφαρμοσθεί σ' έναν διακομιστή που βασίζεται στα Windows ή στο Unix.

6. WEB SERVERS

A. Shared Hosting

Το shared web hosting είναι μια υπηρεσία όπου πολλά websites στεγάζονται σε έναν web server που αυτός συνδέετε στο Internet. Κάθε site του server παραμένει ξεχωριστά από όλα τα υπόλοιπα sites έχοντας όμως κοινές ρυθμίσεις. Γενικά, είναι η πιο οικονομική επιλογή για web hosting καθώς πολλοί άνθρωποι μοιράζονται το συνολικό κόστος συντήρησης αυτού του μηχανήματος.

Η υπηρεσία hosting πρέπει οπωσδήποτε να έχει κάποια διαχείριση αφού μοιράζεται σε πολλά άτομα τα οποία μάλιστα δεν γνωρίζουν την ύπαρξη του άλλου. Αυτό είναι ένα πλεονέκτημα σε αυτούς που δεν θέλουν να ασχοληθούν με θέματα διαχείρισης, αλλά είναι μειονέκτημα γι' αυτούς που θέλουν να έχουν καλύτερο έλεγχο των ρυθμίσεων. Γενικά το shared hosting είναι ακατάλληλο για χρήστες που χρειάζονται εξειδικευμένο λογισμικό που δεν παρέχετε-υποστηρίζεται από την εταιρία που μας παρέχει στον web server. Συνήθως οι πιο χρήσιμες εφαρμογές υποστηρίζονται στον shared server, όμως ο shared server έχει κάποια όρια χρήσης τα οποία προκύπτουν από τον διαχωρισμό των ολικών πόρων του συστήματος.

Β. Πως αναπτύσσεται ένας ιστοχώρος

Ένα από τα βασικότερα θέματα που έπρεπε να αντιμετωπιστούν ήταν το «στήσιμο» ενός ιστοχώρου. Το πρώτο θέμα όπου χρειαζόταν μια απόφαση ήταν η ονομασία του ιστοχώρου. Έπρεπε να είναι μια ονομασία σχετική με το περιεχόμενο και ταυτόχρονα να είναι ελεύθερο και το domain. Όπως θα διαπιστωθεί αργότερα, η ονομασία-domain παίζει πολύ σημαντικό ρόλο σε

θέματα CEO! Έτσι λοιπόν μετά από αναζήτηση το όνομα movieplace.gr θεωρήθηκε ως το καταλληλότερο.

Αυτή ήταν η πρώτη φάση και η πιο εύκολη. Το θέμα που προέκυπτε τώρα ήταν η επιλογή web hosting. Χωρίς κάποια ιδιαίτερη εμπειρία σε θέματα hosting, ο web developer μπορεί να πέσει στην παγίδα του server χαμηλού κόστους, δηλαδή shared hosting.

Οι μόνες απαιτήσεις ήταν η υποστήριξη PHP4 και MySQL 4. Έτσι λοιπόν ο ιστοχώρος στήθηκε και λειτούργησε κανονικά για τους πρώτους 2 μήνες όπου και η επισκεψιμότητα ήταν χαμηλή. Μετά το πέρας των 2 μηνών όπου και η επισκεψιμότητα άρχισε να αυξάνεται με γρήγορους ρυθμούς, παρατηρήθηκαν αρκετές καθυστερήσεις στην επεξεργασία των SQL ερωτημάτων. Στην αρχή θεωρήθηκε λανθασμένα πως ήταν κάποιο αρχιτεκτονικό πρόβλημα και γι' αυτό έγιναν προσπάθειες να γίνουν πιο ελαφριά κώδικας και ερωτήματα, όπως θα δούμε και στις επόμενες ενότητες. Όντως η ταχύτητα βελτιώθηκε αλλά παρόλα αυτά τα προβλήματα εξακολούθησαν να υπάρχουν.

Γ. Semi-Dedicated

Μετά από αναζήτηση πληροφοριών σχετικά με την επίδοση ενός ιστοχώρου, διαπιστώθηκε πως πολύ σημαντικό ρόλο έπαιζε ο server. Έτσι χρησιμοποιήθηκε semi-dedicated server όπου στην ουσία είναι μια υπηρεσία διαχωρισμού των πόρων από 4 μέχρι 6 «κομμάτια», σε αντίθεση με τους shared που μπορεί να είναι 100-500, που μοιράζονται τους συνολικούς πόρους. Η μεταφορά έγινε και όντως το ήταν αρκετά γρηγορότερο.

Αρκετό καιρό μετά όμως όπου η επισκεψιμότητα είχε φτάσει σε πάρα πολύ υψηλά επίπεδα (4000visits/day) τα προβλήματα άρχισαν να ξανά εμφανίζονται. Σε εκείνο το σημείο όπου το κόστος ήταν απαγορευτικό για μια ακόμα αναβάθμιση, το optimization ήταν η καλύτερη λύση έτσι ώστε να αποφευχθεί μια ακόμα μεταφορά του server.

Όντως η βελτιστοποίηση βοήθησε αρκετά και για αρκετό καιρό ο ιστοχώρος δεν αντιμετώπιζε κανένα πρόβλημα, μέχρι όμως την στιγμή όπου μια νέα αναβάθμιση του ιστοχώρου το έκανε να έχει ανάγκη σε περισσότερους πόρους (RAM). Έτσι για μια ακόμα φορά αντιμετωπίστηκε το πρόβλημα με μεταφορά σε έναν πιο ισχυρό server.

Δ. Dedicated Server

Έτσι μετά από αναζήτηση ο dedicated server, ένας ολόκληρος server αφιερωμένος(dedicated) στον ιστοχώρο, ήταν η λύση Η μόνη αναβάθμιση που χρειάστηκε να γίνει ήταν η αύξηση της RAM από 1GB σε 2GB όταν η επισκεψιμότητα του ιστοχώρου είχε φτάσει στα ανώτερα επίπεδα, και αυτό για προληπτικούς λόγους. Αξίζει να σημειωθεί πως η διαθέσιμη RAM στον semi-dedicated server ήταν 256MB.

7. ΠΕΡΙΓΡΑΦΗ ΒΑΣΗΣ

Ένα ακόμα πολύ σημαντικό θέμα το οποίο έπρεπε να αντιμετωπιστεί ήταν η επιλογή βάσης δεδομένων. Αναζητώντας πληροφορίες για την επιλογή βάσης διαπιστώθηκε πως υπήρχαν πολλά περισσότερα βοηθήματα για την MySQL. Έτσι ξεκίνησε η ανάπτυξη του ιστοχώρου με την χρήση αυτής.

Αρκετό καιρό μετά, και πάρθηκε η απόφαση να μετατραπεί τον ιστοχώρο από MySQL σε PostreSQL αφού αυτή είχε κάποια χαρακτηριστικά-δυνατότητες παραπάνω. Η διαδικασία ήταν αρκετά χρονοβόρα αφού δεν υπήρχαν αρκετοί διαθέσιμοι οδηγοί για postgeSQL στο διαδίκτυο σε σχέση με την MySQL.

Όταν τελικά κατασκευάστηκε ο ιστοχώρος προέκυψε το εξής πρόβλημα. Έπρεπε ο server να υποστηρίζει και αυτός PostgreSQL. Στην αρχή θεωρήθηκε λανθασμένα ότι όλοι οι έτοιμοι servers είχαν αυτήν την δυνατότητα. Όμως είτε το κόστος ανέβαινε είτε υπήρχαν κάποιες δυσλειτουργίες. Παρόλα αυτά στήθηκε ένας server όπου να υποστηρίζονται όλα.

Αρκετό καιρό όμως μετά και ενώ βρισκόταν σε semi-dedicated server, παρατηρήθηκε πως η βάση κατανάλωνε αρκετούς πόρους. Έτσι εγκαταλείφτηκαν οι πρόσθετες δυνατότητες και η επιστροφή σε μια περιορισμένων δυνατοτήτων γλώσσα, αλλά αρκετά πιο γρήγορη, ήταν μονόδρομος.

Το συμπέρασμα ήταν πως, αν διατηρούνταν η PostgreSQL σε ένα γρηγορότερο μηχάνημα δεν θα υπήρχαν προβλήματα, και μάλιστα θα μπορούσε να απλουστευτεί ο κώδικα php αφού η βάση θα εξυπηρετούσε πλέον τις ανάγκες μας. Όμως λόγω των περιορισμένων δυνατοτήτων η MySQL τελικά ήταν πιο γρήγορη και πιο οικονομική.

Παρακάτω δίνετε ένας συγκριτικός πίνακας ανάμεσα στην MySQL και στην PostgreSQL.

	POSTGRESQL	MYSQL
ANSI SQL compliance	Closer to ANSI SQL standard	Follows some of the ANSI SQL standards
Performance	Slower	Faster
Sub-selects	Yes	No
Transactions	Yes	Yes, however InnoDB table type must be used
Database replication	Yes	Yes
Foreign key support	Yes	No
Views	Yes	No
Stored procedures	Yes	No
Triggers	Yes	No
Unions	Yes	No
Full joins	Yes	No
Constraints	Yes	No
Windows support	Yes	Yes
Vacuum (cleanup)	Yes	No
ODBC	Yes	Yes
JDBC	Yes	Yes
Different table types	No	Yes

8. ΠΟΤΕ ΝΑ ΧΡΗΣΙΜΟΠΟΙΗΣΟΥΜΕ MYSQL?

Το ερώτημα είναι το εξής: Γιατί να χτησιμοποιηθεί MySQL αντι για PostgreSQL? Πρώτα πρέπει να μελετηθούν οι ανάγκες της εφαρμογής σε απαιτήσεις βάσης δεδομένων. Αν ο σκοπός είναι η δημιουργία μιας εφαρμογής για Web και η απόδοση είναι σοβαρό θέμα, η MySQL είναι μια πολύ καλή επιλογή γιατί είναι γρήγορη και σχεδιάστηκε να δουλεύει εξαιρετικά με Web Servers. Ωστόσο αν θέλει κάποιος να δημιουργήσει μια εφαρμογή που απαιτεί transaction και ξένα κλειδιά η PostgreSQL είναι καλύτερη επιλογή.

Ένας προγραμματιστής που θέλει να χρησιμοποιεί ελεύθερο λογισμικό προγράμματα, καλό είναι να δουλεύει και με τις 2 γλώσσες προγραμματισμού και

ανάλογα με τις ανάγκες (απόδοση/δυνατότητες) να επιλέγει κάθε φορά την κατάλληλη γλώσσα.

Τέλος πρέπει να αναφερθεί πως η MySQL δεν είναι πλήρως συμβατή με τα ANSI SQL standarts σε σχέση με την PostgreSQL που πλησιάζει αρκετά τα πρότυπα αυτά. Η MySQL είναι κοντύτερα στο ODBC standard.

Μερικοί λόγοι για να επιλέξουμε MySQL αντί για PostgreSQL είναι:

- Η MySQL είναι σχετικά πιο γρήγορη από την PostgreSQL.
- Η σχεδίαση της βάσης είναι σχετικά πιο εύκολη
- Μπορεί να δημιουργηθεί ευκολότερα ο βασικό σκελετό ενός Web site.

9. ΠΟΤΕ ΝΑ ΧΡΗΣΙΜΟΠΟΙΗΣΟΥΜΕ POSTGRESQL?

Πολλοί web developers δεν χρησιμοποιούν την PostgreSQL γιατί αισθάνονται πως μειώνει την απόδοση. Όντως υπάρχουν βάσιμοι λόγοι στην υπόθεση αυτοί, όμως η PostgreSQL έχει πολλά πλεονεκτήματα σε σχέση με την MySQL.

Για παράδειγμα, μερικά από τα χαρακτηριστικά είναι foreign key references, triggers, and views. Αυτά τα χαρακτηριστικά επιτρέπουν να μεταφέρουν την πολυπλοκότητα του κώδικα της εφαρμογής στην βάση δεδομένων όπου εδώ λύνονται πολύ πιο εύκολα. Μια από τις πιο αξιοσημείωτες διαφορές ανάμεσα στην MySQL και στην PostgreSQL είναι το γεγονός ότι δεν μπορείς να κάνεις εμφωλευμένα υποερωτήματα σε υποεπιλογές στην MySQL. Επίσης η PostgreSQL επειδή είναι συμβατή με τα ANSI standards επιτρέπει την δημιουργία σύνθετων βάσεων

Μερικοί λόγοι για να επιλέξουμε PostgreSQL αντί για MySQL είναι:

- Σύνθετη βάση δεδομένων
- Χρήση διαδικασιών στην βάση δεδομένων
- Transactions
- Αποθηκευμένες διαδικασίες

- R-Trees (indexes)

10. TEST ENVIRONMENT

Παρακάτω θα επιχειρήσουμε να συγκρίνουμε τις βάσεις δεδομένων MySQL και PostgreSQL. Τα τεστ αυτά εκτελέστηκαν και στις 2 πλατφόρμες στον ίδιο υπολογιστή χωρίς να γίνει κάποια προσπάθεια βελτιστοποίησης. Για τα παρακάτω τεστ χρησιμοποιήθηκε 2.66GHz Intel® CORE™2 Duo με 3.24GB μνήμη και IDE δίσκος. Το λειτουργικό σύστημα που χρησιμοποιήθηκε είναι Windows 7 Home Premium. Επίσης οι εκδόσεις των προγραμμάτων που χρησιμοποιήθηκαν ήταν οι εξής

Apache Version : 2.2.11
PHP Version : 5.2.5
MySQL Version : 5.1.36
PostgreSQL Version : 8.4

Δεν έγινε καμία ιδιαίτερη ρύθμιση για καλύτερες επιδόσεις. (Να επισημανθεί ότι η MySQL δεν υποστηρίζει transactions σ' αυτήν την έκδοση)

Θα δώσουμε συγκριτικά τεστ για INSERTS, SELECTS, UPDATES και DELETE με ή χωρίς index/πολλαπλά index. Ο κώδικας εκτελέστηκε μέσω της PHP και οι χρόνοι είναι η διαφορά timestamp κατά την έναρξη και λήξης του script που είχε ήδη δημιουργηθεί.

Επίσης θα δοθούν συγκριτικοί χρόνοι εκτέλεσης για πραγματικά ερωτήματα της εφαρμογής μας.

Test 1: 20000 INSERTs

PostgreSQL :

```
CREATE TABLE t1(a INTEGER, b INTEGER, c INTEGER, b1 VARCHAR(100), c1 VARCHAR(255));

INSERT INTO t1 VALUES(1,965,752835,'nine hundred and sixty-five','seven hundred and fifty-two thousand eight hundred and thirty-five')
INSERT INTO t1 VALUES(2,709,476144,'seven hundred and nine','four hundred and seventy-six thousand one hundred and fourty-four')
```

INSERT INTO t1 VALUES(3,390,870498,'three hundred and ninety','eight hundred and seventy thousand four hundred and ninety-eight')
INSERT INTO t1 VALUES(4,540,399569,'five hundred and fourty','three hundred and ninety-nine thousand five hundred and sixty-nine')
.....
INSERT INTO t1 VALUES(19997,745,468289,'seven hundred and fourty-five','four hundred and sixty-eight thousand two hundred and eighty-nine')
INSERT INTO t1 VALUES(19998,181,861325,'one hundred and eighty-one','eight hundred and sixty-one thousand three hundred and twenty-five')
INSERT INTO t1 VALUES(19999,862,476968,'eight hundred and sixty-two','four hundred and seventy-six thousand nine hundred and sixty-eight')
INSERT INTO t1 VALUES(20000,733,141390,'seven hundred and thirty-three','one hundred and fourty-one thousand three hundred and ninety')

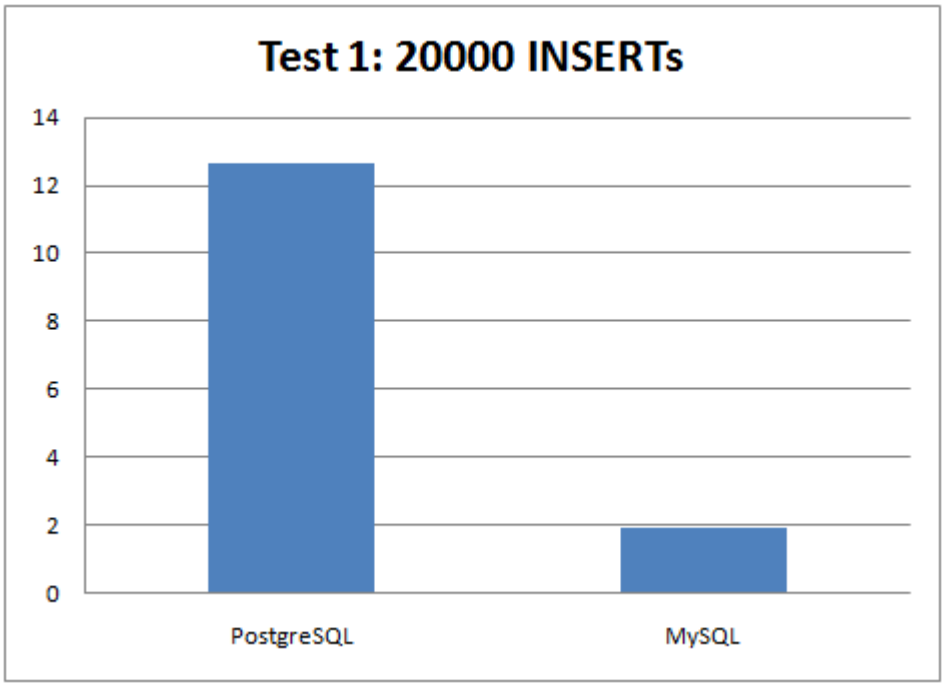
12.64sec

MySQL :

CREATE TABLE t1(a INTEGER, b INTEGER, c INTEGER, b1 VARCHAR(100), c1 VARCHAR(255));

INSERT INTO t1 VALUES(1,745,202639,'seven hundred and fourty-five','two hundred and two thousand six hundred and thirty-nine')
INSERT INTO t1 VALUES(2,137,305636,'one hundred and thirty-seven','three hundred and five thousand six hundred and thirty-six')
INSERT INTO t1 VALUES(3,705,967178,'seven hundred and five','nine hundred and sixty-seven thousand one hundred and seventy-eight')
INSERT INTO t1 VALUES(4,278,650936,'two hundred and seventy-eight','six hundred and fifty thousand nine hundred and thirty-six')
.....
INSERT INTO t1 VALUES(19997,356,424343,'three hundred and fifty-six','four hundred and twenty-four thousand three hundred and fourty-three')
INSERT INTO t1 VALUES(19998,790,747067,'seven hundred and ninety','seven hundred and fourty-seven thousand and sixty-seven')
INSERT INTO t1 VALUES(19999,108,179898,'one hundred and eight','one hundred and seventy-nine thousand eight hundred and ninety-eight')
INSERT INTO t1 VALUES(20000,752,449008,'seven hundred and fifty-two','four hundred and fourty-nine thousand and eight')

1.937sec



Test 2: 20000 INSERTs με index

PostgreSQL :

CREATE TABLE t2(a INTEGER, b INTEGER, c INTEGER, b1 VARCHAR(100), c1 VARCHAR(255));
CREATE INDEX i1 ON t2(b1);

INSERT INTO t2 VALUES(1,976,102444,'nine hundred and seventy-six','one hundred and two thousand four hundred and fourty-four')
INSERT INTO t2 VALUES(2,916,585128,'nine hundred and sixteen','five hundred and eighty-five thousand one hundred and twenty-eight')
INSERT INTO t2 VALUES(3,569,613858,'five hundred and sixty-nine','six hundred and thirteen thousand eight hundred and fifty-eight')
INSERT INTO t2 VALUES(4,466,452304,'four hundred and sixty-six','four hundred and fifty-two thousand three hundred and four')
.....
INSERT INTO t2 VALUES(19997,418,830398,'four hundred and eighteen','eight hundred and thirty thousand three hundred and ninety-eight')
INSERT INTO t2 VALUES(19998,950,182809,'nine hundred and fifty','one hundred and eighty-two thousand eight hundred and nine')
INSERT INTO t2 VALUES(19999,704,332827,'seven hundred and four','three hundred and thirty-two thousand eight hundred and twenty-seven')
INSERT INTO t2 VALUES(20000,321,306625,'three hundred and twenty-one','three hundred and six thousand six hundred and twenty-five')

13.619sec

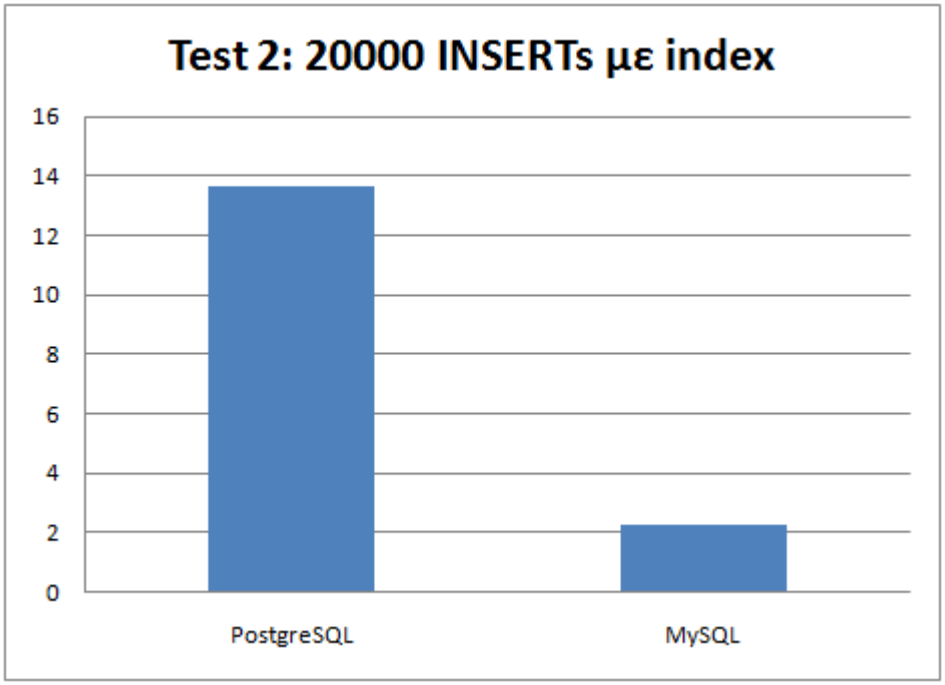
MySQL :

CREATE TABLE t2(a INTEGER, b INTEGER, c INTEGER, b1 VARCHAR(100), c1 VARCHAR(255));
CREATE INDEX i1 ON t2(b1);

INSERT INTO t2 VALUES(1,756,452249,'seven hundred and fifty-six','four hundred and fifty-two thousand two hundred and fourty-nine')

INSERT INTO t2 VALUES(2,345,414620,'three hundred and fourty-five','four hundred and fourteen thousand six hundred and twenty')
INSERT INTO t2 VALUES(3,884,710537,'eight hundred and eighty-four','seven hundred and ten thousand five hundred and thirty-seven')
INSERT INTO t2 VALUES(4,204,703671,'two hundred and four','seven hundred and three thousand six hundred and seventy-one')
.....
INSERT INTO t2 VALUES(19997,929,786453,'nine hundred and twenty-nine','seven hundred and eighty-six thousand four hundred and fifty-three')
INSERT INTO t2 VALUES(19998,660,968551,'six hundred and sixty','nine hundred and sixty-eight thousand five hundred and fifty-one')
INSERT INTO t2 VALUES(19999,850,935757,'eight hundred and fifty','nine hundred and thirty-five thousand seven hundred and fifty-seven')
INSERT INTO t2 VALUES(20000,341,614242,'three hundred and fourty-one','six hundred and fourteen thousand two hundred and fourty-two')

2.285sec



Test 3: 20000 INSERTs με 5 index

PostgreSQL :

CREATE TABLE t2(a INTEGER, b INTEGER, c INTEGER, b1 VARCHAR(100), c1 VARCHAR(255));
CREATE INDEX i1 ON t2(a);
CREATE INDEX i2 ON t2(b);
CREATE INDEX i3 ON t2(c);
CREATE INDEX i4 ON t2(b1);
CREATE INDEX i5 ON t2(c1);

INSERT INTO t2 VALUES(1,986,352053,'nine hundred and eighty-six','three hundred and fifty-two thousand and fifty-three')
INSERT INTO t2 VALUES(2,223,694113,'two hundred and twenty-three','six hundred and ninety-four thousand one hundred and thirteen')
INSERT INTO t2 VALUES(3,748,357217,'seven hundred and fourty-eight','three hundred and fifty-seven thousand two hundred and seventeen')
INSERT INTO t2 VALUES(4,392,505038,'three hundred and ninety-two','five hundred and five thousand and thirty-eight')
.....
INSERT INTO t2 VALUES(19997,991,292507,'nine hundred and ninety-one','two hundred and ninety-two thousand five hundred and seven')
INSERT INTO t2 VALUES(19998,820,404293,'eight hundred and twenty','four hundred and four thousand two hundred and ninety-three')
INSERT INTO t2 VALUES(19999,546,188687,'five hundred and fourty-six','one hundred and eighty-eight thousand six hundred and eighty-seven')
INSERT INTO t2 VALUES(20000,810,471859,'eight hundred and ten','four hundred and seventy-one thousand eight hundred and fifty-nine')

35.207sec

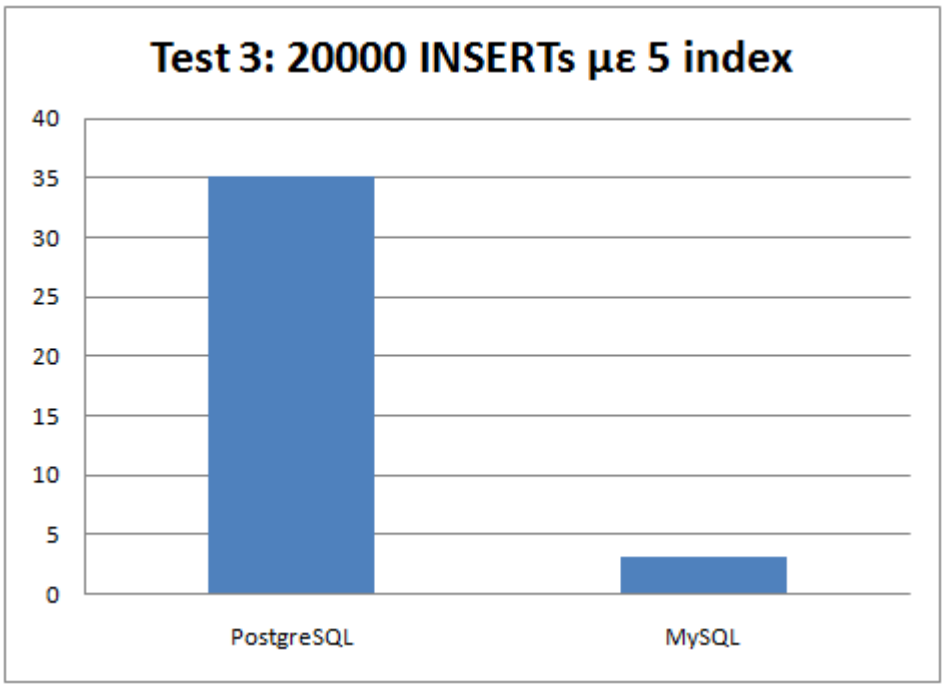
MySQL :

CREATE TABLE t2(a INTEGER, b INTEGER, c INTEGER, b1 VARCHAR(100), c1 VARCHAR(255));
CREATE INDEX i1 ON t2(a);
CREATE INDEX i2 ON t2(b);
CREATE INDEX i3 ON t2(c);
CREATE INDEX i4 ON t2(b1);
CREATE INDEX i5 ON t2(c1);

INSERT INTO t2 VALUES(1,986,352053,'nine hundred and eighty-six','three hundred and fifty-two thousand and fifty-three')
INSERT INTO t2 VALUES(1,767,701858,'seven hundred and sixty-seven','seven hundred and one thousand eight hundred and fifty-eight')
INSERT INTO t2 VALUES(2,552,523605,'five hundred and fifty-two','five hundred and twenty-three thousand six hundred and five')
INSERT INTO t2 VALUES(3,163,453897,'one hundred and sixty-three','four hundred and fifty-three thousand eight hundred and ninety-seven')
INSERT INTO t2 VALUES(4,131,756405,'one hundred and thirty-one','seven hundred and fifty-six thousand four hundred and five')
.....
INSERT INTO t2 VALUES(19997,602,248562,'six hundred and two','two hundred and fourty-eight thousand five hundred and sixty-two')
INSERT INTO t2 VALUES(19998,530,290036,'five hundred and thirty','two hundred and ninety thousand and thirty-six')
INSERT INTO t2 VALUES(19999,692,791616,'six hundred and ninety-two','seven hundred and ninety-one thousand six hundred and sixteen')

INSERT INTO t2 VALUES(20000,829,779476,'eight hundred and twenty-nine','seven hundred and seventy-nine thousand four hundred and seventy-six')

3.222sec



Test 4: 20000 INSERTs χωρίς index & 200 SELECTs χωρίς index

PostgreSQL INSERTS :

CREATE TABLE t1(a INTEGER, b INTEGER, c INTEGER, b1 VARCHAR(100), c1 VARCHAR(255));

INSERT INTO t1 VALUES(1,118,200881,'one hundred and eighteen','two hundred thousand eight hundred and eighty-one')
INSERT INTO t1 VALUES(2,846,121066,'eight hundred and fourty-six','one hundred and twenty-one thousand and sixty-six')
INSERT INTO t1 VALUES(3,386,487295,'three hundred and eighty-six','four hundred and eighty-seven thousand two hundred and ninety-five')
INSERT INTO t1 VALUES(4,171,663241,'one hundred and seventy-one','six hundred and sixty-three thousand two hundred and fourty-one')
.....
INSERT INTO t1 VALUES(19997,910,478836,'nine hundred and ten','four hundred and seventy-eight thousand eight hundred and thirty-six')
INSERT INTO t1 VALUES(19998,430,168746,'four hundred and thirty','one hundred and sixty-eight thousand seven hundred and fourty-six')
INSERT INTO t1 VALUES(19999,971,656265,'nine hundred and seventy-one','six hundred and fifty-six thousand two hundred and sixty-five')
INSERT INTO t1 VALUES(20000,476,967562,'four hundred and seventy-six','nine hundred and sixty-seven thousand five hundred and sixty-two')

12.135sec

PostgreSQL SELECTS :

SELECT count(*), avg(b) FROM t1 WHERE b>=0 AND b<1000
SELECT count(*), avg(b) FROM t1 WHERE b>=100 AND b<1100
SELECT count(*), avg(b) FROM t1 WHERE b>=200 AND b<1200
SELECT count(*), avg(b) FROM t1 WHERE b>=300 AND b<1300
.....
SELECT count(*), avg(b) FROM t1 WHERE b>=19600 AND b<20600
SELECT count(*), avg(b) FROM t1 WHERE b>=19700 AND b<20700
SELECT count(*), avg(b) FROM t1 WHERE b>=19800 AND b<20800
SELECT count(*), avg(b) FROM t1 WHERE b>=19900 AND b<20900

12.725sec

MySQL INSERTS :

CREATE TABLE t1(a INTEGER, b INTEGER, c INTEGER, b1 VARCHAR(100), c1 VARCHAR(255));

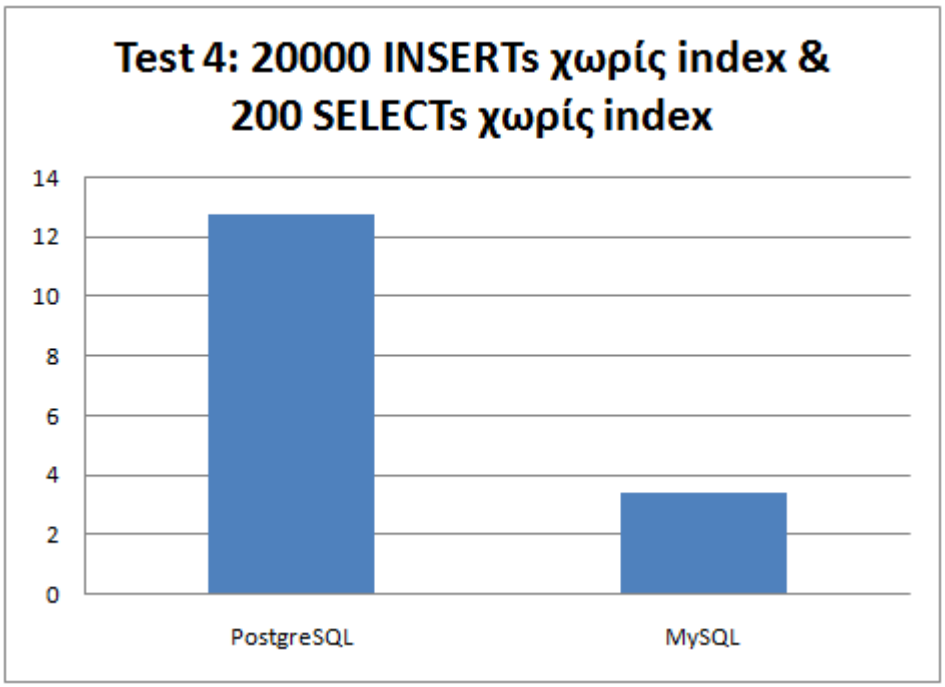
INSERT INTO t1 VALUES(1,798,550686,'seven hundred and ninety-eight','five hundred and fifty thousand six hundred and eighty-six')
INSERT INTO t1 VALUES(2,274,850558,'two hundred and seventy-four','eight hundred and fifty thousand five hundred and fifty-eight')
INSERT INTO t1 VALUES(3,701,583975,'seven hundred and one','five hundred and eighty-three thousand nine hundred and seventy-five')
INSERT INTO t1 VALUES(4,809,914608,'eight hundred and nine','nine hundred and fourteen thousand six hundred and eight')
.....
INSERT INTO t1 VALUES(19997,522,434890,'five hundred and twenty-two','four hundred and thirty-four thousand eight hundred and ninety')
INSERT INTO t1 VALUES(19998,140,954489,'one hundred and fourty','nine hundred and fifty-four thousand four hundred and eighty-nine')
INSERT INTO t1 VALUES(19999,217,359194,'two hundred and seventeen','three hundred and fifty-nine thousand one hundred and ninety-four')
INSERT INTO t1 VALUES(20000,496,375180,'four hundred and ninety-six','three hundred and seventy-five thousand one hundred and eighty')

1.936sec

MySQL SELECTS :

```
SELECT count(*), avg(b) FROM t1 WHERE b>=0 AND b<1000
SELECT count(*), avg(b) FROM t1 WHERE b>=100 AND b<1100
SELECT count(*), avg(b) FROM t1 WHERE b>=200 AND b<1200
SELECT count(*), avg(b) FROM t1 WHERE b>=300 AND b<1300
.....
SELECT count(*), avg(b) FROM t1 WHERE b>=19600 AND b<20600
SELECT count(*), avg(b) FROM t1 WHERE b>=19700 AND b<20700
SELECT count(*), avg(b) FROM t1 WHERE b>=19800 AND b<20800
SELECT count(*), avg(b) FROM t1 WHERE b>=19900 AND b<20900
```

3.409sec



Test 5: 20000 INSERTs με string index & 200 SELECTs με search string index

PostgreSQL INSERTS :

```
CREATE TABLE t1(a INTEGER, b INTEGER, c INTEGER, b1 VARCHAR(100), c1 VARCHAR(255));
CREATE INDEX i1 ON t1(b1);
CREATE INDEX i2 ON t1(c1);

INSERT INTO t1 VALUES(1,129,450491,'one hundred and twenty-nine','four hundred and fifty thousand four hundred and ninety-one')
INSERT INTO t1 VALUES(2,153,230050,'one hundred and fifty-three','two hundred and thirty thousand and fifty')
INSERT INTO t1 VALUES(3,566,230654,'five hundred and sixty-six','two hundred and thirty thousand six hundred and fifty-four')
INSERT INTO t1 VALUES(4,997,715975,'nine hundred and ninety-seven','seven hundred and fifteen thousand nine hundred and seventy-five')
.....
INSERT INTO t1 VALUES(19997,583,840945,'five hundred and eighty-three','eight hundred and fourty thousand nine hundred and fourty-five')
INSERT INTO t1 VALUES(19998,300,390231,'three hundred','three hundred and ninety thousand two hundred and thirty-one')
INSERT INTO t1 VALUES(19999,813,512124,'eight hundred and thirteen','five hundred and twelve thousand one hundred and twenty-four')
INSERT INTO t1 VALUES(20000,965,232797,'nine hundred and sixty-five','two hundred and thirty-two thousand seven hundred and ninety-seven')
```

14.254sec

PostgreSQL SELECTS :

```
SELECT count(*), avg(b) FROM t1 WHERE c1 LIKE '%one hundred and one%';
SELECT count(*), avg(b) FROM t1 WHERE c1 LIKE '%one hundred and two%';
SELECT count(*), avg(b) FROM t1 WHERE c1 LIKE '%one hundred and three%';
SELECT count(*), avg(b) FROM t1 WHERE c1 LIKE '%one hundred and four%';
.....
SELECT count(*), avg(b) FROM t1 WHERE c1 LIKE '%two hundred and ninety-seven%';
SELECT count(*), avg(b) FROM t1 WHERE c1 LIKE '%two hundred and ninety-eight%';
SELECT count(*), avg(b) FROM t1 WHERE c1 LIKE '%two hundred and ninety-nine%';
SELECT count(*), avg(b) FROM t1 WHERE c1 LIKE '%three hundred%';
```

15.024sec

MySQL INSERTS :

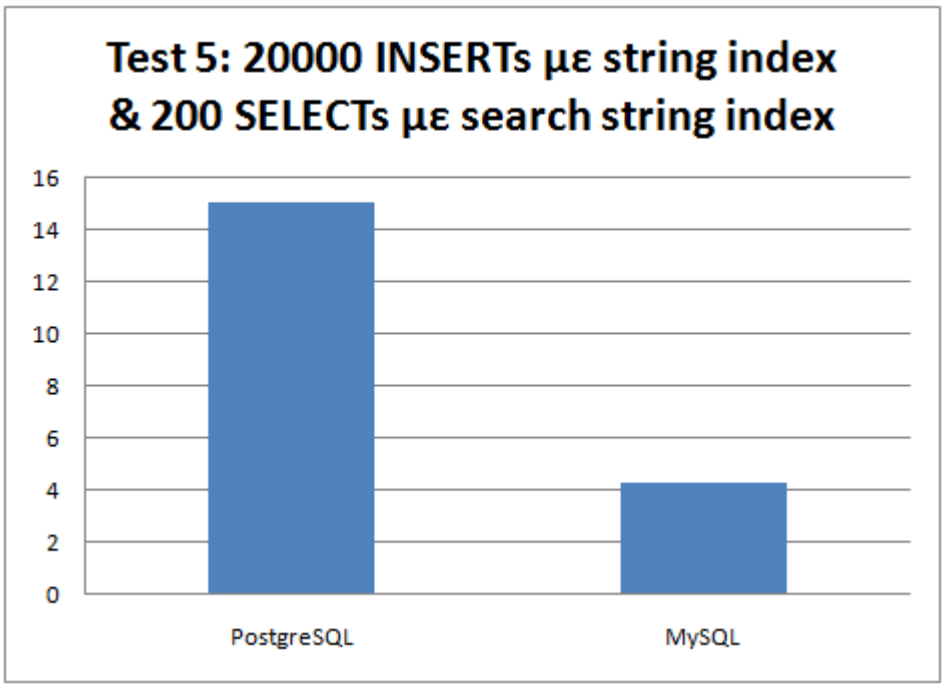
```
CREATE TABLE t1(a INTEGER, b INTEGER, c INTEGER, b1 VARCHAR(100), c1 VARCHAR(255));
CREATE INDEX i1 ON t1(b1);
CREATE INDEX i2 ON t1(c1);

INSERT INTO t1 VALUES(1,788,800296,'seven hundred and eighty-eight','eight hundred thousand two hundred and ninety-six')
INSERT INTO t1 VALUES(2,424,959542,'four hundred and twenty-four','nine hundred and fifty-nine thousand five hundred and fourty-two')
INSERT INTO t1 VALUES(3,867,327334,'eight hundred and sixty-seven','three hundred and twenty-seven thousand three hundred and thirty-four')
INSERT INTO t1 VALUES(4,706,967343,'seven hundred and six','nine hundred and sixty-seven thousand three hundred and fourty-three')
.....
INSERT INTO t1 VALUES(19997,105,797000,'one hundred and five','seven hundred and ninety-seven thousand')
INSERT INTO t1 VALUES(19998,900,275973,'nine hundred','two hundred and seventy-five thousand nine hundred and seventy-three')
INSERT INTO t1 VALUES(19999,954,215054,'nine hundred and fifty-four','two hundred and fifteen thousand and fifty-four')
INSERT INTO t1 VALUES(20000,982,540414,'nine hundred and eighty-two','five hundred and fourty thousand four hundred and fourteen')
```

2.507sec

MySQL SELECTS :

```
ELECT count(*), avg(b) FROM t1 WHERE c1 LIKE '%one hundred and one%';
SELECT count(*), avg(b) FROM t1 WHERE c1 LIKE '%one hundred and two%';
SELECT count(*), avg(b) FROM t1 WHERE c1 LIKE '%one hundred and three%';
SELECT count(*), avg(b) FROM t1 WHERE c1 LIKE '%one hundred and four%';
.....
SELECT count(*), avg(b) FROM t1 WHERE c1 LIKE '%two hundred and ninety-seven%';
SELECT count(*), avg(b) FROM t1 WHERE c1 LIKE '%two hundred and ninety-eight%';
SELECT count(*), avg(b) FROM t1 WHERE c1 LIKE '%two hundred and ninety-nine%';
SELECT count(*), avg(b) FROM t1 WHERE c1 LIKE '%three hundred%';
```

4.266sec



Test 6: 20000 INSERTs χωρίς index & δημιουργεία 2 index (string)

PostgreSQL INSERTS :

```
CREATE TABLE t1(a INTEGER, b INTEGER, c INTEGER, b1 VARCHAR(100), c1 VARCHAR(255));
INSERT INTO t1 VALUES(1,55,949710,'','nine hundred and fourty-nine thousand seven hundred and ten')
INSERT INTO t1 VALUES(2,520,448019,'five hundred and twenty','four hundred and fourty-eight thousand and nineteen')
INSERT INTO t1 VALUES(3,916,617373,'nine hundred and sixteen','six hundred and seventeen thousand three hundred and seventy-three')
INSERT INTO t1 VALUES(4,833,821444,'eight hundred and thirty-three','eight hundred and twenty-one thousand four hundred and fourty-four')
.....
INSERT INTO t1 VALUES(19997,810,665164,'eight hundred and ten','six hundred and sixty-five thousand one hundred and sixty-four')
INSERT INTO t1 VALUES(19998,933,833200,'nine hundred and thirty-three','eight hundred and thirty-three thousand two hundred')
INSERT INTO t1 VALUES(19999,441,223843,'four hundred and fourty-one','two hundred and twenty-three thousand eight hundred and fourty-three')
INSERT INTO t1 VALUES(20000,47,563265,'','five hundred and sixty-three thousand two hundred and sixty-five')
```

12.142sec

PostgreSQL CREATE INDEXs :

CREATE INDEX i1 ON t1(b1);
CREATE INDEX i2 ON t1(c1);

0.59sec

MySQL INSERTS :

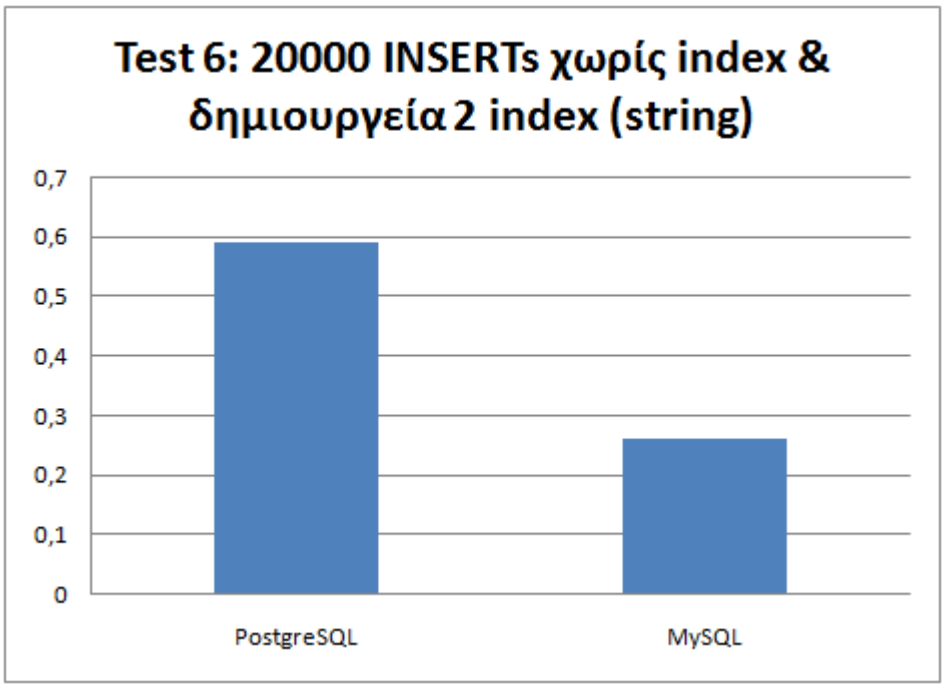
CREATE TABLE t1(a INTEGER, b INTEGER, c INTEGER, b1 VARCHAR(100), c1 VARCHAR(255));
INSERT INTO t1 VALUES(1,830,399514,'eight hundred and thirty','three hundred and ninety-nine thousand five hundred and fourteen')
INSERT INTO t1 VALUES(2,897,277511,'eight hundred and ninety-seven','two hundred and seventy-seven thousand five hundred and eleven')
INSERT INTO t1 VALUES(3,339,714053,'three hundred and thirty-nine','seven hundred and fourteen thousand and fifty-three')
INSERT INTO t1 VALUES(4,588,172811,'five hundred and eighty-eight','one hundred and seventy-two thousand eight hundred and eleven')
.....
INSERT INTO t1 VALUES(19997,441,621218,'four hundred and fourty-one','six hundred and twenty-one thousand two hundred and eighteen')
INSERT INTO t1 VALUES(19998,650,718942,'six hundred and fifty','seven hundred and eighteen thousand nine hundred and fourty-two')
INSERT INTO t1 VALUES(19999,642,826773,'six hundred and fourty-two','eight hundred and twenty-six thousand seven hundred and seventy-three')
INSERT INTO t1 VALUES(20000,162,870883,'one hundred and sixty-two','eight hundred and seventy thousand eight hundred and eighty-three')

1.906sec

MySQL CREATE INDEXs :

CREATE INDEX i1 ON t1(b1);
CREATE INDEX i2 ON t1(c1);

0.262sec



Test 7: 20000 INSERTs με 5 index & 200 SELECTs με index (integer)

PostgreSQL INSERTS :

CREATE TABLE t1(a INTEGER, b INTEGER, c INTEGER, b1 VARCHAR(100), c1 VARCHAR(255));
CREATE INDEX i1 ON t1(a);
CREATE INDEX i2 ON t1(b);
CREATE INDEX i3 ON t1(c);
CREATE INDEX i4 ON t1(b1);
CREATE INDEX i5 ON t1(c1);
INSERT INTO t1 VALUES(1,118,377981,'one hundred and eighteen','three hundred and seventy-seven thousand nine hundred and eighty-one')
INSERT INTO t1 VALUES(2,574,491525,'five hundred and seventy-four','four hundred and ninety-one thousand five hundred and twenty-five')
INSERT INTO t1 VALUES(3,491,460488,'four hundred and ninety-one','four hundred and sixty thousand four hundred and eighty-eight')
INSERT INTO t1 VALUES(4,174,323544,'one hundred and seventy-four','three hundred and twenty-three thousand five hundred and fourty-four')
.....
INSERT INTO t1 VALUES(19997,249,613748,'two hundred and fourty-nine','six hundred and thirteen thousand seven hundred and fourty-eight')
INSERT INTO t1 VALUES(19998,961,609518,'nine hundred and sixty-one','six hundred and nine thousand five hundred and eighteen')
INSERT INTO t1 VALUES(19999,640,812271,'six hundred and fourty','eight hundred and twelve thousand two hundred and seventy-one')
INSERT INTO t1 VALUES(20000,606,923178,'six hundred and six','nine hundred and twenty-three thousand one hundred and seventy-eight')

14.878sec

PostgreSQL SELECTS με index :

SELECT count(*), avg(b) FROM t1 WHERE b>=0 AND b<100
SELECT count(*), avg(b) FROM t1 WHERE b>=100 AND b<200
SELECT count(*), avg(b) FROM t1 WHERE b>=200 AND b<300
SELECT count(*), avg(b) FROM t1 WHERE b>=300 AND b<400
.....
SELECT count(*), avg(b) FROM t1 WHERE b>=19600 AND b<19700
SELECT count(*), avg(b) FROM t1 WHERE b>=19700 AND b<19800
SELECT count(*), avg(b) FROM t1 WHERE b>=19800 AND b<19900
SELECT count(*), avg(b) FROM t1 WHERE b>=19900 AND b<20000

15.997sec

MySQL INSERTS :

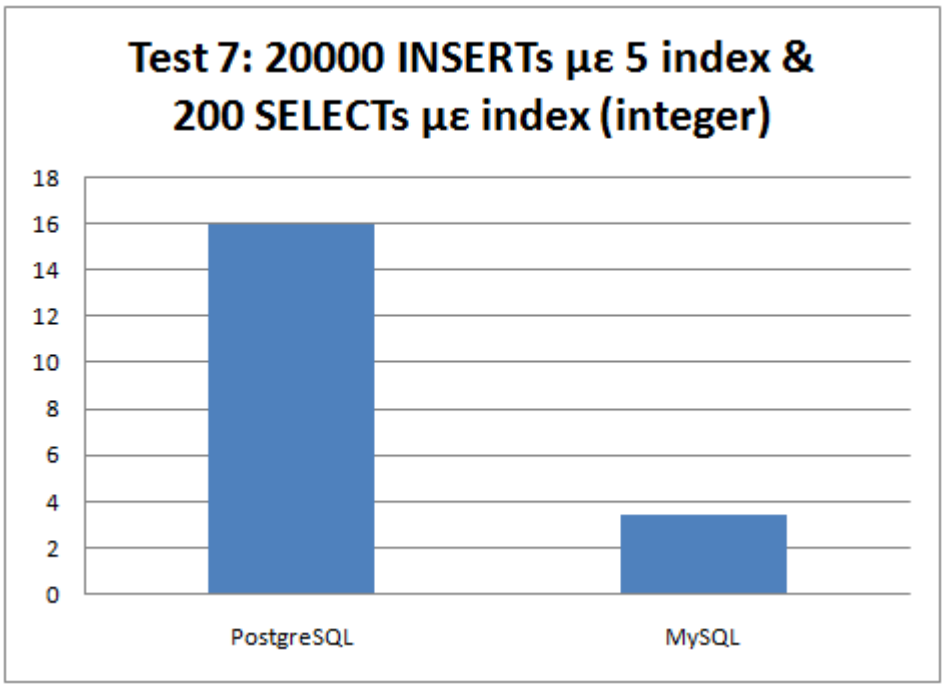
CREATE TABLE t1(a INTEGER, b INTEGER, c INTEGER, b1 VARCHAR(100), c1 VARCHAR(255));
CREATE INDEX i1 ON t1(a);
CREATE INDEX i2 ON t1(b);
CREATE INDEX i3 ON t1(c);
CREATE INDEX i4 ON t1(b1);
CREATE INDEX i5 ON t1(c1);
INSERT INTO t1 VALUES(1,685,165286,'six hundred and eighty-five','one hundred and sixty-five thousand two hundred and eighty-six')
INSERT INTO t1 VALUES(2,791,658517,'seven hundred and ninety-one','six hundred and fifty-eight thousand five hundred and seventeen')
INSERT INTO t1 VALUES(3,694,894668,'six hundred and ninety-four','eight hundred and ninety-four thousand six hundred and sixty-eight')
INSERT INTO t1 VALUES(4,700,912411,'seven hundred','nine hundred and twelve thousand four hundred and eleven')
.....
INSERT INTO t1 VALUES(19997,648,907302,'six hundred and fourty-eight','nine hundred and seven thousand three hundred and two')
INSERT INTO t1 VALUES(19998,558,832760,'five hundred and fifty-eight','eight hundred and thirty-two thousand seven hundred and sixty')
INSERT INTO t1 VALUES(19999,674,852700,'six hundred and seventy-four','eight hundred and fifty-two thousand seven hundred')
INSERT INTO t1 VALUES(20000,513,668295,'five hundred and thirteen','six hundred and sixty-eight thousand two hundred and ninety-five')

3.258sec

MySQL SELECTS με index :

SELECT count(*), avg(b) FROM t1 WHERE b>=0 AND b<100
SELECT count(*), avg(b) FROM t1 WHERE b>=100 AND b<200
SELECT count(*), avg(b) FROM t1 WHERE b>=200 AND b<300
SELECT count(*), avg(b) FROM t1 WHERE b>=300 AND b<400
.....
SELECT count(*), avg(b) FROM t1 WHERE b>=19600 AND b<19700
SELECT count(*), avg(b) FROM t1 WHERE b>=19700 AND b<19800
SELECT count(*), avg(b) FROM t1 WHERE b>=19800 AND b<19900
SELECT count(*), avg(b) FROM t1 WHERE b>=19900 AND b<20000

3.431sec



Test 8: 20000 INSERTs χωρις index & 20000 UPDATES χωρις index

PostgreSQL INSERTS :

CREATE TABLE t1(a INTEGER, b INTEGER, c INTEGER, b1 VARCHAR(100), c1 VARCHAR(255));
INSERT INTO t1 VALUES(1,619,490151,'six hundred and nineteen','four hundred and ninety thousand one hundred and fifty-one')
INSERT INTO t1 VALUES(2,820,175668,'eight hundred and twenty','one hundred and seventy-five thousand six hundred and sixty-eight')
INSERT INTO t1 VALUES(3,432,651760,'four hundred and thirty-two','six hundred and fifty-one thousand seven hundred and sixty')
INSERT INTO t1 VALUES(4,941,213351,'nine hundred and fourty-one','two hundred and thirteen thousand three hundred and fifty-one')
.....
INSERT INTO t1 VALUES(19997,859,905215,'eight hundred and fifty-nine','nine hundred and five thousand two hundred and fifteen')
INSERT INTO t1 VALUES(19998,949,219833,'nine hundred and fourty-nine','two hundred and nineteen thousand eight hundred and thirty-three')
INSERT INTO t1 VALUES(19999,633,451589,'six hundred and thirty-three','four hundred and fifty-one thousand five hundred and eighty-nine')
INSERT INTO t1 VALUES(20000,610,457907,'six hundred and ten','four hundred and fifty-seven thousand nine hundred and seven')

12.254sec

PostgreSQL UPDATES χωρις index :

UPDATE t1 SET b=b*2 WHERE a>=0 AND a<10;
UPDATE t1 SET b=b*2 WHERE a>=10 AND a<20;
UPDATE t1 SET b=b*2 WHERE a>=20 AND a<30;
UPDATE t1 SET b=b*2 WHERE a>=30 AND a<40;
.....
UPDATE t1 SET b=b*2 WHERE a>=19960 AND a<19970;
UPDATE t1 SET b=b*2 WHERE a>=19970 AND a<19980;
UPDATE t1 SET b=b*2 WHERE a>=19980 AND a<19990;
UPDATE t1 SET b=b*2 WHERE a>=19990 AND a<20000;

19.815sec

MySQL INSERTS :

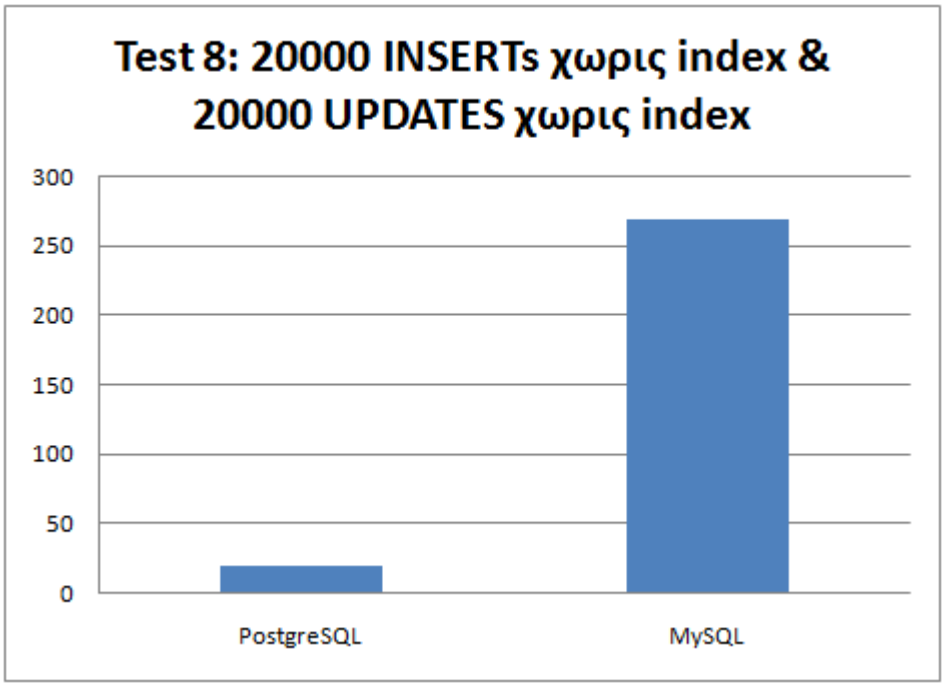
CREATE TABLE t1(a INTEGER, b INTEGER, c INTEGER, b1 VARCHAR(100), c1 VARCHAR(255));
INSERT INTO t1 VALUES(1,540,868081,'five hundred and fourty','eight hundred and sixty-eight thousand and eighty-one')
INSERT INTO t1 VALUES(2,614,258285,'six hundred and fourteen','two hundred and fifty-eight thousand two hundred and eighty-five')
INSERT INTO t1 VALUES(3,437,326565,'four hundred and thirty-seven','three hundred and twenty-six thousand five hundred and sixty-five')
INSERT INTO t1 VALUES(4,595,267843,'five hundred and ninety-five','two hundred and sixty-seven thousand eight hundred and fourty-three')
.....
INSERT INTO t1 VALUES(19997,611,889395,'six hundred and eleven','eight hundred and eighty-nine thousand three hundred and ninety-five')
INSERT INTO t1 VALUES(19998,125,358700,'one hundred and twenty-five','three hundred and fifty-eight thousand seven hundred')
INSERT INTO t1 VALUES(19999,470,632644,'four hundred and seventy','six hundred and thirty-two thousand six hundred and fourty-four')
INSERT INTO t1 VALUES(20000,545,568649,'five hundred and fourty-five','five hundred and sixty-eight thousand six hundred and fourty-nine')

1.891sec

MySQL UPDATES χωρις index:

UPDATE t1 SET b=b*2 WHERE a>=0 AND a<10;
UPDATE t1 SET b=b*2 WHERE a>=10 AND a<20;
UPDATE t1 SET b=b*2 WHERE a>=20 AND a<30;
UPDATE t1 SET b=b*2 WHERE a>=30 AND a<40;
.....
UPDATE t1 SET b=b*2 WHERE a>=19960 AND a<19970;
UPDATE t1 SET b=b*2 WHERE a>=19970 AND a<19980;
UPDATE t1 SET b=b*2 WHERE a>=19980 AND a<19990;
UPDATE t1 SET b=b*2 WHERE a>=19990 AND a<20000;

268.483sec



Test 9: 20000 INSERTs με 5 index & 20000 UPDATES με index (integer)

PostgreSQL INSERTS :

```
CREATE TABLE t1(a INTEGER, b INTEGER, c INTEGER, b1 VARCHAR(100), c1 VARCHAR(255));
CREATE INDEX i1 ON t1(a);
CREATE INDEX i2 ON t1(b);
CREATE INDEX i3 ON t1(c);
CREATE INDEX i4 ON t1(b1);
CREATE INDEX i5 ON t1(c1);
INSERT INTO t1 VALUES(1,910,796011,'nine hundred and ten','seven hundred and ninety-six thousand and eleven')
INSERT INTO t1 VALUES(2,858,790902,'eight hundred and fifty-eight','seven hundred and ninety thousand nine hundred and two')
INSERT INTO t1 VALUES(3,892,451370,'eight hundred and ninety-two','four hundred and fifty-one thousand three hundred and seventy')
INSERT INTO t1 VALUES(4,699,772335,'six hundred and ninety-nine','seven hundred and seventy-two thousand three hundred and thirty-five')
.....
INSERT INTO t1 VALUES(19997,813,423574,'eight hundred and thirteen','four hundred and twenty-three thousand five hundred and seventy-four')
INSERT INTO t1 VALUES(19998,650,947567,'six hundred and fifty','nine hundred and fourty-seven thousand five hundred and sixty-seven')
INSERT INTO t1 VALUES(19999,756,363699,'seven hundred and fifty-six','three hundred and sixty-three thousand six hundred and ninety-nine')
INSERT INTO t1 VALUES(20000,930,229391,'nine hundred and thirty','two hundred and twenty-nine thousand three hundred and ninety-one')
```

14.722sec

PostgreSQL UPDATES χωρις index :

```
UPDATE t1 SET b=30269 WHERE a=1;
UPDATE t1 SET b=20187 WHERE a=2;
UPDATE t1 SET b=89766 WHERE a=3;
UPDATE t1 SET b=50613 WHERE a=4;
.....
UPDATE t1 SET b=56779 WHERE a=19997;
UPDATE t1 SET b=91653 WHERE a=19998;
UPDATE t1 SET b=43942 WHERE a=19999;
UPDATE t1 SET b=22675 WHERE a=20000;
```

15.75sec

MySQL INSERTS :

```
CREATE TABLE t1(a INTEGER, b INTEGER, c INTEGER, b1 VARCHAR(100), c1 VARCHAR(255));
CREATE INDEX i1 ON t1(a);
CREATE INDEX i2 ON t1(b);
CREATE INDEX i3 ON t1(c);
CREATE INDEX i4 ON t1(b1);
CREATE INDEX i5 ON t1(c1);
INSERT INTO t1 VALUES(1,831,273941,'eight hundred and thirty-one','two hundred and seventy-three thousand nine hundred and fourty-one')
INSERT INTO t1 VALUES(2,653,873519,'six hundred and fifty-three','eight hundred and seventy-three thousand five hundred and nineteen')
INSERT INTO t1 VALUES(3,898,126174,'eight hundred and ninety-eight','one hundred and twenty-six thousand one hundred and seventy-four')
INSERT INTO t1 VALUES(4,353,826828,'three hundred and fifty-three','eight hundred and twenty-six thousand eight hundred and twenty-eight')
```

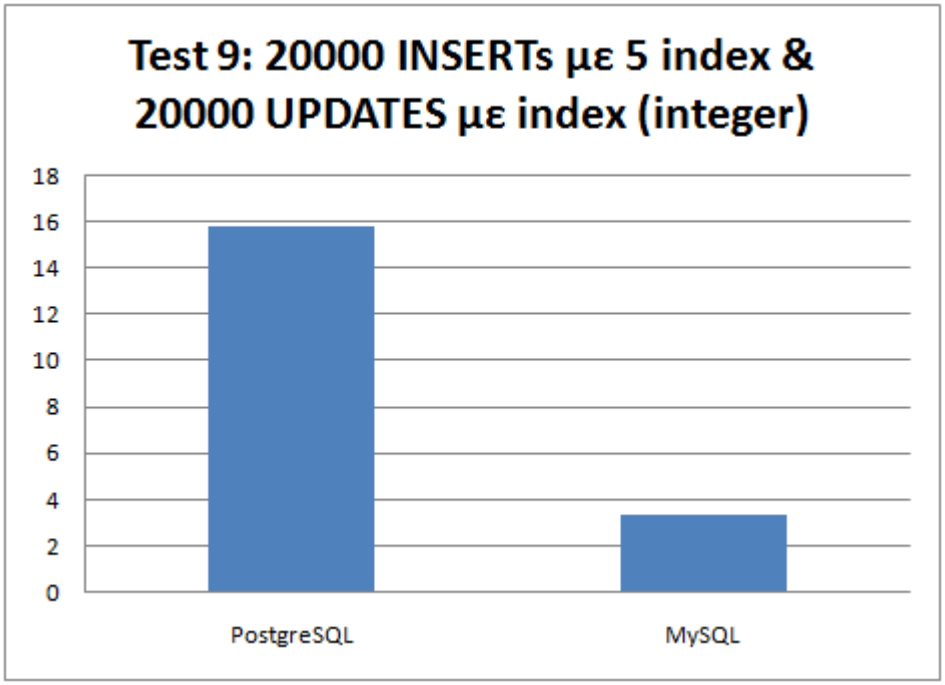
.....
INSERT INTO t1 VALUES(19997,565,407754,'five hundred and sixty-five','four hundred and seven thousand seven hundred and fifty-four')
INSERT INTO t1 VALUES(19998,726,186434,'seven hundred and twenty-six','one hundred and eighty-six thousand four hundred and thirty-four')
INSERT INTO t1 VALUES(19999,593,544754,'five hundred and ninety-three','five hundred and fourty-four thousand seven hundred and fifty-four')
INSERT INTO t1 VALUES(20000,865,340133,'eight hundred and sixty-five','three hundred and fourty thousand one hundred and thirty-three')

3.171sec

MySQL UPDATES χωρις index:

UPDATE t1 SET b=99439 WHERE a=1;
UPDATE t1 SET b=44708 WHERE a=2;
UPDATE t1 SET b=17608 WHERE a=3;
UPDATE t1 SET b=60369 WHERE a=4;
.....
UPDATE t1 SET b=87980 WHERE a=19997;
UPDATE t1 SET b=27580 WHERE a=19998;
UPDATE t1 SET b=35065 WHERE a=19999;
UPDATE t1 SET b=90087 WHERE a=20000;

3.386sec



Test 10: 20000 INSERTs με 5 index & 20000 UPDATES με index (string)

PostgreSQL INSERTS :

CREATE TABLE t1(a INTEGER, b INTEGER, c INTEGER, b1 VARCHAR(100), c1 VARCHAR(255));
CREATE INDEX i1 ON t1(a);
CREATE INDEX i2 ON t1(b);
CREATE INDEX i3 ON t1(c);
CREATE INDEX i4 ON t1(b1);
CREATE INDEX i5 ON t1(c1);
INSERT INTO t1 VALUES(1,223,579800,'two hundred and twenty-three','five hundred and seventy-nine thousand eight hundred')
INSERT INTO t1 VALUES(2,691,588754,'six hundred and ninety-one','five hundred and eighty-eight thousand seven hundred and fifty-four')
INSERT INTO t1 VALUES(3,458,825784,'four hundred and fifty-eight','eight hundred and twenty-five thousand seven hundred and eighty-four')
INSERT INTO t1 VALUES(4,110,485812,'one hundred and ten','four hundred and eighty-five thousand eight hundred and twelve')
.....
INSERT INTO t1 VALUES(19997,520,826113,'five hundred and twenty','eight hundred and twenty-six thousand one hundred and thirteen')
INSERT INTO t1 VALUES(19998,427,914169,'four hundred and twenty-seven','nine hundred and fourteen thousand one hundred and sixty-nine')
INSERT INTO t1 VALUES(19999,716,456863,'seven hundred and sixteen','four hundred and fifty-six thousand eight hundred and sixty-three')
INSERT INTO t1 VALUES(20000,285,111618,'two hundred and eighty-five','one hundred and eleven thousand six hundred and eighteen')

14.75sec

PostgreSQL UPDATES με index :

UPDATE t1 SET b1='five hundred and thirty-nine' WHERE a=1;

UPDATE t1 SET b1='five hundred and sixty-five' WHERE a=2;
UPDATE t1 SET b1='and sixty-seven' WHERE a=3;
UPDATE t1 SET b1='three hundred and ninety-eight' WHERE a=4;
.....
UPDATE t1 SET b1='two hundred and thirty-six' WHERE a=19997;
UPDATE t1 SET b1='four hundred and sixty-nine' WHERE a=19998;
UPDATE t1 SET b1='nine hundred and thirty' WHERE a=19999;
UPDATE t1 SET b1='three hundred and ninety-eight' WHERE a=20000;

14.966sec

MySQL INSERTS :

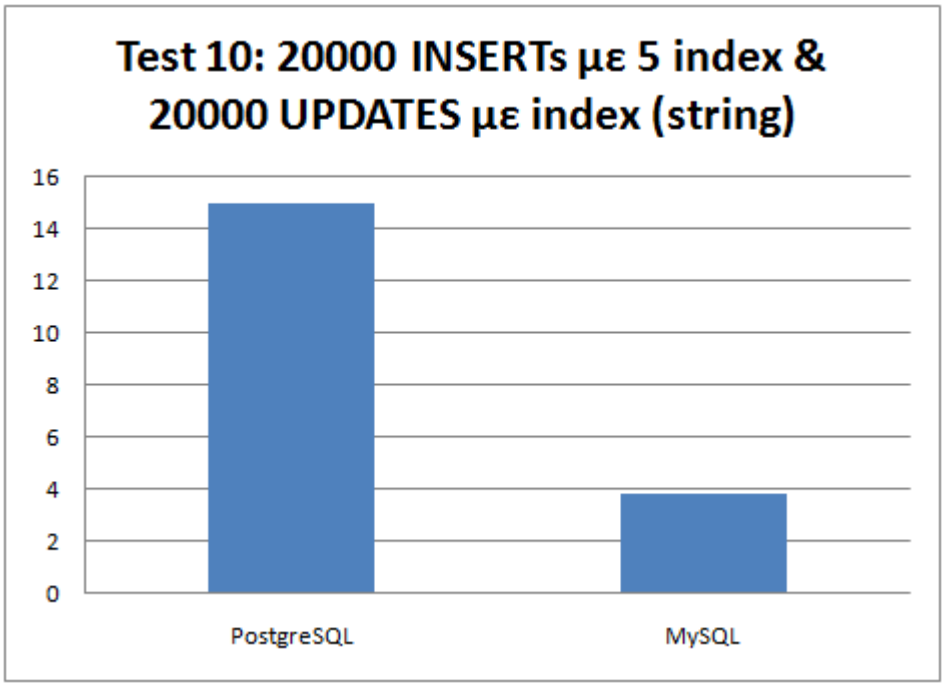
CREATE TABLE t1(a INTEGER, b INTEGER, c INTEGER, b1 VARCHAR(100), c1 VARCHAR(255));
CREATE INDEX i1 ON t1(a);
CREATE INDEX i2 ON t1(b);
CREATE INDEX i3 ON t1(c);
CREATE INDEX i4 ON t1(b1);
CREATE INDEX i5 ON t1(c1);
INSERT INTO t1 VALUES(1,594,507730,'five hundred and ninety-four','five hundred and seven thousand seven hundred and thirty')
INSERT INTO t1 VALUES(2,936,221371,'nine hundred and thirty-six','two hundred and twenty-one thousand three hundred and seventy-one')
INSERT INTO t1 VALUES(3,913,950588,'nine hundred and thirteen','nine hundred and fifty thousand five hundred and eighty-eight')
INSERT INTO t1 VALUES(4,214,990304,'two hundred and fourteen','nine hundred and ninety thousand three hundred and four')
.....
INSERT INTO t1 VALUES(19997,722,360293,'seven hundred and twenty-two','three hundred and sixty thousand two hundred and ninety-three')
INSERT INTO t1 VALUES(19998,953,603036,'nine hundred and fifty-three','six hundred and three thousand and thirty-six')
INSERT INTO t1 VALUES(19999,102,187918,'one hundred and two','one hundred and eighty-seven thousand nine hundred and eighteen')
INSERT INTO t1 VALUES(20000,670,672360,'six hundred and seventy','six hundred and seventy-two thousand three hundred and sixty')

3.278sec

MySQL UPDATES με index:

UPDATE t1 SET b1='two hundred and nine' WHERE a=1;
UPDATE t1 SET b1='five hundred and eighty-seven' WHERE a=2;
UPDATE t1 SET b1='four hundred and eight' WHERE a=3;
UPDATE t1 SET b1='six hundred and fifty-four' WHERE a=4;
.....
UPDATE t1 SET b1='nine hundred and thirty-seven' WHERE a=19997;
UPDATE t1 SET b1='eight hundred and ninety-seven' WHERE a=19998;
UPDATE t1 SET b1='five hundred and fifty-three' WHERE a=19999;
UPDATE t1 SET b1='three hundred and ten' WHERE a=20000;

3.805sec



Test 11: 20000 INSERTs με 5 index & 20000 UPDATES με index (string)

PostgreSQL INSERTS :

CREATE TABLE t1(a INTEGER, b INTEGER, c INTEGER, b1 VARCHAR(100), c1 VARCHAR(255));
INSERT INTO t1 VALUES(1,196,354251,'one hundred and ninety-six','three hundred and fifty-four thousand two hundred and fifty-one')
INSERT INTO t1 VALUES(2,238,249826,'two hundred and thirty-eight','two hundred and fourty-nine thousand eight hundred and twenty-six')
INSERT INTO t1 VALUES(3,696,732070,'six hundred and ninety-six','seven hundred and thirty-two thousand and seventy')
INSERT INTO t1 VALUES(4,526,489657,'five hundred and twenty-six','four hundred and eighty-nine thousand six hundred and fifty-seven')
.....
INSERT INTO t1 VALUES(4997,154,890164,'one hundred and fifty-four','eight hundred and ninety thousand one hundred and sixty-four')
INSERT INTO t1 VALUES(4998,265,158200,'two hundred and sixty-five','one hundred and fifty-eight thousand two hundred')
INSERT INTO t1 VALUES(4999,721,448843,'seven hundred and twenty-one','four hundred and fourty-eight thousand eight hundred and fourty-three')
INSERT INTO t1 VALUES(5000,367,788265,'three hundred and sixty-seven','seven hundred and eighty-eight thousand two hundred and sixty-five')

3.164sec

PostgreSQL UPDATES χωρις index :

CREATE TABLE t2(a INTEGER, b INTEGER, c INTEGER)
INSERT INTO t2 SELECT b,a,c FROM t1;
INSERT INTO t2 SELECT b,a,c FROM t1;
INSERT INTO t2 SELECT b,a,c FROM t1;
INSERT INTO t2 SELECT b,a,c FROM t1;
.....
INSERT INTO t2 SELECT b,a,c FROM t1;
INSERT INTO t2 SELECT b,a,c FROM t1;
INSERT INTO t2 SELECT b,a,c FROM t1;
INSERT INTO t2 SELECT b,a,c FROM t1;

192.285sec

MySQL INSERTS :

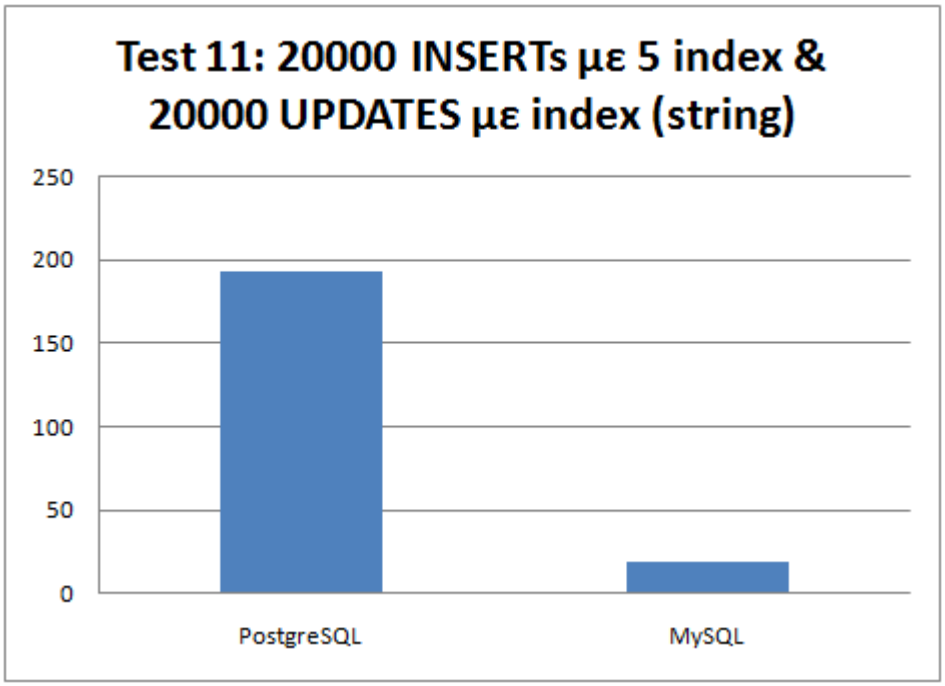
CREATE TABLE t1(a INTEGER, b INTEGER, c INTEGER, b1 VARCHAR(100), c1 VARCHAR(255));
INSERT INTO t1 VALUES(1,155,624514,'one hundred and fifty-five','six hundred and twenty-four thousand five hundred and fourteen')
INSERT INTO t1 VALUES(2,222,502511,'two hundred and twenty-two','five hundred and two thousand five hundred and eleven')
INSERT INTO t1 VALUES(3,564,939053,'five hundred and sixty-four','nine hundred and thirty-nine thousand and fifty-three')
INSERT INTO t1 VALUES(4,813,397811,'eight hundred and thirteen','three hundred and ninety-seven thousand eight hundred and eleven')
.....
INSERT INTO t1 VALUES(4997,718,471365,'seven hundred and eighteen','four hundred and seventy-one thousand three hundred and sixty-five')
INSERT INTO t1 VALUES(4998,741,959323,'seven hundred and fourty-one','nine hundred and fifty-nine thousand three hundred and twenty-three')
INSERT INTO t1 VALUES(4999,969,641763,'nine hundred and sixty-nine','six hundred and fourty-one thousand seven hundred and sixty-three')
INSERT INTO t1 VALUES(5000,920,119857,'nine hundred and twenty','one hundred and nineteen thousand eight hundred and fifty-seven')

0.512sec

MySQL UPDATES χωρις index:

CREATE TABLE t2(a INTEGER, b INTEGER, c INTEGER)
INSERT INTO t2 SELECT b,a,c FROM t1;
INSERT INTO t2 SELECT b,a,c FROM t1;
INSERT INTO t2 SELECT b,a,c FROM t1;
INSERT INTO t2 SELECT b,a,c FROM t1;
.....
INSERT INTO t2 SELECT b,a,c FROM t1;
INSERT INTO t2 SELECT b,a,c FROM t1;
INSERT INTO t2 SELECT b,a,c FROM t1;
INSERT INTO t2 SELECT b,a,c FROM t1;

18.781sec



Test 12: 20000 INSERTs με 5 index & 20000 DELETE TABLE με index

PostgreSQL INSERTS :

```
CREATE TABLE t1(a INTEGER, b INTEGER, c INTEGER, b1 VARCHAR(100), c1 VARCHAR(255));
CREATE INDEX i1 ON t1(a);
CREATE INDEX i2 ON t1(b);
CREATE INDEX i3 ON t1(c);
CREATE INDEX i4 ON t1(b1);
CREATE INDEX i5 ON t1(c1);
INSERT INTO t1 VALUES(1,870,889175,'eight hundred and seventy','eight hundred and eighty-nine thousand one hundred and seventy-five')
INSERT INTO t1 VALUES(2,213,673129,'two hundred and thirteen','six hundred and seventy-three thousand one hundred and twenty-nine')
INSERT INTO t1 VALUES(3,655,685159,'six hundred and fifty-five','six hundred and eighty-five thousand one hundred and fifty-nine')
INSERT INTO t1 VALUES(4,982,120187,'nine hundred and eighty-two','one hundred and twenty thousand one hundred and eighty-seven')
.....
INSERT INTO t1 VALUES(19997,266,235488,'two hundred and sixty-six','two hundred and thirty-five thousand four hundred and eighty-eight')
INSERT INTO t1 VALUES(19998,849,998544,'eight hundred and fourty-nine','nine hundred and ninety-eight thousand five hundred and fourty-four')
INSERT INTO t1 VALUES(19999,913,316238,'nine hundred and thirteen','three hundred and sixteen thousand two hundred and thirty-eight')
INSERT INTO t1 VALUES(20000,257,645993,'two hundred and fifty-seven','six hundred and fourty-five thousand nine hundred and ninety-three')
```

14.102sec

PostgreSQL DELETE:

```
DELETE FROM t1
```

0.043sec

MySQL INSERTS :

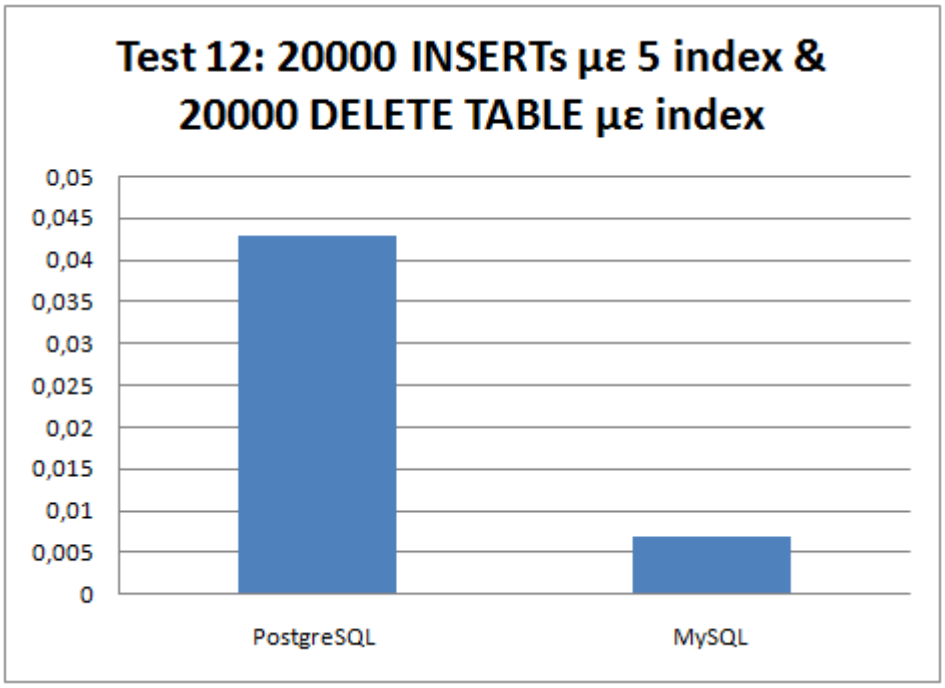
```
CREATE TABLE t1(a INTEGER, b INTEGER, c INTEGER, b1 VARCHAR(100), c1 VARCHAR(255));
CREATE INDEX i1 ON t1(a);
CREATE INDEX i2 ON t1(b);
CREATE INDEX i3 ON t1(c);
CREATE INDEX i4 ON t1(b1);
CREATE INDEX i5 ON t1(c1);
INSERT INTO t1 VALUES(1,341,817105,'three hundred and fourty-one','eight hundred and seventeen thousand one hundred and five')
INSERT INTO t1 VALUES(2,458,305746,'four hundred and fifty-eight','three hundred and five thousand seven hundred and fourty-six')
INSERT INTO t1 VALUES(3,210,809963,'two hundred and ten','eight hundred and nine thousand nine hundred and sixty-three')
INSERT INTO t1 VALUES(4,186,624679,'one hundred and eighty-six','six hundred and twenty-four thousand six hundred and seventy-nine')
.....
INSERT INTO t1 VALUES(19997,469,669668,'four hundred and sixty-nine','six hundred and sixty-nine thousand six hundred and sixty-eight')
INSERT INTO t1 VALUES(19998,475,687411,'four hundred and seventy-five','six hundred and eighty-seven thousand four hundred and eleven')
INSERT INTO t1 VALUES(19999,299,947293,'two hundred and ninety-nine','nine hundred and fourty-seven thousand two hundred and ninety-three')
INSERT INTO t1 VALUES(20000,642,306735,'six hundred and fourty-two','three hundred and six thousand seven hundred and thirty-five')
```

3.125sec

MySQL DELETE:

DELETE FROM t1

0.007sec



Test 13: 20000 INSERTs χωρίς index & 20000 DELETE TABLE χωρίς index

PostgreSQL INSERTS :

```
CREATE TABLE t1(a INTEGER, b INTEGER, c INTEGER, b1 VARCHAR(100), c1 VARCHAR(255));
INSERT INTO t1 VALUES(1,972,302066,'nine hundred and seventy-two','three hundred and two thousand and sixty-six')
INSERT INTO t1 VALUES(2,118,226644,'one hundred and eighteen','two hundred and twenty-six thousand six hundred and fourty-four')
INSERT INTO t1 VALUES(3,588,604299,'five hundred and eighty-eight','six hundred and four thousand two hundred and ninety-nine')
INSERT INTO t1 VALUES(4,268,629953,'two hundred and sixty-eight','six hundred and twenty-nine thousand nine hundred and fifty-three')
.....
INSERT INTO t1 VALUES(19997,706,435879,'seven hundred and six','four hundred and thirty-five thousand eight hundred and seventy-nine')
INSERT INTO t1 VALUES(19998,192,439559,'one hundred and ninety-two','four hundred and thirty-nine thousand five hundred and fifty-nine')
INSERT INTO t1 VALUES(19999,283,122879,'two hundred and eighty-three','one hundred and twenty-two thousand eight hundred and seventy-nine')
INSERT INTO t1 VALUES(20000,781,143258,'seven hundred and eighty-one','one hundred and fourty-three thousand two hundred and fifty-eight')
```

11.93sec

PostgreSQL DELETE:

DELETE FROM t1

0.042sec

MySQL INSERTS :

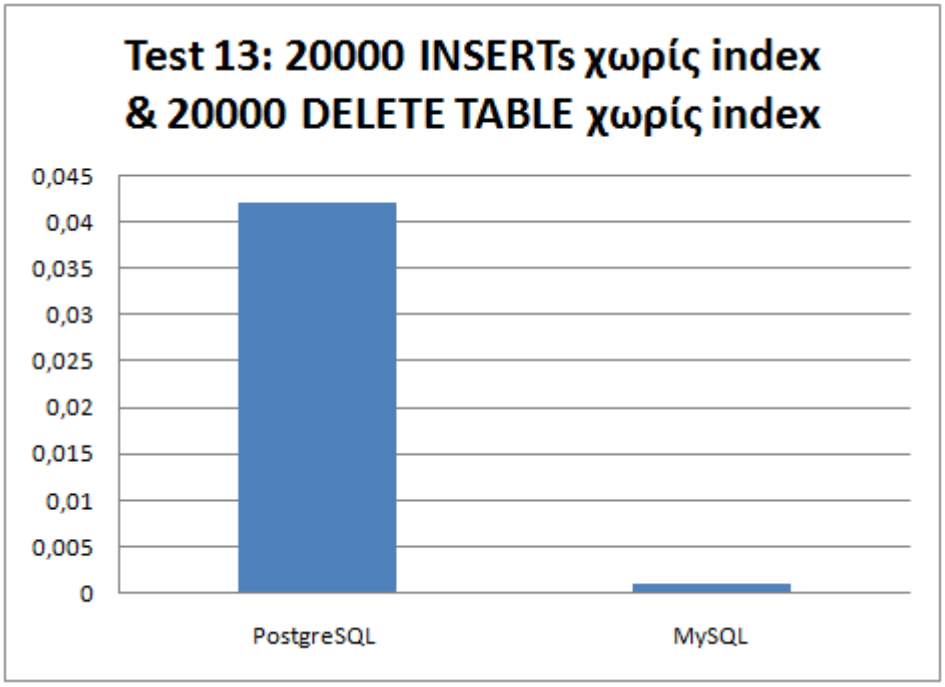
```
CREATE TABLE t1(a INTEGER, b INTEGER, c INTEGER, b1 VARCHAR(100), c1 VARCHAR(255));
INSERT INTO t1 VALUES(1,443,229995,'four hundred and fourty-three','two hundred and twenty-nine thousand nine hundred and ninety-five')
INSERT INTO t1 VALUES(2,363,759262,'three hundred and sixty-three','seven hundred and fifty-nine thousand two hundred and sixty-two')
INSERT INTO t1 VALUES(3,143,729104,'one hundred and fourty-three','seven hundred and twenty-nine thousand one hundred and four')
INSERT INTO t1 VALUES(4,372,234445,'three hundred and seventy-two','two hundred and thirty-four thousand four hundred and fourty-five')
.....
INSERT INTO t1 VALUES(19997,908,870059,'nine hundred and eight','eight hundred and seventy thousand and fifty-nine')
INSERT INTO t1 VALUES(19998,717,128427,'seven hundred and seventeen','one hundred and twenty-eight thousand four hundred and twenty-seven')
INSERT INTO t1 VALUES(19999,570,753933,'five hundred and seventy','seven hundred and fifty-three thousand nine hundred and thirty-three')
INSERT INTO t1 VALUES(20000,266,704000,'two hundred and sixty-six','seven hundred and four thousand')
```

1.884sec

MySQL DELETE:

DELETE FROM t1

0.001sec



Test 14: 20000 INSERTs χωρίς index & INSERT from SELECT (big)

PostgreSQL INSERTS :

```
CREATE TABLE t1(a INTEGER, b INTEGER, c INTEGER, b1 VARCHAR(100), c1 VARCHAR(255));
INSERT INTO t1 VALUES(1,514,339859,'five hundred and fourteen','three hundred and thirty-nine thousand eight hundred and fifty-nine')
INSERT INTO t1 VALUES(2,188,144906,'one hundred and eighty-eight','one hundred and fourty-four thousand nine hundred and six')
INSERT INTO t1 VALUES(3,229,571780,'two hundred and twenty-nine','five hundred and seventy-one thousand seven hundred and eighty')
INSERT INTO t1 VALUES(4,774,365402,'seven hundred and seventy-four','three hundred and sixty-five thousand four hundred and two')
.....
INSERT INTO t1 VALUES(4997,167,837100,'one hundred and sixty-seven','eight hundred and thirty-seven thousand one hundred')
INSERT INTO t1 VALUES(4998,500,492733,'five hundred','four hundred and ninety-two thousand seven hundred and thirty-three')
INSERT INTO t1 VALUES(4999,454,531130,'four hundred and fifty-four','five hundred and thirty-one thousand one hundred and thirty')
INSERT INTO t1 VALUES(5000,955,934713,'nine hundred and fifty-five','nine hundred and thirty-four thousand seven hundred and thirteen')
```

3.28sec

PostgreSQL DELETE:

```
CREATE TABLE t2(a INTEGER, b INTEGER, c INTEGER, b1 VARCHAR(100), c1 VARCHAR(255));
INSERT INTO t2 SELECT b,a,c FROM t1;
INSERT INTO t2 SELECT b,a,c FROM t1;
INSERT INTO t2 SELECT b,a,c FROM t1;
INSERT INTO t2 SELECT b,a,c FROM t1;
.....
INSERT INTO t2 SELECT b,a,c FROM t1;
INSERT INTO t2 SELECT b,a,c FROM t1;
INSERT INTO t2 SELECT b,a,c FROM t1;
INSERT INTO t2 SELECT b,a,c FROM t1;
```

191.156sec

MySQL INSERTS :

```
CREATE TABLE t1(a INTEGER, b INTEGER, c INTEGER, b1 VARCHAR(100), c1 VARCHAR(255));
INSERT INTO t1 VALUES(1,986,757778,'nine hundred and eighty-six','seven hundred and fifty-seven thousand seven hundred and seventy-eight')
INSERT INTO t1 VALUES(2,966,601498,'nine hundred and sixty-six','six hundred and one thousand four hundred and ninety-eight')
```

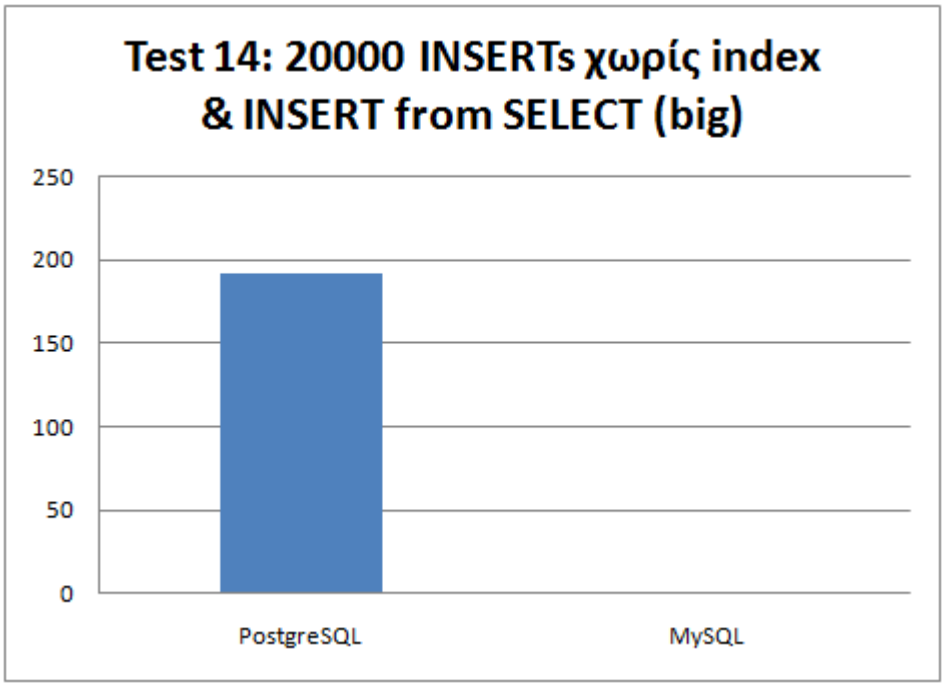
INSERT INTO t1 VALUES(3,722,588919,'seven hundred and twenty-two','five hundred and eighty-eight thousand nine hundred and nineteen')
INSERT INTO t1 VALUES(4,167,139962,'one hundred and sixty-seven','one hundred and thirty-nine thousand nine hundred and sixty-two')
.....
INSERT INTO t1 VALUES(4997,569,790957,'five hundred and sixty-nine','seven hundred and ninety thousand nine hundred and fifty-seven')
INSERT INTO t1 VALUES(4998,195,822763,'one hundred and ninety-five','eight hundred and twenty-two thousand seven hundred and sixty-three')
INSERT INTO t1 VALUES(4999,653,759207,'six hundred and fifty-three','seven hundred and fifty-nine thousand two hundred and seven')
INSERT INTO t1 VALUES(5000,841,357711,'eight hundred and fourty-one','three hundred and fifty-seven thousand seven hundred and eleven')

0.511sec

MySQL DELETE:

CREATE TABLE t2(a INTEGER, b INTEGER, c INTEGER, b1 VARCHAR(100), c1 VARCHAR(255));
INSERT INTO t2 SELECT b,a,c FROM t1;
INSERT INTO t2 SELECT b,a,c FROM t1;
INSERT INTO t2 SELECT b,a,c FROM t1;
INSERT INTO t2 SELECT b,a,c FROM t1;
.....
INSERT INTO t2 SELECT b,a,c FROM t1;
INSERT INTO t2 SELECT b,a,c FROM t1;
INSERT INTO t2 SELECT b,a,c FROM t1;
INSERT INTO t2 SELECT b,a,c FROM t1;

0.382sec



Test 1 (movieplace) : New Movies with Subtitles

PostgreSQL SELECTS :

SELECT movies.id,ellinikostitlos,ksenostitlos,imerominiaupload FROM movies INNER JOIN subtitles ON movies.id = subtitles.id WHERE subtitles.dekto = '1' and subtitles.imerominiaupload <=ALL (SELECT s.imerominiaupload FROM subtitles s WHERE movies.id = s.id and s.dekto = '1' and s.idglossas = 1) and subtitles.idglossas = 1 and movies.dekto = '1' and (movies.episimipremiera>'2009-05-27' or movies.ellinikipremiera>'2009-05-27') and movies.premieradvd is NULL and movies.seira = '0' ORDER BY subtitles.imerominiaupload DESC LIMIT 4
SELECT movies.id,ellinikostitlos,ksenostitlos,imerominiaupload FROM movies INNER JOIN subtitles ON movies.id = subtitles.id WHERE subtitles.dekto = '1' and subtitles.imerominiaupload <=ALL (SELECT s.imerominiaupload FROM subtitles s WHERE movies.id = s.id and s.dekto = '1' and s.idglossas = 1) and subtitles.idglossas = 1 and movies.dekto = '1' and (movies.episimipremiera>'2009-05-28' or movies.ellinikipremiera>'2009-05-28') and movies.premieradvd is NULL and movies.seira = '0' ORDER BY subtitles.imerominiaupload DESC LIMIT 4
SELECT movies.id,ellinikostitlos,ksenostitlos,imerominiaupload FROM movies INNER JOIN subtitles ON movies.id = subtitles.id WHERE subtitles.dekto = '1' and subtitles.imerominiaupload <=ALL (SELECT s.imerominiaupload FROM subtitles s WHERE movies.id = s.id and s.dekto = '1' and s.idglossas = 1) and subtitles.idglossas = 1 and movies.dekto = '1' and (movies.episimipremiera>'2009-05-29' or movies.ellinikipremiera>'2009-05-29') and movies.premieradvd is NULL and movies.seira = '0' ORDER BY subtitles.imerominiaupload DESC LIMIT 4
SELECT movies.id,ellinikostitlos,ksenostitlos,imerominiaupload FROM movies INNER JOIN subtitles ON movies.id = subtitles.id WHERE subtitles.dekto = '1' and subtitles.imerominiaupload <=ALL (SELECT s.imerominiaupload FROM subtitles s WHERE movies.id = s.id and s.dekto = '1' and s.idglossas = 1) and subtitles.idglossas = 1 and movies.dekto = '1' and (movies.episimipremiera>'2009-05-30' or movies.ellinikipremiera>'2009-05-30') and movies.premieradvd is NULL and movies.seira = '0' ORDER BY subtitles.imerominiaupload DESC LIMIT 4
.....
SELECT movies.id,ellinikostitlos,ksenostitlos,imerominiaupload FROM movies INNER JOIN subtitles ON movies.id = subtitles.id WHERE subtitles.dekto = '1' and subtitles.imerominiaupload <=ALL (SELECT s.imerominiaupload FROM subtitles s WHERE movies.id = s.id and s.dekto = '1' and s.idglossas = 1) and subtitles.idglossas = 1 and movies.dekto = '1' and (movies.episimipremiera>'2012-02-17' or movies.ellinikipremiera>'2012-02-17') and movies.premieradvd is NULL

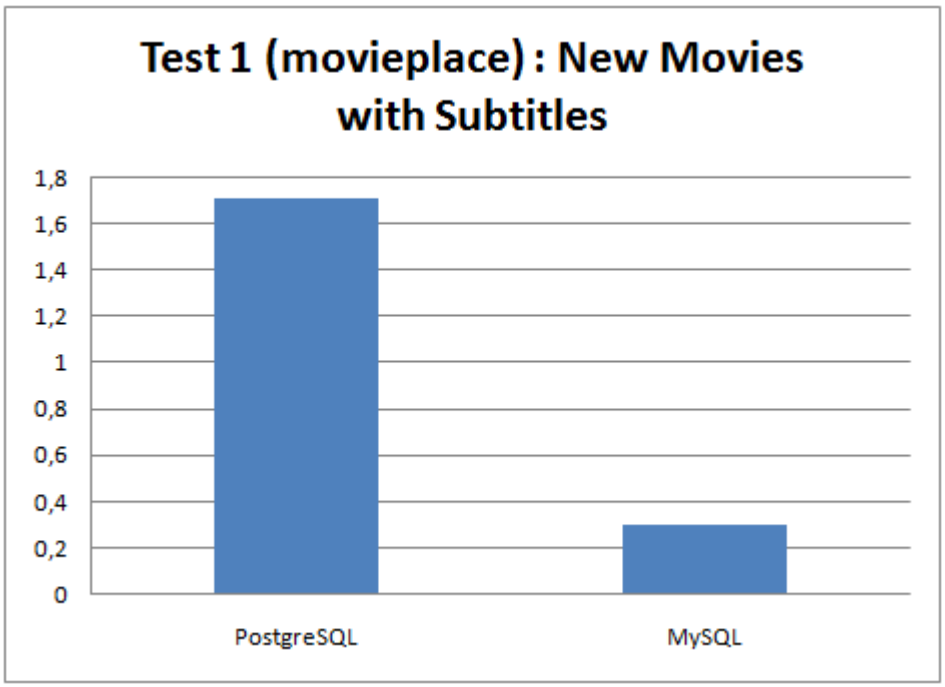
and movies.seira = '0' ORDER BY subtitles.imerominiaupload DESC LIMIT 4
SELECT movies.id,ellinikostitlos,ksenostitlos,imerominiaupload FROM movies INNER JOIN subtitles ON movies.id = subtitles.id WHERE subtitles.dekto = '1' and subtitles.imerominiaupload <=ALL (SELECT s.imerominiaupload FROM subtitles s WHERE movies.id = s.id and s.dekto = '1' and s.idglossas = 1) and subtitles.idglossas = 1 and movies.dekto = '1' and (movies.episimipremiera>'2012-02-18' or movies.ellinikipremiera>'2012-02-18') and movies.premieradvd is NULL and movies.seira = '0' ORDER BY subtitles.imerominiaupload DESC LIMIT 4
SELECT movies.id,ellinikostitlos,ksenostitlos,imerominiaupload FROM movies INNER JOIN subtitles ON movies.id = subtitles.id WHERE subtitles.dekto = '1' and subtitles.imerominiaupload <=ALL (SELECT s.imerominiaupload FROM subtitles s WHERE movies.id = s.id and s.dekto = '1' and s.idglossas = 1) and subtitles.idglossas = 1 and movies.dekto = '1' and (movies.episimipremiera>'2012-02-19' or movies.ellinikipremiera>'2012-02-19') and movies.premieradvd is NULL and movies.seira = '0' ORDER BY subtitles.imerominiaupload DESC LIMIT 4
SELECT movies.id,ellinikostitlos,ksenostitlos,imerominiaupload FROM movies INNER JOIN subtitles ON movies.id = subtitles.id WHERE subtitles.dekto = '1' and subtitles.imerominiaupload <=ALL (SELECT s.imerominiaupload FROM subtitles s WHERE movies.id = s.id and s.dekto = '1' and s.idglossas = 1) and subtitles.idglossas = 1 and movies.dekto = '1' and (movies.episimipremiera>'2012-02-20' or movies.ellinikipremiera>'2012-02-20') and movies.premieradvd is NULL and movies.seira = '0' ORDER BY subtitles.imerominiaupload DESC LIMIT 4

1.707sec

MySQL SELECTS :

SELECT movies.id,ellinikostitlos,ksenostitlos,imerominiaupload FROM movies INNER JOIN subtitles ON movies.id = subtitles.id WHERE subtitles.dekto = 1 and subtitles.imerominiaupload <=ALL (SELECT s.imerominiaupload FROM subtitles s WHERE movies.id = s.id and s.dekto = 1 and s.idglossas = 1) and subtitles.idglossas = 1 and movies.dekto = 1 and (movies.episimipremiera>'2009-05-27' or movies.ellinikipremiera>'2009-05-27') and movies.premieradvd is NULL and movies.seira = '0' ORDER BY subtitles.imerominiaupload DESC LIMIT 4
SELECT movies.id,ellinikostitlos,ksenostitlos,imerominiaupload FROM movies INNER JOIN subtitles ON movies.id = subtitles.id WHERE subtitles.dekto = 1 and subtitles.imerominiaupload <=ALL (SELECT s.imerominiaupload FROM subtitles s WHERE movies.id = s.id and s.dekto = 1 and s.idglossas = 1) and subtitles.idglossas = 1 and movies.dekto = 1 and (movies.episimipremiera>'2009-05-28' or movies.ellinikipremiera>'2009-05-28') and movies.premieradvd is NULL and movies.seira = '0' ORDER BY subtitles.imerominiaupload DESC LIMIT 4
SELECT movies.id,ellinikostitlos,ksenostitlos,imerominiaupload FROM movies INNER JOIN subtitles ON movies.id = subtitles.id WHERE subtitles.dekto = 1 and subtitles.imerominiaupload <=ALL (SELECT s.imerominiaupload FROM subtitles s WHERE movies.id = s.id and s.dekto = 1 and s.idglossas = 1) and subtitles.idglossas = 1 and movies.dekto = 1 and (movies.episimipremiera>'2009-05-29' or movies.ellinikipremiera>'2009-05-29') and movies.premieradvd is NULL and movies.seira = '0' ORDER BY subtitles.imerominiaupload DESC LIMIT 4
SELECT movies.id,ellinikostitlos,ksenostitlos,imerominiaupload FROM movies INNER JOIN subtitles ON movies.id = subtitles.id WHERE subtitles.dekto = 1 and subtitles.imerominiaupload <=ALL (SELECT s.imerominiaupload FROM subtitles s WHERE movies.id = s.id and s.dekto = 1 and s.idglossas = 1) and subtitles.idglossas = 1 and movies.dekto = 1 and (movies.episimipremiera>'2009-05-30' or movies.ellinikipremiera>'2009-05-30') and movies.premieradvd is NULL and movies.seira = '0' ORDER BY subtitles.imerominiaupload DESC LIMIT 4
.....
SELECT movies.id,ellinikostitlos,ksenostitlos,imerominiaupload FROM movies INNER JOIN subtitles ON movies.id = subtitles.id WHERE subtitles.dekto = 1 and subtitles.imerominiaupload <=ALL (SELECT s.imerominiaupload FROM subtitles s WHERE movies.id = s.id and s.dekto = 1 and s.idglossas = 1) and subtitles.idglossas = 1 and movies.dekto = 1 and (movies.episimipremiera>'2012-02-17' or movies.ellinikipremiera>'2012-02-17') and movies.premieradvd is NULL and movies.seira = '0' ORDER BY subtitles.imerominiaupload DESC LIMIT 4
SELECT movies.id,ellinikostitlos,ksenostitlos,imerominiaupload FROM movies INNER JOIN subtitles ON movies.id = subtitles.id WHERE subtitles.dekto = 1 and subtitles.imerominiaupload <=ALL (SELECT s.imerominiaupload FROM subtitles s WHERE movies.id = s.id and s.dekto = 1 and s.idglossas = 1) and subtitles.idglossas = 1 and movies.dekto = 1 and (movies.episimipremiera>'2012-02-18' or movies.ellinikipremiera>'2012-02-18') and movies.premieradvd is NULL and movies.seira = '0' ORDER BY subtitles.imerominiaupload DESC LIMIT 4
SELECT movies.id,ellinikostitlos,ksenostitlos,imerominiaupload FROM movies INNER JOIN subtitles ON movies.id = subtitles.id WHERE subtitles.dekto = 1 and subtitles.imerominiaupload <=ALL (SELECT s.imerominiaupload FROM subtitles s WHERE movies.id = s.id and s.dekto = 1 and s.idglossas = 1) and subtitles.idglossas = 1 and movies.dekto = 1 and (movies.episimipremiera>'2012-02-19' or movies.ellinikipremiera>'2012-02-19') and movies.premieradvd is NULL and movies.seira = '0' ORDER BY subtitles.imerominiaupload DESC LIMIT 4
SELECT movies.id,ellinikostitlos,ksenostitlos,imerominiaupload FROM movies INNER JOIN subtitles ON movies.id = subtitles.id WHERE subtitles.dekto = 1 and subtitles.imerominiaupload <=ALL (SELECT s.imerominiaupload FROM subtitles s WHERE movies.id = s.id and s.dekto = 1 and s.idglossas = 1) and subtitles.idglossas = 1 and movies.dekto = 1 and (movies.episimipremiera>'2012-02-20' or movies.ellinikipremiera>'2012-02-20') and movies.premieradvd is NULL and movies.seira = '0' ORDER BY subtitles.imerominiaupload DESC LIMIT 4

0.297sec



Test 2 (movieplace) : 20000 SELECT INFOS

PostgreSQL SELECTS :

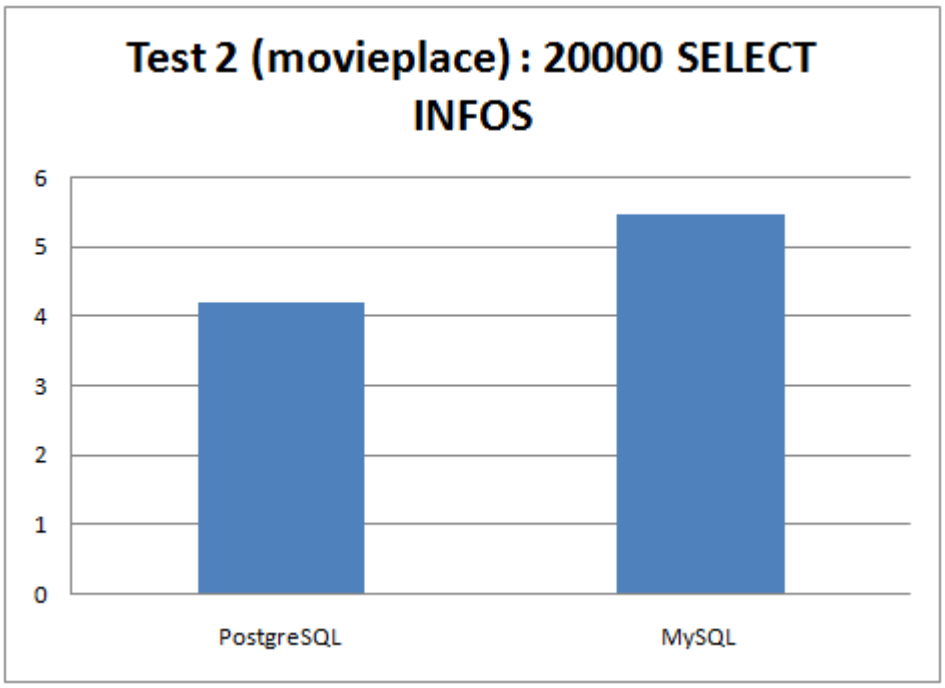
SELECT * FROM movies WHERE id='1' and dekto='1'
SELECT * FROM movies WHERE id='2' and dekto='1'
SELECT * FROM movies WHERE id='3' and dekto='1'
SELECT * FROM movies WHERE id='4' and dekto='1'
.....
SELECT * FROM movies WHERE id='19997' and dekto='1'
SELECT * FROM movies WHERE id='19998' and dekto='1'
SELECT * FROM movies WHERE id='19999' and dekto='1'
SELECT * FROM movies WHERE id='20000' and dekto='1'

4.208sec

MySQL SELECTS :

SELECT * FROM movies WHERE id='1' and dekto=1
SELECT * FROM movies WHERE id='2' and dekto=1
SELECT * FROM movies WHERE id='3' and dekto=1
SELECT * FROM movies WHERE id='4' and dekto=1
.....
SELECT * FROM movies WHERE id='19997' and dekto=1
SELECT * FROM movies WHERE id='19998' and dekto=1
SELECT * FROM movies WHERE id='19999' and dekto=1
SELECT * FROM movies WHERE id='20000' and dekto=1

5.46sec



Test 3 (movieplace) : 1150 SELECT SEARCH MOVIES

PostgreSQL SELECTS :

SELECT * FROM movies WHERE movies.dekto='1' and (movies.ksenostitlos like '%what about morgans%' or movies.ellinikostitlos like '%what about morgans%') ORDER BY episimipremiera DESC LIMIT 30
SELECT * FROM movies WHERE movies.dekto='1' and (movies.ksenostitlos like '%harry potter 1%' or movies.ellinikostitlos like '%harry potter 1%') ORDER BY episimipremiera DESC LIMIT 30
SELECT * FROM movies WHERE movies.dekto='1' and (movies.ksenostitlos like '%harry potter%' or movies.ellinikostitlos like '%harry potter%') ORDER BY episimipremiera DESC LIMIT 30
SELECT * FROM movies WHERE movies.dekto='1' and (movies.ksenostitlos like '%Shutter Island%' or movies.ellinikostitlos like '%Shutter Island%') ORDER BY episimipremiera DESC LIMIT 30
.....
SELECT * FROM movies WHERE movies.dekto='1' and (movies.ksenostitlos like '% the color %' or movies.ellinikostitlos like '% the color %') ORDER BY episimipremiera DESC LIMIT 30
SELECT * FROM movies WHERE movies.dekto='1' and (movies.ksenostitlos like '%alvin 2%' or movies.ellinikostitlos like '%alvin 2%') ORDER BY episimipremiera DESC LIMIT 30
SELECT * FROM movies WHERE movies.dekto='1' and (movies.ksenostitlos like '%My Super Ex-Girlfriend%' or movies.ellinikostitlos like '%My Super Ex-Girlfriend%') ORDER BY episimipremiera DESC LIMIT 30
SELECT * FROM movies WHERE movies.dekto='1' and (movies.ksenostitlos like '%SHARK TALE%' or movies.ellinikostitlos like '%SHARK TALE%') ORDER BY

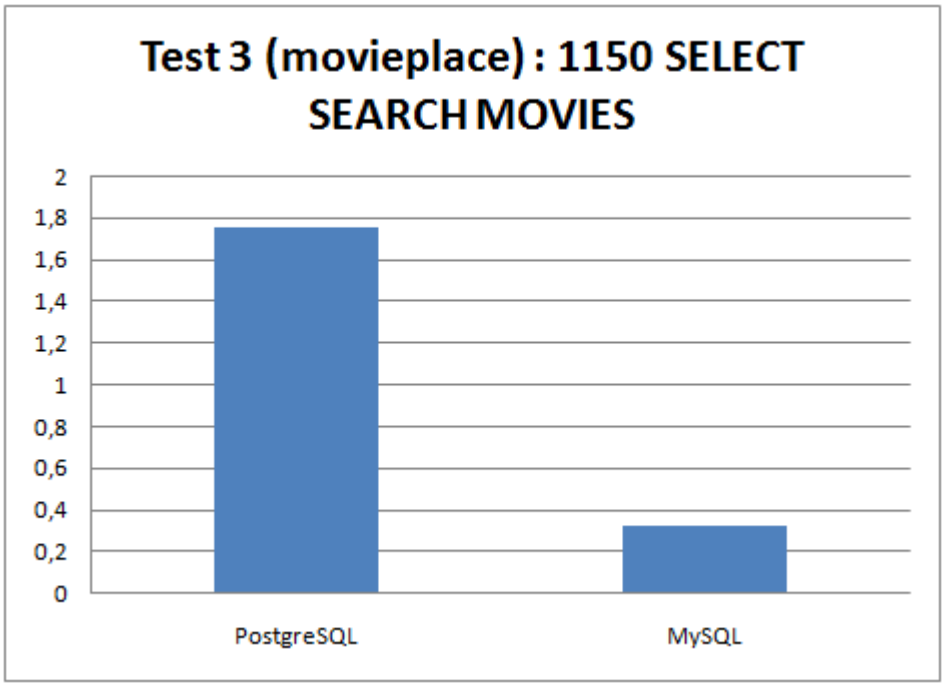
episimipremiera DESC LIMIT 30

1.756sec

MySQL SELECTS :

SELECT * FROM movies WHERE movies.dekto=1 and (movies.ksenostitlos like '%what about morgans%' or movies.ellinikostitlos like '%what about morgans%') ORDER BY episimipremiera DESC LIMIT 30
SELECT * FROM movies WHERE movies.dekto=1 and (movies.ksenostitlos like '%harry potter 1%' or movies.ellinikostitlos like '%harry potter 1%') ORDER BY episimipremiera DESC LIMIT 30
SELECT * FROM movies WHERE movies.dekto=1 and (movies.ksenostitlos like '%harry potter%' or movies.ellinikostitlos like '%harry potter%') ORDER BY episimipremiera DESC LIMIT 30
SELECT * FROM movies WHERE movies.dekto=1 and (movies.ksenostitlos like '%Shutter Island%' or movies.ellinikostitlos like '%Shutter Island%') ORDER BY episimipremiera DESC LIMIT 30
.....
SELECT * FROM movies WHERE movies.dekto=1 and (movies.ksenostitlos like '%the color%' or movies.ellinikostitlos like '% the color %') ORDER BY episimipremiera DESC LIMIT 30
SELECT * FROM movies WHERE movies.dekto=1 and (movies.ksenostitlos like '%alvin 2%' or movies.ellinikostitlos like '%alvin 2%') ORDER BY episimipremiera DESC LIMIT 30
SELECT * FROM movies WHERE movies.dekto=1 and (movies.ksenostitlos like '%My Super Ex-Girlfriend%' or movies.ellinikostitlos like '%My Super Ex-Girlfriend%') ORDER BY episimipremiera DESC LIMIT 30
SELECT * FROM movies WHERE movies.dekto=1 and (movies.ksenostitlos like '%SHARK TALE%' or movies.ellinikostitlos like '%SHARK TALE%') ORDER BY episimipremiera DESC LIMIT 30

0.319sec



Test 4 (movieplace) : 2070 SELECT SEARCH SUBTITLES

PostgreSQL SELECTS :

SELECT * FROM subtitles LEFT JOIN glosses ON subtitles.idglossas = glosses.idglossas LEFT JOIN movies ON subtitles.id=movies.id WHERE subtitles.dekto = '1' and (subtitles.onomatainias like '%agatha%' or movies.ksenostitlos like '%agatha%') ORDER BY idipotitlou DESC
SELECT * FROM subtitles LEFT JOIN glosses ON subtitles.idglossas = glosses.idglossas LEFT JOIN movies ON subtitles.id=movies.id WHERE subtitles.dekto = '1' and (subtitles.onomatainias like '%Motherhood%' or movies.ksenostitlos like '%Motherhood%') ORDER BY idipotitlou DESC
SELECT * FROM subtitles LEFT JOIN glosses ON subtitles.idglossas = glosses.idglossas LEFT JOIN movies ON subtitles.id=movies.id WHERE subtitles.dekto = '1' and (subtitles.onomatainias like '%human body: pushing the limits%' or movies.ksenostitlos like '%human body: pushing the limits%') ORDER BY idipotitlou DESC
SELECT * FROM subtitles LEFT JOIN glosses ON subtitles.idglossas = glosses.idglossas LEFT JOIN movies ON subtitles.id=movies.id WHERE subtitles.dekto = '1' and (subtitles.onomatainias like '%Percy Jackson and the Olympians%' or movies.ksenostitlos like '%Percy Jackson and the Olympians%') ORDER BY idipotitlou DESC
.....
SELECT * FROM subtitles LEFT JOIN glosses ON subtitles.idglossas = glosses.idglossas LEFT JOIN movies ON subtitles.id=movies.id WHERE subtitles.dekto = '1' and (subtitles.onomatainias like '%csi 13%' or movies.ksenostitlos like '%csi 13%') ORDER BY idipotitlou DESC
SELECT * FROM subtitles LEFT JOIN glosses ON subtitles.idglossas = glosses.idglossas LEFT JOIN movies ON subtitles.id=movies.id WHERE subtitles.dekto = '1' and (subtitles.onomatainias like '%the color purple%' or movies.ksenostitlos like '%the color purple%') ORDER BY idipotitlou DESC
SELECT * FROM subtitles LEFT JOIN glosses ON subtitles.idglossas = glosses.idglossas LEFT JOIN movies ON subtitles.id=movies.id WHERE subtitles.dekto = '1' and (subtitles.onomatainias like '%thecolor%' or movies.ksenostitlos like '%the color%') ORDER BY idipotitlou DESC
SELECT * FROM subtitles LEFT JOIN glosses ON subtitles.idglossas = glosses.idglossas LEFT JOIN movies ON subtitles.id=movies.id WHERE subtitles.dekto = '1' and (subtitles.onomatainias like '%csi las vegas%' or movies.ksenostitlos like '%csi las vegas%') ORDER BY idipotitlou DESC

25.651sec

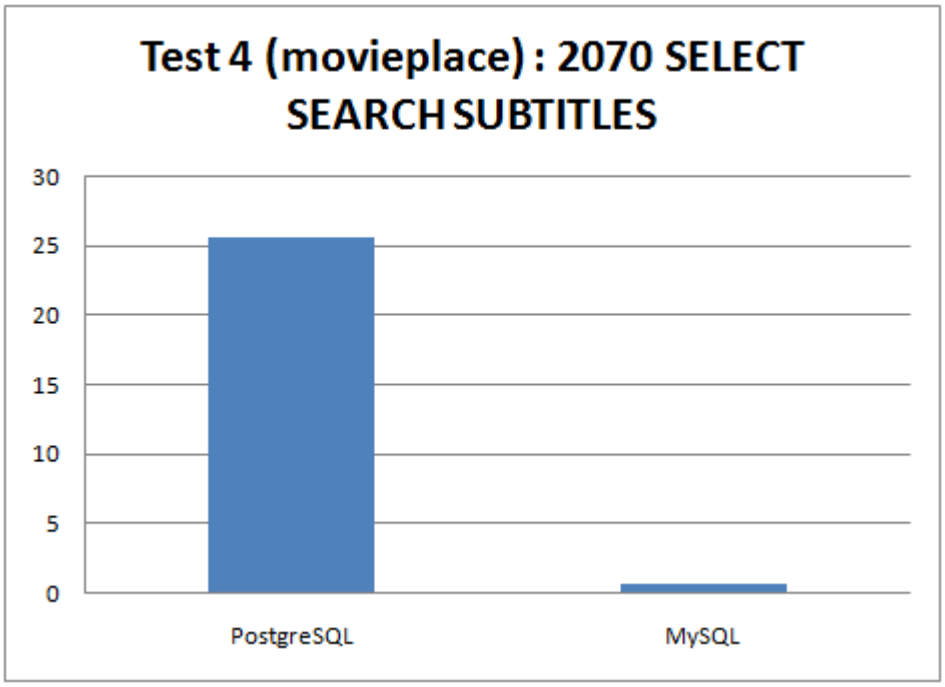
MySQL SELECTS :

SELECT * FROM subtitles LEFT JOIN glosses ON subtitles.idglossas = glosses.idglossas LEFT JOIN movies ON subtitles.id=movies.id WHERE subtitles.dekto = 1 and (subtitles.onomatainias like '%agatha%' or movies.ksenostitlos like '%agatha%') ORDER BY idipotitlou DESC
SELECT * FROM subtitles LEFT JOIN glosses ON subtitles.idglossas = glosses.idglossas LEFT JOIN movies ON subtitles.id=movies.id WHERE subtitles.dekto = 1 and (subtitles.onomatainias like '%Motherhood%' or movies.ksenostitlos like '%Motherhood%') ORDER BY idipotitlou DESC
SELECT * FROM subtitles LEFT JOIN glosses ON subtitles.idglossas = glosses.idglossas LEFT JOIN movies ON subtitles.id=movies.id WHERE subtitles.dekto = 1 and (subtitles.onomatainias like '%human body: pushing the limits%' or movies.ksenostitlos like '%human body: pushing the limits%') ORDER BY idipotitlou DESC
SELECT * FROM subtitles LEFT JOIN glosses ON subtitles.idglossas = glosses.idglossas LEFT JOIN movies ON subtitles.id=movies.id WHERE subtitles.dekto = 1 and (subtitles.onomatainias like '%Percy Jackson and the Olympians%' or movies.ksenostitlos like '%Percy Jackson and the Olympians%') ORDER BY idipotitlou DESC

.....

SELECT * FROM subtitles LEFT JOIN glosses ON subtitles.idglossas = glosses.idglossas LEFT JOIN movies ON subtitles.id=movies.id WHERE subtitles.dekto = 1 and (subtitles.onomatainias like '%csi 13%' or movies.ksenostitlos like '%csi 13%') ORDER BY idipotitlou DESC
SELECT * FROM subtitles LEFT JOIN glosses ON subtitles.idglossas = glosses.idglossas LEFT JOIN movies ON subtitles.id=movies.id WHERE subtitles.dekto = 1 and (subtitles.onomatainias like '%the color purple%' or movies.ksenostitlos like '%the color purple%') ORDER BY idipotitlou DESC
SELECT * FROM subtitles LEFT JOIN glosses ON subtitles.idglossas = glosses.idglossas LEFT JOIN movies ON subtitles.id=movies.id WHERE subtitles.dekto = 1 and (subtitles.onomatainias like '%the color%' or movies.ksenostitlos like '%the color%') ORDER BY idipotitlou DESC
SELECT * FROM subtitles LEFT JOIN glosses ON subtitles.idglossas = glosses.idglossas LEFT JOIN movies ON subtitles.id=movies.id WHERE subtitles.dekto = 1 and (subtitles.onomatainias like '%csi las vegas%' or movies.ksenostitlos like '%csi las vegas%') ORDER BY idipotitlou DESC

0.614sec



11. ΠΕΡΙΓΡΑΦΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

Η μέχρι τώρα περιγραφή έχει γίνει όσον αφορά τον εσωτερικό μηχανισμό της εφαρμογής. Έκτος όμως από τον τρόπο που θα λειτουργεί ο ιστοχώρος πρέπει να δωθεί σημαντικό βάρος στα λειτουργικά και οπτικά χαρακτηριστικά που θα κάνουν τους χρήστες του, να το εμπιστευτούν και να το χρησιμοποιήσουν και στο μέλλον.

Διάφορες μελέτες που έχουν γίνει κατά καιρούς έχουν δείξει πως όσο δυσκολεύεται ο χρήστης να βρει και να εκτελέσει μια επιθυμητή ενέργεια τόσο πιο δύσκολα επιστρέφει στο μέλλον στην εφαρμογή. Σκοπός μας όμως είναι να κρατήσουμε τον χρήστη και γι' αυτό πρέπει να βρεθεί ένα αποδοτικός τρόπος.

Ένα σημαντικό θέμα είναι να κατανοηθεί πως σκέφτεται ο χρήστης, όπως τι βλέπει πρώτα και τι στο τέλος. Η παρακάτω εικόνα δείχνει χαρακτηριστικά την πορεία του ανθρώπινου ματιού πάνω σε ένα ιστοχώρο

Το περιεχόμενο ενός ιστοχώρου είναι πολύ πιο σημαντικό απ' ότι το design. Άρα ένα τέλεια σχεδιασμένο ιστοχώρο, που όμως δεν έχει πλούσιο υλικό σιγά σιγά ο χρήστης θα το εγκαταλείψει.

Επίσης πρέπει να τονιστεί πως ο χρήστης δεν διαβάζει... αλλά scanαρει. Ελέγχουν κάποια τονισμένα μέρη του ιστοχώρου και αν τους τραβήξουν την προσοχή τότε ίσως να διαβάσουν και το υπόλοιπο κείμενο.

Αρά το συμπέρασμα ήταν πως για να στηθεί ένα σωστό ιστοχώρο θα έπρεπε να ακολουθηθεί μια συγκεκριμένη στρατηγική έτσι ώστε να πετύχουμε το βέλτιστο αποτέλεσμα.

Η στρατηγική αυτή περιλάμβανε τα εξής στάδια

1) Προσδιορισμός Στόχων
2) Προσδιορισμός Κοινού
3) Προσδιορισμός Αναγνωριστικού
4) Μετρήσεις Απόδοσης

Προσδιορισμός Στόχων

Ένα από τα πρώτα πράγματα που χρειάζεται πριν αρχίσει η σχεδίαση ενός ιστοχώρου είναι να ξεκαθαριστούν οι στόχοι. Τι προσπαθούμε να πετύχουμε με το καινούργιο ή βελτιωμένο design που θα φτιαχτεί. Ποιος είναι ο βασικός στόχος? Αυτά είναι ερωτήματα τα οποία πρέπει οπωσδήποτε να απαντηθούν έτσι ώστε να υπάρχει μια ξεκάθαρη πορεία.

Πρέπει να ξεκαθαριστεί πως ένας ιστοχώρος δεν είναι ένα κομμάτι τέχνης αλλά ένα περιβάλλον που εξυπηρετεί μια συγκεκριμένη λειτουργία. Αυτή η λειτουργία είναι που θα κάνει τους χρήστες μας να επισκεφτούν τον ιστοχώρο μας αλλά και να μείνουν ικανοποιημένοι έτσι ώστε να το ξαναεπισκεφτούν. Οποιαδήποτε και αν είναι αυτή η λειτουργία, το design πρέπει να εστιαστεί εκεί ώστε να την εκπληρώσει. Οι στόχοι είναι πολύ σημαντικοί, ιδιαίτερα όταν αφορά επανασχεδιασμό. Πρέπει να αναρωτηθεί ο σχεδιαστής για ποιο σκοπό ξεκίνησε τον επανασχεδιασμό: αύξηση χρηστών? Αύξηση συμμετοχής χρηστών?

Βασικός στόχος της δική μας προσπάθειας ήταν η συλλογή πληροφοριών από ταινίες αλλά και συλλογή από υπότιτλους για κάθε ταινία.

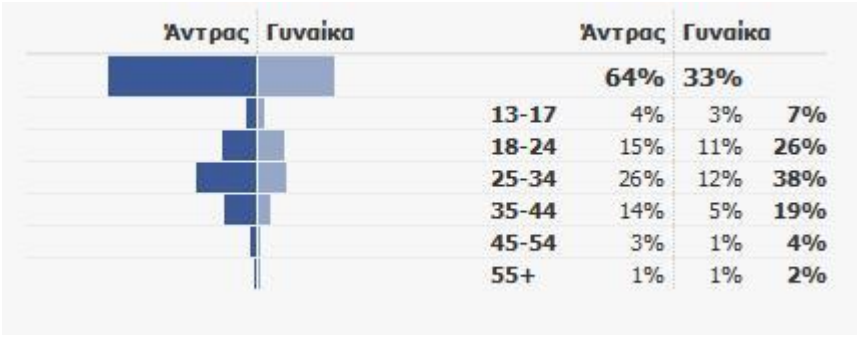
Ο επανασχεδιασμός που γινόταν όλα αυτά τα χρόνια είχε ως σκοπό να κάνει τον ιστοχώρο πιο φιλικό στους υπάρχοντες χρήστες έτσι ώστε να μην φύγουν, αφού ο ιστοχώρος βρισκόταν ήδη σε πολλή καλό στάδιο. Επίσης διευκολύνθηκε με τον τελευταίο σχεδιασμό το social networking όπου είναι μια νέα «μόδα» στον κλάδο του internet αλλά και της διαφήμισης.

Προσδιορισμός Κοινού

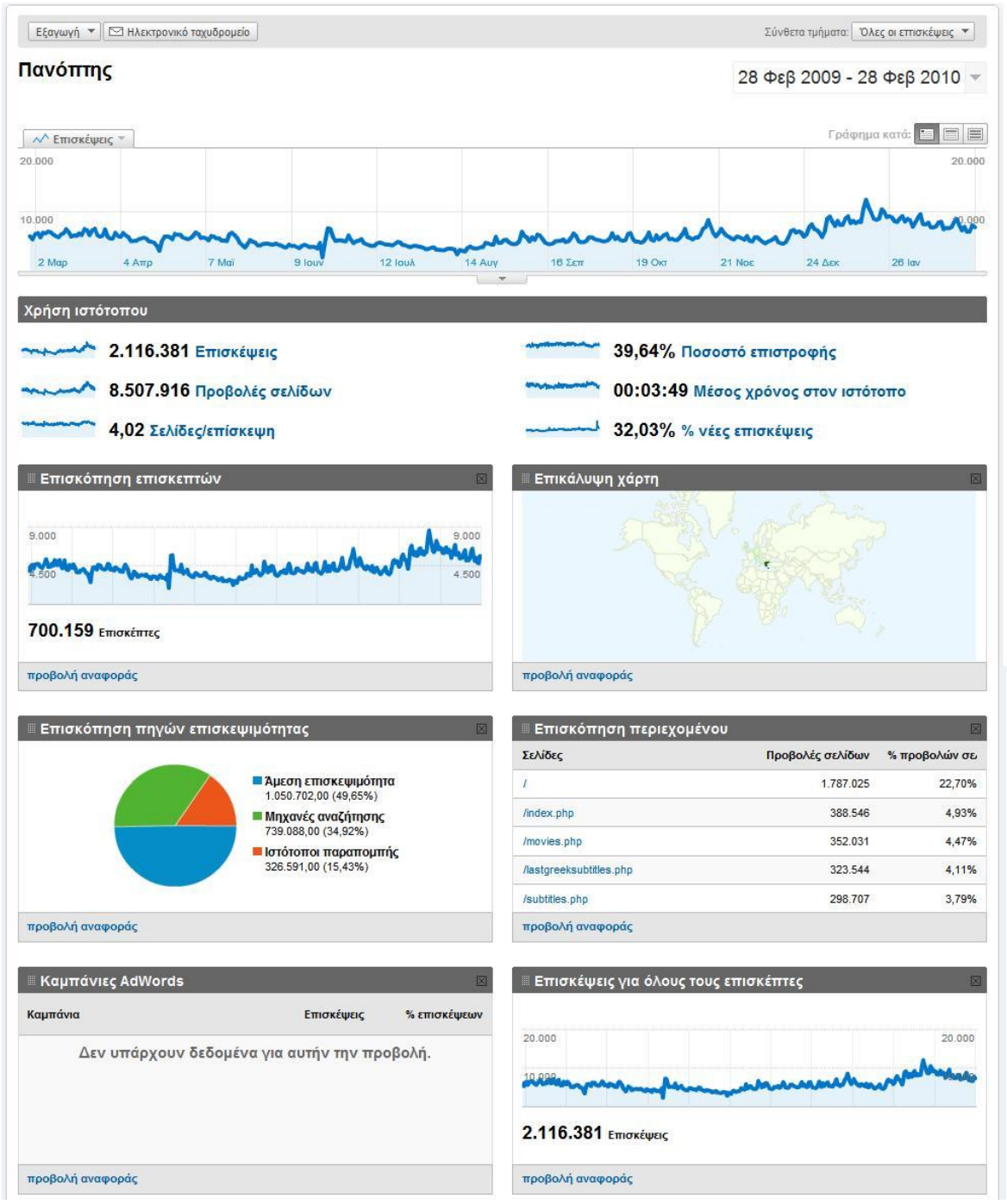
Οι χρήστες είναι αυτοί που θα παίξουν τον μεγαλύτερο ρόλο. Υπάρχουν διάφορα δημογραφικά στατιστικά τα οποία παίζουν σημαντικό ρόλο. Η ηλικία, το γένος, η εμπειρία σε υπολογιστές είναι μερικά από τα χαρακτηριστικά τα οποία πρέπει να ληφθούνε υπόψιν. Ένας ιστοχώρος ο οποίος απευθύνεται σε πολύ μικρές ηλικίες πρέπει να έχει τα κατάλληλα χρώματα, σχέδια, και δομή για να είναι εύκολα

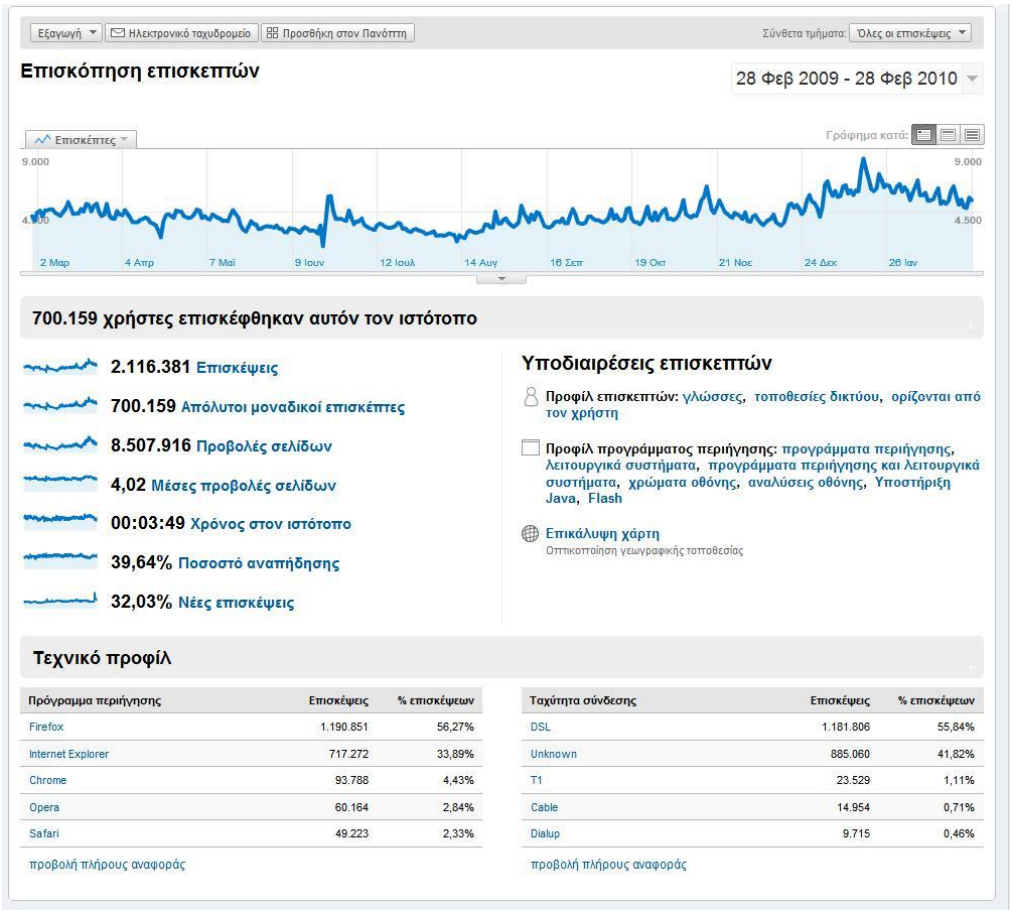
προσβάσιμο. Ένας τεχνολογικός ιστοχώρος έχει εντελώς διαφορετικό κοινό και διαφορετικές απαιτήσεις στην δομή και στα χρώματα. Επίσης η ηλικία θα μπορούσε να επηρεάσει ακόμα και το μέγεθος της γραμματοσειράς που θα χρησιμοποιηθεί. Γι' αυτό χρειάζεται να αναλυθεί το κοινό που απευθύνεται , ή να γίνει συλλογή στοιχείων από το υπάρχων site έτσι ώστε στον επανασχεδιασμό του να τα ληφθούνε υπόψιν.

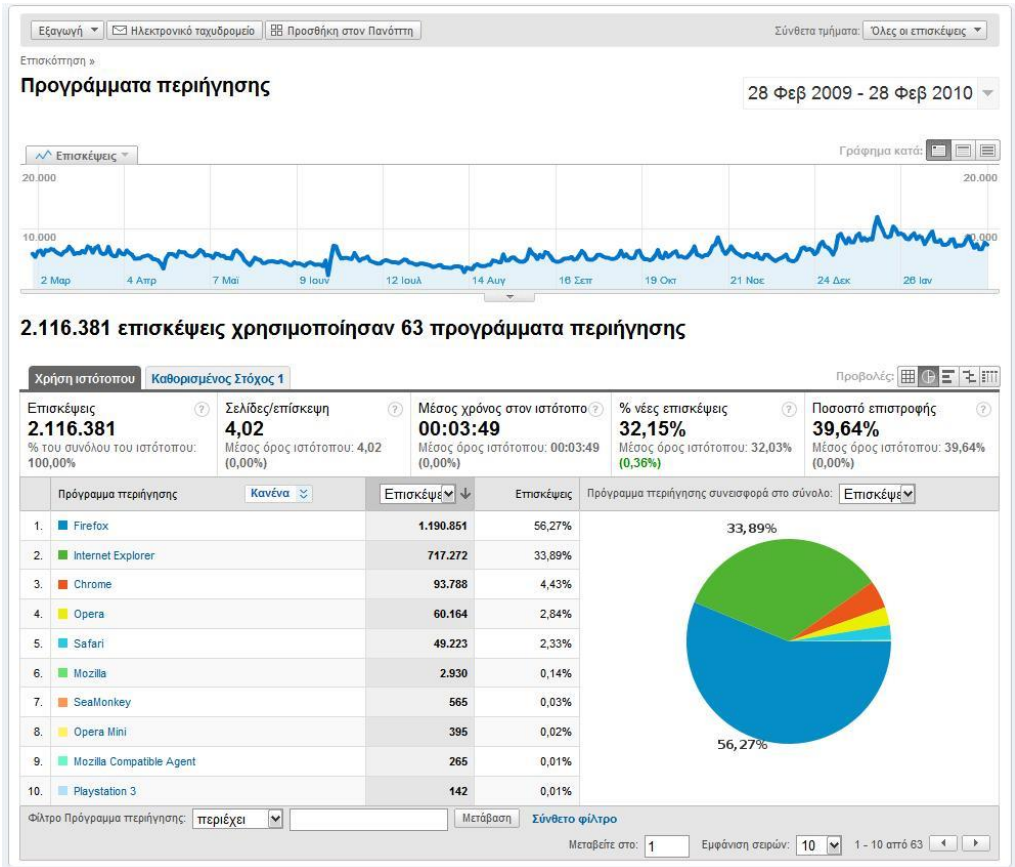
Στην δική μας περίπτωση τα στατιστικά δείχνουν:

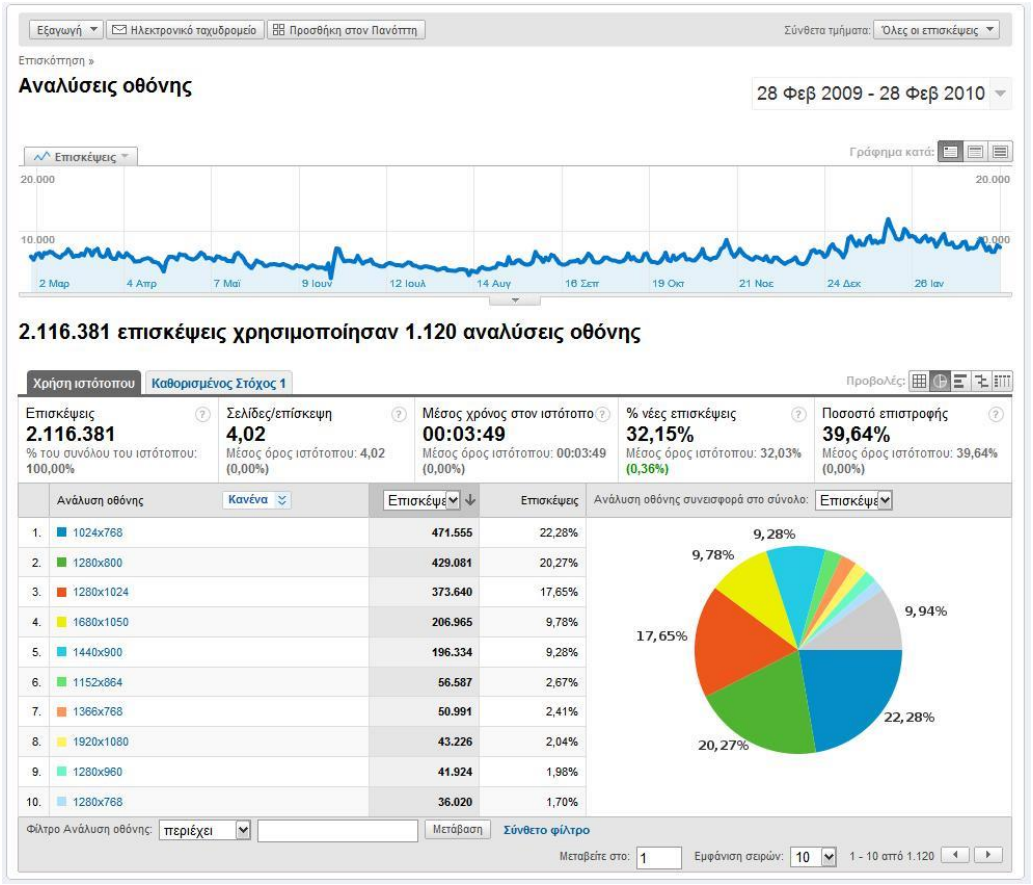


	Άντρας	Γυναίκα	
	64%	33%	
13-17	4%	3%	7%
18-24	15%	11%	26%
25-34	26%	12%	38%
35-44	14%	5%	19%
45-54	3%	1%	4%
55+	1%	1%	2%


Εξαγωγή
Ηλεκτρονικό ταχυδρομείο
Σύνθετα τμήματα: Όλες οι επισκέψεις
Πανόπτης
28 Φεβ 2009 - 28 Φεβ 2010
Επισκέψεις
Γράφημα κατά:
20.000
10.000
2 Μαρ
4 Απρ
7 Μαϊ
9 Ιουν
12 Ιουλ
14 Αυγ
16 Σεπ
19 Οκτ
21 Νοε
24 Δεκ
26 Ιαν
Χρήση ιστότοπου
2.116.381 Επισκέψεις
8.507.916 Προβολές σελίδων
4,02 Σελίδες/επίσκεψη
39,64% Ποσοστό επιστροφής
00:03:49 Μέσος χρόνος στον ιστότοπο
32,03% % νέες επισκέψεις
Επισκόπηση επισκεπτών
9.000
4.500
700.159 Επισκέπτες
προβολή αναφοράς
Επικάλυψη χάρτη
προβολή αναφοράς
Επισκόπηση πηγών επισκεψιμότητας
Άμεση επισκεψιμότητα
1.050.702,00 (49,65%)
Μηχανές αναζήτησης
739.088,00 (34,92%)
Ιστότοποι παραπομπής
326.591,00 (15,43%)
προβολή αναφοράς
Επισκόπηση περιεχομένου
Σελίδες
Προβολές σελίδων
% προβολών σε
/ 1.787.025 22,70%
/index.php 388.546 4,93%
/movies.php 352.031 4,47%
/lastgreeksubtitles.php 323.544 4,11%
/subtitles.php 298.707 3,79%
προβολή αναφοράς
Καμπάνιες AdWords
Καμπάνια
Επισκέψεις
% επισκέψεων
Δεν υπάρχουν δεδομένα για αυτήν την προβολή.
προβολή αναφοράς
Επισκέψεις για όλους τους επισκέπτες
20.000
2.116.381 Επισκέψεις
προβολή αναφοράς


Εξαγωγή
Ηλεκτρονικό ταχυδρομείο
Προσθήκη στον Πανόπτη
Σύνθετα τμήματα: Όλες οι επισκέψεις
Επισκόπηση επισκεπτών
28 Φεβ 2009 - 28 Φεβ 2010
Επισκέπτες
Γράφημα κατά:
9.000
4.500
2 Μαρ
4 Απρ
7 Μαϊ
9 Ιουν
12 Ιουλ
14 Αυγ
16 Σεπ
19 Οκτ
21 Νοε
24 Δεκ
26 Ιαν
700.159 χρήστες επισκέφθηκαν αυτόν τον ιστότοπο
2.116.381 Επισκέψεις
700.159 Απόλυτοι μοναδικοί επισκέπτες
8.507.916 Προβολές σελίδων
4,02 Μέσες προβολές σελίδων
00:03:49 Χρόνος στον ιστότοπο
39,64% Ποσοστό αναπήδησης
32,03% Νέες επισκέψεις
Υποδιαιρέσεις επισκεπτών
Προφίλ επισκεπτών: γλώσσες, τοποθεσίες δικτύου, ορίζονται από τον χρήστη
Προφίλ προγράμματος περιήγησης: προγράμματα περιήγησης, λειτουργικά συστήματα, προγράμματα περιήγησης και λειτουργικά συστήματα, χρώματα οθόνης, αναλύσεις οθόνης, Υποστήριξη Java, Flash
Επικάλυψη χάρτη
Οπτικοποίηση γεωγραφικής τοποθεσίας
Τεχνικό προφίλ
Πρόγραμμα περιήγησης | Επισκέψεις | % επισκέψεων
Firefox | 1.190.851 | 56,27%
Internet Explorer | 717.272 | 33,89%
Chrome | 93.788 | 4,43%
Opera | 60.164 | 2,84%
Safari | 49.223 | 2,33%
προβολή πλήρους αναφοράς
Ταχύτητα σύνδεσης | Επισκέψεις | % επισκέψεων
DSL | 1.181.806 | 55,84%
Unknown | 885.060 | 41,82%
T1 | 23.529 | 1,11%
Cable | 14.954 | 0,71%
Dialup | 9.715 | 0,46%
προβολή πλήρους αναφοράς


Εξαγωγή
Ηλεκτρονικό ταχυδρομείο
Προσθήκη στον Πανόπτη
Σύνθετα τμήματα: Όλες οι επισκέψεις
Επισκόπηση »
Προγράμματα περιήγησης
28 Φεβ 2009 - 28 Φεβ 2010
Επισκέψεις
Γράφημα κατά:
20.000
10.000
2 Μαρ
4 Απρ
7 Μαϊ
9 Ιουν
12 Ιουλ
14 Αυγ
16 Σεπ
19 Οκτ
21 Νοε
24 Δεκ
26 Ιαν
2.116.381 επισκέψεις χρησιμοποίησαν 63 προγράμματα περιήγησης
Χρήση ιστότοπου
Καθορισμένος Στόχος 1
Προβολές:
Επισκέψεις 2.116.381 % του συνόλου του ιστότοπου: 100,00%
Σελίδες/επίσκεψη 4,02 Μέσος όρος ιστότοπου: 4,02 (0,00%)
Μέσος χρόνος στον ιστότοπο 00:03:49 Μέσος όρος ιστότοπου: 00:03:49 (0,00%)
% νέες επισκέψεις 32,15% Μέσος όρος ιστότοπου: 32,03% (0,36%)
Ποσοστό επιστροφής 39,64% Μέσος όρος ιστότοπου: 39,64% (0,00%)
Πρόγραμμα περιήγησης | Κανένα | Επισκέψεις | Επισκέψεις | Πρόγραμμα περιήγησης συνεισφορά στο σύνολο: Επισκέψεις
1. Firefox 1.190.851 56,27%
2. Internet Explorer 717.272 33,89%
3. Chrome 93.788 4,43%
4. Opera 60.164 2,84%
5. Safari 49.223 2,33%
6. Mozilla 2.930 0,14%
7. SeaMonkey 565 0,03%
8. Opera Mini 395 0,02%
9. Mozilla Compatible Agent 265 0,01%
10. Playstation 3 142 0,01%
33,89%
56,27%
Φίλτρο Πρόγραμμα περιήγησης: περιέχει
Μετάβαση
Σύνθετο φίλτρο
Μεταβείτε στο: 1
Εμφάνιση σειρών: 10
1 - 10 από 63



- Η χρήση firefox υποδηλώνει χρήστες με κάποια παραπάνω εμπειρία.

- Η χρήση DSL αφήνει τα περιθώρια ανοιχτά για ένα ιστοχώρο το οποίο θα έχει γραφικά

- Η ανάλυση της οθόνης μας δείχνει το μέγιστο πλάτος σχεδιασμού ενός ιστοχώρου έτσι ώστε να εξυπηρετεί όσο το δυνατόν μεγαλύτερο κοινό. Ο ιστοχώρος που θα κατασκευαστεί θα έχει πλάτος 980. Λίγα pixels λιγότερα από το 1024 αφού μπορεί να έχει μπάρα ολίσθησης που «τρώει» κάποια pixels

Προσδιορισμός Αναγνωριστικού

Πολλοί web designers κάνουν υπερβολές σε θέματα design προσπαθώντας με υλοποιήσουν τις τελευταίες τάσεις της μόδας χωρίς όμως πρώτα να σκεφτούνε αν όλα αυτά είναι υπερβολικά και αν κουράζουν τον επισκέπτη.

Το χρώμα είναι κάτι πολύ σημαντικό. Πρέπει να δημιουργούν τα ανάλογα συναισθήματα στον επισκέπτη. Επίσης το χρώμα πρέπει να ταιριάζει στο χαρακτήρα του ιστοχώρου.

Μετρήσεις Απόδοσης

Όταν τελικά έχει σχεδιαστεί και υλοποιηθεί το site, είναι η στιγμή όπου πρέπει να παρθούν κάποιες μετρήσεις. Είναι πολύ σημαντικό όσο και τα πρώτα 2 βήματα επειδή μέχρι τώρα δεν υπάρχουν αποδείξεις για το αν όντως αυτά που εφαρμόζονται έχουμε απήχηση. Είναι απλά κάποιες υποθέσεις. Οι μετρήσεις θα δείξουν πραγματικά αν επετεύχθησαν οι στόχοι.

Έχοντας θέσει κάποιους στόχους, όπως η αύξηση των εγγραφών στον ιστοχώρο, η εκτέλεση μια μέτρησης δείχνει το ποσοστό της επιτυχίας.

Τέλος πολύ σημαντικό είναι η συνεχής βελτίωση του ιστοχώρου αφού η σταθερότητα είναι κάτι το οποίο ο χρήστης σιγά σιγά το βαριέται. Θέλει αλλαγές όπου δεν θα του αλλάξουν εντελώς την λογική αλλά που απλά θα τον κάνουν να νιώθει ότι δεν παραμένει σταθερό, αφού αυτή η αίσθηση δημιουργεί την εικόνα ενός παλιού ιστοχώρου.

12. OPTIMIZATION (ΒΕΛΤΙΣΤΟΠΟΙΗΣΗ)

Optimizing hardware για την MySQL

* Αν χρειαστούν πίνακες μεγαλύτερους από > 2GB, πρέπει να σκεφτούμε για ένα 64 bit hardware όπως Alpha, Sparc ή IA64. Καθώς η MySQL χρησιμοποιεί αρκετούς 64 bit ακέραιους εσωτερικάγ, 64 bit CPUs θα μασ δώσει πολύ καλήτερη απόδοση.

* Για μεγάλες βάεις δεδομένων η σειρά optimization είναι RAM, Γρήγοροι Δίσκοι, CPU.

* Περισσότερη RAM μπορεί να επιταχύνει τα updates κρατώντας πολλές σελίδες στην RAM.

* Αν δεν χρησιμοποιήσουμε transaction-safe tables ή αν έχουμε μεγάλους δίσκους και θέλουμε να αποφύγουμε ελέγχους αρχείων καλή ιδέα είναι να χρησιμοποιήσουμε ένα UPS για να απενεργοποιήσουμε το σύστημα ομαλά ή αν έχουμε διακοπές ρεύματος.

* Για συστήματα όπου η βάση δεδομένων βρίσκεται σε dedicated server καλό είναι να χρησιμοποιηθεί 1G Ethernet.

Βρίσκουμε τις καλύτερες ρυθμίσεις εκκίνησης της MYSQL.

Ελέγχουμε την MySQL.

Χρησιμοποιούμε EXPLAIN SELECT, SHOW VARIABLES, SHOW STATUS and SHOW PROCESSLIST.

Μαθαίνουμε πως λειτουργεί η βελτιστοποίηση ερωτημάτων.

Κάνουμε Optimize στους τύπους των πινάκων ανάλογα με τις ανάγκες μας.

Διατηρούμε τους πίνακες μας (myisamchk, CHECK TABLE, OPTIMIZE TABLE).

Γράφουμε MySQL UDF function αν εντοπίσουμε πως χρειαζόμαστε μια συνάρτηση σε πολλά σημεία

Optimizing disks

* Καλή τεχνική είναι να υπάρχει ένας δίσκος για το σύστημα, για τα προγράμματα και τα προσωρινά αρχεία και ένας για update logs και transactions logs αν γίνονται πολλές αλλαγές..

* Μικροί χρόνοι απόκρισης είναι σημαντικό για τους δίσκους που έχουν την βάση. Για μεγάλους πίνακας μπορούμε να υπολογίσουμε ότι θα χρειαστούμε: log(row_count) / log(index_block_length/3*2/(key_length + data_ptr_length))+1 seeks για να βρούμε μια εγγραφή. Για ένα πίνακα με 500,000 rows indexing: log(500,000)/log(1024/3*2/(3+4)) +1 = 4 seeks Το παραπάνω θα χρειαστεί: 500,000 * 7 * 3/2 = 5.2M. Στην πραγματικότητα τα περισσότερα blocks θα μείνουν σε buffer και γι' αυτό πιθανόν να χρειαστούν 1-2 seeks

"row_count" = πλήθος εγγραφών

" index_block_length" = μέγεθος του index σε bytes που συνήθως είναι 1024,

"key_length" = bytes τύπου δεδομένων

"data_ptr_length" = μέγεθος data pointer σε bytes που συνήθως είναι 4

TINYINT = 1byte, SMALLINT = 2bytes, MEDIUMINT = 3bytes, INT = 4bytes, BIGINT = 8bytes

* Για εγγραφές θα χρειαστούμε όπως και πριν 4 seek requests. Ωστόσο για να βρούμε που να τοποθετήσουμε ένα νέο κλειδί, κανονικά χρειαζόμαστε 2 seeks για να αλλάξουμε και να γράψουμε το περιεχόμενο μιας εγγραφής.

* Για πραγματικά μεγάλες βάσεις δεδομένων η εφαρμογή θα χρειαστεί πολύ μεγάλες ταχύτητες δίσκου αφού η αύξηση των seek requests αυξάνετε με N log N καθώς διαβάζονται όλο και περισσότερα δεδομένα.

* Διαχωρίζουμε την βάση δεδομένων και τους πίνακες σε ξεχωριστούς δίσκους.

* Το RAID 0 θα αυξήσει τις επιδόσεις και της εγγραφή και της ανάγνωσης.

* Χωρίζοντας με RAID 0+1 (mirroring) θα έχουμε ασφαλή τα δεδομένα μας και θα αυξήσουμε την ταχύτητα ανάγνωσης αλλά η εγγραφή θα είναι ελάχιστα πιο αργή.

* Δεν χρησιμοποιούμε RAID (εκτός του 0) σε δίσκους όπου περιέχουν προσωρινά αρχεία, όπου μπορούμε να τα αναπαράγουμε εύκολα.

* Στο linux μπορούμε να χρησιμοποιήσουμε την εντολή hdparm -m16 -d1 για τους δίσκους κατά την εκκίνηση για να ενεργοποιήσουμε την ανάγνωση/εγγραφή σε πολλαπλά sectors την ίδια στιγμή. Αυτό ίσως αυξήσει την απόδοση κατά 5-50 %.

* Για κάποιες πολύ ιδιαίτερες εφαρμογές ίσως να χρειαστούμε να δεσμεύσουμε χώρο στην RAM για κάποιους πίνακες, αλλά συνήθως αυτό δεν χρειάζεται.

Optimizing OS

* Αν υπάρχουν προβλήματα με την ποσότητα της μνήμης είναι προτιμότερο να αυξήσουμε την μνήμη παρά να προσπαθήσουμε να κάνουμε το σύστημα να χρησιμοποιεί λιγότερη/
* Μην χρησιμοποιείτε NFS δίσκους για δεδομένα/
* Αυξήστε την ρύθμιση για ανοιχτά αρχεία στο σύστημα και για τον SQL Server.
* Αυξήστε τον αριθμό των διεργασιών και των threads για το σύστημα.
* Αν έχετε λίγους μεγάλους πίνακες, κάντε το σύστημα σας να μην σπάει το αρχείο σε διαφορετικούς τομείς (Solaris).
* Χρησιμοποιείστε σύστημα αρχείων που υποστηρίζει μεγάλα αρχεία.
* Διαλέξτε ποιο σύστημα αρχείων θα χρησιμοποιήσετε. Το Reiserfs στα Linux είναι πολύ γρήγορο στο άνοιγμα αρχείων, διάβασμα και εγγραφή αλλά ο έλεγχος αρχείων θέλει κάποια δευτερόλεπτα.

Για περισσότερη ταχύτητα:

* CPU, disk, memory, SQL server, OS, API.
* Χρησιμοποιείστε προεκτάσεις για βελτίωση ταχύτητας και ευελιξία.
* Βέλτιστα ερωτήματα και σχήμα πινάκων
* Αν έχετε χαμηλή ταχύτητα σύνδεσης με την βάση δεδομένων χρησιμοποιείστε client/server με πρωτόκολλο συμπίεσης .

Optimizing MySQL

* Βρείτε τις καλύτερες ρυθμίσεις εκκίνησης για την MySQL.
* Χρησιμοποιείστε EXPLAIN SELECT, SHOW VARIABLES, SHOW STATUS και SHOW PROCESSLIST.
* Μαθαίνουμε πως να βελτιώνουμε τα ερωτήματα μας

* Διαλέγουμε σωστό τύπο πίνακα.

* Συντηρούμε τους πίνακες μας (myisamchk, CHECK TABLE, OPTIMIZE TABLE).

* Χρησιμοποιούμε διάφορα plugins για περισσότερη ταχύτητα

* Γράφουμε MySQL UDF function αν εντοπίσουμε πως χρειαζόμαστε μια συνάρτηση σε πολλά σημεία

Συντήρηση

* Αν είναι δυνατόν να εκτελείτε συχνά η εντολή OPTIMIZE για τους πίνακες. Αυτό είναι ιδιαίτερα σημαντικό σε πίνακες με εγγραφές που αλλάζουν αρκετά. Αν τα δεδομένα αλλάζουν συχνά πρέπει να γίνετε βελτιστοποίηση έτσι ώστε οι εγγραφές να τοποθετούνται στα σωστά σημεία, σύμφωνα με τα κλειδιά, έτσι ώστε η ανάγνωση αυτών αργότερα να γίνετε ταχύτερα.

* Αν υπάρχουν κατακερματισμένα αρχεία, καλό είναι να αντιγραφούν όλα τα αρχεία σε έναν άλλο δίσκο, να καθαριστεί, και να ξανά αντιγράψουμε τα αρχεία πίσω. Η διαδικασία αυτή επιτρέπει να τοποθετηθούν ξανά από την αρχή με σωστή πλέον θέση τα δεδομένα χωρίς να χωρίζονται σε πολλούς δίσκους και διαφορετικά σημεία μέσα στον δίσκο.

* Αν υπάρχουν προβλήματα, ελέγξτε τους πίνακες με την εντολή myisamchk ή CHECK table.

* Ελέγχουμε την MySQL με την εντολή: mysqladmin -i10 processlist extended-status

Optimizing SQL

Χρησιμοποιούμε την SQL για αυτό που έχει φτιαχτεί, όπως

* Βρίσκουμε εγγραφές με την εντολή WHERE.

* JOIN tables

* GROUP BY

* ORDER BY
* DISTINCT

Δεν την χρησιμοποιούμε για:

* Ελέγξουμε τα δεδομένα μας
* Σαν υπολογιστικό σύστημα

Tips

* Χρησιμοποιούμε τα κλειδιά με μέτρο αφού και η αναζήτηση του σωστότερου index πίνακα απαιτεί χρόνο.
* Τα κλειδιά είναι καλά για τις αναζητήσεις, αλλά όχι για inserts / updates αφού κατακερματίζουν τους πίνακες μας.
* Δεν φοβόμαστε να έχουμε διπλότυπες πληροφορίες αν αυτό μπορεί να μας εξασφαλίσει ταχύτητα.
* Αντί να κάνουμε πολλά GROUP BYs σε έναν μεγάλο πίνακα, δημιουργούμε άλλους πίνακες με βασικά στοιχεία και κάνουμε τα ερωτήματα σε αυτούς.
* Χρησιμοποιούμε τις default τιμές για να επιταχύνουμε και να απλουστεύσουμε ένα INSERT query.

Σημαντικές αρχικές τιμές για την MySQL

back_log	Change if you do a lot of new connections.
thread_cache_size	Change if you do a lot of new connections.
key_buffer_size	Pool for index pages; Can be made very big
bdb_cache_size	Record and key cache used by BDB tables.
table_cache	Change if you have many tables or simultaneous connections
delay_key_write	Set if you need to buffer all key writes

log_slow_queries	Find queries that takes a lot of time
max_heap_table_size	Used with GROUP BY
sort_buffer	Used with ORDER BY and GROUP BY
myisam_sort_buffer_size	Used with REPAIR TABLE
join_buffer_size	When doing a join without keys

Optimizing tables

* Η MySQL έχει πολλούς και διάφορους τύπους δεδομένων. Προσπαθούμε να χρησιμοποιούμε τον καταλληλότερο τύπο για κάθε στήλη.

* Η εντολή/διαδικασία ANALYSE μπορεί να μας βοηθήσει να βρούμε τον καλύτερο τύπο για τον πίνακα.

* Χρησιμοποιούμε NOT NULL για στήλες όπου δεν έχουν null τιμές. Αυτό είναι ιδιαίτερα σημαντικό για στήλες με επιλογή index.

* Προτιμάμε το πρότυπο MyISAM από το ISAM. Το MyISAM είναι προέκταση του ISAM και παρέχει ένα πλήθος από συναρτήσεις indexing και διαχείριση πεδίων που δεν παρέχονται στο ISAM πρότυπο. Επίσης το MyISAM χρησιμοποιεί έναν μηχανισμό κλειδώματος πίνακα για να βελτιστοποιήσει τις ταυτόχρονες αναγνώσεις και εγγραφές.

* Δεν δημιουργούμε indexes όταν δεν τα χρειαζόμαστε αφού απαιτούν χώρο και χρόνο επιλογής indexed πίνακα

* Χρησιμοποιούμε το καλύτερο τύπο πίνακα για κάθε πίνακα.

* Στήλες με παρόμοιο περιεχόμενο με άλλους πίνακες, να έχουν το ίδιο όνομα σε όλους τους πίνακες για να χρησιμοποιούμε στα joins ένα όνομα και για να μην δημιουργείτε σύγχυση με τα ονόματα.

Η MySQL είναι πολύ καλή:

* Σύστημα login.

* Όταν υπάρχουν πολλές συνδέσεις. Η σύνδεση είναι πολύ γρήγορη.

* Όταν χρησιμοποιούμε SELECT και INSERT ταυτόχρονα.

* Όταν δεν συνδυάζουμε αλλαγές με επιλογές που καθυστερούν αρκετά.

* Όταν τα περισσότερα selects/updates χρησιμοποιούν μοναδικά κλειδιά/

* Όταν χρησιμοποιούμε αρκετούς πίνακες χωρίς μεγάλα και παράξενα κλειδώματα.

* Όταν έχεις πολύ μεγάλους πίνακες.

Πράγματα που αποφεύγουμε με την MySQL

* Updates ή INSERT σε πίνακες με διαγραμμένες εγγραφές, συνδυάζοντας με SELECTS που κρατάνε πολύ ώρα.

* HAVING σε στήλες όπου τις έχουμε σε WHERE

* JOINS χωρίς την χρήση κλειδιού.

* JOINS σε στήλες όπου έχουν διαφορετικό τύπο.

13. AGENTS

Ένα από τα πιο σημαντικά θέματα ενός ιστοχώρου είναι η συνεχής ενημέρωσή του κάτι το οποίο απαιτεί διαχειριστές και αρκετό χρόνο που πολλές φόρες μπορεί να μην είναι αρκετός με αποτέλεσμα ο ιστοχώρος μας να μένει ανενημέρωτο με αποτέλεσμα την απώλεια των επισκεπτών.

Για να λυθεί εν μέρη το πρόβλημα αυτό πρέπει να δημιουργηθούν αυτόματοι μηχανισμοί που θα μπορούν να ενημερώνουν τον ιστοχώρο στον βέλτιστο χρόνο. Χαρακτηριστικό παράδειγμα στο δικό μας project είναι η ενημέρωση του box office. Είναι μια διαδικασία που χειροκίνητα απαιτεί αρκετό χρόνο. Επίσης περιλαμβάνεται και η πιθανότητα σφάλματος στην πληκτρολόγηση.

Γι' αυτό το λόγο αναπτύχθηκε ένας μηχανισμός που εκμεταλλεύεται τα διεθνή box offices που υπάρχουν σε παγκόσμιους φορείς που όμως δεν τα παρέχουν σε κάποια έτοιμη εκμεταλλεύσιμη πηγή. Για παράδειγμα αν είχαμε ένα XML αρχείο που μας δίνει αυτές τις πληροφορίες θα μπορούσαμε να εξάγουμε τις πληροφορίες από αυτό και να τις εισάγουμε αυτόματα στην δική μας βάση. Στις περισσότερες όμως περιπτώσει αυτό δεν δίνεται και γι' αυτό πρέπει να φτιάξουμε έναν custom μηχανισμό που θα εξάγει μόνο τις πληροφορίες που χρειαζόμαστε.

Έτσι αναζητώντας διευθύνσεις με box office, βρέθηκε η παρακάτω σελίδα που έχει όλες τις απαραίτητες πληροφορίες

http://www.boxofficemojo.com/weekend/chart/

Έχοντας πλέον αντιγράψει το αρχείο αυτό στον δικό μας server και μετατρέποντας το σε κάποιο string έχουμε την δυνατότητα χρησιμοποιώντας τα κατάλληλα regular expressions να εξάγουμε τις πληροφορίες που θέλουμε

```
$file = 'http://www.boxofficemojo.com/weekend/chart/';
$newfile = 'boxoffice.html';if (!copy($file, $newfile)) ;
$contents = implode('', file($newfile));
```

preg_match_all('/<td align=\"center\"><font size=\"2\">(.)<\/font><\/td>.*\n<td align=\"center\"><font size=\"2\">(.*)<\/font><\/td>.*\n<td><font size=\"2\"><a href=.*><b>(.*)<\/b><\/a><\/font><\/td>.*\n<td><font size=\"2\"><a href=.*>(.*)<\/a><\/font><\/td>.*\n<td align=\"right\"><font size=\"2\"><b>(.*)<\/b><\/font><\/td>.*\n.*\n.*\n.*\n<td align=\"right\"><font size=\"2\">(.*)<\/font><\/td>.*\n<td align=\"right\"><font size=\"2\">(.*)<\/font><\/td>.*\n<td align=\"center\"><font size=\"2\">(.*)<\/font><\/td>/', $contents, $match, PREG_SET_ORDER);*

Χρησιμοποιώντας το παραπάνω regular expression καταφέρνουμε να αποσπάσουμε τα στοιχεία που βρίσκονται μέσα στις παρενθέσεις. (Θέση Box Office, προηγούμενη θέση στο Box Office, Τίτλος ταινίας, Studio παραγωγής ταινίας, Έσοδα εβδομάδας, Συνολικά έσοδα, Έξοδα, Εβδομάδες Κυκλοφορίας)

imdb.com παραδείγματα

Τίτλος : preg_match_all('/<div id=\"tn15title\">\n<h1>(.*)/', $contents, $match, PREG_SET_ORDER);
Σκηνοθέτης : preg_match_all('/<h5>Director.*:<\/h5>\n(.*)\n<\/div>/', $contents, $match, PREG_SET_ORDER);
Σενάριο : preg_match_all('/<h5>Writer.*:<\/h5>\n(.*)\n<\/div>/', $contents, $match, PREG_SET_ORDER);
Ημερομηνία Κυκλοφορίας : preg_match_all('/Release Date:<\/h5> \n([1-3][0-9]|\d)
(January|February|March|April|May|June|July|August|September|October|November|December) ([12]\d\d\d) (.*)\n/', $contents, $match, PREG_SET_ORDER);

Η τεχνική αυτή χρησιμοποιήθηκε στον ιστοχώρο έτσι ώστε να έχουμε αυτόματα το box office, και πληροφορίες ταινιών από την σελίδα imdb.com. Το πρόγραμμα της τηλεόρασης το πήραμε έτοιμο μέσω ενός rss αρχείου και κάναμε εξαγωγή των δεδομένων στην δική μας βάση μια φορά την ημέρα (06:00). http://tv.pathfinder.gr/rss.php

Τέλος υπάρχει η δυνατότητα να οριστεί μέσω ενός cron προγράμματος η αυτόματη εκτέλεση του script ανά κάποιες περιόδους έτσι ώστε πραγματικά να είναι αυτόματη η ενημέρωση του ιστοχώρου.

Οι Κανονικές Εκφράσεις (Regular Expressions)

Μια κανονική έκφραση (regular expression) είναι ένα κομμάτι (string) κειμένου που περιέχει ειδικούς κωδικούς που του δίνουν τη δυνατότητα να μπορεί να χρησιμοποιηθεί με κάποιες συναρτήσεις της PHP για εντοπισμό και διαχείριση κειμένου. Για παράδειγμα, η επόμενη είναι μια κανονική έκφραση που κάνει αναζήτηση για το κείμενο PHP : «PHP»

Για να μπορέσουμε να χρησιμοποιήσουμε μια κανονική έκφραση, πρέπει να είμαστε εξοικειωμένοι με τις συναρτήσεις των κανονικών εκφράσεων που υπάρχουν στην PHP. Η ereg() είναι η βασικότερη απ' αυτές και μπορεί να χρησιμοποιηθεί για να καθορίσουμε αν μια κανονική έκφραση ικανοποιείται από ένα συγκεκριμένο string κειμένου. Ας δούμε τον επόμενο κώδικα :

```
$text = "Κανόνες της PHP!";
if (ereg("PHP", $text)) {
echo( '$το κείμενο περιέχει το string "PHP".' );
} else {
echo( '$το κείμενο δεν περιέχει το string "PHP".' );
}
```

Σ' αυτό το παράδειγμα, η κανονική έκφραση ικανοποιείται επειδή το string που είναι αποθηκευμένο στη μεταβλητή $text περιέχει το PHP. Ο παραπάνω κώδικας θα εξάγει συνεπώς τα εξής :

$το κείμενο περιέχει το string "PHP".

Δεν πρέπει να ξεχνάμε όταν τα μονά εισαγωγικά αποτρέπουν την PHP από το να εμφανίσει την τιμή της μεταβλητής $text.

Η eregi() είναι μια συνάρτηση που συμπεριφέρεται σχεδόν παρόμοια με την ereg(), εκτός από το ότι αγνοεί τα πεζά και τα κεφαλαία γράμματα όταν αναζητά ταιριάσματα :

```
$text = "Τι είναι η Php;";
if (eregi("PHP", $text)) {
echo( '$το κείμενο περιέχει το string "PHP".' );
} else {
echo( '$το κείμενο δεν περιέχει το string "PHP".' );
}
```

Αυτό εμφανίζει πάλι το ίδιο μήνυμα :

$το κείμενο περιέχει το string "PHP".

Θα δούμε μερικά παραδείγματα για να μάθουμε τη βασική σύνταξη των κανονικών εκφράσεων. Πρώτα απ' όλα, το σύμβολο caret (^) μπορεί να χρησιμοποιηθεί για να δείξει την αρχή ενός string, ενώ το σύμβολο του δολαρίου ($) χρησιμοποιείται για να δείξει το τέλος :

```
PHP          // Ταιριάζει με το "What is PHP?"
^PHP         // Ταιριάζει με το "PHP rules!", όχι με το "What is PHP?"
PHP$         // Ταιριάζει με το "I love PHP", όχι με το "What is PHP?"
^PHP$        // Ταιριάζει με το "PHP" και τίποτα άλλο
```

Προφανώς, θα υπάρχουν φορές που θα θέλουμε να χρησιμοποιήσουμε τα σύμβολα ^, $ ή και άλλους ειδικούς χαρακτήρες για να συμπεριλάβουμε τον αντίστοιχο χαρακτήρα στο string αναζήτησης. Για να αφαιρέσουμε το ειδικό νόημα ενός χαρακτήρα, προτάσσουμε τον χαρακτήρα \ (backslash), ως εξής :

\$\$\$ // Ταιριάζει με το "Show me the $$$!"

Οι αγκύλες μπορούν να χρησιμοποιηθούν για να ορίσουν ένα σύνολο χαρακτήρων που μπορεί να ταιριάξει. Για παράδειγμα, η ακόλουθη κανονική έκφραση ταιριάζει μ' ένα μόνο από τα ψηφία 1 έως 5.

[12345] // Ταιριάζει με το "1" και το "3", αλλά όχι με το "a" ή το "12"

Μπορούμε να καθορίσουμε και περιοχές αριθμών ή γραμμάτων.

[1-5] // Το ίδιο όπως προηγουμένως
[a-z] // Ταιριάζει μ' ένα οποιοδήποτε πεζό γράμμα
[0-9a-zA-Z] // Ταιριάζει μ' ένα οποιοδήποτε γράμμα ή ψηφίο

Οι χαρακτήρες ?, + και * έχουν επίσης ειδικό νόημα. Συγκεκριμένα, το ? σημαίνει ότι ο προηγούμενος χαρακτήρας είναι προαιρετικός, το + σημαίνει έναν ή περισσότερους από τους προηγούμενους χαρακτήρες και το * σημαίνει κανέναν ή έναν από τους προηγούμενους χαρακτήρες.

bana?na // Ταιριάζει με τα "banana" και "banna",
// αλλά όχι με το "banaana"

bana+na // Ταιριάζει με τα "banana" και "banaana",
// αλλά όχι με το "banna"

bana*na // Ταιριάζει με τα "banna", "banana" και "banaaana",
// αλλά όχι με το "bnana"

Ù[a-zA-z]+$ // Ταιριάζει μ' ένα οποιοδήποτε string που έχει έναν
// τουλάχιστον χαρακτήρα

Οι παρενθέσεις μπορούν να χρησιμοποιηθούν για να ομαδοποιήσουμε strings και να εφαρμόσουμε τα ?, + ή * σ' αυτά σαν σύνολο.

ba(na)+na // Ταιριάζει με τα "banana" και "bananananana",
// αλλά όχι με το "bana" ή το "banaana"

Ακολουθούν μερικοί κωδικοί για να μπορέσουμε να ταιριάξουμε τους ειδικούς χαρακτήρες στις κανονικές εκφράσεις :

\n // Ταιριάζει μ' έναν χαρακτήρα νέας γραμμής (newline character)
Ù // Ταιριάζει μ' έναν οποιονδήποτε χαρακτήρα εκτός από τον χαρακτήρα νέας γραμμής
\r // Ταιριάζει μ' έναν χαρακτήρα carriage return
\t // Ταιριάζει με τον χαρακτήρα tab

Αντικατάσταση Strings με Κανονικές Εκφράσεις

Χρησιμοποιώντας την ereg() ή την eregi() με τη σύνταξη των κανονικών εκφράσεων που μόλις είδαμε, μπορούμε να εντοπίσουμε εύκολα την παρουσία tags σ' ένα δεδομένο string κειμένου. Αυτό που πρέπει να κάνουμε, όμως, είναι να σημειώσουμε με ακρίβεια αυτά τα tags και να τα αντικαταστήσουμε με κατάλληλα HTML tags.

Για να γίνει αυτό, πρέπει να δούμε μερικές ακόμα συναρτήσεις κανονικών εκφράσεων που υπάρχουν στην PHP : την ereg_replace() και την eregi_ replace().

Η ereg_replace(), σαν την ereg(), δέχεται μια κανονική έκφραση και ένα string κειμένου και προσπαθεί να ταιριάξει την κανονική έκφραση με το string. Επιπλέον, όμως, η ereg_replace() δέχεται ένα δεύτερο string κειμένου και αντικαθιστά κάθε ταίριασμα της κανονικής έκφρασης σ' αυτό το string. Η σύνταξη της ereg_replace() είναι ως εξής :

$newstring = ereg_replace(<regexp>, <replacewith>, <oldstring>);

όπου η <regexp> είναι η κανονική έκφραση και το <replacewith> είναι το string που θα αντικαταστήσει τα ταιριάσματα στην <regexp> όπου υπάρχει το <oldstring>. Η συνάρτηση επιστρέφει το νέο string που προκύπτει από τη

λειτουργία της αντικατάστασης. Στην παραπάνω πρόταση, αυτό αποθηκεύεται στη μεταβλητή $newstring.

Η eregi_replace(), όπως είναι αναμενόμενο, είναι παρόμοια με την ereg_replace(), εκτός από το ότι δεν ελέγχει τα πεζά/κεφαλαία όταν κάνει αναζήτηση για ταιριάσματα.

Είμαστε τώρα έτοιμοι να αρχίσουμε να δημιουργούμε τη δική μας προσαρμοσμένη γλώσσα σήμανσης (custom markup language).

Χαρακτηριστικά παραδείγματα

- Επικύρωση Email:
  /(\w[-._\w]*\w@\w[-._\w]*\w\.\w{2,3})/
- Ταχυδρομικοί Κώδικες Αυστραλίας:
  /^[0-9]{4}/
- Αριθμός Τηλεφώνου Αμερικής:
  /^(?:\([2-9]\d{2}\)\ ?|[2-9]\d{2}(?:\-?|\ ?))[2-9]\d{2}[-]?\d{4}$/
- Αριθμός Τηλεφώνου Αυστραλίας:
  /^(?:\([0]\d{1}\)\ ?|[0]\d{1}(?:\-?|\ ?))[8-9]\d{3}[-]?\d{4}$/
- Ταχυδρομικοί Κώδικες Αμερικής:
  /^[0-9]{5}([- /]?[0-9]{4})?$/
- Γερμανικές Εκφράσεις (με προαιρετικά + ή -)
  /^[1-6]{1}[\+|\-]?$/
- Επικύρωση Ηλικίας (Παράδειγμα : Ηλικία 20-99)
  /([2-9][0-9])/

14. ΒΙΒΛΙΟΓΡΑΦΙΑ

http://php.net/index.php

http://el.wikipedia.org/wiki/PHP

http://www.freestuff.gr/forums/viewtopic.php?t=19080

http://www.tizag.com

http://www.w3schools.com/php/default.asp

http://dide.flo.sch.gr/Plinet/Tutorials/Tutorials-Php-MySQL.html

http://www.mysqlvspostgresql.com