Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων
και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης
και Υπολογιστων

Διαχείριση χωροχρονικών δεδομένων από εφαρμογές κοινωνικής δικτύωσης σε κατανεμημένα περιβάλλοντα μεγάλης κλίμακας

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αλέξανδρος Α. Μπόκος

Επιβλέπων : Νεκτάριος Κοζύρης
Αναπληρωτής Καθηγητής Ε.Μ.Π.

Αθήνα, Νοέμβριος 2012

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων
και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης
και Υπολογιστων

Διαχείριση χωροχρονικών δεδομένων από εφαρμογές κοινωνικής δικτύωσης σε κατανεμημένα περιβάλλοντα μεγάλης κλίμακας

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αλέξανδρος Α. Μπόκος

Επιβλέπων : Νεκτάριος Κοζύρης
Αναπληρωτής Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 5η Νοεμβρίου 2012:

.....	.....	.....
Νεκτάριος Κοζύρης	Δημήτριος Τσουμάκος	Παναγιώτης Τσανάκας
Αναπληρωτής Καθηγητής Ε.Μ.Π	Επίκουρος Καθηγητής Ι.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Νοέμβριος 2012

.....
Αλέξανδρος Α. Μπόκος

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Η εξάπλωση των ασύρματων δικτύων σε συνδυασμό με την εξέλιξη των κινητών τηλεφώνων σε υπερσύγχρονες συσκευές με ιδιαίτερες δυνατότητες έχει προάγει την ευρεία χρήση κοινωνικών δικτύων και υπηρεσιών από κινητούς χρήστες. Στο πλαίσιο αυτό, δημιουργήθηκαν νέες υπηρεσίες, που κάνουν χρήση των δυνατοτήτων των συσκευών αυτών, χρησιμοποιώντας το γεωγραφικό στίγμα του χρήστη για να εμπλουτίσουν την εμπειρία που προσφέρουν. Τα δεδομένα τέτοιων υπηρεσιών αποτελούν σημαντική πηγή πληροφορίας για πληθώρα εφαρμογών, όπως για παράδειγμα προσωποποιημένη διαφήμιση ή εφαρμογές κυκλοφορίας .

Με τα κοινωνικά δίκτυα να συλλέγουν συνεχώς πληροφορία από εκατομμύρια χρήστες και την απαίτηση για διατήρηση των δεδομένων αυτών για μελλοντική χρήση, δημιουργήθηκε ανάγκη για όλο και μεγαλύτερους αποθηκευτικούς πόρους. Το μέχρι τότε πρότυπο της κεντρικής αποθήκευσης στάθηκε αδύναμο να προσαρμοστεί στη συνεχή αύξηση των δεδομένων που έπρεπε να αποθηκευτούν και τη θέση του πήραν σε πολλές επιχειρήσεις οι κατανεμημένες βάσεις δεδομένων. Μία από αυτές είναι και η HBase με την οποία θα ασχοληθούμε εμείς. Επίσης, τα γεωγραφικά δεδομένα που συλλέγονται από τις υπηρεσίες θέσης απαιτούν ειδικά σχήματα αποθήκευσης. Το σχεσιακό σχήμα δεν αρκεί, αφού στην περίπτωση ερωτημάτων που αφορούν την επιλογή σημείων μιας περιοχής για παράδειγμα, είναι ιδιαίτερα χρονοβόρο. Για το λόγο αυτό, τα χωροχρονικά δεδομένα συνοδεύονται με την επιπρόσθετη ανάγκη της αποδοτικής αποθήκευσης και δεικτοδότησης, ανάλογα με τα ερωτήματα που θα εφαρμοστούν επάνω τους.

Σκοπός της διπλωματικής είναι η συλλογή, αποθήκευση και δεικτοδότηση δεδομένων από υπηρεσίες θέσης σε κατανεμημένη βάση δεδομένων με κατάλληλο τρόπο, ώστε να γίνεται αποδοτικά η αναζήτησή τους. Σε αυτή την κατεύθυνση αναπτύξαμε ένα σύστημα που συλλέγει δεδομένα για τις θέσεις των χρηστών των τριών πιο δημοφιλών κοινωνικών δικτύων στην Ελλάδα, του Facebook, του Foursquare και του Twitter, και έπειτα τα αποθηκεύει σε ένα σχήμα HBase. Η δεικτοδότηση έχει επιλεχθεί έτσι ώστε να κάνει αποδοτικές τις ερωτήσεις που αφορούν ανάκτηση εγγραφών σε ένα δεδομένο γεωγραφικό τετράγωνο. Τέλος, σχεδιάσαμε μια εφαρμογή, που να κάνει χρήση της βάσης, υπολογίζοντας το συνολικό αριθμό χρηστών που βρίσκεται σε κάποια περιοχή της Αθήνας για ένα δεδομένο χρονικό διάστημα και να εμφανίζει στο χρήστη ένα χάρτη, όπου θα φαίνεται ποιοτικά η πληροφορία αυτή.

Λέξεις Κλειδιά

Υπηρεσίες βασισμένες στη θέση, Foursquare, Facebook, Twitter, χωροχρονικά δεδομένα, HBase, z-order curve, Google Maps

Abstract

The spread of wireless networks, combined with the evolution of mobile phones to sophisticated devices with special abilities, boosted the wide use of social networks and services from mobile users. New services, which can make use of these devices' abilities, have been launched, using the location of their users to add spatio-temporal information to enrich the experience they offer. Data from these services are a useful source of information for a variety of applications, such as personalized advertisment or traffic applications.

Social networks continuously collect informations from millions of users, demanding to hold these data for future use, so the need for bigger save resources became bigger. Central storing could not adjust to the constant growth fo data, which had to be stored and it was replaced with distributed databases. One popular distributed database is HBase, which we will use in this thesis. Also, spatial data which are collected from the location based services demand special store schemas. Relation data model is not enough, since it is extremely slow in queries that ask for tupples within a regional area for example. For this reason, spatio-temporal data come along with the extra need of special store and indexing schemas, which will most efficiently fit to the queries' needs.

The purpose of this thesis is the collection, storing and indexing of data from location based services in a distributed database, in a way that will make queries over them more efficient. We have developed a system which collects data from the three most popular location based services in Greece – Foursquare, Facebook and Twitter – and then stores them in an HBase schema. A special indexing is used in order to make range queries more efficient. Finally, we developed an application which makes use of this database and calculates the total number of users who are inside an area for a specific range of time, providing a map which will better show this information.

KeyWords

Location based services, Foursquare, Facebook, Twitter, spatio-temporal data, HBase, z-order curve, Google Maps

Ευχαριστώ την οικογένειά μου που με στήριξε, την Κατερίνα Δόκα που με βοήθησε πρόθυμα σε όλη την πορεία της διπλωματικής και τον Ν. Κοζύρη που μου έδωσε τη δυνατότητα να ασχοληθώ με το θέμα...

Κατάλογος περιεχομένων

Κατάλογος εικόνων

Ευρετήριο πινάκων

Κεφάλαιο 1

Εισαγωγή

Τα τελευταία χρόνια παρατηρούμε μια μεγάλη αύξηση της χρήσης του διαδικτύου, από άτομα που ανήκουν σε μία ευρεία γκάμα ηλικιών. Αυτό έγινε ευκολότερο αρχικά με τις μόνιμες ADSL συνδέσεις, που πλέον προσφέρονται σε λογικό οικονομικό κόστος, και αργότερα από τα όλο και περισσότερα σημεία ασύρματης δικτύωσης. Ταυτόχρονα, είδαμε την έξαρση των υπηρεσιών κοινωνικής δικτύωσης, που απέκτησαν γρήγορα μεγάλο αριθμό χρηστών και συνεχώς επεκτείνονται φέρνοντας σε επαφή ανθρώπους από διάφορα πλάτη και μήκη του κόσμου. Στο πλαίσιο αυτό, η ανάπτυξη της τεχνολογίας και η δημιουργία και πλασάρισμα στην αγορά κινητών συσκευών, που ξεφεύγουν από τα πλαίσια των συμβατικών τηλεφώνων και προσφέρουν ισχυρές δυνατότητες επεξεργασίας, εκτέλεσης εφαρμογών και σύνδεσης στο διαδίκτυο, έφεραν στο παιχνίδι της κοινωνικής δικτύωσης μία ακόμα μεταβλητή: αυτή της γεωγραφικής θέσης.

Υπηρεσίες που επιτρέπουν στους χρήστες τους να δηλώσουν τη θέση τους και να έρθουν έτσι σε επαφή με φίλους τους, ή να κερδίσουν κάποια ανταλλάγματα από την επιχείρηση στην οποία βρίσκονται γίνονται όλο και πιο δημοφιλείς. Οι υπηρεσίες αυτές λέγονται υπηρεσίες βασισμένες στη θέση (location based services ή LBS) και έχουν γίνει γενικότερο θέμα συζήτησης και έρευνας. Είναι ιδιαίτερα σημαντικές, αφού προσφέρουν πολύτιμες πληροφορίες που μπορούν να εκμεταλλευτούν από διάφορες εφαρμογές. Για να γίνει όμως κάτι τέτοιο εφικτό, είναι απαραίτητο τα δεδομένα των υπηρεσιών αυτών να συλλεχθούν, αποθηκευτούν και δεικτοδοτηθούν με κατάλληλο τρόπο, δεδομένου του όγκου τους και της ιδιαιτερότητας που εισάγει η χωροχρονική τους ταυτότητα.

Στα πλαίσια της διπλωματικής αυτής θα εξετάσουμε τη δυνατότητα συλλογής δεδομένων από υπηρεσίες θέσης, καθώς και τρόπους αποθήκευσης και δεικτοδότησης των δεδομένων αυτών. Συγκεκριμένα, θα αναφερθούμε στις πιο δημοφιλείς LBS, θα δούμε πώς οργανώνει και προσφέρει η καθεμία τα χωροχρονικά δεδομένα των χρηστών της και θα αναπτύξουμε έναν crawler που συλλέγει πληροφορία από αυτές ανά μία ώρα. Η συλλογή θα γίνεται στα πλαίσια της Αθήνας και θα αφορά τη συσσώρευση κόσμου στα διάφορα σημεία της πόλης και τη δυνατότητα ανάκτησης του αριθμού των χρηστών που βρίσκονται σε μια περιοχή - ή βρισκόταν για κάποιο χρονικό διάστημα. Τα δεδομένα θα αποθηκεύονται σε μία κατανεμημένη βάση, ενώ θα χρησιμοποιηθούν ειδικές τεχνικές για τη δεικτοδότηση και αποδοτική ανάκτησή τους. Τέλος, θα αναπτυχθεί μια εφαρμογή, σαν use case της πλατφόρμας συλλογής, αποθήκευσης και δεικτοδότησης δεδομένων, που θα κάνει χρήση τους, αξιοποιώντας μια βιβλιοθήκη δημιουργίας χαρτών.

1.1 Κίνητρο για την ενασχόληση με το θέμα

Κίνητρο για τη συγγραφή της παρούσας διπλωματικής ήταν η παρουσία μεγάλης ποσότητας πληροφορίας, που προέρχεται από τις υπηρεσίες βασισμένες στη θέση και μπορεί κανείς να εκμεταλλευτεί για να εξάγει συμπεράσματα. Οι υπηρεσίες αυτές βασίζονται στην ιδέα, ότι οι χρήστες μπορούν να κάνουν χρήση των εξελιγμένων κινητών συσκευών (smartphones), ώστε να συνδεθούν στο διαδίκτυο και με τη χρήση κάποιας ειδικής εφαρμογής

ή της σελίδας της υπηρεσίας να δηλώσουν τη θέση τους. Πρόκειται για υπηρεσίες που ανήκουν στο πλαίσιο των κοινωνικών δικτύων, εισάγοντας όμως επιπλέον τη γεωγραφική θέση του χρήστη, σαν πληροφορία που μπορεί να γνωστοποιηθεί.

Η πρώτη LBS που έγινε ευρέως γνωστή ήταν το Foursquare[1]. Το Foursquare βασίστηκε στην ιδέα ότι οι χρήστες του μπορούν να δηλώνουν τη θέση τους σε κάποια επιχείρηση ή γενικότερα τοποθεσία, κάνοντας το λεγόμενο check-in. Αυτό κοινοποιείται στους "φίλους" τους στα πλαίσια της υπηρεσίας, ώστε να μπορέσουν ενδεχομένως να έρθουν σε επαφή ή και να συναντηθούν. Επίσης το σύστημα των check-in συνδέθηκε με ένα σύστημα ανταμοιβών, σύμφωνα με το οποίο όσο περισσότερες φορές χρησιμοποιήσει την υπηρεσία ένας χρήστης, τόσο περισσότερους πόντους θα συγκεντρώσει και όταν έχει αρκετούς μπορεί να τους εξαργυρώσει σαν κάποια έκπτωση ή δωρεάν υπηρεσία από την επιχείρηση στην οποία βρίσκεται. Το Foursquare έγινε γρήγορα δημοφιλές και σήμερα απαριθμεί 20 εκατομμύρια εγγεγραμμένους χρήστες. Έχοντας συλλέξει για αρκετά χρόνια μεγάλη ποσότητα χωροχρονικής πληροφορίας, αυτή τη στιγμή προσανατολίζεται στην αξιοποίηση των δεδομένων αυτών για εξατομίκευση των υπηρεσιών που προσφέρει στον κάθε χρήστη και θα γίνεται μέσω προτάσεων που πιθανώς να τον ενδιαφέρον σύμφωνα με το ιστορικό του.

Βλέποντας την επιτυχία του Foursquare το Facebook[2], που είχε γίνει ήδη εκείνη τη στιγμή το μεγαλύτερο σε αριθμό χρηστών κοινωνικό δίκτυο, αποφάσισε να προσφέρει και αυτό μια παρόμοια υπηρεσία, που την ονόμασε Places και έδινε στους χρήστες του τη δυνατότητα check-in όταν βρισκόταν σε κάποια από τις εγγεγραμμένες επιχειρήσεις. Η προσπάθεια δεν πήγε τόσο καλά όσο αναμενόταν και σύντομα εγκαταλείφθηκε. Ωστόσο, η σελίδα δεν εγκατέλειψε το project και τώρα βρίσκεται ενσωματωμένο μέσα στις γενικότερες υπηρεσίες του Facebook, επιτρέποντας στους χρήστες να κάνουν αναρτήσεις, που συνδέονται άμεσα με το μέρος στο οποίο βρίσκονται, μαρκάροντας μάλιστα και "φίλους" τους, που βρίσκονται μαζί τους. Αυτή τη στιγμή, η υπηρεσία σχεδιάζει να πρωτοτυπήσει προσφέροντας μια εφαρμογή, στην οποία το γεωγραφικό στίγμα του χρήστη θα παρακολουθείται συνεχώς στο παρασκήνιο και, εξετάζοντας το ιστορικό και της προτιμήσεις του, όταν βρεθεί σε κοντινό χώρο με κάποιον χρήστη που πιθανώς θα ταίριαζε θα του εμφανίζει μια ειδοποίηση. Με τον τρόπο αυτό το Facebook ελπίζει να πετύχει αυθόρμητες συναντήσεις μεταξύ χρηστών, ενώ η σημασία που δίνει στη χωροχρονική πληροφορία είναι εμφανής.

Η τελευταία υπηρεσία που ασχοληθήκαμε είναι το Twitter[3], το οποίο αυτοαποκαλείται υπηρεσία microblogging και επιτρέπει στους χρήστες του να δημοσιοποιούν μηνύματα με μέγιστο μέγεθος 140 χαρακτήρων (tweets), αλλά και να ακολουθούν (follow) τα tweets των χρηστών με τους οποίους είναι συνδεδεμένοι. Αυτή τη στιγμή απαριθμεί το εντυπωσιακό νούμερο των 200 εκατομμυρίων εγγεγραμμένων χρηστών. Από το 2010 πρόσφερε και αυτό τη δυνατότητα μαρκαρίσματος της γεωγραφικής θέσης του χρήστη, δίνοντάς του την επιλογή αν θα ήθελε να εμφανίζεται το στίγμα του κάθε φορά που θα αναρτεί μια δημοσίευση. Το ότι στο Twitter δε χρησιμοποιείται το σύστημα των check-in αλλά η χωροχρονική πληροφορία δένεται με tweets, της παρέχει ένα επιπλέον εννοιολογικό περιεχόμενο που στερούνται οι δύο προηγούμενες LBS. Για παράδειγμα με ένα check-in σε ένα σημείο της Κηφισίας μπορούμε να ξέρουμε πως ένας χρήστης πιθανόν να πίνει καφέ εκεί. Ένα tweet όμως με κείμενο "Έχει τρομερή κίνηση σήμερα" που αναρτάται σε ένα ύψος του ίδιου δρόμου μας δίνει την επιπλέον πληροφορία της κατάστασης που επικρατεί στο σημείο αυτό. Παρά τις μεγάλες δυνατότητες του γνωρίσματος αυτού δεν έχει γίνει ακόμα δημοφιλές ανάμεσα στους χρήστες, αφού το συντριπτικό ποσοστό τους το κρατούν απενεργοποιημένο. Μένει να δούμε στο μέλλον ωστόσο το ρόλο που μπορεί να παίξει στο χώρο των LBS.

Ανεξάρτητα από το πώς θα κινηθούν οι υπηρεσίες που ασχολούνται με τη χωροχρονική πληροφορία στο μέλλον ένα είναι σίγουρο: ήρθαν εδώ για να μείνουν. Και καθώς η χρήση τους όλο και αυξάνεται, τόσο περισσότερο θα αυξάνονται και τα δεδομένα που σχετίζονται με

τους χρήστες τους και δένουν μια ανάρτηση ή ένα check-in τους με μία γεωγραφική θέση. Τα δεδομένα αυτά μάλιστα είναι σημαντικό να μην χάνονται μετά από κάποιο σύντομο χρονικό διάστημα αλλά να διατηρούνται όσο περισσότερο γίνεται αποθηκευμένα, ώστε η υπηρεσία να μπορεί να εξατομικεύσει τις προτάσεις της για τον κάθε χρήστη ανάλογα με το ιστορικό και τις προτιμήσεις του. Αυτό φυσικά δημιουργεί προβλήματα για τα κλασσικά συστήματα διαχείρισης βάσεων δεδομένων, που στηρίζονται στο σχεσιακό μοντέλο και την κεντρική αποθήκευση, αφού θα χρειαζόταν τεράστια αποθηκευτικά μέσα και εξίσου μεγάλη υπολογιστική ισχύς για τη διαχείρισή των δεδομένων. Τη λύση στο παραπάνω πρόβλημα έρχονται να δώσουν οι κατανεμημένες βάσεις, που συνδυάζουν τα πλεονεκτήματα των κατανεμημένων συστημάτων και των βάσεων δεδομένων. Τώρα αντί για ένα ισχυρό κεντρικό σύστημα, έχουμε ένα μεγάλο αριθμό υπολογιστών εμπορίου, οι οποίοι είναι συνδεδεμένοι μεταξύ τους και συνδυάζονται για τη διάσπαση του αρχικού προβλήματος σε άλλα μικρότερα και την αποδοτικότερη αντιμετώπισή τους.

Οι κατανεμημένες βάσεις συνοδεύονται με μια σειρά από πλεονεκτήματα. Προσφέρουν στο χρήστη διαφάνεια, δηλαδή του αποκρύπτουν πώς είναι κατανεμημένα τα δεδομένα παρουσιάζοντας τη βάση όπως θα ήταν σε ένα κεντρικό σύστημα. Προσφέρουν αντοχή στα λάθη με ένα σύστημα διατήρησης αντιγράφων των εγγραφών, που κάνει δυνατή τη λειτουργία της κατανεμημένης βάσης ακόμα και στην περίπτωση που πέσει κάποιος κόμβος της. Μπορούν να διατηρούν κόμβους σε διαφορετικές γεωγραφικές περιοχές, μακρινές μεταξύ τους, και με το σωστό διαχωρισμό των δεδομένων που αποθηκεύονται στην κάθε μία να οδηγούν σε αποδοτικότερη απάντηση των ερωτήσεων. Τέλος, ένα γνώρισμά τους που είναι ιδιαίτερα χρήσιμο σε εφαρμογές που μόλις ξεκινάν και δεν μπορούν να ξέρουν πόσους χρήστες θα έχουν ή άλλες για τις οποίες οι απαιτήσεις αποθήκευσης δεδομένων αυξήθηκαν κατά την πορεία του χρόνου είναι η εύκολη επέκταση: σε ένα κατανεμημένο περιβάλλον η επέκταση του συστήματος για την προσθήκη περισσότερων δεδομένων είναι πλέον εύκολη υπόθεση.

Ακόμα και αν λυθεί όμως το πρόβλημα πόρων για την αποθήκευση χωροχρονικών δεδομένων το σχήμα αποθήκευσης που θα χρησιμοποιηθεί δεν αποτελεί τετριμμένη διαδικασία. Η παράμετρος του χρόνου συνδέει το χρήστη των υπηρεσιών με διαφορετικές καταστάσεις στην πάροδο του χρόνου. Έτσι μπορεί η πρόταση “Ο χρήστης Κ βρίσκεται στο μέρος Μ” να είναι αληθής τη στιγμή t_1, να γίνεται όμως ψευδής τη χρονική στιγμή t_2. Βλέπουμε λοιπόν πως είναι απαραίτητη η εισαγωγή της χρονικής πληροφορίας στη βάση και ένας τρόπος ερωτημάτων με βάση αυτή. Επίσης, από τη στιγμή που ασχολούμαστε με γεωγραφικά δεδομένα, είναι φυσικό να αποθηκεύσουμε σε έναν πίνακα όλα τα σημεία τα οποία μας ενδιαφέρουν. Έχουμε οπότε εγγραφές που πρέπει να δεικτοδοτηθούν με δύο μεταβλητές, το γεωγραφικό πλάτος και μήκος τους, ενώ η βάση πρέπει να είναι σχεδιασμένη έτσι ώστε να απαντά σε ερωτήματα όπως τα σημεία που ανήκουν σε μια περιοχή γρήγορα. Για το σκοπό αυτό τα “κλασσικά” συστήματα δεικτοδότησης αποδεικνύονται αναποτελεσματικά και χρειάζεται σχεδιασμός ειδικών βάσεων για το σκοπό αυτό.

Βλέπουμε λοιπόν, πως ξεκινώντας από την περιγραφή των υπηρεσιών βασισμένων στη θέση προέκυψαν μια σειρά από θέματα, που αφορούν την αποθήκευση των χωροχρονικών δεδομένων που οι υπηρεσίες αυτές συλλέγουν, τους πόρους που μπορεί να χρειαστούν για το σκοπό αυτό, την αποδοτική δεικτοδότησή τους και εκτέλεση ερωτημάτων πάνω τους. Στα πλαίσια της διπλωματικής θα δώσουμε λύση στα παραπάνω, υλοποιώντας μια πλατφόρμα συλλογής, αποθήκευσης και δεικτοδότησης χωροχρονικών δεδομένων, που προέρχονται από LBS, σχεδιασμένη έτσι ώστε να απαντάει σε ερωτήματα γεωγραφικού εύρους σε μικρό χρόνο.

1.2 Συλλογή, αποθήκευση και δεικτοδότηση δεδομένων από LBS

Η βασική ιδέα γύρω από την οποία θα κινηθεί η διπλωματική είναι η δημιουργία μιας κατανεμημένης πλατφόρμας στην οποία θα συγκεντρώνονται χωροχρονικά δεδομένα, χρησιμοποιώντας τις υπηρεσίες Foursquare, Facebook και Twitter, σχετικά με τους χρήστες τους στην Αθήνα. Τα δεδομένα αυτά θα πρέπει να αποθηκεύονται και να δεικτοδοτούνται αποδοτικά, ώστε αργότερα να προσπελαστούν από εφαρμογές που θα μπορούν να εκμεταλλευτούν την πληροφορία αυτή και να προσφέρουν χρήσιμα συμπεράσματα σχετικά με την κατανομή του κόσμου στις διάφορες περιοχές της πόλης ή τον αριθμό των χρηστών που υπάρχουν σε μια επιλεγμένη περιοχή.

Η δημιουργία της βάσης απαιτεί τη συλλογή δεδομένων από τις τρεις υπηρεσίες που μας ενδιαφέρουν και συνεπάγεται τη δημιουργία ενός crawler. Για την ανάπτυξή του χρησιμοποιήσαμε τα API[4][5][6] που μας προσφέρουν οι location based services και εφαρμόσαμε αλγορίθμους που να μαζεύουν όσο το δυνατόν περισσότερα δεδομένα, όσο το δυνατόν γρηγορότερα. Φροντίσαμε να πάρουμε στοιχεία για γεωγραφικά σημεία σε ολόκληρο το λεκανοπέδιο της Αττικής, χωρίζοντάς το σε ένα τριγωνικό πλέγμα κύκλων με ακτίνα 500 μέτρων και ζητώντας όλα τα σημεία για τα οποία οι υπηρεσίες έχουν δεδομένα στην ακτίνα αυτή. Ο απώτερος σκοπός της πλατφόρμας είναι η χρήση πληροφορίας σχετικά με τον αριθμό των χρηστών που βρίσκονται σε κάθε περιοχή της Αθήνας, για το λόγο αυτό, για κάθε σημείο που μας επιστρεφόταν από τις υπηρεσίες προσπαθούσαμε να βρούμε πόσοι χρήστες βρίσκονται εκεί τη στιγμή που γίνεται η ερώτηση. Για το Foursquare αυτό ήταν κάτι εύκολο, αφού μας το δίνει το API, ενώ για το Facebook χρειάστηκε να το υπολογίσουμε μέσω των διαφορών των συνολικών check-in, που παίρναμε ανά τακτά χρονικά διαστήματα. Για το Twitter αντιστοιχήσαμε απλά ένα tweet με την παρουσία ενός χρήστη του στη θέση από την οποία το tweet αναρτήθηκε. Μετρήσεις παίρνονταν μέσω του crawler ανά μία ώρα και αποθηκεύονταν στη βάση της πλατφόρμας. Το τρέξιμο του crawler κράτησε μια βδομάδα σε cluster του εργαστηρίου που αποτελούταν από 8 μηχανήματα. Αξίζει να σημειωθεί ότι λόγω του τρόπου συλλογής δεδομένων του crawler, μέσω των ερωτημάτων δηλαδή για διάφορες περιοχές της Αθήνας, είναι εύκολο να επεκταθεί και σε άλλες LBS, αφού το μόνο που χρειάζεται είναι η ανάπτυξη του αντίστοιχου κομματιού, που θα χρησιμοποιεί το API της υπηρεσίας για να κάνει ερωτήματα στις παραπάνω περιοχές.

Η αποθήκευση των δεδομένων είναι το επόμενο που μας απασχόλησε. Χρησιμοποιήσαμε μια κατανεμημένη βάση δεδομένων, την HBase[7], ώστε να εξασφαλίσουμε ότι θα έχουμε αρκετούς πόρους για την αποθήκευση του μεγάλου όγκου δεδομένων που θα συλλέξουμε από τις location based services. Η τεχνολογία αυτή μας εξασφαλίζει επίσης εύκολη επεκτασιμότητα, απαραίτητη για τη μακροχρόνια λειτουργία της πλατφόρμας, ενώ ταυτόχρονα θα διευκόλυνε στην περίπτωση επέκτασης και σε άλλες πόλεις εκτός της Αθήνας.

Στα πλαίσια της HBase, υλοποιήσαμε δύο τρόπους δεικτοδότησης των γεωγραφικών δεδομένων. Ο πρώτος στηρίζεται στο γεωγραφικό πλάτος των σημείων και λειτουργεί αποδοτικά σε ερωτήματα, που αφορούν ένα συγκεκριμένο εύρος πλατών. Στην περίπτωση όμως που ζητήσουμε ένα εύρος μηκών θα πρέπει να προσπελάσουμε όλα τα σημεία που είναι αποθηκευμένα, ενώ αν το ερώτημα περιορίζεται από ένα ελάχιστο και ένα μέγιστο τόσο στο πλάτος όσο και στο μήκος, πρέπει να προσπελαστούν όλα τα σημεία που ανήκουν στο εύρος των πλατών. Στον δεύτερο πίνακα χρησιμοποιήσαμε για την δεικτοδότηση μια γραμμικοποιημένη τιμή και των δύο διαστάσεων. Για το σκοπό αυτό κάναμε χρήση μιας space-filling curve, της z-order curve[8], η οποία περιπλέκει το γεωγραφικό πλάτος και μήκος δημιουργώντας μια μοναδική τιμή για κάθε σημείο ενός διασδιάστατου επιπέδου. Έτσι προσφέρει καλύτερη περίπλεξη των γεωγραφικών σημείων που οδηγεί σε λιγότερες λανθασμένες εγγραφές όταν ζητάμε ένα εύρος. Βέβαια η καμπύλη από μόνη της δεν αρκεί, αλλά χρειάστηκε να εφαρμόσουμε έναν αλγόριθμο διάσπασης τους αρχικού ερωτήματος –

άρα και εύρους – σε επί μέρους μικρότερα και τότε πετύχαμε την ακρίβεια που επιθυμούσαμε.

Επίσης, χρησιμοποιήσαμε έναν πίνακα ο οποίος αποθηκεύει τα ίδια δεδομένα με αυτά των άλλων δύο με μία όμως διαφορά: αυτή τη φορά για τη δεικτοδότηση χρησιμοποιείται το timestamp, ο χρόνος δηλαδή κατά τον οποίο πάρθηκαν τα δεδομένα. Αυτό έγινε με σκοπό να είναι πιο γρήγορα τα ερωτήματα που αφορούν ολόκληρη την περιοχή της Αθήνας για μια συγκεκριμένη χρονική περίοδο, αφού στην ουσία θα χρειαζόμαστε όλα τα σημεία που έχουν αποθηκευτεί, αλλά η αναζήτηση στον δεικτοδοτημένο με βάση το χρόνο πίνακα αυτό, θα δίνει προβάδισμα σε σχέση με τους άλλους δύο.

Τέλος, αναπτύξαμε μια εφαρμογή που να κάνει χρήση της πλατφόρμας αυτής και να προσφέρει κάποια χρήσιμα συμπεράσματα στο χρήστη της σχετικά με τον κόσμο στην Αθήνα. Δέχεται σαν είσοδο από το χρήστη τα όρια ενός γεωγραφικού τετραγώνου και ένα χρονικό διάστημα μέσα στη βδομάδα που ο crawler έτρεχε και επιστρέφει ένα χάρτη πυκνοτήτων (heatmap), που δείχνει τη συγκέντρωση του κόσμο στα πλαίσια αυτής της περιοχής, καθώς και το συνολικό αριθμό των χρηστών εκεί. Ο χάρτης δημιουργείται με τη βοήθεια του API του Google Maps[9] και προσφέρει εκτός από απεικόνιση και τη δυνατότητα ζουμ.

1.3 Οργάνωση του εγγράφου

Σε αυτό το πρώτο κεφάλαιο κάναμε μια εισαγωγή στο θέμα που μας απασχόλησε και παρουσιάσαμε τις τρεις location based services που μας έδωσαν κίνητρο για την ενασχόληση με τα χωροχρονικά δεδομένα. Επίσης περιγράψαμε με συντομία την αρχιτεκτονική της πλατφόρμας που χτίσαμε για τη συγκέντρωση και αποθήκευση δεδομένων, καθώς και της εφαρμογής που θα τρέξει δοκιμαστικά πάνω της. Θα ακολουθήσουν:

- Στο κεφάλαιο 2 περισσότερες πληροφορίες σχετικά με τις 3 LBS και την αποθήκευση των χωροχρονικών δεδομένων. Θα καλυφθεί επίσης το θέμα των κατανεμημένων βάσεων δεδομένων με ιδιαίτερη προσοχή στην HBase και θα αναλυθεί με λεπτομέρεια η αρχιτεκτονική της πλατφόρμας.

- Στο κεφάλαιο 3 θα παρουσιαστούν τα πειράματα που τρέξαμε πάνω στην πλατφόρμα και που θα μας οδηγήσουν στην επιλογή των κατάλληλων παραμέτρων για κάθε τεχνική ερωτημάτων και στο καταλληλότερο σχήμα αποθήκευσης. Επίσης θα δείξουμε στιγμιότυπα από τη χρήση της εφαρμογής που αναπτύξαμε για τη χρήση της πλατφόρμας.

- Στο κεφάλαιο 4 θα αναπτύξουμε τα συμπεράσματα που προέκυψαν από το κεφάλαιο των ερωτημάτων.

Κεφάλαιο 2

Θεωρητικό μέρος

Στο κεφάλαιο αυτό θα καλύψουμε όλο το θεωρητικό κομμάτι που μας χρειάστηκε στα πλαίσια της διπλωματικής αυτής. Θα εξετάσουμε τις location based services με τις οποίες ασχοληθήκαμε και από τις οποίες πήραμε δεδομένα, βλέποντας την πορεία της κάθε μίας στο χώρο των χωροχρονικών δεδομένων και τη μορφή των δεδομένων που μας προσφέρει. Θα ασχοληθούμε με τις κατανεμημένες βάσεις δεδομένων, τις συνθήκες που τις ανέδειξαν και τα προβλήματα που κλήθηκαν να λύσουν στα σύγχρονα προγραμματιστικά περιβάλλοντα, ενώ ιδιαίτερη προσοχή θα δείξουμε στην HBase, την οποία χρησιμοποιήσαμε σαν αποθήκη των δικών μας δεδομένων. Τέλος θα αναπτύξουμε την αρχιτεκτονική της εφαρμογής που χτίσαμε και απαρτίζεται από έναν crawler για τη συλλογή χωροχρονικής πληροφορίας, ένα σχήμα αποθήκευσης στην HBase και μια τεχνοτροπία δεικτοδότησης για την αποδοτική αποθήκευση και ανάκτηση δεδομένων.

2.1 Location Based Services

Οι location based services είναι υπηρεσίες που κάνουν χρήση του γεωγραφικού στίγματος των χρηστών τους για να τους βοηθήσουν να αλληλεπιδράσουν με τους γύρω τους ή να εξατομικεύσουν την εμπειρία που τους προσφέρουν. Βλέποντας πως τέτοιες υπηρεσίες γίνονται όλο και πιο γνωστές στο κοινό και αρχίζουν να χρησιμοποιούνται ευρέως, προσπαθήσαμε να δούμε πώς θα μπορούσαμε να χρησιμοποιήσουμε τη χωροχρονική πληροφορία που προσφέρουν σχετικά με τη θέση των ατόμων που τις χρησιμοποιούν. Θα μπορούσαμε να εξετάσουμε τη συγκέντρωση των χρηστών σε κάποια μεγαλούπολη, για παράδειγμα την Αθήνα, τον τρόπο μετακίνησής τους από μέρος σε μέρος και τα σημεία που συχνάζουν ανάλογα με τη μέρα και την ώρα; Για να απαντήσουμε στα ερωτήματα αυτά χρησιμοποιήσαμε τρία από τα πιο γνωστά κοινωνικά δίκτυα, που προσφέρουν υπηρεσίες βασισμένες στη θέση, το Facebook, το Foursquare και το Twitter.

2.1.1 Facebook

Το Facebook είναι μια υπηρεσία κοινωνικής δικτύωσης που ξεκίνησε τον Ιανουάριο του 2004, ενώ ανήκε και βρισκόταν υπό τη διαχείριση της εταιρείας Facebook Inc. Μετά από 8 χρόνια λειτουργίας, τον Ιούνιο του 2012, το Facebook απαριθμούσε περισσότερους από 955 εκατομμύρια ενεργούς χρήστες, πάνω από το μισό των οποίων χρησιμοποιούν το Facebook μέσω κινητής συσκευής. Οι χρήστες πρέπει να εγγραφούν πριν χρησιμοποιήσουν τη σελίδα. Έπειτα, μπορούν να δημιουργήσουν το προσωπικό τους προφίλ, να προσθέσουν άλλους χρήστες σαν “φίλους” τους, να ανταλλάσσουν μηνύματα και να συμπεριλαμβάνουν αυτόματες ανακοινώσεις όταν ανανεώνουν το προφίλ τους. Επιπλέον, οι χρήστες μπορούν να γίνουν μέλη και να παρακολουθούν γκρουπ γενικού ενδιαφέροντος, οργανωμένα κατά χώρο εργασίας, σχολείο, κολέγιο ή κάποιο άλλο χαρακτηριστικό και να κατηγοριοποιούν τους φίλους τους σε λίστες όπως “Άτομα από τη δουλειά” ή “Κοντινοί φίλοι”.

Το Facebook ιδρύθηκε από τον Mark Zuckerberg σε συνεργασία με τους συγκατοίκους και συναδέλφους του Eduardo Saverin, Andrew McCollum, Dustin Moskovitz και Chris Hughes. Τα μέλη της σελίδας αρχικά περιορίζονταν στους ιδρυτές της και τους φοιτητές του Harvard, αλλά σύντομα επεκτάθηκε και σε άλλα κολέγια της περιοχής της Βοστώνης, στο πανεπιστήμιο Stanford και στο Ivy League, έναν αθλητικό όμιλο ομάδων από οχτώ διαφορετικά ιδιωτικά ιδρύματα ανώτερης εκπαίδευσης στις βορειοανατολικές ΗΠΑ. Σταδιακά προσέφερε υποστήριξη για φοιτητές και σε διάφορα άλλα πανεπιστήμια, ενώ στη συνέχεια επέτρεψε τη χρήση στους μαθητές λυκείου και τελικά σε όσους ήταν 13 χρονών και πάνω. Ωστόσο, σύμφωνα με μια έρευνα του αμερικανικού περιοδικού Consumer Reports που εκδόθηκε το Μάιο του 2011, υπήρχαν 7.5 εκατομμύρια παιδιά κάτω των 13 και 5 εκατομμύρια κάτω των 10 που διατηρούσαν λογαριασμούς και παραβίαζαν τους όρους της υπηρεσίας.

Μία μελέτη της Compete.com[10] - μιας υπηρεσίας ανάλυσης διαδικτυακής κυκλοφορίας - που παρουσιάστηκε τον Ιανουάριο του 2009 κατέταξε το Facebook σαν την πιο χρησιμοποιούμενη υπηρεσία κοινωνικής δικτύωσης παγκοσμίως, σύμφωνα με τους μηνιαίους ενεργούς χρήστες. Το αμερικανικό περιοδικό Entertainment Weekly συμπεριέλαβε τη σελίδα στη λίστα με τα “best-of” του τέλους της δεκαετίας κάνοντας λόγο για το βαθμό που είχε εισχωρήσει το Facebook στην καθημερινότητα[11]. Η εταιρεία ανάλυσης και στατιστικών διαδικτύου Quantcast τον Μάιο του 2011 υπολόγιζε πως η υπηρεσία είχε 138.9 εκατομμύρια διαφορετικούς επισκέπτες μηνιαίως[12], ενώ σύμφωνα με την Social Media Today ήδη από τον Απρίλιο του 2010 ένα ποσοστό περίπου 41.6% του αμερικανικού πληθυσμού διατηρούσε λογαριασμό στο Facebook[13]. Η αλματώδης ανάπτυξης της υπηρεσίας ξεκίνησε να σταματά σε μερικές περιοχές, με τη σελίδα να χάνει 7 εκατομμύρια ενεργούς χρήστες στις Ηνωμένες Πολιτείες και τον Καναδά το Μάιο του 2011[14], κρατώντας όμως τους χρήστες της σε εντυπωσιακά νούμερα.

Το όνομα της υπηρεσίας πηγάζει από ένα βιβλίο που δινόταν στους νεοεισερχόμενους φοιτητές κάποιων πανεπιστημίων των Ηνωμένων Πολιτειών στην αρχή της ακαδημαϊκής χρονιάς, ώστε να τους βοηθήσουν να γνωριστούν με τους συμφοιτητές τους πιο εύκολα. Το βιβλίο αυτό οι φοιτητές το έλεγαν μεταξύ τους Facebook.

Το Facebook ανακοινώνει το Places

Στις 18 Αυγούστου του 2010 το Facebook ανακοίνωσε το “Places”, μια εφαρμογή που επέτρεπε στους χρήστες να κάνουν “check-in” σε κάποιο μέρος χρησιμοποιώντας κάποια κινητή συσκευή και να γνωστοποιούν στους φίλους τους πού βρίσκονται τη στιγμή αυτή. Αυτό το χαρακτηριστικό ήταν ήδη γνωστό από το Foursquare, ένα κοινωνικό δίκτυο στο οποίο οι χρήστες μοιράζονται πληροφορίες γεωγραφικής θέσης μέσω κινητών συσκευών.

Το Νοέμβριο του 2010 το Facebook ανακοίνωσε το “Deals”, ένα υποσύνολο του Places, που επέτρεπε στους χρήστες να κάνουν check-in από εστιατόρια, σούπερ-μάρκετ, μπαρ και καφετέριες χρησιμοποιώντας μια εφαρμογή για κινητές συσκευές και στη συνέχεια να κερδίζουν εκπτώσεις, κουπόνια και εμπορεύματα. Αυτό το γνώρισμα προωθήθηκε στην αγορά σαν μια ψηφιακή εκδοχή της κάρτας πόντων, με την οποία προμηθεύουν και την ελληνική αγορά αρκετά σούπερ-μάρκετ και ανταμείβουν τον αγοραστή ανάλογα με το πλήθος και την αξία των αγορών που κάνουν.[15]

Τον Αύγουστο του 2011, το Facebook Places μετατράπηκε από μία υπηρεσία check-in που ήταν, σε ένα εργαλείο μαρκαρίσματος (tag) γεωγραφικής θέσης. Η εφαρμογή για κινητές συσκευές είναι πλέον νεκρή και η δυνατότητα γεωγραφικού μαρκαρίσματος έχει ενσωματωθεί απευθείας στο ίδιο το Facebook.

Έτσι, οι χρήστες του Facebook μπορούν να μαρκάρουν μία συγκεκριμένη τοποθεσία ή επιχείρηση και να συμπεριλάβουν τη θέση αυτή σε μια ανανέωση του στάτους τους, μια εικόνα ή ένα βίντεο που θα αναρτήσουν. Οι χρήστες μπορούν επίσης να μαρκάρουν τους φίλους τους σε συγκεκριμένες τοποθεσίες μέσω των ανανεώσεων και των αναρτήσεών τους. Αυτό το μαρκάρισμα εμφανίζεται στο “News Feed” αυτού του ατόμου - τη λίστα δηλαδή με τα “νέα” του - (και στο “News Feed” των φίλων αυτών που έχουν μαρκαριστεί στη συγκεκριμένη θέση). Όταν μια επιχείρηση διατηρεί σελίδα στο Facebook και έχει συμπεριλάβει σε αυτή την ακριβή της θέση, μια ανάρτηση που μαρκάρεται στην τοποθεσία της επιχείρησης θα εμφανίσει έναν υπερσύνδεσμο, ο οποίος με τη σειρά του θα οδηγήσει τους χρήστες, που θα κάνουν κλικ πάνω του, στη σελίδα της επιχείρησης.

Έτσι, ενώ παλιότερα οι ιδιοκτήτες επιχειρήσεων έπρεπε να δημιουργήσουν και να δηλώσουν ένα ξεχωριστό Facebook Place, επιπρόσθετα με την ήδη υπάρχουσα σελίδα τους στο Facebook, τώρα απλά περιλαμβάνουν τη φυσική τους θέση στη σελίδα και η πληροφορία αυτή γίνεται αυτόματα γνωστή τόσο στον κατάλογο του Facebook Places όσο και στις σχετικές αναζητήσεις.[16]

Βλέπουμε δηλαδή ότι μετά την ενσωμάτωση του Places στη γενικότερη διεπαφή του Facebook, έχει υπάρξει μια μετατόπιση του ενδιαφέροντος και της λειτουργικότητας από την εγγραφή των επιχειρήσεων στο Places, στη σελίδα που διατηρούν στο Facebook. Πράγματι, εκτός από τις γεωγραφικές πληροφορίες και τη δυνατότητα check-in, η σελίδα κάθε επιχείρησης διατηρεί και τον αριθμό των συνολικών check-in που έχουν γίνει σε αυτή από τη στιγμή της δημιουργίας της. Η πληροφορία αυτή, που είναι διαθέσιμη στο διαχειριστή, είναι συνήθως ορατή και στους χρήστες του Facebook, καθώς και προσβάσιμη μέσω API για developers που θα θελήσουν πιθανώς να τη χρησιμοποιήσουν.

Σκέψεις για το μέλλον

Μέχρι στιγμής το Facebook, μέσω της σελίδας του για κινητές συσκευές και τις εφαρμογές του, περιλαμβάνει τη δυνατότητα check-in, που δείχνει πού βρίσκονταν οι χρήστες όταν μάρκαραν τον εαυτό τους – με δυνατότητα μάλιστα πρόσφατα για μαρκάρισμα και παλιότερων αναρτήσεων, πριν δηλαδή γίνει διαθέσιμη μέσω Facebook η βασισμένη στη γεωγραφική θέση υπηρεσία των Places. Ωστόσο δεν παρέχει την πληροφορία, αν ο χρήστης που έκανε το check-in βρίσκεται ακόμα σε αυτή την τοποθεσία. Αυτό το κενό στοχεύει να καλύψει με την απόκτηση του Glancee, μιας εφαρμογής που μέχρι πρότινος βρισκόταν στο App Store της Apple και ειδοποιεί τους χρήστες όταν άτομα με παρόμοια ενδιαφέροντα βρίσκονται σε κοντινή τοποθεσία. Η βασική του αρχή είναι ότι οι χρήστες μοιράζονται συνεχώς τη θέση τους ενώ η εφαρμογή τρέχει στο παρασκήνιο. Έτσι θα μπορούν να δουν ποιοι άλλοι χρήστες της εφαρμογής βρίσκονται στην γύρω περιοχή, με ποιους έχουν κοινά ενδιαφέροντα ή να λάβουν ανακοινώσεις για άτομα, με τα οποία πιθανώς θα ταίριαζαν. Αυτό κάνει την ανακοίνωση τρέχουσας θέσης ακόμα ευκολότερη και μπορεί να οδηγήσει σε αυθόρμητες συναντήσεις μεταξύ φίλων – με την προυπόθεση βέβαια οι χρήστες να συμφωνήσουν να δώσουν την πληροφορία της θέσης τους. Το Facebook πιθανώς να χρησιμοποιήσει την τεχνολογία αυτή επίσης για παρελθοντικές επισκέψεις των χρηστών ή για μελλοντικά σχέδιά τους και ενώ δεν είναι ακόμα γνωστό αν θα διατεθεί η παραπάνω υπηρεσία σαν ανεξάρτητη εφαρμογή ή θα ενσωματωθεί στο υπόλοιπο Facebook, βλέπουμε πως το ενδιαφέρον της σελίδας για τη χωροχρονική πληροφορία των χρηστών του όλο και μεγαλώνει.[17][18]

2.1.2 Foursquare

Το Foursquare είναι μια σελίδα κοινωνικής δικτύωσης βασισμένη στη γεωγραφική θέση των χρηστών, για κινητές συσκευές όπως τα smartphones. Οι χρήστες κάνουν check-in σε venues χρησιμοποιώντας την έκδοση της ιστοσελίδας του Foursquare για κινητές συσκευές, στέλνοντας γραπτό μήνυμα ή χρησιμοποιώντας μια ειδική εφαρμογή, η οποία εντοπίζει τα venues στην περιοχή που βρίσκεται ο χρήστης και τα παρουσιάζει σε μια λίστα. Η θέση του χρήστη εντοπίζεται χρησιμοποιώντας τεχνολογία GPS στην κινητή συσκευή ή μέσω εύρεσης θέσης δικτύου.

Η υπηρεσία δημιουργήθηκε το 2009 από τους Dennis Crowley και Naveen Selvadurai. Ο Crowley είχε ξεκινήσει πρωτύτερα ένα παρόμοιο project, το Dodgeball, σαν τη διπλωματική του εργασία στο Interactive Telecommunications Program (ITP) του πανεπιστημίου της Νέας Υόρκης (New York University - NYU). Η Google αγόρασε το Dodgeball το 2005 και το σταμάτησε το 2009, αντικαθιστώντας το με το Google Latitude. Η χρήση του Dodgeball στηριζόταν κυρίως σε τεχνολογία γραπτών μηνυμάτων (SMS), και όχι σε κάποια εφαρμογή.

Το Foursquare είναι η δεύτερη εκτέλεση της ίδιας ιδέας, ότι οι άνθρωποι δηλαδή μπορούν να χρησιμοποιήσουν κινητές συσκευές για να αλληλεπιδράσουν με το περιβάλλον τους . Πρόκειται για μια εφαρμογή ιστού που επιτρέπει στους εγγεγραμμένους χρήστες της να αναρτούν την παρουσία τους σε ένα venue (κάνοντας check-in) και να επικοινωνούν με τους φίλους τους. Το check-in απαιτεί ο χρήστης να έχει γραφτεί στην υπηρεσία και να είναι ενεργός. Μετά την πραγματοποίησή του, γίνεται ορατό στους φίλους του, δίνοντάς τους την πληροφορία για το πού βρίσκεται εκείνη τη στιγμή. Επίσης, βοηθάει το Foursquare να προσαρμόζει τις προτάσεις που θα του εμφανίζει ανάλογα με την ιστορία του. Η πραγματοποίηση ενός check-in είναι εύκολη, αφού το μόνο που έχει να κάνει ο χρήστης είναι να ανοίξει την εφαρμογή στην κινητή συσκευή του και το Foursquare θα χρησιμοποιήσει το GPS της για να βρει την τρέχουσα θέση του και να εμφανίσει μια λίστα με τα venues της περιοχής. Με την επιλογή ενός από αυτά τα μέρη και το πάτημα ενός κουμπιού το check-in πραγματοποιείται. Σε περίπτωση που το επιθυμητό venue δεν υπάρχει στη λίστα, δίνεται η δυνατότητα για ανεξάρτητη αναζήτησή του ή προσθήκη του κατευθείαν από την κινητή συσκευή.

Μετά το check-in υπάρχει η δυνατότητα κοινοποίησης της τοποθεσίας του χρήστη στους φίλους του στο Foursquare, αλλά και ανάρτηση στους λογαριασμούς του στο Facebook και το Twitter. Στα πλαίσια του σεβασμού στην ιδιωτικότητα ωστόσο υπάρχει και η επιλογή να μη μοιραστεί το check-in με κανέναν από τους φίλους του αλλά να παραμείνει ιδιωτικό και ορατό μόνο στον ίδιο.

Ένα από τα χαρακτηριστικά που συνέβαλλε στην ευρεία διάδοση της υπηρεσίας είναι ότι με κάθε check-in ο χρήστης ανταμείβεται με κάποιους πόντους. Κάθε check-in προσφέρει τουλάχιστον έναν πόντο, με έξτρα πόντους για συγκεκριμένες καταστάσεις όπως το να είναι ο “δήμαρχος” του venue, να κάνει check-in με έναν καινούριο φίλο ή ακόμα και να κάνει check-in σε κάποιες συγκεκριμένες αργίες. Οι πόντοι ουσιαστικά δεν σημαίνουν κάτι, αποτελούν όμως κίνητρο για τους χρήστες να ανταγωνιστούν με τους φίλους τους για το ποιος θα έχει τους περισσότερους και μέσω αυτού του ανταγωνισμού να αυξήσουν τη χρήση της εφαρμογής και τη συχνότητα των check-in τους.

Το άτομο μάλιστα που έχει κάνει τα περισσότερα check-in σε ένα venue τις τελευταίες 60 ημέρες ανακηρύσσεται σε “δήμαρχό” του. Έτσι, εκτός από τους έξτρα πόντους που αναφέραμε παραπάνω, ο “δήμαρχος” κερδίζει την αναγνώριση των άλλων χρηστών του Foursquare για την αγάπη του στο venue και η φωτογραφία του γίνεται ορατή κατά την είσοδο στη σελίδα του venue αυτού.

Πέρα από τους πόντους υπάρχουν και τα “εμβλήματα” (badges), τα οποία είναι ανταμοιβές που δίνονται για κάποιο συγκεκριμένο αριθμό check-in σε κάποιο συγκεκριμένο venue. Μερικά από αυτά σπονσοράρονται από διάφορες εταιρείες, όπως για παράδειγμα το History Channel, το έμβλημα του οποίου για να κερδίσει κάποιος θα πρέπει να “ακολουθεί” τη σελίδα του καναλιού στο Foursquare και να έχει κάνει check-in σε τρία από τα προτεινόμενα venues του.

Το δεύτερο χαρακτηριστικό που συνέβαλλε στην έκρηξη που προκάλεσε το Foursquare κατά την εμφάνισή του είναι η σύνδεσή του με τις επιχειρήσεις, οι οποίες αποτελούν τα venues στα οποία πραγματοποιούνται τα check-in, και η δημιουργία από τις τελευταίες προσφορών για τους χρήστες της υπηρεσίας. Μερικές επιχειρήσεις χρησιμοποιούν δηλαδή το Foursquare σαν μια ψηφιακή κάρτα πόντων και σαν ένα δέλεαρ για τους χρήστες του ώστε να επισκεφτούν τα καταστήματά τους. Έτσι, για να ανταμείψουν κάποιον τακτικό πελάτη μπορεί να του προσφέρουν ένα δωρεάν ποτό σε κάθε πέμπτη επίσκεψη ή να κάνουν έκπτωση στο 10% της αξίας του λογαριασμού τη δέκατη φορά που κάνει check-in. Άλλα μέρη προσφέρουν εκπτώσεις στο δήμαρχο και σε κάθε περίπτωση οι προσφορές που δίνονται στους χρήστες φαίνονται στην εφαρμογή του Foursquare κάθε φορά που κάποιος ψάχνει για κοντινά venues. Με το Foursquare ήταν η πρώτη φορά που οι επιχειρήσεις αντιλήφθηκαν την ευκαιρία αξιοποίησης των κοινωνικών δικτύων για marketing, βασισμένο πλέον στη χωροχρονική πληροφορία του χρήστη, και σταμάτησαν να αγνοούν και άλλες υπηρεσίες κοινωνικής δικτύωσης όπως το Facebook και το Twitter.[19][20]

Οι χρήστες ενθαρρύνονται να είναι ιδιαίτερα ακριβείς όσον αφορά τη θέση των check-in τους, αφού τους δίνεται η δυνατότητα να επιλέξουν ένα συγκεκριμένο όροφο ή περιοχή ενός κτιρίου ή να υποδείξουν συγκεκριμένη δραστηριότητα κατά τη διάρκεια του venue.

Μπορούν επίσης να δημιουργήσουν μια λίστα με μέρη που θέλουν να επισκεφτούν (“To Do” list) για προσωπική τους χρήση και να προσθέσουν συμβουλές σε venues (“Tips”) τις οποίες μπορούν να διαβάσουν αργότερα άλλοι χρήστες. Οι συμβουλές μπορεί να περιλαμβάνουν προτάσεις για το τι μπορεί κάποιος να κάνει στο συγκεκριμένο μέρος, ένα αξιοθέατο που θα μπορούσε να επισκεφτεί κατά τη διαμονή του σε μια πόλη ή μια σπεσιαλιτέ που πρέπει να δοκιμάσει.

Επιπλέον οι χρήστες μπορούν να δουν πού έχουν κάνει check-in στην υπηρεσία με την επιλογή “ιστορία”. Η λειτουργία αυτή τους επιτρέπει να ψάξουν τα προηγούμενά τους check-in, και μάλιστα ανάλογα με την κατηγορία, με το άτομο με το οποίο ήταν μαζί ή με την ώρα.

Όσον αφορά την ιστορία της εφαρμογής, το Foursquare ξεκίνησε το 2009 με περιορισμένη διαθεσιμότητα σε 100 μόνο σταθμούς μετρό παγκοσμίως. Τον Ιανουάριο του 2010 άλλαξε το μοντέλο τοποθεσιών του ώστε να επιτρέπει τα check-in από οποιαδήποτε τοποθεσία παγκοσμίως. Στις 21 Φεβρουαρίου 2011 έφτασε τους 7 εκατομμύρια χρήστες, ενώ τα συνολικά check-in ξεπέρασαν τα 750 εκατομμύρια ήδη πριν από το τέλος του Ιουνίου του 2011, με ένα μέσο όρο περίπου τριών εκατομμυρίων check-in την ημέρα. Στις 8 Αυγούστου του 2011 ο Αμερικανός πρόεδρος Barack Obama έγινε μέλος του Foursquare, ώστε το προσωπικό του Λευκού Οίκου να χρησιμοποιήσει την υπηρεσία για να αναρτεί συμβουλές από μέρη τα οποία ο πρόεδρος είχε επισκεφτεί. Τον Απρίλιο του 2012 η εταιρεία ανέφερε ότι είχε 20 εκατομμύρια εγγεγραμμένους χρήστες και απαριθμούσε πάνω από 2 δισεκατομμύρια check-in συνολικά.[21]

Το μέλλον του Foursquare

Παρόλο που το Foursquare δεν σκοπεύει να αλλάξει τα βασικά χαρακτηριστικά που το προσδιορίζουν, δηλαδή τη δυνατότητα που δίνει στους χρήστες του να κάνουν check-in σε διάφορα σημεία επιβραβευόμενοι με πόντους, όσον αφορά την εξέλιξή του στοχεύει σε

αλλαγές που θα του δώσουν καινούρια ώθηση. Το επίκεντρο των αλλαγών πέφτει σε ένα κουμπί “αναζήτησης”, το οποίο δίνει στους χρήστες προτάσεις για το πού θα πάνε, βασισμένο σε πληροφορίες όπως η ώρα της ημέρας, η δημοτικότητα των κοντινών σημείων και τα παλαιότερα check-in τους. Για παράδειγμα κάποιος που συνήθως κάνει check-in σε ιταλικά εστιατόρια θα δει περισσότερα μέρη της κατηγορίας αυτής στις προτάσεις του, από ότι κάποιος που προτιμάει μαγαζιά με παραδοσιακό φαγητό.

Ο διευθύνων σύμβουλος του Foursquare, Dennis Crowley, δηλώνει πως παρότι το κοινό έχει ταυτίσει το Foursquare με μια υπηρεσία που κάνεις check-in και παίρνεις πόντους, στόχος της ομάδας του είναι να εκμεταλλευτούν τα πλούσια δεδομένα σχετικά με το πώς οι άνθρωποι αλληλεπιδρούν με τα μέρη που επισκέπτονται και να τις μετατρέψει σε χρήσιμες προτάσεις. Σύμφωνα με τις δηλώσεις του η εταιρεία έχει συγκεντρώσει πάνω από δύο δισεκατομμύρια στοιχεία δεδομένων σχετικά με τους 20 εκατομμύρια χρήστες του, όπως το πού τους αρέσει να πηγαίνουν και πότε. Μπορεί να χρησιμοποιήσει αυτά τα δεδομένα για να προσφέρει ιδέες για συγκεκριμένα venues, όπως να ειδοποιήσει έναν χρήστη αν το αγαπημένο του εστιατόριο φαίνεται να έχει λιγότερο κόσμο από το συνηθισμένο και άρα να είναι πιο ελκυστικό για δείπνο.[22]

Βλέπουμε λοιπόν ότι το Foursquare, η υπηρεσία που σύστησε στις επιχειρήσεις την έννοια των κοινωνικών δικτύων βασισμένων στη γεωγραφική θέση των χρηστών τους, έδωσε νέα ώθηση στη χωροχρονική πληροφορία στα κοινωνικά δίκτυα. Τώρα, μετά την είσοδο στο χώρο των location based services τόσο του Facebook όσο και του Twitter, προσπαθεί να καινοτομήσει ξανά χρησιμοποιώντας τις μεγάλες αποθήκες δεδομένων του για να εξατομικεύσει τις συμβουλές που θα δώσει στον κάθε χρήστη.

2.1.3 Twitter

Το Twitter είναι μια διαδικτυακή υπηρεσία κοινωνικής δικτύωσης, που αυτοχαρακτηρίζεται σαν μια υπηρεσία microblogging και επιτρέπει στους χρήστες της να στέλνουν και να διαβάζουν μηνύματα κειμένου με μήκος μέχρι 140 χαρακτήρες, γνωστά και σαν “tweets”.

Δημιουργήθηκε τον Μάρτιο του 2006 από τον Jack Dorsey και δόθηκε στην κυκλοφορία τον Ιούλιο του ίδιου έτους. Η υπηρεσία γνώρισε γρήγορα παγκόσμια δημοσιότητα με πάνω από 500 εκατομμύρια ενεργούς χρήστες για το 2012, που παράγουν πάνω από 340 εκατομμύρια tweets καθημερινά και πραγματοποιούν πάνω από 1.6 δισεκατομμύρια ερωτήματα αναζήτησης την ημέρα. Από τη στιγμή της κυκλοφορίας του, το Twitter έχει γίνει μία από τις 10 ιστοσελίδες με τη μεγαλύτερη επισκεψιμότητα στο ίντερνετ και έχει χαρακτηριστεί σαν το “SMS του διαδικτύου”. Μη εγγεγραμμένοι χρήστες μπορούν επίσης να διαβάσουν tweets, με τη διαφορά ότι μόνο εγγεγραμμένοι χρήστες μπορούν να αναρτήσουν tweets μέσω της διεπαφής της ιστοσελίδας, γραπτών μηνυμάτων ή μιας πληθώρας εφαρμογών για κινητές συσκευές.

Η εταιρεία γνώρισε αστραπιαία ανάπτυξη. Τετρακόσιες χιλιάδες tweets αναρτούνταν κάθε τρίμηνο ήδη από το 2007. Ο αριθμός αυτός αυξήθηκε σε 100 εκατομμύρια tweets για το 2008, ενώ τον Φεβρουάριο του 2010 οι χρήστες της υπηρεσίας έστελναν 50 εκατομμύρια tweets κάθε μέρα. Τον Μάρτιο του ίδιου έτους η εταιρεία κατέγραφε πάνω από 70 χιλιάδες εγγεγραμμένες εφαρμογές. Τον Ιούνιο του 2010 περίπου 65 εκατομμύρια tweets αναρτούνταν καθημερινά, αριθμός που ισοδυναμεί σε 750 tweets το δευτερόλεπτο, σύμφωνα με πληροφορίες που έδωσε το ίδιο το Twitter. Τον Μάρτιο του 2011 αναρτούνταν περίπου 140 εκατομμύρια tweets παγκοσμίως και η σελίδα ανέβηκε έτσι στην 3η θέση των σελίδων κοινωνικής δικτύωσης, σύμφωνα με την ιστοσελίδα Compete.com, από την 22η που

βρισκόταν πρωτύτερα[23].

Η χρήση του Twitter βρίσκεται σε αιχμή κατά τη διάρκεια προεξέχοντων γεγονότων, όπως στο παγκόσμιο κύπελλο ποδοσφαίρου του 2011, όταν κατά τη νίκη της Ιαπωνίας κόντρα στη Δανία οι χρήστες έγραφαν με ρυθμό 3.085 tweets το δευτερόλεπτο.

Για να χρησιμοποιήσει κάποιος το Twitter, πρέπει πρώτα από όλα να δημιουργήσει ένα λογαριασμό σε αυτό, να εγγραφεί δηλαδή στην υπηρεσία. Έπειτα οδηγείται στη σελίδα του προφίλ του. Εκεί μπορεί να επεξεργαστεί κάποιες πληροφορίες για αυτόν, όπως η εικόνα που θα εμφανίζεται στις αναρτήσεις του, το όνομά του, το γεωγραφικό του στίγμα – στο οποίο θα επανέλθουμε σε λίγο – την ιστοσελίδα του και κάποια λόγια σχετικά με αυτόν. Του δίνεται επίσης η δυνατότητα να αναρτούνται αυτόματα τα tweets του στη σελίδα του στο Facebook.

Κάθε χρήστης έχει δύο λίστες ατόμων που σχετίζονται μαζί του. Η πρώτη είναι η λίστα των “following”, των ατόμων δηλαδή που ο ίδιος “ακολουθεί”. Το “ακολουθώ” στο Twitter έχει την έννοια ότι κάθε tweet που αναρτεί κάποιος από αυτούς που ένας χρήστης ακολουθεί, θα εμφανίζεται αυτόματα στην αρχική του σελίδα. Έτσι θα είναι πάντα ενημερωμένος σχετικά με τα μηνύματα που τα άτομα αυτά έχουν αναρτήσει. Το αντίστροφο συμβαίνει με τα άτομα που ανήκουν στη λίστα των “followers”. Πρόκειται για χρήστες που έχουν κάνει “follow” στον εν λόγω χρήστη, πράγμα που σημαίνει πως κάθε νέο tweet που αναρτεί, θα εμφανίζεται στις αρχικές σελίδες τους.

Τα tweets είναι δημόσια ορατά σαν προεπιλογή. Ωστόσο αν οι χρήστες το επιθυμούν μπορούν να περιορίσουν την ορατότητα των κειμένων τους μόνο στα άτομα που τους “ακολουθούν”. Οι χρήστες μπορούν να αναρτούν περιεχόμενο μέσω της σελίδας του Twitter, συμβατών εξωτερικών εφαρμογών (όπως αυτές που έχουν δημιουργηθεί για smartphones), ή μέσω SMS σε μερικές χώρες.[24]

Το twitter μπαίνει στις LBS

Τον Μάρτιο του 2010 και ενώ οι υπηρεσίες βασισμένες στη θέση γνώριζαν άνθηση, το Twitter ανακοίνωσε την ενσωμάτωση γεωγραφικής πληροφορίας στις υπηρεσίες του, επιτρέποντας στους χρήστες να επιλέξουν αν θα ήθελαν τα tweets τους να περιλαμβάνουν και τη θέση από την οποία τα έγραφαν. Αυτό ήταν δυνατό για αρκετούς μήνες μέσω τρίτων εφαρμογών, αφού η υπηρεσία είχε κάνει το νέο αυτό κομμάτι του API της διαθέσιμο σε όσους προγραμματιστές ήθελαν να γράψουν εφαρμογές που να ενσωματώνουν πληροφορία θέσης. Η αλλαγή ήταν μικρή: ο χρήστης επέλεγε πλέον μέσω μιας διεπαφής ότι ήθελε να περιέχεται η θέση του κάθε φορά που αναρτεί ένα κείμενο, και όταν το tweet εμφανιζόταν στους followers του, ένα εικονίδιο που έδειχνε τη θέση του το συνόδευε.[25]

Το Twitter, σε αντίθεση με το Foursquare και το Facebook, δεν κινήθηκε στη λογική του ότι ο χρήστης θα μπορεί να κάνει check-in, όταν βρεθεί σε κάποιο καθορισμένο venue, με σκοπό να συλλέξει κάποιους πόντους ή να επωφεληθεί από κάποια προσφορά. Αντίθετα, θεώρησε δεδομένο ότι από τη στιγμή που αναρτεί ένα tweet στο οποίο φαίνεται η θέση του, αυτό είναι ισοδύναμο με τη βεβαίωση ότι όντως βρίσκεται στο μέρος αυτό και εστίασε στη σύνδεση του περιεχομένου των tweet με τον εν λόγω χώρο. Λόγω της φύσης του Twitter, με τα tweet σαν βασική μονάδα, είναι η μόνη από τις τρεις υπηρεσίες που είδαμε μέχρι τώρα, που επιτρέπει στους χρήστες να αναρτήσουν πληροφορία με εννοιολογικό περιεχόμενο, δηλαδή γραπτό λόγο. Αυτό το γνώρισμα που την ξεχωρίζει προσπάθησε να το αξιοποιήσει, δημιουργώντας το έδαφος για πραγματικού χρόνου συζήτηση γύρω από ένα μέρος, που θα μπορούσε να είναι από χώρος δουλειάς, ή γραφείο μέχρι κάποιο μουσείο. Για παράδειγμα μια συζήτηση για τον πίνακα της Μόνα Λίζα έχει διαφορετική αξία όταν οι χρήστες βρίσκονται στο μουσείο του Λούβρου και μπορούν να εξετάσουν μόνοι τους τις παρατηρήσεις, αλλά και το tweet “Έχει φοβερό μποτιλιάρισμα σήμερα.”, έχει επιπλέον σημασία όταν ξέρουμε πως ο

χρήστης βρίσκεται τη στιγμή που το γράφει στο αυτοκίνητό του, σε κάποιο σημείο της Λεωφόρου Κηφισίας.

Δυστυχώς το γνώρισμα αυτό του Twitter, η δυνατότητα δηλαδή γεωγραφικής θέσης στα tweets, δεν έχει χρησιμοποιηθεί μέχρι σήμερα ευρέως, με χαρακτηριστική μία έρευνα της εταιρείας Semiocast για το πρώτο μισό του 2012, σύμφωνα με την οποία μόλις το 0.77% των χρηστών μαρκάρουν γεωγραφικά τις αναρτήσεις τους. Το ποσοστό βέβαια είναι ιδιαίτερα μικρό και μας κάνει σκεπτικούς για την αξιοπιστία του, ωστόσο είναι ενδεικτικό του κλίματος που επικρατεί. Μένει ο χρόνος να δείξει πόσο και πώς θα αξιοποιηθεί αυτή η υπηρεσία του Twitter, που σίγουρα καλύπτει αυτή τη στιγμή ένα κενό στις location based services.[26][27]

2.2 Χωροχρονικά δεδομένα

Τα χωροχρονικά δεδομένα είναι μια κατηγορία δεδομένων που περιλαμβάνουν τόσο το χρόνο όσο και το χώρο σαν μεταβλητές. Αν και η αποθήκευσή και επεξεργασία τους με τυπικά σχήματα βάσεων δεδομένων είναι δυνατή, δεν αποτελεί την πιο αποδοτική δυνατή λύση. Για το χειρισμό τους ήταν αναγκαίο να αναπτυχθούν νέες μορφές βάσεων δεδομένων που θα κάνουν δυνατή την αξιοποίηση και την επεξεργασία των δύο νέων αυτών μεταβλητών: του χρόνου και του χώρου.

2.2.1 Ο χρόνος στις βάσεις δεδομένων

Τα περισσότερα συστήματα βάσεων δεδομένων μοντελοποιούν μια συγκεκριμένη κατάσταση του κόσμου, για παράδειγμα τους τρέχοντες πελάτες μιας επιχείρησης, τους τρέχοντες φοιτητές ενός πανεπιστημίου ή τους χρήστες μιας location based service που έχουν κάνει check-in αυτή τη στιγμή σε κάποιο venue. Σε πολλές εφαρμογές ωστόσο, είναι ιδιαίτερα σημαντικό να αποθηκεύονται και να ανακτούνται πληροφορίες σχετικά με παρελθοντικές καταστάσεις, όπως θα ήταν το σύνολο των πελατών μιας επιχείρησης από το άνοιγμά της και οι αγορές που έχει κάνει ο καθένας. Τέτοιου είδους πληροφορία μπορεί να ενσωματωθεί σε ένα σχήμα σχεδίασης σε τυπικές σχεσιακές βάσεις δεδομένων, ωστόσο η διαδικασία αυτή απλοποιείται σημαντικά από βάσεις που προσφέρουν υποστήριξη για χρονολογικά δεδομένα.

Μια βάση δεδομένων μοντελοποιεί όπως είπαμε την κατάσταση κάποιας πτυχής του πραγματικού κόσμου. Τυπικά, οι βάσεις δεδομένων μοντελοποιούν μόνο μία κατάσταση, την τρέχουσα, και δεν αποθηκεύουν πληροφορίες σχετικά με παλαιότερες καταστάσεις, εκτός ίσως από ίχνη ελέγχων. Όταν η κατάσταση του πραγματικού κόσμου αλλάξει, η βάση ανανεώνεται και η πληροφορία σχετικά με την παλιότερη κατάσταση χάνεται. Ωστόσο, σε πολλές εφαρμογές είναι σημαντικό να μπορεί να αποθηκευτεί και να ανακτηθεί πληροφορία σχετικά με παλιότερες καταστάσεις, όπως για παράδειγμα μία βάση δεδομένων ενός χρήστη LBS που περιέχει όλο το ιστορικό των check-in του. Οι βάσεις που αποθηκεύουν πληροφορία σχετικά με καταστάσεις του πραγματικού κόσμου κατά το πέρασμα του χρόνου λέγονται χρονολογικές βάσεις δεδομένων.

Οι χρονολογικές βάσεις είναι άρρηκτα συνδεδεμένες με τις χρονολογικές σχέσεις, σχέσεις δηλαδή των οποίων κάθε εγγραφή έχει ένα σχετιζόμενο χρόνο κατά τη διάρκεια του οποίου είναι αληθής. Σε ένα απλό παράδειγμα θα μπορούσε κάθε εγγραφή να έχει μόνο ένα χρονικό διάστημα που σχετίζεται μαζί της και να αντιπροσωπεύεται έτσι μία φορά για κάθε διαφορετικό χρονικό διάστημα στο οποίο είναι αληθής. Βλέπουμε παρακάτω ένα τέτοιο παράδειγμα, στο οποίο καταγράφεται ο μισθός ενός υπαλλήλου, του κυρίου Παπαδόπουλου, τόσο για το έτος 2011, κατά το οποίο ήταν 1200 ευρώ, όσο και για το έτος 2012, στο οποίο

δέχτηκε αύξηση 100 ευρώ.

Κωδικός	Επίθετο	Μισθός	Από	Μέχρι
4823	Παπαδόπουλος	1200	01/01/11	31/12/11
4823	Παπαδόπουλος	1300	01/01/12	31/12/12

Πίνακας 1: Παράδειγμα χωροχρονικής σχέσης

Χρονολογικές γλώσσες ερωτημάτων

Μία σχέση βάσης δεδομένων χωρίς χρονολογική πληροφορία καλείται μερικές φορές σχέση στιγμιότυπου, αφού απεικονίζει μια κατάσταση ενός στιγμιότυπου του πραγματικού κόσμου. Με τον ίδιο τρόπο, ένα στιγμιότυπο μιας χρονολογικής σχέσης, για κάποια χρονική στιγμή t, είναι το σύνολο των εγγραφών της σχέσης που είναι αληθείς τη στιγμή t, με βάση τις τιμές των χρονικών διαστημάτων που σχετίζονται μαζί τους. Η λειτουργία του στιγμιότυπου μιας χρονολογικής σχέσης δίνει ένα στιγμιότυπο της σχέσης σε κάποιο συγκεκριμένο χρόνο ή τον τρέχοντα χρόνο αν αυτός δεν καθορίζεται.

Μία χρονολογική επιλογή είναι μια επιλογή που περιλαμβάνει χρονολογικά γνωρίσματα. Με τον ίδιο τρόπο μια χρονολογική προβολή είναι μια προβολή, στην οποία οι εγγραφές που εμφανίζονται κληρονομούν τους χρόνους τους από τις εγγραφές της αρχικής σχέσης. Ένας χρονολογικός σύνδεσμος (temporal join), είναι ένας σύνδεσμος στον οποίο ο χρόνος που εμφανίζεται σε μια εγγραφή του αποτελέσματος είναι η τομή των χρονικών διαστημάτων κατά τα οποία είναι αληθής οι εγγραφές από τις οποίες προήλθε. Αν οι αρχικές εγγραφές δεν είναι αληθείς σε κάποιο κοινό χρονολογικά διάστημα, τότε η πλειάδα αφαιρείται από το τελικό αποτέλεσμα.

Σε χρονικά διαστήματα μπορούν να εφαρμοστούν τα κατηγορήματα precedes (προηγείται), overlaps (επικαλύπτεται) και contains (περιέχει). Η λειτουργία intercect (τομή) μπορεί να εφαρμοστεί σε δύο διαστήματα για να δώσει ένα μόνο (πιθανά κενό) διάστημα, ενώ η ένωση (unio) δύο διαστημάτων δεν είναι απαραίτητο ότι θα δώσει σαν αποτέλεσμα ένα μόνο διάστημα.

2.2.2 Χωροταξικά και γεωγραφικά δεδομένα

Τα χωροταξικά δεδομένα περιλαμβάνουν τα γεωγραφικά δεδομένα και τα δεδομένα σχεδίασης με τη βοήθεια υπολογιστή. Στην πρώτη κατηγορία ανήκουν παραδείγματα όπως χάρτες με τις σχετικές με αυτούς πληροφορίες, ενώ στην δεύτερη σχέδια κτιρίων και ολοκληρωμένων κυκλωμάτων. Οι εφαρμογές χωροταξικών δεδομένων αρχικά αποθήκευαν τα δεδομένα σαν αρχεία σε τυπικά συστήματα αρχείων, με τον ίδιο τρόπο που οι εμπορικές εφαρμογές στις πρώτες γενιές αποθήκευαν τις δικές τους πληροφορίες. Καθώς όμως η πολυπλοκότητα, το μέγεθος των δεδομένων και το πλήθος των χρηστών αυξανόταν, τέτοιες προσεγγίσεις αποθήκευσης και ανάκτησης δεδομένων αποδείχθηκαν ανεπαρκείς για τις ανάγκες πολλών εφαρμογών που χρησιμοποιούν χωροταξικά δεδομένα.

Οι χωροταξικές εφαρμογές απαιτούν δυνατότητες που προσφέρονται από συστήματα βάσεων δεδομένων, και συγκεκριμένα τη δυνατότητα αποθήκευσης και εκτέλεσης ερωτημάτων σε μεγάλες ποσότητες δεδομένων με αποδοτικό τρόπο. Κάποιες εφαρμογές μπορεί να απαιτούν και άλλα γνωρίσματα βάσεων όπως ατομικές ενημερώσεις των αποθηκευμένων δεδομένων, ανθεκτικότητα και έλεγχο συγχρονικότητας.

Η υποστήριξη χωροταξικών δεδομένων σε βάσεις δεδομένων είναι σημαντική για την αποτελεσματική αποθήκευση των δεδομένων, τη δημιουργία ευρετηρίων και την εκτέλεση ερωτημάτων σε δεδομένα που βασίζονται σε χωροταξικές θέσεις. Για παράδειγμα, αν θέλουμε να αποθηκεύσουμε ένα σύνολο πολυγώνων σε μία βάση και να ρωτήσουμε για τα πολύγωνα τα οποία επικαλύπτονται δεν μπορούμε να χρησιμοποιήσουμε δομές δεικτοδότησης όπως τα B-δέντρα. Η αποδοτική επεξεργασία του ερωτήματος αυτού χρειάζεται ειδικού σκοπού δομές δεικτοδότησης όπως είναι τα R-δέντρα.

Δύο τύποι χωροταξικών δεδομένων είναι ιδιαίτερα σημαντικοί:

- Τα δεδομένα σχεδίασης με τη βοήθεια υπολογιστή (Computer-aided-design – CAD), που περιλαμβάνουν χωροταξικές πληροφορίες για το πώς κατασκευάζονται αντικείμενα όπως κτίρια, αυτοκίνητα ή αεροπλάνα ή περιέχουν τα σχέδια για ολοκληρωμένα κυκλώματα και ηλεκτρονικές συσκευές.
- Τα γεωγραφικά δεδομένα, όπως χάρτες δρόμων, χάρτες χρήσης γης, τοπογραφικοί χάρτες που δείχνουν όρια, χάρτες ιδιοκτησίας γης κτλ. Για την αποθήκευση των γεωγραφικών αυτών δεδομένων έχουν αναπτυχθεί ειδικά γεωγραφικά συστήματα πληροφοριών τα γνωστά και σαν GIS (geographic information systems).

Αναπαράσταση γεωμετρικών πληροφοριών

Για την αναπαράσταση των γεωμετρικών δομών σε μία βάση δεδομένων έχουν αναπτυχθεί διάφορες μέθοδοι. Για παράδειγμα, ένα ευθύγραμμο τμήμα μπορεί να αναπαρασταθεί από τις συντεταγμένες των δύο ακραίων του σημείων, οι οποίες σε ένα δισδιάστατο x-y επίπεδο θα αντιστοιχούσαν στη x και στην y τιμή των σημείων, ενώ στην περίπτωση που το ευθύγραμμο τμήμα βρισκόταν πάνω σε κάποιον χάρτη τα άκρα του θα αντιπροσωπευόταν από τα γεωγραφικά πλάτη και μήκη τους. Ένα πολύγραμμο (polyline) αποτελείται από μία συνδεδεμένη σειρά από ευθύγραμμα τμήματα και μπορεί να αναπαρασταθεί σαν μία λίστα που περιέχει τις συντεταγμένες των ακραίων σημείων των τμημάτων του.

Μπορούμε να αναπαραστήσουμε ένα πολύγωνο αναφέροντας τις κορυφές του με τη σειρά ή διαιρώντας το σε ένα σύνολο από τρίγωνα, σε μια διαδικασία που ονομάζεται τριγωνοποίηση.

Η αναπαράσταση σημείων και ευθύγραμμων τμημάτων σε τρισδιάστατο χώρο είναι παρόμοια με την αναπαράσταση στο δισδιάστατο χώρο, όπου η μόνη διαφορά είναι ότι τα σημεία έχουν ένα επιπλέον στοιχείο το z.

Βάσεις δεδομένων σχεδίων

Τα συστήματα Computer-Aided-Design (CAD) αποθήκευαν παραδοσιακά στη μνήμη στη διάρκεια τροποποιήσεων ή κάποιας άλλης επεξεργασίας και έγραφαν τα δεδομένα ξανά σε ένα αρχείο στο τέλος της επεξεργασίας. Καθώς όμως τα σχέδια γινόταν όλο και πιο μεγάλα και περίπλοκα κάτι τέτοιο οδηγούσε σε μεγάλη πολυπλοκότητα όσον αφορά τις τροποποιήσεις ενώ πολύ μεγάλα σχέδια πιθανόν να μην χωρούσαν ολόκληρα στη μνήμη. Έτσι οι σχεδιαστές στράφηκαν σε μια αντικειμενοστραφή προσέγγιση.

Τα αντικείμενα που αποθηκεύονται σε μία βάση δεδομένων σχεδίασης είναι γενικά γεωμετρικά αντικείμενα. Απλά δισδιάστατα γεωμετρικά αντικείμενα περιλαμβάνουν γραμμές και πολύγωνα, ενώ περίπλοκα δισδιάστατα μπορούν να προκύψουν από απλά με ένωση, τομή και διαφορά. Με τον ίδιο τρόπο περίπλοκα τρισδιάστατα αντικείμενα μπορούν να προκύψουν από σφαίρες, κυλίνδρους και κύβους, ενώ οι τρισδιάστατες επιφάνειες μπορούν να αναπαρασταθούν από μοντέλα πλέγματος που μοντελοποιούν την επιφάνεια ως ένα σύνολο

από απλούστερα αντικείμενα, όπως ευθύγραμμα τμήματα, τρίγωνα και ορθογώνια.

Γεωγραφικά δεδομένα

Τα γεωγραφικά δεδομένα είναι χωροταξικά στη φύση τους, με εικόνες από χάρτες και δορυφόρους να αποτελούν χαρακτηριστικά παραδείγματα. Οι βάσεις δεδομένων που τα χειρίζονται έχουν διάφορες χρήσεις, όπως online υπηρεσίες χαρτών, στις οποίες οι χρήστες μπορούν να κάνουν ζουμ στα σημεία που τους ενδιαφέρουν ή να ζητήσουν οδηγίες για το πώς θα πάνε από ένα ένα σημείο σε ένα άλλο, και συστήματα πλοήγησης οχημάτων, τα οποία απαιτούν μονάδα Global Positioning System (GPS) στο όχημα και χρησιμοποιούνται στη χάραξη πορείας ταξιδιών.

Αναπαράσταση γεωγραφικών δεδομένων

Τα γεωγραφικά δεδομένα μπορούν να κατηγοριοποιηθούν με δύο τύπους:

- Δεδομένα raster. Τα δεδομένα αυτά αποτελούνται από bitmap ή από pixel, σε δύο ή περισσερες διαστάσεις. Ένα τυπικό παράδειγμα είναι η δορυφορική εικόνα μιας περιοχής, μαζί με την οποία περιλαμβάνονται στα δεδομένα η θέση της, που καθορίζεται από τις γεωγραφικές συντεταγμένες των γωνιών της και η ανάλυσή της, που καθορίζεται από το συνολικό αριθμό των pixel ή πιο συχνά από την περιοχή που καλύπτεται από κάθε pixel.

  Τα δεδομένα raster αντιπροσωπεύονται συνήθως από ψηφίδες σε παράθεση, όπου κάθε μία καλύπτει μια περιοχή σταθερού μεγέθους. Μια μεγαλύτερη περιοχή μπορεί να παρουσιαστεί εμφανίζοντας όλες τις ψηφίδες που επικαλύπτονται με την περιοχή και για την εμφάνιση δεδομένων σε διαφορετικά επίπεδα ζουμ δημιουργείται ένα ξεχωριστό σύνολο ψηφίδων για κάθε επίπεδο ζουμ.
- Διανυσματικά δεδομένα. Τα δεδομένα αυτά κατασκευάζονται από απλά γεωμετρικά αντικείμενα, όπως σημεία, ευθύγραμμα τμήματα, πολύγραμμα, σφαίρες, κύβους και πολύεδρα. Στο πλαίσιο των γεωγραφικών δεδομένων τα σημεία αντιπροσωπεύονται συνήθως από το γεωγραφικό πλάτος, το γεωγραφικό μήκος και το ύψος στο οποίο βρίσκονται.

  Τα δεδομένα χαρτών συνήθως αντιπροσωπεύονται με διανυσματική μορφή. Οι δρόμοι αντιπροσωπεύονται σαν πολύγραμμα, οι λίμνες και οι νομοί με πολύπλοκα πολύγωνα, ενώ μπορούν να χρησιμοποιηθούν και πιο περίπλοκα σχήματα όπως πολύπλοκες καμπύλες για τους ποταμούς.

Χωροταξικά ερωτήματα

Υπάρχουν διάφοροι τύποι ερωτημάτων που περιλαμβάνουν χωροταξικές θέσεις. Τα ερωτήματα εγγύτητας ζητούν αντικείμενα που βρίσκονται κοντά σε μία συγκεκριμένη θέση. Ένα παράδειγμα είναι ένα ερώτημα που ζητάει όλα τα εστιατόρια που βρίσκονται σε δεδομένη απόσταση από ένα σημείο. Το ερώτημα εγγύτερου γείτονα ζητά το αντικείμενο που είναι πιο κοντά σε ένα συγκεκριμένο σημείο, χωρίς να υπάρχει κάποιος περιορισμός σχετικά με την απόσταση.

Τα ερωτήματα περιοχών σχετίζονται με χωροταξικές περιοχές. Ένα τέτοιο ερώτημα μπορεί να ζητά αντικείμενα που βρίσκονται, εν μέρει ή πλήρως, μέσα σε μία συγκεκριμένη περιοχή.

Τα ερωτήματα μπορεί επίσης να ζητούν τομές και ενώσεις περιοχών. Για παράδειγμα, με δεδομένες τις πληροφορίες μιας περιοχής για την επισκεψιμότητα των μαγαζιών της και το είδος του κάθε καταστήματος, μπορεί να ζητηθούν όλα τα εστιατόρια που έχουν πάνω από 20

πελάτες τη βραδιά.

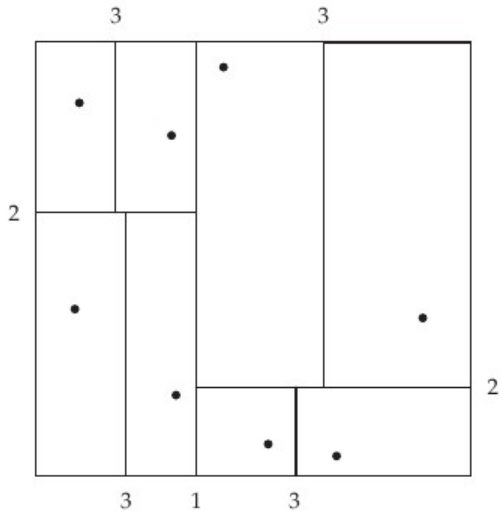
Ευρετήρια σε χωροταξικά δεδομένα

Τα ευρετήρια είναι απαραίτητα για αποτελεσματική πρόσβαση σε χωροταξικά δεδομένα. Οι παραδοσιακές δομές ευρετηρίων, όπως τα ευρετήρια hash και τα B-δέντρα δεν είναι κατάλληλα, αφού συσχετίζονται μόνο με μονοδιάστατα δεδομένα, ενώ τα χωροταξικά δεδομένα είναι τυπικά δύο ή περισσότερων διαστάσεων.

K-d δέντρα[28]

Τα k-d δέντρα ήταν μία από τις πρώτες δομές που χρησιμοποιήθηκαν σαν ευρετήρια πολλών διαστάσεων. Κάθε επίπεδο ενός k-d δέντρου διαιρεί το χώρο στα δύο. Η τμηματοποίηση γίνεται κατά μήκος μίας διάστασης, στον κόμβο του κορυφαίου επιπέδου του δέντρου, μαζί με μία άλλη διάσταση στους κόμβους του επόμενου επιπέδου και συνεχίζει έτσι πηγαίνοντας κυκλικά στις διαστάσεις. Η τμηματοποίηση προχωρεί με τέτοιο τρόπο ώστε, σε κάθε κόμβο, σχεδόν μισά από τα σημεία να αποθηκεύονται στο υποδέντρο που είναι στη μία πλευρά και τα μισά στο άλλο υποδέντρο. Η τμηματοποίηση σταματά όταν ένας κόμβος έχει λιγότερα από ένα μέγιστο αριθμό σημείων. Η εικόνα 1 παρακάτωδείχνει ένα σύνολο από σημεία στο δισδιάστατο χώρο και μία αναπαράσταση ενός k-d δέντρου του συνόλου των σημείων. Κάθε γραμμή αντιστοιχεί σε έναν κόμβο του δέντρου και ο μέγιστος αριθμός σημείων σε ένα φύλλο έχει οριστεί στο 1. Κάθε γραμμή στην εικόνα (εκτός από το εξωτερικό πλαίσιο) αντιστοιχεί σε έναν κόμβο στο k-d δέντρο. Η αρίθμηση των γραμμών της εικόνας δείχνει το επίπεδο του δέντρου στο οποίο εμφανίζεται ο αντίστοιχος κόμβος.

Το k-d-B δέντρο επεκτείνει το k-d δέντρο, ώστε να επιτρέπει πολλούς κόμβους παιδιά σε κάθε εσωτερικό κόμβο, όπως ένα B δέντρο επεκτείνει ένα δυαδικό δέντρο, προκειμένου να περιοριστεί το ύψος του δέντρου. Τα k-d-B δέντρα είναι πιο κατάλληλα για δευτερεύουσα αποθήκευση από ότι τα k-d δέντρα.

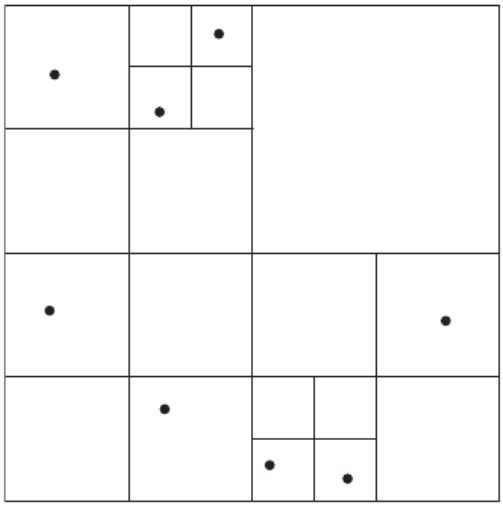


Εικόνα 1: Διαίρεση χώρου με ένα k-d δέντρο

Τετρα-δέντρα[29]

Μια εναλλακτική αναπαράσταση δισδιάστατων δεδομένων είναι ένα τετρα-δέντρο (quadtree). Ένα παράδειγμα της διαίρεσης του χώρου με ένα τετρα-δέντρο φαίνεται στην παρακάτω εικόνα. Το σύνολο των σημείων είναι το ίδιο με της προηγούμενης. Κάθε κόμβος ενός τετρα-δέντρου σχετίζεται με μια ορθογώνια περιοχή στο χώρο. Ο κορυφαίος κόμβος σχετίζεται με ολόκληρο το χώρο που μας ενδιαφέρει. Κάθε εσωτερικός κόμβος διαιρεί την περιοχή του σε τέσσερα ίσου μεγέθους τεταρτημόρια και αντίθετα, κάθε τέτοιος κόμβος έχει τέσσερις κόμβους παιδιά που αντιστοιχούν στα τέσσερα τεταρτημόρια. Τα φύλλα έχουν έναν αριθμό σημείων μεταξύ 0 και κάποιου σταθερού μέγιστου αριθμού. Αν η περιοχή που αντιστοιχεί σε έναν κόμβο έχει περισσότερα σημεία από τον μέγιστο αριθμό, δημιουργούνται κόμβοι παιδιά σε αυτόν τον κόμβο. Στο παράδειγμα της εικόνας ο μέγιστος αριθμός σημείων σε ένα φύλλο έχει οριστεί στο 1.

Αυτός ο τύπος τετρα-δέντρου ονομάζεται PR τετρα-δέντρο, για να υποδειχθεί ότι αποθηκεύει σημεία και ότι η διαίρεση του χώρου γίνεται ανάλογα με τις περιοχές, αντί ανάλογα με το πραγματικό σύνολο των σημείων που αποθηκεύονται. Μπορούμε να χρησιμοποιήσουμε τετρα-δέντρα περιοχών για να αποθηκεύουμε πληροφορίες πινάκων. Ένας κόμβος μιας περιοχής ενός τετρα-δέντρου είναι ένα φύλλο, αν είναι ίδιες όλες οι τιμές του πίνακα στην περιοχή που καλύπτει. Διαφορετικά, υποδιαιρείται περισσότερο σε τέσσερα παιδιά ίσων περιοχών και είναι συνεπώς εσωτερικός κόμβος. Κάθε κόμβος της περιοχής αντιστοιχεί σε έναν υπο-πίνακα από τιμές. Οι υπο-πίνακες αντιστοιχούν σε φύλλα, που είτε περιέχουν απλώς ένα στοιχείο πίνακα είτε έχουν πολλά στοιχεία, τα οποία έχουν όλα την ίδια τιμή.



Εικόνα 2: Διαίρεση χώρου από ένα τετρα-δέντρο

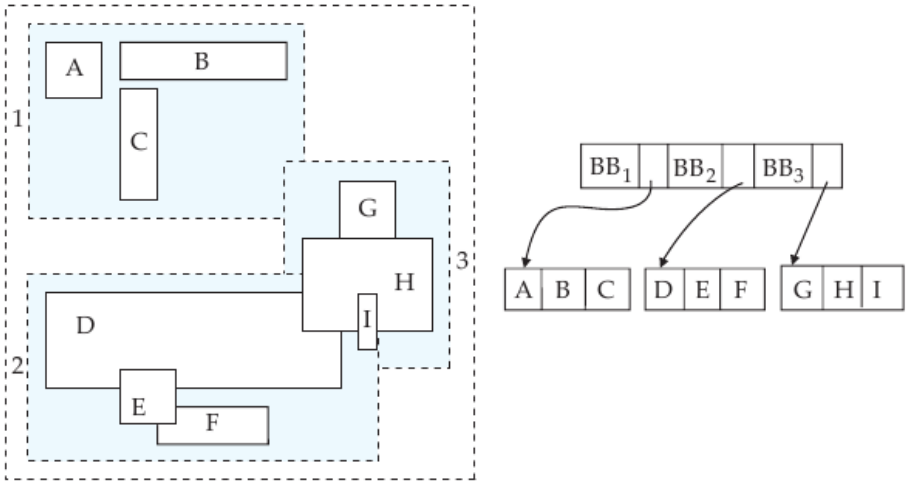
R-δέντρα[30]

Μια δομή αποθήκευσης, που ονομάζεται R-δέντρο, είναι χρήσιμη για ευρετήρια διαφόρων αντικειμένων, όπως ορθογωνίων, ευθυγράμμων τμημάτων και άλλων πολυγώνων. Ένα R-δέντρο είναι μία ισορροπημένη δομή δέντρου, με τα πολύγωνα να είναι αποθηκευμένα σε φύλλα, όπως σε ένα B+ δέντρο. Ωστόσο, αντί για ένα εύρος από τιμές, ένα ορθογώνιο περιβάλλον πλαίσιο σχετίζεται με κάθε κόμβο του δέντρου. Το περιβάλλον πλαίσιο ενός φύλλου είναι το μικρότερο ορθογώνιο που είναι παράλληλο με τους άξονες και περιέχουν όλα

τα αντικείμενα που είναι αποθηκευμένα στο φύλλο. Το περιβάλλον πλαίσιο των εσωτερικών κόμβων είναι, παρόμοια, το μικρότερο ορθογώνιο που είναι παράλληλο με τους άξονες που περιέχουν τα περιβάλλοντα πλαίσια των παιδιών του. Το περιβάλλον πλαίσιο ενός αντικειμένου (όπως ενός πολυγώνου) ορίζεται ως το μικρότερο ορθογώνιο, που είναι παράλληλο στους άξονες που περιέχουν το πολύγωνο.

Κάθε εσωτερικός κόμβος αποθηκεύει τα περιβάλλοντα πλαίσια των κόμβων-παιδιών, μαζί με τους δείκτες προς τους κόμβους-παιδιά. Κάθε φύλλο αποθηκεύει τα αντικείμενα των ευρετηρίων και μπορεί προαιρετικά να περιέχει τα περιβάλλοντα πλαίσια των αντικειμένων. Τα περιβάλλοντα πλαίσια βοηθούν να επιταχυνθούν οι έλεγχοι για επικάλυψη των ορθογωνίων με τα αντικείμενα ευρετηρίων, δηλαδή αν ένα ορθογώνιο δεν επικαλύπτεται από το περιβάλλον πλαίσιο ενός αντικειμένου, δεν μπορεί να επικαλύπτεται ούτε από το αντικείμενο.

Η παρακάτω εικόνα δείχνει ένα παράδειγμα ενός συνόλου ορθογωνίων (που έχει σχεδιαστεί με μία συνεχόμενη γραμμή) και τα περιβάλλοντα πλαίσια (που έχουν σχεδιαστεί με διακεκομμένη γραμμή) των κόμβων ενός R-δέντρου για το σύνολο των ορθογωνίων. Παρατηρούμε ότι τα περιβάλλοντα πλαίσια φαίνονται με επιπλέον χώρο ανάμεσά τους, ώστε να ξεχωρίζουν στην εικόνα. Στην πραγματικότητα, τα πλαίσια θα μπορούσαν να είναι μικρότερα και να βρίσκονται πολύ κοντά μέσα στα αντικείμενα που περιέχουν. Δηλαδή, κάθε πλευρά ενός περιβάλλοντος πλαισίου Β θα μπορούσε να αγγίζει τουλάχιστον ένα από τα αντικείμενα ή τα περιβάλλοντα πλαίσια που περιέχονται στο Β.[31]



Εικόνα 3: Ένα R-δέντρο

Space filling curves

Το ιδιαίτερο χαρακτηριστικό που έχουν τα γεωγραφικά δεδομένα και απαιτούν ιδιαίτερη μεταχείριση στη δεικτοδότησή τους είναι η ανάγκη για δημιουργία ευρετηρίων που θα λαμβάνουν υπόψιν και τις δύο διαστάσεις. Μία τεχνική για την αντιμετώπιση του προβλήματος αυτού, είναι η γραμμικοποίηση των δισδιάστατων δεδομένων σε μία μεταβλητή. Αυτό γίνεται μέσω καμπυλών των οποίων το εύρος περιέχει ολόκληρο το δισδιάστατο μοναδιαίο τετράγωνο και πρωτοεισήχθησαν από τον Giusseppe Peano[32]. Η γενική ιδέα την οποία ακολουθούν είναι η απόδοση μίας τιμής σε κάθε σημείο του επιπέδου με τρόπο τέτοιο, που σημεία που βρίσκονται “κοντά” στη γραμμή, να βρίσκονται “κοντά” και στο επίπεδο. Δύο παραδείγματα space filling καμπυλών αποτελούν η z-order curve[33] και η Hilbert curve[34].

2.3 Έννοιες κατανεμημένων βάσεων δεδομένων

Οι κατανεμημένες βάσεις δεδομένων συνδυάζουν τα πλεονεκτήματα των κατανεμημένων συστημάτων με την περιοχή της διαχείρισης βάσεων δεδομένων. Ένα κατανεμημένο υπολογιστικό σύστημα αποτελείται από ένα πλήθος υπολογιστικών στοιχείων, όχι απαραίτητα ομογενών που διασυνδέονται με ένα υπολογιστικό δίκτυο και που συνεργάζονται στην εκτέλεση μιας εργασίας. Σαν γενικό στόχο, τα κατανεμημένα συστήματα έχουν τη διάσπαση ενός μεγάλου, δύσκολα διαχειριζόμενου προβλήματος σε μικρότερα τμήματα και το λύνουν αποδοτικά με ένα συντονισμένο τρόπο. Η οικονομική βιωσιμότητα αυτής της προσέγγισης απορρέει από δύο λόγους: ότι διατίθεται περισσότερη υπολογιστική δύναμη για τη λύση μιας πολύπλοκης εργασίας και ότι η διαχείριση για κάθε αυτόνομο υπολογιστικό στοιχείο μπορεί να γίνει ανεξάρτητα και καθένα να αναπτύξει τις δικές του εφαρμογές.

Μπορούμε να ορίσουμε μια κατανεμημένη βάση δεδομένων (ΚΒΔ) σαν μια συλλογή πολλαπλών λογικά συσχετισμένων βάσεων δεδομένων διαμοιρασμένων σε ένα υπολογιστικό δίκτυο, και ένα κατανεμημένο σύστημα διαχείρισης βάσεων δεδομένων (ΚΣΔΒΔ) σαν ένα σύστημα λογισμικού που διαχειρίζεται μια κατανεμημένη βάση δεδομένων, ενώ η κατανομή είναι διαφανής στο χρήστη. Μια συλλογή από αρχεία σε διαφορετικούς κόμβους ενός δικτύου, και η διατήρηση των μεταξύ τους συσχετίσεων μέσω υπερσυνδέσμων, αποτελεί τη συνηθισμένη οργάνωση στο διαδίκτυο, με αρχεία Web σελίδων. Παλιότερα, στο σενάριο αυτό δεν εφαρμόζονταν οι συνηθισμένες λειτουργίες διαχείρισης δεδομένων, που συμπεριλαμβάνουν ομοιόμορφη επεξεργασία επερωτήσεων και επεξεργασία δοσοληψιών, ωστόσο η τεχνολογία προχωρήσει στην κατεύθυνση όπου οι βάσεις δεδομένων στο World Wide Web έχουν γίνει πλέον πραγματικότητα.

Πλεονεκτήματα των κατανεμημένων βάσεων δεδομένων

Η κατανεμημένη διαχείριση βάσεων δεδομένων έχει προταθεί για διάφορους λόγους που ποικίλουν από οργανωτική αποκέντρωση και οικονομική επεξεργασία έως μεγαλύτερη αυτονομία. Τονίζουμε μερικά από τα πλεονεκτήματα αυτά εδώ.

1. Διαχείριση κατανεμημένων δεδομένων με διαφορετικά επίπεδα διαφάνειας: στην ιδανική περίπτωση ένα ΣΔΒΔ πρέπει να είναι διαφανές στην κατανομή με την έννοια ότι κρύβει τις λεπτομέρειες του πού είναι φυσικά αποθηκευμένο στο σύστημα κάθε αρχείο (πίνακας, σχέση). Είναι δυνατοί δύο τύποι διαφάνειας:
    - Κατανομή ή διαφάνεια δικτύου: Αυτό αναφέρεται σε απελευθέρωση του χρήστη από λεπτομέρειες του δικτύου. Μπορεί να χωρισθεί σε διαφάνεια θέσης και διαφάνεια ονόματος. Η διαφάνεια θέσης αναφέρεται στο γεγονός ότι η εντολή που χρησιμοποιείται για την εκτέλεση μιας εργασίας είναι ανεξάρτητη από την τοποθεσία των δεδομένων και την τοποθεσία του συστήματος στο οποίο εκδόθηκε η εντολή. Η διαφάνεια ονόματος συνεπάγεται ότι όταν προσδιορισθεί ένα όνομα, το αντικείμενο με το όνομα μπορεί να προσπελασθεί χωρίς ασάφεια χωρίς επιπλέον προσδιορισμό.
    - Διαφάνεια ομοιοτυπίας: για λόγους καλύτερης διαθεσιμότητας, απόδοσης και αξιοπιστίας μπορεί να αποθηκεύονται αντίγραφα των δεδομένων σε πολλούς κόμβους. Η διαφάνεια ομοιοτυπίας θέλει το χρήστη να μην είναι ενήμερος της ύπαρξης των αντιγράφων.
    - Διαφάνεια κατατεμαχισμού: Δύο τύποι κατατεμαχισμού είναι δυνατοί. Ο οριζόντιος κατατεμαχισμός κατανέμει μία σχέση σε σύνολα πλειάδων (γραμμών). Ο κατακόρυφος κατατεμαχισμός από την άλλη κατανέμει μια σχέση σε υπό-σχέσεις όπου κάθε υπό-σχέση ορίζεται από ένα υποσύνολο των στηλών της αρχικής σχέσης. Μια καθολική επερώτηση από το χρήστη πρέπει να

μετασχηματισθεί σε πολλές επερωτήσεις κατατεμαχισμού. Η διαφάνεια κατατεμαχισμού θέλει το χρήστη να μην είναι ενήμερος για την ύπαρξη τεμαχίων.
    - Άλλοι τύποι διαφάνειας περιλαμβάνουν τη διαφάνεια σχεδιασμού και τη διαφάνεια εκτέλεσης – που αναφέρονται στη μη αναγκαιότητα γνώσης του τρόπου σχεδιασμού της κατανεμημένης βάσης δεδομένων και του πού εκτελείται μια δοσοληψία.
2. Αυξημένη αξιοπιστία και διαθεσιμότητα: Αυτά είναι τα δύο πιο γνωστά πλεονεκτήματα που αποδίδονται στις κατανεμημένες βάσεις δεδομένων. Υπό την ευρεία έννοια, η αξιοπιστία ορίζεται ως η πιθανότητα να είναι ένα σύστημα σε λειτουργία μια δεδομένη στιγμή, ενώ διαθεσιμότητα είναι η πιθανότητα να είναι το σύστημα διαθέσιμο συνεχώς κατά τη διάρκεια ενός χρονικού διαστήματος. Όταν τόσο τα δεδομένα όσο και το λογισμικό του ΣΔΒΔ είναι κατανεμημένα σε διαφορετικές εγκαταστάσεις, μια εγκατάσταση μπορεί να καταρρεύσει ενώ οι άλλες συνεχίζουν να λειτουργούν. Μόνο τα δεδομένα και το λογισμικό που βρίσκονται στην εγκατάσταση που κατέρρευσε δεν μπορούν να προσπελαστούν, γεγονός που βελτιώνει τόσο την αξιοπιστία όσο και τη διαθεσιμότητα. Περαιτέρω βελτίωση μπορεί να επιτευχθεί χρησιμοποιώντας ορθολογικά ομοιότυπα (replication), αποθηκεύοντας δηλαδή δεδομένα και λογισμικό σε περισσότερες από μία εγκαταστάσεις. Στα κεντρικοποιημένα συστήματα η κατάρρευση μιας εγκατάστασης καθιστά όλο το σύστημα μη διαθέσιμο σε όλους τους χρήστες. Στις κατανεμημένες βάσεις δεδομένων, μπορεί κάποια δεδομένα να μην είναι διαθέσιμα, αλλά οι χρήστες είναι δυνατόν να προσπελάσουν άλλα τμήματα της βάσης δεδομένων.
3. Βελτιωμένη απόδοση: Ένα κατανεμημένο ΣΔΒΔ τεμαχίζει τη βάση δεδομένων διατηρώντας τα δεδομένα κοντά στις θέσεις που τα χρειάζονται περισσότερο. Η τοπικότητα των δεδομένων ελαττώνει τη συμφόρηση στην κεντρική μονάδα επεξεργασίας και στις υπηρεσίες εισόδου/εξόδου και ταυτόχρονα ελαττώνει τις καθυστερήσεις που συμβαίνουν στα δίκτυα ευρείας ζώνης. Όταν μια μεγάλη βάση δεδομένων κατανέμεται σε πολλές εγκαταστάσεις, κάθε εγκατάσταση φιλοξενεί μια μικρή σχετικά βάση δεδομένων. Ως αποτέλεσμα, οι τοπικές επερωτήσεις και δοσοληψίες που προσπελάζουν δεδομένα μιας εγκατάστασης έχουν καλύτερη απόδοση, διότι οι τοπικές βάσεις δεδομένων είναι μικρότερες. Επιπλέον, κάθε εγκατάσταση έχει να εκτελέσει μικρότερο πλήθος από δοσοληψίες, σε σχέση με το συνολικό πλήθος δοσοληψιών που θα υποβαλλόταν σε ένα κεντρικό σύστημα. Επίσης, μπορεί να επιτευχθεί παραλληλισμός εντός της επερώτησης και μεταξύ των επερωτήσεων σε διαφορετικές εγκαταστάσεις ή με τη διάσπαση μιας επερώτησης σε ένα πλήθος υποεπερωτήσεων που εκτελούνται παράλληλα. Αυτό συμβάλλει στη βελτίωση της απόδοσης.
4. Ευκολότερη επέκταση: Σε ένα κατανεμημένο περιβάλλον, η επέκταση του συστήματος είτε με προσθήκη δεδομένων, αυξάνοντας το μέγεθος της βάσης δεδομένων, είτε προσθέτοντας περισσότερους επεξεργαστές είναι πολύ πιο εύκολη.

Οι διαφάνειες που εξετάστηκαν παραπάνω στο (1) οδηγούν σε ένα συμβιβασμό μεταξύ ευκολίας χρήσης και επιπλέον κόστους για υποστήριξη της διαφάνειας. Ολική διαφάνεια παρέχει στον καθολικό χρήστη μια όψη όλης της ΚΒΔ σαν ενός κεντρικοποιημένου συστήματος. Η διαφάνεια παρέχεται σαν συμπλήρωμα της αυτονομίας που δίνει στους χρήστες στενότερο έλεγχο των τοπικών βάσεων δεδομένων τους. Τα χαρακτηριστικά της διαφάνειας μπορεί να υλοποιηθούν σαν μέρος της γλώσσας του χρήστη, που μπορεί να μεταφράσει τις απαιτούμενες υπηρεσίες σε κατάλληλες πράξεις. Επιπλέον, η διαφάνεια παρεμβάλλεται στα χαρακτηριστικά που υποστηρίζονται από το λειτουργικό σύστημα και το

ΣΔΒΔ.

Επιπλέον λειτουργίες κατανεμημένων βάσεων δεδομένων

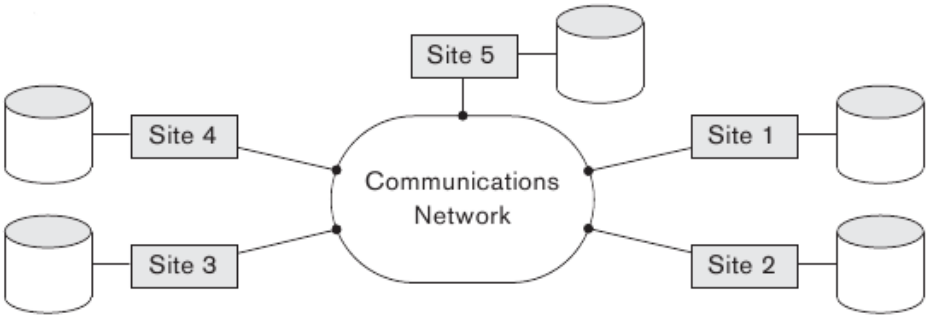
Η κατανομή οδηγεί σε αυξημένη πολυπλοκότητα στο σχεδιασμό και την υλοποίηση του συστήματος. Για να επιτευχθούν τα δυνητικά πλεονεκτήματα που αναφέρθηκαν προηγουμένως, πρέπει το λογισμικό του ΚΣΔΒΔ να έχει τη δυνατότητα να παρέχει τις ακόλουθες πρόσθετες λειτουργίες, σε σχέση με αυτές ενός κεντρικού συστήματος:

- Καταγραφή των δεδομένων: η δυνατότητα καταγραφής της κατανομής και της ομοιοτυπίας των δεδομένων στον κατάλογο του ΣΔΒΔ.
- Κατανεμημένη επεξεργασία επερωτήσεων: η δυνατότητα προσπέλασης σε απομακρυσμένες εγκαταστάσεις και η διακίνηση επερωτήσεων και δεδομένων μεταξύ διαφορετικών εγκαταστάσεων, μέσω ενός επικοινωνιακού δικτύου.
- Κατανεμημένη διαχείριση δοσοληψιών: η δυνατότητα διαμόρφωσης στρατηγικών εκτέλεσης για επερωτήσεις και δοσοληψίες που προσπελάζουν δεδομένα σε σε περισσότερες από μία εγκαταστάσεις.
- Διαχείριση δεδομένων ομοιοτυπίας: η δυνατότητα να αποφασίζει ποιο αντίγραφο ενός δεδομένου για το οποίο χρησιμοποιείται ομοιοτυπία θα προσπελασθεί και η δυνατότητα διατήρησης της συνέπειας μεταξύ των αντιγράφων ενός δεδομένου που επαναλαμβάνεται.
- Ανάκαμψη κατανεμημένων βάσεων δεδομένων: η δυνατότητα ανάκαμψης από αποτυχίες στις επί μέρους εγκαταστάσεις και από νέους τύπους αποτυχιών, όπως η αποτυχία μιας επικοινωνιακής σύνδεσης.
- Ασφάλεια: οι κατανεμημένες δοσοληψίες πρέπει να εκτελούνται με την κατάλληλη διαχείριση της ασφάλειας των δεδομένων και τα προνόμια δικαιοδοσίας/προσπέλασης των χρηστών.
- Διαχείριση κατανεμημένου ευρετηρίου (καταλόγου): ένα ευρετήριο περιέχει πληροφορίες (μεταδεδομένα) για τα δεδομένα στη βάση δεδομένων. Το ευρετήριο μπορεί να είναι καθολικό για όλη τη ΚΒΔ, ή τοπικό για κάθε εγκατάσταση. Η τοποθέτηση και η κατανομή του ευρετηρίου αποτελούν θέμα σχεδιασμού και πολιτικής.

Οι λειτουργίες αυτές αυξάνουν την πολυπλοκότητα ενός ΚΣΒΔ σε σχέση με ένα κεντρικό ΣΔΒΔ. Πριν μπορέσουμε να επωφεληθούμε πλήρως από τα πλεονεκτήματα της κατανομής, πρέπει να βρούμε ικανοποιητικές λύσεις σε αυτά τα σχεδιαστικά θέματα και προβλήματα. Ο στόχος να συμπεριληφθεί όλη αυτή η λειτουργικότητα στο ΚΣΔΒΔ είναι δύσκολο να επιτευχθεί και η εύρεση βέλτιστων λύσεων είναι ακόμη μακριά.

Στο επίπεδο του φυσικού υλικού οι ακόλουθοι βασικοί παράγοντες διαχωρίζουν ένα ΚΣΔΒΔ από ένα συγκεντρωτικό σύστημα:

- Υπάρχουν πολλοί υπολογιστές που ονομάζονται εγκαταστάσεις ή κόμβοι.
- Οι κόμβοι αυτοί πρέπει να συνδέονται με κάποιον τύπο επικοινωνιακού δικτύου για τη μεταφορά δεδομένων και εντολών μεταξύ των κόμβων, όπως φαίνεται στην παρακάτω εικόνα.



Εικόνα 4: Πραγματικά κατανεμημένη αρχιτεκτονική βάσεων δεδομένων

Οι κόμβοι πρέπει να βρίσκονται φυσικώς κοντά – για παράδειγμα στο ίδιο κτίριο ή ομάδα γειτονικών κτιρίων – και να συνδέονται με ένα τοπικό δίκτυο, ή μπορεί να είναι γεωγραφικά κατανεμημένοι σε μεγάλες αποστάσεις και να συνδέονται μέσω δικτύου ευρείας ζώνης. Τυπικά, τα τοπικά δίκτυα χρησιμοποιούν καλώδια ενώ τα δίκτυα ευρείας ζώνης χρησιμοποιούν τηλεφωνικές γραμμές ή δορυφόρους. Είναι επίσης δυνατόν να χρησιμοποιηθεί συνδυασμός των δύο τύπων δικτύων.

Τα δίκτυα μπορεί να έχουν διαφορετικές τοπολογίες, οι οποίες ορίζουν τις διαδρομές άμεσης επικοινωνίας μεταξύ των κόμβων. Ο τύπος και η τοπολογία του δικτύου που χρησιμοποιείται μπορεί να έχει σημαντική επίδραση στην απόδοση και επομένως στις στρατηγικές για κατανεμημένη επεξεργασία επερωτήσεων και στο σχεδιασμό της κατανεμημένης βάσης δεδομένων. Ωστόσο, για θέματα αρχιτεκτονικής υψηλού επιπέδου δεν έχει σημασία ο τύπος του δικτύου που χρησιμοποιείται. Το μόνο που έχει σημασία είναι κάθε κόμβος να επικοινωνεί άμεσα ή έμμεσα με κάθε άλλο κόμβο.[35]

Μειονεκτήματα

Οι κατανεμημένες βάσεις δεδομένων βέβαια εκτός από τα πλεονεκτήματα και τις δυνατότητες που προσφέρουν εμφανίζουν και μειονεκτήματα. Το βασικότερο από αυτά είναι η επιπλέον πολυπλοκότητα που απαιτείται ώστε να εξασφαλιστεί ο σωστός συντονισμός στις διάφορες θέσεις. Αυτή η αυξημένη πολυπλοκότητα παίρνει διάφορες μορφές:

- Κόστος ανάπτυξης λογισμικού. Είναι πιο δύσκολος ο χειρισμός ενός κατανεμημένου συστήματος βάσης δεδομένων. Συνεπώς, κοστίζει περισσότερο.
- Μεγαλύτερη πιθανότητα για λάθη. Αφού οι κόμβοι, που αποτελούν το κατανεμημένο σύστημα λειτουργούν παράλληλα, είναι δυσκολότερο να εξασφαλίσουμε ότι είναι σωστοί οι αλγόριθμοι, ειδικά για λειτουργίες στη διάρκεια προβλημάτων μέρους του συστήματος και κατά την αποκατάσταση από προβλήματα. Υπάρχει πιθανότητα να περάσουν πολύ λεπτά λάθη.
- Αυξημένο κόστος επεξεργασίας. Η ανταλλαγή μηνυμάτων και οι επιπλέον υπολογισμοί που απαιτούνται ώστε να επιτευχθεί συντονισμός μεταξύ των θέσεων είναι μια μορφή κόστους που δεν υπάρχει στα κεντρικά συστήματα.[31]

2.3.2 HBase

Η κατανεμημένη βάση δεδομένων που χρησιμοποιήσαμε στα πλαίσια της διπλωματικής είναι η HBase, η οποία μας προσφέρεται μέσω του Hadoop[36], ενός project που έχει αναπτύξει η Apache για κατανεμημένη επεξεργασία. Πρόκειται για μια βάση δεδομένων ανοιχτού κώδικα, μη σχεσιακή, κατανεμημένη, μοντελοποιημένη σύμφωνα με το BigTable[37] της Google και γραμμένη σε Java. Τρέχει πάνω από το HDFS[38] (Hadoop Distributed Filesystem) παρέχοντας στο Hadoop δυνατότητες σύμφωνες με αυτές του BigTable. Έτσι προσφέρει έναν ανεχτικό σε λάθη τρόπο αποθήκευσης μεγάλων ποσοτήτων αραιών δεδομένων. Στο κεφάλαιο αυτό θα κάνουμε μια επισκόπηση της αρχιτεκτονικής της HBase.

Το πρότυπο του Bigtable

Το 2003 η Google δημοσίευσε ένα paper με τίτλο “The Google File System”[39]. Σε αυτό παρουσίαζε το κλιμακούμενο κατανεμημένο σύστημα αρχείων της, του οποίου η συντόμευση είναι GFS, και χρησιμοποιούσε ένα cluster από μηχανήματα εμπορίου για να αποθηκεύει τεράστιες ποσότητες δεδομένων. Το σύστημα αρχείων υποστήριζε αντιγραφή δεδομένων μεταξύ κόμβων έτσι ώστε η απώλεια ενός σέρβερ αποθήκευσης να μην έχει αντίκτυπο στη διαθεσιμότητα των δεδομένων. Επίσης, είχε βελτιστοποιηθεί σε αναγνώσεις ροής (streaming reads) έτσι ώστε τα δεδομένα να μπορούσαν να διαβαστούν για επεξεργασία κάποια στιγμή αργότερα.

Λίγο αργότερα, δημοσιεύτηκε ένα άλλο paper από τη Google, με τίτλο “MapReduce: Simplified Data Processing on Large Clusters”[40]. Το MapReduce ήταν το κομμάτι που έλειπε από την GFS αρχιτεκτονική για να κάνει χρήση του μεγάλου αριθμού από CPU που παρείχε κάθε σέρβερ εμπορίου στον GFS cluster. Το MapReduce σε συνδυασμό με το GFS σχημάτιζαν τη ραχακοκαλιά για την επεξεργασία ογκωδών ποσών δεδομένων, συμπεριλαμβανομένου και ολόκληρης της δεικτοδότησης αναζήτησης που διατηρεί η Google.

Αυτό που έλειπε ωστόσο, ήταν η ικανότητα πρόσβασης στα δεδομένα με τυχαίο τρόπο και σε σχεδόν πραγματικό χρόνο, εννοώντας χρόνο αρκετά καλό για να τρέξει μια υπηρεσία διαδικτύου για παράδειγμα. Ένα άλλο μειονέκτημα της σχεδίασης με βάση το GFS είναι ότι είναι καλή για μεγάλα ή πολύ μεγάλα αρχεία, αλλά όχι τόσο καλή με εκατομμύρια μικρά αρχεία, επειδή τα δεδομένα που κρατούνται στη μνήμη από τον master node είναι άρρηκτα δεμένα με τον αριθμό των αρχείων. Έτσι όσο περισσότερα είναι τα αρχεία που θέλουμε να επεξεργαστούμε, τόσο μεγαλύτερη θα είναι και η πίεση στη μνήμη του master.

Έτσι η Google προσπάθησε να βρει μια λύση που θα μπορούσε να ανταπεξέλθει σε εφαρμογές αλληλεπίδρασης, όπως το Mail ή το Analytics, ενώ θα έκανε χρήση της ίδιας υποδομής και θα βασιζόταν στο GFS για αντιγραφή και διαθεσιμότητα δεδομένων. Τα δεδομένα που θα αποθηκεύονταν θα έπρεπε να αποτελούνται από πολύ μικρότερες οντότητες, και το σύστημα θα έπρεπε διαφανώς να φροντίζει για την συγκέντρωση των μικρών εγγραφών σε πολύ μεγάλα αρχεία αποθήκευσης και να προσφέρει κάποιο είδος δεικτοδότησης που να επιτρέπει στο χρήστη να ανακτήσει δεδομένα με ένα ελάχιστο αριθμό αναζητήσεων στο δίσκο. Τέλος, θα έπρεπε να είναι ικανή να αποθηκεύσει ολόκληρο το crawling του ιστού που κάνει η Google και να δουλεύει με το MapReduce για να δημιουργεί ολόκληρη τη δεικτοδότηση αναζήτησης σε έναν ικανοποιητικό χρόνο.

Γνωρίζοντας τις αδυναμίες των RDBMS σε θέματα κλιμάκωσης, οι μηχανικοί της Google προσέγγισαν το πρόβλημα διαφορετικά: απαγκιστρώθηκαν από τα χαρακτηριστικά των σχεσιακών βάσεων δεδομένων και χρησιμοποίησαν ένα απλό API με βασικές λειτουργίες δημιουργίας, ανάγνωσης, ενημέρωσης και διαγραφής (γνωστές και σαν CRUD), σε συνδυασμό με μία λειτουργία σάρωσης για περιήγηση σε μεγαλύτερα εύρη κλειδιών ή και ολόκληρους πίνακες. Το αποκορύφωμα αυτών των προσπαθειών ήρθε με την παρουσίαση το

2006 ενός paper με τίτλο “Bigtable: A Distributed Storage System for Structured Data”[41], από το οποίο ακολουθούν δύο αποσπάσματα:

- Το Bigtable είναι ένα κατανεμημένο σύστημα αποθήκευσης για το χειρισμό δομημένων δεδομένων, που είναι σχεδιασμένο για κλιμάκωση σε πολύ μεγάλο μέγεθος, petabytes από δεδομένα δηλαδή ανάμεσα σε χιλιάδες σέρβερ εμπορίου.
- ... μία αραιή, κατανεμημένη, εμμένουσα, πολυδιάστατη, ταξινομημένη απεικόνιση.

Το Bigtable είναι ιδιαίτερα σημαντικό για εμάς, αφού η HBase είναι μία πολύ πιστή εφαρμογή του. Για το λόγο αυτό παρακάτω θα δούμε μερικές από τις βασικές του έννοιες.

Πίνακες, γραμμές, στήλες και κελιά

Η βασική μονάδα του Bigtable είναι η στήλη. Μία ή περισσότερες στήλες σχηματίζουν μία γραμμή, στην οποία απευθυνόμαστε μοναδικά με ένα κλειδί γραμμής. Ένας αριθμός γραμμών, με τη σειρά τους, σχηματίζουν έναν πίνακα, και μέσα στη βάση μπορεί να υπάρχουν πολύ τέτοιοι. Κάθε στήλη μπορεί να έχει πολλαπλές εκδόσεις, με κάθε διακριτή τιμή να περιέχεται σε ένα ξεχωριστό κελί. Τα παραπάνω ακούγονται λογικά για κάποια τυπική βάση δεδομένων με το επιπλέον χαρακτηριστικό της δυνατότητας διατήρησης πολλαπλών εκδόσεων κάθε κελιού, αλλά φυσικά υπάρχουν περισσότερα από αυτά.

```
hbase(main):001:0> scan 'table1'
ROW                           COLUMN+CELL
row-1                          column=cf1:, timestamp=1297073325971 ...
row-10                         column=cf1:, timestamp=1297073337383 ...
row-11                         column=cf1:, timestamp=1297073340493 ...
row-2                          column=cf1:, timestamp=1297073329851 ...
row-22                         column=cf1:, timestamp=1297073344482 ...
row-3                          column=cf1:, timestamp=1297073333504 ...
row-abc                        column=cf1:, timestamp=1297073349875 ...
7 row(s) in 0.1100 seconds
```

Εικόνα 5: Γραμμές ταξινομημένες λεξικογραφικά σύμφωνα με το κλειδί τους

Όλες οι γραμμές πρέπει να είναι πάντα ταξινομημένες λεξικογραφικά σύμφωνα με το κλειδί γραμμής τους. Το παρακάτω παράδειγμα δείχνει σε τι κατάσταση θα βρίσκεται η βάση, μετά την προσθήκη μερικών γραμμών με διαφορετικά κλειδιά.

Παρατηρούμε ότι η ταξινόμηση δε γίνεται με τη σειρά που ίσως θα περιμέναμε. Στη λεξικογραφική ταξινόμηση, κάθε κλειδί συγκρίνεται με τα υπόλοιπα σε δυαδικό επίπεδο, byte προς byte, από αριστερά προς τα δεξιά. Έτσι, αφού το “row-1” βρίσκεται πριν από το “row-2”, ό,τι και να ακολουθεί μετά (πχ “row-11”) θα τοποθετηθεί πριν το “row-2”.

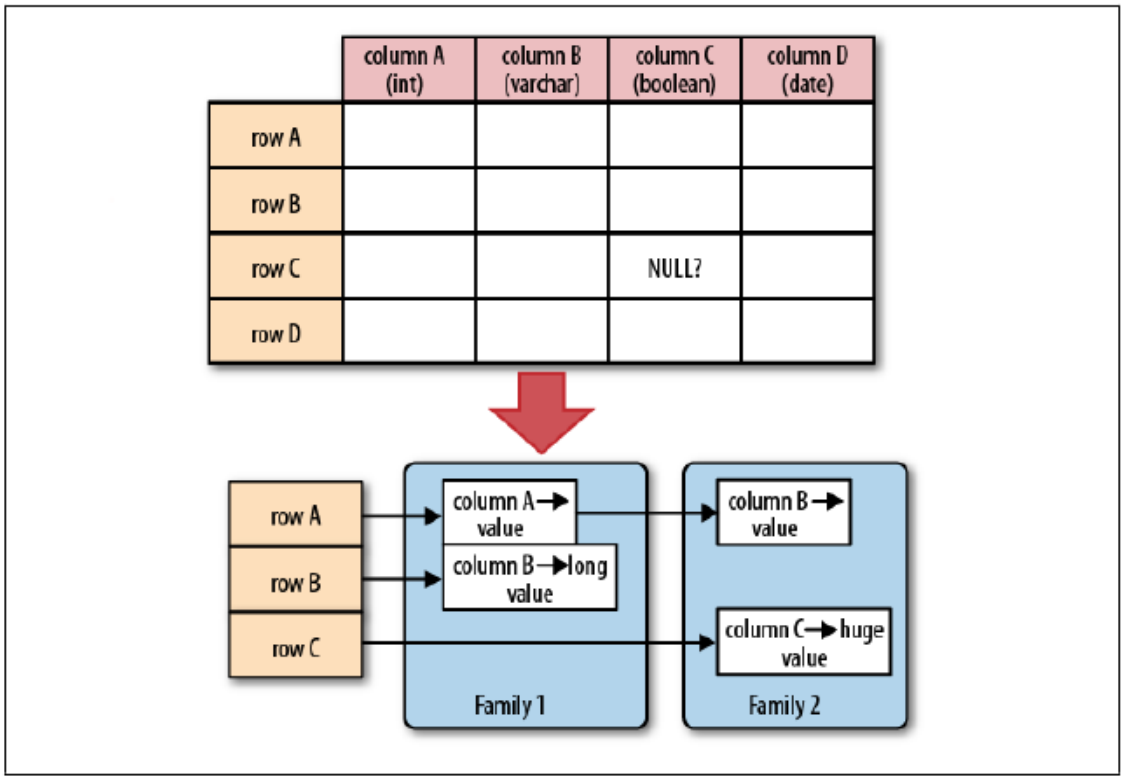
Διατηρώντας τα κλειδιά των γραμμών πάντα ταξινομημένα μπορούμε να έχουμε κάτι σαν δεικτοδότηση πρωτεύοντος κλειδιού, όπως μας είναι αυτό γνωστό από τα RDBMS. Επίσης, κάθε κλειδί είναι πάντα μοναδικό. Αυτό σημαίνει πως μπορούμε να έχουμε κάθε κλειδί γραμμής μόνο μια φορά, ή στην αντίθετη περίπτωση απλά θα κάνουμε ανανέωση της ίδιας γραμμής. Τα κλειδιά των γραμμών μπορεί να είναι οποιοσδήποτε αυθαίρετος πίνακας από bytes και δεν πρέπει να αντιστοιχούν υποχρεωτικά σε αναγνώσιμους χαρακτήρες.

Οι γραμμές σχηματίζονται από στήλες, και αυτές με τη σειρά τους ομαδοποιούνται σε ομάδες στηλών. Αυτό βοηθάει στη δημιουργία σημασιολογικών ή τοπικών ορίων ανάμεσα στα δεδομένα και επιπλέον στην εφαρμογή ορισμένων γνωρισμάτων σε αυτά, όπως για

παράδειγμα τη συμπίεση, ή τη δήλωσή τους να μένουν εντός μνήμης. Όλες οι στήλες σε μία οικογένεια στηλών είναι αποθηκευμένες μαζί στο ίδιο χαμηλού επιπέδου αποθήκευσης αρχείου, που λέγεται HFile.

Οι οικογένειες στηλών πρέπει να δηλώνονται κατά τη δημιουργία του πίνακα και δεν θα πρέπει να αλλάζουν πολύ συχνά, ούτε να υπάρχουν πολλές από αυτές. Υπάρχουν κάποιες γνωστές αδυναμίες στην τρέχουσα εφαρμογή που μας αναγκάζουν να κρατάμε τον αριθμό τους σε λιγότερες από δέκα, αν και στην πράξη πολλές φορές είναι πολύ λιγότερες. Το όνομα της οικογένειας στηλών πρέπει να αποτελείται από εκτυπώσιμους χαρακτήρες, μια σημαντική διαφορά με όλα τα άλλα ονόματα ή τις τιμές.

Η παρακάτω εικόνα μας βοηθάει να φανταστούμε πώς είναι οι διαφορετικές γραμμές σε μια τυπική βάση δεδομένων, σε αντίθεση με τον προσανατολισμένο στις στήλες σχεδιασμό της HBase. Θα πρέπει να φανταστούμε τις γραμμές και τις στήλες τοποθετημένες όχι σύμφωνα με το κλασσικό μοντέλο υπολογιστικού φύλλου, αλλά σαν ένα μοντέλο ετικετών, στο οποίο η πληροφορία βρίσκεται κάτω από μια συγκεκριμένη ετικέτα.



Εικόνα 6: Γραμμές και στήλες στην HBase

Κάθε τιμή στήλης, ή αλλιώς κελί, μπορεί είτε να έχει πάρει ένα timestamp απευθείας από το σύστημα είτε να τεθεί σαφώς από το χρήστη. Αυτό μπορεί να χρησιμοποιηθεί, για παράδειγμα, για την αποθήκευση πολλαπλών εκδόσεων μιας τιμής, καθώς αυτή αλλάζει στο πέρασμα του χρόνου. Διαφορετικές εκδόσεις του κελιού αποθηκεύονται σε φθίνουσα σειρά με βάση τα timestamps, επιτρέποντας στο χρήστη να διαβάζει πρώτα την πιο πρόσφατη τιμή. Αυτή είναι μια βελτιστοποίηση που στοχεύει σε πρότυπα ανάγνωσης που ευνοούν τις πιο πρόσφατες τιμές σε σχέση με τις παλιότερες.

Ο χρήστης μπορεί να ορίσει πόσες εκδόσεις μιας τιμής μπορούν να κρατούνται. Επιπλέον,

υπάρχει υποστήριξη για διαγραφή κατηγορημάτων, που επιτρέπουν τη διατήρηση για παράδειγμα μόνο των τιμών που γράφτηκαν την τελευταία βδομάδα. Οι τιμές (ή κελιά) είναι επίσης ανερμήνευτοι πίνακες από bytes, τους οποίους ο χρήστης πρέπει να ξέρει πώς να χειρίζεται.

Είπαμε πρωτύτερα ότι το μοντέλο του Bigtable, όπως εφαρμόστηκε από την HBase είναι μία αραιή, κατανεμημένη, εμμένουσα, πολυδιάστατη, ταξινομημένη απεικόνιση, που δεικτοδοτείται μέσω του κλειδιού της γραμμής, της στήλης και ενός timestamp. Συμπυκνώνοντας την παραπάνω πληροφορία μπορούμε να εκφράσουμε την πρόσβαση στα δεδομένα ως εξής:

(Table, RowKey, Family, Column, Timestamp) → Value

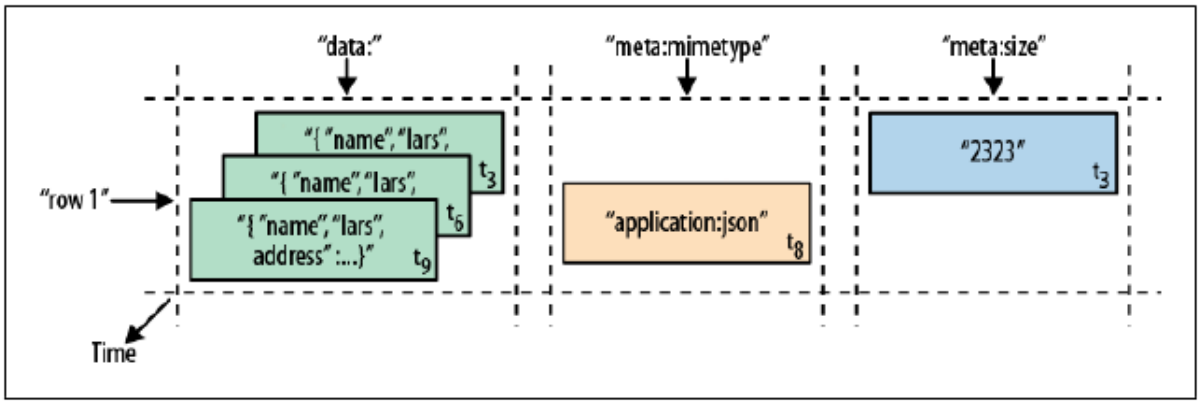
Σε ένα πιο προγραμματιστικό στυλ αυτό θα εκφραζόταν ως:

```
SortedMap<
        RowKey, List<
                SortedMap<
                        Column, List<
                                Value, Timestamp
                        >
                >
        >
>
```

ή με μία γραμμή:

SortedMap<RowKey, List<SortedMap<Column, List<Value, Timestamp>>>>

Το πρώτο SortedMap είναι ένας πίνακας, που περιέχει μια λίστα από οικογένειες στηλών. Οι οικογένειες στηλών περιέχουν ένα άλλο SortedMap, που αναπαριστά τις στήλες και τις σχετικές με αυτές τιμές. Οι τιμές βρίσκονται στην τελευταία λίστα που κρατάει την τιμή και το timestamp που της αντιστοιχεί. Ένα ενδιαφέρον γνώρισμα του μοντέλου είναι ότι τα κελιά μπορεί να υπάρχουν σε πολλαπλές εκδόσεις και διαφορετικές στήλες μπορεί να έχουν γραφτεί σε διαφορετικές στιγμές. Το API, από προεπιλογή, παρέχει μια συνεκτική όψη όλων των στηλών στην οποία επιλέγει την πιο πρόσφατη τιμή κάθε κελιού. Η εικόνα 7 δείχνει ένα κομμάτι μιας συγκεκριμένης γραμμής σε ένα παράδειγμα πίνακα.



Εικόνα 7: Μια προσανατολισμένη στο χρόνο όψη των στοιχείων μιας γραμμής

Το διάγραμμα οπτικοποιεί το στοιχείο του χρόνου χρησιμοποιώντας το σύμβολο t_n σαν το timestamp κατά το οποίο το κελί γράφτηκε. Η αύξουσα σειρά των δεικτών δείχνει ότι οι τιμές προστέθηκαν σε διαφορετικούς χρόνους. Η παρακάτω εικόνα είναι ένας άλλος τρόπος να κοιτάξουμε τα δεδομένα, αυτή τη φορά σε μια μορφή παρόμοια με αυτή των υπολογιστικών φύλλων, στην οποία το timestamp προστέθηκε σε δική του στήλη.

Row Key	Time Stamp	Column "data:"	Column "meta:" "mimetype"	"size"	Column "counters:" "updates"
"row1"	t_3	"{"name":"lars","address":...}"		"2323"	"1"
	t_6	"{"name":"lars","address":...}"			"2"
	t_8		"application/json"		
	t_9	"{"name":"lars","address":...}"			"3"

Εικόνα 8: Τα ίδια στοιχεία της γραμμής με τη μορφή λογιστικού φύλλου

Αν και προστέθηκαν σε διαφορετικούς χρόνους και υπάρχουν σε πολλαπλές εκδόσεις, μπορούμε ακόμη να δούμε τη γραμμή σαν το συνδυασμό όλων των στηλών και των πιο πρόσφατων εκδόσεών τους – με άλλα λόγια το μεγαλύτερο t_n κάθε στήλης. Υπάρχει επίσης ένας τρόπος να ζητήσουμε τις τιμές για ένα συγκεκριμένο timestamp, ή ακόμα και περισσότερες από μία εκδόσεις τη φορά.

Η πρόσβαση στα δεδομένα των γραμμών είναι ατομική και περιλαμβάνει οποιονδήποτε αριθμό από στήλες μπορεί να διαβαστεί ή να γραφτεί. Δεν υπάρχει όμως περαιτέρω εγγύηση ή κάποιο γνώρισμα συναλλαγών που να απλώνεται σε πολλαπλές γραμμές ή μεταξύ πινάκων. Η ατομική πρόσβαση είναι επίσης ένας παράγοντας που συνεισφέρει στην αυστηρή συνέπεια αυτής της αρχιτεκτονικής, αφού κάθε συντρέχων χρήστης που θέλει να διαβάσει ή να γράψει μπορεί να κάνει ασφαλείς υποθέσεις σχετικά με την κατάσταση της γραμμής. Επίσης, η χρήση πολλαπλών εκδόσεων και timestamp μπορεί να βοηθήσει στη συνοχή του στρώματος εφαρμογών.

Συνοψίζοντας για την HBase

Για να συνοψίσουμε τα παραπάνω η HBase είναι ένα κατανεμημένο, εμμένον, αυστηρά συνεπές σύστημα αποθήκευσης με σχεδόν βέλτιστες εγγραφές και άψογες αποδόσεις ανάγνωσης, που κάνει αποδοτική χρήση του χώρου του δίσκου υποστηρίζοντας αλγόριθμους συμπίεσης που μπορούν να επιλεγούν με βάση τη φύση των δεδομένων σε συγκεκριμένες οικογένειες στηλών.

Η HBase επεκτείνει το μοντέλο του Bigtable, το οποίο παρέχει μονοδιάστατη δεικτοδότηση, παρόμοια με το πρωτεύον κλειδί ενός RDBMS, παρέχοντας στη μεριά του σέρβερ αφορμή για την υλοποίηση εύκαμπτων δευτερευόντων δεικτοδοτήσεων.

Δεν υπάρχει δηλωτική γλώσσα ερωτημάτων σαν κομμάτι της υλοποίησης του πυρήνα και έχει περιορισμένη υποστήριξη για συναλλαγές. Η ατομικότητα των γραμμών και οι λειτουργίες διαβάσματος-τροποποίησης-εγγραφής το επανορθώνουν αυτό στην πράξη, αφού καλύπτουν τις περισσότερες περιπτώσεις χρήσης και αφαιρούν την αναμονή ή τις παύσεις που σχετίζονται με deadlocks και εμφανίζονται σε άλλα συστήματα.

Τέλος, η HBase χειρίζεται το μετατοπιζόμενο φορτίο και τις αποτυχίες αποδοτικά και με διαφάνεια από τους χρήστες. Η επεκτασιμότητα είναι ενσωματωμένη και οι clusters μπορούν

να μεγαλώνουν ή να μικραίνουν καθώς το σύστημα είναι σε λειτουργία. Η αλλαγή του cluster δεν περιλαμβάνει καμία περίπλοκη διαδικασία εξισορρόπησης ή τμηματοποίησης, αλλά είναι απολύτως αυτόματη.[42]

2.4 Σχετικές Εργασίες

Στην ενότητα αυτή θα δούμε σχετικές εργασίες με αυτή τη διπλωματική, θα τις περιγράψουμε συνοπτικά και θα καταλήξουμε στις διαφορές με την δική μας εργασία. Λόγω του μεγάλου εύρους των θεμάτων που πραγματευόμαστε θα χωρίσουμε τις σχετικές εργασίες σε δύο μέρη: αυτές που έχουν να κάνουν με την αποθήκευση και δεικτοδότηση χωροχρονικών δεδομένων σε κατανεμημένα συστήματα και αυτές που εκμεταλλεύονται δεδομένα από LBS για να βγάλουν χρήσιμα συμπεράσματα.

2.4.1 Αποθήκευση και δεικτοδότηση χωροχρονικών δεδομένων σε κατανεμημένο σύστημα

Η επεξεργασία κλιμακούμενων δεδομένων, τόσο για μεγάλο όγκο ενημερώσεων όσο και για ανάλυση δεδομένων, είναι μια ενεργή περιοχή ενδιαφέροντος τα τελευταία χρόνια. Τα key-value σχήματα αποθήκευσης είναι εξαιρετικά αποδοτικά στην αντιμετώπιση μεγάλου όγκου ενημερώσεων, που συναντάμε στην περίπτωση των location based services. Παρουσιάζουν όμως έλλειψη αποδοτικής πρόσβασης με βάση πολλαπλά attributes, περιορίζοντας έτσι την εφαρμογή τους στις υπηρεσίες αυτές. Ένα project που αντιμετωπίζει το παραπάνω είναι η MD-HBase[43], που συμπληρώνει τόσο την key-value αποθήκευση όσο και το MapReduce τρόπο επεξεργασίας για πολυδιάστατα δεδομένα. Πρόκειται για ένα κλιμακώσιμο σύστημα διαχείρισης για LBS, που γεφυρώνει το κενό ανάμεσα στη δυνατότητα κλιμάκωσης και τη λειτουργικότητα. Χρησιμοποιεί μία πολυδιάστατη δομή δεικτοδότησης δομημένη πάνω από ένα key-value μοντέλο αποθήκευσης, η οποία επιτρέπει στο σύστημα να διατηρεί υψηλούς ρυθμούς εισαγωγής μεγάλων δεδομένων, ενώ εξασφαλίζει αντοχή στα λάθη και υψηλή διαθεσιμότητα. Το στρώμα δεικτοδότησης προσφέρει αποδοτική επεξεργασία ερωτημάτων για πολυδιάστατα δεδομένα. Χρησιμοποιούνται δύο δομές δεικτοδότησης – τα k-d και τα τετρα-δέντρα – πάνω από ένα κατανεμημένο στρώμα key-value αποθήκευσης. Το πρωτότυπο εφαρμόζεται χρησιμοποιώντας την Hbase και μπορεί να χειριστεί εκατοντάδες χιλιάδες εισαγωγές το δευτερόλεπτο χρησιμοποιώντας έναν cluster 16 κόμβων, ενώ επεξεργάζεται αποδοτικά ερωτήματα πολυδιάστατου εύρους, καθώς και ερωτήματα κοντινότερου γείτονα σε πραγματικό χρόνο με χρόνους απόκρισης που φτάνουν έως και τα εκατοστά του ms.

Την τεχνική της γραμμικοποίησης πολυδιάστατων δεδομένων σε μία διάσταση την έχουν ακολουθήσει εκτός από την MD-HBase και μια σειρά από άλλες εφαρμογές. Για παράδειγμα ο Jensen[44] και οι συνεργάτες του χρησιμοποίησαν Z-ordering και Hilbert καμπύλες σαν space filling curves και για να κατασκευάσουν ένα ευρετήριο στη μορφή B+ δέντρου, χρησιμοποιώντας τις γραμμικοποιημένες τιμές. Ο Tao[45] με την ομάδα του πρότεινε μια αποδοτική τεχνική δεικτοδότησης για αναζήτηση κοντινότερου γείτονα σε υψηλές διαστάσεις χρησιμοποιώντας μια συλλογή από δείκτες B-δέντρων. Αρχικά χρησιμοποίησαν ένα ευαίσθητο στην τοπικότητα hashing για να μειώσουν τη διαστατικότητα των σημείων δεδομένων και έπειτα εφάρμοσαν Z-ordering για να γραμμικοποιήσουν το dataset, που στη συνέχεια δεικτοδοτήθηκε με ευρετήριο B-δέντρου.

2.4.2 Αξιοποίηση δεδομένων από LBS

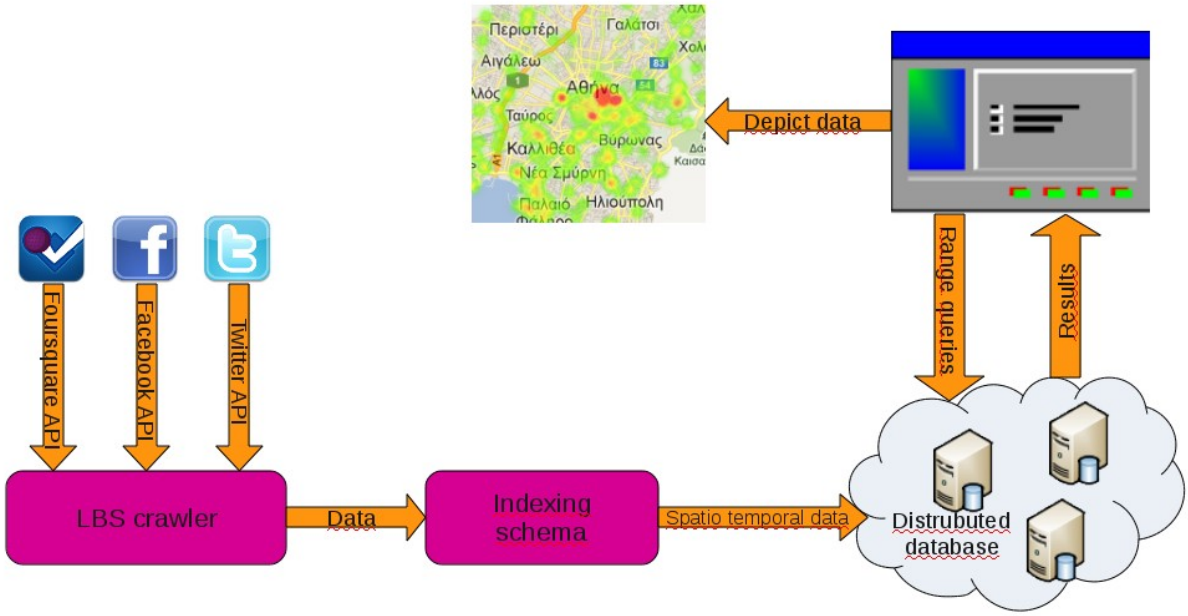
Οι υπηρεσίες βασισμένες στη θέση εισήγαγαν τη γεωγραφική θέση των χρηστών τους σαν γνώρισμα που μπορεί να δημοσιοποιηθεί από τους ίδιους και να αξιοποιηθεί από την υπηρεσία για εξατομίκευση της εμπειρίας που τους προσφέρει. Καθώς ο αριθμός των χρηστών τέτοιων υπηρεσιών αυξάνεται υπάρχει ένα γενικότερο ενδιαφέρον για το πώς θα μπορούσαν να αξιοποιηθούν αυτά τα δεδομένα από τρίτες εφαρμογές.

Η προφανής χρήση είναι αυτή της διαφήμισης ανάλογα με τη θέση του χρήστη, ώστε να γίνει πιο γνωστή μια επιχείρηση, η αξιοποίηση όμως των δεδομένων των υπηρεσιών θέσης δε σταματά εδώ. Η Λεμονιά Ραγιά και ο Michale Deriaz στο paper “Location Based Services for Traffic Management”[46] συλλέγουν δεδομένα από location based services, τα αποθηκεύουν σε μια κεντρική αποθήκη και τα χρησιμοποιούν για να προσφέρουν συμβουλές για την κίνηση στο κέντρο της πόλης. Ιδιαίτερη προσοχή δίνουν στο θέμα της ασφάλειας του συστήματός τους. Στην ίδια κατεύθυνση, το 2003 σε συνέδριο στο Ζάγκρεμπ οι Σταμουλακάτος και Συκάς παρουσίασαν μια αρχιτεκτονική δικτύου για την απόκτηση πληροφορίας κίνησης για location based services[47]. Οι υπηρεσίες θέσης μπορούν επίσης να χρησιμοποιηθούν στην ανάπτυξη του τουρισμού, όπως παρουσιάζει το paper “Location-based Services as a Tool for Developing Tourism in Marginal Regions”.[48]

Κεφάλαιο 3

Αρχιτεκτονική εφαρμογής

Στα πλαίσια της διπλωματικής χρειάστηκε να αναπτύξουμε μία εφαρμογή, η οποία παίρνει δεδομένα από τις τρεις location based services που αναφέραμε προηγουμένως, σχετικά με τον αριθμό και τη θέση των χρηστών τους. Συγκεκριμένα, στόχος της είναι η εκτίμηση του κόσμου που βρίσκεται στα διάφορα μέρη της Αθήνας, σύμφωνα με τα check-in που έχουν κάνει οι χρήστες των υπηρεσιών αυτών ή τη γεωγραφική θέση από την οποία πραγματοποίησαν κάποια ανάρτηση. Για τον σκοπό αυτό, αναπτύξαμε έναν crawler, ο οποίος χρησιμοποιεί τα API και των τριών LBS για να μαζεύει χωροχρονικά δεδομένα, με ερωτήσεις που κάνει προς τις υπηρεσίες ανά διαστήματα μίας ώρας. Αφού η πληροφορία αυτή αποκτηθεί αποθηκεύεται σε ένα σχήμα HBase. Η ενασχόληση με τη συγκεκριμένη βάση μας έδωσε μια καλή εμπειρία πάνω στα κατανεμημένα συστήματα αποθήκευσης, ενώ ταυτόχρονα προσφέρει και δυνατότητα επεκτασιμότητας της εφαρμογής, σε περίπτωση που στη χρήση της συμπεριληφθούν και άλλες πόλεις ή το τρέξιμό της συνεχίσει για μεγάλο χρονικό διάστημα, μαζεύοντας μεγάλες ποσότητες δεδομένων για κάθε ώρα που περνάει. Για την αποδοτική αποθήκευση των δεδομένων στην HBase χρειάστηκε να εξετάσουμε τον τρόπο που θα δεικτοδοτηθεί η πληροφορία εντός της, καταλήγοντας σε δύο σχήματα, από τα οποία θα επιλέξουμε το καλύτερο μέσα από μια σειρά ερωτημάτων που θα τρέξουμε πάνω της στο πειραματικό κομμάτι της διπλωματικής.



Εικόνα 9: Αρχιτεκτονική πλατφόρμας

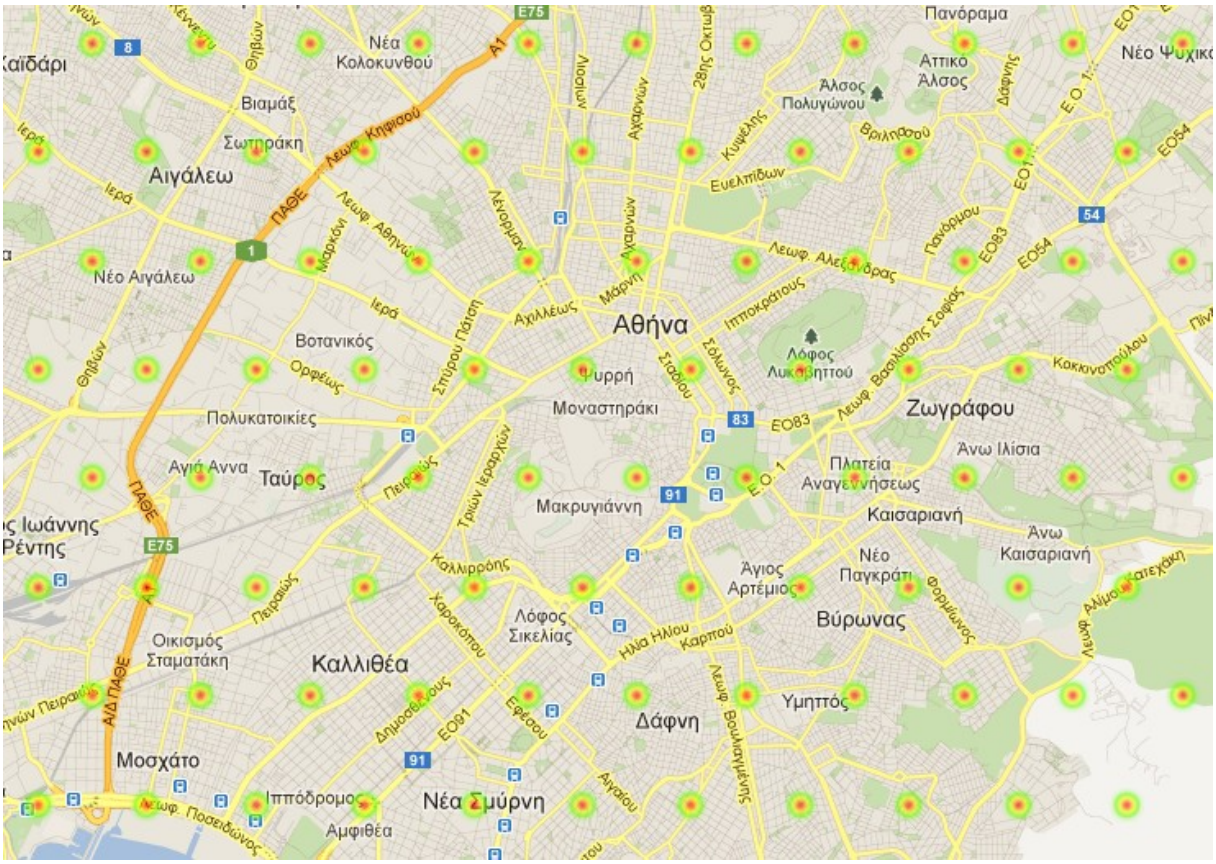
3.1 Crawler

Ο crawler που αναφέραμε παραπάνω χρησιμοποίησε τα API που δίνουν οι υπηρεσίες Facebook, Foursquare και Twitter για τη συγκέντρωση δεδομένων που αφορούν τους χρήστες τους στην Αθήνα. Συγκεκριμένα, κατά τη λειτουργία του μαζεύει πληροφορία σχετικά με τα μέρη που οι χρήστες κάνουν check-in, όσον αφορά το Foursquare, ή τις γεωγραφικές συντεταγμένες με τις οποίες μαρκάρεται μία ανάρτηση, στην περίπτωση του Facebook. Όσον αφορά το Twitter, συλλέγεται μέσω των tweet των χρηστών τόσο η γεωγραφική τους θέση όταν έκαναν την ανάρτηση, όσο και η ώρα που πραγματοποιήθηκε αυτή. Τέσσερις είναι οι κύριες κλάσεις που χρησιμοποιήθηκαν και θα περιγραφούν παρακάτω.

ThesisMaster

Η κλάση αυτή είναι ο συντονιστής του crawling, καθορίζει δηλαδή κάθε πότε θα γίνει η συλλογή των δεδομένων και από ποια υπηρεσία. Επίσης είναι η κλάση που αρχικοποιεί την αναζήτηση ορίζοντας σε ποιες γεωγραφικές περιοχές θα ζητηθούν venues. Για να γίνουμε πιο συγκεκριμένοι και οι τρεις υπηρεσίες έχουν σαν κοινό χαρακτηριστικό των API τους κλήσεις που παίρνουν σαν όρισμα ένα γεωγραφικό κέντρο και μια ακτίνα και επιστρέφουν τα venues που υπάρχουν μέσα σε αυτή – ή τα tweets στην περίπτωση του Twitter. Για το λόγο αυτό είναι σημαντικό να δοθούν σαν ορίσματα κέντρα κύκλων που θα καλύπτουν ολόκληρο το λεκανοπέδιο, έτσι ώστε η συγκέντρωση των venues να είναι ομοιογενής. Για να το φροντίσουμε αυτό, δημιουργήσαμε ένα τριγωνικό πλέγμα με κέντρα κυκλικών περιοχών που καλύπτει ολόκληρη την πόλη της Αθήνας και χρησιμοποιείται για να ζητηθούν venues. Κάθε κυκλική περιοχή έχει ακτίνα 500 μέτρα. Τα κέντρα των κύκλων φαίνονται στην παρακάτω εικόνα, με απεικόνιση που έγινε με τη βοήθεια του API του Google Maps, καθώς και μια ακόμη εκδοχή της στην οποία έχει γίνει ζουμ στο κέντρο της πόλης. Να σημειωθεί ότι αυτές οι περιοχές χρησιμοποιήθηκαν μόνο για τα Facebook και Foursquare, καθώς το Twitter δεν έδινε τη δυνατότητα για αρκετές κλήσεις στο API του, που να καλύπτουν όλες αυτές τις περιοχές.

Εικόνα 10: Τριγωνικό πλέγμα κέντρων περιοχών



Εικόνα 11: Λεπτομερής απεικόνιση μέρους των κέντρων των κυκλικών περιοχών

Αφού η ThesisMaster δημιουργήσει τις κυκλικές περιοχές για τις οποίες θα ζητούνται venues – και τα κέντρα τους απεικονίζονται στις εικόνες 9 και 10 – , αρχικοποιεί τις κλάσεις που είναι υπεύθυνες για κάθε μία από τις υπηρεσίες βασισμένες στη θέση που χρησιμοποιήσαμε και καλεί τη μέθοδό τους που συλλέγει δεδομένα here now. Αφού ολοκληρωθεί η συλλογή δεδομένων και για τις τρεις υπηρεσίες ο master “κοιμάται” για όσο χρόνο χρειάζεται, ώστε να συμπληρωθεί μία ώρα από τη στιγμή που ξεκίνησε η συλλογή δεδομένων. Έτσι εξασφαλίζεται ότι μαζεύουμε δεδομένα κάθε ώρα για τα venues και όχι πιο συχνά, χαρακτηριστικό που είναι απαραίτητο όπως θα δούμε στον τρόπο που υπολογίζουμε το here now για τα places του Facebook. Αν το crawling των LBS κρατήσει παραπάνω από μια ώρα, τότε ο master ξεκινά τη συλλογή δεδομένων χωρίς να “κοιμηθεί”, αφού η μία ώρα που θέλουμε να μεσολαβεί ανάμεσα σε δύο crawl έχει ήδη περάσει. Ο crawler έτρεξε σε μηχανήματα του εργαστηρίου για διάρκεια μίας εβδομάδας, από την Πέμπτη 20/9/12 μέχρι την επόμενη Πέμπτη 27/9/12 μαζεύοντας δεδομένα και αποθηκεύοντάς τα σε HBase που βρισκόταν σε 8 μηχανήματα. Ο master έτρεξε κεντρικά και ο μέσος χρόνος που διαρκούσε το crawling και για τις τρεις υπηρεσίες που ασχοληθήκαμε ήταν περίπου 40 λεπτά κάθε ώρα. Παρακάτω φαίνεται ένας χάρτης στον οποίο παριστάνεται η πυκνότητα των venues που συγκεντρώθηκαν.

Εικόνα 12: Χάρτης πυκνότητας των venues που συγκεντρώθηκαν από τον crawler

FacebookAreasSearch

Πρόκειται για την κλάση που είναι υπεύθυνη για τα ερωτήματα στο Facebook, την απόκτηση πληροφορίας από αυτό και την αποθήκευσή της στην HBase. Παρότι το API του Facebook είναι ιδιαίτερα εύχρηστο, επειδή για τη διπλωματική χρησιμοποιήσαμε τη γλώσσα προγραμματισμού Java, για τα ερωτήματα και την αλληλεπίδραση με το Facebook χρησιμοποιήσαμε τη βιβλιοθήκη restfb. Η βιβλιοθήκη αυτή προσφέρει ένα API απλό και ευέλικτο, συνδυάζει το Facebook Graph και το Old REST API και είναι γραμμένο σε Java. Είναι επίσης ανοιχτού κώδικα και εκδίδεται κάτω από τους όρους της άδειας MIT.

Επιλέξαμε το restfb γιατί χαρακτηρίζεται από:

- το μινιμαλιστικό public API του
- ιδιαίτερα μεγάλη επεκτασιμότητα
- την ευρωστία του στην αντιμετώπιση των συχνών αλλαγών του Facebook API
- το απλό metadata-drive configuration
- και τις μηδενικές εξαρτήσεις.

Αντίθετα παρότι υστερεί στα παρακάτω δεν θεωρήσαμε ότι αποτελούν εμπόδιο για τη χρήση του:

- υποστήριξη για non-Graph/non-REST API κομμάτια της πλατφόρμας του Facebook
- παροχή μηχανισμού απόκτησης session keys ή OAuth access tokens
- χρήση XML σαν σχήμα μεταφοράς δεδομένων επιπρόσθετα του JSON
- επίσημα τυποποιημένες εκδόσεις από όλες τις μεθόδους, λάθη σφαλμάτων κτλ του API του Facebook

Μία από τις πιο σημαντικές κλάσεις του API αυτού είναι η FacebookClient, η οποία μέσω της μεθόδου της fetchConnection επιτρέπει την απόκτηση από την υπηρεσία του Facebook όλων των places που βρίσκονται σε μία ακτίνα γύρω από κάποιο κέντρο που δίνει ο χρήστης σε μια μορφή λίστας. Εμείς κάναμε ερωτήσεις για όλες τις κυκλικές περιοχές που περιγράψαμε στο κομμάτι του master και δημιουργήσαμε έτσι λίστες με venues, που συνδέονται με κάθε περιοχή.

Το επόμενο βήμα ήταν να βρούμε πόσα άτομα βρίσκονται σε κάθε venue κάθε στιγμή και να το καταγράψουμε στη βάση. Η δυνατότητα αυτή όμως δεν προσφέρεται άμεσα από το API του Facebook, δηλαδή δεν μπορούμε να κάνουμε άμεσο ερώτημα για το here now κάποιου place με γνωστό id. Αυτό το οποίο μπορούμε να κάνουμε ωστόσο είναι να ζητήσουμε τα συνολικά check-in από τη στιγμή που κατασκευάστηκε η σελίδα του εν λόγω μέρους. Έτσι, ζητώντας κάθε μία ώρα τα συνολικά check-in και βρίσκοντας τη διαφορά από δύο συνεχόμενα μπορούμε να ξέρουμε πόσα άτομα έφτασαν σε αυτό το venue την ώρα που πέρασε. Πληροφορία για το πόση ώρα θα καθίσει ένα άτομο σε ένα place δεν υπάρχει, αφού δεν δίνεται η δυνατότητα “check-out”, οπότε υποθέσαμε ότι ένας φυσιολογικός αριθμός για να καθίσει κάποιος σε ένα μέρος είναι οι δύο ώρες. Για αυτές τις δύο ώρες λοιπόν μετράμε πόσα χρήστες έχουν συνολικά προσέλθει και τους αθροίζουμε σε ένα δικό μας here now.

Παρατηρήσαμε επίσης πως υπάρχουν places με συνολικά check-in που κινούνται σε μονοψήφιους αριθμούς και αφορούν κυρίως γραφεία ή γενικότερα επιχειρήσεις που προσφέρουν υπηρεσίες – όπως για παράδειγμα ένα τουριστικό γραφείο – των οποίων τα check-in ίσως έγιναν από τον ιδιοκτήτη και τους εργαζόμενους όταν πρωτοπροσφέρθηκε η δυνατότητα, αλλά η χρήση της υπηρεσίας αργότερα εγκαταλείφθηκε. Επειδή οι κλήσεις στο Facebook για κάθε venue γίνεται ξεχωριστά και κλήσεις για venues που δεν μας δίνουν ουσιαστική πληροφορία μπορεί να καθυστερήσει αισθητά τη λειτουργία του crawler, αποφασίσαμε να μην συμπεριλάβουμε places που έχουν λιγότερα από 20 check-in από τη στιγμή δημιουργίας της σελίδας.

Αφού έχουμε αποφασίσει με ποια venues θα ασχοληθούμε και τα έχουμε αποθηκεύσει σε μία λίστα, χρησιμοποιούμε για κάθε ένα από αυτά τη μέθοδο fetchObject του FacebookClient της restfb και παίρνουμε για αυτό πληροφορίες που περιέχουν μεταξύ άλλων τα συνολικά check-in. Αφού υπολογίσουμε όπως είπαμε το here now αποθηκεύουμε την πληροφορία αυτή στην HBase.

OAuth access tokens

Κατά την περιγραφή του restfb κάναμε μια αναφορά στα OAuth access tokens, οπότε εδώ είναι μια καλή ευκαιρία εδώ να μιλήσουμε για την πιστοποίηση στα πλαίσια του Facebook.

Η πιστοποίηση είναι αυτή που δίνει στην εφαρμογή τη δυνατότητα να γνωρίζει την ταυτότητα ενός χρήστη του Facebook και να διαβάζει και να γράφει δεδομένα μέσω του Facebook API. Η πλατφόρμα του Facebook χρησιμοποιεί το OAuth 2.0 για πιστοποίηση και εξουσιοδότηση, που αποτελεί την επόμενη εξέλιξη του OAuth.

Το OAuth δημιουργήθηκε στα τέλη του 2006 και αποτελεί ένα ανοιχτό πρότυπο για πιστοποίηση. Επιτρέπει στους χρήστες να μοιράζονται τους ιδιωτικούς πόρους τους (για παράδειγμα φωτογραφίες, βίντεο, λίστες επαφών), που είναι αποθηκευμένες σε μια σελίδα με κάποια άλλη σελίδα, χωρίς να είναι αναγκασμένες να ανταλλάξουν τα πιστοποιητικά τους. Αντίθετα για το λόγο αυτό παρέχονται username και password tokens. Κάθε token παρέχει πρόσβαση σε μια συγκεκριμένη σελίδα για συγκεκριμένους πόρους και για συγκεκριμένη διάρκεια. Για παράδειγμα ένα token θα μπορούσε να παρέχει πρόσβαση σε μια σελίδα για επεξεργασία βίντεο, παρέχοντας πρόσβαση μόνο στα βίντεο ενός συγκεκριμένου άλμπουμ και μόνο για τις επόμενες δύο ώρες. Αυτό επιτρέπει στο χρήστη να παρέχει σε μια τρίτη σελίδα πρόσβαση στις πληροφορίες του που είναι αποθηκευμένες σε κάποιον άλλο πάροχο υπηρεσιών, χωρίς να μοιραστεί δικαιώματα πρόσβασης σε όλη την έκταση των δεδομένων του.

Το Oauth 2.0 επικεντρώνεται στην απλότητα ανάπτυξης εφαρμογών στη μεριά του client, ενώ παρέχει απλή ροή εξουσιοδότησης για εφαρμογές web, εφαρμογές desktop, κινητά τηλέφωνα και οικιακές συσκευές. Εκτός από το Facebook το χρησιμοποιούν και μεγάλες εταιρείες όπως η Google και η Microsoft.

Στο Facebook μια εφαρμογή μπορεί να αποκτήσει ένα user acess token μέσω του Oauth 2.0, ζητώντας από το χρήστη να την εξουσιοδοτήσει. Με τον τρόπο αυτό αποκτά πρόσβαση στις βασικές πληροφορίες του χρήστη ενώ αν θέλει να διαβάσει ή να γράψει επιπρόσθετα στοιχεία από το Facebook πρέπει να κάνει αίτημα για επιπρόσθετες άδειες.

Όταν μια εφαρμογή πάρει ένα acess token από το Facebook αυτό θα είναι άμεσα έγκυρο και χρήσιμο σε αιτήματα στο API για κάποια χρονική περίοδο που ορίζεται από το Facebook. Αφού η περίοδος αυτή περάσει, το acess token θεωρείται λήξαν και ο χρήστης θα πρέπει να πιστοποιηθεί ξανά ώστε η εφαρμογή να αποκτήσει ένα καινούριο token. Η περίοδος για την οποία είναι έγκυρο ένα token εξαρτάται από τον τρόπο με τον οποίο αυτό παράχθηκε.

Υπάρχουν επίσης γεγονότα που μπορούν να προκαλέσουν την ακύρωση του token ακόμα κι αν δεν έχει παρέλθει ακόμα η περίοδος κατά την οποία είναι έγκυρο. Τέτοια γεγονότα είναι η αλλαγή password από τη μεριά του χρήστη ή η ανανέωση του App Secret από τη μεριά της εφαρμογής. Για τον λόγο αυτό ο χειρισμός των διάφορων tokens και των χρονικών στιγμών κατά τα οποία αυτά λήγουν θεωρείται ουσιώδης για τη δημιουργία εύρωστων εφαρμογών.

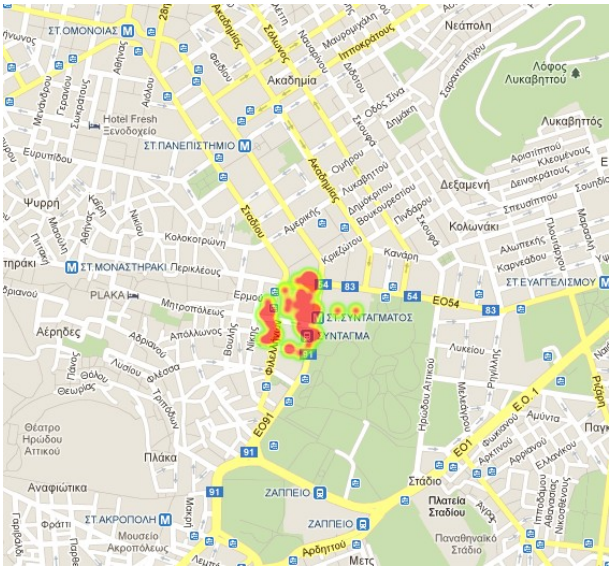
Ο μέσος χρόνος ζωής ενός user access token είναι περίπου μία ώρα και συνδέεται άμεσα με το χρήστη ο οποίος το δίνει σε κάποια εφαρμογή. Ωστόσο, εμείς στον crawler δεν θέλουμε στοιχεία που να σχετίζονται με κάποιους χρήστες συγκεκριμένα, αλλά τα συνολικά check-in για κάποια places, για τα οποία δεν απαιτείται token συνδεδεμένο με κάποιο χρήστη. Για το λόγο αυτό δημιουργήσαμε μία “dummy” εφαρμογή για το Facebook, η οποία μας εξασφάλισε ένα app access token, με το οποίο παίρναμε τα δεδομένα που χρειαζόμασταν χωρίς κάποιον περιορισμό χρόνου.

FoursquareAreasSearch

Αυτό είναι το όνομα της κλάσης που κάνει ερωτήματα στο Foursquare. Αρχικοποιείται όπως και η αντίστοιχη για το Facebook με τις περιοχές στις οποίες θα γίνουν τα ερωτήματα, τις οποίες αποθηκεύει τοπικά σε μια λίστα. Έπειτα, βρίσκει για κάθε περιοχή τα venues που της αντιστοιχούν και την πληροφορία here now για κάθε ένα από αυτά, χρησιμοποιώντας το foursquare-api-java, ένα API γραμμένο σε Java, που μοντελοποιεί το API του Foursquare με έναν πιο εύχρηστο για εμάς τρόπο.

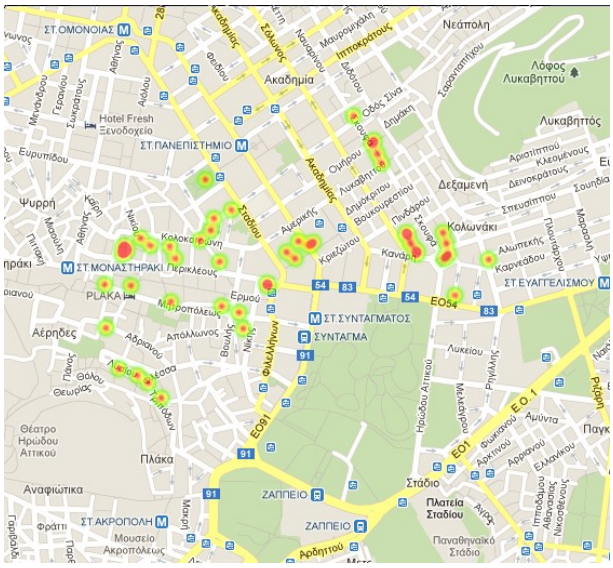
Μια από τις σημαντικότερες κλάσεις του foursquare-api-java είναι η FoursquareAPI, που επεκτείνει την κλάση Object της Java και αποτελεί αρχικό σημείο για το Foursquare. Για να αρχικοποιήσουμε την κλάση αυτή έπρεπε να ξεκινήσουμε μια εφαρμογή για Foursquare, χρησιμοποιώντας το “Foursquare Developers”. Η εφαρμογή την οποία δημιουργήσαμε έχει όνομα Doka Bokos Diploma Thesis και μέσω αυτής πήραμε από την υπηρεσία τα client id και client secret που είναι απαραίτητα για την αλληλεπίδραση με την υπηρεσία. Μετά την απόκτηση των παραπάνω στοιχείων και την αρχικοποίηση της FoursquareAPI μπορούσαμε να χρησιμοποιήσουμε τις μεθόδους της για να πάρουμε venues που αντιστοιχούν σε κάθε περιοχή και πληροφορία για το πόσος κόσμος υπάρχει κάθε στιγμή σε αυτά. Σε αντίθεση όμως με το Facebook, που παρέχει μόνο μία μέθοδο για την ανάκτηση των places μιας area, το Foursquare προσφέρει δύο διαφορετικές μεθόδους που διαφέρουν κυρίως στην ακτίνα των αποτελεσμάτων.

Η πρώτη είναι η search, που υλοποιείται μέσω του foursquare-api-java με την searchVenues, η οποία αποτελεί μέθοδο της FoursquareAPI. Κατά την κλήση της δέχεται μια σειρά από παραμέτρους, από τις οποίες εμείς χρησιμοποιήσαμε τις “near” και “limit” που συμβολίζουν το γεωγραφικό κέντρο και την ακτίνα της περιοχής στην οποία θέλουμε να αναζητήσουμε venues. Η ιδιαιτερότητα της μεθόδου είναι πως δίνει προτεραιότητα σε εγγεγραμμένα στην υπηρεσία μέρη που βρίσκονται κοντά στο κέντρο που δώσαμε, ανεξάρτητα με το αν αυτά έχουν αυτή τη στιγμή πολύ κόσμο ή είναι γενικά δημοφιλή. Έτσι για τις περιοχές ακτίνας 500 μέτρων, παρότι συμπεριλάβαμε έλεγχο για να δούμε αν τα venues που επιστρεφόταν βρισκόταν όντως εντός ακτίνας, δεν χρειάστηκε να αποκλείσουμε κάποιο αυτά. Σαν παράδειγμα παρουσιάζουμε παρακάτω την πυκνότητα και τη θέση των venues που επιστράφηκαν για ένα ερώτημα search στην πλατεία Συντάγματος. Η απόσταση του μακρύτερου σημείου από το κέντρο του ερωτήματος είναι το πολύ 50 μέτρα.



Εικόνα 13: Venues που επιστρέφονται από το ερώτημα search του Foursquare

Αντίθετα η δεύτερη μέθοδος την οποία χρησιμοποιήσαμε, η explore, που υλοποιείται μέσω της venuesExplore, δέχεται μεν σαν παράμετρο ένα κέντρο περιοχής, αλλά είναι περισσότερο ευέλικτη στο θέμα της ακτίνας, δίνοντας περισσότερο βάρος στα προτεινόμενα και δημοφιλή venues και επιστρέφοντάς τα ακόμα και αν δεν βρίσκονται στην ακτίνα που έχουμε ορίσει. Για το λόγο αυτό ο έλεγχος στον crawler για το αν τα venues που πήραμε από τη μέθοδο αυτή βρισκόταν όντως μέσα στην περιοχή που θέλαμε ήταν απαραίτητος, και μόνο μετά την εξακρίβωσή του ζητούσαμε πληροφορία για το here now. Στην Εικόνα 13 φαίνονται τα venues που επιστράφηκαν στο ίδιο ερώτημα με πριν, για την πλατεία Συντάγματος, και είναι εμφανώς πιο διασκορπισμένα από την περίπτωση της search. Μερικά μάλιστα βρίσκονται στο Μοναστηράκι που απέχει περίπου ένα χιλιόμετρο από το Σύνταγμα.



Εικόνα 14: Venues που επιστρέφονται από το ερώτημα explore του Foursquare

Κάθε κλήση search ή explore δεν είναι απαραίτητο να εμφανίσει τα ίδια αποτελέσματα που έφερε στην προηγούμενη κλήση της και έτσι ο έλεγχος για το αν τα venues που παίρνει ο crawler βρίσκονται όντως εντός ακτίνας είναι απαραίτητος για κάθε κλήση των δύο παραπάνω μεθόδων. Για να επιταχύνουμε τη διαδικασία του ελέγχου, για κάθε περιοχή κρατούσαμε λίστες με τα venues που έχουν συναντηθεί και ανήκουν εξακριβωμένα σε αυτές. Έτσι με κάθε νέο venue που συναντούσαμε έπρεπε απλά να δούμε αν ανήκει στη λίστα και αν όντως υπήρχε σε αυτή ξέραμε πως ανήκει στην περιοχή και παίρναμε την πληροφορία του here now για αυτό. Επίσης, επειδή μέρη που δεν ανήκουν στην περιοχή εμφανίζονται επανειλημμένως – κυρίως λόγω της explore – είτε για λόγους δημοφιλίας, είτε λόγω γειτνίασης με το κέντρο της περιοχής, κρατούσαμε επίσης λίστα με τα venues που δεν ανήκουν στην περιοχή και έχουν συναντηθεί, ώστε να μην είναι απαραίτητος ο γεωγραφικός έλεγχος όταν συναντηθούν venues που δεν ανήκουν στην πρώτη λίστα.

Μετά την εξακρίβωση ότι ένα venue ανήκει όντως στην περιοχή που εξετάζουμε, ο crawler παίρνει την πληροφορία για το here now του και την αποθηκεύει στη βάση, ενώ κατά την πρώτη εισαγωγή του στην HBase αποθηκεύεται και πληροφορία σχετικά με την ονομασία, το είδος του και την υπηρεσία από την οποία το πήραμε.

TwitterAreasSearch

Για την αναζήτηση χωροχρονικής πληροφορίας μέσω του Twitter αναπτύξαμε την κλάση με το όνομα TwitterAreasSearch, που αρχικοποιείται με το γνωστό μοτίβο μιας λίστας κυκλικών περιοχών, ακτίνας 500 μέτρων. Επειδή όμως το Twitter θέτει αυστηρό όριο στον αριθμό των request που μπορεί να ικανοποιήσει σε διάστημα μίας ώρας και δεν είναι αρκετός για την εξερεύνηση όλων των περιοχών, χρησιμοποιήσαμε μόνο ένα υποσύνολό τους.

Για τα ερωτήματα στην υπηρεσία κάναμε χρήση του γραμμένου σε Java API twitter4j, που κάνει ευκολότερη την ανάπτυξη του κώδικα και ενσωματώνεται χωρίς προβλήματα στα πλαίσια του crawler. Βασικό ρόλο παίζει η κλάση Twitter, η οποία με τη μέθοδό της search μας δίνει τη δυνατότητα εξερεύνησης των tweets που αναρτήθηκαν από τους χρήστες της υπηρεσίας. Μάλιστα μπορεί να παραμετροποιηθεί, ώστε να κάνει ερωτήματα για κυκλικές περιοχές ακτίνας που δίνει ο χρήστης, κάτι που κάνει ακόμη πιο εύκολη την ανεύρεση των αναρτήσεων για τις περιοχές της λίστας αρχικοποίησης. Επειδή ο crawler τρέχει ανά μία ώρα, ζητούσαμε μόνο τα tweets που έγιναν στο πλαίσιο της ώρας αυτής.

Επίσης, αφού πάρουμε από το API τη λίστα με τα tweets της τελευταίας ώρας για μια περιοχή που μας ενδιαφέρει, υπάρχουν μερικά ακόμη πράγματα που πρέπει να προσέξουμε. Για παράδειγμα εμφανίζεται συχνά το φαινόμενο ένας χρήστης που έκανε μία ανάρτηση από το κινητό του τηλέφωνο σε κάποιο μέρος να συνεχίζει να αναρτεί tweets κατά το διάστημα που παραμένει στην τοποθεσία αυτή. Αν δεν λάβουμε το παραπάνω υπόψιν, ο crawler εσφαλμένα θα μετρήσει τις αναρτήσεις σαν διαφορετικούς χρήστες και θα αποθηκεύσει στη βάση αριθμό χρηστών μεγαλύτερο από αυτό που αντιστοιχεί στην πραγματικότητα. Για το λόγο αυτό κρατούσαμε για κάθε λίστα με tweets και μια άλλη λίστα, στην οποία προσθέταμε τα id των χρηστών την πρώτη φορά που εμφανίζονταν. Έτσι πριν μετρήσουμε ένα tweet κοιτούσαμε αν ο χρήστης που το ανάρτησε έχει ξαναεμφανιστεί και αν κάτι τέτοιο συνέβαινε, δεν το λαμβάναμε υπόψιν. Ένα ακόμη χαρακτηριστικό της υπηρεσίας που θα μπορούσε να οδηγήσει σε ψευδή αποτελέσματα είναι η δυνατότητα ανάρτησης ενός check-in που έχει γίνει μέσω Foursquare. Επειδή ο crawler μετράει ήδη τα check-in που έχουν γίνει μέσω αυτής της LBS στην ίδια περιοχή θα ήταν λάθος να μετρήσουμε tweets των ίδιων χρηστών για το ίδιο check-in. Για το λόγο αυτό εξετάζαμε το κείμενο κάθε ανάρτησης, αφού αναρτήσεις που αντιστοιχούν σε check-in του Foursquare ξεκινούν με τη συντομευμένη ηλεκτρονική διεύθυνση αυτής της υπηρεσίας, και αν αντιλαμβανόμασταν κάτι τέτοιο δεν λαμβάναμε

υπόψιν το tweet.
Αξίζει να παρατηρήσουμε ότι οι συντεταγμένες στις οποίες γίνεται η ανάρτηση μπορεί να βρίσκονται είτε σαν πεδία συντεταγμένων με τη μορφή μιας κλάσης , μέσα πάντα από τις κλάσεις που υλοποιεί το API του Twitter το twitter4j και συγκεκριμένα την GeoLocation, είτε σαν κείμενο με τη μορφή “γεωγραφικός πλάτος,γεωγραφικό μήκος”. Στη δεύτερη περίπτωση δεν είχαμε την επιστροφή μιας ειδικής κλάσης που περιέχει double αριθμούς, αλλά ένα string, που έπρεπε να επεξεργαστούμε κατάλληλα για να ανακαλύψουμε τις συντεταγμένες που έκρυβε. Υπήρχε ακόμη η περίπτωση να μην συνοδεύεται ένα tweet από συντεταγμένες τοποθεσίας, παρά το ότι μας το επέστρεψε το ίδιο το Twitter. Σε αυτή την περίπτωση απλά το προσπερνούσαμε.
Αφού είχε γίνει όλη η επεξεργασία ώστε να βεβαιωθούμε ότι ένα tweet είναι έγκυρο και πρέπει να το μετρήσουμε, αποθηκευόταν στην HBase με τις τρεις μορφές που έχουμε περιγράψει ως τώρα, δηλαδή σε έναν πίνακα με κλειδί τις συντεταγμένες του όπως τις παίρνουμε από την υπηρεσία, σε έναν με κλειδί τη z-τιμή των συντεταγμένων του και σε έναν τελευταίο που δεικτοδοτείται με το timestamp κατά το οποίο το tweet αναρτήθηκε.

3.2 Αποθήκευση δεδομένων

Για την αποθήκευση και επεξεργασία των δεδομένων που πήραμε μέσω του crawler χρησιμοποιήσαμε την HBase. Συγκεκριμένα χρησιμοποιήσαμε δύο πίνακες για να αποθηκεύουν την πληροφορία που παίρναμε για τον κόσμο που βρίσκεται κάθε στιγμή σε ένα venue με βάση τις γεωγραφικές συντεταγμένες του venue αυτού και έναν ακόμη που αποθηκεύει την παραπάνω πληροφορία με βάση τη χρονική στιγμή που την πήραμε – τη χρονική στιγμή δηλαδή για την οποία η πληροφορία του here now ήταν αληθής στο συγκεκριμένο μέρος – κρατώντας τη γεωγραφική θέση του venue σε δευτερεύουσα θέση. Αυτό έγινε για να εξετάσουμε στη συνέχεια, με διάφορα ερωτήματα στη βάση, ποια μορφή αποθήκευσης οδηγεί σε πιο γρήγορες απαντήσεις και με τι τρόπο μπορούμε να συνδυάσουμε τους παραπάνω πίνακες.

Πίνακες με κλειδί το γεωγραφικό στίγμα

Ο πρώτος πίνακας που δημιουργήσαμε και χρησιμοποιήσαμε στην HBase ήταν ο πίνακας 'venue'. Ο πίνακας αυτός διαθέτει δύο οικογένειες στηλών, την 'info' και την 'here_now'. Κάθε γραμμή του παριστάνει ένα venue και για το λόγο αυτό δεικτοδοτείται με τις γεωγραφικές συντεταγμένες του, τις οποίες παίρνουμε μέσω του crawler από τις location based services με τις οποίες ασχοληθήκαμε. Το κλειδί αποτελείται από έναν πίνακα από bytes, κατά τη πρότυπο της HBase, στον οποίο το κάθε byte αποτελεί ένα δεκαδικό αριθμό των γεωγραφικών συντεταγμένων της γραμμής. Πρώτα εμφανίζεται το γεωγραφικό πλάτος (latitude) και έπειτα το γεωγραφικό μήκος σε μορφή πραγματικών αριθμών με δύο ακέραια ψηφία. Οι δύο αριθμοί είναι χωρισμένοι με κόμμα και πετυχαίνουν μια ταξινόμηση των αποθηκευμένων στοιχείων σε πρώτη φάση κατά γεωγραφικό πλάτος και σε δεύτερη κατά γεωγραφικό μήκος. Έτσι, το σημείο (37.969237,23.704694) θα βρίσκεται πριν από το σημείο (38.455237,23.704694), αφού έχει μικρότερο γεωγραφικό πλάτος, ενώ το (37.969237,23.804694) θα βρίσκεται ανάμεσα στα δύο αυτά σημεία, αφού έχει ίδιο πλάτος με το πρώτο, μεγαλύτερο όμως γεωγραφικό μήκος. Αφού πήραμε μια ιδέα για τον τρόπο που ο πίνακας 'venue' δεικτοδοτείται θα ασχοληθούμε με τα υπόλοιπά του χαρακτηριστικά και θα αφήσουμε την εμβάθυνση στη δεικτοδότηση για αργότερα.

Η οικογένεια στηλών 'info' αποτελεί την οικογένεια στηλών που αποθηκεύει γενικές πληροφορίες για το κάθε venue. Οι πληροφορίες αυτές αφορούν το όνομα του μέρους, όπως

το παίρνουμε από την υπηρεσία που μας το επέστρεψε, την κατηγορία στην οποία ανήκει (για παράδειγμα εστίαση, καφετέρια κτλ) και την εφαρμογή η οποία μας το έδωσε. Θέλοντας να προβλέψουμε το ενδεχόμενο να μας επιστρέψουν το ίδιο venue περισσότερες από μια LBS χρησιμοποιήσαμε το όνομα της υπηρεσίας σαν όνομα στήλης και σαν τιμή βάλαμε την dummy τιμή 'LBS'. Έτσι, το κομμάτι του crawler που αντιστοιχεί στο Facebook και βάζει για πρώτη φορά τα venue στη βάση, εισάγει στην οικογένεια στηλών 'info' του κάθε ένα τη στήλη 'Facebook' με τιμή 'LBS'. Αν τώρα για παράδειγμα και ο κώδικας που αντιστοιχεί στην πρώτη εισαγωγή ενός venue από το Foursquare συναντήσει το ίδιο σημείο και μας επιστρέψει τις ίδιες συντεταγμένες σαν κλειδί, θα εισάγει με τη σειρά του μια στήλη με όνομα 'Foursquare' και τιμή 'LBS'. Οι στήλες όνομα και κατηγορία βέβαια θα διαμορφωθούν από την εφαρμογή η οποία έγραψε τελευταία στη γραμμή, αυτό όμως μικρή σημασία έχει αφού πρόκειται για το ίδιο venue, άρα θα έχει το ίδιο όνομα και θα ανήκει στην ίδια κατηγορία. Σαρώνοντας εμείς αργότερα τη γραμμή του venue αυτού θα μπορούμε να πάρουμε την πληροφορία ότι συναντήθηκε και στις δύο παραπάνω υπηρεσίες και άρα έχει στατιστικά και από τις δύο. Φυσικά το όνομα της εφαρμογής είναι αναγκαστικά ένα από τα “Facebook”, “Foursquare” και “Twitter”, η δομή όμως προσφέρει εύκολη επέκταση σε περίπτωση που θελήσουμε να συμπεριλάβουμε και άλλες εφαρμογές. Ένα πιθανό παράδειγμα φαίνεται παρακάτω, στο οποίο το καφέ-μπαρ Ted's έχει προστεθεί στη βάση τόσο από το Facebook όσο και από το Foursquare.

	info			
37.98101,23.717371	name	category	Foursquare	Facebook
	Ted's	coffe/bar	LBS	LBS

Πίνακας 2: Παράδειγμα εγγραφής του πίνακα 'venue' που αρχικοποιήθηκε από τις υπηρεσίες Foursquare και Facebook. Φαίνεται μόνο η οικογένεια στηλών 'info'

Σε αντίθεση με την οικογένεια στηλών 'info' που περιέχει γενικού περιεχομένου πληροφορίες για τα venues, η οικογένεια 'here_now' είναι αυτή που περιέχει την ουσία της χωροχρονικής πληροφορίας που θέλουμε να αποθηκεύσουμε, το πλήθος δηλαδή των χρηστών που βρίσκονται σε ένα venue κάθε χρονική στιγμή. Σαν όνομα στηλών χρησιμοποιούμε το timestamp της στιγμής που πήραμε τα δεδομένα για το συγκεκριμένο venue και σαν τιμή το here now του venue αυτού, το πόσα δηλαδή άτομα βρίσκονται συνδεδεμένα στην εφαρμογή σε αυτό. Δίνουμε παρακάτω ένα παράδειγμα ενός υποθετικού venue του κέντρου της Αθήνας, για το οποίο παίρνουμε μετρήσεις στις 27 Σεπτεμβρίου του 2012 από τις εννιά το βράδυ μέχρι τα μεσάνυχτα, με τον κόσμο του συνεχώς να αυξάνεται.

	here now			
37.98101,23.717371	12-09-27 21:00	12-09-27 22:00	12-09-27 23:00	12-9-28 00:00
	12	15	20	35

Πίνακας 3: Η οικογένεια 'here now' για μια εγγραφή του πίνακα 'venue' με μετρήσεις για 4

Η μορφή αυτή αποθήκευσης μας βοήθησε πολύ να αντιμετωπίσουμε την περίπτωση που περισσότερες από μία εφαρμογές επιστρέφουν ακριβώς τις ίδιες συντεταγμένες για το ίδιο venue, περίπτωση στην οποία θα έχουμε μόνο μία γραμμή να το αντιπροσωπεύει και θα πρέπει σε αυτή να περιλάβουμε το here now και από τις δύο υπηρεσίες. Αφού λοιπόν ο crawler τρέχει κεντρικά είμαστε σίγουροι πως το timestamp από τις δύο υπηρεσίες θα είναι διαφορετικό και έτσι δε θα υπάρχει επικάλυψη. Για παράδειγμα το παραπάνω υποθετικό venue θα μπορούσε να έχει τις παρακάτω εγγραφές αν μας έδιναν για αυτό πληροφορία και οι τρεις εφαρμογές με τις οποίες ασχοληθήκαμε.

	here now			
37.98101,23.717371	12-09-27 21:01	12-09-27 21:03	12-09-27 21:12	12-09-27 21:18
	12	15	1	1

Πίνακας 4: Η οικογένεια 'here now' για μια εγγραφή του πίνακα 'venue' για την οποία πήραμε χωροχρονική πληροφορία και από τις τρεις LBS σε διάστημα μίας ώρας.

Το παραπάνω αποτέλεσμα θα μπορούσε να έχει προκύψει αν το Facebook μας έδινε πρώτα για αυτό 12 άτομα here now, δύο λεπτά αργότερα ο crawler ζητούσε από το Foursquare την αντίστοιχη πληροφορία και αυτό του απαντούσε 15, ενώ είχαν κάνει tweet δύο άτομα στις 21:12 και 21:18 αντίστοιχα.

Επίσης η δομή αυτή μας βοηθάει στον υπολογισμό των ατόμων που έχουν κάνει check-in συνολικά σε μια περιοχή, αφού το μόνο που έχουμε να κάνουμε είναι να ψάξουμε τα venue που ανήκουν σε αυτή την περιοχή και να φιλτράρουμε τις στήλες που ανήκουν στο χρονικό διάστημα που μας ενδιαφέρει.

Αξίζει να επισημάνουμε ότι παρότι από το Twitter δεν παίρνουμε θέση venues, παρά μόνο τη θέση των tweet που έγιναν μέσω κινητών συσκευών, εισάγουμε τη θέση αυτή σαν venue, με timestamp στήλης το χρόνο που έγινε το tweet και here now την τιμή 1, που παριστάνει τον χρήστη που έκανε την ανάρτηση.

Ο δεύτερος πίνακας που κρατάει πληροφορία δεικτοδοτούμενη με βάση τη γεωγραφική θέση των venues είναι ο 'zBinVenue'. Πρόκειται στην ουσία για τον ίδιο πίνακα, με τις ίδιες ακριβώς στήλες. Το μόνο που αλλάζει είναι τα row-keys, τα οποία αποτελούν τις συντεταγμένες του κάθε venue. Ενώ προηγουμένως στον 'venue' χρησιμοποιούσαμε δεκαδικό σύστημα και ταξινόμηση με βάση το γεωγραφικό πλάτος, εδώ μετατρέπουμε τις συντεταγμένες σε δυαδικούς αριθμούς και έπειτα εφαρμόζουμε πάνω τους την z-order curve, μια καμπύλη που χρησιμοποιείται στην γραμμικοποίηση n-διάστατων δεδομένων (εδώ n=2) στη μία διάσταση με σκοπό γειτονικά σημεία στη μία διάσταση να είναι γειτονικά και στις n διαστάσεις. Λεπτομέρειες για τον τρόπο υλοποίησης της καμπύλης αυτής, τους λόγους για τους οποίους τη χρησιμοποιήσαμε αλλά και τι περιμένουμε από τη χρήση της θα δώσουμε στο επόμενο κεφάλαιο.

Ένας τελευταίος πίνακας που χρησιμοποιήσαμε και έχει σαν κλειδί γεωγραφικές συντεταγμένες είναι ο 'fb_venue'. Τον πίνακα αυτόν τον χρησιμοποιήσαμε λόγω της ιδιομορφίας που έχει το Facebook στον τρόπο που μας δίνει δεδομένα. Όπως είπαμε, για να πάρουμε πληροφορίες για τα places μιας περιοχής, πρέπει να ζητήσουμε για την περιοχή αυτή (της οποίας δίνουμε στην εφαρμογή το γεωγραφικό κέντρο και μια ακτίνα) τα id που αντιστοιχούν στα venue. Έπειτα κρατάμε τα id αυτά στη μνήμη και ζητάμε για το κάθε ένα ξεχωριστά τις πληροφορίες σχετικά με τα check-in. Για να αποφύγουμε να ψάχνουμε από την

αρχή όλα τα places μιας περιοχής, στην περίπτωση που χρειαστεί να επανεκκινήσουμε τον crawler στον πίνακα 'fb_venue' αποθηκεύουμε για κάθε περιοχή σε μια γραμμή τις γεωγραφικές συντεταγμένες όλων των venues που της αντιστοιχούν. Έτσι γλιτώνουμε χρόνο, αφού το scan από την HBase είναι λιγότερο ακριβό από τα χρονοβόρα get στο Facebook. Ο 'fb_venue' λοιπόν έχει σαν κλειδί κάθε γραμμής τις συντεταγμένες της area που η γραμμή αυτή αντιπροσωπεύει, και διαθέτει μόνο μία οικογένεια στηλών, την 'venues', στην οποία αποθηκεύονται τα places που ανήκουν στην περιοχή με όνομα στήλης τις δικές τους συντεταγμένες και με τιμή το id που χρησιμοποιεί το Facebook για αυτά. Ένα παράδειγμα μιας εγγραφής στον πίνακα αυτό θα ήταν η παρακάτω, στο οποίο φαίνεται μια area με τρία places να ανήκουν σε αυτή και το id κάθε place.

	venues		
37.98101,23.717371	37.985137,23.725134	37.983851,23.725049	37.983716,23.732516
	256985789	254936998	58633215

Πίνακας 5: Παράδειγμα εγγραφής του πίνακα 'fb_venue'

Πίνακας με κλειδί το timestamp

Οι πίνακες με τους οποίους ασχοληθήκαμε μέχρι τώρα έχουν σαν κλειδί γεωγραφικές συντεταγμένες, γεγονός που τους κάνει χρήσιμους σε κάποια αναζήτηση με βάση τη θέση. Λόγω των πολλών στηλών τους όμως (που αντιπροσωπεύουν τους χρόνους στους οποίους έχουμε πάρει δεδομένα για τον κόσμο και αυξάνονται κάθε ώρα, όσο τρέχει ο crawler) θα ήταν ιδιαίτερα χρονοβόρο να σαρώσουμε όλο τον πίνακα για να βρούμε τον κόσμο που υπάρχει συνολικά στην Αθήνα για κάποια συγκεκριμένη ώρα. Για να ελέγξουμε τους χρόνους στους οποίους γίνονται τέτοιου είδους ερωτήματα στους παραπάνω πίνακες, αλλά και για να προτείνουμε μια λύση στο πρόβλημα αυτό, χρησιμοποιήσαμε έναν ακόμη πίνακα, τον 'timestamps' που έχει σαν κλειδί το timestamp στο οποίο έγινε κάποιο ερώτημα και μοναδική οικογένεια στηλών τη 'here_now'. Στην οικογένεια αυτή εισάγονται στήλες με όνομα τις συντεταγμένες του κάθε venue και τιμή το here now. Επίσης χρησιμοποιούμε το ίδιο timestamp για venues της ίδιας area, αφού τα αποτελέσματα αυτά μας επιστρέφονται ταυτόχρονα και παρατηρήσαμε ότι ο τρόπος αυτός αποθήκευσης βελτιώνει την ταχύτητα του διαβάσματος, σε αντίθεση με την περίπτωση που κάθε timestamp αντιστοιχεί σε ένα μόνο venue. Μια γραμμή του πίνακα αυτού θα ήταν ως εξής:

	here now		
12-09-22 22:33	37.985137,23.725134	37.983851,23.725049	37.983716,23.732516
	2	10	7

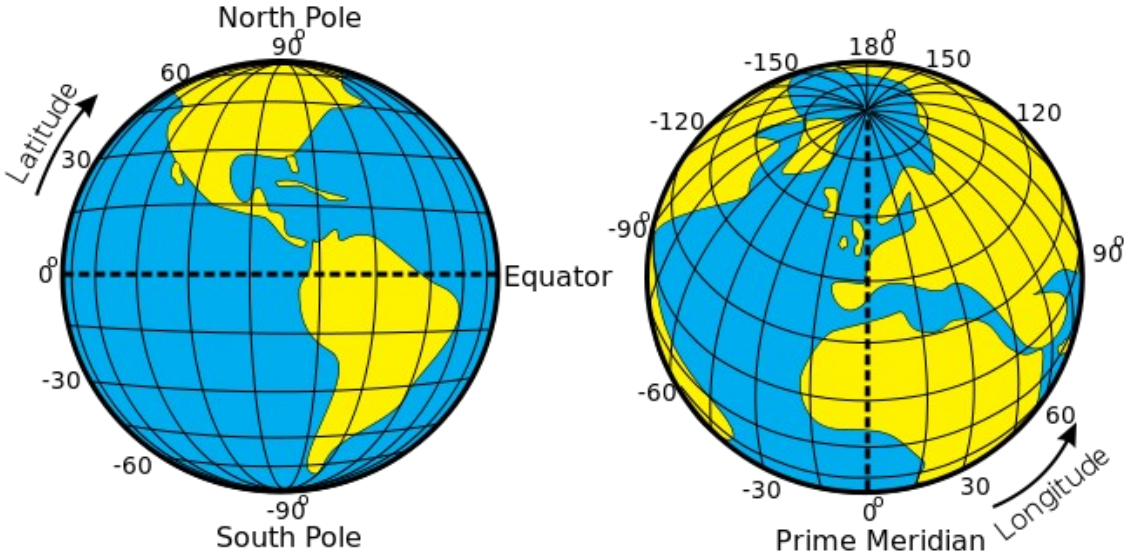
Πίνακας 6: Παράδειγμα εγγραφής του πίνακα 'timestamps'

Λόγω του τρόπου που ταξινομεί η HBase τις γραμμές των πινάκων της, τα timestamps του πίνακα αυτού θα είναι πάντα ταξινομημένα σε αύξουσα σειρά και όταν πραγματοποιήσουμε ένα ερώτημα για όλα τα venues στο πλαίσιο μιας συγκεκριμένης ώρας, ο πίνακας 'timestamps' θα μας δώσει γρήγορα τις εγγραφές του αποτελέσματος μέσω του πρωτεύοντος indexing, ενώ στους πίνακες 'venue' και 'zBinVenue' θα πρέπει να χρησιμοποιήσουμε φίλτρο που θα προσπελάσει όλες τις στήλες και θα αποφασίσει ποιες από αυτές τηρούν τη συνθήκη.

3.3 Δεικτοδότηση δεδομένων

Ένα σημαντικό στοιχείο της σχεδίασης αποτελεί ο τρόπος δεικτοδότησης των δεδομένων που αποθηκεύονται στους πίνακες 'venue' και 'zBinVenue', τους πίνακες δηλαδή που αποθηκεύουν την πληροφορία για το here now κάθε venue, χρησιμοποιώντας σαν κλειδί τις γεωγραφικές συντεταγμένες του.

Για να αναδείξουμε τη σημασία των διαφορετικών σχημάτων δεικτοδότησης θα δώσουμε αρχικά μερικές πληροφορίες για το σύστημα γεωγραφικών συντεταγμένων που χρησιμοποιούν οι location based services που χρησιμοποιήθηκαν στη διπλωματική αυτή και χρησιμοποιήσαμε και εμείς. Οι γεωγραφικές συντεταγμένες είναι δύο μεγέθη τα οποία προσδιορίζουν τη θέση διάφορων σημείων στην επιφάνεια της γης, είτε πρόκειται για κάποιον τόπο είτε για ένα κινούμενο αντικείμενο όπως ένα πλοίο ή ένα αυτοκίνητο. Ως βάση των συντεταγμένων λαμβάνονται ο ισημερινός και ο πρώτος μεσημβρινός, ενώ τις γεωγραφικές συντεταγμένες αποτελούν το γεωγραφικό πλάτος (latitude) και το γεωγραφικό μήκος (longtitude). Το γεωγραφικό πλάτος (latitude ή φ) ενός σημείου που βρίσκεται στην επιφάνεια της γης είναι η γωνία που σχηματίζει η κατακόρυφος του σημείου με το επίπεδο του ισημερινού και χαρακτηρίζεται από Βόρειο ή Νότιο ανάλογα με το ημισφαίριο στο οποίο βρίσκεται το σημείο. Από την άλλη το γεωγραφικό μήκος είναι η στερεή γωνία που σχηματίζεται από το επίπεδο του μεσημβρινού που διέρχεται από το εν λόγω σημείο με το επίπεδο του πρώτου μεσημβρινού και χαρακτηρίζεται Ανατολικό ή Δυτικό. Όλα αυτά συμπυκνώνονται στην εικόνα που δίνουμε παρακάτω.

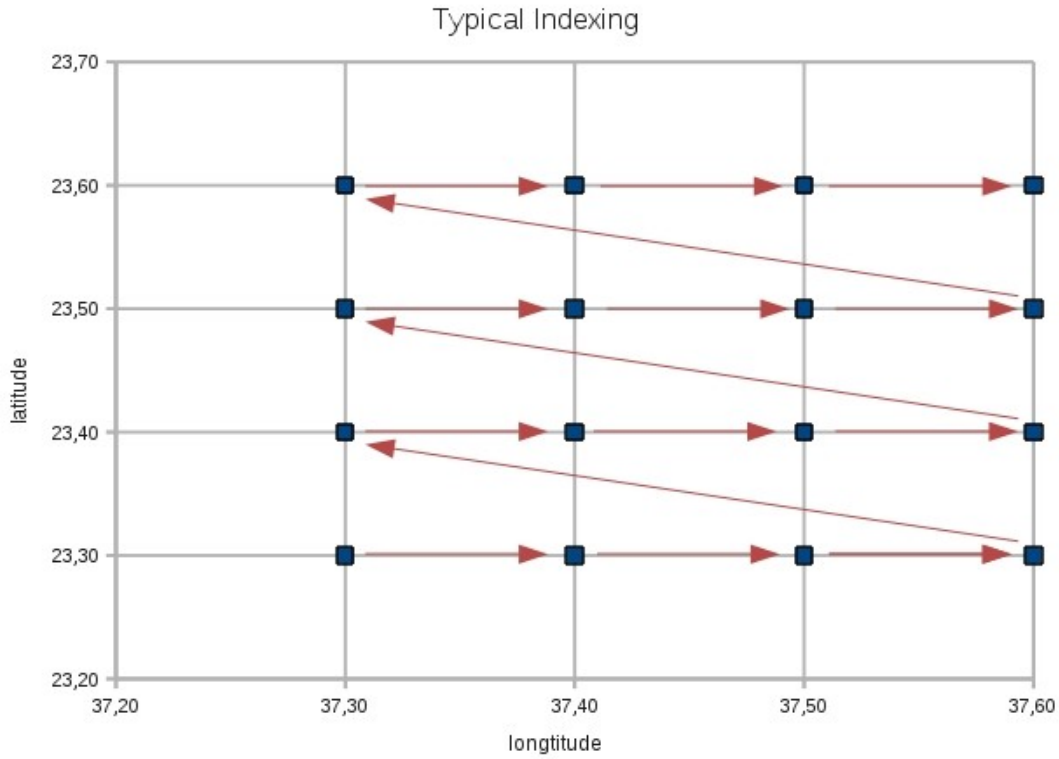


Εικόνα 15: Γεωγραφικό σύστημα συντεταγμένων

Στην περιοχή της Αττικής που μας απασχόλησε όλα τα γεωγραφικά σημεία βρίσκονται ανάμεσα στις 37 και τις 39 μοίρες γεωγραφικό πλάτος και τις 23 και 25 μοίρες γεωγραφικό μήκος. Μια αναπαράσταση της θέσης των venues που πήραμε πάνω στον χάρτη της Αθήνας δείξαμε στην Εικόνα 11. Αξίζει να σημειώσουμε ότι ο χάρτης δεν παρουσιάζει την πυκνότητα του κόσμου σύμφωνα με τα here now, αλλά την πυκνότητα των venues, έτσι ώστε πιο κόκκινες περιοχές να συμβολίζουν περιοχές με περισσότερα εγγεγραμμένα venues και όχι απαραίτητα και περισσότερο κόσμο.

Τυπική δεικτοδότηση

Με τον όρο αυτό περιγράφουμε τη δεικτοδότηση που γίνεται με βάση το γεωγραφικό πλάτος. Είναι ένας εύκολος τρόπος δεικτοδότησης, δεδομένου ότι το key-value σχήμα της HBase με πάντα ταξινομημένη τη λίστα των κλειδιών ενδείκνυται για τη δεικτοδότηση πάνω σε μία διάσταση. Έτσι τα κλειδιά των venues που αποθηκεύονται στη βάση είναι της μορφής (latitude, longtitude), με αποτέλεσμα να γίνεται σε πρώτη φάση ταξινόμηση με βάση το γεωγραφικό πλάτος και σε δεύτερη με βάση το γεωγραφικό μήκος. Αυτό σημαίνει πως σημεία με γεωγραφικό πλάτος μικρότερο από κάποια άλλα θα βρίσκονται πάντα πιο ψηλά στη λίστα με τα κλειδιά, γεγονός που δεν ισχύει όμως και για το γεωγραφικό μήκος των σημείων. Το μόνο που μπορούμε να γνωρίζουμε είναι πως αν κάποια σημεία έχουν το ίδιο latitude, τότε θα βρίσκονται το ένα κάτω από το άλλο στη βάση, ταξινομημένα ως προς το longtitude. Βλέπουμε λοιπόν πως τα γεωγραφικά σημεία αποθηκεύονται στη βάση με μία κατεύθυνση από κάτω προς τα πάνω και από αριστερά προς τα δεξιά όπως φαίνεται στην παρακάτω εικόνα.



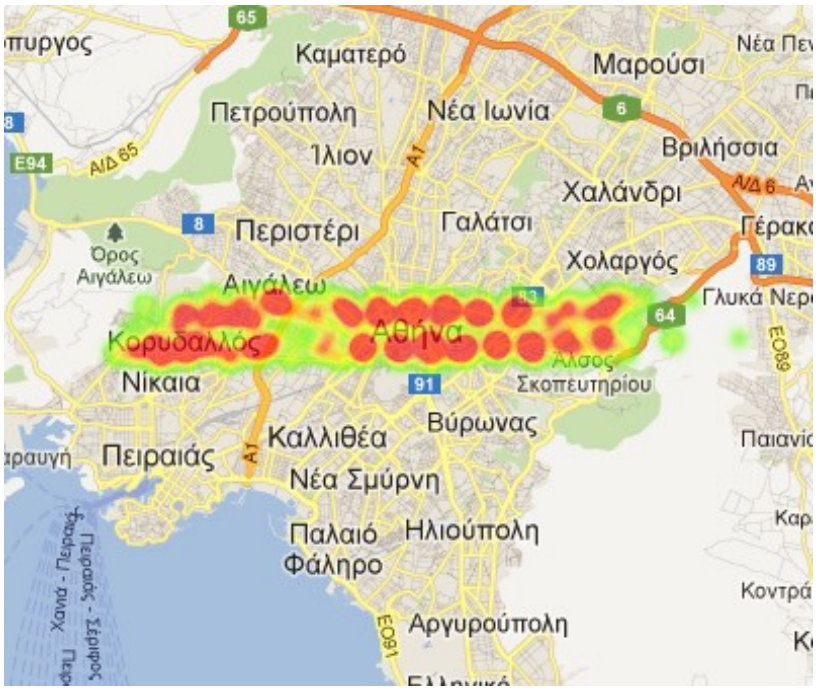
Εικόνα 16: "Τυπική" δεικτοδότηση

Τα σημεία της εικόνας δηλαδή θα αποθηκεύονταν στην HBase στη μορφή:

37.30,23.30
37.30,23.40
37.30,23.50
37.30,23.60
37.40,23.30
37.40,23.40
37.40,23.50
37.40,23.60
37.50,23.30
37.50,23.40
37.50,23.50
37.50,23.60
37.60,23.30
37.60,23.40
37.60,23.50
37.60,23.60

Πίνακας 7: Ταξινόμηση κλειδιών σύμφωνα με την "τυπική" ταξινόμηση

Το σχήμα αυτό μας δίνει τη δυνατότητα να ζητήσουμε από τη βάση ένα εύρος γεωγραφικών πλατών και να πάρουμε τα αποτελέσματα γρήγορα. Αντίθετα, δεν υπάρχει η προφανής δυνατότητα ανάκτησης εγγραφών που αντιστοιχούν σε ένα εύρος γεωγραφικών μηκών, αφού η πληροφορία αυτή βρίσκεται "κρυμμένη" πίσω από τα πλάτη. Για να κάνουμε ένα ερώτημα για ένα εύρος από longtitude θα έπρεπε να μπούμε βαθύτερα στην HBase και να εφοδιάσουμε το ερώτημα με ένα φίλτρο που θα κάνει ανάλυση των γραμμών και θα εφαρμόζει μια σύγκριση πάνω στα longtitudes, πράγμα που θα ήταν χρονοβόρο. Επίσης, στη γενική περίπτωση που θα ζητάμε ένα γεωγραφικό τετράγωνο που θα έχει έναν περιορισμό εύρους τόσο σε γεωγραφικό πλάτος όσο και σε μήκος, τα αποτελέσματα θα είναι "καλά" όσον αφορά τα πλάτη, δηλαδή δε θα μας επιστρέφονται venues που δεν ανήκουν στο range των πλατών που ζητήσαμε, αλλά θα μας επιστρέφονται και τα venues όλων των γεωγραφικών μηκών που έχουν εισηχθεί στη βάση και έχουν latitude που ταιριάζει με το ζητούμενο εύρος. Τα παραπάνω φαίνονται χαρακτηριστικά στην εικόνα 16, που απεικονίζει τα venues της βάσης μας για ένα περιορισμένο γεωγραφικό τετράγωνο.

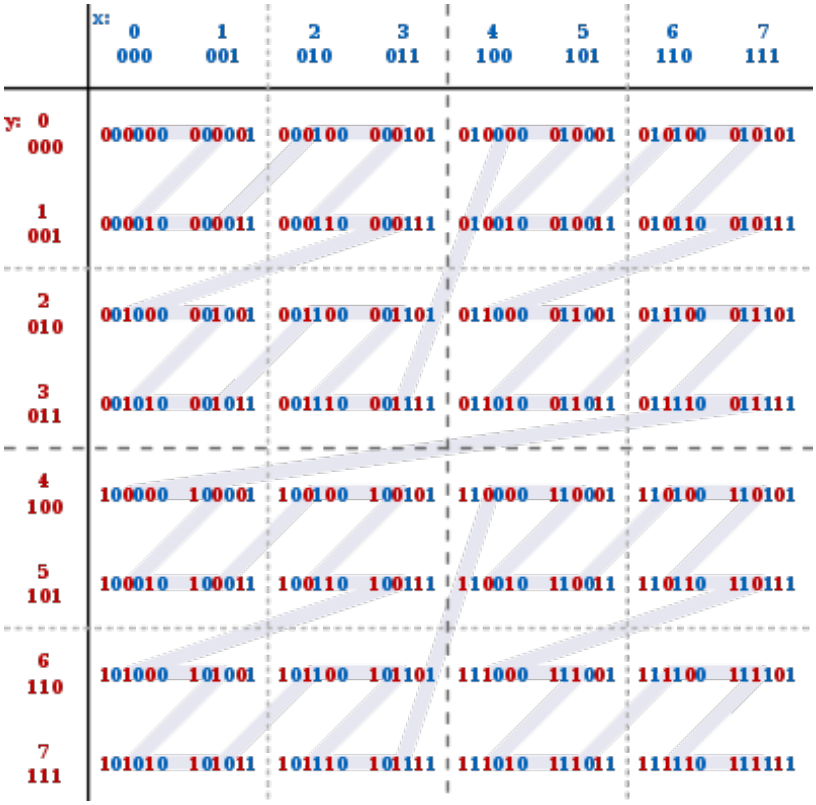


Εικόνα 17: Venues που επιστράφηκαν με "τυπική" δεικτοδότηση

Δεικτοδότηση με την z-order curve

Για να ξεπεράσουμε τις αδυναμίες της δεικτοδότησης που περιγράψαμε παραπάνω και στηρίζεται στην ταξινόμηση με βάση το γεωγραφικό πλάτος, χρησιμοποιήσαμε την z-order curve. Πρόκειται για μια συνάρτηση, γνωστή και σαν Morton order ή κώδικας Morton, που απεικονίζει πολυδιάστατα δεδομένα στη μία διάσταση, ενώ ταυτόχρονα διατηρεί την τοπικότητα των σημείων. Για να το πετύχει αυτό δίνει σε κάθε σημείο του χώρου μία τιμή, η οποία χρησιμοποιείται για την ταξινόμηση των σημείων του χώρου στη μία διάσταση. Η τιμή αυτή, που λέγεται z-τιμή, υπολογίζεται εύκολα περιπλέκοντας μεταξύ τους ανά δύο τα bits της δυαδικής αναπαράστασης των συντεταγμένων του κάθε σημείου. Έτσι οι n αριθμοί που αποτυπώνουν την τιμή του σημείου στις n-διαστάσεις του χώρου μετατρέπονται σε έναν και ταξινομούνται με βάση αυτόν. Από τη στιγμή που γίνεται κάτι τέτοιο, μπορεί να χρησιμοποιηθεί κάθε μονοδιάστατη δομή δεδομένων, όπως δέντρα δυαδικής αναζήτησης, B-δέντρα, skip-lists ή hash-tables για την αποθήκευση των δεδομένων.

Εμάς μας ενδιαφέρει ο δισδιάστατος χώρος και για το λόγο αυτό οι διαστάσεις μας θα είναι n=2. Παρακάτω δίνουμε ένα παράδειγμα, στο οποίο οι δύο διαστάσεις περιγράφονται από τους άξονες x και y και παίρνουν τιμές από το 0 έως το 7.



Εικόνα 18: Z-order curve

Στους δύο άξονες κάτω από τις δεκαδικές τιμές φαίνονται και οι αντίστοιχες δυαδικές, ενώ τα σημεία του x-y επιπέδου απεικονίζουν την z-τιμή του εκάστοτε σημείου. Επίσης φαίνεται με γκρι γραμμή η σύνδεση των σημείων του επιπέδου σύμφωνα με την ταξινόμηση που δημιουργείται από την καμπύλη, η οποία δημιουργεί το χαρακτηριστικό Z σχήμα. Έτσι, η σειρά των σημείων του δισδιάστατου χώρου είναι:

(0,0) -> (1,0) -> (0,1) -> (1,1) -> (2,0) -> (3,0) -> (2,1) -> (3,1) -> (0,2) -> ...

Επειδή για να χρησιμοποιήσουμε την καμπύλη αυτή οι συντεταγμένες των σημείων πρέπει να βρίσκονται σε δυαδική μορφή μετατρέψαμε τα γεωγραφικά πλάτη και μήκη που συλλέξαμε από τον crawler σε δυαδικούς αριθμούς. Επίσης προτιμήσαμε να χρησιμοποιήσουμε ακέραιους αριθμούς και όχι δεκαδικούς, οπότε μετατρέψαμε τις συντεταγμένες των σημείων σε ακέραιους πολλαπλασιάζοντας με τον κατάλληλο αριθμό και έπειτα μετατρέψαμε το νούμερο αυτό σε δυαδικό. Για παράδειγμα οι συντεταγμένες

(37.986151,23.730443)
μετατράπηκαν σε

(37986151,23730443)

και αυτές με τη σειρά τους σε

(11100111110110010101111,10010000110101101101 00).

Έπειτα υπολογίσαμε για κάθε venue την z-τιμή των συντεταγμένων του περιπλέκοντας μεταξύ τους τα μηδέν και ένα (τα οποία αν και συνήθως παριστάνονται σαν bits, εμείς τα αναπαραστήσαμε με ένα byte το καθένα, αφού η διάταξη που εφαρμόζει η HBase στα κλειδιά της γίνεται ανά byte) και τα χρησιμοποιήσαμε σαν κλειδιά για τον πίνακα 'zBinVenue'. Τα υπόλοιπα στοιχεία που μπήκαν στον 'zBinVenue' πίνακα αφορούν τις γενικές πληροφορίες για το κάθε venue και τα here now για κάθε timestamp που πήραμε στις μετρήσεις. Είναι στην ουσία ο ίδιος πίνακας με τον 'venue' και το μόνο που αλλάζει είναι η δεικτοδότηση.

Η z-order curve δεν αρκεί

Παρότι θα περίμενε κανείς ότι η δεικτοδότηση με τις z-τιμές των γεωγραφικών σημείων θα είναι αρκετή για την καλύτερη γραμμικοποίηση των δεδομένων και την επιστροφή λιγότερων false-positive venues όταν ζητάμε κάποιο γεωγραφικό τετράγωνο, κάτι τέτοιο σε πρώτη φάση δεν συνέβη. Για να καταλάβουμε καλύτερα τους λόγους που κρύβονται πίσω από αυτό ας ρίξουμε μια ματιά στην Εικόνα 18.

Longitude Values

Latitude Values	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	00000000 0	00000001 1	00000100 4	00000101 5	00010000 16	00010001 17	00010100 20	00010101 21	01000000 64	01000001 65	01000100 68	01000101 69	01010000 80	01010001 81	01010100 84	01010101 85
1	00000010 2	00000011 3	00000110 6	00000111 7	00010010 18	00010011 19	00010110 22	00010111 23	01000010 66	01000011 67	01000110 70	01000111 71	01010010 82	01010011 83	01010110 86	01010111 87
2	00001000 8	00001001 9	00001100 12	00001101 13	00011000 24	00011001 25	00011100 28	00011101 29	01001000 72	01001001 73	01001100 76	01001101 77	01011000 88	01011001 89	01011100 92	01011101 93
3	00001010 10	00001011 11	00001110 14	00001111 15	00011010 26	00011011 27	00011110 30	00011111 31	01001010 74	01001011 75	01001110 78	01001111 79	01011010 90	01011011 91	01011110 94	01011111 95
4	00100000 32	00100001 33	00100100 36	00100101 37	00110000 48	00110001 49	00110100 52	00110101 53	01100000 96	01100001 97	01100100 100	01100101 101	01110000 112	01110001 113	01110100 116	01110101 117
5	00100010 34	00100011 35	00100110 38	00100111 39	00110010 50	00110011 51	00110110 54	00110111 55	01100010 98	01100011 99	01100110 102	01100111 103	01110010 114	01110011 115	01110110 118	01110111 119
6	00101000 40	00101001 41	00101100 44	00101101 45	00111000 56	00111001 57	00111100 60	00111101 61	01101000 104	01101001 105	01101100 108	01101101 109	01111000 120	01111001 121	01111100 124	01111101 125
7	00101010 42	00101011 43	00101110 46	00101111 47	00111010 58	00111011 59	00111110 62	00111111 63	01101010 106	01101011 107	01101110 110	01101111 111	01111010 122	01111011 123	01111110 126	01111111 127
8	10000000 128	10000001 129	10000100 132	10000101 133	10010000 144	10010001 145	10010100 148	10010101 149	11000000 192	11000001 193	11000100 196	11000101 197	11010000 208	11010001 209	11010100 212	11010101 213
9	10000010 130	10000011 131	10000110 134	10000111 135	10010010 146	10010011 147	10010110 150	10010111 151	11000010 194	11000011 195	11000110 198	11000111 199	11010010 210	11010011 211	11010110 214	11010111 215
10	10001000 136	10001001 137	10001100 140	10001101 141	10011000 152	10011001 153	10011100 156	10011101 157	11001000 200	11001001 201	11001100 204	11001101 205	11011000 216	11011001 217	11011100 220	11011101 221
11	10001010 138	10001011 139	10001110 142	10001111 143	10011010 154	10011011 155	10011110 158	10011111 159	11001010 202	11001011 203	11001110 206	11001111 207	11011010 218	11011011 219	11011110 222	11011111 223
12	10100000 160	10100001 161	10100100 164	10100101 165	10110000 176	10110001 177	10110100 180	10110101 181	11100000 224	11100001 225	11100100 228	11100101 229	11110000 240	11110001 241	11110100 244	11110101 245
13	10100010 162	10100011 163	10100110 166	10100111 167	10110010 178	10110011 179	10110110 182	10110111 183	11100010 226	11100011 227	11100110 230	11100111 231	11110010 242	11110011 243	11110110 246	11110111 247
14	10101000 168	10101001 169	10101100 172	10101101 173	10111000 184	10111001 185	10111100 188	10111101 189	11101000 232	11101001 233	11101100 236	11101101 237	11111000 248	11111001 249	11111100 252	11111101 253
15	10101010 170	10101011 171	10101110 174	10101111 175	10111010 186	10111011 187	10111110 190	10111111 191	11101010 234	11101011 235	11101110 238	11101111 239	11111010 250	11111011 251	11111110 254	11111111 255

Εικόνα 19: Ερώτημα εύρους με z-order curve

Πρόκειται για ένα δισδιάστατο επίπεδο στο οποίο τα σημεία παίρνουν τιμές από το 0 μέχρι το 15. Σε κάθε σημείο του επιπέδου αναγράφεται τόσο η z-τιμή του σημείου αυτού, όσο και η σειρά του σημείου στην ταξινόμηση του επιπέδου. Το κεντρικό τετράγωνο με το κίτρινο χρώμα συμβολίζει ένα τετράγωνο, μέσα από το οποίο θέλουμε να εξετάσουμε τα σημεία και να τα ανακτήσουμε από τη βάση. Επειδή η HBase τα αποθηκεύει σύμφωνα με την z-τιμή τους, για να τα ανακτήσουμε από αυτή θα πρέπει να της ζητήσουμε όλα τα σημεία που βρίσκονται μεταξύ των τιμών 51 και 193. Ανάμεσα σε αυτά τα σημεία όμως βλέπουμε ότι υπάρχουν και πολλά false positive, σημεία δηλαδή των οποίων οι z-τιμές ανήκουν στο εύρος που ζητάμε, τα σημεία όμως δεν αντιστοιχούν στο τετράγωνο που θέλουμε να πάρουμε στο επίπεδο.

Για να μειώσουμε τα false positive σημεία που μας επιστρέφει ένα range query σε ένα τετράγωνο του επιπέδου, η τεχνική που θα ακολουθήσουμε είναι η εξής: θα χωρίσουμε το αρχικό τετράγωνο σε άλλα μικρότερα και θα σκανάρουμε τη βάση για κάθε ένα από αυτά, ώστε να περιορίσουμε τον αριθμό λανθασμένων σημείων που θα επιστραφούν. Ο αλγόριθμος που πραγματοποιεί την ιδέα αυτή αναπτύσσεται παρακάτω. Επίσης, παρουσιάζουμε το τετράγωνο με το ερώτημα εύρους για ευκολία.

00110011 51	00110110 54	00110111 55	01100010 98	01100011 99
00111001 57	00111100 60	00111101 61	01101000 104	01101001 105
00111011 59	00111110 62	00111111 63	01101010 106	01101011 107
10010001 145	10010100 148	10010101 149	11000000 192	11000001 193

Εικόνα 20: Λεπτομέρεια της Εικόνας 18

1. Χωρίζουμε το αρχικό τετράγωνο εύρους σε δύο υπο-τετράγωνα. Για να το κάνουμε αυτό παίρνουμε αρχικά τη δυαδική αναπαράσταση της ελάχιστης (min) και της μέγιστης (max) z-τιμής του ερωτήματος. Για το παράδειγμά μας αυτές είναι οι

   51 = 00110011 = (0101,0101), και
   193 = 11000001 = (1001,1000)

   Το πρώτο πράγμα που πρέπει να κάνουμε είναι να βρούμε το πρώτο πιο σημαντικό ψηφίο στο οποίο τα min και max διαφέρουν. Αυτό θα μας βοηθήσει να βρούμε αν η τομή που θα κάνουμε στο εύρος θα είναι κάθετη ή οριζόντια. Στους παραπάνω αριθμούς το πρώτο πιο σημαντικό ψηφίο που διαφέρει είναι το z_8 το οποίο αντιστοιχεί στο bit y_4. Στην περίπτωση που το πρώτο ψηφίο στο οποίο παρατηρείται αλλαγή βρίσκεται στον y άξονα θα έχουμε οριζόντια τομή, ενώ αν βρίσκεται στον x άξονα η τομή θα είναι κάθετη.

   Στο παράδειγμά μας παρατηρούμε ότι έχουμε οριζόντια τομή, η οποία φαίνεται στο παραπάνω σχήμα με κόκκινη γραμμή και χωρίζει το αρχικό τετράγωνο σε δύο υπο-τετράγωνα, ένα “πάνω” και ένα “κάτω”. Επίσης βλέπουμε ότι όπως στο αρχικό τετράγωνο είχαμε ένα min και ένα max, τα σημεία με z-τιμές 51 και 193 αντίστοιχα, έτσι και στα δύο νέα υπο-τετράγωνα έχουμε δύο νέα ζεύγη min-max, τα σημεία 51-107 και 145-193. Το σημείο 107 – και γενικότερα το σημείο που αποτελεί άνω όριο του νέου υπο-τετραγώνου – θα το ονομάζουμε LitMax και το σημείο 145 – και γενικότερα το νέο κάτω όριο του δεύτερου υπο-τετραγώνου – θα το ονομάζουμε BigMin. Από τη στιγμή που γνωρίζουμε τις z-τιμές των δύο αυτών σημείων μπορούμε να σπάσουμε το ερώτημα του αρχικού τετραγώνου σε δύο νέα ερωτήματα για τα δύο υπο-τετράγωνα, οι τιμές όμως αυτές δεν είναι τόσο προφανείς σε ένα πιο γενικό παράδειγμα. Θα δείξουμε ωστόσο στο επόμενο βήμα πως ο υπολογισμός τους είναι δυνατός.

2. Υπολογισμός των συντεταγμένων και της z-τιμής των LitMax και BigMin. Από τη

στιγμή που έχουμε ανιχνεύσει μία οριζόντια τομή μπορούμε να ξέρουμε πως το LitMax (εδώ 107) θα έχει την ίδια x τιμή με το αρχικό max, δηλαδή το 193. Με τον ίδιο τρόπο μπορούμε να ξέρουμε πως το BigMin θα έχει το ίδιο x με το αρχικό min και μας απομένει ο υπολογισμός των y των δύο αυτών σημείων.

Για να βρούμε τα y που μας λείπουν ας θυμηθούμε ότι το αρχικό τετράγωνο χωρίστηκε με βάση ένα διαφορετικό ψηφίο στον άξονα των y. Αυτό σημαίνει πως καθώς το y αλλάζει τιμές σε κάποιο σημείο ένα ψηφίο από 0 θα γίνει 1. Μάλιστα το ψηφίο αυτό θα είναι το πρώτο από αριστερά που θα αλλάξει. Εμείς θέλουμε να δώσουμε στο LitMax τη μέγιστη τιμή που μπορεί να πάρει το y πριν το συγκεκριμένο ψηφίο γίνει από 0 1. Για το λόγο αυτό το y του LitMax θα είναι στη μορφή:

<κοινά ψηφία> 0 111...111

Αντίθετα στο BigMin θέλουμε να δώσουμε την ελάχιστη τιμή του y αφού το συγκεκριμένο ψηφίο αλλάξει, οπότε θα έχει τη μορφή

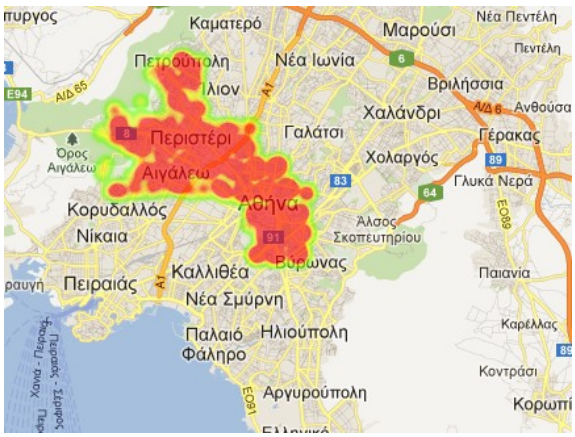
<κοινά ψηφία> 1 000...000

Ένας μνημονικός κανόνας για να βρίσκουμε τα y των LitMax και BigMin είναι να βάζουμε αρχικά τα κοινά σημεία των y του max και του min, και στη συνέχεια να συμπληρώνουμε στο LitMax ένα μηδέν και άσσους, ενώ στο BigMin έναν άσσο και μηδενικά.

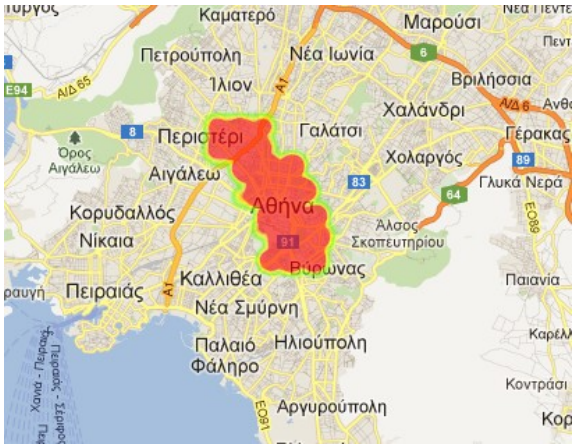
Αφού έχουμε χωρίσει το αρχικό τετράγωνο σε δύο νέα και έχουμε βρει για αυτά τα min και max, μπορούμε να χωρίσουμε το αρχικό ερώτημα σε δύο ερωτήματα που θα γίνουν σε αυτά τα δύο υπο-τετράγωνα. Το όφελος είναι ήδη ορατό: τα false-positive σημεία 108-144 σε αυτή την περίπτωση δεν ανακτούνται από τη βάση, ενώ ανοίγει η δυνατότητα για περαιτέρω τομές σε περισσότερα υπο-τετράγωνα και μείωση στον επιθυμητό βαθμό των false-positive σημείων.

Επίσης για να ολοκληρώσουμε τον αλγόριθμο να σημειώσουμε ότι ακριβώς το αντίστοιχο συμβαίνει στην περίπτωση που έχουμε κάθετη τομή του αρχικού τετραγώνου. Αρχικά την αναγνωρίζουμε επειδή το πρώτο ψηφίο των z-τιμών των min και max που αλλάζει ανήκει στον άξονα x. Η διαφορά που παρατηρείται είναι ότι είναι ότι το LitMax παίρνει y και όχι x από το max και το BigMin αντίστοιχα y από το min. Ο υπολογισμός των x των LitMax και BigMin ωστόσο γίνεται με τον ίδιο ακριβώς τρόπο και μπορεί να χρησιμοποιηθεί ο μνημονικός κανόνας που περιγράφηκε παραπάνω.[49]

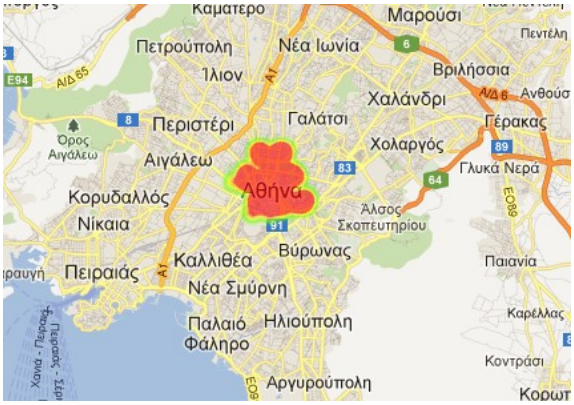
Για να δείξουμε το ρόλο που παίζει η τεχνική αυτή διαμέρισης στη μείωση των λανθασμένων σημείων που επιστρέφονται, παραθέτουμε παρακάτω μια σειρά εικόνων με τα venues που επιστράφηκαν από τη βάση ανάλογα με τον αριθμό υπο-ερωτημάτων στα οποία χωρίστηκε η αρχική ερώτηση. Η αρχική ερώτηση αφορούσε ένα τετράγωνο ακτίνας 2.000 μέτρων στην περιοχή του κέντρου και ήταν κοινή και για όλες περιπτώσεις. Οι περιπτώσεις που εξετάστηκαν αφορούσαν τη διάσπαση του ερωτήματος σε 0, 2, 4 και 8 υπο-ερωτήματα. Ο ρυθμός με τον οποίο τα λανθασμένα venues που επιστρέφονται μειώνονται, καθώς οι διαμερίσεις αυξάνονται είναι εντυπωσιακός.



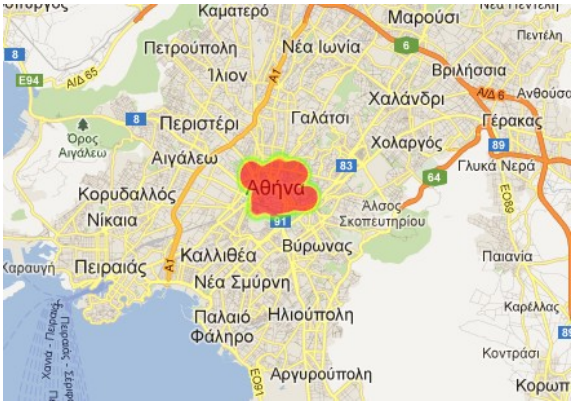
Εικόνα 21: 0 υπο-ερωτήματα



Εικόνα 22: 2 υπο-ερωτήματα



Εικόνα 23: 4 υπο-ερωτήματα



Εικόνα 24: 8 υπο-ερωτήματα

Κεφάλαιο 5

Πειραματικό μέρος

Στο κεφάλαιο αυτό θα παρουσιάσουμε μια σειρά από ερωτήματα που κάναμε στην HBase, στην οποία αποθήκευσε ο crawler τα δεδομένα που συνέλεξε κατά τη μία βδομάδα που έτρεχε στο εργαστήριο, και αφορούν κυρίως ερωτήματα εύρους πάνω στο σύνολο των venues. Θα δούμε τις παραμέτρους που βελτιστοποιούν το χρόνο απόκρισης των ερωτημάτων αυτών και θα συγκρίνουμε την “τυπική” με την z-order curve δεικτοδότηση. Επίσης, θα εξετάσουμε αν ο πίνακας 'timestamps' όντως βοηθάει στην περίπτωση που ζητείται το here now για όλα τα αποθηκευμένα venues και θα αναπτύξουμε έναν τρόπο ομαδοποίησης των ερωτημάτων, ώστε να γίνεται ακόμα πιο γρήγορη η εκτέλεση του κάθε scan. Τέλος, θα παρουσιάσουμε ένα use case εφαρμογής που στηρίζεται στη βάση και χρησιμοποιεί τα δεδομένα που βρίσκονται αποθηκευμένα μέσα της για να απεικονίσει με τη βοήθεια του Google Maps API έναν χάρτη της πυκνότητας του κόσμου στην Αθήνα.

Ο crawler έτρεξε από την Τετάρτη 19 Σεπτεμβρίου του 2012 μέχρι την επόμενη Τετάρτη 26 Σεπτεμβρίου, ξεκινώντας αμέσως μετά τα μεσάνυχτα της 18ης και συνεχίζοντας μέχρι και τα μεσάνυχτα της 26ης. Συνέλεγε δεδομένα ανά μία ώρα από τις υπηρεσίες Facebook, Foursquare και Twitter, ενώ το crawling διαρκούσε συνήθως 40 λεπτά. Στο διάστημα μέχρι να ολοκληρωθεί η μία ώρα η εφαρμογή έμενα αδρανής και όταν τα υπολειπόμενα λεπτά συμπληρωνόταν ξεκινούσε τον επόμενο γύρο crawling. Στο διάστημα αυτό της μίας βδομάδας συνέλεξε συνολικά 22.065 venues και εισήγαγε στην HBase 2.654.143 entries που αντιστοιχούν σε here now.

4.1 Αναζήτηση βέλτιστου σχήματος αποθήκευσης

Το βασικό κομμάτι των μετρήσεων που θα κάνουμε στην HBase και η ουσιαστική πληροφορία που θα προσπαθήσουμε να πάρουμε αφορά το χρόνο που χρειάζεται ένα ερώτημα εύρους για να πραγματοποιηθεί και να μας δώσει το αποτέλεσμα των συνολικών ατόμων που βρίσκονται σε μία περιοχή για μια συγκεκριμένη ώρα. Με τον όρο ερώτημα εύρους (range query) εννοούμε ένα ερώτημα, το οποίο ζητάει από τη βάση τα venues που βρίσκονται μέσα σε μια γεωγραφική περιοχή. Για αυτά τα venues η χρήσιμη πληροφορία είναι το here now του κάθε ενός για το χρονικό διάστημα που ζητείται, και όλα μαζί αθροίζονται για να δώσουν το τελικό αποτέλεσμα.

Ένα ερώτημα στη βάση, σε προγραμματιστικό πλαίσιο, δεν αποτελεί τίποτα περισσότερο από ένα scan στην HBase, το οποίο ζητάει από τον πίνακα με τα venues τις εγγραφές αυτές, οι οποίες βρίσκονται μέσα στη γεωγραφική περιοχή που ορίζεται. Επειδή η συγκεκριμένη βάση χρησιμοποιεί μονοδιάστατο κλειδί για τους πίνακές της, είναι σίγουρο πως η γραμμικοποίηση των δύο διαστάσεων των γεωγραφικών σημείων, για χρήση τους σαν κλειδιά, θα οδηγήσει σε απώλεια πληροφορίας εγγύτητας των σημείων. Αυτό σημαίνει πως δύο σημεία, τα οποία είναι γειτονικά στο δισδιάστατο επίπεδο, δεν είναι απαραίτητο πως θα είναι γειτονικά και στο μονοδιάστατο χώρο των κλειδιών τους. Έτσι τα αποτελέσματα ενός range query δε θα περιέχουν μόνο τα αποτελέσματα που θέλουμε είτε χρησιμοποιήσουμε την “τυπική” δεικτοδότηση, είτε το indexing με βάση τη z-order curve, αλλά το εύρος των

αποτελεσμάτων του scan θα περιέχει και venues τα οποία βρίσκονται ανάμεσα στην ελάχιστη και τη μέγιστη τιμή κλειδιών που ζητήσαμε, χωρίς να ανήκουν όμως στη ζητούμενη γεωγραφική περιοχή. Τα venues αυτά τα ονομάζουμε false-positive. Σκοπός μας είναι μέσω μετρήσεων να δούμε τι σχήμα πρέπει να χρησιμοποιήσουμε για να ελαχιστοποιήσουμε το χρόνο που χρειάζεται για ένα μέσο ερώτημα, αλλά και τι σχέση έχει ο χρόνος αυτός με το ποσοστό των false-positive venues που επιστρέφονται.

Στα πλαίσια της διπλωματικής θεωρήσαμε ότι όλα τα range queries θα αφορούν γεωγραφικά τετράγωνα, περιοχές δηλαδή που περικλείουν σημεία ανάμεσα σε ένα ελάχιστο και ένα μέγιστο γεωγραφικό πλάτος και τα αντίστοιχα μήκη. Για να περιγράψουμε τα τετράγωνα αυτά χρειαζόμαστε τις συντεταγμένες της κάτω αριστερής γωνίας (που αποτελούν τις μικρότερες συντεταγμένες) και τις αντίστοιχες της πάνω αριστερής γωνίας (που αποτελούν τις μέγιστες). Έτσι τα ερωτήματα τετραγωνικού εύρους που κάναμε μπορούμε να τα φανταστούμε σαν ερωτήματα που ψάχνουν τα venues ανάμεσα στις συντεταγμένες *(minLat, minLng)* και *(maxLat, maxLng)*. Τα τετράγωνα που χρησιμοποιήσαμε ήταν πλευράς 500, 1000 και 2000 μέτρων. Επίσης κάθε ερώτημα έχει επιπλέον χρονική πληροφορία, δηλαδή μπορεί να ζητάει τα here now ενός τετραγώνου για οποιοδήποτε χρονικό διάστημα για το οποίο ο crawler έχει συλλέξει δεδομένα.

Επειδή τα ερωτήματα γίνονται σε μία μεγάλη πόλη όπως η Αθήνα, θεωρήσαμε ότι τα μισά ερωτήματα θα αφορούν περιοχές του κέντρου, ενώ τα άλλα μισά τα προάστια. Επίσης, ένα 10% των ερωτημάτων, θα αφορά τους χρήστες που είναι στιγματισμένοι μέσω κάποιας από τις τρεις LBS σε ολόκληρη την πόλη και θα έχουν σαν κριτήριο τον χρόνο για τον οποίο θέλουμε τα δεδομένα αυτά. Η κατηγορία αυτή των ερωτημάτων είναι ιδιαίτερα σημαντική, αφού θα μας επιτρέψει να ελέγξουμε το ρόλο που μπορεί να παίξει ένας πίνακας όπως ο timestamps, δεικτοδοτημένος με βάση τη χρονική στιγμή που πάρθηκαν τα δεδομένα και σχεδιασμένος ώστε να επιστρέφει γρήγορα αποτελέσματα με βάση το χρόνο και όχι τη γεωγραφική περιοχή. Για να δημιουργήσουμε τυχαία ερωτήματα χρησιμοποιήσαμε την κλάση της Java Random. Συγκεκριμένα, αρχικά ζητούσαμε έναν τυχαίο πραγματικό αριθμό από το μηδέν μέχρι το ένα, και αν αυτός ήταν μικρότερος από 0,1 θεωρούσαμε πως το ερώτημα θα αφορά όλη την περιοχή της Αθήνας (10% των ερωτημάτων εξαρτώνται μόνο από το χρόνο). Αν το παραπάνω δεν ίσχυε ζητούσαμε έναν ακόμα πραγματικό από το 0 μέχρι το 1 και εξετάζαμε αν ήταν μικρότερος από το 0,5. Σε αυτή την περίπτωση θεωρούσαμε πως το ερώτημα πρέπει να γίνει στο κέντρο, ενώ στην αντίθετη στα προάστια (μισά ερωτήματα στο κέντρο και μισά στα προάστια).

Μέσω της πραγματοποίησης πολλών ερωτημάτων, που ισοδυναμούν όπως είπαμε με scans στην HBase και επεξεργασία των αποτελεσμάτων στην πλευρά του client, θα βρούμε το μέσο όρο του χρόνου που χρειάζεται ένα ερώτημα για να πραγματοποιηθεί σε κάθε μία από τις παρακάτω περιπτώσεις:

- “τυπική” δεικτοδότηση χωρίς τη βοήθεια του πίνακα timestamps
- “τυπική” δεικτοδότηση με τη βοήθεια του πίνακα timestamps
- δεικτοδότηση με z-order curve χωρίς τη βοήθεια του πίνακα timestamps
- δεικτοδότηση με z-order curve με τη βοήθεια του πίνακα timestamps

Τα ερωτήματα θα γίνονται σειριακά, δηλαδή αν χρησιμοποιήσουμε ένα σύνολο 10.000 queries, κάθε ένα από αυτά θα γίνει σαν να μην υπήρχαν τα προηγούμενα και θα ζητήσει από την HBase όλα τα venues που χρειάζεται, ακόμα κι αν αυτά έχουν ξαναγίνει fetch, ίσως και από το προηγούμενο ερώτημα. Με την άθροιση του συνολικού χρόνου που χρειάστηκε για τη συγκεκριμένη ομάδα ερωτημάτων και το πλήθος των queries που την απαρτίζουν θα υπολογίσουμε τον μέσο χρόνο που χρειάζεται ένα ερώτημα για κάθε περίπτωση και μέσω της

σύγκρισης των χρόνων για τις διάφορες περιπτώσεις θα καταλήξουμε στο αποδοτικότερο σχήμα αποθήκευσης για την εφαρμογή μας.

Για να φτάσουμε όμως σε αυτό, πρέπει μέσω πειραμάτων και μετρήσεων να βρούμε τις κατάλληλες παραμέτρους για κάθε είδος ερωτημάτων, ώστε να εξασφαλίσουμε ότι τόσο τα scan στον πίνακα με την τυπική δεικτοδότηση, όσο και σε αυτόν με τις z-τιμές των συντεταγμένων σαν κλειδιά να είναι τα γρηγορότερα δυνατά.

4.1.1 Βέλτιστος αριθμός διαμερίσεων ενός range query σε δεικτοδοτημένο με z-order curve πίνακα

Το πρώτο πράγμα που πρέπει να κοιτάξουμε όσον αφορά τα ερωτήματα στον 'zBinVenue', τον πίνακα δηλαδή που είναι δεικτοδοτημένος με βάση τη z-order curve, είναι ο αριθμός των διαμερίσεων στις οποίες πρέπει να σπάσει το αρχικό range query, ώστε να πετύχουμε τον μικρότερο δυνατό χρόνο απόκρισης. Όπως αναπτύξαμε και στο αντίστοιχο κομμάτι της θεωρίας, η χρήση της z-order curve και συγκεκριμένα της z-τιμής των συντεταγμένων των venues σαν row-key, δεν αρκεί ούτε για τη μείωση των false-positive αποτελεσμάτων που γίνονται fetch, ούτε για την μείωση του χρόνου που χρειάζεται για ένα ερώτημα. Για το λόγο αυτό, απαιτείται η διάσπαση του αρχικού τετραγώνου ερώτησης σε άλλα μικρότερα, που θα περιέχουν μέσα τους λιγότερα false-positive δεδομένα και άρα λιγότερο χρόνο ανάκτησης από τη βάση, αλλά και επεξεργασίας για την εξακρίβωση αν όντως ανήκουν στην περιοχή που επιθυμούμε. Από την άλλη, μεγάλη μείωση του ποσοστού εσφαλμένων εγγραφών από τη βάση θα συμβεί μόνο με διαμέριση του αρχικού ερωτήματος σε πολλά υπο-ερωτήματα. Αυτό σημαίνει περισσότερες RPC κλήσεις στην HBase, με το κόστος που τις συνοδεύουν, που μπορούν να οδηγήσουν σε αντίθετα από τα επιθυμητά αποτελέσματα, δηλαδή μεγαλύτερο μέσο χρόνο ανά ερώτημα. Στο χειρότερο σενάριο, μπορεί ένας υπερβολικά μεγάλος αριθμός διαμερίσεων να οδηγήσει στην απόκτηση των ίδιων εγγραφών που θα γινόταν σε άλλη περίπτωση με μία μόνο κλήση στη βάση, από μεγαλύτερο αριθμό RPC.

Για να εξετάσουμε αν τα παραπάνω όντως ισχύουν και να βρούμε τον καλύτερο αριθμό διαμερίσεων του αρχικού ερωτήματος σε κάθε περίπτωση, τρέξαμε για όλες τις πλευρές των τετραγώνων έναν αριθμό ερωτημάτων, για τα οποία πήραμε τους μέσους χρόνους απόκρισης, κρατώντας περαιτέρω στοιχεία για τα ερωτήματα εντός και εκτός κέντρου. Χρησιμοποιήσαμε τρία σύνολα ερωτήσεων, που περιλάμβαναν 100, 1.000 και 10.000 queries το κάθε ένα. Επειδή όπως έχουμε πει σε αυτή την κατηγορία ερωτήσεων ψάχνουμε το μέσο χρόνο απόκρισης της βάσης και τα αποτελέσματα δεν δρουν αθροιστικά, περιμένουμε οι μέσοι χρόνοι για κάθε σύνολο ερωτήσεων να είναι περίπου οι ίδιοι, με μεγαλύτερη ακρίβεια όσο ο αριθμός των queries σε κάθε σύνολο μεγαλώνει. Επίσης, για όλες τις περιπτώσεις χρησιμοποιήσαμε το ίδιο testcase ώστε να έχουμε όσο το δυνατόν μεγαλύτερη αντικειμενικότητα. Παρακάτω θα παρουσιάσουμε ένα σύνολο πινάκων που συνοψίζουν τα αποτελέσματα των μετρήσεων αυτών, μαζί με συνοδευτικά σχόλια για την καλύτερη κατανόησή τους.

Side = 500m - CENTER						
Sub-queries	Number of queries	Total ms	Ms/query	venues fetched	false positive percentage	true venues
1	56	11683	208,63	53443	0,98	1079
1	505	72274	143,12	399160	0,98	9381
1	5032	749779	149,00	4049728	0,98	91672
2	56	1718	30,68	5441	0,80	1079
2	505	13312	26,36	56379	0,83	9381
2	5032	133038	26,44	623515	0,85	91672
4	56	990	17,68	1921	0,44	1079
4	505	9399	18,61	17046	0,45	9381
4	5032	81272	16,15	168602	0,46	91672
8	56	1584	28,29	1543	0,30	1079
8	505	14449	28,61	13539	0,31	9381
8	5032	129486	25,73	131058	0,30	91672
16	56	2327	41,55	1325	0,19	1079
16	505	22555	44,66	11625	0,19	9381
16	5032	237208	47,14	113200	0,19	91672
32	56	4939	88,20	1189	0,09	1079
32	505	46357	91,80	10499	0,11	9381
32	5032	460432	91,50	102365	0,10	91672
64	56	8938	159,61	1149	0,06	1079
64	505	90315	178,84	9956	0,06	9381
64	5032	903157	179,48	97136	0,06	91672

Πίνακας 8: Διαμερίσεις για ερωτήματα τετραγώνων πλευράς 500 μέτρων στο κέντρο.

Στον Πίνακα 8 βλέπουμε τα αποτελέσματα που πήραμε για ερωτήματα με τετράγωνα πλευράς 500 μέτρων που έγιναν στο κέντρο της Αθήνας. Στην πρώτη στήλη φαίνεται ο αριθμός των διαμερίσεων στον οποίο σπάσαμε σε κάθε περίπτωση το αρχικό ερώτημα. Το sub-query 1 σημαίνει πως δεν πραγματοποιήσαμε καμία διαμέριση στο αρχικό ερώτημα, ενώ όπως βλέπουμε φτάσαμε να αντικαταστήσουμε το αρχικό query με 64 ισοδύναμα. Στη δεύτερη στήλη φαίνεται ο αριθμός των ερωτήσεων που έγιναν σε κάθε περίπτωση στο κέντρο. Παρατηρούμε το ίδιο μοτίβο να επαναλαμβάνεται, γεγονός που δικαιολογείται από το ίδιο testcase που είπαμε πως χρησιμοποιήσαμε σε κάθε περίπτωση – αν το testcase ήταν τυχαίο αλλά πιστό στα ποσοστά ερωτημάτων εντός και εκτός κέντρου θα υπήρχαν μικρές διαφοροποιήσεις.

Στις επόμενες στήλες φαίνονται οι συνολικοί και μέσοι χρόνοι πραγματοποίησης των ερωτημάτων. Σαν μέσο χρόνο πήραμε φυσικά το χρόνο για την ολοκλήρωση του αρχικού ερωτήματος και όχι των άλλων, μικρότερων, στα οποία αυτό αργότερα χωρίστηκε. Στους μέσους χρόνους βλέπουμε όπως περιμέναμε για κάθε αριθμό sub-queries τα αποτελέσματα που δίνουν οι 100, 1.000 και 10.000 ερωτήσεις να μη διαφέρουν σημαντικά. Επειδή μεγαλύτερη ακρίβεια έχουμε στο σύνολο των 10.000 θα ασχοληθούμε με τα αποτελέσματα που μας δίνει αυτό. Το σπάσιμο του αρχικού ερωτήματος σε δύο ισοδύναμα μειώνει το χρόνο

απόκρισης του scan κατά περίπου 5 φορές, γεγονός που αποδεικνύει το πόσο σημαντική είναι αυτή η τεχνική για τέτοιου είδους ερωτήματα. Με τη διαμέριση σε 4 υπο-ερωτήματα ο χρόνος κατεβαίνει ακόμη λίγο από τα 26 στα 16 ms/scan. Καθώς όμως οι διαμερίσεις γίνονται όλο και περισσότερες ο μέσος χρόνος αρχίζει να ανεβαίνει μέχρι να φτάσει τα 179ms στις 64 διαμερίσεις που είναι μεγαλύτερος από το χρόνο που είχε πραγματοποιηθεί χωρίς καμία διαμέριση. Αυτό έρχεται σε συμφωνία με τις υποψίες μας ότι όταν τα υπο-ερωτήματα αρχίσουν να γίνονται πολλά θα πετύχουμε αντίθετη απόδοση από αυτό που επιδιώκουμε.

Στις δύο επόμενες στήλες φαίνονται τα venues που έγιναν fetch από τη βάση και το ποσοστό από αυτά που ήταν false-positive. Το 98% που πραγματοποιήθηκε χωρίς διαμέριση αποδεικνύει και έμπρακτα ότι ένα απλό indexing με βάση τις z-τιμές των συντεταγμένων των venues δεν πρόκειται να επιτύχει λιγότερα false-positive αποτελέσματα από την “τυπική” δεικτοδότηση. Στις 4 διαμερίσεις, στις οποίες εμφανίστηκε ο καλύτερος χρόνος έχουμε 45 λάθος εγγραφές για κάθε 100 που παίρνουμε από τη βάση, ενώ στις 64 διαμερίσεις πετυχαίνουμε το εντυπωσιακό 6% false-positive στοιχείων.

Η τελευταία στήλη δείχνει τον αριθμό των πραγματικών venues που ανήκουν στις περιοχές που ζητήθηκαν. Το γεγονός ότι το ίδιο μοτίβο επαναλαμβάνεται δείχνει για άλλη μια φορά ότι χρησιμοποιήθηκε ότι ίδιο testcase ενώ ταυτόχρονα αποδεικνύει ότι ο αλγόριθμος διαμέρισης του αρχικού query σε υπο-ερωτήματα είναι σωστός και δεν πρόκειται να “χάσει” αποτελέσματα.

Side = 500m - SUBURBIA						
Sub-queries	Number of queries	Total ms	Ms/query	venues fetched	false positive percentage	true venues
1	44	5640	128,18	12697	0,97	1079
1	495	89446	180,70	150976	0,98	9381
1	4968	1044062	210,16	1334939	0,98	91672
2	44	2322	52,77	3791	0,92	1079
2	495	13884	28,05	16881	0,85	9381
2	4968	152477	30,69	140262	0,82	91672
4	44	712	16,18	512	0,37	1079
4	495	11315	22,86	4441	0,45	9381
4	4968	96666	19,46	43951	0,43	91672
8	44	1990	45,23	456	0,29	1079
8	495	14475	29,24	3505	0,30	9381
8	4968	150921	30,38	34412	0,28	91672
16	44	2422	55,05	380	0,15	1079
16	495	26059	52,64	2941	0,17	9381
16	4968	262669	52,87	29818	0,16	91672
32	44	4655	105,80	347	0,07	1079
32	495	55763	112,65	2671	0,08	9381
32	4968	478794	96,38	27187	0,08	91672
64	44	8228	187,00	332	0,03	1079
64	495	94292	190,49	2559	0,04	9381
64	4968	925162	186,22	26024	0,04	91672

Πίνακας 9: Διαμερίσεις για ερωτήματα τετραγώνων πλευράς 500 μέτρων στα προάστια.

Στον παραπάνω πίνακα φαίνονται τα αποτελέσματα που πήραμε για το ίδιο σετ ερωτήσεων (τα σύνολα 100, 1.000 και 10.000 ερωτήσεων δηλαδή) για τα queries όμως που έγιναν στα προάστια. Ο αριθμός των υπο-ερωτήσεων στις οποίες σπάσαμε την αρχική ερώτηση (sub-queries) κυμαίνεται στις ίδιες τιμές και από την στήλη του αριθμού των ερωτημάτων που έγιναν στα προάστια, βλέπουμε τόσο ότι είναι η διαφορά του συνολικού αριθμού ερωτήσεων του σετ, από τις ερωτήσεις που έγιναν στο κέντρο, όσο και ότι επαναλαμβάνουν το ίδιο μοτίβο, γεγονός που φανερώνει χρήση κοινού testcase. Ο μέσος χρόνος που αντιστοιχεί στο κάθε ερώτημα ακολουθεί την ίδια διακύμανση με αυτή των ερωτημάτων του κέντρου: φτάνει στο ελάχιστο στις 4 διαμερίσεις και έπειτα αρχίζει να αυξάνεται, φτάνοντας στις 64 διαμερίσεις σε χρόνο εφάμιλλο με αυτόν που πήραμε όταν δεν υπήρχε καμία διαμέριση.

Ο αριθμός των venues που ανακτούνται από τη βάση είναι όμως σαφώς μικρότερος από τον αντίστοιχο του κέντρου της Αθήνας. Αυτό οφείλεται στο ότι το κέντρο βρίσκεται στο ίδιο γεωγραφικό μήκος με μεγάλο κομμάτι της πόλης, που ξεκινάει από τον Πειραιά και φτάνει μέχρι την Αγία Παρασκευή. Έτσι, όταν ζητάμε εγγραφές που βρίσκονται στα όρια του γεωγραφικού πλάτους του κέντρου και περιορίζονται από κάποιο γεωγραφικό μήκος, λόγω της “τυπικής”, με βάση το latitude, δεικτοδότησης παίρνουμε και όλα τα άλλα venues που ανήκουν στα πλαίσια του μήκους αυτού. Αντίθετα, αν ζητήσουμε ένα τετράγωνο το οποίο βρίσκεται στην περιοχή της Γλυφάδας η δεικτοδότηση θα φροντίσει να πάρουμε γρήγορα όλα τα σημεία που ανήκουν στο γεωγραφικό πλάτος που ορίζει το τετράγωνο και τα false-positive που έχουν το επιθυμητό μήκος, δεν ανήκουν όμως στο τετράγωνο θα είναι λίγα, αφού δυτικά της Γλυφάδας υπάρχει θάλασσα και ανατολικά μη κατοικήσιμες περιοχές. Επίσης, λόγω του αυξημένου ενδιαφέροντος που παρουσιάζει η περιοχή του κέντρου βρίσκονται σε αυτή εγγεγραμμένα περισσότερα venues από ότι σε μια αντίστοιχη περιοχή των προαστίων, που οδηγεί σε περισσότερες καταχωρήσεις στη βάση. Αντίθετα, στα βόρεια προάστια όταν γίνει ερώτημα σε περιοχές όπως οι Αχαρνές ή οι Θρακομακεδόνες, παρότι το γεωγραφικό μήκος καλύπτει μεγάλη κατοικήσιμη περιοχή τα venues που είναι εγγεγραμμένα στις υπηρεσίες και άρα βρίσκονται στη βάση θα είναι λιγότερα. Όλα τα παραπάνω συνηγορούν στην εμφάνιση μεγαλύτερου αριθμού venues που γίνονται fetch στο κέντρο παρά στα προάστια.

Να σημειώσουμε εδώ πως στον αλγόριθμο ο οποίος δίνει τα τυχαία τετράγωνα στα οποία γίνονται οι ερωτήσεις δεν λάβαμε μέριμνα να μην επιστρέφει τετράγωνα που αντιστοιχούν σε μη κατοικήσιμες περιοχές. Μπορεί να επιστραφεί δηλαδή ένα τετράγωνο που να βρίσκεται εξολοκλήρου στη θάλασσα, έχουν όμως και αυτά τα ερωτήματα το ενδιαφέρον τους: η βάση επιστρέφει τα venues που έχουν τον ίδιο γεωγραφικό πλάτος και έτσι ο χρόνος ολοκλήρωσής τους δεν είναι μηδενικός. Το ποσοστό των περιοχών για τα οποία έγιναν queries και δεν ανήκαν σε κατοικήσιμη περιοχή αντιστοιχούν στο 28% για το σύνολο των 100 ερωτήσεων, και στο 30% και 31% για τα σετ των 1.000 και 10.000 ερωτήσεων αντίστοιχα.

Παρότι ο αριθμός των venues που επιστρέφονται για ερωτήματα στα προάστια είναι πολύ μικρότερος από αυτών του κέντρου, το ποσοστό των false-positive εγγραφών παραμένει αν όχι πάντα στα ίδια, σε πολύ κοντινά ποσοστά. Αυτό μας φανερώνει ότι ο αριθμός των εγγραφών που ανακτούμε χωρίς να το θέλουμε δεν έχει να κάνει με τον αριθμό των venues που παίρνουμε από τη βάση, αλλά με τον τρόπο που τα δεδομένα αποθηκεύονται και δεικτοδοτούνται σε αυτή. Το επαναλαμβανόμενο μοτίβο των true venues μας επιβεβαιώνει για άλλη μια φορά ότι χρησιμοποιήσαμε το ίδιο testcase αλλά και ότι ο αλγόριθμος διαμέρισης δεν περιέχει κάποιο λάθος που μπορεί να οδηγήσει σε απώλεια δεδομένων.

Στον τελευταίο πίνακα για τις ερωτήσεις σε τετράγωνα με πλευρά 500 μέτρα που φαίνεται παρακάτω, παρουσιάζεται ο συνολικός μέσος χρόνος των ερωτημάτων για τις διάφορες διαμερίσεις της αρχικής ερώτησης, όπως υπολογίστηκε για τα σετ των 100, 1.000 και 10.000 ερωτήσεων. Ξανά για λόγους ακρίβεια παρατηρούμε τους χρόνους για το σετ των 10.000 queries και βλέπουμε πως η διαμέριση με το λιγότερο μέσο χρόνο απόκρισης είναι αυτή των

4 υπο-ερωτήσεων, με χρόνο 10,54 ms/query. Τη διαμέριση αυτή θα υιοθετήσουμε για τις ερωτήσεις στον 'zBinVenue' με τετράγωνα πλευράς 500 μέτρων στα πειράματα που θα ακολουθήσουν.

Side = 500m – OVERALL QUERIES		
Sub-queries	Number of queries	Ms/query
1	100	173,23
1	1000	161,72
1	10000	179,38
2	100	28,79
2	1000	20,25
2	10000	20,93
4	100	11,68
4	1000	12,23
4	10000	10,54
8	100	18,33
8	1000	16,26
8	10000	14,84
16	100	24,78
16	1000	24,18
16	10000	25,36
32	100	50,84
32	1000	48,10
32	10000	47,54
64	100	90,67
64	1000	91,79
64	10000	91,76

Πίνακας 10: Διαμερίσεις για ερωτήματα τετραγώνων πλευράς 500 μέτρων – συνολικοί χρόνοι για κέντρο και προάστια.

Αντίστοιχα με τα ερωτήματα με τετράγωνα πλευράς 500 μέτρων, εργαστήκαμε και για τετράγωνα 1.000 και 2.000 μέτρων. Παρουσιάζουμε τα αποτελέσματα παρακάτω με τη μορφή πινάκων, χρησιμοποιώντας μόνο το σετ των 10.000 ερωτήσεων που είναι και το πιο ακριβές.

Side = 1000m - CENTER						
Sub-queries	Number of queries	Total ms	Ms/query	venues fetched	false positive percentage	true venues
1	4968	1347276	271,19	7658873	0,95	411468
2	4968	345374	69,52	2230749	0,82	411468
4	4968	162312	32,67	774631	0,47	411468
8	4968	195394	39,33	611453	0,33	411468
16	4968	291057	58,59	522051	0,21	411468
32	4968	501123	100,87	469635	0,12	411468
64	4968	942496	189,71	444493	0,07	411468

Πίνακας 11: Διαμερίσεις για ερωτήματα τετραγώνων πλευράς 1000 μέτρων στο κέντρο.

Side = 1000m - SUBURBIA						
Sub-queries	Number of queries	Total ms	Ms/query	venues fetched	false positive percentage	true venues
1	5032	1966716	390,84	2831703	0,96	102842
2	5032	398716	79,24	516811	0,80	102842
4	5032	205249	40,79	197601	0,48	102842
8	5032	251656	50,01	151161	0,32	102842
16	5032	349639	69,48	128069	0,20	102842
32	5032	583122	115,88	115621	0,11	102842
64	5032	1024791	203,65	109200	0,06	102842

Πίνακας 12: Διαμερίσεις για ερωτήματα τετραγώνων πλευράς 1000 μέτρων στα προάστια.

Side = 2000m - CENTER						
Sub-queries	Number of queries	Total ms	Ms/query	venues fetched	false positive percentage	true venues
1	5036	2235089	443,82	12772326	0,86	1750288
2	5036	882993	175,34	5890334	0,70	1750288
4	5036	478207	94,96	3322068	0,47	1750288
8	5036	469046	93,14	2610548	0,33	1750288
16	5036	541331	107,49	2233383	0,22	1750288
32	5036	747264	148,38	2026274	0,14	1750288
64	5036	1189946	236,29	1912338	0,08	1750288

Πίνακας 13: Διαμερίσεις για ερωτήματα τετραγώνων πλευράς 2000 μέτρων στο κέντρο.

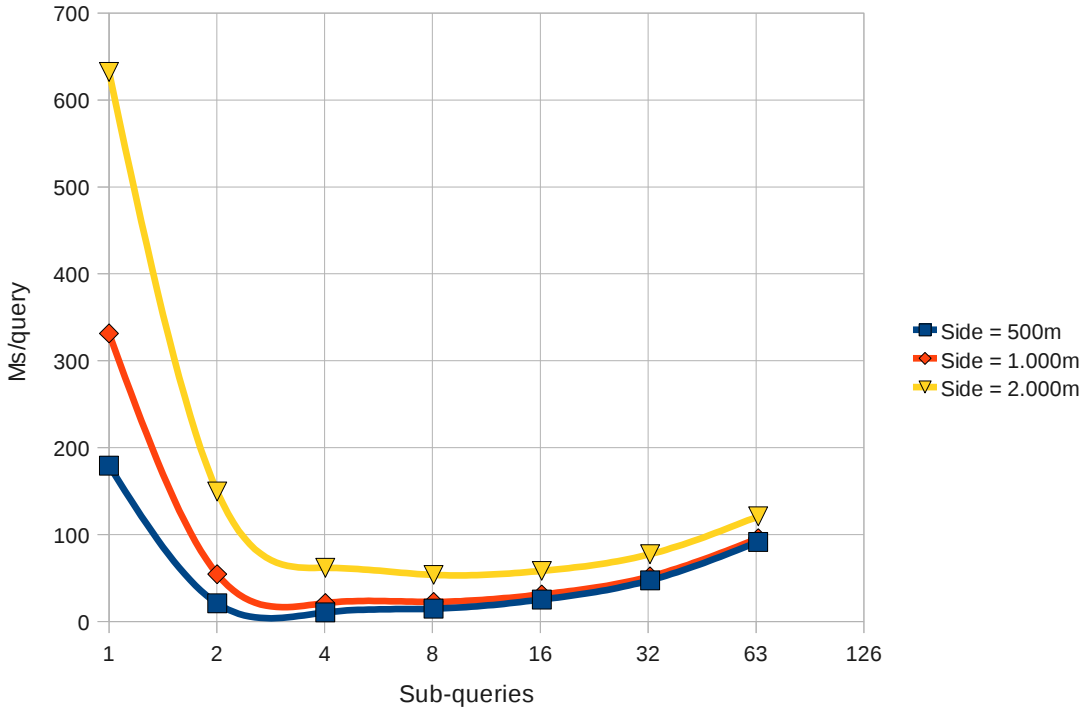
Side = 2000m - SUBURBIA						
Sub-queries	Number of queries	Total ms	Ms/query	venues fetched	false positive percentage	true venues
1	4964	4087413	823,41	6412579	0,93	435778
2	4964	1225813	246,94	1848972	0,76	435778
4	4964	562827	113,38	865444	0,50	435778
8	4964	541922	109,17	672332	0,35	435778
16	4964	673464	135,67	567830	0,23	435778
32	4964	890285	179,35	505137	0,14	435778
64	4964	1349117	271,78	473422	0,08	435778

Πίνακας 14: Διαμερίσεις για ερωτήματα τετραγώνων πλευράς 2000 μέτρων στα προάστια.

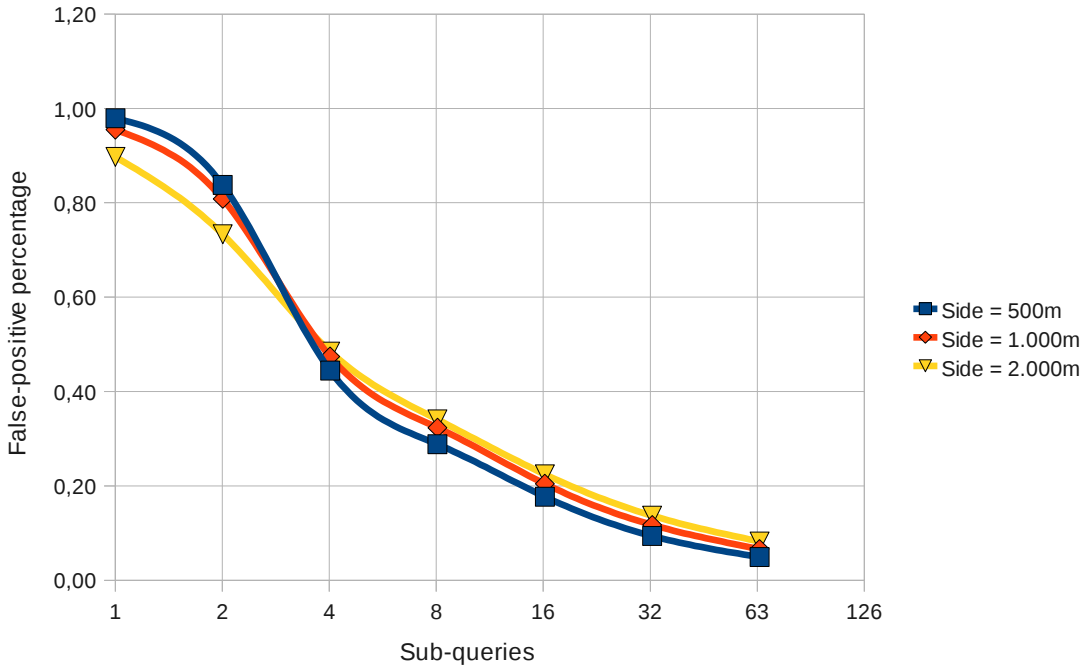
OVERALL QUERIES			
Sub-queries	Number of queries	Ms/ query Side 1000m	Ms/query Side 2000m
1	10000	331,40	632,25
2	10000	54,47	149,59
4	10000	21,36	61,89
8	10000	22,69	53,68
16	10000	31,29	58,34
32	10000	51,93	77,51
64	10000	95,85	121,10

Πίνακας 15: Διαμερίσεις για ερωτήματα τετραγώνων πλευράς 1.000 και 2.000 μέτρων – συνολικοί χρόνοι για κέντρο και προάστια.

Όλα τα παραπάνω φαίνονται συνοπτικά στα διαγράμματα που ακολουθούν και απεικονίζουν την πορεία του μέσου χρόνου και των τριών ειδών ερωτημάτων καθώς μεταβάλλεται ο αριθμός των υπο-ερωτήσεων, καθώς και τη μεταβολή του ποσοστού των false-positive venues που επιστρέφονται. Τέτοιου είδους διαγράμματα που συνοψίζουν το περιεχόμενο των πειραμάτων που κάναμε στην πλατφόρμα και δίνουν μια γενική εικόνα των αποτελεσμάτων που πήραμε θα παρουσιάζονται σε κάθε ενότητας αυτού του υποκεφαλαίου, μετά από τους αντίστοιχους πίνακες. Στα συγκεκριμένα διαγράμματα παραστήσαμε στον y άξονα το μέσο χρόνο ερωτήματος και το ποσοστό εσφαλμένων εγγραφών που ανακτήθηκαν, ενώ στον άξονα x τον αριθμό των υπο-ερωτημάτων, στα οποία έσπασε η αρχική ερώτηση. Λόγω της μεγάλης και ανομοιόμορφης γκάμας τιμών που παίρνουν τα τελευταία, χρησιμοποιήσαμε λογαριθμική κλίμακα.



Εικόνα 25: Συνολικός μέσος χρόνος ερωτήματος για διάφορες τιμές διαμέρισης του αρχικού ερωτήματος



Εικόνα 26: Ποσοστό false-positive venues που ανακτήθηκαν για διάφορες τιμές διαμέρισης του αρχικού ερωτήματος

Τα συμπεράσματα που βγάζουμε από τους πίνακες και τα διαγράμματα συνοψίζονται στα εξής:

- Εντός των στατιστικών για τετράγωνα της ίδιας πλευράς συμβαίνει ό,τι και στα τετράγωνα πλευράς 500 μέτρων, δηλαδή: οι χρόνοι ανά query πέφτουν όσο μεγαλώνουν οι διαμερίσεις μέχρι ένα σημείο και έπειτα αυξάνονται ξανά. Αυτό οφείλεται στο ότι από ένα σημείο και μετά η διαμέριση του αρχικού ερωτήματος σε υπο-ερωτήματα οδηγεί σε περισσότερες RPC από όσες είναι αναγκαίες για την ανάκτηση του ίδιου αριθμού venues. Αντίθετα τα ποσοστά false-positive venues μειώνονται συνεχώς. Οι χρόνοι στα προάστια είναι λίγο μεγαλύτεροι ανά ερώτηση από τους αντίστοιχους για το κέντρο, και τα venues που γίνονται fetch αισθητά λιγότερα. Τα false-positive τόσο στο κέντρο όσο και στα προάστια κυμαίνονται στα ίδια επίπεδα.
- Ο ιδανικός αριθμός διαμερίσεων για τετράγωνα πλευράς 1000 μέτρων είναι οι τέσσερις, ενώ για αυτά των 2.000 μέτρων οι οχτώ. Αυτό οφείλεται στη μεγαλύτερη έκταση των περιοχών στις οποίες γίνεται η ερώτηση, που συνεπάγονται περισσότερα venues που πρέπει να ανακτηθούν από τη βάση και οδηγούν στην ανάγκη για περισσότερες διαμερίσεις. Βλέπουμε έτσι πως ο αλγόριθμος κατάτμησης της αρχικής ερώτησης σε περισσότερες δεν είναι μία πανάκεια που εφαρμόζεται με συγκεκριμένο αριθμό διαμερίσεων, αλλά προσφέρει ευελιξία για την αποδοτική αντιμετώπιση όσο μεγάλων περιοχών και να προκύψουν.
- Όσον αφορά τις μετρήσεις σε σχέση με τετράγωνα διαφορετικής πλευράς οι χρόνοι απόκρισης ερωτημάτων είναι οι μικρότεροι για τετράγωνα με πλευρά 500 μέτρα και αυξάνονται όσο αυξάνεται και η πλευρά. Τα venues που επιστρέφονται αυξάνονται όσο μεγαλώνει η πλευρά των τετραγώνων, γεγονός λογικό αφού μεγαλώνει και το εύρος στο οποίο ζητάμε εγγραφές. Τα ποσοστά false-positive εγγραφών παραμένουν περίπου τα ίδια, με εξαίρεση τα τετράγωνα 2000 μέτρων για 0 και μία διαμέριση, όπου παρατηρείται αισθητή πτώση σε σχέση με τα αντίστοιχα τετράγωνα μικρότερων πλευρών. Αυτό μας δείχνει ότι στην z-order curve δεικτοδότηση, όταν δεν εφαρμόζεται διάσπαση του αρχικού ερωτήματος – ή εφαρμόζεται μικρή διάσπαση σε δύο υπο-ερωτήματα – μεγαλύτερο εύρος περιοχής αναζήτησης, θα οδηγήσει σε λιγότερα false-positive αποτελέσματα. Αν θυμηθούμε τον τρόπο με τον οποίο η εν λόγω καμπύλη γραμμικοποιεί τα σημεία του δισδιάστατου επιπέδου και φαίνεται στην Εικόνα 17, καταλαβαίνουμε ότι μεγαλύτερο τετράγωνο αναζήτησης οδηγεί σε περισσότερα “συνεχόμενα” σημεία, που συνδέονται μέσω της γραμμής. Τα σημεία αυτά είναι τα επιθυμητά, ενώ όταν η γραμμή “βγαίνει” έξω από το τετράγωνο αναζήτησης έχουμε τα false-positive που στην περίπτωση αυτή λιγοστεύουν. Αντίθετα όταν οι διαμερίσεις γίνουν αρκετές για να μην συναντάμε το παραπάνω φαινόμενο και κάθε υπο-ερωτήματα έχει λίγα false-positive αποτελέσματα, τότε μεγαλύτερο τετράγωνο συνεπάγεται περισσότερες εσφαλμένες εγγραφές, αφού τα false-positive κάθε sub-query δρουν αθροιστικά.

4.1.2 Caching

Για να καταλάβουμε τις παραμέτρους που θα προσπαθήσουμε να ρυθμίσουμε σε αυτή την ενότητα πρέπει να πρώτα να ρίξουμε λίγο φως στον τρόπο που η HBase προσφέρει πρόσβαση στα δεδομένα της. Χρησιμοποιώντας το API, όταν θέλουμε να πραγματοποιήσουμε ένα scan για κάποιες γραμμές ενός πίνακα, ζητάμε από τον πίνακα αυτόν έναν scanner, τον ResultScanner, από τον οποίο παίρνουμε πρόσβαση στις γραμμές του πίνακα. Έπειτα με έναν

iterator που μας προσφέρει η παραπάνω κλάση περιηγούμαστε στα αποτελέσματα και τα επεξεργαζόμαστε κατάλληλα. Αυτό δε σημαίνει όμως πως με το που ζητήσουμε τον scanner θα πάρουμε αυτόματα στη μνήμη και όλα τα αποτελέσματα. Αντίθετα, θα μας επιστραφεί μόνο ένας χάρτης με τα κλειδιά των αποτελεσμάτων και θα πρέπει να κάνουμε μία ξεχωριστή RPC κλήση για κάθε ένα από αυτά. Αυτό δεν είναι ιδιαίτερα αποδοτικό όταν έχουμε να κάνουμε με μικρά κελιά και θα ήταν φυσικό να προσπαθήσουμε να φέρουμε στη μνήμη περισσότερες από μία γραμμές σε κάθε RPC. Η λειτουργία αυτή λέγεται scanner caching και δεν είναι ενεργοποιημένη από προεπιλογή.

Μπορούμε να την ενεργοποιήσουμε είτε σε επίπεδο scan είτε σε επίπεδο πίνακα. Στην πρώτη περίπτωση το caching ισχύει μόνο για το scan αυτό, με τη δυνατότητα να εφαρμόσουμε διαφορετικό caching σε ένα επόμενο scan στον ίδιο πίνακα, ενώ στη δεύτερη όλες οι ερωτήσεις που θα γίνουν στον πίνακα αυτό θα χρησιμοποιούν την ίδια παράμετρο caching. Το ζητούμενο σε κάθε περίπτωση είναι η εύρεση του κατάλληλου σημείου ανάμεσα σε ένα μικρό αριθμό από RPC και στην απαιτούμενη μνήμη στη μεριά του client και του server. Συνήθως όσο μεγαλύτερο caching χρησιμοποιούμε τόσο καλύτερα αποτελέσματα παίρνουμε, ωστόσο αν το θέσουμε υπερβολικά υψηλά μπορεί να έχουμε αντίθετα αποτελέσματα από τα επιθυμητά, αφού κάθε κλήση για δεδομένα θα αργεί περισσότερο όσο το μέγεθος των δεδομένων αυτών αυξάνει και όταν ο client υπερβεί το μέγιστο heap που είναι διαθέσιμο για τη διεργασία του μπορεί να τερματίσει λόγω μίας OutOfMemoryException.

Στα πλαίσια της βελτιστοποίησης των παραμέτρων για ερωτήσεις που γίνονται στη βάση που θα αποτελεί τη ραχοκοκαλιά της πλατφόρμας που θέλουμε να αναπτύξουμε, προσπαθήσαμε να βρούμε το βέλτιστο caching, το βέλτιστο αριθμό γραμμών δηλαδή που θα επιστρέφεται σε κάθε RPC, για κάθε είδος ερωτήματος που θέλουμε να κάνουμε. Τα ερωτήματα αφορούν τους δύο πίνακες που αποθηκεύουν δεδομένα χρησιμοποιώντας σαν index τις συντεταγμένες των venues και range queries τετραγώνων διαφορετικών πλευρών. Για το λόγο αυτό προτιμήσαμε να δοκιμάσουμε διαφορετικά caching για κάθε είδος τετραγώνου. Τρέξαμε σε όλε τις περιπτώσεις το ίδιο testcase που αποτελείται από τέσσερα σετ ερωτήσεων με 10, 100, 1.000 και 10.000 ερωτήσεις το κάθε ένα. Τα αποτελέσματα φαίνονται παρακάτω.

SIDE = 500M – TYPICAL INDEXING				
	Millis/scan			
Caching	10 iterations	100 iterations	1000 iterations	10000 iterations
-1	496,60	221,21	123,68	113,69
0	144,10	100,16	115,49	112,94
10	57,30	31,59	36,35	34,67
20	25,50	27,39	31,79	30,07
30	23,70	30,56	28,18	28,08
40	22,00	25,21	27,83	26,92
50	21,40	24,76	27,38	27,76
60	21,00	22,35	27,26	29,41
70	39,60	28,89	29,56	28,84
80	44,30	28,75	31,85	29,47
90	67,50	65,12	71,33	63,00
100	58,70	57,35	65,32	60,42
150	68,90	48,83	50,88	48,90
200	35,10	43,95	45,30	43,49
250	27,60	35,58	42,23	39,60
300	25,50	31,24	34,71	32,14
350	27,00	30,16	33,29	31,93
400	27,20	29,36	33,57	31,71
450	23,10	28,20	30,77	30,69
500	40,70	27,55	28,34	29,12
600	21,20	25,66	28,73	28,95
700	19,50	27,85	29,36	28,93
800	19,30	25,85	31,31	29,29
900	41,20	23,93	29,44	29,42
1000	19,40	24,27	28,66	28,97
1500	19,10	27,13	29,60	28,89
2000	20,40	28,91	28,65	28,80
2500	19,30	26,49	29,35	30,24
5000	19,10	25,97	27,67	28,85
7500	19,10	25,84	29,01	28,78
10000	19,20	30,58	28,72	29,14
12500	19,10	27,93	27,67	28,77
15000	19,50	25,85	27,17	29,00

Πίνακας 16: Μέσος χρόνος ερωτημάτων τετραγώνων πλευράς 500 μέτρων σε πίνακα με "τυπική" δεικτοδότηση για διάφορες τιμές caching.

Ξεκινώντας από το σετ των 10 ερωτήσεων βλέπουμε ότι η προεπιλογή του caching, δηλαδή το -1, μας δίνει πολύ άσχημο χρόνο ανά scan, ακόμα χειρότερο και από ότι με 0 caching, κατά το οποίο παίρνουμε μόνο μια γραμμή ανά RPC. Με την ανάκτηση άλλης μίας γραμμής ανά RPC, ο μέσος χρόνος μειώνεται σε λιγότερο από το μισό και συνεχίζει να ακολουθεί καθοδική πορεία μέχρι τις 60 γραμμές ανά κλήση, που φτάνει τα 21 ms/scan.

Έπειτα ακολουθεί εκ νέου ανοδική πορεία μέχρι και τις 150 γραμμές, γεγονός που δείχνει ότι η αύξηση του caching αυτή καθεαυτή δεν είναι αρκετή για τη μείωση του μέσου χρόνου ανά σάρωση. Από τις 200 γραμμές και μετά οι χρόνοι πέφτουν για να φτάσουν στο μικρότερό τους επίπεδο για caching ίσο με 12.000. Στις 900 γραμμές παρατηρείται μια ανωμαλία, που οφείλεται στο μικρό σετ από το οποίο πήραμε το μέσο όρο και δείχνει την ανάγκη για μεγαλύτερα σετ και πιο ακριβείς χρόνους.

Στο σετ των εκατό ερωτήσεων τα αποτελέσματα γίνονται πιο ομοιογενή και η απότομη αύξηση στις 900 γραμμές εξαφανίζεται. Ακολουθείται όμως η ίδια μορφή, με μείωση μέχρι τις 60 γραμμές, τοπικό μέγιστο στις 90 και έπειτα φθίνουσα πορεία. Το ίδιο και στα σετ των 1.000 και 10.000 ερωτήσεων με τη διαφορά ότι το τοπικό ελάχιστο καθορίζεται στις 40 γραμμές ανά RPC και τα στατιστικά γίνονται πιο ακριβή. Για το λόγο αυτό θα χρησιμοποιήσουμε για την απόφασή μας τη στήλη των 10.000 ερωτήσεων και θα θέτουμε από εδώ και πέρα το caching για ερωτήματα τετραγώνων πλευράς 500 μέτρων στον πίνακα 'venue' με caching ίσο με 40.

Για τα τετράγωνα πλευράς 1.000 και 2.000 μέτρων στον πίνακα με την “τυπική” δεικτοδότηση θα εμφανίσουμε μόνο τα αποτελέσματα για το σετ των 10.000 ερωτήσεων. Αυτά συμπυκνώνονται στον επόμενο πίνακα 17.

Για τα τετράγωνα πλευράς 1.000 μέτρων παρατηρούμε παρόμοιο μοτίβο με αυτά των 500 μέτρων, μείωση δηλαδή του μέσου χρόνου μέχρι τις 40 γραμμές όπου εμφανίζεται ένα τοπικό ελάχιστο με 51,93 ms/scan, ένα τοπικό μέγιστο στις 90 γραμμές στα 119 ms και από εκεί και πέρα οι χρόνοι αρχίζουν και ξαναπέφτουν. Η διαφορά σε αυτή την περίπτωση είναι ότι οι χρόνοι φτάνουν τελικά σε μικρότερο επίπεδο από ότι τα 52 περίπου ms των 40 γραμμών, με ελάχιστο 48,21 ms ανά σάρωση στις 10.000 γραμμές ανά RPC. Η διαφορά μπορεί να μην είναι ιδιαίτερα μεγάλη, μπορεί όμως να κάνει γίνει σημαντική για πολύ μεγάλο αριθμό σαρώσεων, για αυτό επιλέξαμε το 10.000 σαν caching για τα ερωτήματα τετραγώνων πλευράς 1.000 μέτρων στον πίνακα με την “τυπική” δεικτοδότηση.

Στη στήλη με τα τετράγωνα πλευράς 2.000 μέτρων εμφανίζεται ξανά το ίδιο μοτίβο, με μεγαλύτερους χρόνους όμως, γεγονός λογικό αφού σε αυτή την περίπτωση ζητάμε να ανακτήσουμε μεγαλύτερο αριθμό venues. Ο μικρότερος χρόνος που πετυχαίνεται είναι 89,74 για 2.500 caching, που είναι αρκετή διαφορά από το τοπικό ελάχιστο 97,8 στις 40 γραμμές.

Αφού ολοκληρώσαμε την αναζήτηση για τη βέλτιστη τιμή caching για ερωτήματα στον 'venue', προσπαθήσαμε να βρούμε τις αντίστοιχες παραμέτρους για τον πίνακα δεικτοδοτημένο σύμφωνα με τις z-τιμές των venues. Τα αποτελέσματα φαίνονται στον πίνακα 18.

Τα συμπεράσματα είναι παρόμοια με αυτά για τον πίνακα με την “τυπική” δεικτοδότηση, ενώ σε όλες τις περιπτώσεις καλύτερα αποτελέσματα έχουμε για τις 40 γραμμές ανά RPC, αριθμός που θα χρησιμοποιηθεί από εδώ και πέρα σαν caching.

Όπως έχουμε πει, ένα μέρος των ερωτημάτων που θα τρέξουν στη βάση για την αξιολόγηση της απόδοσης κάθε δεικτοδότησης μέσω των χρόνων απόκρισης των ερωτημάτων, θα γίνει για ολόκληρη την Αθήνα. Για αυτό το λόγο θελήσαμε να βρούμε ποιο είναι το ιδανικό caching για τέτοιες περιπτώσεις, το scan δηλαδή του πίνακα 'timestamps' για την περίοδο μιας ώρας και όλων των εγγραφών των 'venue' και 'zBinVenue' με φίλτρο που θα περιορίσει τα here now στο διάστημα μίας συγκεκριμένης ώρας. Οι μετρήσεις έγιναν για σετ 100 ερωτήσεων και τα αποτελέσματα φαίνονται στον πίνακα 19.

TYPICAL INDEXING		
	Millis/scan	
Caching	Side = 1000m	Side = 2000m
-1	227,11	440,05
0	226,85	442,73
10	67,73	128,04
20	58,01	107,63
30	53,80	100,50
40	51,93	97,80
50	52,68	101,14
60	54,86	105,98
70	55,51	106,36
80	56,51	106,15
90	119,21	251,44
100	113,15	237,08
150	94,60	183,48
200	84,19	158,59
250	76,52	148,93
300	60,86	110,81
350	55,12	105,80
400	51,35	110,69
450	51,66	100,26
500	55,37	99,87
600	56,90	99,26
700	51,56	97,27
800	48,47	103,10
900	49,34	94,56
1000	48,67	94,01
1500	48,64	96,54
2000	48,75	91,95
2500	49,35	89,74
5000	49,80	90,19
7500	48,49	91,65
10000	48,21	92,07
12500	49,16	92,50
15000	48,58	92,77

Πίνακας 17: Μέσοι χρόνοι ερωτημάτων τετραγώνων πλευράς 1.000 και 2.000 μέτρων σε πίνακα με "τυπική" δεικτοδότηση για διάφορες τιμές caching.

Z-ORDER CURVE INDEXING			
	Millis/scan		
Caching	Side = 500m	Side = 1000m	Side = 2000m
-1	26,89	65,17	194,03
0	25,53	64,29	192,66
10	16,00	27,00	91,00
20	16,00	24,00	82,00
30	14,00	23,00	82,00
40	14,00	23,00	81,00
50	16,00	23,00	81,00
60	14,00	26,00	81,00
70	14,00	24,00	97,00
80	16,00	24,00	83,00
90	14,00	23,00	81,00
100	14,51	24,17	82,68
150	14,08	27,16	81,80
200	15,05	24,28	81,67
250	15,24	24,17	81,52
300	14,72	23,50	83,85
350	14,70	23,83	84,41
400	15,25	23,58	82,37
450	14,95	23,96	81,15
500	15,44	23,50	81,35
600	14,73	24,17	82,74
700	16,12	23,69	82,01
800	15,01	23,22	81,28
900	14,85	23,98	81,10
1000	16,17	24,48	82,42
1500	14,61	24,23	82,36
2000	15,25	23,82	81,40
2500	14,44	23,88	81,11
5000	14,48	23,56	80,11
7500	15,02	23,46	81,16
10000	15,09	24,00	81,19
12500	14,88	24,29	81,97
15000	15,20	23,98	81,40

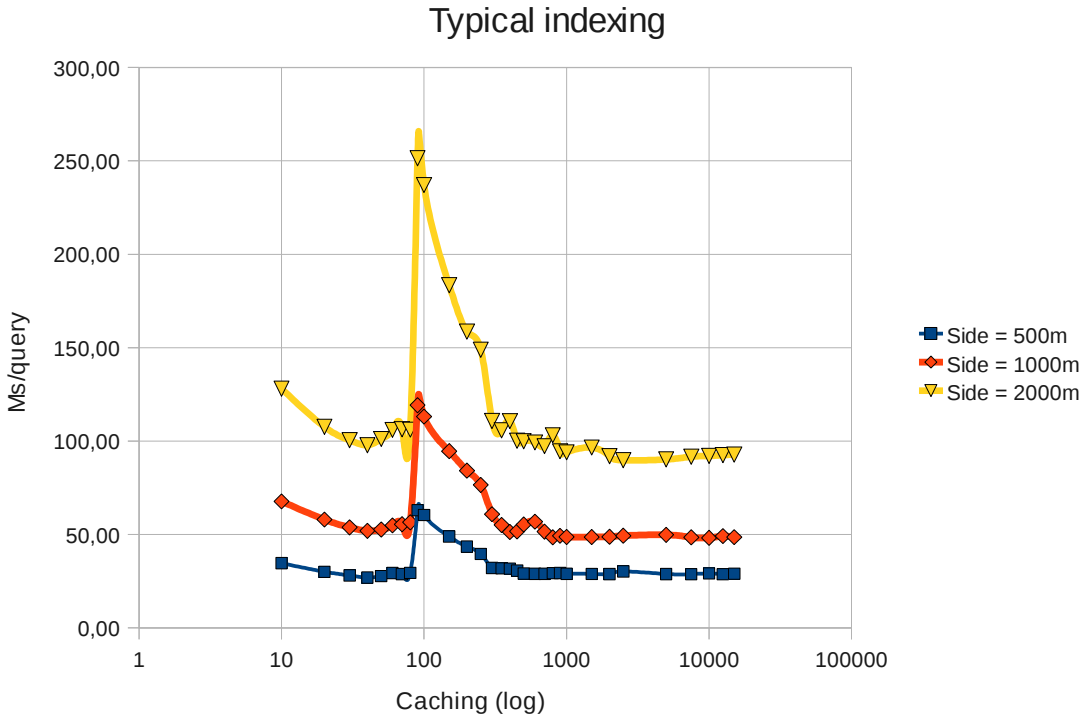
Πίνακας 18: Μέσοι χρόνοι ερωτημάτων τετραγώνων πλευράς 500, 1.000 και 2.000 μέτρων σε πίνακα με z-order curve δεικτοδότηση για διάφορες τιμές caching.

	Millis/scan		
Caching	timestamps	venue	zBinVenue
-1	229,97	4959,12	21710,51
0	178,76	4943,55	19893,87
10	771,12	1340,31	7826,33
20	156,45	1147,58	6686,42
30	78,67	1046,30	6326,52
40	81,51	1030,93	5886,37
50	73,69	1075,40	5513,94
60	69,51	1099,30	6476,33
70	70,32	1105,99	5077,38
80	66,97	1076,29	3973,49
90	65,84	2287,40	2635,19
100	63,04	2145,65	2554,12
150	58,49	1738,69	2497,06
200	54,69	1488,10	2432,64
250	51,72	1349,24	2417,22
300	53,22	939,15	2462,69
350	51,67	915,18	2369,70
400	49,63	925,12	2394,26
450	55,24	917,81	2379,16
500	48,49	908,93	2388,55
600	51,96	902,98	2373,66
700	46,94	918,13	2442,02
800	52,57	904,24	2416,34
900	51,64	881,06	2372,60
1000	46,61	884,77	2410,65
1500	49,19	889,86	2403,91
2000	47,70	873,91	2376,39
2500	48,47	885,27	2337,84
5000	46,77	873,94	2353,29
7500	44,80	874,53	2395,56
10000	54,35	869,66	2379,93
12500	48,42	893,65	2377,91
15000	51,04	859,78	2365,33

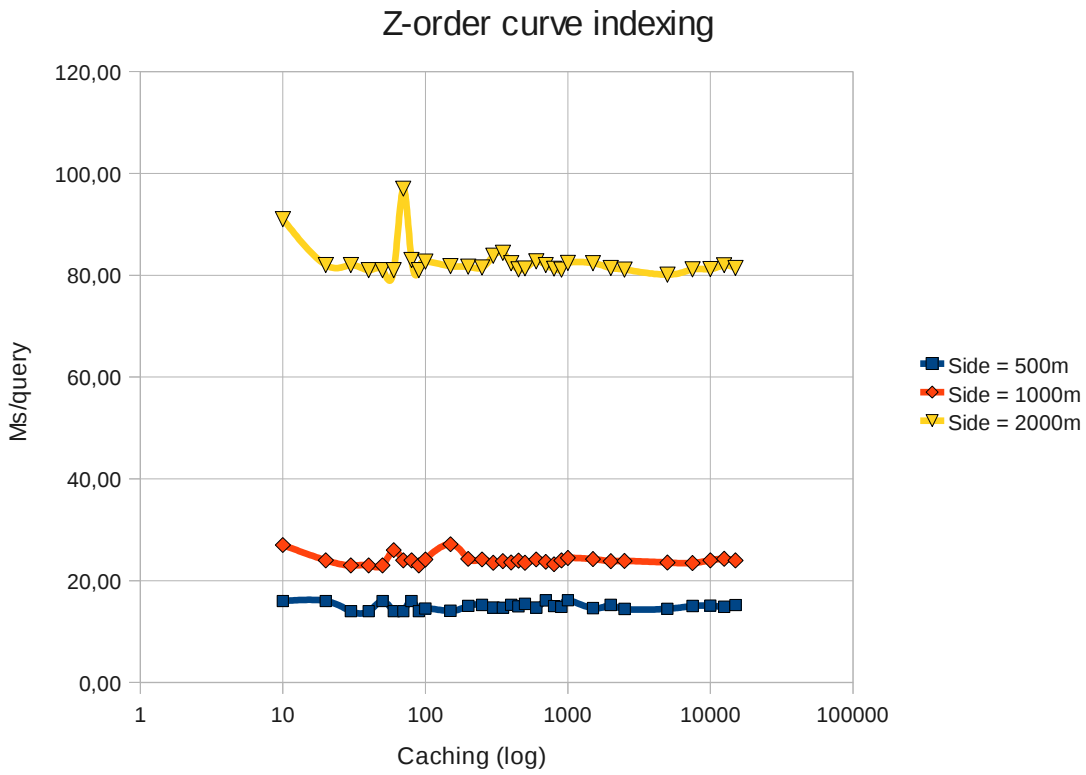
Πίνακας 19: Μέσος χρόνος full scan ερωτημάτων για τους τρεις πίνακες για διάφορες τιμές caching.

Παρατηρούμε ότι στην περίπτωση του full scan, τα καλύτερα caching είναι 7500 για τον πίνακα 'timestamps' και 15000 για τους πίνακες 'venue' και 'zBinVenue'.

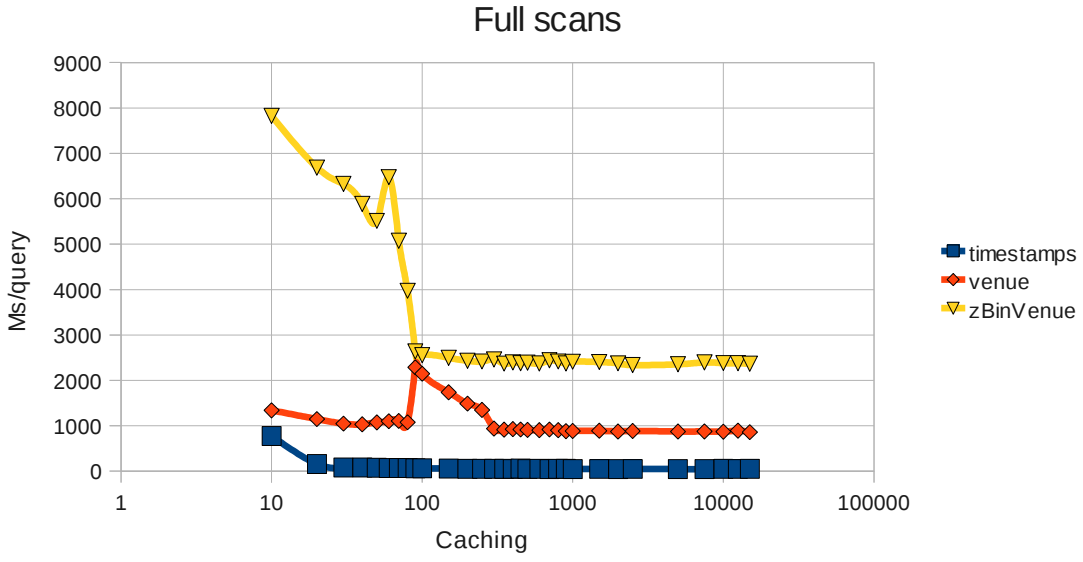
Συνεχίζουμε παραθέτοντας διαγράμματα που συμπυκνώνουν την πληροφορία των παραπάνω πινάκων. Εμφανίζουμε μόνο τα αποτελέσματα που πάρθηκαν για το ακριβές σετ των 10.000 ερωτήσεων, όσον αφορά τα range queries, και το σετ των 100 ερωτήσεων, όσον αφορά τα full scans. Σε όλα τα διαγράμματα ο άξονας x που συμβολίζει το caching, βρίσκεται σε λογαριθμική μορφή ώστε να φαίνονται καλύτερα οι αλλαγές κατά τη διακύμανση των τιμών του.



Εικόνα 27: Μέσος χρόνος ερωτήματος σε πίνακα με "τυπική" δεικτοδότηση για διάφορες τιμές caching.



Εικόνα 28: Μέσος χρόνος ερωτήματος σε πίνακα με z-order curve δεικτοδότηση για διάφορες τιμές caching.



Εικόνα 29: Μέσος χρόνος ερωτήματος για full scan στους τρεις πίνακες για διάφορες τιμές caching.

4.1.3 Σύγκριση επιδόσεων των σχημάτων δεικτοδότησης

Εδώ ακολουθεί το σημαντικότερο κομμάτι αυτού του κεφαλαίου που αναφέρεται στις επιδόσεις των δύο σχημάτων δεικτοδότησης στην HBase, την “τυπική” και τη στηριζόμενη στη z-τιμή των συντεταγμένων των venues. Η σύγκριση γίνεται μέσω του μέσου χρόνου ολοκλήρωσης ενός ερωτήματος, που αφορά στην ανάκτηση από τη βάση του here now, όλων των venues μιας τετραγωνικής περιοχής για μια συγκεκριμένη ώρα.

Χρησιμοποιήσαμε ένα testcase με 100, 1.000 και 10.000 ερωτήσεις, κοινές για κάθε περίπτωση. Κάθε ερώτηση αφορά ένα τυχαίο γεωγραφικό τετράγωνο που μπορεί να ανήκει με τις ίδιες πιθανότητες στο κέντρο ή στα προάστια. Επίσης ένα δέκα τοις εκατό των ερωτημάτων υποθέσαμε ότι αφορά όλα τα σημεία της πόλης. Κάθε ερώτημα σχετίζεται μόνο με μία ώρα, αλλά η ώρα αυτή μπορεί να είναι μια οποιαδήποτε ώρα μέσα στη βδομάδα που έτρεχε ο crawler.

Μέσω των testcases θα προσπαθήσουμε να βρούμε διαφορές στο μέσο χρόνο που χρειάζεται για να ολοκληρωθεί ένα range query και να βγάλουμε για τις διάφορες περιπτώσεις χρήσιμα συμπεράσματα. Για το λόγο αυτό θα παρουσιάσουμε χωριστά τα στατιστικά που μαζέψαμε για κάθε κατηγορία και στο τέλος της ενότητας θα τα παρουσιάσουμε συμπυκνωμένα, μαζί με τον κατάλληλο σχολιασμό. Ο διαχωρισμός των κατηγοριών έγκειται στο αν για τα scan ολόκληρης της Αθήνας χρησιμοποιείται ο βοηθητικός πίνακας 'timestamps' ή όχι και αν τα range queries γίνονται στον πίνακα με την “τυπική” ή τη z-order curve δεικτοδότηση.

4.1.3.1 “Τυπική” δεικτοδότηση χωρίς τη βοήθεια του 'timestamps'

Στην περίπτωση εκτελούμε στον πίνακα 'venue' range queries και μετράμε το χρόνο ολοκλήρωσης του κάθε ενός, ώστε να υπολογίσουμε στο τέλος το μέσο όρο τους. Ο πίνακας 'timestamps' που δημιουργήθηκε με σκοπό να βοηθήσει σε ερωτήματα που έχουν να κάνουν με όλο το εύρος της Αθήνας δε θα χρησιμοποιηθεί εδώ, αλλά αντίθετα για κάθε τέτοιο ερώτημα, που θα το ονομάζουμε full scan, θα ελέγχονται όλες οι εγγραφές του 'venue' και μέσω ενός φίλτρου θα επιλέγονται μόνο οι στήλες που ικανοποιούν το διάστημα της μιας ώρας που ζητάμε. Επειδή ο έλεγχος όλων των στηλών είναι πιο χρονοβόρος από την επιλογή ενός range με βάση το κλειδί ενός πίνακα – που θα γινόταν στην περίπτωση που χρησιμοποιούσαμε τον 'timestamps' – περιμένουμε μεγάλους χρόνους στα ερωτήματα αυτά. Παρακάτω παρουσιάζουμε τα αποτελέσματα σε μορφή πινάκων.

Side = 500m - CENTER					
Number of queries	Total ms	Ms/query	Venues fetched	False-positive percentage	True venues
43	1789	41,60	16568	0,95	894
444	15413	34,71	166717	0,96	7403
4431	152726	34,47	1671920	0,95	79416

Πίνακας 20: Ερωτήματα τετραγώνου 500 μέτρων σε “τυπική” δεικτοδότηση χωρίς τη βοήθεια του πίνακα 'timestamps' – κέντρο.

Side = 500m - SUBURBIA					
Number of queries	Total ms	Ms/query	Venues fetched	False-positive percentage	True venues
45	1004	22,31	5011	0,94	303
468	10353	22,12	58259	0,96	2278
4584	96475	21,05	581561	0,96	23388

Πίνακας 21: Ερωτήματα τετραγώνου 500 μέτρων σε "τυπική" δεικτοδότηση χωρίς τη βοήθεια του πίνακα 'timestamps' – προάστια.

Side = 500m – FULL SCANS		
Number of queries	Total ms	Ms/query
12	11238	936,50
88	76788	872,59
985	855148	868,17

Πίνακας 22: Ερωτήματα τετραγώνου 500 μέτρων σε "τυπική" δεικτοδότηση χωρίς τη βοήθεια του πίνακα 'timestamps' – full scans.

Side = 1000m - CENTER					
Number of queries	Total ms	Ms/query	Venues fetched	False-positive percentage	True venues
43	2831	65,84	36941	0,89	3929
412	26848	65,17	334750	0,90	34235
4496	293605	65,30	3632580	0,90	369180

Πίνακας 23: Ερωτήματα τετραγώνου 1.000 μέτρων σε "τυπική" δεικτοδότηση χωρίς τη βοήθεια του πίνακα 'timestamps' – κέντρο.

Side = 1000m - SUBURBIA					
Number of queries	Total ms	Ms/query	Venues fetched	False-positive percentage	True venues
45	1662	36,93	18959	0,95	1024
500	15512	31,02	147690	0,93	10451
4519	149897	33,17	1287708	0,93	92752

Πίνακας 24: Ερωτήματα τετραγώνου 1.000 μέτρων σε "τυπική" δεικτοδότηση χωρίς τη βοήθεια του πίνακα 'timestamps' – προάστια.

Side = 1000m – FULL SCANS		
Number of queries	Total ms	Ms/query
12	9911	825,92
88	78172	888,32
985	849026	861,96

Πίνακας 25: Ερωτήματα τετραγώνου 1.000 μέτρων σε "τυπική" δεικτοδότηση χωρίς τη βοήθεια του πίνακα 'timestamps' – full scans.

Side = 2000m - CENTER					
Number of queries	Total ms	Ms/query	Venues fetched	False-positive percentage	True venues
45	5209	115,76	61610	0,78	13292
445	56997	128,08	708592	0,78	154083
4419	547051	123,80	7071945	0,78	1534994

Πίνακας 26: Ερωτήματα τετραγώνου 2.000 μέτρων σε "τυπική" δεικτοδότηση χωρίς τη βοήθεια του πίνακα 'timestamps' – κέντρο.

Side = 2000m - SUBURBIA					
Number of queries	Total ms	Ms/query	Venues fetched	False-positive percentage	True venues
43	2488	57,86	29651	0,88	3563
467	26039	55,76	299962	0,87	39390
4596	280132	60,95	3181930	0,87	413263

Πίνακας 27: Ερωτήματα τετραγώνου 2.000 μέτρων σε "τυπική" δεικτοδότηση χωρίς τη βοήθεια του πίνακα 'timestamps' – προάστια.

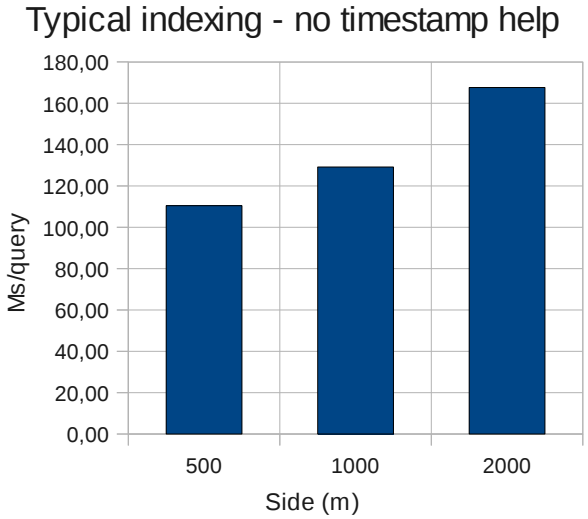
Side = 2000m – FULL SCANS		
Number of queries	Total ms	Ms/query
12	10725	893,75
88	74959	851,81
985	849767	862,71

Πίνακας 28: Ερωτήματα τετραγώνου 2.000 μέτρων σε "τυπική" δεικτοδότηση χωρίς τη βοήθεια του πίνακα 'timestamps' – full scans.

OVERALL QUERIES			
	Ms/query		
Number of queries	Side = 500m	Side = 1000m	Side = 2000m
100	140,31	144,04	184,22
1000	102,55	120,53	158,00
10000	110,43	129,25	167,70

Πίνακας 29: Σύνοψη συνολικών χρόνων και για τα τρία είδη ερωτημάτων

Ακολουθεί το διάγραμμα με τους συνολικούς χρόνους των τριών ειδών ερωτημάτων για το σετ των 10.000 ερωτήσεων.



Εικόνα 30: Μέσος χρόνος ανά ερώτημα σε "τυπική" δεικτοδότηση χωρίς τη βοήθεια του πίνακα timestamps για τα τρία είδη ερωτημάτων.

Από τους παραπάνω πίνακες και το διάγραμμα μπορούμε να βγάλουμε τα ακόλουθα συμπεράσματα:

- Οι μέσοι χρόνοι που παίρνουμε από το σετ των 100 ερωτήσεων, διαφέρουν σε μερικές περιπτώσεις αρκετά από τους αντίστοιχους χρόνους που προκύπτουν από το σύνολο των 1.000 ερωτήσεων. Αυτό οφείλεται στο ότι το δείγμα του πρώτου συνόλου είναι ανεπαρκές για την εξαγωγή ασφαλών συμπερασμάτων και είναι φυσικό να είναι λιγότερο ακριβές από το δεύτερο. Αντίθετα, ανάμεσα στα σετ των 1.000 και 10.000 ερωτήσεων οι διαφορές είναι σαφώς μικρότερες, ενώ παρατηρείται και τάση σταθεροποίησης των αριθμών, οπότε θεωρούμε το δείγμα των 10.000 ερωτήσεων επαρκές για την εξαγωγή συμπερασμάτων. Για το λόγο αυτό στη συνέχεια της ενότητας θα παρουσιάζουμε μόνο τα αποτελέσματα για το εν λόγω σύνολο ερωτήσεων.

- Εντός των πλαισίων κάθε κατηγορίας ερωτημάτων – ανάλογα με το μήκος της πλευράς – παρατηρούμε μια σειρά από κοινά χαρακτηριστικά. Ο μέσος χρόνος εκτέλεσης των ερωτημάτων είναι μικρότερος για τα προάστια, γεγονός που οφείλεται στα λιγότερα venues που γίνονται fetch σε σχέση με το κέντρο. Το ποσοστό των false-positive εγγραφών που επιστρέφονται συνεχώς μειώνεται όσο αυξάνεται η πλευρά των τετραγώνων. Αυτό συμβαίνει γιατί όσο το γεωγραφικό μήκος του τετραγώνου αποτελεί μεγαλύτερο μέρος του εύρους μηκών που επιστρέφονται τόσο περισσότερα venues θα είναι εντός ορίων. Η μείωση των false-positive εγγραφών γίνεται γρηγορότερα για το κέντρο και οφείλεται στην μεγαλύτερη πυκνότητα venues που βρίσκονται στην περιοχή αυτή, σε σχέση με την πυκνότητα στα προάστια. Επίσης, τα full scans χρειάζονται πολύ περισσότερο χρόνο για να εκτελεστούν από τα range queries, αφού σαρώνουν ολόκληρο τον πίνακα 'venue' και όχι μόνο ένα κομμάτι του, που πιστεύεται ότι ανήκει στην περιοχή που ζητάμε.
- Συγκρίνοντας τα αποτελέσματα μεταξύ των τριών πλευρών για τις οποίες γίνονται queries, βλέπουμε ότι ο χρόνος που απαιτείται για την εκτέλεση των ερωτημάτων τόσο στο κέντρο όσο και στα προάστια αυξάνεται όσο αυξάνεται το εύρος της περιοχής, σαν επακόλουθο των περισσότερων εγγραφών της βάσης που ανακτούνται. Το γεγονός αυτό αντικατοπτρίζεται και στον πίνακα με τους μέσους χρόνους όλων των ερωτημάτων, που περιλαμβάνουν scans στο κέντρο, τα προάστια και ολόκληρη την Αθήνα. Οι συνολικοί αυτοί χρόνοι δεν φανερώνουν πλήρως την αύξηση του χρόνου ερωτήματος όσο αυξάνουν οι πλευρές των τετραγώνων, γιατί σταθμίζονται πολύ από τους μεγάλους χρόνους των full scans που παραμένουν ίδιοι. Δεν θα περιμέναμε κάτι διαφορετικό γιατί όπως είπαμε στη συγκεκριμένη κατηγορία σαρώνονται όλες οι εγγραφές του πίνακα, ανεξάρτητα με την κατηγορία πλευράς στην οποία βρισκόμαστε.

4.1.3.2 “Τυπική” δεικτοδότηση με τη βοήθεια του 'timestamps'

Στην ενότητα αυτή θα πραγματοποιήσουμε τα ίδια ερωτήματα με αυτά της προηγούμενης στον πίνακα 'venue', με τη διαφορά ότι τώρα για τα full scans θα χρησιμοποιηθεί ο πίνακας 'timestamps'. Ο πίνακας αυτός δημιουργήθηκε με σκοπό την επιτάχυνση τέτοιου είδους ερωτημάτων, ερωτήματα δηλαδή που ζητάνε το here now όλων των venues για μια συγκεκριμένη ώρα, γεγονός που φαίνεται και από την υλοποίησή του. Θυμίζουμε ότι σαν κλειδί γραμμής χρησιμοποιείται το timestamp της στιγμής που πάρθηκε η μέτρηση, ενώ σε κάθε γραμμή περιέχονται τα here now των venues μιας περιοχής, που επιστράφηκαν από την υπηρεσία. Στα αποτελέσματα, θα περιμένουμε λοιπόν ίδιους χρόνους όσον αφορά τα queries στο κέντρο και τα προάστια, πολύ μικρότερους χρόνους όμως όσον αφορά τα full scans.

CENTER						
Side	Number of queries	Total ms	Ms/query	Venues fetched	False-positive percentage	True venues
500	4431	146179	32,99	1671920	0,95	79416
1000	4496	295446	65,71	3632580	0,90	369180
2000	4419	549726	124,40	7071945	0,78	1534994

Πίνακας 30: Μέσοι χρόνοι ερωτημάτων για ερωτήματα και των τριών ειδών σε “τυπική” δεικτοδότηση με τη βοήθεια του 'timestamps' – κέντρο.

SUBURBIA						
Side	Number of queries	Total ms	Ms/query	Venues fetched	False-positive percentage	True venues
500	4584	90657	19,78	581561	0,96	23388
1000	4519	146465	32,41	1287708	0,93	92752
2000	4596	282981	61,57	3181930	0,87	413263

Πίνακας 31: Μέσοι χρόνοι για ερωτήματα και των τριών ειδών σε "τυπική" δεικτοδότηση με τη βοήθεια του 'timestamps' – προάστια.

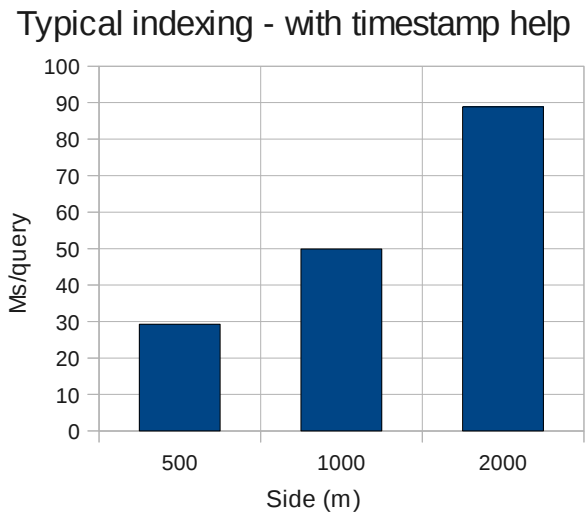
FULL SCANS			
Side	Number of queries	Total ms	Ms/query
500	985	55618	56,46
1000	985	57226	58,10
2000	985	56073	56,93

Πίνακας 32: Μέσοι χρόνοι για ερωτήματα και των τριών ειδών σε "τυπική" δεικτοδότηση με τη βοήθεια του 'timestamps' – full scans.

OVERALL QUERIES		
Side	Number of queries	Ms/query
500	10000	29,25
1000	10000	49,91
2000	10000	88,88

Πίνακας 33: Συνολικοί μέσοι χρόνοι για ερωτήματα και των τριών ειδών σε "τυπική" δεικτοδότηση με τη βοήθεια του 'timestamps'

Το διάγραμμα της εικόνας 31 δείχνει πού κυμαίνονται οι συνολικοί μέσοι χρόνοι των ερωτημάτων για τετράγωνα 500, 1.000 και 2.000 μέτρων για τα ερωτήματα που είδαμε σε αυτή την ενότητα.



Εικόνα 31: Μέσος χρόνος ανά ερώτημα σε "τυπική" δεικτοδότηση με τη βοήθεια του πίνακα timestamps για τα τρία είδη ερωτημάτων.

Από τους πίνακες αυτούς και το διάγραμμα βλέπουμε ότι:

- Στο κέντρο και στα προάστια έγιναν ακριβώς οι ίδιες ερωτήσεις και ανακτήθηκαν ακριβώς τα ίδια venues, με τα ίδια false-positive. Μικρές διαφορές υπάρχουν στο μέσο χρόνο απόκρισης και προέρχονται από την κατάσταση στην οποία μπορεί να βρίσκεται ο cluster την κάθε στιγμή, δεν ξεπερνούν όμως σε καμία περίπτωση τα 2 ms.
- Όσον αφορά τα full scans ο χρόνος ολοκλήρωσης του ερωτήματος έπεσε εντυπωσιακά από την τάξη των 800 στα 55 περίπου ms. Αυτό μας βεβαιώνει ότι η φιλοσοφία ενός πίνακα δεικτοδοτημένου με βάση τα timestamps κατά τα οποία πάρθηκε η πληροφορία είναι ιδιαίτερα χρήσιμος στην περίπτωση που ζητούνται δεδομένα για όλη τη γεωγραφική περιοχή από την οποία έχουμε συλλέξει venues, για κάποιο χρονικό διάστημα.
- Ο μέσος χρόνος των συνολικών ερωτημάτων συμπαρασύρεται προς τα κάτω από τη μείωση του χρόνου των full scans και τώρα εκτός από την εμφανή διαφορά με την προηγούμενη περίπτωση, φαίνεται καθαρότερα και η αύξηση του χρόνου που επιφέρει ένα τετράγωνο μεγαλύτερης πλευράς.

4.1.3.3 Z-order curve δεικτοδότηση χωρίς τη βοήθεια του 'timestamps'

Έχοντας ολοκληρώσει τις ερωτήσεις που θέλαμε να κάνουμε στον πίνακα 'venue' και καταλήγοντας στα συμπεράσματα που περιγράψαμε παραπάνω, θα πραγματοποιήσουμε εδώ τα ίδια ερωτήματα με αυτά που έγιναν στην "τυπική" δεικτοδότηση χωρίς τη βοήθεια του 'timestamps', με τη μόνη διαφορά ότι ο πίνακας που θα δέχεται τα ερωτήματα θα είναι ο 'zBinVenue'. Περιμένουμε να δούμε τα ίδια πράγματα, δηλαδή πολύ μεγαλύτερο χρόνο σάρωσης για full scans και μικρότερους για τα range queries, ενώ θα έχει ενδιαφέρον η σύγκριση με τα αντίστοιχα αποτελέσματα του πίνακα 'venue'. Χρησιμοποιήσαμε διαμέριση του αρχικού ερωτήματος σε 4 υπο-ερωτήματα για τα τετράγωνα μήκους 500 και 1.000 μέτρων και 8 υπο-ερωτήματα για τα τετράγωνα πλευράς 2.000 μέτρων, αφού για αυτά

πετυχαίνουμε τους καλύτερους χρόνους, όπως είδαμε στο κεφάλαιο 3.1.1.

CENTER						
Side	Number of queries	Total ms	Ms/query	Venues fetched	False-positive percentage	True venues
500	4431	83910	18,94	147372	0,46	79416
1000	4496	143246	31,86	696090	0,47	369180
2000	4419	301527	68,23	2301642	0,33	1534994

Πίνακας 34: Μέσοι χρόνοι ερωτημάτων για ερωτήματα και των τριών ειδών σε z-order curve δεικτοδότηση χωρίς τη βοήθεια του 'timestamps' – κέντρο.

SUBURBIA						
Side	Number of queries	Total ms	Ms/query	Venues fetched	False-positive percentage	True venues
500	4584	80651	17,59	42353	0,45	23388
1000	4519	101464	22,45	179725	0,48	92752
2000	4596	186666	40,93	637935	0,35	413263

Πίνακας 35: Μέσοι χρόνοι ερωτημάτων για ερωτήματα και των τριών ειδών σε z-order curve δεικτοδότηση χωρίς τη βοήθεια του 'timestamps' – προάστια.

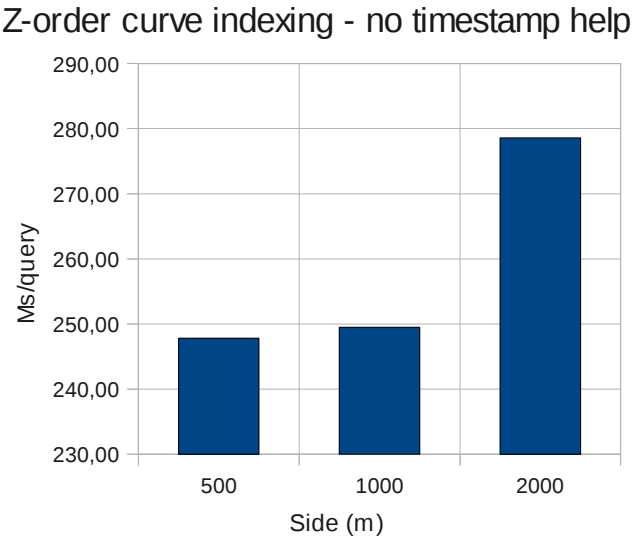
FULL SCANS			
Side	Number of queries	Total ms	Ms/query
500	985	2313590	2348,82
1000	985	2250042	2284,31
2000	985	2296351	2331,32

Πίνακας 36: Μέσοι χρόνοι ερωτημάτων για ερωτήματα και των τριών ειδών σε z-order curve δεικτοδότηση χωρίς τη βοήθεια του 'timestamps' – full scans.

OVERALL QUERIES		
Side	Number of queries	Ms/query
500	10000	247,82
1000	10000	249,48
2000	10000	278,60

Πίνακας 37: Συνολικοί μέσοι χρόνοι ερωτημάτων για ερωτήματα και των τριών ειδών σε z-order curve δεικτοδότηση χωρίς τη βοήθεια του 'timestamps'.

Ακολουθεί το διάγραμμα στο οποίο φαίνονται οι συνολικοί μέσοι χρόνοι των ερωτημάτων των οποίων εξετάσαμε σε αυτή την ενότητα.



Εικόνα 32: Μέσος χρόνος ανά ερώτημα σε z-order curve δεικτοδότηση χωρίς τη βοήθεια του πίνακα timestamps για τα τρία είδη ερωτημάτων.

Από τα παραπάνω προκύπτουν ορισμένα πολύ ενδιαφέροντα στοιχεία:

- Όπως και με την “τυπική” δεικτοδότηση, έτσι και εδώ βλέπουμε τα ερωτήματα που γίνονται στα προάστια να ανακτούν λιγότερα venues από τη βάση και να έχουν έτσι και μικρότερους χρόνους εκτέλεσης. Επίσης συγκρίνοντας τα αποτελέσματα για τις τρεις περιπτώσεις πλευρών παρατηρούμε τους μέσους χρόνους να αυξάνουν από τη μία στην άλλη. Αυτό οφείλεται όπως έχουμε ξαναπεί στο ότι περισσότερα venues πρέπει να γίνουν fetch λόγω της μεγαλύτερου εύρους της περιοχής, γεγονός που φαίνεται και από την αντίστοιχη στήλη.
- Το ποσοστό των false-positive venues που ανακτούνται από τη βάση κινείται περίπου στα ίδια επίπεδα για τα ερωτήματα σε τετράγωνα πλευράς 500 και 1.000 μέτρων, λόγω του ίδιου αριθμού διαμέρισης του βασικού ερωτήματος σε υπο-ερωτήματα που χρησιμοποιήσαμε και στα δύο. Η μικρή ανοδική τάση του ποσοστού μας δείχνει πως όσο το εύρος της περιοχής για την οποία ζητάμε στοιχεία μεγαλώνει, τόσο θα αυξάνονται και οι εκτός περιοχής εγγραφές που θα γίνονται ανακτώνται. Από την άλλη, στα range queries για τετράγωνα πλευράς 2.000 μέτρων το false-positive ποσοστό έχει πέσει αισθητά στο 33% και 35% για κέντρο και προάστια αντίστοιχα, γεγονός που οφείλεται στις 8 διαμερίσεις που χρησιμοποιήσαμε για αυτά τα τετράγωνα, σε σχέση με τις 4 των προηγουμένων.
- Οι μέσοι χρόνοι για τα full scan, όπως αναμενόταν, κινούνται στο ίδιο επίπεδο και για τις τρεις περιπτώσεις, αφού η πλευρά του τετραγώνου για τα ερωτήματα αυτά δεν παίζει κανένα ρόλο. Οι μικρές διαφορές που παρατηρούνται δεν έχουν σχέση με κάποιο χαρακτηριστικό που δώσαμε εμείς στη βάση, αλλά στην κατάσταση που θα βρίσκεται ο cluster κάθε φορά που κάνουμε ερώτημα. Οι χρόνοι αυτοί είναι τόσο μεγαλύτεροι από τους αντίστοιχους για κέντρο και προάστια, που συμπαρασύρουν τους συνολικούς χρόνους αρκετά προς τα πάνω και μειώνοντας πολύ τις διαφορές που

βλέπαμε για τα range queries.

- Σε σχέση με τα ερωτήματα που έγιναν για την “τυπική” δεικτοδότηση χωρίς τη βοήθεια του πίνακα 'timestamps', βλέπουμε ότι οι χρόνοι είναι σαφώς καλύτεροι όσον αφορά τα ερωτήματα σε κέντρο και προάστια και γενικά σε επίπεδο 50% μικρότεροι. Αυτό οφείλεται στα λιγότερα false-positive venues που ανακτούνται από τη βάση. Αντίθετα, όσον αφορά τα full scan οι χρόνοι είναι αρκετά μεγαλύτεροι, αφού η δεικτοδότηση με z-order curve εισάγει τόσο μεγαλύτερα κλειδιά, που χρειάζονται άρα περισσότερο χρόνο να ανακτηθούν, όσο και πολυπλοκότητα στην επεξεργασία: κάθε φορά που κάνουμε fetch ένα κλειδί, πρέπει να μετατρέψουμε τα bytes του από μια σειρά δεδομένων σε δύο διαφορετικούς αριθμούς, που παριστάνουν το γεωγραφικό πλάτος και μήκος του venue. Αυτό βέβαια απαιτεί περισσότερο χρόνο από ότι για την “τυπική” δεικτοδότηση, όπου η επεξεργασία του κλειδιού είναι πολύ πιο στοιχειώδης. Κοιτώντας του συνολικούς χρόνους, αν δεν έχουμε τη βοήθεια του πίνακα 'timestamps' είναι προτιμότερο να χρησιμοποιήσουμε την “τυπική” δεικτοδότηση.

4.1.3.4 Z-order curve δεικτοδότηση με τη βοήθεια του 'timestamps'

Στην τελευταία κατηγορία ερωτημάτων θα δοκιμάσουμε τις επιδόσεις της z-order curve δεικτοδότησης, χρησιμοποιώντας τη βοήθεια του πίνακα 'timestamps' όταν έχουμε να κάνουμε σάρωση για όλη την περιοχή για την οποία κρατάμε δεδομένα. Αυτά είναι τα scans στα οποία όπως είδαμε πριν ο πίνακας 'zBinVenue' υστερούσε, οπότε τώρα περιμένουμε τους καλύτερους χρόνους που έχουμε δει στα πειράματα αυτά.

CENTER						
Side	Number of queries	Total ms	Ms/query	Venues fetched	False-positive percentage	True venues
500	4431	66314	14,97	147372	0,46	79416
1000	4496	123878	27,55	696090	0,47	369180
2000	4419	302597	68,48	2301642	0,33	1534994

Πίνακας 38: Μέσοι χρόνοι ερωτημάτων για ερωτήματα και των τριών ειδών σε z-order curve δεικτοδότηση με τη βοήθεια του 'timestamps' – κέντρο.

SUBURBIA						
Side	Number of queries	Total ms	Ms/query	Venues fetched	False-positive percentage	True venues
500	4584	71425	15,58	42353	0,45	23388
1000	4519	81277	17,99	179725	0,48	92752
2000	4596	185379	40,33	637935	0,35	413263

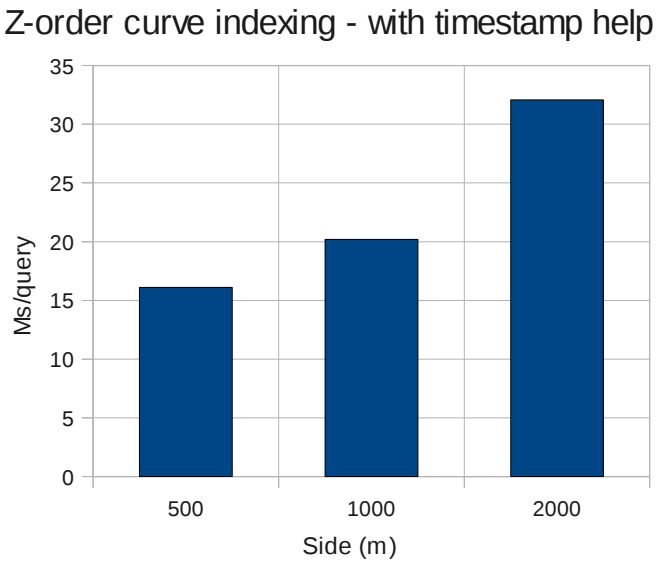
Πίνακας 39: Μέσοι χρόνοι ερωτημάτων για ερωτήματα και των τριών ειδών σε z-order curve δεικτοδότηση με τη βοήθεια του 'timestamps' – προάστια.

FULL SCANS			
Side	Number of queries	Total ms	Ms/query
500	985	52138	52,93
1000	985	51691	52,48
2000	985	49771	50,53

Πίνακας 40: Μέσοι χρόνοι ερωτημάτων για ερωτήματα και των τριών ειδών σε z-order curve δεικτοδότηση με τη βοήθεια του 'timestamps' – full scans.

OVERALL QUERIES		
Side	Number of queries	Ms/query
500	10000	16,10
1000	10000	20,19
2000	10000	32,07

Πίνακας 41: Συνολικοί μέσοι χρόνοι ερωτημάτων για ερωτήματα και των τριών ειδών σε z-order curve δεικτοδότηση με τη βοήθεια του 'timestamps' .



Εικόνα 33: Μέσος χρόνος ανά ερώτημα σε z-order curve δεικτοδότηση με τη βοήθεια του πίνακα timestamps για τα τρία είδη ερωτημάτων.

Εδώ συναντάμε την καλύτερη περίπτωση που έχουμε εξετάσει μέχρι τώρα, αφού συνδυάζει τόσο γρήγορους χρόνους όσον αφορά τα ερωτήματα σε κέντρο και προάστια, όσο και full scans για όλη την περιοχή της Αθήνας. Αυτό έχει αντίκτυπο στο συνολικό μέσο χρόνο των ερωτημάτων που είναι και ο μικρότερος που έχουμε δει μέχρι τώρα. Το Διάγραμμα 9 παραπάνω παρουσιάζει συμπυκνωμένα τους χρόνους αυτούς σε μορφή στηλών.

4.1.3.5 Παρουσίαση συνολικά των δεδομένων και συμπεράσματα

Παρακάτω παρουσιάζουμε δύο πίνακες που παραθέτουν συνοπτικά τους συνολικούς μέσους χρόνους για κάθε περίπτωση που εξετάσαμε ως τώρα, ώστε να βγάλουμε κάποια συμπεράσματα σχετικά με αυτά και την καταλληλότητα των σχημάτων που χρησιμοποιήσαμε στην HBase. Κάθε τιμή αντιπροσωπεύει το μέσο χρόνο για το μεγαλύτερο και ακριβέστερο σετ των 10.000 ερωτήσεων.

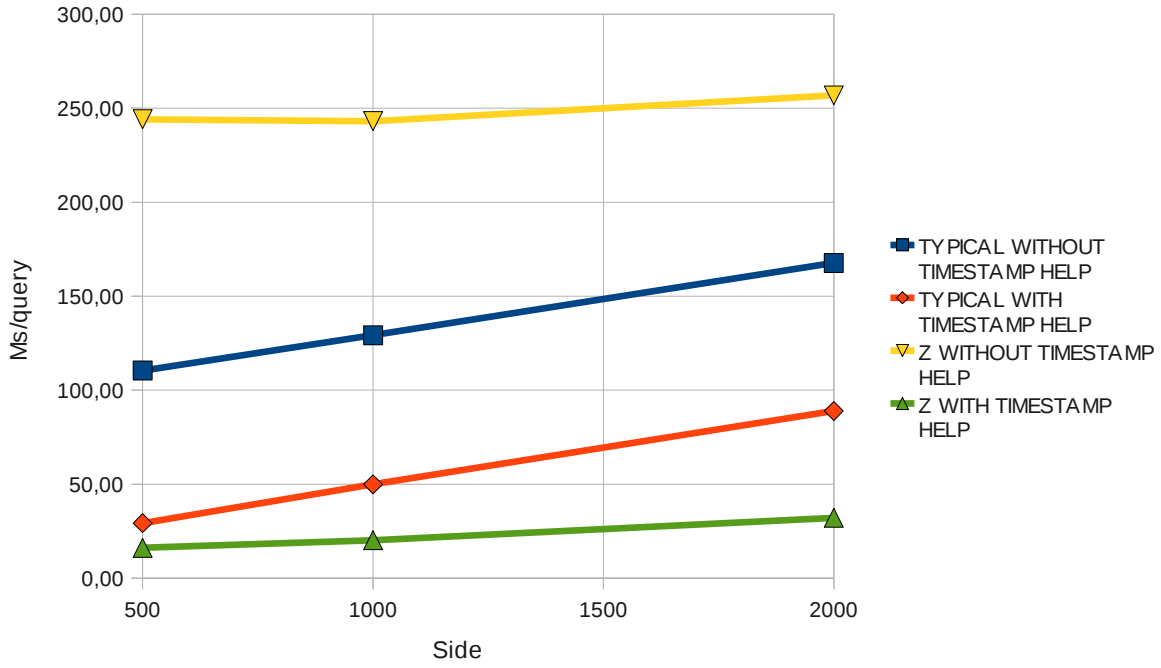
OUTCOMES FOR 10.000 QUERIES USING TYPICAL INDEXING		
Side	WITHOUT TIMESTAMP HELP (ms/query)	WITH TIMESTAMP HELP (ms/query)
500	110,43	29,25
1000	129,25	49,91
2000	167,70	88,88

Πίνακας 42: Συνολικοί μέσοι χρόνοι και για τα τρία είδη ερωτημάτων σε "τυπική" δεικτοδότηση με και χωρίς τη βοήθεια του πίνακα 'timestamps'.

OUTCOMES FOR 10.000 QUERIES USING Z-ORDER CRUVE INDEXING		
Side	WITHOUT TIMESTAMP HELP (ms/query)	WITH TIMESTAMP HELP (ms/query)
500	244,16	16,10
1000	243,12	20,19
2000	256,84	32,07

Πίνακας 43: Συνολικοί μέσοι χρόνοι και για τα τρία είδη ερωτημάτων σε z-order curve δεικτοδότηση με και χωρίς τη βοήθεια του πίνακα 'timestamps'.

Για την καλύτερη σύγκριση των τιμών παραθέτουμε και ένα διάγραμμα για κάθε είδος ερωτημάτων, ανάλογα με την πλευρά του τετραγώνου που γίνεται η ερώτηση, που περιέχει τους μέσους συνολικούς χρόνους για κάθε είδος δεικτοδότησης που χρησιμοποιήσαμε.



Εικόνα 34: Συνολικοί χρόνοι για ερωτήματα τετραγώνων πλευράς 500, 1.000 και 2.000 μέτρων ανάλογα με το είδος της δεικτοδότησης.

Κοιτάζοντας προσεχτικά το μέσο χρόνο κάθε περίπτωσης μπορούμε να καταλήξουμε πως:

- Όποιο τρόπο δεικτοδότησης και να χρησιμοποιήσουμε η χρήση του πίνακα 'timestamps' πάντα βοηθάει στους χρόνους.
- Η “τυπική” δεικτοδότηση είναι καλύτερη από την z-order curve όταν δεν χρησιμοποιείται ο πίνακας 'timestamps', λόγω του πολύ μεγάλου χρόνου που απαιτείται στη δεύτερη περίπτωση για τα full scans και οφείλεται στην επιπλέον πολυπλοκότητα που εισάγει η καμπύλη.
- Όταν ο 'timestamps' χρησιμοποιείται η δεικτοδότηση με z-order curve δίνει τους καλύτερους χρόνους. Τώρα πλέον οι χρόνοι για full scans είναι ίδιοι και στις δύο περιπτώσεις και φαίνεται η δυνατότητα της δεικτοδότησης με βάση τη z-order curve να μειώνει τα false-positive venues που επιστρέφονται και κατά συνέπεια τους χρόνους ολοκλήρωσης των ερωτημάτων.

Έτσι, με το τέλος της ενότητας αυτής φτάνουμε στο συμπέρασμα ότι ο πίνακας 'timestamps' είναι απαραίτητος για μια ολοκληρωμένη χωροχρονική πλατφόρμα που θα προσφέρει τα δεδομένα της σε διάφορες εφαρμογές που θα τα χρησιμοποιούν, αφού καλύπτει την περίπτωση να ζητηθούν οι συνολικοί χρόνοι για όλη την Αθήνα. Στην περίπτωση που ζητηθούν ερωτήματα εύρους, τότε παρά την μεγαλύτερη πολυπλοκότητα και μήκος κλειδιών, η δεικτοδότηση με την z-order curve είναι η πλέον ευέλικτη και ενδεδειγμένη.

4.2 Εξερεύνηση δυνατότητας batch ερωτημάτων

Μέχρι τώρα εξετάσαμε το αποτέλεσμα που θα είχαν διάφορες μορφές αποθήκευσης και δεικτοδότησης χωροχρονικών δεδομένων στην HBase εφαρμόζοντας ερωτήματα και μετρώντας μέσους χρόνους - σε όλες τις περιπτώσεις όμως τα range queries γινόταν σειριακά.

Αυτό σημαίνει ότι αν χρησιμοποιούσαμε, για παράδειγμα, το σετ των 10.000 ερωτήσεων, κάθε ερώτηση θα πραγματοποιούταν σειριακά, θα μετρούνταν ο χρόνος εκτέλεσής της και μέσω αυτής της διαδικασίας θα υπολογιζόταν τελικά ο μέσος χρόνος των 10.000 ερωτήσεων. Έτσι, ένας μέσος χρόνος 20 ms/query, θα σήμαινε για το σετ αυτό ότι θα χρειαζόταν 200.000 ms περίπου για την ολοκλήρωση του σετ, κάτι που αντιστοιχεί σε 3,33 λεπτά. Ο χρόνος που θα χρειαζόταν για ένα αντίστοιχο σετ 100.000 ερωτήσεων θα ήταν περίπου μισή ώρα.

Η πλατφόρμα χωροχρονικών δεδομένων που στήσαμε κρατάει δεδομένα που έχουν άμεση σχέση με την κατάσταση που επικρατεί στους δρόμους της πόλης την τρέχουσα ώρα, και έτσι σε περίπτωση που δεχθεί αριθμό ερωτημάτων της παραπάνω τάξης, μισή ώρα θα ήταν μεγάλη καθυστέρηση. Για το λόγο αυτό στην ενότητα θα εξετάσουμε αν μπορούμε να επιταχύνουμε την απάντηση ερωτημάτων, που φτάνουν στην HBase σε μεγάλο αριθμό. Η τεχνοτροπία που θα εξετάσουμε βασίζεται στην ομαδοποίηση πολλών ερωτημάτων scan σε μια ομάδα (batch) και η εκτέλεσή τους σαν σύνολο στη βάση.

Η ιδέα αυτή δεν είναι καινούρια στην HBase. Λόγω των ακριβών διαβασμάτων και εγγραφών δίνει τη δυνατότητα στο χρήστη να ομαδοποιεί τα Put και Get, τις μεθόδους δηλαδή που προσφέρει μέσω του API της για ανάγνωση και εγγραφή δεδομένων, σε ομάδες που μπορεί να περιλαμβάνουν μεγάλο αριθμό μεμονωμένων πράξεων και την εκτέλεσή τους σαν ομάδα σε αυτή. Έτσι επιταχύνεται η εγγραφή ή ανάγνωση, ενώ ταυτόχρονα δεν επιβαρύνεται το πρόγραμμα client περιμένοντας απόκριση για κάποια δοσοληψία. Ωστόσο η HBase δεν δίνει κάποια αντίστοιχη δυνατότητα ομαδοποίησης των scan, με αποτέλεσμα να πρέπει ο χρήστης να τα καλεί σειριακά.

Αυτό το κενό προσπαθήσαμε να το καλύψουμε με τον αλγόριθμο που θα περιγραφεί παρακάτω, αφού σκεφτήκαμε πως θα μπορούσε να βοηθήσει καταστάσεις με μεγάλο αριθμό ερωτήσεων στη βάση. Το κύριο σκεπτικό είναι πως αν εκτελούμε πολλά range queries για μια πόλη όπως η Αθήνα, είναι πολύ πιθανό αρκετά από αυτά τα scan να επικαλύπτονται. Με αυτό εννοούμε πως μπορεί, για παράδειγμα, δύο συνεχόμενα scan να μοιράζονται πολλά από τα venues που θέλουν να ανακτήσουν από τη βάση. Με τον σειριακό τρόπο που εκτελούνται όμως στην HBase τα κοινά venues θα έρθουν στη μνήμη του προγράμματος πελάτη, θα επεξεργαστούν μια φορά για τη μία ερώτηση και κατόπιν θα ξαναζητηθούν για το επόμενο query. Αντίθετα αν κρατούνταν στη μνήμη ο πελάτης θα γλίτωνε το χρονοβόρο scan και θα μπορούσε να επεξεργαστεί το δεύτερο ερώτημα απευθείας από τη μνήμη του.

Ο αλγόριθμος που εφαρμόσαμε είναι απλός. Για ένα σετ ερωτημάτων, ομαδοποιούμε τις ερωτήσεις (scans) σε μια λίστα και τη στέλνουμε ολόκληρη στη μέθοδο που θα την επεξεργαστεί. Αυτή για ολόκληρο το σετ βρίσκει το ελάχιστο και το μέγιστο κλειδί, ανάμεσα στα οποία βρίσκονται τα venues που θα ζητηθούν από τη βάση. Επίσης, βρίσκουμε το μικρότερο και μεγαλύτερο timestamp, αφού σε μια πραγματική εφαρμογή και την πλατφόρμα να τρέχει για μήνες – εμείς συλλέξαμε δεδομένα για μια βδομάδα μόνο – κάτι τέτοιο θα μπορούσε να μειώσει κατά πολύ τα δεδομένα που θα ανακτηθούν – πόσο μάλλον αν υποθέσουμε ότι μεγάλο ποσοστό των χρηστών θα ζητάει timestamps που θα αφορούν την τρέχουσα ώρα.

Αφού γνωρίζουμε τις παραμέτρους αυτές ζητάμε χρησιμοποιώντας τες ένα μεγάλο scan από την HBase και αποθηκεύουμε τα αποτελέσματα στη μνήμη. Τα venues μπαίνουν σε μια ταξινομημένη λίστα, ενώ με ταξινομημένη μορφή αποθηκεύονται και τα here now που αντιστοιχούν σε διάφορα timestamps του ίδιου venue μέσα σε μία κλάση που αντιπροσωπεύει το αντικείμενο αυτό.

Έχοντας όλη την πληροφορία που χρειαζόμαστε στη μνήμη έρχεται η σειρά να απαντήσουμε στα ερωτήματα. Μεγάλο ρόλο εδώ παίζει ότι η λίστα με τα venues είναι όπως είπαμε ταξινομημένη. Έτσι για ένα δεδομένο scan, μέλος του σετ με τα ερωτήματα, γνωρίζουμε ήδη τις συντεταγμένες που οριοθετούν τις χρήσιμες εγγραφές και μπορούμε με

δύο binary search να βρούμε την αρχή και το τέλος του κομματιού της λίστας που μας ενδιαφέρει. Binary search εφαρμόζεται και στα here now κάθε venue ξεχωριστά, ώστε να υπολογιστεί ο συνολικός κόσμος που βρίσκεται στην περιοχή που οριοθετεί το τρέχον ερώτημα.

Η παραπάνω τεχνοτροπία περιμένουμε να δώσει καλά αποτελέσματα σε ομάδες πολλών scan, αφού η πρόσβαση στη μνήμη που εξασφαλίζει για τα ζητούμενα δεδομένα είναι γρηγορότερη από τις RPC κλήσεις στην HBase. Επειδή η πιθανότητα να μην υπάρχουν πολλά κοινά σημεία σε μικρό αριθμό ερωτημάτων είναι μεγάλη, ωστόσο, είναι λογικό να μην αποδίδει τόσο καλά σε αυτή την περίπτωση. Επίσης, θα έχει ενδιαφέρον το μέγεθος των ερωτημάτων που μπορεί να υποστηρίξει η μνήμη, αφού όλες αυτές οι λίστες που αποθηκεύονται τοπικά πιάνουν χώρο.

Για να εξετάσουμε όλα τα παραπάνω χρησιμοποιήσαμε σετ ερωτήσεων που ξεκινούν από ένα ερώτημα και φτάνουν τα 10 εκατομμύρια. Κάθε σετ εκτελούταν για τετράγωνα πλευράς 500, 1.000 και 2.000 μέτρων. Για κάθε σετ, μετρήσαμε τις εγγραφές που χρειάστηκε να ανακτηθούν από τη βάση, τα venues που συνολικά σαρώθηκαν, το ποσοστό των venues αυτών σχετικά με το συνολικό μέγεθος της βάσης, το χρόνο ολοκλήρωσης του σετ και τον μέσο χρόνο για κάθε ερώτημα. Παρουσιάζουμε τα αποτελέσματα παρακάτω μέσω πινάκων.

Side = 500m - TYPICAL					
Scans	Venues fetched	% Base percentage	Total millis	Millis/scan	Venues examined
1	483	2,19	315	315,00	4
10	3900	17,68	2360	236,00	436
100	5008	22,70	3124	31,24	5120
1000	5168	23,42	3749	3,75	46491
10000	5174	23,45	9009	0,90	472388
100000	5174	23,45	60559	0,61	4715215
1000000	5174	23,45	576477	0,58	47060678

Πίνακας 44: Ομαδοποιημένα ερωτήματα τετραγώνων πλευράς 500 μέτρων για "τυπική" δεικτοδότηση.

Side = 500m – Z-ORDER INDEXING					
Scans	Venues fetched	% Base percentage	Total millis	Millis/scan	Venues examined
1	2725	12,35	1273	1272,56	4
10	3732	16,91	2512	251,20	436
100	5006	22,69	3876	38,76	5120
1000	5167	23,42	4742	4,74	46491
10000	5174	23,45	11377	1,14	472388
100000	5174	23,45	87142	0,87	4715215
1000000	5174	23,45	5616571	5,62	47060678

Πίνακας 45: Ομαδοποιημένα ερωτήματα τετραγώνων πλευράς 500 μέτρων για z-order curve δεικτοδότηση.

Side = 1000m - TYPICAL					
Scans	Venues fetched	% Base percentage	Total millis	Millis/scan	Venues examined
1	1357	6,15	886	886,00	60
10	4736	21,46	2958	295,80	2466
100	5136	23,28	3328	33,28	22128
1000	5174	23,45	4632	4,63	217010
10000	5174	23,45	17428	1,74	2137391
100000	5174	23,45	145755	1,46	21359867
1000000	5174	23,45	1424128	1,42	213794612

Πίνακας 46: Ομαδοποιημένα ερωτήματα τετραγώνων πλευράς 1.000 μέτρων για "τυπική" δεικτοδότηση.

Side = 1000m – Z-ORDER INDEXING					
Scans	Venues fetched	% Base percentage	Total millis	Millis/scan	Venues examined
1	262	14,27	1631	1631,52	60
10	5513	24,98	4282	428,17	2466
100	4941	22,39	4107	41,07	22128
1000	5173	23,45	6032	6,03	217010
10000	5174	23,45	24573	2,46	2137391
100000	5174	23,45	208144	2,08	21359867
1000000	5174	23,45	33517711	33,52	213794612

Πίνακας 47: Ομαδοποιημένα ερωτήματα τετραγώνων πλευράς 1.000 μέτρων για z-order curve δεικτοδότηση.

Side = 2000m - TYPICAL					
Scans	Venues fetched	% Base percentage	Total millis	Millis/scan	Venues examined
1	3694	16,74	2209	2209,00	1342
10	4938	22,38	3100	310,00	9116
100	5126	23,23	3433	34,33	97997
1000	5174	23,45	7230	7,23	931450
10000	5174	23,45	43799	4,38	9331158
100000	5174	23,45	402703	4,03	92833427
1000000	5174	23,45	3990733	3,99	927930774

Πίνακας 48: Ομαδοποιημένα ερωτήματα τετραγώνων πλευράς 2.000 μέτρων για "τυπική" δεικτοδότηση.

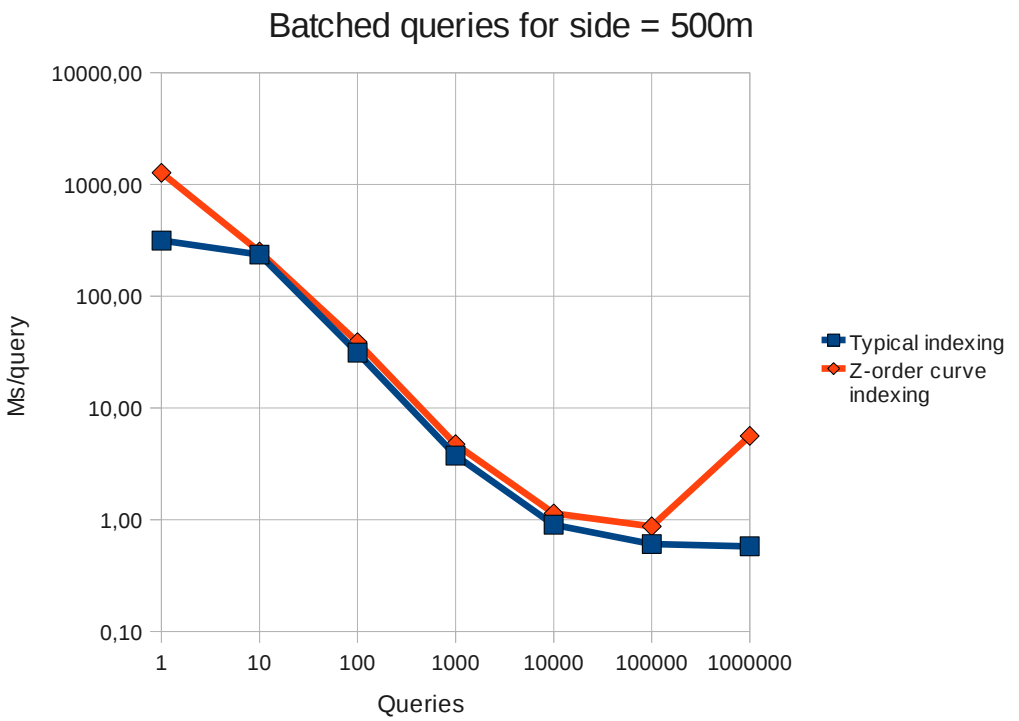
Side = 2000m – Z-ORDER INDEXING					
Scans	Venues fetched	% Base percentage	Total millis	Millis/scan	Venues examined
1	4328	19,61	2857	2857,01	1342
10	4964	22,50	3708	370,81	9116
100	5052	22,89	4259	42,59	97997
1000	5174	23,45	9566	9,57	931450
10000	5174	23,45	64923	6,49	9331158
100000	5174	23,45	711855	7,12	92833427
1000000	5174	23,45	154181469	154,18	927930774

Πίνακας 49: Ομαδοποιημένα ερωτήματα τετραγώνων πλευράς 2.000 μέτρων για z-order curve δεικτοδότηση.

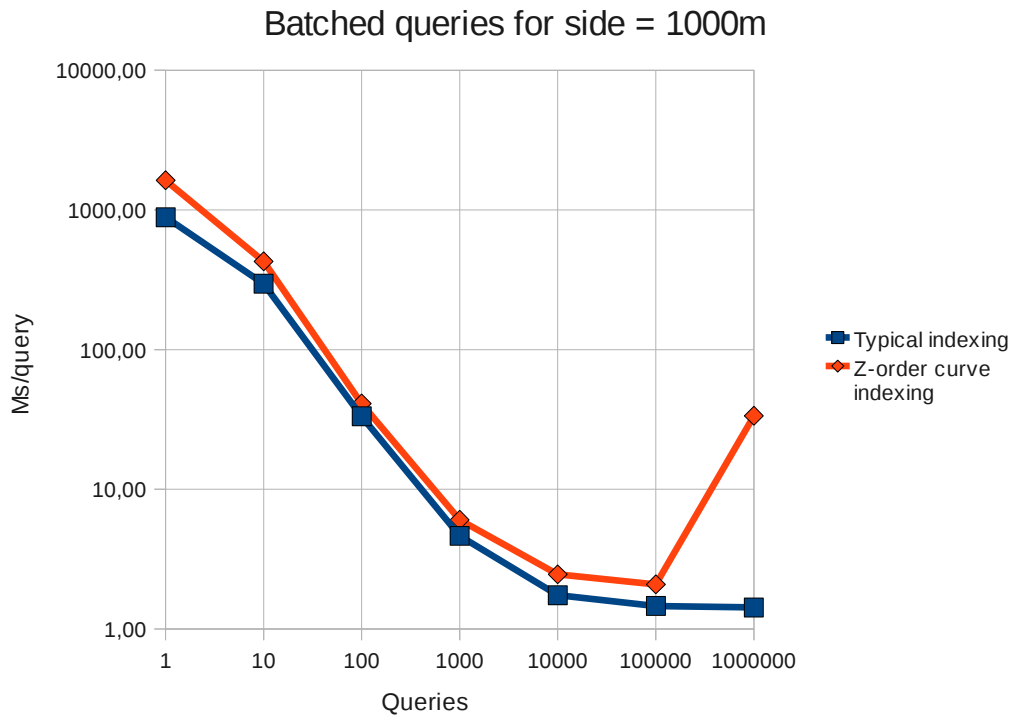
scans	Total minutes to complete all scans					
	Side = 500m		Side = 1000m		Side = 2000m	
	Typical	Z-order curve	Typical	Z-order curve	Typical	Z-order curve
1	0,01	0,02	0,01	0,03	0,04	0,05
10	0,04	0,04	0,05	0,07	0,05	0,06
100	0,05	0,06	0,06	0,07	0,06	0,07
1.000	0,06	0,08	0,08	0,10	0,12	0,16
10.000	0,15	0,19	0,29	0,41	0,73	1,08
100.000	1,01	1,45	2,43	3,47	6,71	11,86
1.000.000	9,61	93,61	23,74	558,63	66,51	2569,69

Πίνακας 50: Συνολικά λεπτά για την ολοκλήρωση των ομάδων ερωτημάτων.

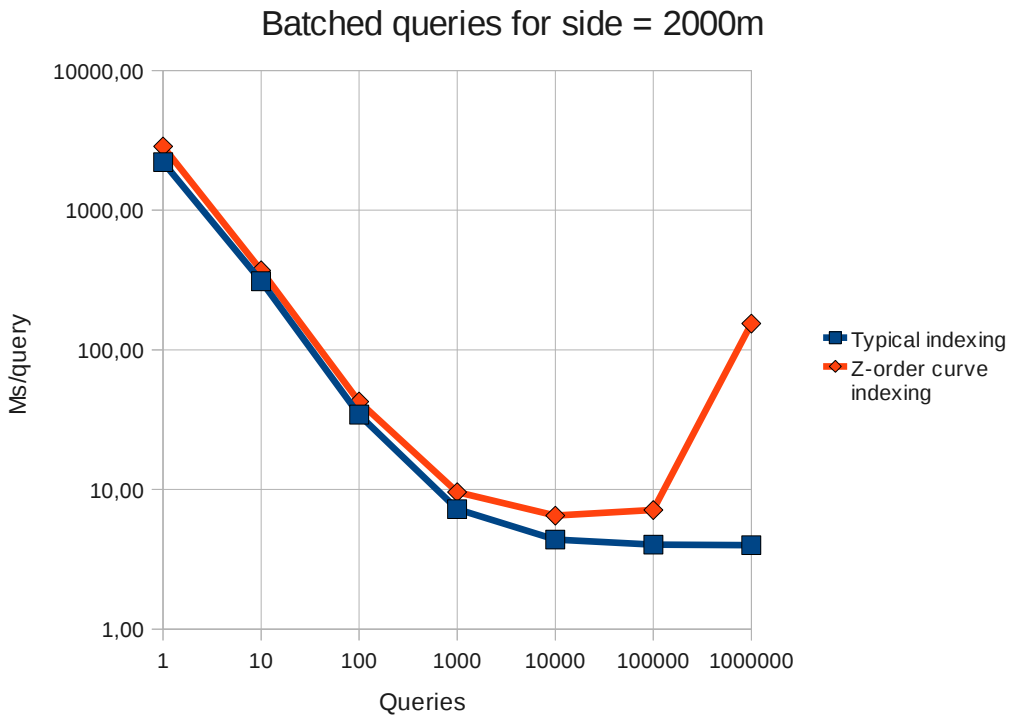
Επίσης για την οπτικοποίηση της πληροφορίας που πήραμε από τα πειράματα, παραθέτουμε παρακάτω και διαγράμματα που παρουσιάζουν το μέσο χρόνο ολοκλήρωσης ερωτήματος σε σχέση με το μέγεθος του αριθμού των ερωτημάτων, που απαρτίζουν μια ομάδα. Και για τους δύο άξονες χρησιμοποιήθηκε λογαριθμική κλίμακα ώστε να φανούν καλύτερα οι διαφορές και οι τάσεις των τιμών.



Εικόνα 35: Μέσοι χρόνοι ομαδοποιημένων ερωτήσεων τετραγώνων πλευράς 500 μέτρων.



Εικόνα 36: Μέσοι χρόνοι ομαδοποιημένων ερωτήσεων τετραγώνων πλευράς 1.000 μέτρων.



Εικόνα 37: Μέσοι χρόνοι ομαδοποιημένων ερωτήσεων τετραγώνων πλευράς 2.000 μέτρων.

Τα συμπεράσματα που βγάζουμε από τα παραπάνω είναι τα εξής:

- Μέχρι και την ομαδοποίηση των 10 ερωτήσεων, το σχήμα αυτό επιφέρει χρόνους ανά ερώτηση πολύ μεγαλύτερους από τους αντίστοιχους που είδαμε όταν τα ερωτήματα πραγματοποιούνταν σειριακά. Ένα ακόμη ενδιαφέρον στοιχείο για αυτά τα σετ ερωτήσεων είναι ο αριθμός των venues που ανακτούνται από τη βάση, σχετικά με αυτά που ουσιαστικά χρειάζονται στην ερώτηση. Και στις δύο πρώτες περιπτώσεις τα false-positive venues είναι πολύ περισσότερα από τα απαιτούμενα και το πόσο μεγάλος θα είναι ο μέσος χρόνος καθορίζεται από τον αριθμό των false-positive venues που θα ανακτηθούν από τη βάση, καθυστερώντας τη μεταφορά και την επεξεργασία και οδηγώντας έτσι σε μεγαλύτερο χρόνο.
- Στο σετ των 100 ερωτήσεων οι μέσοι χρόνοι που παίρνουμε με αυτή την τεχνική γίνονται συγκρίσιμοι με αυτούς που παίρναμε στη σειριακή εκτέλεση των ερωτημάτων. Εδώ βλέπουμε πως ο αριθμός των venues που πρέπει να διαβαστούν γίνεται περίπου ίσος ή μεγαλύτερος από τις εγγραφές που συνολικά ανακτούνται από τη βάση και φανερώνεται έτσι η επίδραση που έχει η επαναχρησιμοποίηση εγγραφών, λόγω της διατήρησής τους στη μνήμη.
- Από την ομαδοποίηση που περιέχει χίλιες, μέχρι το σετ των 100 χιλιάδων ερωτήσεων, τα αποτελέσματα εμφανίζουν ένα σταθερό μοτίβο συμπεριφοράς. Τα venues που χρειάζονται στα ερωτήματα αυξάνονται σταδιακά ανά μια δεκάδα, ξεπερνώντας σε αριθμό τις εγγραφές που βρίσκονται στη μνήμη και δημιουργώντας τις καταστάσεις που θέλουμε να εκμεταλλευτούμε και εξηγούσαμε στις αρχές της ενότητας. Σαν αποτέλεσμα βλέπουμε τους μέσους χρόνους των σαρώσεων να πέφτουν συνεχώς,

φτάνοντας στο εντυπωσιακό 0,58 ms/scan για τετράγωνα πλευράς 500 μέτρων στο σετ του ενός εκατομμυρίου ερωτημάτων.. Η εξήγηση είναι απλή: ο συνολικός χρόνος επεξεργασίας του σετ αυξάνεται με πολύ μικρότερο ρυθμό από ότι τα ερωτήματα που γίνονται, μειώνοντας έτσι τους μέσους χρόνους ανά ερώτημα. Αξίζει να παρατηρήσουμε επίσης ότι οι χρόνοι αυξάνονται όσο αυξάνεται το εύρος των range queries, αφού μεγαλώνει και ο αριθμός των venues που πρέπει να ανακτηθούν και επεξεργαστούν από τη μνήμη. Επίσης η z-order curve δεικτοδότηση πετυχαίνει μεγαλύτερους χρόνους από την “τυπική”. Αυτό οφείλεται στη δυσκολότερη επεξεργασία των κλειδιών, που αποτελούν τις z-τιμές των συντεταγμένων και χρειάζονται επιπλέον χρόνο για την εξαγωγή των γεωγραφικών πλατών και μηκών που κρύβουν μέσα τους.

- Στο σετ του ενός εκατομμυρίου ερωτήσεων οι μέσοι χρόνοι ανά ερώτημα όσον αφορά την “τυπική” δεικτοδότηση συνεχίζουν να πέφτουν ενώ αντίθετα αυτοί της z-order curve δεικτοδότησης ανεβαίνουν. Αυτό οφείλεται στο ότι συνολικός χρόνος επεξεργασίας του batch αυξάνεται απότομα, σε σχέση με την αύξηση που βλέπαμε μέχρι τώρα και σχετίζεται με τις μεγάλες απαιτήσεις μνήμης από το πρόγραμμα. Οι αντίστοιχες απαιτήσεις για την περίπτωση της “τυπικής” δεικτοδότησης είναι αρκετά μικρότερες, αφού όπως θυμόμαστε χρησιμοποιεί κλειδιά με μήκος περίπου 30 bytes, σε αντίθεση με τα 100 bytes που αντιστοιχούν στις z-τιμές των συντεταγμένων.
- Η τρίτη στήλη των πινάκων αφορά το ποσοστό της βάσης που ανακτάται και αντιπροσωπεύει τα venues που πιστεύεται από την τεχνική πως θα χρειαστούν για το σετ των ερωτημάτων που πρόκειται να γίνουν. Βλέπουμε πως για ένα μεμονωμένο ερώτημα το ποσοστό είναι πολύ μικρό, αφού γίνονται fetch μόνο οι εγγραφές που ανήκουν σε ένα τετράγωνο. Το ποσοστό είναι μεγαλύτερο για τον z πίνακα, γιατί δε χρησιμοποιήσαμε κάποια τεχνική διάσπασης του αρχικού ερωτήματος, αλλά εξετάσαμε ποια venues είναι όντως εντός εύρους από τη μνήμη. Σταδιακά το ποσοστό των εγγραφών που ανακτούνται αυξάνεται και σταθεροποιείται στο 23,45%. Αυτό μας δείχνει πως ο αριθμός των venues που βρίσκονται δεικτοδοτημένα στην περιοχή που εξετάζουμε είναι αυτός ακριβώς και φανερώνει τη δυνατότητα των ομαδοποιημένων ερωτημάτων να διαβάζουν από τη βάση μέρος των εγγραφών ανάλογα με την περιοχή στην οποία βρίσκονται. Ο αριθμός των venues δεν αυξάνεται από εκεί και πέρα και δεν είναι αρκετός για να οδηγήσει τη μνήμη που χρησιμοποιείται σε επικίνδυνα υψηλά επίπεδα. Αυτό συμβαίνει λόγω των μεταδεδομένων που χρησιμοποιούνται σε κάθε σετ και αφορούν πληροφορίες για κάθε ερώτημα.
- Ενδιαφέρον προς αυτή την κατεύθυνση παρουσιάζει και ο τελευταίος πίνακας που εμφανίζει σε λεπτά το χρόνο ολοκλήρωσης ολόκληρου του σετ ερωτήσεων. Παρατηρούμε ότι ακόμη και στην απαιτητική ομάδα των 100 χιλιάδων ερωτήσεων ο χρόνος αυτός κινείται κατά μέσο όρο στα δύο λεπτά, χρόνος ο οποίος μπορεί να θεωρηθεί λογικός για μια real-time ερώτηση. Αν τον συγκρίνουμε μάλιστα με τη μισή ώρα που θα πετυχαίναμε με σειριακή εκτέλεση των αντίστοιχων ερωτήσεων, ακόμη και με το καλύτερο σχήμα της χρήσης των πινάκων 'zBinVenue' και 'timestamps', το κέρδος είναι εμφανές. Από την άλλη στις ένα εκατομμύριο ερωτήσεις ο χρόνος αυτός ανεβαίνει σημαντικά, ειδικά στη z-order curve δεικτοδότηση. Ακόμη όμως και ημία ώρα της “τυπικής” δεικτοδότησης για ερωτήματα εύρους 2.000 μέτρων δεν θεωρείται επαρκής για μια real-time εφαρμογή.
- Το όριο στη χρήση μνήμης εμφανίζεται στο τελευταίο testcase που δεν εμφανίζεται στης μετρήσεις και αποτελούταν από 10 εκατομμύρια ερωτήσεις. Στην περίπτωση αυτή δεν πήραμε στατιστικά αφού το πρόγραμμα δεν τελείωσε ποτέ, με την Java να εμφανίζει εξαίρεση που δηλώνει ότι ο χώρος μνήμης που δεσμεύτηκε για τη

συγκεκριμένη διεργασία δεν ήταν αρκετός. Έτσι βλέπουμε στην πράξη τον περιορισμό της μνήμης που υπάρχει για αυτή την τεχνική.

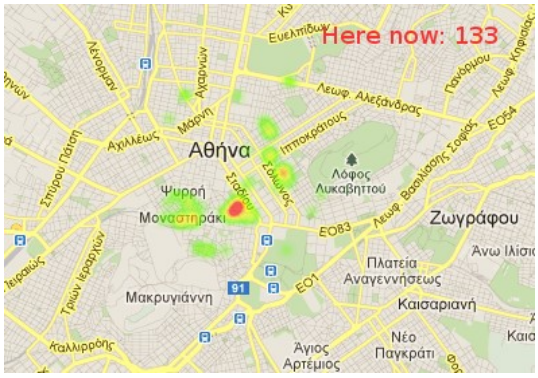
Σαν κατακλείδα της ενότητας μπορούμε να πούμε ότι μια εφαρμογή που θα στηριχθεί στην χωροχρονική πλατφόρμα που στήσαμε στα πλαίσια της διπλωματικής, μπορεί να χρησιμοποιήσει την τεχνική της ομαδοποίησης ερωτημάτων στην περίπτωση που αυτά αυξηθούν κάποια στιγμή πάρα πολύ και η σειριακή εκτέλεσή τους οδηγεί σε μεγάλες καθυστερήσεις. Το σχήμα αυτό δουλεύει πολύ καλά από τις χίλιες μέχρις τις 100 χιλιάδες ερωτήσεις, ενώ για σετ του ενός εκατομμυρίου ερωτήσεων παρουσιάζει πολύ μεγάλους χρόνους απόκρισης. Σε κάθε περίπτωση όμως τα αποτελέσματα είναι πολύ καλύτερα από τα αντίστοιχα που θα είχαμε με τη σειριακή εκτέλεση αυτού του μεγάλου αριθμού ερωτημάτων.

4.3 Usecase: μια εφαρμογή που στηρίζεται στη χωροχρονική πλατφόρμα

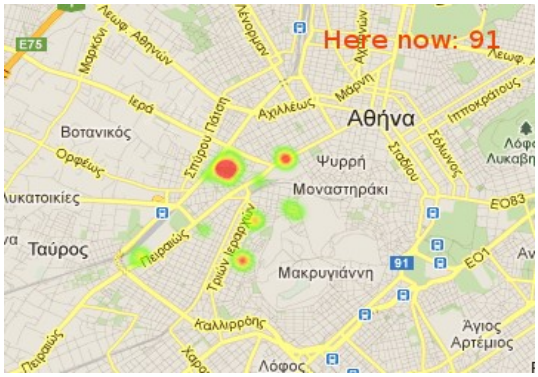
Έχουμε αναφέρει πως η πλατφόρμα που δημιουργήσαμε μπορεί να χρησιμοποιηθεί από διάφορες εφαρμογές, που μέσω ερωτημάτων θα προσπελάζουν τη χωροχρονική πληροφορία της και θα την προσφέρουν στους χρήστες τους. Ένα τέτοιο παράδειγμα θα δώσουμε εδώ, μέσω του usecase μιας εφαρμογής που δέχεται από τους χρήστες ερωτήματα σχετικά με το here now μιας τετραγωνικής περιοχής και τους επιστρέφει τόσο τον ζητούμενο αριθμό, όσο και ένα χάρτη πυκνότητας των χρηστών των LBS.

Ένας χάρτης πυκνοτήτων (heat map) είναι μια γραφική παρουσίαση των δεδομένων, στην οποία οι τιμές που περιέχονται σε κάποιο πίνακα παριστάνονται με χρώματα. Αποτελεί ιδιαίτερα χρήσιμη τεχνική στο χώρο των γεωγραφικών δεδομένων, αφού μετατρέπει μεγάλη μη-αναγνώσιμη πληροφορία που δεικτοδοτείται με γεωγραφικές συντεταγμένες σε γνώριμο για το χρήστη χάρτη. Έτσι και εμείς απεικονίσαμε την πληροφορία των χρηστών των location based services που βρίσκονται σε κάθε venue της Αθήνας, από την HBase στην οποία είναι αποθηκευμένη σε ένα χάρτη της Αθήνας με χρώματα που απεικονίζουν την παρουσία χρηστών. Συγκεκριμένα ο χάρτης θα εμφανίζεται χωρίς κανένα επιπλέον χρώμα όταν δεν υπάρχουν χρήστες που έχουν κάνει check-in για την περιοχή που ζητάμε, ενώ για κάθε check-in θα προστίθεται μία κηλίδα που θα αλλάζει χρώμα ανάλογα με τον αριθμό των χρηστών που βρίσκονται εκεί. Στην περίπτωση που οι χρήστες είναι λίγοι το χρώμα της θα είναι πράσινο, ενώ όσο μεγαλώνει ο αριθμός τους η κηλίδα θα γίνεται πορτοκαλί και έπειτα κόκκινη. Η θέση της θα είναι η θέση ακριβώς του venue στο οποίο οι χρήστες έχουν κάνει check-in. Για τη δημιουργία των heat map αυτών χρησιμοποιήσαμε το API του google-maps, που μας πρόσφερε ένα σύστημα χαρτών γνωστό στο μέσο χρήστη καθώς και ευκολία δημιουργίας των χαρτών πυκνοτήτων και ενσωμάτωσης στην εφαρμογή υπολογισμού του here now μιας περιοχής.

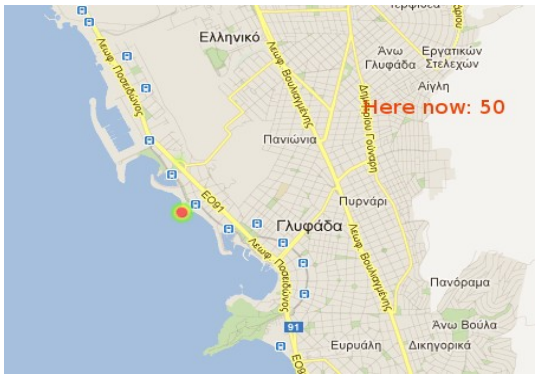
Παρακάτω παραθέτουμε τέσσερα στιγμιότυπα από την εφαρμογή αυτή, που αντιπροσωπεύουν ερωτήματα που έγιναν για το κέντρο και τα προάστια της Αθήνας. Πάνω στις εικόνες φαίνεται και το here now κάθε εφαρμογής, το σύνολο δηλαδή των χρηστών που βρίσκονταν εκεί κατά τη στιγμή λήψης του στιγμιοτύπου.



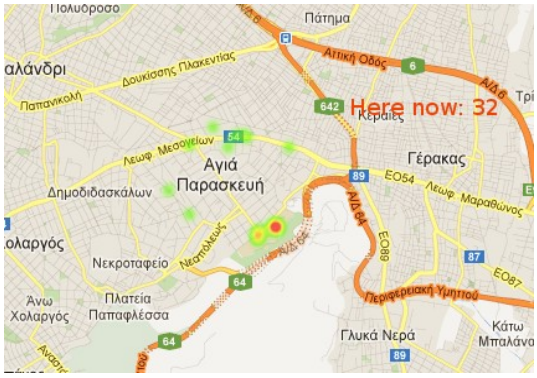
Εικόνα 38: Στιγμιότυπο 1



Εικόνα 39: Στιγμιότυπο 2

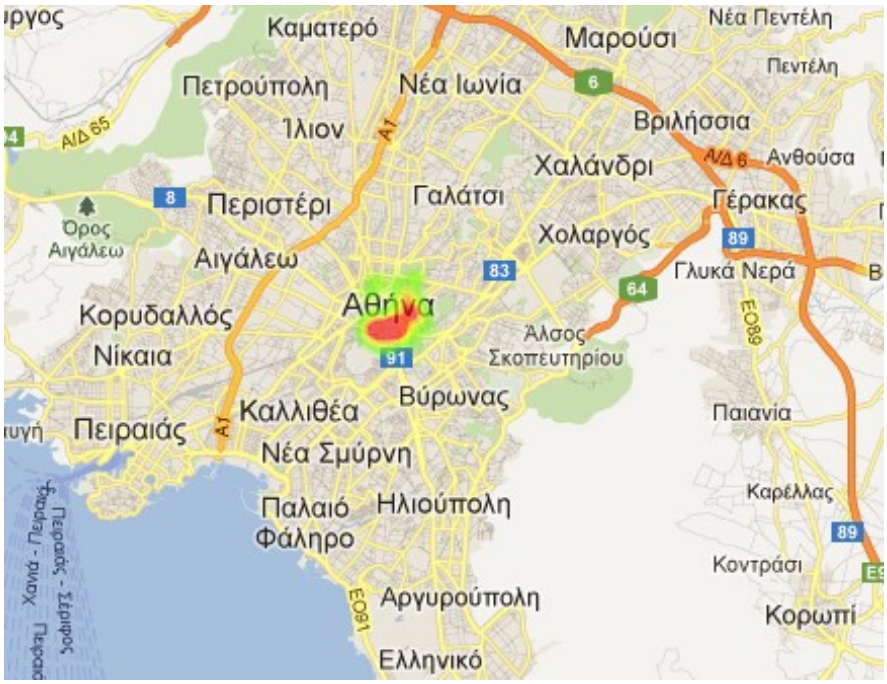


Εικόνα 41: Στιγμιότυπο 3

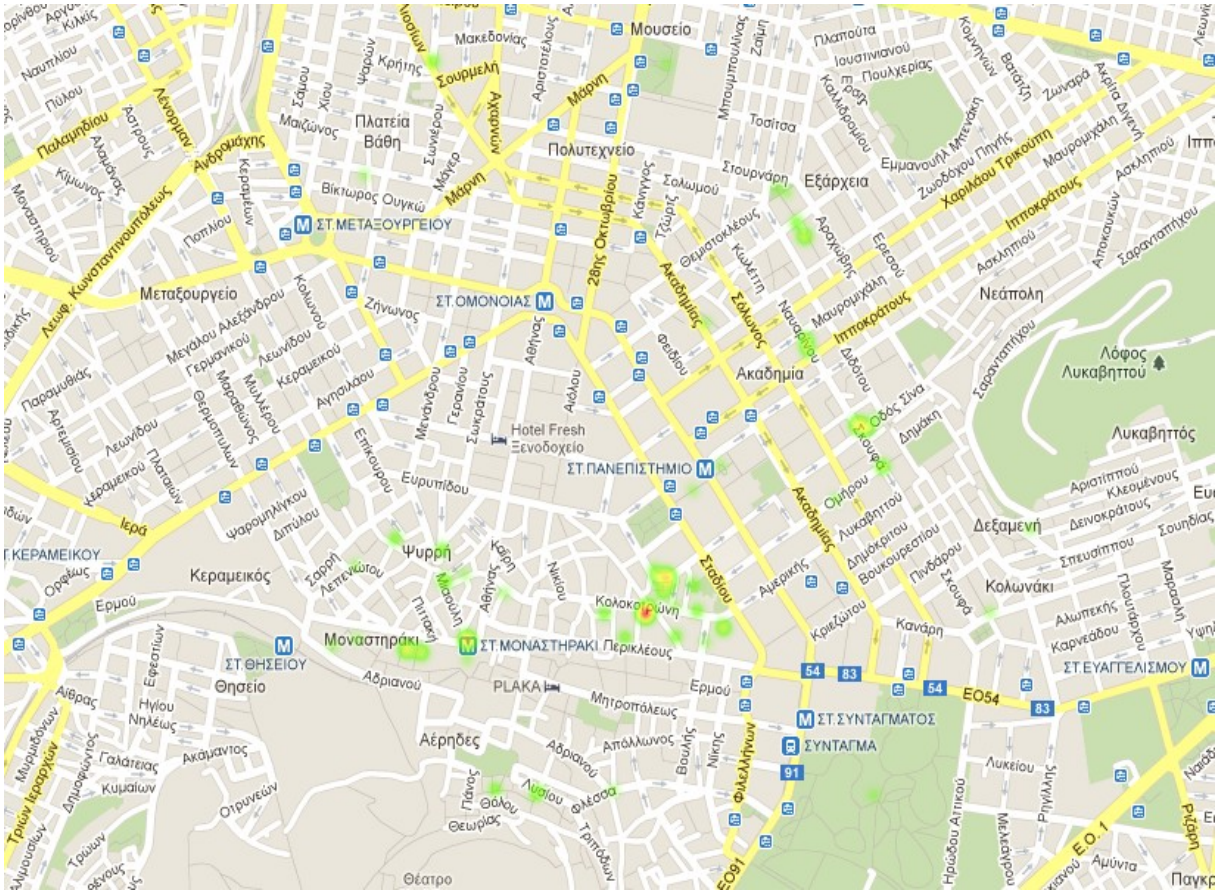


Εικόνα 40: Στιγμιότυπο 4

Η εφαρμογή δεν προσφέρει απλώς μια εικόνα που να παρουσιάζει ποιοτικά τον καταμερισμό του κόσμου στα διάφορα venues αλλά έναν πλήρη χάρτη κατά τα πρότυπα του Google Maps, με δυνατότητα ζουμ. Για να αναδείξουμε καλύτερα το γνώρισμα αυτό παραθέτουμε στιγμιότυπα του πρώτης περίπτωσης που παρουσιάσαμε παραπάνω σε δύο διαφορετικά επίπεδα ζουμ. Η ακρίβεια των θέσεων των venues βρίσκεται έτσι στο χέρι του χρήστη, που μπορεί να ζουμάρει αρκετά ώστε να βρει την ακριβή τοποθεσία κάποιου venue.



Εικόνα 42: Στιγμιότυπο του κέντρου της Αθήνας με μικρό επίπεδο ζουμ.



Εικόνα 43: Το στιγμιότυπο της Εικόνας 43 με μεγαλύτερο επίπεδο ζουμ.

Κεφάλαιο 5

Συμπεράσματα

5.1 Σύνοψη και συμπεράσματα

Καθώς η διπλωματική αυτή, που ξεκίνησε με στόχο την αποδοτική συλλογή, αποθήκευση και δεικτοδότηση δεδομένων από LBS σε κατανεμημένη βάση, φτάνει στο τέλος της ήρθε η ώρα εδώ να ανακεφαλαιώσουμε όσα είδαμε, να τα συνοψίσουμε και να βγάλουμε κάποια χρήσιμα συμπεράσματα. Κατ' αρχάς, ασχοληθήκαμε με τρεις location based services: το Facebook, το Foursquare και το Twitter. Ανεξάρτητα από το πώς μπήκε η καθεμία στο χώρο των χωροχρονικών δεδομένων, αυτή τη στιγμή όλες τους προσφέρουν στους χρήστες τους δυνατότητα προσθήκης της γεωγραφικής του θέσης στις αναρτήσεις που κάνουν. Χρησιμοποιώντας τα API των υπηρεσιών αυτών αναπτύξαμε έναν crawler, σχεδιασμένο έτσι ώστε να τρέχει συνεχώς και ανά μία ώρα να συλλέγει πληροφορίες από τις τρεις υπηρεσίες σχετικά με τη θέση των χρηστών τους. Οι πληροφορίες πάρθηκαν στα πλαίσια της Αθήνας και αφορούσαν το πλήθος του κόσμου, που σύμφωνα με τις LBS, βρισκόταν σε κάθε σημείο.

Αφού ο crawler έβρισκε αναλυτικά τους χρήστες που βρισκόταν σε κάθε σημείο, αποθήκευε την πληροφορία αυτή σε ένα σχήμα HBase, που είχαμε δημιουργήσει για τις ανάγκες της πλατφόρμας. Επιλέχτηκε μια κατανεμημένη βάση, αφού προσφέρει την επεκτασιμότητα που είναι απαραίτητη στις location based services και γενικότερα σε υπηρεσίες των οποίων ο όγκος δεδομένων συνεχώς διογκώνεται. Μπορεί στη μία βδομάδα που έτρεξε ο crawler να μη συλλέξαμε τεράστια ποσότητα δεδομένων – κάτι που οφείλεται και στον περιορισμό που θέτουν οι LBS στα ερωτήματα που μπορούν να απαντάν κάθε ώρα – ωστόσο οι ανάγκες για αποθήκευση θα αυξανόταν σίγουρα πολύ αν συνεχίζαμε το τρέξιμο της πλατφόρμας για μήνες ή συμπεριλαμβάναμε και άλλες πόλεις εκτός της Αθήνας. Σε κάθε περίπτωση, η ενασχόληση με την τεχνολογία των κατανεμημένων βάσεων στα πλαίσια των υπηρεσιών βασισμένων στη θέση και τα χωροχρονικά δεδομένα που τις συνοδεύουν δικαιολογείται απόλυτα.

Για να εξερευνήσουμε τα διάφορα είδη αποθήκευσης που μπορεί να χρειαζόταν για τη λειτουργία της πλατφόρμας χρησιμοποιήσαμε τρεις διαφορετικούς πίνακες και μέσω πειραμάτων καταλήξαμε αργότερα στον αποδοτικότερο συνδυασμό. Οι δύο από τους πίνακες αυτούς χρησιμοποιούσαν σαν κλειδί των εγγραφών τους τις γεωγραφικές συντεταγμένες των θέσεων. Η διαφορά έγκειται στο πώς μετέτρεπαν τις δυο διαστάσεις σε μία τιμή, αφού η HBase με το key-value μοντέλο της δεν επιτρέπει άμεσα τη δεικτοδότηση με βάση δύο μεταβλητές. Ο πρώτος πίνακας χρησιμοποίησε μια δεικτοδότηση που ονομάσαμε “τυπική”: τα κλειδιά αποτελούνται από το γεωγραφικό πλάτος του σημείου ακολουθούμενο από το γεωγραφικό μήκος. Είναι η μορφή που οι γεωγραφικές συντεταγμένες συνήθως συναντιούνται και ενώ είναι αποδοτική σε σχέση με το πλάτος, το μήκος ουσιαστικά δεν υπολογίζεται όταν ζητήσουμε κάποιο εύρος σημείων. Ο δεύτερος πίνακας χρησιμοποίησε την z-order curve, μια καμπύλη που αντιστοιχεί στα σημεία ενός δισδιάστατου (για την ακρίβεια στη γενικότερη περίπτωση n-διάστατου) επιπέδου μία z-τιμή. Έτσι τα σημεία ταξινομούνται

μέσα στον πίνακα αυτό ανάλογα με τη z-τιμή τους, εξασφαλίζοντας ότι σημεία που βρίσκονται κοντά στον πίνακα δεικτοδότησης, θα είναι γειτονικά και στο δισδιάστατο επίπεδο.

Ο τρίτος πίνακας που χρησιμοποιήσαμε δεν ασχολήθηκε με το πώς θα μπορούσαν να δεικτοδοτηθούν πιο αποδοτικά τα γεωγραφικά σημεία, αλλά χρησιμοποίησε σαν κλειδί των εγγραφών του τη χρονική στιγμή, το timestamp δηλαδή, κατά το οποίο πάρθηκε η μέτρηση. Η λογική πίσω από τον πίνακα αυτόν ήταν ότι σε περίπτωση που ζητηθούν δεδομένα για όλα τα σημεία που έχουν συλλεχθεί – κάτι που θα ήταν πολύ λογικό για μια εφαρμογή που θα έτρεχε πάνω στην πλατφόρμα και θα μπορούσε να θέλει στατιστικά για ολόκληρο το λεκανοπέδιο για κάποιες ώρες – η ανάκτηση της πληροφορίας από τους δύο παραπάνω πίνακες θα ήταν ιδιαίτερα χρονοβόρα. Για την ακρίβεια θα έπρεπε να προσπελαστούν όλες οι εγγραφές και για κάθε μια τους να εφαρμοστεί φίλτρο που να εξακριβώνει αν κάθε εγγραφή της ανήκει ή όχι στο ζητούμενο χρονικό διάστημα. Αντίθετα, αν η ταξινόμηση γίνεται με βάση το χρόνο θα είναι πιο γρήγορη μία σάρωση που ζητάει ένα χρονικό διάστημα, αφού θα γίνεται με βάση το “πρωτεύον” κλειδί.

Σε κάθε περίπτωση βέβαια, υπενθυμίζουμε ότι η κύρια πληροφορία που παίρναμε από τις location based services ήταν το here now, ο αριθμός δηλαδή των χρηστών που βρισκόταν σε κάθε θέση. Η πληροφορία αυτή αποθηκευόταν στα κελιά των πινάκων, ανεξάρτητα με το πώς ήταν δημιουργημένοι οι πίνακες και ζητούταν αργότερα στα διάφορα ερωτήματα που κάναμε.

Στο πειραματικό κομμάτι συγκρίναμε τις διάφορες τεχνικές αποθήκευσης για να επιλέξουμε την πιο αποδοτική. Για να το κάνουμε όμως αυτό, έπρεπε να εξασφαλίσουμε ότι όλες θα τρέξουν με τις καλύτερες δυνατές παραμέτρους. Έτσι, εξερευνήσαμε αρχικά το βέλτιστο αριθμό ερωτημάτων στις οποίες πρέπει να χωριστεί ένα range querie όταν γίνεται στον z-order curve δεικτοδοτημένο πίνακα - βλέποντας στην πράξη πως από μόνο του το σχήμα αυτό δεικτοδότησης δεν αρκεί, αλλά πρέπει να εφαρμοστεί μία για να πετύχουμε καλούς χρόνους και μικρά ποσοστά false-positive εγγραφών. Τα ερωτήματα που εκτελέσαμε αφορούσαν range queries για τετράγωνα 500, 1.000 και 2.000 μέτρων και γινόταν με ίδιο ποσοστό στο κέντρο και στα προάστια. Έπειτα βρήκαμε τον βέλτιστο αριθμό γραμμών που πρέπει να επιστρέφονται σε κάθε RPC από τη βάση ανάλογα με τον πίνακα στον οποίο γίνεται η ερώτηση.

Στο πιο ενδιαφέρον μέρος των μετρήσεων, συγκρίναμε τα τέσσερα διαφορετικά σχήματα: “τυπική” δεικτοδότηση με και χωρίς τη βοήθεια του πίνακα 'timestamps' και το ίδιο για τον πίνακα που δεικτοδοτούταν με τις z-τιμές των συντεταγμένων. Συμπεράναμε ότι η χρήση του 'timestamps' είναι ουσιώδης στα ερωτήματα που αφορούν ολόκληρη την Αθήνα, αφού μειώνει το χρόνο απόκρισης κατά μεγάλο ποσοστό, ανεξάρτητα από το είδος της δεικτοδότησης που χρησιμοποιείται (για παράδειγμα 93% μείωση για πίνακα με z-order curve δεικτοδότηση και ερωτήματα τετραγώνων πλευράς 500 μέτρων). Μάλιστα στην περίπτωση που τα λεγόμενα full scans πρέπει να εξυπηρετηθούν από τους δύο πίνακες με τη βασισμένη στις συντεταγμένες δεικτοδότηση, αυτό με την “τυπική” τα πηγαίνει καλύτερα, λόγω της επιπλεόν πολυπλοκότητας που εισάγουν στον άλλο οι z-τιμές των κλειδιών. Επίσης, είδαμε την ανωτερότητα του z-order curve σχήματος, τόσο σε μέσους χρόνους όσο και σε false-positive εγγραφές, για τα queries που αφορούσαν το κέντρο και τα προάστια, ανεξάρτητα από το πόσο μεγάλωνε η πλευρά των τετραγώνων που οριοθετούσαν τα ερωτήματα. Αυτό οφειλόταν, όπως εξηγήσαμε, στο μικρότερο αριθμό false-positive εγγραφών που επιστρεφόταν σε αυτή την περίπτωση και το μικρότερο φόρτο επεξεργασίας που απαιτούταν συνεπώς από το πρόγραμμα-πελάτη.

Έτσι, καταλήξαμε στο ότι για την αποθήκευση χωροχρονικής πληροφορίας με σκοπό την αποδοτική εξυπηρέτηση ερωτημάτων που αφορούν range queries είναι με τη χρήση της z-order curve, αλγορίθμου διάσπασης του αρχικού ερωτήματος σε κατάλληλο αριθμό υπο-

ερωτημάτων και χρήση ενός βοηθητικού πίνακα δεικτοδοτημένου με βάση το χρόνο απόκτησης της πληροφορίας για ερωτήματα που αφορούν όλο το εύρος των εγγραφών.

Στη συνέχεια ασχοληθήκαμε με τη δυνατότητα ομαδοποίησης ερωτημάτων, για τη μαζική απάντησή τους από εφαρμογή που βρίσκεται ανάμεσα στους χρήστες και την πλατφόρμα και πρέπει να ανταποκριθεί σε μεγάλο όγκο ερωτήσεων. Συμπεράναμε ότι μέχρι 100 ερωτήσεις, δεν υπάρχει λόγος χρήσης ενός τέτοιου σχήματος, αφού μεμονωμένα τα ερωτήματα απαντιούνται σε μικρότερο χρόνο. Για σετ ερωτημάτων όμως ανάμεσα σε 1.000 και 100.000 queries μια τέτοια τεχνική μπορεί να κατεβάσει εντυπωσιακά το μέσο χρόνο απόκρισης ερωτήματος, αφού ανακτά από τη βάση εγγραφές που τις χρειάζονται περισσότερες από μια φορές, χωρίς να τις ξαναζητήσει. Το trade-off που πρέπει να πληρώσουμε είναι οι απαιτήσεις σε μνήμη. Στο σετ του ενός εκατομμυρίου ερωτημάτων ο χρόνος ανά ερώτημα αρχίζει να αυξάνεται ξανά, ενώ για σετ των 10 εκατομμυρίων ερωτημάτων η μνήμη που δίνεται από την Java στη διεργασία δεν αρκεί.

Τέλος, σαν usecase της πλατφόρμας που δημιουργήσαμε και γεμίσαμε με δεδομένα με το τρέξιμο του crawler για μια βδομάδα, αναπτύξαμε εφαρμογή που δέχεται από το χρήστη ένα γεωγραφικό τετράγωνο στο οποίο θέλει να κάνει ερώτηση σε σχέση με τον κόσμο που υπάρχει για κάποιο χρονικό διάστημα και επιστρέφει το συνολικό here now και ένα χάρτη με μια ποιοτική απεικόνιση της πληροφορίας, που προσφέρει δυνατότητα ζουμ και απεικονίζει κάθε venue με μεγάλη λεπτομέρεια θέσης.

Κλείνουμε άρα τα συμπεράσματά μας έχοντας καταλήξει σε ένα αποδοτικό σχήμα αποθήκευσης και δεικτοδότησης χωροχρονικών δεδομένων με βάση τις συντεταγμένες των διάφορων θέσεων, που συνεργάζεται με ένα πίνακα που περιέχει την ίδια πληροφορία ταξινομημένη όμως σύμφωνα με το χρόνο απόκτησής της. Τα παραπάνω σχηματίζουν μια πλατφόρμα που περιέχει δεδομένα παρμένα από τις τρεις πιο δημοφιλείς στην Ελλάδα LBS και κάνει δυνατή την ανάκτησή της όταν κάτι τέτοιο ζητηθεί. Μια εφαρμογή που αναπτύχθηκε πάνω σε αυτή την πλατφόρμα, κάνει range queries για διάφορα τετράγωνα σε δεδομένο χρόνο και εμφανίζει στο χρήστη ένα χάρτη με την πυκνότητα του κόσμου, μας δείχνει ότι μια τέτοια βάση δεδομένων μπορεί να είναι χρήσιμη και εύκολα αξιοποιήσιμη από εφαρμογές που κινούνται στο χώρο τον χωροχρονικών δεδομένων.

5.2 Μελλοντικές επεκτάσεις

Η πλατφόρμα που περιγράψαμε στη διπλωματική κινείται σε δύο κύριους άξονες: τη συλλογή και την αποθήκευση δεδομένων. Όσον αφορά τη συλλογή, το crawling δηλαδή των υπηρεσιών βασισμένων στη θέση, μπορεί να γίνει εύκολη επέκταση και σε άλλες πόλεις εκτός της Αθήνας. Αυτό οφείλεται στον τρόπο που αρχικοποιείται ο crawler, μέσω μιας λίστας με τα κέντρα των περιοχών στα οποία θα ψάξει δηλαδή, και αρκεί μόνο μια επέκταση της ήδη υπάρχουσας λίστας για την προσθήκη νέων περιοχών. Ένα από τα ζητήματα στα οποία πρέπει να δοθεί προσοχή είναι ο μέγιστος αριθμός ερωτημάτων που μπορούν να γίνουν στο API κάθε υπηρεσίας, αν και συνήθως οι υπηρεσίες είναι πρόθυμες να δώσουν περισσότερες ερωτήσεις σε μια εφαρμογή, αρκεί να έχει υπάρξει συνεννόηση με την αρμόδια ομάδα. Επίσης, για τη συλλογή δεδομένων από την Αθήνα ο crawler έκανε ερωτήματα για περίπου 40 λεπτά, ενώ τα υπόλοιπα 20 λεπτά της ώρας, ήταν κενά και απλά περίμενε για το νέο γύρο ερωτημάτων. Σε περίπτωση που συμπεριληφθεί στην πλατφόρμα και άλλη μεγάλη πόλη, είναι πιθανό ακόμα και αν εξασφαλίσουμε επαρκή αριθμό ερωτημάτων να μην αρκεί ένα μηχάνημα για τη σειριακή εκπόνηση όλων στο διάστημα μίας μόνο ώρας. Μία πιθανή λύση του προβλήματος θα ήταν να τρέχει παράλληλα σε δύο μηχανήματα η συλλογή δεδομένων, χωρισμένη σε μία πόλη για το κάθε μηχάνημα. Έτσι θα αρκεί πρόσβαση και από

τις δύο μεριές στην Hbase και οι διαφορετικές γεωγραφικές περιοχές των σημείων που θα λαμβάνονται θα είναι αρκετή για να μην υπάρχουν προβλήματα απώλειας ή υπερκάλυψης πληροφορίας.

Στο σκέλος της αποθήκευσης των δεδομένων δείξαμε ήδη πως, όσον αφορά τα range queries, η δεικτοδότηση με τη z-order curve και ένας αλγόριθμος διάσπασης του αρχικού ερωτήματος σε ισοδύναμα υπο-ερωτήματα αποτελούν την καλύτερη λύση. Για την εφαρμογή των παραπάνω χρησιμοποιήσαμε το ήδη υπάρχον σύστημα αποθήκευσης και δεικτοδότησης της HBase χωρίς να μπούμε βαθύτερα στον τρόπο που αποθηκεύονται τα δεδομένα. Μία βαθύτερη κατανόηση του σχήματος που χρησιμοποιεί η HBase για να αποθηκεύει στη σκληρή μνήμη των μηχανημάτων του cluster τις διάφορες εγγραφές και η χρήση κάποιου πιο κατάλληλου σχήματος δεικτοδότησης (για παράδειγμα κάποιο τετρα-δέντρο, που να λαμβάνει υπόψιν τη γεωγραφική πληροφορία της εγγραφής) θα μπορούσε να οδηγήσει στην ανάκτηση λιγότερων εγγραφών από το σκληρό για ένα ερώτημα εύρους και κατά συνέπεια ακόμη μικρότερους χρόνους.

Αναφορές

[1] Foursquare. https://foursquare.com/
[2] Facebook. http://www.facebook.com/
[3] Twitter. https://twitter.com/
[4] Foursquare API. https://developer.foursquare.com/
[5] Facebook API. https://developers.facebook.com/
[6] Twitter API. https://dev.twitter.com/
[7] HBase. http://hbase.apache.org/
[8] Morton, G. M. (1966), A computer Oriented Geodetic Data Base; and a New Technique in File Sequencing, Technical Report, Ottawa, Canada: IBM Ltd.
[9] Google Maps API. https://developers.google.com/maps/
[10] http://blog.compete.com/2009/02/09/facebook-myspace-twitter-social-network/
[11] Geier, Thom; Jensen, Jeff; Jordan, Tina; Lyons, Margaret; Markovitz, Adam; et al.(December 11, 2009). "THE 100 Greatest Movies, TV Shows, Albums, Books, Characters, Scenes, Episodes, Songs, Dresses, Music Videos, and Trends that entertained us over the 10 Years". Entertainment Weekly (New York) ((1079/1080):74–84).
[12] http://www.quantcast.com/facebook.com Quantcast.com. April 29, 2011. Retrieved May 15, 2011.
[13] http://socialmediatoday.com/index.php?q=roywells1/158020/416-us-population-has-facebook-account
[14] http://edition.cnn.com/2011/TECH/social.media/06/13/facebook.dropping.america/index.html
[15] http://en.wikipedia.org/wiki/Facebook
[16] http://sproutsocial.com/insights/2012/01/facebook-places-vs-foursquare/
[17] http://www.insidefacebook.com/category/places/
[18] http://www.insidefacebook.com/2012/05/04/facebook-makes-another-mobile-acquisition-location-app-glancee/
[19] http://aboutfoursquare.com/foursquare-101/
[20] http://www.bitrebels.com/social/why-foursquare-is-worth-watching-i
[21] http://en.wikipedia.org/wiki/Foursquare
[22] http://www.nytimes.com/2012/06/07/technology/in-app-overhaul-foursquare-shifts-focus-to-recommendations.html?
[23] http://blog.compete.com/2009/02/09/facebook-myspace-twitter-social-network/
[24] http://en.wikipedia.org/wiki/Twitter
[25] http://mashable.com/2010/03/10/twitter-geolocation-tweets/
[26] http://betanews.com/2012/07/31/twitter-500-million-accounts-billions-of-tweets-and-less-than-one-percent-use-their-location/
[27] http://semiocast.com/publications/2012_07_30_Twitter_reaches_half_a_billion_accounts_140m_in_the_US
[28] J. L. Bentley. Multidimensional binary search trees used for associative searching
[29] Raphael Finkel and J.L. Bentley (1974). "Quad Trees: A Data Structure for Retrieval on Composite Keys"
[30] Guttman, A. (1984). "R-Trees: A Dynamic Index Structure for Spatial Searching"
[31] A. Silberschatz, H. Korth, S. Sudarshan - Database System Concepts, Sixth Edition
[32] Peano, G. (1890), "Sur une courbe, qui remplit toute une aire plane", Mathematische Annalen 36 (1): 157–160, doi:10.1007/BF01199438.
[33] Morton, G. M. (1966), A computer Oriented Geodetic Data Base; and a New Technique in File Sequencing, Technical Report, Ottawa, Canada: IBM Ltd.
[34] D. Hilbert: Über die stetige Abbildung einer Linie auf ein Flächenstück. Mathematische Annalen 38 (1891), 459–460.
[35] http://hadoop.apache.org/
[36] http://www.apache.org/

[37]
http://static.googleusercontent.com/external_content/untrusted_dlcp/research.google.com/el//archive/bigtable-osdi06.pdf
[38] http://hadoop.apache.org/docs/hdfs/current/hdfs_design.html
[39]
ttp://static.googleusercontent.com/external_content/untrusted_dlcp/research.google.com/el//archive/gfs-sosp2003.pdf
[40] http://static.usenix.org/event/osdi04/tech/full_papers/dean/dean.pdf
[41]
http://static.googleusercontent.com/external_content/untrusted_dlcp/research.google.com/el//archive/bigtable-osdi06.pdf
[42] Lars George - HBase The Definitive Guide
[43] http://www.cs.ucsb.edu/~sudipto/papers/md-hbase.pdf
[44] C. S. Jensen, D. Lin, and B. C. Ooi, "Query and update efficient b+-
tree based indexing of moving objects," in VLDB. VLDB Endowment,
2004, pp. 768–779.
[45] Y. Tao, K. Yi, C. Sheng, and P. Kalnis, "Quality and efficiency in high
dimensional nearest neighbor search," in SIGMOD, 2009, pp. 563–576.
[46] http://asg.unige.ch/publications/TR07/04LBSforTrafficMgt.pdf
[47] Theodore S. Stamoulakatos, Efstathios D. Sykas, "A network architecture to obtain traffic information for
Location Based Service applications", ConTEL 2003 7th International Conference on Telecommunications,
Zagreb, Croatia, June 11-13, 2003
[48] Antikainen, H., Rusanen, J., Vartiainen, S., Myllyaho, M., Karvonen, J., Oivo, M.,
Similä, J. & Laine, K., 2006: Location-based Services as a Tool for Developing
Tourism in Marginal Regions. Nordia Geographical Publications, 35: 2, pp. 39-50.
[49] http://www.raima.com/wp-content/uploads/COTS_embedded_database_solving_dynamic_pois_2012.pdf