Θέματα μεταπτυχιακών & διπλωματικών εργασιών για το ακ. έτος 2025- 2026

*Π. Βασιλειάδης – **2025-02-03***

Τα θέματα διπλωματικών και μεταπτυχιακών εργασιών συνήθως επεκτείνουν κάποια από τα υπάρχοντα εργαλεία που αναπτύσσουμε στην ομάδα μου – λίγο πιο σπάνια, ξεκινούν νέα εργαλεία.

https://github.com/DAINTINESS-Group

https://www.youtube.com/playlist?list=PL3G-N7ZzyiDfpsMLCQcm_L9KEVLDkC454

Το παρόν κείμενο περιγράφει τα προτεινόμενα θέματα και, για να διευκολύνει την κατανόηση, δίνει και μια γενική εικόνα για τα εργαλεία που αναπτύσσουμε.

Αναγκαστικά μια εργασία πρέπει να ολοκληρωθεί αυστηρά εντός ενός έτους από την ανάληψή της. Απαιτούμενα προσόντα είναι η πολύ καλή γνώση Java & σχεδίασης ΟΟ λογισμικού. Στις περισσότερες διπλωματικές χρειάζεται να μπορείτε να συνδυάσετε frameworks – e.g., Apache Spark, και ενίοτε να πρέπει να δουλέψετε με το συνδυασμό κώδικα και βάσεων δεδομένων.

Γενικά προχωράμε ως ομάδα με agile working methods, ήτοι **weekly runs & group meetings**. Κείμενο και κώδικας γράφονται εναλλάξ. Παντού δουλεύουμε github && εκτός ειδικών περιπτώσεων, Eclipse.

Επειδή πλέον οι καιροί είναι πονηροί, **δεν θα δεχτώ να εξετάσω διπλωματικές από φοιτητές, που δεν βλέπω στο εξάμηνο και μου παρουσιάζουν ξαφνικά ένα κείμενο και ένα κώδικα προς εξέταση στο τέλος**. Θα πρέπει να ασχολείστε συστηματικά με την εργασία σας στη διάρκεια του εξαμήνου.

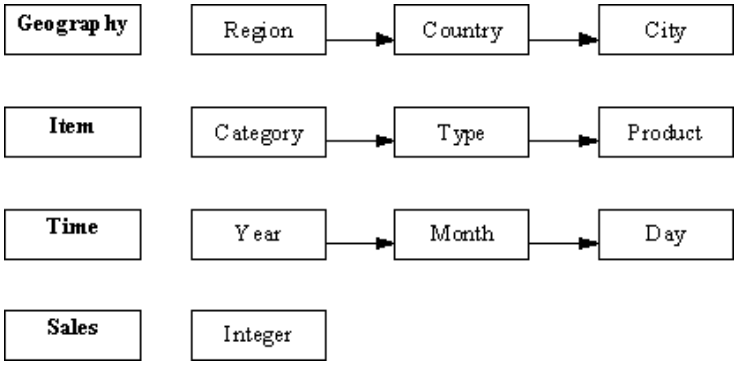
Contents

1 Business Intelligence Analytics

Οι εφαρμογές On-Line Analytical Processing (OLAP) αντιμετωπίζουν τα δεδομένα σαν *κύβους* (*data cubes*), οι οποίοι αποτελούνται από διαστάσεις και μετρήσιμες τιμές. Οι κύβοι αποτελούνται από *διαστάσεις* (χρόνος, γεωγραφική περιοχή κ.λπ.) και *μετρήσιμα μεγέθη* (π.χ. σύνολο πωλήσεων). Οι διαστάσεις οργανώνονται σε λογικές *ιεραρχίες*. Για παράδειγμα, ο χρόνος μπορεί να οργανωθεί στην ιεραρχία χρόνος, μήνας, μέρα. Για παράδειγμα, οι πωλήσεις ενός οργανισμού μπορούν να μοντελοποιηθούν σαν ένας κύβος της μορφής

Day (Year-month-day)	Product	City	Sales
1997-01-01	“Report to El Greco”	Rhodes	15
1997-01-01	“Ace of Spades”	Paris	8
1997-01-01	“Report to El Greco”	Athens	11

με διαστάσεις οργανωμένες σε *ιεραρχίες*, όπως:



Ο χρήστης μπορεί να κάνει ερωτήσεις όπως μετασχηματισμός του κύβου σε διαφορετικά επίπεδα ιεραρχίας, επιλογή κάποιων υποσυνόλων του κύβου κ.λ.π. Σε οποιοδήποτε textbook βάσεων δεδομένων, θα βρείτε ένα κεφάλαιο για αποθήκες δεδομένων και OLAP για τις βασικές έννοιες.

Οι παρακάτω εργασίες εντάσσονται στο μεγάλο στόχο να αναθεωρηθεί εκ βάθρων το μοντέλο κύβων και η απάντηση κάθε ερώτησης να επαυξηθεί με αποτελέσματα μηχανικής μάθησης, εκτός από τα δεδομένα αυτά καθαυτά.

Ο χρήστης υποβάλλει ένα OLAP ερώτημα (πρακτικά ένα ερώτημα συνάθροισης που περιλαμβάνει group-by και where clause). Το ερώτημα εκτελείται σε μία back-end OLAP engine, η οποία είναι ήδη σε μια πρώτη εκδοχή. Τα αποτελέσματα πρέπει να παρουσιαστούν στον χρήστη σε μια εκδοχή “ευέλικτου Excel” που επιτρέπει την οπτικοποίηση των αποτελεσμάτων και τη διαδραστική τους επερώτηση.

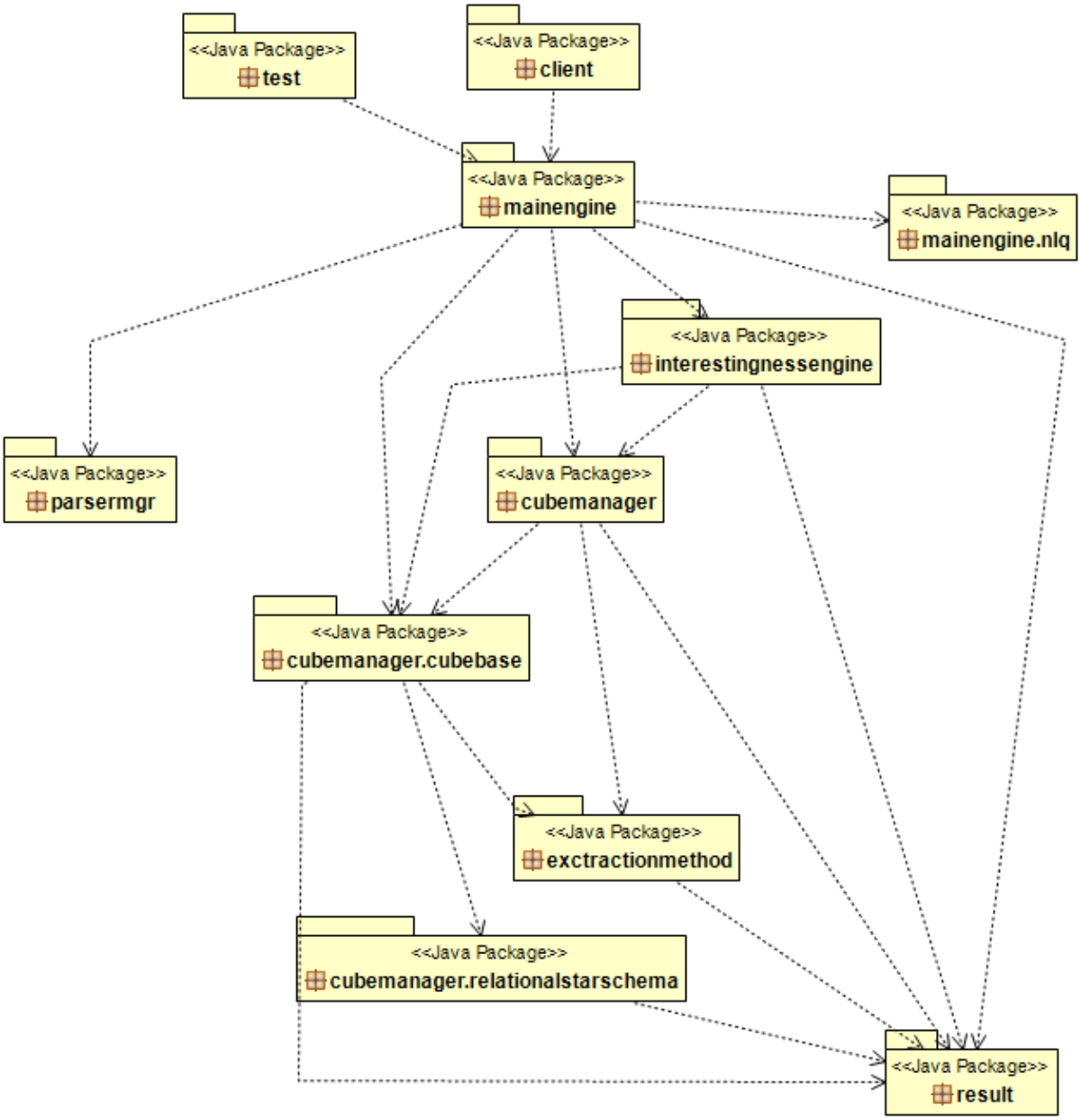
1.1 Υπόβαθρο για τις εργασίες

Στο Παν. Ιωαννίνων, έχουμε ήδη κατασκευάσει ένα σύστημα επερώτησης data cubes. Το σχετικό project έχει το δικό του repository στο Github:

https://github.com/pvassil/DelianCubeEngine

Στο σύστημα Delian Cubes, εξετάζουμε την απάντηση σε ερωτήματα μέσω *intentional queries & result mining*. Ο χρήστης υποβάλλει ένα OLAP ερώτημα (πρακτικά ένα ερώτημα συνάθροισης που

περιλαμβάνει group-by και where clause). Το σύστημα παράγει ως έξοδο (α) τα αποτελέσματα της ερώτησης, (β) μια λίστα από μοντέλα που είναι το αποτέλεσμα του τρεξίματος αλγορίθμων εξόρυξης γνώσης και στατιστικών (correlations, clusters, …) και (γ) ένα mapping μεταξύ των (α) και (β).



Το διάγραμμα πακέτων του Delian Cubes.

Η βασική κλάση που ξεκινά την απάντηση ερωτήσεων είναι η κλάση SessionQueryProcessingEngine στο πακέτο mainengine.

Αν θέλετε να καταλάβετε πώς δουλεύει το σύστημα, προσπαθήστε να καταλάβετε πώς απαντά ερωτήσεις (α) με models, και (β) με metadata, πώς δηλαδή μια εισαγόμενη συμβολοσειρά μετατρέπεται σε CubeQuery, και πώς αυτό εκτελείται και συμπληρώνεται με αποτελέσματα. (ναι, είναι ένα μονοπάτι all the way down to the result package).

Η δομή ενός ερωτήματος κύβων που υποστηρίζεται από το Delian είναι η παρακάτω:

CubeName: [το όνομα του κύβου που θα υποβάλλουμε το ερώτημα]
Name: [το ψευδώνυμο του ερωτήματος]
AggrFunc: [η συναθροιστική συνάρτηση που εφαρμόζεται στη μετρική του ερωτήματος]
Measure: [η μετρική που επιστρέφει το ερώτημα]
Gamma Expressions: [συνθήκες ομαδοποίησης του ερωτήματος]
Sigma Expressions: [συνθήκες φιλτραρίσματος του ερωτήματος]

Η παραπάνω έκφραση μετατρέπεται από το Delian στη παρακάτω ισοδύναμη SQL έκφραση, η οποία εκτελείται είτε από τη MySQL είτε από το Apache Spark.

SELECT [Gamma Expressions], [AggrFunc]([Measure])
FROM [Dimension & Fact Tables involved in WHERE & GROUP BY clause]
WHERE [JOIN Expressions] AND [Sigma Expressions]
GROUP BY [Gamma Expressions]
ORDER BY [Measure];

Στο Παν. Ιωαννίνων, έχουμε μια πρόταση για το πώς θα επερωτώνται οι κύβοι στο μέλλον. Δείτε στο

http://www.cs.uoi.gr/~pvassil/projects/olap_III/index.html

- το σχετικό άρθρο στο DOLAP 2018 [http://ceur-ws.org/Vol-2062/paper07.pdf]

- το πιο αναλυτικό άρθρο στο Information Systems 2019 [https://www.cs.uoi.gr/~pvassil/publications/2019_IS_IntentionalModel/IS2019_VassiliadisMarcelRizzi_CR_PreprintUoI.pdf]

Στόχος των παραπάνω εργασιών είναι να προστεθούν στο Delian Cubes οι υλοποιήσεις κάποιων νέων λογικών τελεστών πρόθεσης (*intentional operators*). Με τον όρο intentional operator, εννοούμε έναν αυστηρά ορισμένο τελεστή, με την δική του σύνταξη και σημασιολογία, ο οποίος θα δίνει στον χρήστη την δυνατότητα να εκφράζει μέσω ενός ερωτήματος ποια είναι η πρόθεση του για τα δεδομένα.

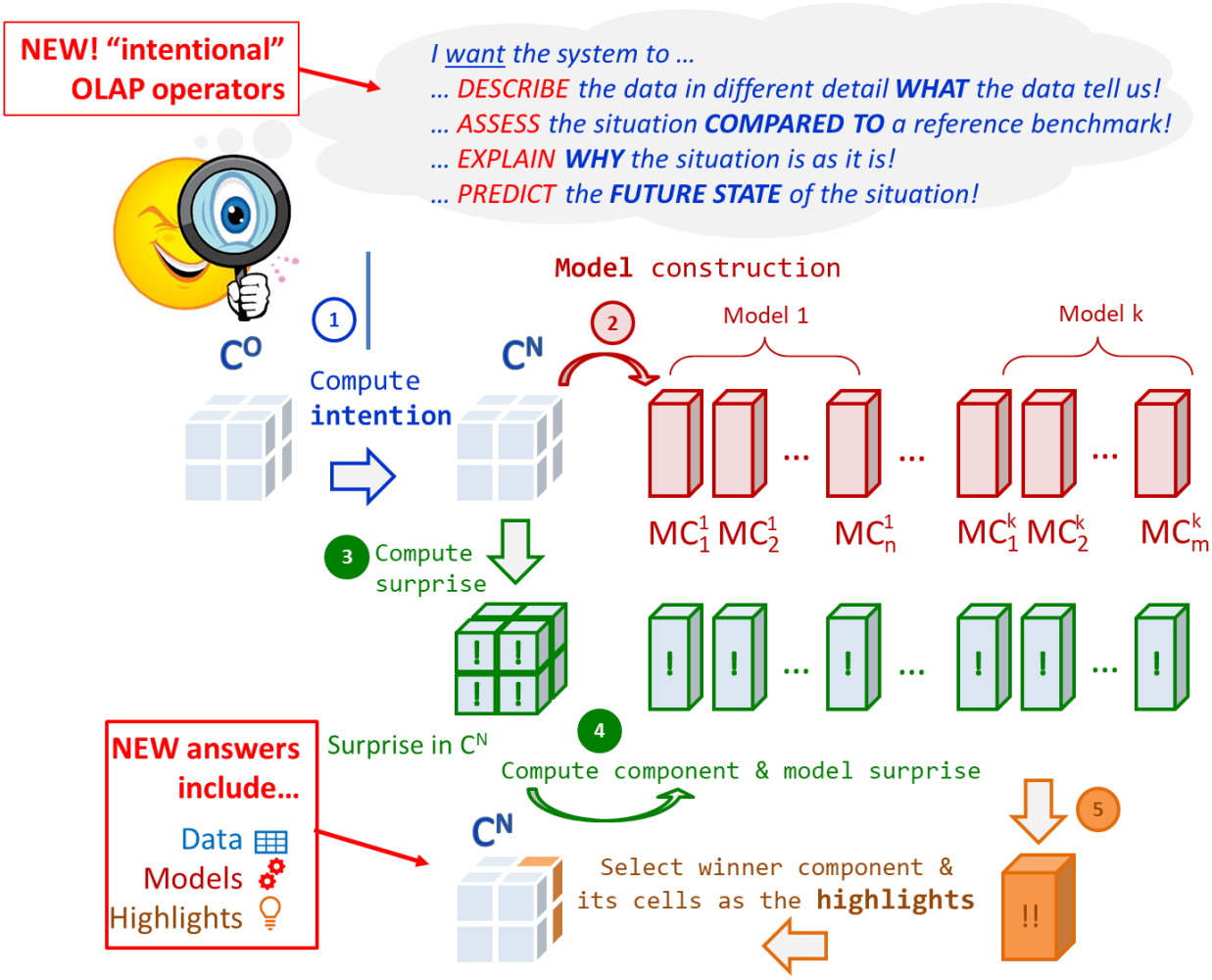
Για παράδειγμα, έστω ότι ο χρήστης θέλει να αναλύσει ένα φαινόμενο όπως η πτώση των πωλήσεων των γαλακτοκομικών τον Μάιο. Το παραπάνω κρύβει την πρόθεση της ανάλυσης. Η μέθοδος που ο χρήστης θα ακολουθούσε χωρίς την ύπαρξη ενός operator θα ήταν η υποβολή μιας σειράς ερωτημάτων τα αποτελέσματα των οποίων θα εξηγούσαν το φαινόμενο. Μια πιθανή μέθοδος θα ήταν:

- Ο έλεγχος των καθημερινών πωλήσεων των γαλακτοκομικών για τον Μάιο
- Η έλεγχος των καθημερινών πωλήσεων των γαλακτοκομικών για προηγούμενους μήνες που οι πωλήσεις ήταν «φυσιολογικές»
- Η σύγκριση των καθημερινών πωλήσεων για τον Μάιο με των προηγούμενων μηνών

Όπως γίνεται εύκολα αντιληπτό, η παραπάνω μέθοδος συνεπάγεται μια σειρά από ερωτήματα καθώς και περαιτέρω ανάλυση των αποτελεσμάτων για την εξαγωγή συμπερασμάτων. Η δημιουργία ενός operator θα αυτοματοποιούσε και θα επιτάχυνε την διαδικασία.
Παρακάτω περιγράφονται οι intentional operators που επιδιώκουμε να υλοποιηθούν και να ενταχθούν στο σύστημα Delian Cubes. Κάθε operator θα αντιστοιχεί και σε μία διπλωματική εργασία.

Η βασική ιδέα πίσω από το intentional model είναι η εξής:

- Ρίχνω μια intentional ερώτηση (predict, assess, …) η οποία κρύβει ουσιαστικά ένα group by query (μαζί με τις σχετικές επιλογές -φίλτρα, και τα αναγκαία joins)
- Το ερώτημα απαντιέται με δεδομένα, αλλά σε αυτά τα δεδομένα βάζω να τρέξουν κάποια models (για παράδειγμα: ranking of result cells, outlier detection, K-means clustering of result cells – στο πακέτο results υπάρχουν οι σχετικές υλοποιήσεις).
- Κάθε μοντέλο, έχει ένα σύνολο από vectors, όπου για κάθε κελί βάζει μια τιμή. Π.χ., στο clustering έχει από ένα vector για κάθε cluster και κάθε κελί του αποτελέσματος, αναλόγως αν είναι στο εν λόγω cluster έχει 1, αλλιώς 0 – ή το outlier detection έχει 2 components, true if the cell is indeed an outlier και false if not.
- Το άρθρο (αλλά όχι το σύστημα) προτείνει ένα μηχανισμό μέτρησης της έκπληξης και της επιλογής των πιο ενδιαφερόντων από τα highlights.



Σε όσους τελεστές ακολουθούν, δίνεται μια «προσεγγιστική» γλώσσα υποβολής ερωτημάτων σαν SQL. Αυτό που μας νοιάζει όμως, είναι να υλοποιηθεί η σχετική μέθοδος στο API του Delian με τις σωστές παραμέτρους – το συντακτικό είναι για να καταλαβαίνει κανείς χοντρικά τι ζητά η κάθε ερώτηση.

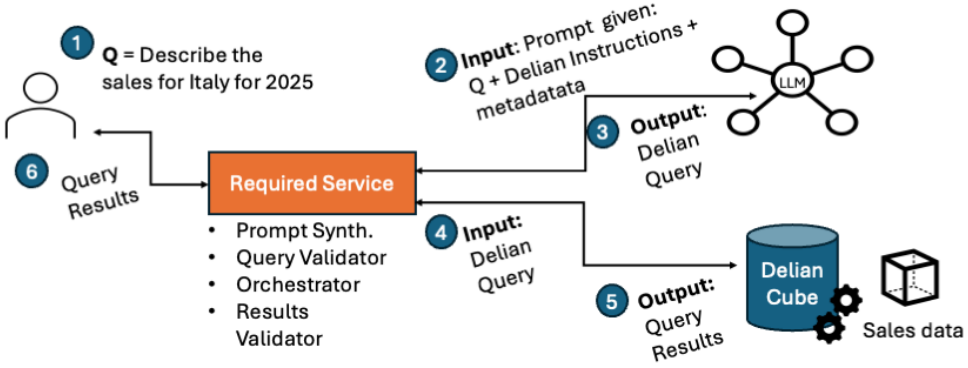
1.2 Υπηρεσία LLM για υποβολή ερωτημάτων σε κύβους Delian (με αξιοποίηση GPU)

Υπό την εποπτεία: ΒΑΣΙΛΗΣ ΣΤΑΜΑΤΟΠΟΥΛΟΣ <v.stamatopoulos@uoi.gr>

Σκοπός της διπλωματικής είναι ο σχεδιασμός και η υλοποίηση ενός εργαλείου, το οποίο θα δέχεται ερωτήματα σε Φυσική Γλώσσα και θα τα μετατρέπει, με τη βοήθεια ενός LLM που θα έχει γίνει deploy ως ξεχωριστό inference endpoint, σε ένα εκτελέσιμο ερώτημα (query) προς το DelianCubeEngine (SessionQueryProcessingEngine / API), ώστε το παραγόμενο Delian query να εκτελείται σωστά και να επιστρέφονται τα αποτελέσματα στον χρήστη.

Η λειτουργική ροή φαίνεται στο παρακάτω διάγραμμα: Ο χρήστης υποβάλλει ένα ερώτημα σε Φυσική Γλώσσα (1) (π.χ., Περίγραψε μου τις πωλήσεις για την Ιταλία το 2025), το service (2) συνθέτει ένα κατάλληλο prompt που περιλαμβάνει αφενός το ίδιο το αίτημα και αφετέρου την «περιγραφή χρήσης» του Delian (διαθέσιμες API/SQP κλήσεις και βασικά μεταδεδομένα για κύβους/διαστάσεις/measures), το οποίο αποστέλλεται στο LLM. Το LLM παράγει ως έξοδο ένα Delian query (3), το εργαλείο ελέγχει την ορθότητα/συμβατότητα της εξόδου και στη συνέχεια εκτελεί το query μέσω του Delian (4), επιστρέφοντας τα τελικά αποτελέσματα στον χρήστη (5, 6).

Το εργαλείο θα υλοποιεί: (i) prompt synthesizing (NL αίτημα + Delian API/μεταδεδομένα), (ii) orchestration μεταξύ User–LLM–Delian, (iii) query validation πριν την εκτέλεση, και (iv) results validation μετά την εκτέλεση.



1.3 Selectivity Estimation for Delian

Υπό την εποπτεία: ΜΑΡΙΟΣ ΙΑΚΩΒΙΔΗΣ <m.iakovidis@uoi.gr>

Στο Delian έχουμε ερωτήματα που περιλαμβάνουν φίλτρα επιλογής. Για παράδειγμα μπορεί να θέλουμε το ερώτημα

```
Cube: SALES

Aggregate: sum(quantity)

FOR product.type = 'Bananas', geo.country ='Greece' BY product
```

Εμείς τώρα πρέπει να ενισχύσουμε το βελτιστοποιητή ερωτήσεων του Delian με την πληροφορία: τι ποσοστό του πίνακα SALES πωλήσεις μπανάνας στην Ελλάδα? Αυτό σπάει σε 2 απλές, ανεξάρτητες γι' αρχή εκτιμήσεις: (α) τι ποσοστό του πίνακα αφορά μπανάνες, και (β) τι ποσοστό αφορά την Ελλάδα.

Θα εξετάσουμε δύο εναλλακτικούς τρόπους, όπου για κάθε τιμή που εμφανίζεται σε κάθε κάθε πεδίο, κάθε διάστασης, θα πάμε να μετρήσουμε τον απόλυτο αριθμό, και κατά συνέπεια, το ποσοστό του πίνακα που την αφορά. Οι δύο αυτοί τρόποι είναι

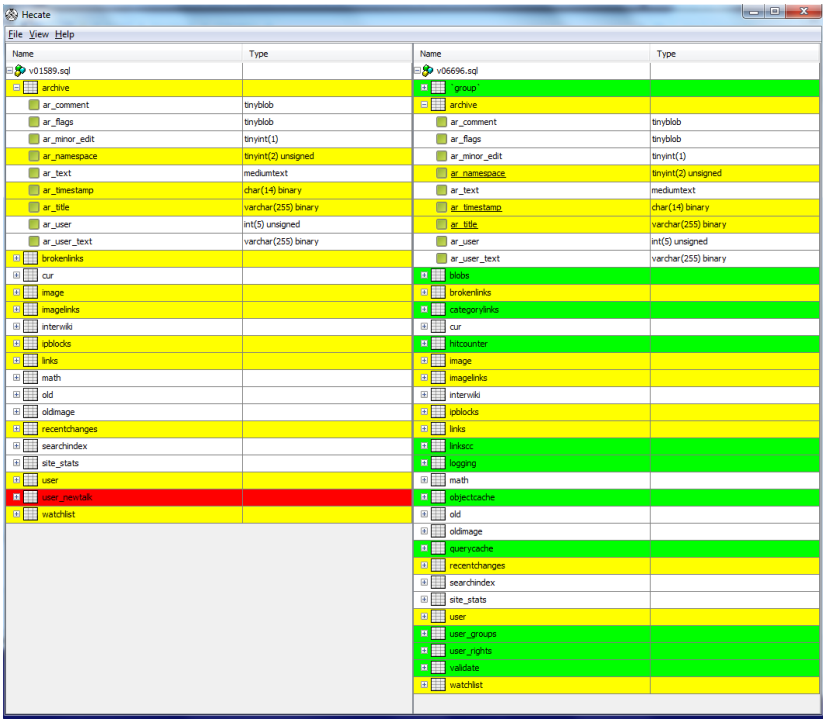
1. Full table scan: θα περάσουμε μία μία όλες τις εγγραφές του πίνακα και θα ενημερώσουμε τους σωστούς counters και τα αντίστοιχα ποσοστά.
2. Sampling: θα πειραματιστούμε με διαφορετικά sampling sizes για να δούμε τι sample πρέπει να πάρουμε για να έχουμε μια ανεκτή ακρίβεια. Για παράδειγμα https://en.wikipedia.org/wiki/Reservoir_sampling

Όλο αυτό είναι ένα trade-off ακρίβειας και καθυστέρησης. Η διπλωματική περιλαμβάνει εκτενέστατο πειραματισμό. Επίσης, στην αρχή, και αναζήτηση βιβλιογραφίας.

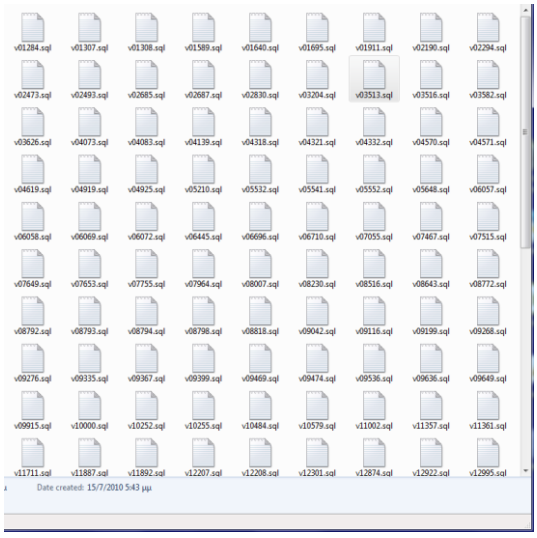
2 Schema Evolution

Μια βάση δεδομένων, από τη στιγμή που θα δημιουργηθεί, αλλάζει εσωτερική δομή με το πέρασμα του χρόνου: νέοι πίνακες δημιουργούνται, παλαιοί καταστρέφονται, πεδία διαγράφονται, μετονομάζονται κλπ. Η διαδικασία αυτή ονομάζεται «εξέλιξη του σχήματος της βάσης δεδομένων» (schema evolution).

Το εργαλείο **Hecate** [https://github.com/DAINTINESS-Group/Hecate] μπορεί να συγκρίνει δύο σχήματα και να βρει τις διαφορές τους (κίτρινο: updated attributes, red: deletions, green: insertions).

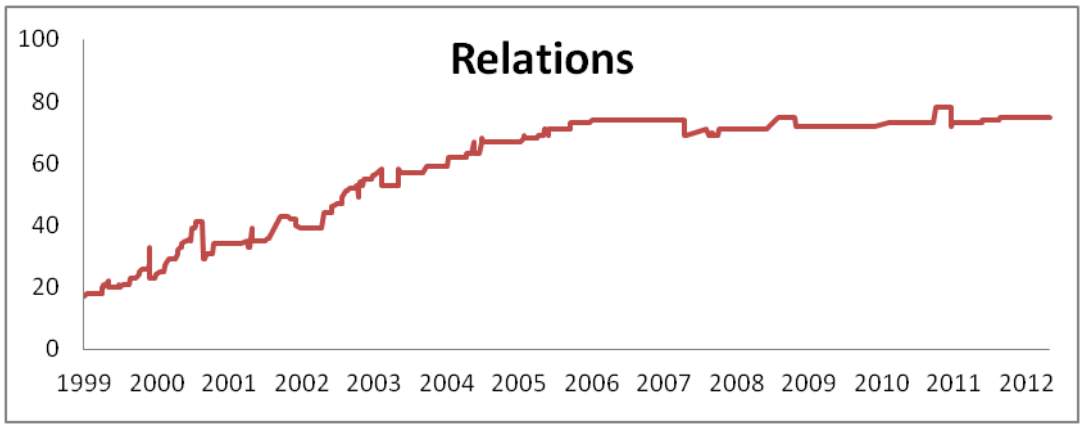


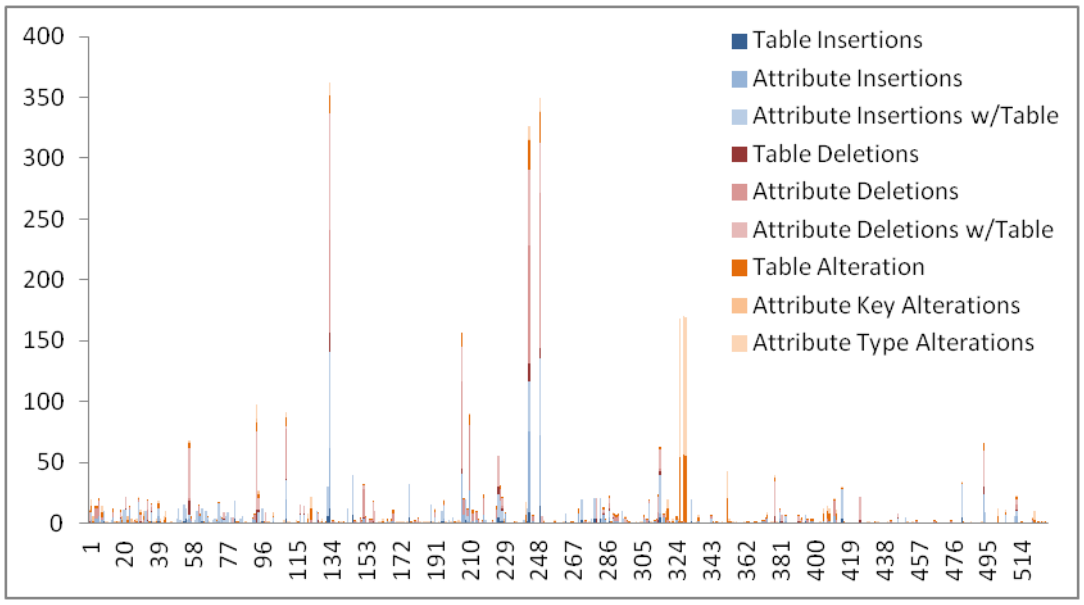
Επιπλέον, υπάρχουν αρκετές συλλογές από εκδόσεις του σχήματος της ιδίας βάσης (παρακάτω ένα screenshot από τη βάση της Wikimedia).



Η Εκάτη μπορεί να ταξινομήσει τις επί μέρους εκδοχές του σχήματος και να τις συγκρίνει διαδοχικά.

Έχουμε ήδη χρησιμοποιήσει την Εκάτη για να επεξεργαστούμε την εξέλιξη σχήματος διαφόρων βάσεων δεδομένων ανοιχτού λογισμικού.

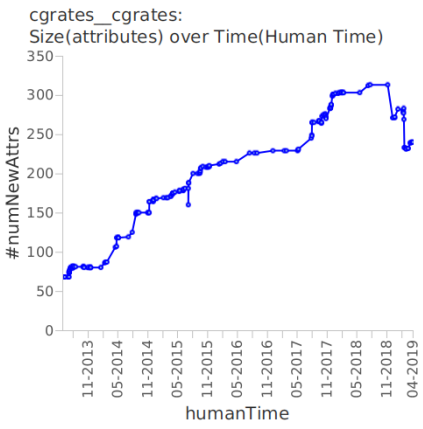


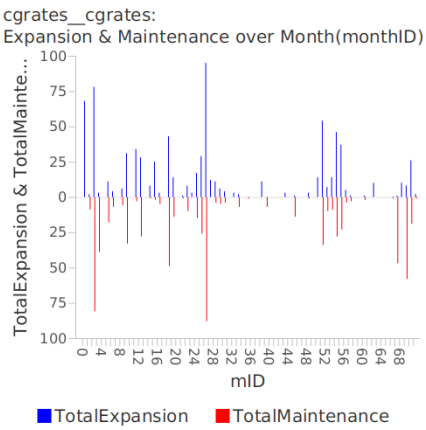


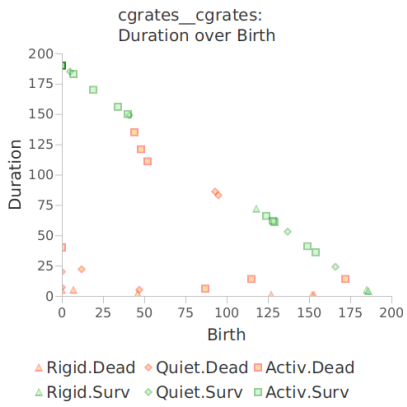
Στο παραπάνω σχήμα βλέπετε (α) το πώς εξελίχθηκε το μέγεθος του σχήματος της βάσης στο χρόνο και (β) τον παλμό των αλλαγών (το πώς διαρθρώθηκαν οι αλλαγές σε κάθε monitored version) για τη βάση Ensembl.

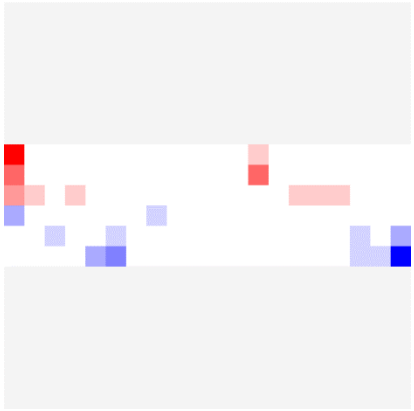
Το εργαλείο Heraclitus Fire [https://github.com/pvassil/HeraclitusFire] χρησιμοποιείται για να συμπληρώσει τα αρχικώς εξαχθέντα αποτελέσματα της Εκάτης με επιπλέον στατιστικά -- ενδεικτικά:

- Ανάλυση των αλλαγών σαν (α) επέκταση (προσθέσεις πινάκων και πεδίων) και (β) συντήρηση (διαγραφές πινάκων και πεδίων, αλλαγές τύπων, κλπ) και περαιτέρω ανάλυση σε άλλες υποκατηγορίες και μετρικές
- Συγκεντρωτικά στατιστικά για τις αλλαγές ανά commit και αλλαγές μήνα
- Έλεγχο κάποιων πρωτόλειων προτύπων
- Γραφικές παραστάσεις









2.1 Μελέτη της Εξέλιξης Σχημάτων Βάσεων Δεδομένων

Έχουμε ένα σύνολο από ιστορίες του πώς εξελίχθηκαν τα σχήματα από ένα μεγάλο αριθμό βάσεων δεδομένων. Δείτε τα άρθρα στο ICDE21, EDBT23, EDBT25 όπως θα τα βρείτε στο:

https://www.cs.uoi.gr/~pvassil/projects/schemaBiographies/publications.html

Τις ιστορίες αυτές τις έχουμε ήδη κωδικοποιήσει. Π.χ., για το project *CityGrid__twonicorn*, η εξέλιξη του σχήματός του (μετρώντας το άθροισμα των αλλαγών ανά μήνα) ήταν

-(6)b(80)c(39)_(1)c(11)c(11)_(1)c(9)_(20)

που μεταφράζεται: 6 μήνες από την αρχή του project δεν υπήρχε σχήμα, μετά γεννήθηκε ένα σχήμα που συνολικά είχε 80 πεδία στους πίνακές του, τον επόμενο μήνα γίναν 39 αλλαγές, μετά για 1 μήνα δεν έγινε τπτ, μετά για δύο συνεχόμενους μήνες είχαμε από 11 αλλαγές σε πεδία, μετά για 1 μήνα δεν έγινε τπτ, τον επόμενο μήνα είχε αλλαγή σε 9 πεδία, και μετά για 20 μήνες δεν έγινε τπτ.

Με βάση αυτές τις ιστορίες, θέλουμε

(α) να γκρουπάρουμε (cluster) τα σχήματα αυτά σε ομοειδείς ομάδες που ζουν παρόμοιες ζωές, και για κάθε ομάδα να βρούμε μια χαρακτηριστική περιγραφή και τα χαρακτηριστικά της στοιχεία

(β) να φτιάξουμε πιθανότικα΄μοντέλα για να προβλέψουμε την εξέλιξη του σχήματος

(γ) να βρούμε πιθανοτικά sequential patterns στις ιστορίες των σχημάτων

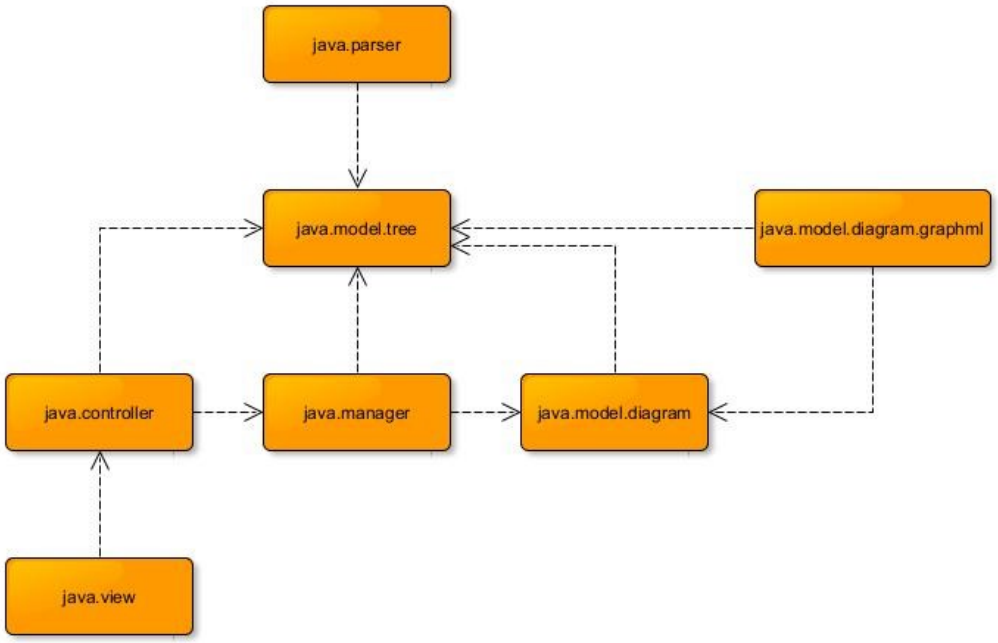
Όλοι οι αλγόριθμοι θα γίνουν σε Java.

3 UML diagramming

Υπό την εποπτεία: ΒΑΣΙΛΕΙΟΣ ΖΑΦΕΙΡΗΣ <bzafiris@uoi.gr>

Το ObjectAid έπαψε να υποστηρίζεται. Χρειαζόμαστε ένα εργαλείο που να κάνει reverse engineer ένα Java project και να μπορούμε να βγάλουμε UML Diagrams. Το Object-Oriented Architecture Diagrammer είναι ένα δικό μας ObjectAid για UML διαγράμματα

https://github.com/DAINTINESS-Group/ObjectOrientedArchitectureDiagrammer

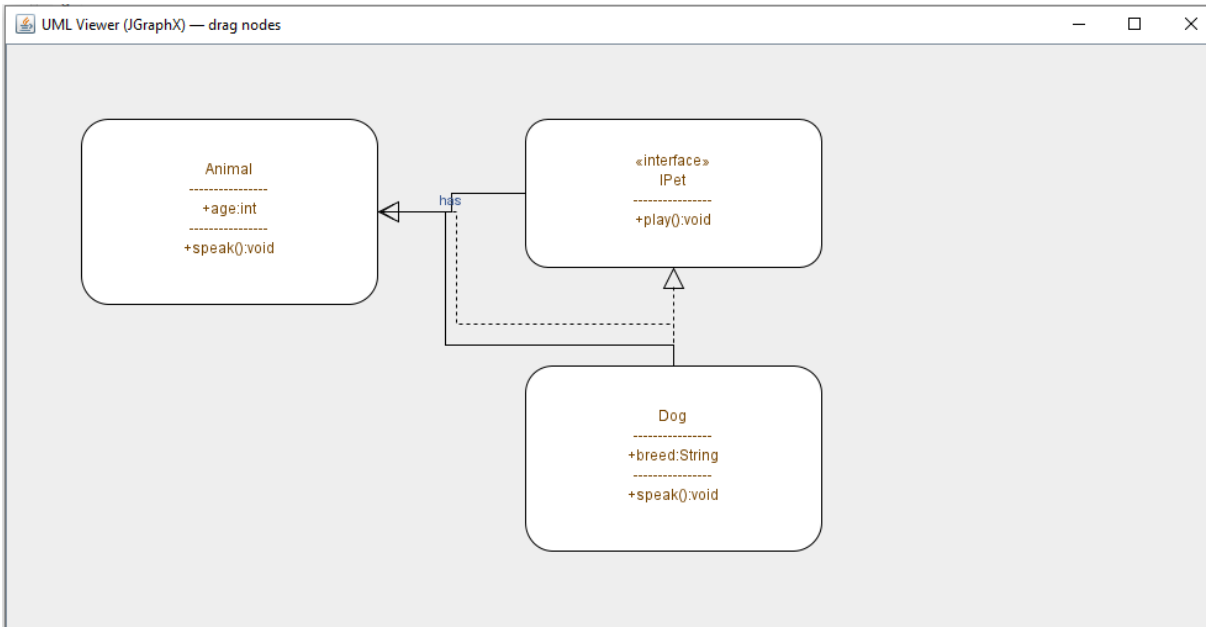


Ένα βασικό χαρακτηριστικό είναι ότι καταφέραμε να αξιοποιήσουμε parsers που επεξεργάζονται ένα java project, βγάζουν ένα AST δέντρο και από κει προκύπτει το γράφημα πακέτων, κλάσεων, και λοιπών δομικών συστατικών του source code.

Ζητούμενο: να φτιάξουμε ένα πιο lightweight εργαλείο, που να χρησιμοποιήσει αυτούσιο του parsing & domain model του OOAD και να επιτρέπει την αλληλεπίδραση με το χρήστη με (α) απλότητα, και (β) υψηλή αισθητική, σαν το ObjectAid.

Λύση 1: JGraphX μια απλή βιβλιοθήκη για κατασκευή boxes & lines

```
<dependency>
  <groupId>com.github.vlsi.mxgraph</groupId>
  <artifactId>jgraphx</artifactId>
  <version>4.2.2</version>
</dependency>
```



A simple diagram with JgraphX (100 lines of java code)

Λύση 2: plain JavaFX (the same code takes 180 lines)

Η έμφαση θα δοθεί στην ορθότητα του εργαλείου, την ευχρηστία και την (πολύ καλύτερη απ' ό,τι βλέπετε εδώ) αισθητική