ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΔΙΟΙΚΗΣΗΣ ΕΠΙΧΕΙΡΗΣΕΩΝ

ΤΜΗΜΑ ΔΙΟΙΚΗΤΙΚΗΣ ΕΠΙΣΤΗΜΗΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΠΑΥΛΟΥ ΚΑΛΟΓΕΡΑ

Πολλαπλές Βάσεις Δεδομένων στην εποχή των Big Data

ΠΡΑΚΤΙΚΗ ΆΣΚΗΣΗ ΣΤΟΝ ΟΡΓΑΝΙΣΜΟ

Nestlé Ελλάς

Επιβλέπων : Δαμιανός Χατζηαντωνίου – Αναπληρωτής Καθηγητής

Υποβληθείσα ως μέρος των απαιτήσεων για την απόκτηση
Μεταπτυχιακού Διπλώματος (MSc) στη Διοικητική Επιστήμη και Τεχνολογία

Αθήνα, Ιανουάριος 2022

Η σελίδα αυτή είναι σκόπιμα λευκή.

Βεβαίωση εκπόνησης Διπλωματικής εργασίας

«Δηλώνω υπεύθυνα ότι η συγκεκριμένη μεταπτυχιακή εργασία για τη λήψη του μεταπτυχιακού τίτλου σπουδών του ΠΜΣ στη Διοικητική Επιστήμη και Τεχνολογία του Τμήματος Διοικητικής Επιστήμης και Τεχνολογίας του Οικονομικού Πανεπιστημίου Αθηνών έχει συγγραφεί από εμένα προσωπικά και δεν έχει υποβληθεί ούτε έχει εγκριθεί στο πλαίσιο κάποιου άλλου μεταπτυχιακού ή προπτυχιακού τίτλου σπουδών στην Ελλάδα ή το εξωτερικό. Η εργασία αυτή έχοντας εκπονηθεί από εμένα, αντιπροσωπεύει τις προσωπικές μου απόψεις επί του θέματος. Οι πηγές στις οποίες ανέτρεξα για την εκπόνηση της συγκεκριμένης διπλωματικής αναφέρονται στο σύνολό τους, δίνοντας πλήρεις αναφορές στους συγγραφείς, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο».

(Υπογραφή)

.....

ΚΑΛΟΓΕΡΑΣ ΠΑΥΛΟΣ

Φοιτητής MSc στη Διοικητική Επιστήμη και Τεχνολογία

Περίληψη

Ο σκοπός της παρούσας διπλωματικής εργασίας είναι η παρουσίαση και ανάλυση μίας αναδυόμενης και πολλά υποσχόμενης προσέγγισης στον τομέα των Μεγάλων Δεδομένων (Big Data) και της ενσωμάτωσης αυτών (Data Integration), μέσα από μία κατηγορία συστημάτων τα οποία αναφέρονται από την βιβλιογραφία ως Ενοποιημένα Συστήματα Διαχείρισης Βάσεων Δεδομένων (Federated Database Management Systems). Ειδικότερα, η εργασία εστιάζει σε μία υποκατηγορία των συστημάτων αυτών, με την ονομασία Polystores, η οποία αφορά μία εναλλακτική τεχνολογία διαχείρισης μεγάλων όγκων δεδομένων που προέρχονται από ετερογενείς πηγές, σε σχέση με πιο παραδοσιακές τεχνικές όπως είναι το μοντέλο της Αποθήκης Δεδομένων (Data Warehouse). Για την επίτευξη του σκοπού της εργασίας, αξιοποιήθηκε το Bigdawg, που αποτελεί μία polystore εφαρμογή ανοικτού κώδικα. Αυτή η εφαρμογή προσφέρει λύσεις διαχείρισης ετερογενών δεδομένων μέσα από μία ενιαία διεπαφή (Interface), που δίνει τη δυνατότητα στους χρήστες να επεξεργαστούν δεδομένα στο μητρικό τους μοντέλο.
Η συμβολή της εργασίας επεκτείνεται στην παρουσίαση μίας μεθοδολογίας εκχώρησης των δεδομένων του χρήστη στις ετερογενείς μηχανές αποθήκευσης που απαρτίζουν το BigDAWG. Ο κώδικας της εφαρμογής παραμετροποιήθηκε καταλλήλως, ενώ για την αναπαράσταση των δεδομένων παρουσιάστηκε ένα σενάριο μελέτης περίπτωσης σχετικό με την τεχνολογία των Έξυπνων Δικτύων (Smart Grids). Στο τελευταίο μέρος της εργασίας, διενεργήθηκαν προκαταρκτικά πειράματα συγκριτικής αξιολόγησης του polystore συστήματος, τα ευρήματα των οποίων ανέδειξαν χρήσιμα πλεονεκτήματα της εφαρμογής, αλλά και την ανάγκη για περαιτέρω βελτιώσεις.

Λέξεις Κλειδιά: Μεγάλα Δεδομένα, ενσωμάτωση δεδομένων, ετερογένεια, polystores, BigDAWG

Η σελίδα αυτή είναι σκόπιμα λευκή.

Abstract

The purpose of this thesis is the presentation and examination of an emerging and promising approach in the field of Big Data and Data Integration, through a system category which is mentioned by the literature as Federated Database Management Systems. In particular, this dissertation focuses on a subcategory of those systems, known as Polystores, concerning an alternative management technology of a sheer volume of data, which are derived from heterogeneous data sources, in contrast to more traditional techniques such as Data Warehouse models. In order to fulfill the purpose of this dissertation, the BigDAWG was utilized, which constitutes an open source polystore application. This application offers heterogeneous data management solutions by making use of a single interface, thus enabling users to process data in their native model.

The contribution of this thesis extends to the presentation of a methodology referring to the importing of user's data to the heterogeneous storage engines constituting the BigDAWG. The code of the application was properly reconfigured, while for data representation purposes, there was a case study exhibition regarding Smart Grid technology. In the last part of this study, some preliminary experiments of benchmarking were conducted, the findings of whom showcased the advantages of this application, but also the need for further improvement.

Keywords: Big Data, data integration, heterogeneity, polystores, BigDAWG

Η σελίδα αυτή είναι σκόπιμα λευκή.

Πίνακας περιεχομένων

Πίνακας εικόνων

Πίνακας πινάκων

Πίνακας διαγραμμάτων

1

Εισαγωγή

1.1 Η νέα πραγματικότητα των Big Data και η πρόκληση της ετερογένειας για την ενσωμάτωσή τους

Η σημερινή εποχή δε χαρακτηρίζεται τυχαία ως η εποχή των *Μεγάλων Δεδομένων* ***(Big Data)***. Ο όρος των Big Data χρησιμοποιείται για να περιγράψει τη ραγδαία αύξηση αυτών, σε τέτοιο βαθμό που κρίνεται εξαιρετικά πολύπλοκη η διαχείρισή τους με τη χρήση των παραδοσιακών τεχνικών και εργαλείων *διαχείρισης βάσεων δεδομένων* ***(Database Management Systems ή DBMS)***. Παράλληλα, η εφαρμογή τους συναντάται σε ποικίλους τομείς, όπως είναι οι τηλεπικοινωνίες, τα μέσα κοινωνικής δικτύωσης, το ηλεκτρονικό εμπόριο, η ιατρική, η αστρονομία, η βιομηχανία και γενικότερα οποιοσδήποτε κλάδος μπορεί να ευνοηθεί από την επεξεργασία τους για την εξαγωγή χρήσιμων συμπερασμάτων. Τα κύρια χαρακτηριστικά των Μεγάλων Δεδομένων είναι ο συνεχώς αυξανόμενος *Όγκος* ***(Volume)*** τους, η παραγωγή αυτών σε μεγάλη *Ταχύτητα* ***(Velocity)***, σε συνδυασμό με την ανάγκη για γρήγορη επεξεργασία τους σε πραγματικό χρόνο, και τέλος η *Ποικιλία* ***(Variety)*** με την οποία αυτά συναντώνται. Τα γνωρίσματα αυτά αναφέρονται εν συντομία ως τα 3V των Big Data (Laney, 2001). Μία ακόμη διάσταση που έρχεται να προστεθεί στις προηγούμενες τρείς, είναι αυτή της *Εγκυρότητας* ***(Veracity)***, δηλαδή η αβεβαιότητα που μπορεί να προκύψει για το κατά πόσο τα δεδομένα που διασπείρονται είναι αληθή και ανταποκρίνονται στην πραγματικότητα (Dong & Srivastava, 2013). Μάλιστα, στη διεύρυνση των χαρακτηριστικών αυτών έχει συντελέσει και η εμφάνιση των εφαρμογών που ανήκουν στο *Διαδίκτυο των*

Πραγμάτων ***(Internet of Things ή IoT)***, οι οποίες ευθύνονται για την παραγωγή τεράστιων όγκων δεδομένων, εξαιτίας του συσχετισμού αυτών με άλλα, χωροχρονικά (σχετικά με τον χώρο και τον χρόνο) δεδομένα, καθώς η άντληση αυτών από συσκευές IoT συνοδεύεται από πληροφορίες για τον τόπο και χρόνο καταγραφής τους (Wang, 2017).

Τα τελευταία χρόνια ολοένα και περισσότερες επιχειρήσεις και οργανισμοί έχουν αντιληφθεί ότι τα δεδομένα προσθέτουν αξία και κέρδος στις διεργασίες τους, με αποτέλεσμα να στρέφονται σε μία λογική λήψης επιχειρηματικών αποφάσεων με βάση αυτά *(data driven decision)*, η οποία θεωρείται ευρέως ότι θα προσδώσει ανταγωνιστικό πλεονέκτημα σε αυτούς. Ωστόσο, η μεγάλη πρόκληση που εμφανίζεται για τους οργανισμούς, και που η επιστημονική κοινότητα καλείται να αντιμετωπίσει, αφορά την *ετερογένεια* ***(heterogeneity)*** που παρουσιάζουν τα Μεγάλα Δεδομένα. Καθώς ο όγκος και η ροή αυτών αυξάνεται με πολύ μεγάλη ταχύτητα, λόγω των διαφορετικών *πηγών* ***(data sources)*** από τις οποίες προέρχονται, η αποκλειστική χρήση σχεσιακών ΒΔ και μοντέλων κρίνεται ανεπαρκής, με αποτέλεσμα οι επιχειρήσεις να στρέφονται σε λύσεις *μη σχεσιακών ΒΔ* ***(Non-relational ή NoSQL)*** για λόγους ευελιξίας και βέλτιστης απόδοσης (Fink et al., 2020). Στην πραγματικότητα όμως κάθε είδος ΒΔ (σχεσιακή και μη-σχεσιακή) εξυπηρετεί συγκεκριμένες περιπτώσεις, και επομένως η παράλληλη χρήση και των δύο τύπων είναι επιβεβλημένη, οδηγώντας στην ετερογένεια των δεδομένων, εφόσον αυτά ανήκουν σε ξεχωριστά συστήματα ΒΔ τα οποία χαρακτηρίζονται από διαφορετικά μοντέλα ***(schemas)***, κανόνες, τύπους αλλά και γλώσσα εκμαίευσης δεδομένων ***(query language)***. Η ετερογένεια επηρεάζει τον τρόπο με τον οποίο θα αποθηκευτούν τα δεδομένα, τις μεθόδους εισροής τους στις εφαρμογές των χρηστών, αλλά και τη διαδικασία *ενσωμάτωσής* τους ***(data integration)***. Οι λειτουργίες αυτές είναι ζωτικής σημασίας, εφόσον καθεμιά από αυτές είναι προαπαιτούμενο για την ενιαία αναπαράσταση αλλά και μεταγενέστερη ανάλυση των δεδομένων, μέσα από τεχνικές *Εξόρυξης Δεδομένων* ***(Data Mining)*** και *Μηχανικής Μάθησης* ***(Machine Learning ή ML)***, οι οποίες τελικά θα οδηγήσουν στη λήψη των αποφάσεων.

Επομένως, οι επιχειρήσεις καλούνται να διαχειριστούν ετερογενή δεδομένα που βρίσκονται αποθηκευμένα σε διαφορετικά συστήματα (π.χ. σχεσιακές/μη-σχεσιακές/αντικειμενοστραφείς/XML ΒΔ, αποθήκες σελίδων σε HTML, διεπαφές προγραμματισμού εφαρμογών ***(Application Programming Interface ή API***[1]***)*** και οποιοδήποτε γενικά σύστημα υποστηρίζει τη λήψη δεδομένων). Τα συστήματα αυτά παρέχουν πολύτιμη πληροφορία, όμως κατά την παράλληλη διαχείριση και επεξεργασία τους από τους χρήστες μοιραία θα προκύψουν προβλήματα ασυμβατότητας, πολυπλοκότητας και απόδοσης (Patidar et al., 2018). Επιπλέον, η ξεχωριστή ανάλυση όλων των επιμέρους δεδομένων που βρίσκονται στα διακριτά αυτά συστήματα, μέσα από αμέτρητα ερωτήματα προς τις βάσεις αυτών, αλλά και η σύνθεση και σύγκριση των υπολογισμένων απαντήσεων από αυτά τα ερωτήματα, είναι μια περίπλοκη διαδικασία η οποία κοστίζει σε χρόνο και χρήμα, καθώς απαιτείται κατάλληλα εξειδικευμένο προσωπικό με αντίστοιχες γνώσεις. Προκύπτει έτσι η ανάγκη για τους χρήστες να έχουν μια ενιαία οπτική ενός συνολικού

[1] Προγραμματιστική γέφυρα μεταξύ δύο διαφορετικών εφαρμογών που επιτρέπει την μεταξύ τους επικοινωνία μέσω ανταλλαγής δεδομένων.

συστήματος, παρά διαφορετικές οπτικές που ανήκουν στην εκάστοτε πηγή δεδομένων. Για το σκοπό αυτό, πρέπει να ξεπεραστεί η μεγάλη πρόκληση στον τομέα των Big Data που δεν είναι άλλη από την ενσωμάτωση των δεδομένων που ανήκουν σε ετερογενείς πηγές, συχνά αποκαλούμενες ως *πληροφοριακά σιλό* ***(information silos)***.

Η ανάγκη του data integration δεν εμφανίστηκε με την έλευση των Big Data. Ήδη από το 1990 οι επιχειρήσεις ασχολήθηκαν με το μοντέλο της *αποθήκης δεδομένων* ***(Data Warehouse ή DW)***, προκειμένου να δοθεί λύση στο πρόβλημα της ετερογένειας. Στην περίπτωση αυτή, εφαρμόζεται η τεχνική ETL *(Extract Transform Loading)*, η οποία αφορά την εξαγωγή δεδομένων από τις πηγές τους, με στόχο τη μετατροπή και φόρτωση αυτών σε μία αποθήκη δεδομένων. Η μεγάλη επιτυχία αυτής της προσέγγισης σχετίζεται με τη δημιουργία ενός κοινού πλέον μοντέλου, σύμφωνα με το οποίο οργανώνονται τα μέχρι πρότινος διακριτά δεδομένα. Βαδίζοντας όμως στη σημερινή εποχή, η ETL διαδικασία μοιάζει να υστερεί στη διαχείριση των Big Data λόγω του υψηλού κόστους *μετατροπής (Transform)* των δεδομένων αλλά και της ανάγκης για συχνή *ενημέρωση (Loading)* στο DW. Το γνωστό στην επιστημονική κοινότητα *"one size fits all"* που προήλθε από το Data Warehouse μοντέλο, άρχισε να αμφισβητείται και θεωρήθηκε ότι δε μπορεί να υποστηρίξει πλήρως αποδοτικά τη σύγχρονη διαδικασία του data integration (Stonebraker & Cetintemel, 2005). Από το 2010 και μετά και λόγω της έκρηξης των δεδομένων, εμφανίστηκαν *τα ενοποιημένα συστήματα διαχείρισης βάσεων δεδομένων* ***(Federated DBMS)***. Ένα τέτοιο σύστημα αποτελεί ένα λογισμικό σε ρόλο μεσάζοντα το οποίο έχει τη δυνατότητα να επικοινωνεί απευθείας με τις διακριτές ΒΔ, διατηρώντας άθικτα τα τοπικά μοντέλα δεδομένων (Stonebraker, 2015).

Το παλαιό σλόγκαν μετονομάστηκε στην κοινότητα των δεδομένων σε *"one size does not fit all"*, ενώ η πιο πρόσφατη συνεισφορά στον τομέα του data integration αφορά μία παραλλαγή των federated DBMS, με την ονομασία Polystores. Η έκρηξη των δεδομένων από peta-scale σε exa-scale, επιβάλλει την εξερεύνηση τους στο αντίστοιχο μητρικό μοντέλο της ΒΔ η οποία τα φιλοξενεί, μία ιδιότητα η οποία χαρακτηρίζει τα Polystore συστήματα (Patidar et al., 2018).

Συμπερασματικά, γίνεται κατανοητό ότι οι επιχειρήσεις οδεύουν σε μία νέα πραγματικότητα data driven decision λογικής. Η έλευση των Big Data προσφέρει συνδυαστικά νέες ευκαιρίες και επιπρόσθετη αξία στους οργανισμούς, αλλά και προκλήσεις, καθώς βασική προϋπόθεση αποτελεί η σωστή διαχείριση των διακριτών πηγών δεδομένων. Το γεγονός όμως ότι οι πηγές αυτές παρουσιάζουν μεγάλη ετερογένεια και διαθέτουν σημαντικό όγκο δεδομένων, επιδρά ως ανασταλτικός παράγοντας στην επεξεργασία τους, καθιστώντας συχνά τα αρχικά μοντέλα (Data Warehouses) ως μη αποδοτικά. Σημαντικά βήματα προόδου έχουν σημειωθεί μέσα από νέα συστήματα (federated DBMS). Η πρόσφατη ανακάλυψη μίας διαφορετικής εκδοχής αυτών, με την ονομασία Polystores, επιτρέπει την βέλτιστη ανάλυση των δεδομένων στο δικό τους μοντέλο, εξασφαλίζοντας μέγιστη αποδοτικότητα και εξαλείφοντας παράλληλα τα περαιτέρω κόστη που προκύπτουν από την εφαρμογή παλαιότερων αρχιτεκτονικών για τη διαδικασία του data integration.

1.2 Αντικείμενο διπλωματικής

Αντικείμενο της παρούσας διπλωματικής εργασίας αποτελεί η ανάλυση και μελέτη μιας υποκατηγορίας ενοποιημένων συστημάτων διαχείρισης βάσεων δεδομένων, που αποκαλούνται polystores. Για τη βέλτιστη απεικόνιση των δυνατοτήτων αυτών των συστημάτων, η εργασία θα εστιάσει σε μία συγκεκριμένη polystore εφαρμογή ανοικτού κώδικα, η οποία ονομάζεται BigDAWG.

Η δυσκολία του εγχειρήματος αυτού, έγκειται στην εύρεση πρακτικών τρόπων, ώστε να εισαχθούν δεδομένα στις ετερογενείς ΒΔ που απαρτίζουν το BigDAWG, κάτι το οποίο προϋποθέτει την παραμετροποίηση του κώδικά του μέσα από ένα πλήθος διαφορετικών προγραμματιστικών γλωσσών. Για τον σκοπό αυτό, θα παρουσιαστεί μία μελέτη περίπτωσης ιδανική για τη χρήση των polystores, τα δεδομένα της οποίας θα ενσωματωθούν στην εφαρμογή. Τέλος, θα πραγματοποιηθούν έλεγχοι απόδοσης του συστήματος αυτού, έναντι της χρήσης μεμονωμένων βάσεων δεδομένων.

1.2.1 Συνεισφορά

Η συνεισφορά της διπλωματικής εργασίας συνοψίζεται ως εξής:

1. Παρουσίαση βέλτιστων πρακτικών για την ενσωμάτωση δεδομένων του χρήστη σε μία polystore εφαρμογή.
2. Συγκριτική αξιολόγηση αναφορικά με την υλοποίηση μίας polystore προσέγγισης μέσω του BigDAWG, σε σχέση με τη χρήση αυτοτελών μηχανών αποθήκευσης.

1.3 Οργάνωση κειμένου

Εργασίες σχετικές με το αντικείμενο της διπλωματικής παρουσιάζονται στο Κεφάλαιο 2. Το Κεφάλαιο 3 συζητά εις βάθος τις απαραίτητες θεωρητικές γνώσεις, ώστε να γίνει κατανοητή από τον αναγνώστη η διαδικασία υλοποίησης της εργασίας. Στο Κεφάλαιο 4 περιγράφεται η δομή των δεδομένων της μελέτης περίπτωσης που θα εισαχθούν στην εφαρμογή, οι βέλτιστες πρακτικές για την ενσωμάτωσή τους στο BigDAWG, και τέλος η επίδειξη των δυνατοτήτων του συστήματος. Η συγκριτική αξιολόγηση μεταξύ του BigDAWG και μεμονωμένων storage engines, όπως και κάποια ποιοτικά χαρακτηριστικά του συστήματος περιλαμβάνονται στο Κεφάλαιο 5. Στο Κεφάλαιο 6 παρουσιάζεται η πρακτική άσκηση που εκπονήθηκε στα πλαίσια φοίτησης στο μεταπτυχιακό πρόγραμμα, καθώς και δραστηριότητες και έργα που πραγματοποιήθηκαν κατά τη διάρκειά της. Τέλος, το Κεφάλαιο 7 περιέχει τη σύνοψη των συμπερασμάτων αλλά και τις μελλοντικές επεκτάσεις της παρούσας εργασίας.

2

Σχετικές εργασίες

2.1 Big Data: Heterogeneity and Data Integration

Μαζί με την έλευση της Big Data εποχής, επήλθε και η ανάγκη για χρήση διακριτών συστημάτων διαχείρισης δεδομένων. Η χρήση ενός και μοναδικού DBMS ενδέχεται να επιφέρει εκπτώσεις στην απόδοση και την ευελιξία των εφαρμογών. Συνεπώς, επιβάλλεται η υλοποίηση διαφορετικών ΒΔ, όπου κάθε μία από αυτές είναι ιδανική για την αποθήκευση συγκεκριμένου τύπου δεδομένων και την εκτέλεση επιμέρους διεργασιών. Φυσικά, η ύπαρξη της ετερογένειας των διαφορετικών πηγών, δημιουργεί προκλήσεις τις οποίες η επιστημονική κοινότητα οφείλει να αντιμετωπίσει, σχετικά με την ταυτόχρονη πρόσβαση στις ξεχωριστές ΒΔ, αλλά και την ενοποίηση αυτών.

Οι **Bondiombouy & Valduriez**, **2016** παρουσίασαν τις διαφορετικές λύσεις που είναι διαθέσιμες για την αποθήκευση των ετερογενών δεδομένων, πέρα από αυτήν των παραδοσιακών σχεσιακών συστημάτων διαχείρισης βάσεων δεδομένων (από εδώ και πέρα ***RDBMS***). Σύμφωνα με τους ίδιους συγγραφείς, οι 3 κύριες κατηγορίες ***συστημάτων αποθήκευσης*** είναι: (i) *Κατανεμημένες Αποθήκες* ***(Distributed Storages)***, (ii) *Συστήματα Διαχείρισης Βάσεων δεδομένων* ***(DBMS)*** και (iii) *Κατανεμημένη Επεξεργασία* ***(Distributed Processing)***.

Τα *Distributed Storages* συστήματα ακολουθούν την αρχιτεκτονική "client/server", όπου τα δεδομένα αποθηκεύονται σε έναν απομακρυσμένο *διακομιστή* ***(server)***, και η προσπέλαση αυτών γίνεται από *συστήματα/πελάτες* ***(clients)***. Τα δεδομένα που υποστηρίζονται μπορεί να είναι *αρχεία* ***(Distributed File Systems ή DFS)*** (π.χ. GFS, HDFS) ή *αντικείμενα* ***(Object Storage)***, δηλαδή "πακέτα" πληροφορίας όπου κάθε ένα συνοδεύεται από ένα *αναγνωριστικό* ***(identifier)***. Όσον αφορά τα *DBMS*, οι ίδιοι ερευνητές αναφέρθηκαν κυρίως στα συστήματα NoSQL τα οποία αποτελούν εναλλακτική των RDMBS. Σε αυτήν την περίπτωση τα δεδομένα δεν αποθηκεύονται σε στατικούς πίνακες, με αποτέλεσμα η μορφή τους να είναι δυναμική. Το γεγονός αυτό καθιστά τα NoSQL συστήματα ιδιαίτερα ευέλικτα, θυσιάζοντας όμως τη συνοχή του μοντέλου. Τα συστήματα αυτά χωρίζονται στις εξής υποκατηγορίες: (i)

Key/Value Systems (π.χ. Redis, Dynamo), δηλαδή αντιστοιχίσεις μεταξύ κλειδιών και των τιμών τους, (ii) **Wide Column Systems** (π.χ. Cassandra, Accumulo), που διασφαλίζουν την αποθήκευση δεδομένων σε δυναμικούς πίνακες όπου το όνομα της στήλης μπορεί να είναι διαφορετικό, ανάλογα με τη γραμμή της εγγραφής, και (iii) **Document Systems** (π.χ. MongoDB, CouchDB), δηλαδή μια εξελιγμένη μορφή των **Key/Value Systems**, με τη διαφορά ότι οι τιμές είναι αρχεία της μορφής " JSON " *(JavaScript Object Notation)*, "XML" *(Extensive Markup Language)* και "YAML" *(Yet Another Markup Language)*. Επιπλέον υποκατηγορίες των NoSQL συστημάτων είναι οι εξής: (iv) **Graph Database Systems** (π.χ. Neo4J, Trinity), όπου τα δεδομένα αναπαρίστανται ως κόμβοι αλλά και ακμές οι οποίες συνδέουν τους κόμβους αυτούς, (v) **Triplestores/RDF Stores** (π.χ. AllegroGraph, GraphDB), δηλαδή συστήματα τα οποία απεικονίζουν τα δεδομένα ως τριάδες της μορφής (υποκείμενο/subject, περιουσία/predicate, αντικείμενο/object), όπως για παράδειγμα (IBM, δημιούργησε, SQL), και (vi) **Hybrid Data Stores** (π.χ. Hybrid Transactional/Analytical Processing ή HTAP), υβριδικά συστήματα τα οποία ενσωματώνουν ταυτόχρονα διαφορετικά μοντέλα αποθήκευσης δεδομένων. Τέλος, τα *Distributed Processing* συστήματα (π.χ. Hadoop MapReduce), υποστηρίζουν την ταυτόχρονη επεξεργασία μεγάλου όγκου δεδομένων σε πραγματικό χρόνο, μέσω εκτέλεσης παράλληλων διεργασιών *(parallel programming)* (Bondiombouy & Valduriez, 2016).

Η ταυτόχρονη ύπαρξη διακριτών συστημάτων αποθήκευσης οδηγεί στην εμφάνιση της ***ετερογένειας***, καθώς κάθε ένα από αυτά διαθέτει ξεχωριστή γλώσσα και κανόνες. Οι **Euzenat & Shvaiko, 2013** υποστήριξαν ότι δεν υπάρχει μόνο ένα είδος ετερογένειας, και προχώρησαν στην κατηγοριοποίηση του όρου αυτού σε 4 τύπους. Ο πρώτος αφορά την *συντακτική ετερογένεια* ***(syntactic heterogeneity)***, η οποία χαρακτηρίζει δύο πηγές δεδομένων εκφρασμένες σε διαφορετική γλώσσα. Η *ορολογική ετερογένεια* ***(terminological heterogeneity)*** αναφέρεται στην ύπαρξη διαφορετικών παραλλαγών στην ονομασία της ίδια *οντότητας* [2] ***(entity)***, όπως αυτή ορίζεται στην αντίστοιχη πηγή δεδομένων. Ακόμα, η *εννοιολογική ετερογένεια* ***(conceptual heterogeneity)***, εντοπίζεται κατά τη διαφορετική αναπαράσταση του μοντέλου για την περιγραφή του ίδιου πεδίου ενδιαφέροντος. Ο τελευταίος τύπος ορίζεται ως *σημειολογική ετερογένεια* ***(semiotic heterogeneity)*** και σχετίζεται με τη διαφορετική αντίληψη του ανθρώπου ως προς τις οντότητες. Κάθε άνθρωπος αντιλαμβάνεται ξεχωριστά τις οντότητες, με αποτέλεσμα συχνά να επιχειρεί να τις συνδυάσει με τρόπο ο οποίος προκαλεί νοητικά σφάλματα, και τα οποία οι υπολογιστικές μηχανές δεν έχουν την ευφυΐα να αντιμετωπίσουν (Euzenat & Shvaiko, 2013). Αξίζει να σημειωθεί ότι διαφορετικοί τύποι ετερογένειας μπορούν να εμφανιστούν παράλληλα, αυξάνοντας περισσότερο την πολυπλοκότητα των εφαρμογών.

Εφόσον τα δεδομένα προσφέρουν ακόμα μεγαλύτερη αξία όταν συνδυάζονται μεταξύ τους σε μία ενοποιημένη μορφή, το πεδίου του data integration αποτελεί ένα από τα σημαντικότερα αντικείμενα μελέτης της επιστημονικής κοινότητας. Οι **Dong & Srivastava, 2013** ανέλυσαν τις τρείς κύριες τεχνικές

[2] Ένα αντικείμενο ενδιαφέροντος στον πραγματικό κόσμο. Διαθέτει χαρακτηριστικά ή ιδιότητες, δηλαδή στοιχεία που την χαρακτηρίζουν.

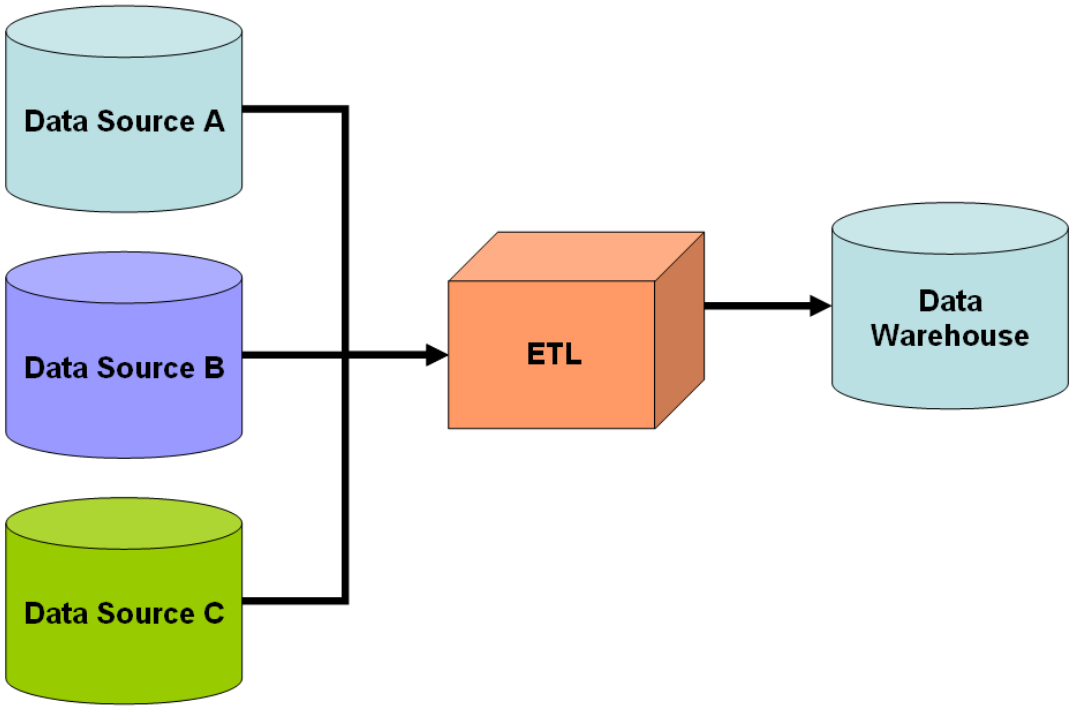
ενοποίησης δεδομένων. Η πρώτη τεχνική αναφέρεται ως αυτή της *αντιστοίχισης σχήματος* ***(schema mapping)***, και περιλαμβάνει δύο φάσεις. Στην πρώτη φάση δημιουργείται ένα "παγκόσμιο" *(global)* schema που δρα ως μεσολαβητής, ενώ στη δεύτερη φάση πραγματοποιούνται *αντιστοιχίσεις* ***(mappings)*** μεταξύ του μεσολαβητή αυτού και των τοπικών schemas που ανήκουν στις εκάστοτε πηγές δεδομένων. Σε αυτήν την περίπτωση, τα queries εκφράζονται μέσω του διαμεσολαβητή, με τέτοιο τρόπο ώστε οι διακριτές ΒΔ να συμπεριφέρονται ως μία κοινή πηγή δεδομένων. Η δεύτερη τεχνική ονομάζεται *σύνδεση εγγραφής* ***(record linkage)***, και αφορά την εύρεση και σύνδεση εγγραφών που ανήκουν σε διαφορετικές ΒΔ, οι οποίες όμως αναφέρονται στις ίδιες οντότητες. Η τεχνική αυτή χρησιμοποιείται κυρίως σε στατικές ΒΔ οι οποίες περιλαμβάνουν *δομημένα* (παριστάνονται με αυστηρή μορφοποίηση) δεδομένα, με αποτέλεσμα η εφαρμογή της σε δυναμικές πηγές που χαρακτηρίζονται από έντονη ετερογένεια να παρουσιάζει αντικειμενικές δυσκολίες.

Οι πρώτες δύο τεχνικές χρησιμοποιήθηκαν κατά κόρον στην παραδοσιακή μορφή του data integration, και πιο συγκεκριμένα αποτέλεσαν μέρος της ETL *(Extract Transform Load)* διαδικασίας κατά τη φόρτωση των δεδομένων σε Data Warehouses (*αναφορά στην ενότητα 2.2*). Η τελευταία τεχνική εμφανίστηκε σχετικά πρόσφατα, με την έλευση των Big Data και σχετίζεται με το 4ο V που αφορά την εγκυρότητά τους. Πρόκειται για τη *σύντηξη δεδομένων* ***(data fusion)***, δηλαδή ένα συνδυασμό μεθόδων που στοχεύουν στην εύρεση των διαφορών μεταξύ των διακριτών πηγών δεδομένων, η οποία θα οδηγήσει στην "*αλήθεια* του πραγματικού κόσμου". Μία από αυτές τις μεθόδους είναι αυτή της *ψηφοφορίας*, όπου επιλέγεται το *χαρακτηριστικό* ***(attribute)*** με τις περισσότερες εμφανίσεις μεταξύ των πηγών (Dong & Srivastava, 2013).

2.2 ETL και Αποθήκες Δεδομένων στο Big Data Integration

Η παραδοσιακή τεχνική για το data integration επιτεύχθηκε μέσα από τις αποθήκες δεδομένων. Δεδομένα από διαφορετικές πηγές προέλευσης και αρχιτεκτονικά μοντέλα, μέσω της ETL διαδικασίας, ενσωματώνονται σε μία κοινή αποθήκη/στόχο που χαρακτηρίζεται από ένα ενιαίο schema. Η ETL μέθοδος παρέχει 3 βασικές λειτουργίες:

i. **Extract**: Ανάγνωση δεδομένων από τις πηγές τους.
ii. **Transform**: Μετατροπή της αρχικής μορφής των δεδομένων, σε μία νέα η οποία είναι εναρμονισμένη με τις απαιτήσεις της αποθήκης.
iii. **Load**: Φόρτωση των δεδομένων στο data warehouse.



Εικόνα 1: Αποθήκη Δεδομένων μέσα από την ETL διαδικασία (https://www.wikiwand.com/en/Data_integration).

Στη σύγχρονη εποχή τα data warehouse συστήματα μοιάζουν να υστερούν στην υποστήριξη των μεγάλων δεδομένων και των χαρακτηριστικών τους. Η μετατροπή *(transform)*, δεν είναι το ίδιο αποδοτική σε περιπτώσεις που τα δεδομένα είναι αδόμητα *(δεν είναι οργανωμένα με προκαθορισμένο τρόπο*) ή ημιδομημένα (*υπάρχουν περιορισμένες ενδείξεις για τον τύπο τους*), και για να ξεπεραστεί το πρόβλημα αυτό χρειάζονται επιπλέον τροποποιήσεις μέσα από την ένταξη επιπρόσθετου κώδικα (Hurwitz et al., 2013). Εκ παραλλήλου, το στάδιο της μετατροπή των δεδομένων συνοδεύεται από υψηλά κόστη, ενώ το γεγονός ότι τα περισσότερα εταιρικά δεδομένα δεν είναι "καθαρά" στην αρχική τους μορφή, επιβαρύνει το πρόβλημα αυτό (Stonebraker, 2015).

Οι **Stonebraker & Cetintemel, 2005** ήταν οι πρώτοι που αμφισβήτησαν το μότο "One size fits all", δηλαδή την υποστήριξη όλων των εφαρμογών από **ένα** και μόνο DBMS, ανεξαρτήτου χαρακτηριστικών και απαιτήσεων. Ήδη από το 2005, οι δύο ερευνητές διέβλεψαν ότι τα δεδομένα αρχίζουν να παράγονται σε μεγάλους όγκους λόγω των διάφορων τύπων αισθητήρων που εφαρμόζονταν από τότε σε ποικίλους τομείς, όπως ο στρατός και ο έλεγχος της οδικής κυκλοφορίας. Χαρακτηριστικό των τομέων αυτών, είναι η ανάγκη για παρακολούθηση και καταγραφή των δεδομένων σε πραγματικό χρόνο. Ως αποτέλεσμα, η επιστημονική κοινότητα έστρεψε την προσοχή της σε *εφαρμογές επεξεργασίας ροών δεδομένων* (**stream processing applications**).

Τα συμβατικά DBMS μοντέλα εκτελούν *εξερχόμενη* (**outbound**) επεξεργασία, καθώς πρώτα αποθηκεύουν τα δεδομένα και ύστερα αυτά υπόκεινται σε ανάλυση. Αντίθετα, οι stream processing εφαρμογές πραγματοποιούν *εισερχόμενη* (**inbound**) επεξεργασία, δηλαδή αποθηκεύονται μόνο τα queries προς

εκτέλεση και στη συνέχεια "μεταβιβάζονται" τα δεδομένα προς επεξεργασία. Το στάδιο της αποθήκευσης πριν την επεξεργασία, είναι αυτό που επιφέρει μεγάλη καθυστέρηση αλλά και επιπλέον κόστη, και καθιστά τις παραδοσιακές λύσεις DBMS μη βιώσιμες σε real time processing εφαρμογές. (Stonebraker & Cetintemel, 2005). Εντούτοις, δεν υπάρχει ένα DBMS το οποίο να προσφέρει βέλτιστη απόδοση ως προς όλα τα είδη και τις απαιτήσεις των data-centric εφαρμογών. Το ίδιο ισχύει και για τα **data warehouse** συστήματα, καθώς αυτά χαρακτηρίζονται από ένα ενιαίο μοντέλο ("one size") και όχι από τα διακριτά schemas των διαφορετικών πηγών δεδομένων. Συνεπώς, είναι προτιμότερη η φόρτωση των δεδομένων σε πολλαπλά DBMS, δηλαδή για παράδειγμα τα δομημένα δεδομένα σε RDBMS, τα real time σε stream processing engines, τα ιστορικά δεδομένα σε μορφή πίνακα, και τα ημιδομημένα σε αρχεία JSON (Stonebraker 2015).

Οι **Sabtu et al., 2017** ασχολήθηκαν επίσης με τα *real time δεδομένα* και ειδικότερα τόνισαν πως η παραδοσιακή ETL διαδικασία υστερεί στην υποστήριξή τους, ενώ αντίθετα ενδείκνυται σε εφαρμογές που περιλαμβάνουν *batch data*. Όσον αφορά τα τελευταία, πρόκειται για up to date δεδομένα που εισέρχονται ανά παρτίδες σε προγραμματισμένα χρονικά διαστήματα, και η φόρτωσή τους λαμβάνει χώρα σε ώρες εκτός αιχμής (συνήθως το βράδυ). Από την άλλη πλευρά, τα δεδομένα σε πραγματικό χρόνο ή αλλιώς stream data, διαφοροποιούνται από τα batch data κατέχοντας τρία βασικά χαρακτηριστικά, τα οποία είναι η **υψηλή διαθεσιμότητά/ high availability** (συνεχόμενη ροή αυτών), **χαμηλή αδράνεια/ low latency** (σε αντίθεση με τα batch data που η ανανέωσή τους πραγματοποιείται περιοδικά, στα stream data γίνεται ανά μικρά χρονικά διαστήματα) και τέλος **οριζόντια επεκτασιμότητα/horizontal scalability.** Η τελευταία έννοια, σχετίζεται με την προσθήκη ανεξάρτητων servers για την διεκπεραίωση εργασιών με σκοπό την αύξηση της απόδοσης και μείωση του κινδύνου για διακοπή της συνεχόμενης ροής, σε αντιδιαστολή με την κάθετη επεκτασιμότητα όπου η απόδοση εξαρτάται από τις δυνατότητες μίας και μόνο υπολογιστικής μηχανής.

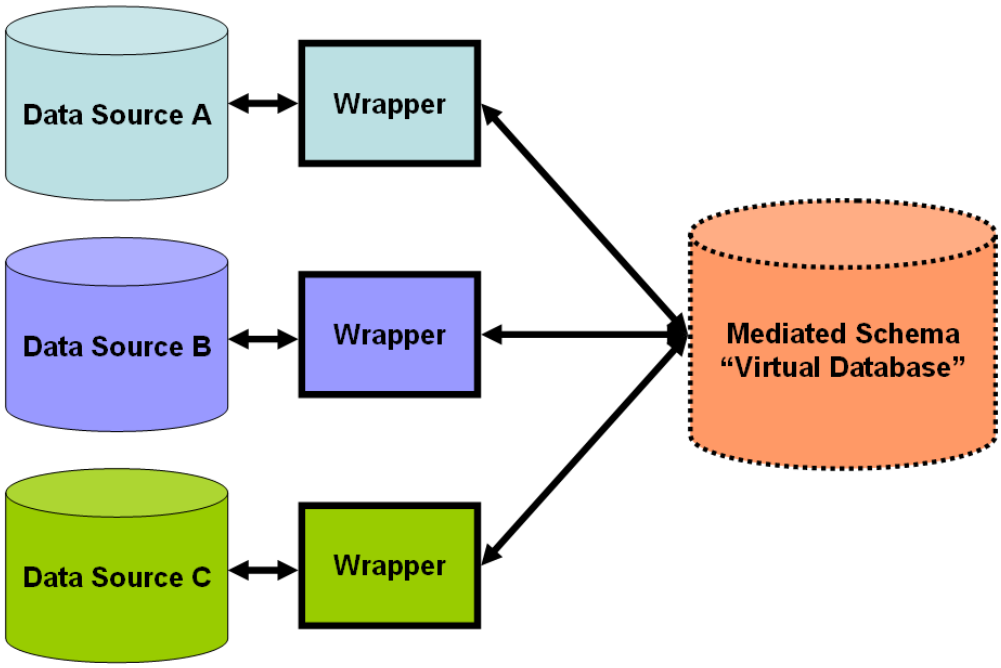
Ως συνέπεια των παραπάνω χαρακτηριστικών, προκύπτει ότι η μεθοδολογία του ETL υστερεί στην εξυπηρέτηση των near real time environments ή αλλιώς εφαρμογών που παράγουν δεδομένα σε πραγματικό χρόνο. Κατά το στάδιο του ***Extraction*** είναι υψίστης σημασίας η άρτια λήψη των δεδομένων δίχως να χαθεί πολύτιμη πληροφορία. Η ύπαρξη ετερογενών data sources (*high availability*) αποτελεί κίνδυνο τόσο για την ομαλή εξαγωγή των δεδομένων όσο και για την ενοποίησή τους. Προβλήματα παρουσιάζονται και στη διαδικασία του ***Transformation***, εφόσον τα real time δεδομένα ανανεώνονται σε ρυθμό πολύ μεγαλύτερο (*low latency*) από ότι τα batch data. Κάθε τέτοια ανανέωση συνεπάγεται και την αντίστοιχη επεξεργασία (transform) των master data, δηλαδή των δεδομένων μέσα από τα οποία γίνονται ομαδοποιήσεις επί των transactional data (π.χ. πελάτης, κατάστημα). Επομένως, η διαδικασία του transformation παύει να είναι το ίδιο αποδοτική καθώς γίνεται συχνότερα και μάλιστα για μικρότερο επιμέρους όγκο δεδομένων. Η τελευταία πρόκληση σχετίζεται με το ***Loading***, υπό την έννοια ότι δε θα πρέπει να επικαλύπτεται η εργασία της φόρτωσης των δεδομένων με αυτήν της πολυδιάστατης

επεξεργασία τους (OLAP ή Online Analytical Processing), καθώς σε μία τέτοια περίπτωση οι αναλύσεις ενδέχεται να οδηγήσουν σε αντιφατικά συμπεράσματα (Sabtu et al., 2017).

2.3 Ενοποιημένα συστήματα διαχείρισης βάσεων δεδομένων

Όπως διαπιστώθηκε από την προηγούμενη ενότητα, η μέθοδος του ETL για την αποθήκευση δεδομένων σε data warehouse συστήματα, επιφέρει επιπλέον κόστη κατά τη διαδικασία του data integration, και ταυτόχρονα μειώνει την απόδοση των εφαρμογών που στηρίζονται σε ετερογενείς βάσεις Μεγάλων Δεδομένων. Ως αποτέλεσμα, η επιστημονική κοινότητα εξερεύνησε λύσεις *ενοποιημένων συστημάτων **(federated systems)***, τα οποία επιτρέπουν την απευθείας επεξεργασία των δεδομένων στις διακριτές ΒΔ που ανήκουν. Τα συστήματα αυτά ακολουθούν μια διαφορετική αρχιτεκτονική από την αντίστοιχη των αποθηκών δεδομένων, και πιο συγκεκριμένα βασίζονται στο μοντέλο "mediator/wrapper".

Ένα federated ή αλλιώς virtual system, παρέχει επί της ουσίας μία κοινή ***οπτική*** πάνω στα δεδομένα *(Global View to Applications ή GVA)*, καθώς επιτρέπει την επεξεργασία τους ως ένα ενιαίο σύνολο μέσα από queries τα οποία επικοινωνούν **απευθείας** με τις αντίστοιχες ΒΔ. Αυτή η διεργασία επιτυγχάνεται μέσα από ένα λογισμικό με την ονομασία "mediator", το οποίο δρα ως μεσάζοντας και επικοινωνεί με τις πηγές δεδομένων με τη βοήθεια των "wrappers". Πιο συγκεκριμένα, το "mediator" λογισμικό είναι υπεύθυνο για την "κατασκευή" των queries, ενώ οι "wrappers" αποτελούν διεπαφές *(APIs)* ανάμεσα στο μεσάζοντα και στις βάσεις δεδομένων εξάγοντας το schema τους. Επομένως, τα δεδομένα δεν χρειάζεται να αποθηκευτούν εκ νέου σε μία νέα βάση, γεγονός το οποίο εξοικονομεί κόστη που προκύπτουν από άλλες data integration τεχνικές (π.χ. transformation). Αξίζει να σημειωθεί ότι επιτυγχάνεται και η βέλτιστη εκμετάλλευση των δεδομένων, λόγω της αξιοποίησής τους στο μητρικό μοντέλο που ανήκουν, εν αντιθέσει με τη μεταφορά τους (**migration**) σε άλλα schemas.



Εικόνα 2: Η αρχιτεκτονική "mediator/wrapper" για τα federated systems (https://www.wikiwand.com/en/Data_integration).

Με δεδομένο ότι μπορούν να υπάρξουν διαφορετικά σενάρια εφαρμογών data integration, οι **Tan et al., 2017** προχώρησαν στην κατηγοριοποίηση των ενοποιημένων συστημάτων και πρότειναν μάλιστα ένα θεωρητικό πλαίσιο σύμφωνα με το οποίο μπορεί να γίνει η αξιολόγησή τους. Ειδικότερα, με γνώμονα το είδος των πηγών δεδομένων αλλά και των *διεπαφών υποβολής ερωτημάτων* (**query interfaces**) προς αυτά, διέκριναν τα παρακάτω συστήματα:

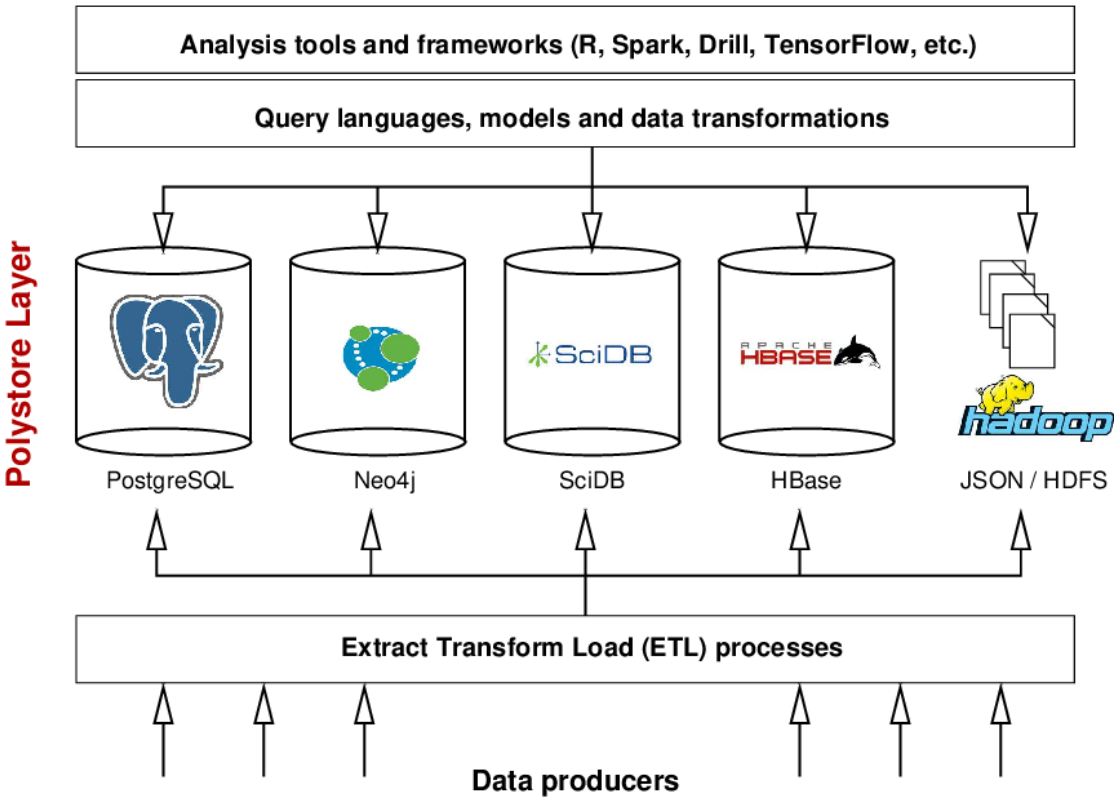
i. ***Federated Database Systems***: Συστήματα τα οποία χαρακτηρίζονται από ομογενείς πηγές δεδομένων και *ένα* query interface.
ii. ***Polyglot Systems***: Σύνολο ομογενών πηγών δεδομένων οι οποίες είναι προσπελάσιμες από πολλαπλά query interfaces.
iii. ***Multistore Systems***: Μία διεπαφή υποβολής ερωτημάτων για την εκμαίευση αποτελεσμάτων από ετερογενείς πηγές δεδομένων.
iv. ***Polystores Systems***: Συνδυασμός των Polyglot και Multistore συστημάτων. Ενσωματώνουν ετερογενείς πηγές δεδομένων παρέχοντας στους χρήστες μια πληθώρα από query interfaces.

Η αξιολόγηση των παραπάνω γίνεται με βάση τους εξής παράγοντες: *ετερογένεια*, *αυτονομία*, *διαφάνεια*, *ευελιξία* και *βελτιστοποίηση*. Ειδικότερα, η απόδοση ενός συστήματος κρίνεται από την ικανότητά του να παρέχει πρόσβαση στις **ετερογενείς** ΒΔ, να τις επεξεργάζεται και να μπορεί να αντιστοιχίσει τα

διάφορα μοντέλα των query interfaces σε αυτές. Κάθε επιμέρους DBMS χρειάζεται σε ένα βαθμό να διαθέτει τη δική του **αυτονομία**, έτσι ώστε να "αποφασίζει" για τη σύνδεση/αποσύνδεσή του από το εκάστοτε ενοποιημένο σύστημα, αλλά και την παράλληλη υποστήριξη των τοπικών και ανεξάρτητων διεργασιών του. Ο όρος της **διαφάνειας,** αναφέρεται στο κατά πόσο ευδιάκριτη είναι η δομή ενός τέτοιου συστήματος, ενώ αυτός της **ευελιξίας** στη δυνατότητα ταυτόχρονης υλοποίησης ποικίλων διεργασιών ανάλυσης (π.χ. καθαρισμός δεδομένων, συναρτήσεις ορισμένες από χρήστες). Τέλος, η δομή θα πρέπει να είναι και η **βέλτιστη** δυνατή, με χαρακτηριστικό παράδειγμα την τοποθέτηση των δεδομένων στις κατάλληλες ΒΔ οι οποίες μπορούν να εξυπηρετήσουν τις ανάγκες που υφίστανται ανάλογα με τον σκοπό της εφαρμογής (Tan et al., 2017).

2.3.1 Εργασίες σχετικές με τα Polystore συστήματα

Τα Polystores αποτελούν μια ανερχόμενη λύση όσον αφορά τη διαδικασία ενοποίησης ετερογενών πηγών δεδομένων. Το βασικό τους πλεονέκτημα έγκειται στο γεγονός ότι επιτρέπουν την επεξεργασία των δεδομένων στο δικό τους τοπικό μοντέλο, μέσα από query interfaces τα οποία βρίσκονται ένα επίπεδο "πάνω" από τις επιμέρους ΒΔ. Άλλωστε, η αποθήκευση και επεξεργασία των δεδομένων με τον ίδιο τρόπο που χρησιμοποιούνται από τις εφαρμογές στις οποίες είναι αυτά αποθηκευμένα, απλοποιεί το προγραμματιστικό μοντέλο και διευκολύνει την διαδικασία αποκεντροποίησης των δεδομένων (Ghosh, 2010).



Εικόνα 3: Τα επίπεδα ενός Polystore συστήματος (Leclercq & Savonnet, 2018).

Παρακάτω ακολουθούν πρόσφατες ερευνητικές εργασίες σχετικά με τα συστήματα αυτά:

Οι **Kharlamov et al., 2016** παρουσίασαν ένα σύστημα με την ονομασία OPTIQUE, το οποίο βασίζεται σε μια παρεμφερή αρχιτεκτονική με αυτή των polystores, η οποία λέγεται OBDA[3]. Σκοπός της εφαρμογής είναι η υποστήριξη της διενέργειας διαγνωστικών ελέγχων σε πραγματικό χρόνο των τουρμπινών παραγωγής ενέργειας της εταιρίας Siemens. Ειδικότερα, για τον έλεγχο μιας τουρμπίνας αξιοποιούνται δεδομένα προερχόμενα από ετερογενείς πηγές, τα οποία μπορεί να είναι δεδομένα ροής που παράγονται από αισθητήρες[4], ιστορικά (π.χ. τυχόν παρελθοντικές επισκευές) αποθηκευμένα σε στατική δομή, ακόμα και δεδομένα καιρικών συνθηκών.

Η πλατφόρμα OPTIQUE παρέχει εργαλεία προκειμένου να πραγματοποιηθεί η σύνταξη, μετατροπή και εκτέλεση ερωτημάτων εκφρασμένων από τους χρήστες προς τις ΒΔ. Ένα από αυτά τα εργαλεία, το OPTIQUE_VQS, επιτρέπει το σχηματισμό SQL ερωτημάτων, τα οποία περιέχουν όρους της Οντολογίας[5], σε συνδυασμό με τα δεδομένα που ανήκουν στις λοιπές ΒΔ. Τα ερωτήματα αυτά αντιστοιχίζονται αυτόματα στις γλώσσες των πηγών, δηλαδή σε SPARQL, STARQL και GeoSPARQL. Για παράδειγμα, για την κλάση "Τουρμπίνα" η οποία ορίζεται στο λεξικό της Οντολογίας, υπάρχουν διάφορα SQL queries, οι απαντήσεις των οποίων αντιστοιχίζονται στην κλάση αυτή. Η μαθηματική σχέση του παραπάνω ορίζεται ως: $Τουρμπίνα\left(f(\vec{x})\right) \leftarrow \exists\, \vec{y}\, SQL(\vec{x}, \vec{y})$[6].

Μέσα από την OPTIQUE πλατφόρμα, ο έλεγχος των τουρμπινών πραγματοποιείται σε υψηλό επίπεδο, μέσα από ελάχιστα ερωτήματα της Οντολογίας Siemens, έναντι εκατοντάδων άλλων απευθείας προς τις ΒΔ. Αυτό δε σημαίνει ότι τα ερωτήματα εκφρασμένα στις άλλες γλώσσες εξαφανίζονται, αλλά ότι αποκρύπτονται από το χρήστη. Ένα σημαντικό πλεονέκτημα που προκύπτει από αυτήν τη μέθοδο, είναι το γεγονός ότι δεν απαιτείται γνώση από το χρήστη των διάφορων προγραμματιστικών γλωσσών που χαρακτηρίζουν τις πηγές, παρά μόνο της γλώσσας Οντολογίας αυτής καθαυτής (Kharlamov et al., 2016).

Οι **Patidar et al., 2020** υλοποίησαν ένα polystore σύστημα "mediator/wrapper" αρχιτεκτονικής, με σκοπό τη διαχείριση δεδομένων αστρονομικής φύσεως στο *σύννεφο* (**cloud**[7]). Εκεί "φιλοξενούνται"

[3] Ontology Based Data Access. Αρχιτεκτονική η οποία χαρακτηρίζεται από τρία συστατικά: α) Οντολογία (εννοιολογική περιγραφή του *πεδίου ενδιαφέροντος*), β) σύνολο πηγών δεδομένων και γ) αντιστοιχίσεις μεταξύ των δύο προηγούμενων όρων (Calvanese et al., 2017).

[4] Έως και 2000 αισθητήρες εγκατεστημένοι στα διάφορα τμήματα μίας τουρμπίνας.

[5] Διαθέτει ένα λεξικό στο οποίο ορίζονται: ονόματα κλάσεων και χαρακτηριστικά τους, δυαδικές σχέσεις μεταξύ τους κ.τ.λ. Στην περίπτωση της Siemens Οντολογίας, περιλαμβάνονται πληροφορίες για ηλεκτρικές συσκευές, υλικά, αισθητήρες και περιγραφές διαγνωστικών ενεργειών.

[6] Όπου SQL, αναφέρεται στο ομώνυμο ερώτημα και f είναι η συνάρτηση η οποία μετατρέπει τις πλειάδες (x,y) που επιστρέφει το query, σε αναγνωριστικά (identifiers) αντικειμένων που ανήκουν στην κλάση "Τουρμπίνα" της Οντολογίας SIEMENS.

[7] Απομακρυσμένη τεχνολογία αποθήκευσης δεδομένων, η οποία είναι προσβάσιμη μέσω του Διαδικτύου.

πηγές δεδομένων που ανήκουν στο Palomar Transient Factory (PTF), οι οποίες περιλαμβάνουν αστρονομικές μετρήσεις σχετικές με την αναγνώριση αντικειμένων (π.χ. κομήτες και αστεροειδείς) κοντά στην τροχιά της Γης. Οι μετρήσεις αυτές προέρχονται από τηλεσκόπια που φωτογραφίζουν τα ουράνια σωματίδια στο βάθος του χρόνου, έτσι ώστε οποιαδήποτε διακύμανση παρατηρηθεί στις διαστάσεις τους να καταγράφεται στις βάσεις δεδομένων. Οι φωτογραφίες αυτές αποτελούν τα ανεπεξέργαστα δεδομένα και αντιστοιχούν περίπου σε 4 εκατομμύρια φακέλους της μορφής ".fits" *(Flexible Image Transport System)* με ρυθμό ανανέωσης $4x10^4$ ειδοποιήσεις κάθε νύχτα, ωστόσο εκτός από αυτές περιέρχονται και άλλες δομές δεδομένων όπως κείμενο, δυαδικές τιμές, γράφοι και πίνακες. Η διαχείριση όλων των cloud δεδομένων σε πραγματικό χρόνο από ένα πιθανό DW σύστημα θεωρείται πολύπλοκη και αναμένεται να επιφέρει σημαντικά προβλήματα απόδοσης, ενώ τα κόστη μετατροπής ενός τόσο μεγάλου όγκου δεδομένων σε ένα ενιαίο μοντέλο μέσω της ETL διαδικασίας κρίνονται ασύμφορα.

Για το λόγο αυτό εγκαταστάθηκε μία τοπική σχεσιακή PostgreSQL ΒΔ, στην οποία αποθηκεύονται σε μορφή HTML τα μεταδεδομένα [8]των φωτογραφιών, συνδεόμενα με την URL διεύθυνση της αντίστοιχης φωτογραφίας στο σύννεφο. Το Polystore σύστημα που υλοποιήθηκε από τους συγγραφείς, περιλαμβάνει στο ανώτερο επίπεδο ένα γραφικό περιβάλλον χρήστη *(GUI[9])*, μέσω του οποίου οι χρήστες μπορούν να διαλέγουν αντικείμενα που τους ενδιαφέρουν και να ορίζουν στις παραμέτρους τους κάποια τιμή. Ανάλογα με την τιμή που εισάγεται, συντάσσεται αυτόματα από το σύστημα ένα SQL query προς την τοπική ΒΔ η οποία επιστρέφει την απάντηση σε μορφή πίνακα. Ο χρήστης έχει τη δυνατότητα να επιλέξει επιπρόσθετα αντικείμενα ενδιαφέροντος, τα οποία θα συνδεθούν μέσω Join εντολών με το αρχικό SQL ερώτημα. Στη συνέχεια προκειμένου να πραγματοποιηθεί η οπτικοποίηση του επιλεγμένου αντικειμένου, τα αποτελέσματα των SQL ερωτημάτων συσχετίζονται μέσω κλειδιών:τιμών (μεταδεδομένα εικόνας:URL link), με τις αντίστοιχες εικόνες που βρίσκονται στον cloud server. Οι εικόνες επιστρέφονται εκ νέου στο GUI και αναπαράγονται μέσω ειδικού προγράμματος οπτικοποίησης FITS αρχείων (Patidar et al., 2020).

Οι **Fink et al., 2020** αναφέρθηκαν στον όρο του *Schema Evolution* [10]σε ετερογενείς πηγές δεδομένων, και ειδικότερα παρουσίασαν μία προσέγγιση με σκοπό τη μετατροπή queries που ανήκουν σε ένα αρχικό polystore schema, σε ισοδύναμα εκφρασμένα σε ένα schema στόχο, μέσα από συγκεκριμένους τελεστές αλλαγής. Οι ερευνητές στηρίχθηκαν στις προϋπάρχουσες TyphonML και TyphonQL γλώσσες οι οποίες χρησιμοποιούνται συνδυαστικά. Η πρώτη αποτελεί μία modeling language[11] για την περιγραφή ενός υβριδικού polystore (δηλαδή του συνόλου ετερογενών πηγών δεδομένων), και ορίζει τις

[8] Δεδομένα περιγραφής άλλων δεδομένων. Διευκολύνουν την ανάκτηση και διαχείριση πληροφορίας των αντικειμένων.

[9] Graphical User Interface. Εφαρμογή με σκοπό την διευκόλυνση επικοινωνίας μεταξύ υπολογιστικής μηχανής και χρήστη.

[10] Ο όρος αναφέρεται στην ανάγκη για αναβάθμιση ενός αρχικού μοντέλου δεδομένων από source schema σε ένα target schema. Η αναβάθμιση δεν περιορίζεται μόνο στην αλλαγή του μοντέλου δεδομένων, αλλά επηρεάζει τα ίδια τα δεδομένα καθώς και τα ερωτήματα διαχείρισής τους.

[11] Γλώσσα η οποία ορίζει το schema μίας βάσης δεδομένων.

οντότητες, τα χαρακτηριστικά αυτών, αλλά και τις σχέσεις μεταξύ τους. Η TyphonQL αποτελεί την καθολική query γλώσσα του συνόλου των ΒΔ, καθώς υλοποιεί τις κατάλληλες αντιστοιχίσεις με τις δικές τους τοπικές γλώσσες.

Η καινοτομία των ερευνητών έγκειται στην εισαγωγή ενός συνόλου τελεστών αλλαγής που εισάγονται από τον χρήστη, και υπαγορεύουν τις αλλαγές του αρχικού polystore στο μοντέλο-στόχο, μεταβάλλοντας τις οντότητες, τα χαρακτηριστικά και τις σχέσεις μεταξύ τους. Με άλλα λόγια, έδωσαν λύση στο πώς να προσαρμόσουν την TyphonQL γλώσσα στο εξελιγμένο TyphonML μοντέλο. Για την επίτευξη του σκοπού αυτού, υλοποίησαν ένα εργαλείο με τη μορφή plugin του λογισμικού Eclipse, το οποίο δέχεται ως είσοδο το TyphonML schema, ένα σύνολο TyphonQL ερωτημάτων και ένα ακόμα σύνολο τελεστών αλλαγής. Ως αποτέλεσμα, παράγονται ισοδύναμα queries τα οποία είναι κατάλληλα για το schema-στόχο, ενώ σε περίπτωση αδυναμίας εμφανίζεται αντίστοιχο μήνυμα περιγραφής του λάθους (Fink et al., 2020).

Οι **Leclercq & Savonnet, 2018** παρουσίασαν ένα θεωρητικό μοντέλο με την ονομασία Tensor Data Model (TDM), στόχος του οποίου είναι η γρήγορη τροφοδότηση αλγορίθμων ανάλυσης με δεδομένα διακριτών πηγών, επιτυγχάνοντας παράλληλα όσο το δυνατόν λιγότερους μετασχηματισμούς. Το μοντέλο αυτό στηρίζεται στην έννοια του τανυστή [12], για τον οποίο κάθε διάσταση αναπαρίσταται ως συσχετιζόμενος πίνακας, δηλαδή ως πίνακας όπου αντιστοιχίζει κλειδιά σε συγκεκριμένες τιμές.

Το TDM μοντέλο εφαρμόστηκε σε ένα polystore σύστημα αποτελούμενο από PostgreSQL, HDFS, Neo4j βάσεις με διαφορετικές μορφές δεδομένων τα οποία προέρχονται από την πλατφόρμα Twitter, όπου ανιχνεύθηκε η πιθανή συμμετοχή λογαριασμών λογισμικού, ή bots, στην κυκλοφορία viral tweets. Για το σκοπό αυτό, υλοποιήθηκε ένας tensor 3 διαστάσεων (Λογαριασμοί Twitter, Hashtags, Χρονική Περίοδος), ο οποίος περιέχει ως τιμές το γινόμενο των τιμών των προηγούμενων διαστάσεων (1.077 × 568 × 336). Κατά την παραπάνω top-to-bottom προσέγγιση (από τον τανυστή προς τις ΒΔ), ο τανυστής υπαγορεύει queries στους wrappers. οι οποίοι με τη σειρά τους τα μετατρέπουν σε τοπικά queries της αντίστοιχης πηγής δεδομένων. Ως αποτέλεσμα, οι τιμές του tensor "φορτώνονται" με τα αντίστοιχα δεδομένα. Τέλος, μέσω ειδικών tensor τελεστών (π.χ. selection, projection, union, intersection, join, group by και tensor decomposition), γίνονται μετασχηματισμοί στις τιμές του, οι οποίοι οδηγούν στην ευρύτερη επεξεργασία τους. Με άλλα λόγια, ο τανυστής δρα ως ένας pivot [13]πολυδιάστατος πίνακας μεταξύ διαφορετικών μοντέλων δεδομένων (Leclercq & Savonnet, 2018).

Οι **Vogt et al., 2018** υλοποίησαν ένα κατανεμημένο cloud polystore σύστημα με την ονομασία Polypheny-DB. Η καινοτομία αυτού του συστήματος αφορά την ενσωμάτωση της διαδικασίας *κατάτμησης*

[12] Ή Tensor. Πρόκειται για μαθηματικό στοιχείο αποτελούμενο από Ν διαστάσεις, κάθε μία από τις οποίες χρησιμοποιείται για την αποθήκευση ενός συνόλου τιμών. Για παράδειγμα, ένας Tensor με *N = 0* ονομάζεται ως Scalar, με *N=1* ως Vector και με *N=2* ως Matrix, ενώ με Ν >=3 ως Higher Order.

[13] Δύναται να περιστραφεί.

και αντιγραφής των δεδομένων στο σύννεφο, με αυτήν της παροχής *τοπικών polystore* λύσεων. Η κατάτμηση και αντιγραφή των δεδομένων αφορά τον διαμοιρασμό αυτών κατά μήκος διαφορετικών περιοχών/data centers στο σύννεφο, με τέτοιο τρόπο έτσι ώστε αυτά να είναι προσβάσιμα από όλους του χρήστες των cloud υπηρεσιών. Αντίστοιχα, η υλοποίηση τοπικών polystore μοντέλων σε cloud κόμβους, αντιμετωπίζει το πρόβλημα της ποικιλομορφίας των Big Data.

Σε "global" επίπεδο το Polypheny-DB αποτελείται από ένα σύμπλεγμα data centers ίδιας σημαντικότητας, όπου στόχος είναι ο ισομερής καταμερισμός των δεδομένων σε αυτά. Ζητούμενο είναι να επιτευχθεί η βέλτιστη αξιοποίηση αποθηκευτικού χώρου μεταξύ των κέντρων δεδομένων που βρίσκονται στο σύννεφο, σε συνδυασμό με την ελαχιστοποίηση της δικτυακής απόστασης *(network distance)* αυτών με τους χρήστες, η οποία θα οδηγήσει σε μικρούς χρόνους ανταπόκρισης *(response time)* για την πρόσβαση στα δεδομένα. Σχετικά με τα polystores που βρίσκονται σε τοπικό επίπεδο, παρέχεται από το υπό μελέτη σύστημα ένα εύρος διαφορετικών query interfaces (π.χ.SQL, CRUD [14]κ.α.) προκειμένου κάθε ένα από αυτά να δίνει πρόσβαση σε όλες τις διαθέσιμες πηγές δεδομένων μέσω πινάκων αντιστοίχισης. Οι δύο προαναφερόμενες λειτουργίες ανέκαθεν μπορούσαν να υλοποιηθούν ξεχωριστά, ωστόσο πριν την δημιουργία του Polypheny-DB δεν υπήρχε κάποια αρχιτεκτονική που να τις συνδυάζει. Μία ακόμα καινοτομία σχετίζεται με την αυτονομία του συστήματος, καθώς αυτό είναι σε θέση να προσαρμόζει τις παραμέτρους του ανάλογα με τις απαιτήσεις που προκύπτουν, δηλαδή για παράδειγμα να συνδεθεί ή αποσυνδεθεί αυτοβούλως από κάποια αποθήκη δεδομένων (Vogt et al., 2018).

Οι **Alotaibi et al., 2020** ανέπτυξαν ένα σύστημα με την ονομασία ESTOCADA, το οποίο μπορεί να εφαρμοστεί σε polystore αρχιτεκτονικές με σκοπό να αντιμετωπίσει δύο πιθανούς περιορισμούς που προκύπτουν. Αρχικά, συχνά τα ίδια δεδομένα μπορεί να βρίσκονται αποθηκευμένα σε διαφορετικές πηγές, με αποτέλεσμα το polystore σύστημα να μην επιλέγει τη βέλτιστη ΒΔ για την επεξεργασία ενός συγκεκριμένου query. Άλλωστε, κάθε μηχανή αποθήκευσης *(storage engine[15])* επεξεργάζεται με διαφορετικό τρόπο παρόμοια queries. Επιπλέον, τα polystore συστήματα δεν εκμεταλλεύονται προϋπολογισμένα αποτελέσματα ερωτημάτων, τα οποία πιθανώς είναι διαθέσιμα από τις ίδιες τις πηγές με τη μορφή όψης *(materialized view[16])*.

Το ESTOCADA, υιοθετεί την local-as-view *(LAV)* αρχιτεκτονική, σύμφωνα με την οποία το περιεχόμενο κάθε πηγής δεδομένων περιγράφεται ως μία "όψη". Το σύστημα λαμβάνει υπόψιν του όλους τους πιθανούς συνδυασμούς ΒΔ, κάνοντας "materialize" την κατάλληλη όψη που προέρχεται από την βέλτιστη πηγή δεδομένων και ανάλογα με το ερώτημα που τίθεται από τους χρήστες. Ως αποτέλεσμα,

[14] (Create, Read, Update, Delete). Με την έννοια των διεπαφών που υποστηρίζουν τις λειτουργίες δημιουργίας, ανάγνωσης, ενημέρωσης και διαγραφής δεδομένων.

[15] Λογισμικό το οποίο βρίσκεται ενσωματωμένο σε κάθε σύστημα διαχείρισης βάσεων δεδομένων και εκτελεί CRUD διεργασίες στα δεδομένα.

[16] Στατική δομή η οποία επιτρέπει την αποθήκευση του αποτελέσματος ενός query, προκειμένου αυτό να μην υπολογίζεται κάθε φορά για το ίδιο ερώτημα.

παρατηρείται σημαντική βελτίωση της απόδοσης των queries για τα polystore συστήματα (Alotaibi et al., 2020).

Συνοπτικά, μερικά από τα δημοφιλέστερα polystore συστήματα είναι το CloudMdsQL, το Myria, και το BigDAWG, με βάση το οποίο θα εκπονηθεί η υλοποίηση της παρούσας διπλωματικής εργασίας. Όσον αφορά το CloudMdsQL, αυτό παρέχει πρόσβαση σε ετερογενείς μη-σχεσιακές ΒΔ, μέσα από μία γλώσσα βασισμένη σε SQL, η οποία φιλοξενείται από ένα query interface διαμοιρασμένο σε cloud περιβάλλον. Η επικοινωνία με τις τοπικές ΒΔ επιτυγχάνεται μέσα από subqueries εκφρασμένα στην αντίστοιχη γλώσσα κάθε πηγής, αλλά και wrappers σε γλώσσες Python και Spark (Bondiombouy et al., 2016). Το Myria polystore είναι ένα σύστημα ανάλυσης δεδομένων το οποίο παρέχεται ως υπηρεσία μέσω cloud, και χρησιμοποιεί ως γλώσσα για την επικοινωνία με τις τοπικές ΒΔ τη MyriaL. Μέσω του συστήματος RACO *(Relational Algebra Compiler)* το οποίο χρησιμοποιεί έννοιες σχεσιακής άλγεβρας, τα queries εκφρασμένα σε γλώσσα MyriaL αντιστοιχίζονται στις αποθήκες δεδομένων (Wang et al., 2017).

3

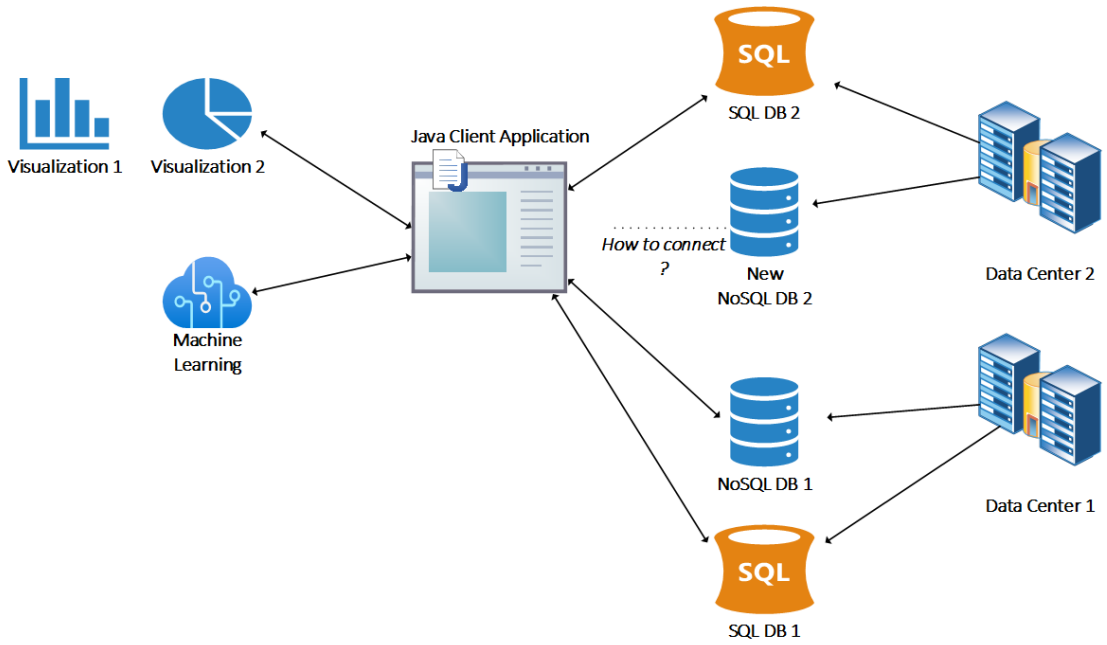
Θεωρητικό υπόβαθρο

Σε αυτό το κεφάλαιο εμπεριέχεται η ανασκόπηση της θεωρίας των polystore συστημάτων και η επισήμανση ορισμένων εννοιών που παρουσιάστηκαν και στο προηγούμενο κεφάλαιο *(Κεφάλαιο 2 – Σχετικές εργασίες)*, έτσι ώστε να γίνει πλήρως κατανοητό το βασικό μοντέλο στο οποίο στηρίζεται η παρούσα εργασία. Έπειτα, παρουσιάζεται η ανάλυση μίας εφαρμογής διαχείρισης ετερογενών βάσεων δεδομένων με την ονομασία *BigDAWG (the Big Data Working Group)*, η οποία αφορά ένα polystore σύστημα ανοικτού κώδικα. Η ανάλυση αυτή είναι απαραίτητη καθώς κατά το στάδιο της υλοποίησης, θα πραγματοποιηθούν κατάλληλες παραμετροποιήσεις στον κώδικα της εφαρμογής με σκοπό να εισαχθούν νέες βάσεις δεδομένων οι οποίες θα αναπαραστήσουν ένα έξυπνο δίκτυο (ακολουθεί περαιτέρω ανάλυση στο Κεφάλαιο 4). Στην τρίτη ενότητα του κεφαλαίου αυτού παρουσιάζεται η εφαρμογή *docker*, η οποία αποτελεί απαραίτητο εργαλείο για την ορθή εγκατάσταση του BigDAWG συστήματος, ενώ στην τελευταία ενότητα παρουσιάζονται σύντομα τα τρία διαφορετικά συστήματα διαχείρισης βάσεων δεδομένων *(PostgreSQL, SciDB και Apache Accumulo)*, τα οποία και πρόκειται να αξιοποιηθούν στο επόμενο κεφάλαιο.

3.1 Τα μοντέλα των polystore συστημάτων

Όπως έχει προαναφερθεί, τα polystore συστήματα ανήκουν σε μια από τις τέσσερις κατηγορίες ενοποιημένων συστημάτων διαχείρισης βάσεων δεδομένων, και ειδικότερα παρέχουν μία ενιαία οπτική σε παραπάνω από μία ετερογενείς ΒΔ. Το γενικό πρόβλημα στο οποίο δίνουν λύση, αφορά τον τρόπο με τον οποίο μία εφαρμογή μπορεί να διαχειριστεί πολλαπλές και ετερογενείς πηγές δεδομένων (π.χ. σύνδεση και επικοινωνία με αυτές), να επεξεργαστεί τα ενδιάμεσα αποτελέσματα που προκύπτουν από τα διάφορα queries προς αυτές (π.χ. σύνθεση subqueries από πολλαπλές ΒΔ σε ένα τελικό query) και τέλος, να διαμοιράσει τα αποτελέσματα στους τελικούς χρήστες. Επιπλέον, πριν την έλευση των polystores, όσα περίπλοκα συστήματα είχαν υλοποιηθεί για να υποστηρίξουν εν μέρει τέτοιες διαδικασίες,

αντιμετώπιζαν δυσκολίες στην επέκτασής τους και δημιουργούσαν επιπλέον κόστη. Η επέκταση των συστημάτων αυτών μεταφράζεται ως προσθήκη νέων μηχανών αποθήκευσης όπου και απαιτούνται νέες συνδέσεις. Το παραπάνω πρόβλημα αποτυπώνεται στην επόμενη εικόνα.



Εικόνα 4: Το πρόβλημα σύνδεσης ετερογενών πηγών δεδομένων μέσα από μία εφαρμογή (π.χ. σε γλώσσα Java).

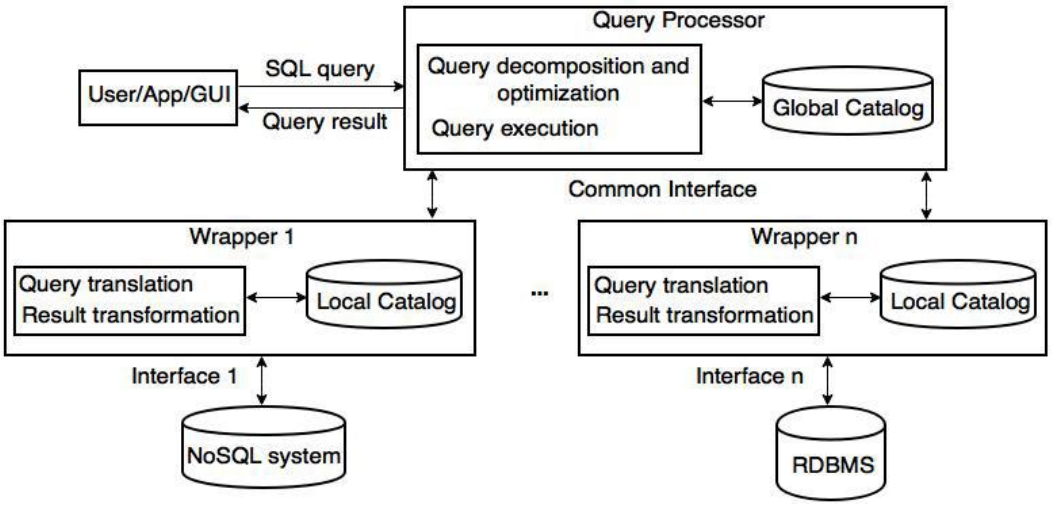
Η επεξεργασία των ετερογενών πηγών δεδομένων μέσω των polystores μπορεί να διαφέρει, καθώς υπάρχουν διαφορετικές αρχιτεκτονικές για την υλοποίησή τους. Τα συστήματα αυτά διακρίνονται σε τρείς υποκατηγορίες ανάλογα με τον βαθμό σύζευξής τους με τις βάσεις δεδομένων, και συγκεκριμένα χαρακτηρίζονται ως: loosely-coupled, tightly-coupled και hybrid systems (Bondiombouy & Valduriez, 2016). Παρακάτω, πραγματοποιείται αναλυτικά η περιγραφή των επιμέρους συστημάτων:

Loosely Coupled Systems

Τα συστήματα αυτά ακολουθούν την αρχιτεκτονική "mediator/wrapper" και μπορούν να αλληλοεπιδράσουν με ένα μεγάλο αριθμό ετερογενών βάσεων δεδομένων. Χαρακτηριστικό αυτών των ΒΔ είναι ότι διατηρούν την αυτονομία τους, δηλαδή την ικανότητα να ελέγχονται και τοπικά, δεδομένου ότι υπόκεινται σε επεξεργασία τόσο σε τοπικό επίπεδο μέσα από το API τους, όσο και από την διεπαφή του polystore συστήματος. Για την υλοποίηση της αρχιτεκτονικής "mediator/wrapper", στο σύστημα υπάρχει ένας κύριος επεξεργαστής ερωτημάτων ("mediator") ο οποίος συνδέεται με τον αντίστοιχο "wrapper"/διεπαφή της κάθε βάσης δεδομένων. Ο ρόλος του "wrapper" είναι να μεσολαβήσει για την επικοινωνία μεταξύ του κεντρικού interface και των τοπικών ΒΔ. Αξίζει να σημειωθεί ότι τόσο ο

"mediator" όσο και οι "wrappers" διαθέτουν ένα είδος καταλόγου το οποίο περιλαμβάνει πληροφορίες για τις οντότητες (π.χ. βάσεις δεδομένων, πίνακες) με τις οποίες συνδέεται κάθε ένα από αυτά.

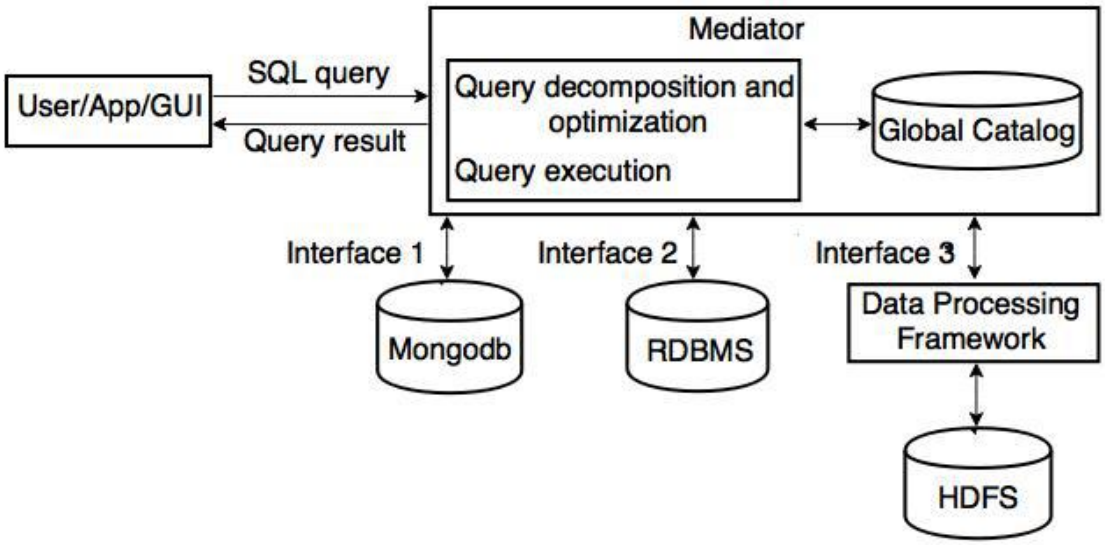
Κατά τη διαδικασία εκτέλεσης ενός ερωτήματος από το interface του polystore, αυτό επεξεργάζεται και μεταφράζεται σε υποερωτήματα προς τις ΒΔ στην αντίστοιχη τοπική γλώσσα. Τα υποερωτήματα αυτά μεταφέρονται μέσω των "wrappers" (σ.σ. από τους οποίους έχει γίνει και η μετάφραση που προαναφέρθηκε) και από εκεί εκτελούνται τοπικά, ενώ τα ενδιάμεσα αποτελέσματα που προκύπτουν θα μεταφραστούν εκ νέου, αυτή τη φορά στην κοινή γλώσσα του polystore συστήματος. Το τελευταίο στάδιο αφορά τη σύνθεση (π.χ. με join ή union διεργασίες) και επιστροφή προς το χρήστη, του τελικού αποτελέσματος, το οποίο προκύπτει από τις επιμέρους απαντήσεις που συλλέχθηκαν.



Εικόνα 5: Η δομή ενός Loosely-Coupled συστήματος (Bondiombouy & Valduriez, 2016).

Tightly Coupled Systems

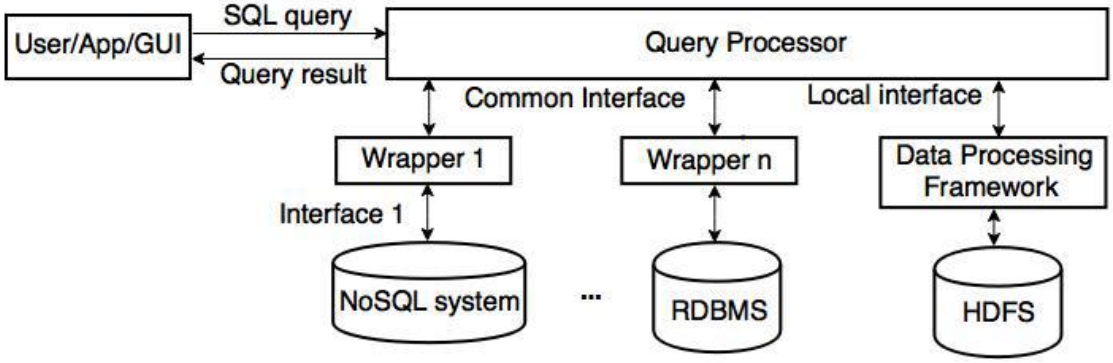
Σε αντίθεση με τα loosely-coupled συστήματα τα οποία προσφέρουν αυτονομία στις τοπικές ΒΔ, στα tightly-coupled συστήματα η αυτονομία θυσιάζεται προκειμένου να επιτευχθεί η βέλτιστη απόδοση της querying διαδικασίας. Η βέλτιστη απόδοση μεταφράζεται ως υψηλή ταχύτητα εκμαίευσης των απαντήσεων και μάλιστα σε πραγματικό χρόνο. Για το σκοπό αυτό η σύνδεση του polystore "mediator" με τις πηγές δεδομένων επιτυγχάνεται ***απευθείας*** μέσα από τα API τους, χωρίς να παρεμβάλλεται κάποιο "wrapper" λογισμικό. Με δεδομένο ότι το ενδιάμεσο λογισμικό του polystore έχει απευθείας πρόσβαση στις διακριτές βάσεις δεδομένων, μπορεί εύκολα να πραγματοποιηθεί η μετακίνηση των δεδομένων από τη μία βάση στην άλλη. Σημαντική παράμετρος αυτής της κατηγορίας polystore μοντέλου αποτελεί και το γεγονός ότι ο αριθμός των βάσεων που μπορεί να φιλοξενήσει είναι σχετικά μικρός.



Εικόνα 6: Η δομή ενός Tightly-Coupled συστήματος (Bondiombouy & Valduriez, 2016).

Hybrid Systems

Τέλος, τα υβριδικά polystore συστήματα είναι σχεδιασμένα με τέτοιο τρόπο ώστε να συνδυάζουν τα πλεονεκτήματα των δύο προηγούμενων τύπων. Γίνεται η προσπάθεια, αφενός να ενσωματώσουν όσο το δυνατόν περισσότερες βάσεις δεδομένων και αφετέρου να εξασφαλίσουν απευθείας σύνδεση με ένα μέρος αυτών, μέσω των APIs που διαθέτει κάθε μία πηγή ώστε να επιτευχθεί και η βέλτιστη απόδοση. Οι υπόλοιπες πηγές δεδομένων προσπελαύνονται μέσω των "wrappers" ακολουθώντας την κλασική "mediator/wrapper" αρχιτεκτονική. Το σύστημα BigDAWG που παρουσιάζεται στην επόμενο ενότητα αποτελεί παράδειγμα ενός υβριδικού polystore.



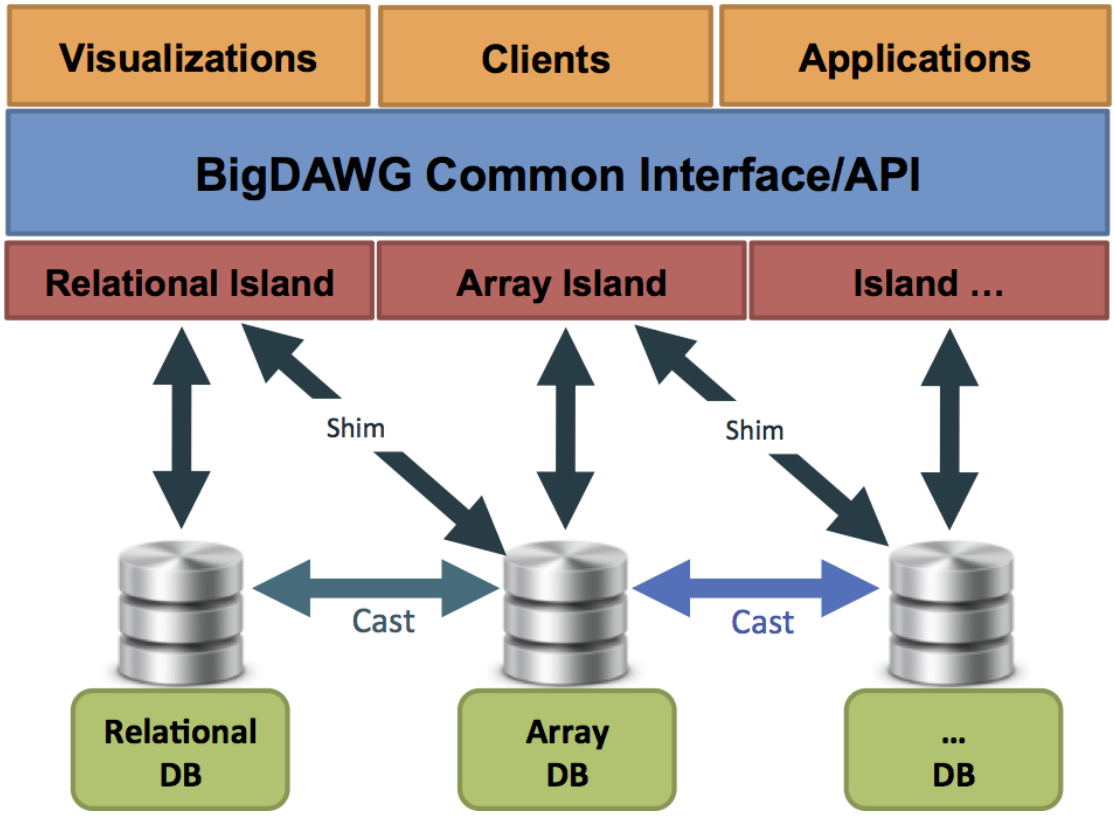
Εικόνα 7: Η δομή ενός hybrid polystore συστήματος (Bondiombouy & Valduriez, 2016).

3.2 Το περιβάλλον ενός polystore συστήματος μέσα από την εφαρμογή BigDAWG

Η εφαρμογή BigDAWG είναι το αποτέλεσμα μίας συλλογικής προσπάθειας στην οποία συμμετείχαν αρκετά πανεπιστημιακά ιδρύματα αλλά και οργανισμοί. Οι δύο μεγάλες επιτυχίες αυτού του εγχειρήματος σχετίζονται με το γεγονός ότι το σύστημα αναγνωρίζει αυτόματα σε ποια μηχανή αποθήκευσης απευθύνεται ένα διατυπωμένο query, ενώ συγχρόνως επιτυγχάνεται η πλήρης εκμετάλλευση των δυνατοτήτων κάθε μηχανής (Duggan et al., 2015).

Ιδιαίτερο ενδιαφέρον παρουσιάζει η αρχιτεκτονική του συστήματος, όπως παρουσιάζεται στην εικόνα που ακολουθεί (*Εικόνα 8*). Πιο συγκεκριμένα, οι ετερογενείς μηχανές αποθήκευσης δεδομένων που περιλαμβάνονται στο κατώτερο επίπεδο οργανώνονται σε δομές με την ονομασία *island* ή αλλιώς νησί. Η κατανομή καθεμίας από αυτές σε ένα island αποτελεί την ειδοποιό διαφορά μεταξύ ενός polystore συστήματος και ενός federated DBMS, υπό την έννοια ότι σε αντίθεση με την τελευταία κατηγορία όπου κάθε engine είναι ανεξάρτητο από το άλλο, στην περίπτωση των polystore αυτά επεξεργάζονται ως ενιαία σύνολα – *"the whole is greater than the sum of their parts"* (Mattson et al., 2017). Η σύνδεση ενός island με μία ή περισσότερες μηχανές αποθήκευσης επιτυγχάνεται μέσα από μία άλλη οντότητα που χαρακτηρίζεται ως *shim* και ο ρόλος της αφορά την αντιστοίχιση των queries μεταξύ ενός island και των σχετιζόμενων με αυτά storage engines.

Επιπρόσθετα, παρέχεται και η δυνατότητα μεταφοράς (*migration*) των δεδομένων μεταξύ των μηχανών αποθήκευσης μέσα από *cast* διεργασίες, δεδομένου ότι κάθε storage engine έχει διαφορετική απόδοση σε σχέση με ένα ερώτημα που αφορά συγκεκριμένο τύπο δεδομένων. Πρακτικά, το BigDAWG αποτελεί το ενδιάμεσο API λογισμικό το οποίο είναι υπεύθυνο για τη διευθέτηση των λειτουργιών μεταξύ των επιμέρους στοιχείων (islands, shims casts και storage engines) του polystore. Μάλιστα, στο API αυτό μπορούν να συνδεθούν σε υψηλότερο επίπεδο άλλες εφαρμογές τελικών χρηστών, οι οποίες αντλούν δεδομένα για την κάλυψη αναγκών, όπως για παράδειγμα η οπτικοποίηση των δεδομένων αλλά και η περαιτέρω ανάλυσή τους μέσα από αλγορίθμους μηχανικής μάθησης.



Εικόνα 8: Η αρχιτεκτονική του BigDAWG Polystore συστήματος (Gadepally et al., 2016).

3.2.1 Τεχνική ανάλυση επιμέρους στοιχείων του BigDAWG συμπλέγματος

Island

Κάθε island διαθέτει ένα εννοιολογικό[17] σύνολο πληροφορίας το οποίο επιτρέπει την αντιστοίχιση με τις εκάστοτε βάσεις δεδομένων. Η πληροφορία αυτή περιλαμβάνει:

I. Γενικές πληροφορίες για το μοντέλο δεδομένων (π.χ. schema και σχέσεις ξένων κλειδιών).
II. Γλώσσα εκμαίευσης δεδομένων ή σύνολο τελεστών. Και οι δύο περιπτώσεις πρέπει να είναι σύμφωνες με το μοντέλο δεδομένων.
III. Μία ή περισσότερες μηχανές αποθήκευσης στις οποίες απευθύνονται τα ερωτήματα των χρηστών. Κάθε μηχανή μπορεί να περιέχει περισσότερες από μία βάσεις δεδομένων.

Όσον αφορά την κατηγοριοποίηση των νησιών, συναντώνται δύο κύριες κατηγορίες. Η πρώτη αφορά τα *εκφυλισμένα νησιά (degenerated islands)*, τα οποία υιοθετούν πλήρως τις εννοιολογικές πληροφορίες ενός storage engine, ενώ στη δεύτερη κατηγορία ανήκουν αυτά που δεν ακολουθούν ξεκάθαρα τη λογική κάποιας μηχανή, και ονομάζονται *ψηφιακά νησιά (virtual islands)*.

[17] Που σχετίζεται με ένα σύνολο παραμέτρων, εννοιών και καταστάσεων οι οποίες χαρακτηρίζουν μία οντότητα.

Όταν ένα island δέχεται ένα ερώτημα, τότε δημιουργεί μία απεικόνιση του ερωτήματος αυτού υπό τη μορφή ενός δέντρου. Τα κλαδιά του δέντρου απεικονίζουν υποερωτήματα του αρχικού query, τα οποία μέσω των shims μεταφράζονται στην τοπική γλώσσα της βάσης δεδομένων για την οποία προορίζονται. Όταν επιστραφούν τα αποτελέσματα των subqueries, το νησί έχει την ευθύνη να συνθέσει τις απαντήσεις αυτές σε μία τελική απάντηση, η οποία και θα επιστραφεί προς τον τελικό χρήστη. Προς το παρόν, το BigDAWG περιλαμβάνει νησιά που μπορούν να υποστηρίξουν τη σύνδεση με σχεσιακές ΒΔ, πίνακες, κείμενα (σε μορφή ζευγαριών key:value) και APIs, ενώ ως επί το πλείστον ανήκουν στην κατηγορία των degenerated islands.

Storage Engines

Στα πλαίσια της ετερογένειας, το BigDAWG polystore δε θα μπορούσε να μην υποστηρίζει διαφορετικές μηχανές αποθήκευσης. Όπως έχει προαναφερθεί, παρέχεται τοπική ανεξαρτησία στα διάφορα storage engines μέσω της κατηγοριοποίησής τους σε αντίστοιχα islands, ωστόσο μία μηχανή μπορεί να ανήκει σε παραπάνω από ένα νησιά και εκ του αποτελέσματος να υποστεί επεξεργασία μέσα από διαφορετικούς τύπους ερωτημάτων του χρήστη.

Η τελευταία έκδοση του BigDAWG παρέχει τρείς υλοποιημένες διακριτές μηχανές αποθήκευσης, στις οποίες έχουν εκχωρηθεί δεδομένα με σκοπό τον πειραματισμό και την εξοικείωση των χρηστών με το περιβάλλον της εφαρμογής. Τα δεδομένα προέρχονται από ένα σετ με την ονομασία “MIMIC II”, το οποίο περιλαμβάνει πληροφορίες σχετικά με 26000 εισαγωγές ασθενών στη μονάδα εντατικής θεραπείας της κλινικής Boston’s Beth Israel Deaconess Hospital σε περίοδο 8 ετών. Πρόκειται για δεδομένα τα οποία εξασφαλίζουν την ανωνυμία των ασθενών, ενώ η μορφή τους θεωρείται ιδανική για την αποθήκευσή τους στο περιβάλλον ενός polystore συστήματος καθώς περιέχουν δομημένα δεδομένα (π.χ. πληροφορίες ασθενών), αδόμητα (π.χ. ιατρικά reports σε μορφή ελεύθερου κειμένου) και χρονοσειρές κυματομορφών (π.χ. οι κυματομορφές σημάτων που παράγονται από έναν ηλεκτροκαρδιογράφο).

Πιο συγκεκριμένα, στο υπό μελέτη σύστημα περιλαμβάνονται οι ακόλουθες μηχανές αποθήκευσης:

I. Σχεσιακή βάση δεδομένων σε γλώσσες PostgreSQL και Myria που υποστηρίζουν τα δομημένα δεδομένα του MIMIC II dataset.
II. Αποθήκη δεδομένων βασισμένη στο Apache Accumulo για την αποθήκευση ελεύθερου κειμένου στη μορφή “key:value”.
III. Αποθήκη δεδομένων SciDB, η οποία φιλοξενεί δεδομένα υπό τη μορφή πίνακα. Στη συγκεκριμένη περίπτωση, αποθηκεύονται οι χρονοσειρές κυματομορφών του dataset.

Η θεωρητική ανάλυση κάθε μίας από τις προαναφερόμενες μηχανές αποθήκευσης περιλαμβάνεται στην ενότητα 3.4.

Shim

Πρόκειται για την δομή που επιτρέπει τη σύνδεση μεταξύ νησιού και βάσης δεδομένων. Σε τεχνικό επίπεδο, ένα shim ή και περισσότερα "καλούνται" από κάποιο island προκειμένου να μεταφράσουν τα subqueries που έχουν παραχθεί από το νησί στην αντίστοιχη γλώσσα της βάσης-στόχου. Για τον σκοπό αυτό, κάθε shim αποτελείται από τα εξής εργαλεία:

I. Query parser, ώστε να "δέχεται" ερωτήματα από ένα νησί.
II. Query generator, προκειμένου να μεταφράζει ένα ερώτημα.
III. Query dispatcher, το οποίο είναι υπεύθυνο για την αποστολή ενός ερωτήματος στη βάση που προορίζεται.

Για την υλοποίηση ενός shim στο περιβάλλον του BigDAWG είναι απαραίτητη η σύνδεση του με το αντίστοιχο JDBC[18] driver κάθε μηχανής αποθήκευσης. Για παράδειγμα, το shim που χρησιμοποιείται ως γέφυρα μεταξύ του σχεσιακού νησιού και της σχεσιακής βάσης δεδομένων βασίζεται στον PostgreSQL JDBC driver.

Cast

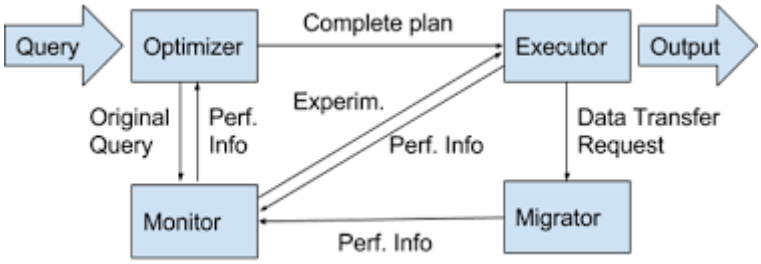
Αναφέρεται στη διαδικασία κατά την οποία δεδομένα από ένα storage engine μεταφέρονται σε ένα άλλο. Η λειτουργία αυτή διευκολύνει περιπτώσεις κατά τις οποίες μία μηχανή αποθήκευσης κρίνεται καταλληλότερη για την επεξεργασία των δεδομένων εκείνων που ανήκουν σε μία διαφορετική. Η ίδια τεχνική μπορεί να επιτευχθεί και μεταξύ των νησιών, δηλαδή όταν μεταφέρονται αποτελέσματα των subqueries από ένα νησί σε ένα άλλο. Οι cast διεργασίες είναι ιδιαίτερα χρήσιμες, καθώς όταν κάποιο island αδυνατεί να υποστηρίξει ένα ερώτημα τότε η εφαρμογή του BigDAWG θα μεταφέρει δεδομένα μεταξύ των storage engines ή απαντήσεις υποερωτημάτων μεταξύ των νησιών.

BigDAWG API

Το ενδιάμεσο λογισμικό δρα ως ο συνδετικός κρίκος μεταξύ των οντοτήτων που έχουν αναφερθεί έως τώρα και συνθέτουν το polystore σύστημα. Ο σκοπός του επιτυγχάνεται μέσα από τα παρακάτω τέσσερα εργαλεία:

I. *Planner/Optimizer*: Δέχεται το εισερχόμενο query, το οποίο και κατακερματίζει σε μικρότερα subqueries με στόχο πιθανά storage engines του συστήματος. Η δομή σύμφωνα με την οποία οργανώνεται το αρχικό query μαζί με τα subqueries έχει τη μορφή ενός δέντρου. Πιο συγκεκριμένα δημιουργούνται αρκετές εκδόσεις δέντρων, κάθε μία από τις οποίες έχει το ρόλο του οδηγού για τον τρόπο με τον οποίο θα εκτελεστεί το ερώτημα. Με άλλα λόγια, ένα δέντρο αποτελεί μία αναπαράσταση του πλάνου εφαρμογής ενός query, περιλαμβάνοντας τα ακριβή βήματα εκτέλεσής του.
II. *Monitor*: Χρησιμοποιεί τη γνώση από παρελθοντικά ή και παρεμφερή queries με σκοπό να επιλέξει το αποδοτικότερο δέντρο.
III. *Executor*: Επωμίζεται την ευθύνη να εκτελέσει το ερώτημα.
IV. *Migrator*: Ενεργοποιείται σε περιπτώσεις όπου κρίνεται απαραίτητη η μεταφορά δεδομένων μεταξύ των μηχανών αποθήκευσης, όταν αυτό ορίζεται από το πλάνο εκτέλεσης του ερωτήματος.

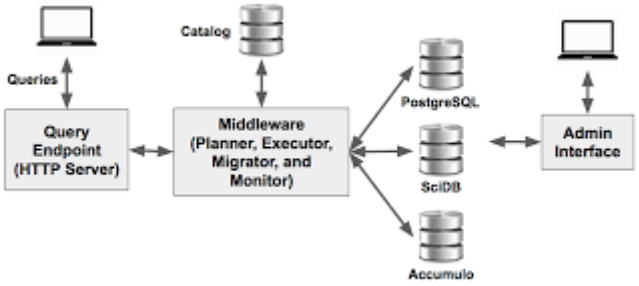
[18] Java Database Connectivity. Λογισμικό που επιτρέπει τη σύνδεση ενός προγράμματος γραμμένο σε γλώσσα Java με μία βάση δεδομένων.



Εικόνα 9: Συστατικά στοιχεία του ενδιάμεσου λογισμικού BigDAWG (Gadepally et al., 2016).

Catalog
Ο κατάλογος αφορά μία ειδική κατηγορία σχεσιακής βάσης δεδομένων, η οποία υποστηρίζεται από τη γλώσσα PostgreSQL. Η βάση αυτή έχει το ρόλο μίας αποθήκης μεταδεδομένων, καθώς σε αυτή περιλαμβάνονται πληροφορίες για όλες τις οντότητες του BigDAWG συμπλέγματος (*island, shim, storage engine, casts*). Συγχρόνως, περιέχονται και οι κατάλληλες αντιστοιχίσεις μεταξύ των μηχανών αποθήκευσης και των δεδομένων τους. Η χρησιμότητα του καταλόγου είναι ύψιστης σημασίας, με δεδομένο ότι ενδεχόμενη απουσία του "κρύβει" την απαιτούμενη πληροφορία από το ενδιάμεσο λογισμικό του BigDAWG σχετικά με την ύπαρξη των στοιχείων τα οποία πρέπει και να συντονίσει.

Query Endpoint (HTTP Server)
Αφορά έναν HTTP εξυπηρετητή[19] ο οποίος τροφοδοτείται με αιτήματα υπό τη μορφή ενός query. Ο ρόλος του σχετίζεται με την παραλαβή του ερωτήματος, τη μεταφορά αυτού στο λογισμικό-μεσάζοντα BigDAWG, και τελικά την επιστροφή των αποτελεσμάτων πίσω στο χρήστη.



Εικόνα 10: Η πορεία ενός ερωτήματος από και προς τον χρήστη μέσα από την εφαρμογή του BigDAWG (Gadepally et al., 2016).

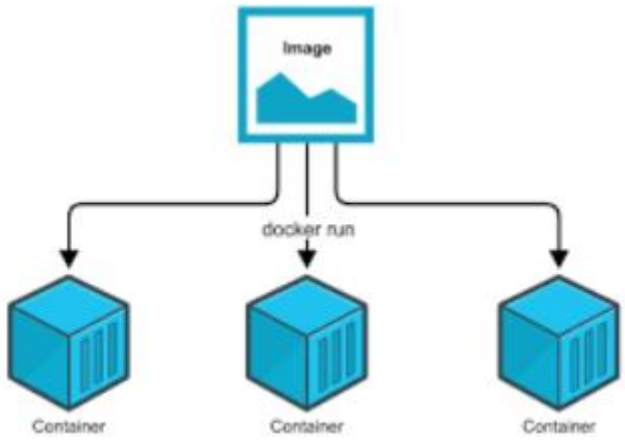
3.3 Το λογισμικό Docker

Το Docker αποτελεί ένα προγραμματιστικό εργαλείο το οποίο επιτρέπει τόσο την εγκατάσταση του BigDAWG polystore όσο και την αλληλεπίδραση με αυτό. Η πλατφόρμα του Docker διευκολύνει την ανάπτυξη και έκδοση εφαρμογών, μέσα από την μετατροπή τους σε πακέτα και την εκχώρησή τους σε οντότητες με την ονομασία "κοντέινερ" (*"container"*). Ειδικότερα, ο όρος κοντέινερ αναφέρεται σε ένα απομονωμένο και ολοκληρωμένο εικονικό περιβάλλον το οποίο επιτρέπει την εκτέλεση μίας εφαρμογής ανεξάρτητα με το λειτουργικό σύστημα το οποίο την φιλοξενεί. Το τελευταίο χαρακτηριστικό

[19] Λογισμικό το οποίο ακολουθεί ένα συγκεκριμένο πρωτόκολλο επικοινωνίας (σε αυτή την περίπτωση το πρωτόκολλο HTTP) προκειμένου να εξυπηρετήσει τα αιτήματα ενός πελάτη-χρήστη.

που περιεγράφηκε, δηλαδή η εκτέλεση της εφαρμογής σε οποιοδήποτε λειτουργικό σύστημα οφείλεται στο γεγονός ότι ένα κοντέινερ περιέχει εκτός από την ίδια την εφαρμογή και τις αντίστοιχες εκδόσεις προγραμματιστικών πακέτων και βιβλιοθηκών που απαιτούνται για την εγκατάστασή της.

Προκειμένου να δημιουργηθούν ένα η περισσότερα κοντέινερ χρησιμοποιείται μία *εικόνα (image)*, δηλαδή αρχεία που περιέχουν κατάλληλες οδηγίες. Πιο συγκεκριμένα, κάθε εικόνα έχει το ρόλο ενός προτύπου το οποίο περιέχει όλα τα απαραίτητα στοιχεία που απαρτίζουν τα κοντέινερ. Συνήθως οι εικόνες βρίσκονται αποθηκευμένες στο σύννεφο απ' όπου οι χρήστες μπορούν να τις "κατεβάσουν" τοπικά στο σύστημά τους, έτσι ώστε να κατασκευάσουν τα αντίστοιχα κοντέινερ.



Εικόνα 11: Σύνθεση κοντέινερ από μία εικόνα σε υψηλό επίπεδο (https://dhathriblog.medium.com/difference-between-docker-image-and-container-85354526c914).

Στην περίπτωση του BigDAWG δημιουργείται ένα δίκτυο με την ονομασία "bigdawg" το οποίο και φιλοξενεί τα docker containers. Για την εγκατάσταση αυτών των κοντέινερ χρησιμοποιούνται 3 εικόνες, μία για κάθε storage engine (accumulo, scidb, postgresql), οι οποίες είναι προσβάσιμες από τη βιβλιοθήκη εικόνων του docker.

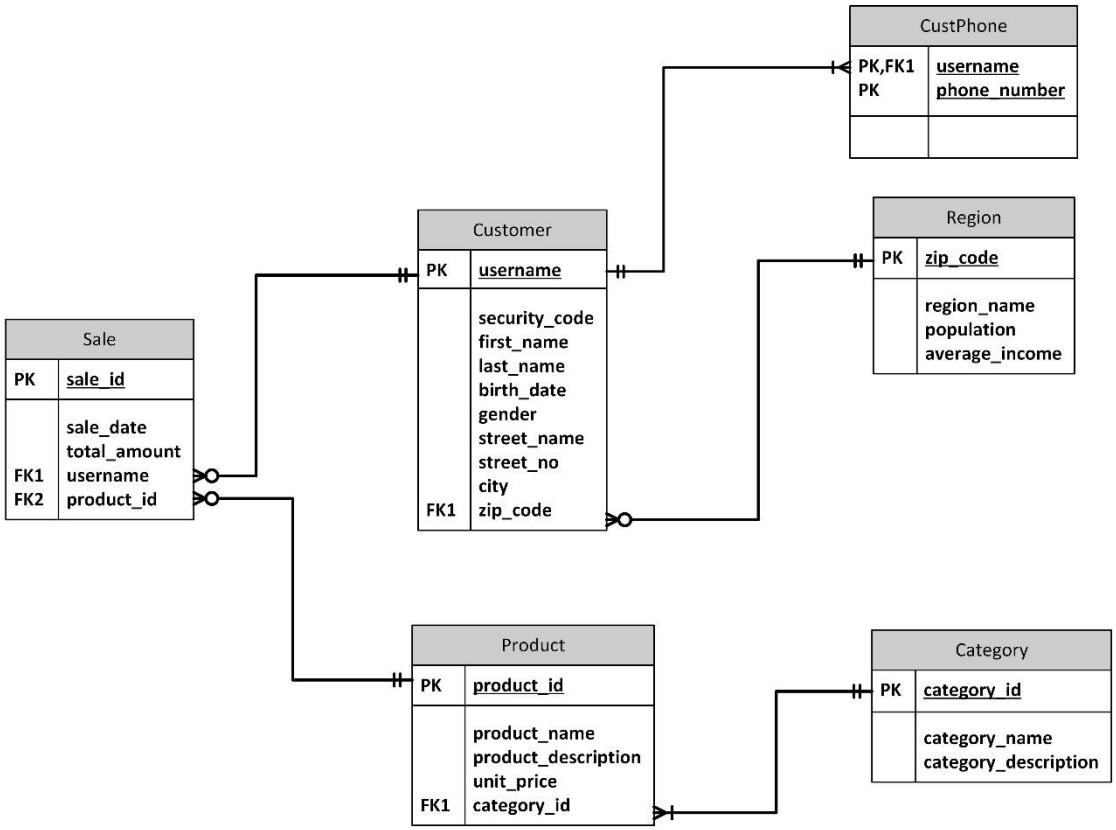
3.4 Θεωρητικά μοντέλα αναπαράστασης δεδομένων

Στην τελευταία ενότητα του κεφαλαίου αυτού, παρουσιάζονται τα τρία storage engines που εμπεριέχονται στην εφαρμογή του BigDAWG, καθώς και τα επιμέρους μοντέλα αποθήκευσης δεδομένων στα οποία βασίζονται. Ο όρος του μοντέλου στην επιστήμη των δεδομένων αναφέρεται στην ανάγκη της περιγραφής μίας πραγματικής ή και όχι κατάστασης, μέσα από ένα τυπικό και αυστηρό τρόπο. Λαμβάνοντας υπόψιν ότι τα μοντέλα αυτά παρουσιάζουν διαφορές στα χαρακτηριστικά τους, και κατ' επέκταση κάθε ένα από αυτά μπορεί να θεωρηθεί καταλληλότερο για την αποθήκευση συγκεκριμένου τύπου δεδομένων, προκύπτει η ανάγκη για την περαιτέρω ανάλυσή τους. Άλλωστε, κατά την υλοποίηση

ενός smart grid συστήματος το οποίο αποτελεί το case study της παρούσας εργασίας και χαρακτηρίζεται από ετερογενείς πηγές δεδομένων, θα πρέπει να παρθούν αποφάσεις σχετικά με το ποια είναι η καταλληλότερη μηχανή αποθήκευσης ώστε να φιλοξενήσει τον εκάστοτε τύπο δεδομένων.

3.4.1 PostgreSQL και Σχεσιακό Μοντέλο

Το σχεσιακό μοντέλο αποτελεί με διαφορά τον πιο διαδεδομένο τρόπο αναπαράστασης και επεξεργασίας δομημένης πληροφορίας. Τα δεδομένα αποθηκεύονται σε αυστηρή μορφή και πιο συγκεκριμένα σε αυτήν ενός πίνακα, ο οποίος εκφράζει μία σχέση. Κάθε πίνακας χαρακτηρίζεται από ένα *σχήμα (schema)*, το οποίο αναφέρεται στο όνομα της σχέσης-πίνακα καθώς και στο σύνολο των *χαρακτηριστικών (attributes)* του, τα οποία αποτελούν τις στήλες του πίνακα. Αντίστοιχα, οι γραμμές του παριστάνουν μία σχέση μεταξύ ενός συνόλου από τιμές-δεδομένα, ενώ κάθε γραμμή χαρακτηρίζεται ως μία *εγγραφή* ή *πλειάδα*. Επομένως, οποιαδήποτε ΒΔ εκφράζεται με το μοντέλο αυτό, εμπεριέχει ένα σύνολο πινάκων, τις σχέσεις μεταξύ αυτών, αλλά και τους περιορισμούς των δεδομένων που αποθηκεύονται σε αυτούς.



Εικόνα 12: Σχεσιακό μοντέλο αναπαράστασης δεδομένων για τις σχέσεις μεταξύ Πελάτη, Τηλεφώνου πελάτη, Περιοχής κατοικίας, Πώλησης προϊόντος, Προϊόντος και Κατηγορίας προϊόντος.

Κάθε πίνακας-σχέση πρέπει να αποτελείται απαραίτητα από ένα ή και περισσότερα attributes τα οποία συνδυαστικά να χαρακτηρίζουν μοναδικά κάθε εγγραφή του πίνακα. Με άλλα λόγια, είναι αναγκαία

συνθήκη κάθε γραμμή ενός πίνακα να περιέχει διαφορετικές τιμές σε μία συγκεκριμένη στήλη ή ένα σύνολο αυτών. Αυτή η μία ή και περισσότερες στήλες χαρακτηρίζονται ως το **πρωτεύων κλειδί** ενός πίνακα, και στο σχεσιακό μοντέλο συμβολίζονται όπως φαίνεται και στην *Εικόνα 12 ως "PK" ("Primary Key"),* συνοδευτικά από μία υπογεγραμμένη. Επιπλέον, με τον συμβολισμό *"FK" ("Foreign Key"),* αναπαρίσταται το **ξένο κλειδί** ενός πίνακα, δηλαδή μία ή και περισσότερες στήλες οι οποίες να αποτελούν το πρωτεύων κλειδί ενός άλλου πίνακα. Θέτοντας ένα ξένο κλειδί, εξασφαλίζεται η σύνδεση μεταξύ δύο πινάκων.

PostgreSQL

Αφορά ένα σύστημα διαχείρισης βάσεων δεδομένων το οποίο υποστηρίζει το σχεσιακό μοντέλο και χρησιμοποιεί τη γλώσσα SQL (Structured Query Language) για την αλληλεπίδρασή του με τις ΒΔ. Μέσα από τη γλώσσα αυτή πραγματοποιούνται εντολές που ανήκουν σε τέσσερις διαφορετικές κατηγορίες:

i. **DDL (Data Definition Language):** Αφορά SQL εντολές οι οποίες ορίζουν το σχήμα της ΒΔ (π.χ. Δημιουργία/Διαγραφή/Τροποποίηση ΒΔ και πινάκων).
ii. **DQL (Data Query language):** Ερωτήματα που απευθύνονται προς τα δεδομένα της ΒΔ και επιστρέφονται υπό τη μορφή πίνακα.
iii. **DML (Data Manipulation Language):** Εντολές οι οποίες σχετίζονται με την επεξεργασία των δεδομένων στους πίνακες (π.χ. Εισαγωγή/Διαγραφή/Τροποποίηση δεδομένων).
iv. **DCL (Data Control Language):** Εντολές σχετικές με την πρόσβαση των χρηστών στις βάσεις δεδομένων.

3.4.2 SciDB και μοντέλο δεδομένων πίνακα (array data model)

Μοντέλο δεδομένων πίνακα

Το μοντέλο δεδομένων σε πίνακα χρησιμοποιείται κυρίως για την αποθήκευση και επεξεργασία πολύ μεγάλων όγκων δεδομένων (της τάξης των petabyte, όπου 1 PB = 1000 TB) σε μαθηματικούς πίνακες (*array*s) N-διαστάσεων. Αυτά προέρχονται συνήθως από αισθητήρες (*sensor data*), ενώ παράλληλα η επεξεργασία τους βασίζεται στη γραμμική άλγεβρα. Μάλιστα, σύμφωνα με τον Stonebraker η επεξεργασία των sensor data μέσα από μαθηματικούς πίνακες είναι 100 φορές ταχύτερη από ότι στους πίνακες (*tables*) που χρησιμοποιούντα στο σχεσιακό μοντέλο (Stonebraker, 2010). Αυτό οφείλεται στο γεγονός ότι ένα RDBMS προκειμένου να εκτελέσει μία πράξη γραμμικής άλγεβρας, χρειάζεται πρώτα να μετατρέψει τον πίνακα σε μαθηματικό πίνακα και εν συνεχεία ξανά στην αρχική του μορφή προκειμένου να επιστρέψει το αποτέλεσμα στον χρήστη (Stonebraker et al., 2011).

SciDB

Πρόκειται για ένα DBMS το οποίο στηρίζεται στο προαναφερθέν μοντέλο δεδομένων. Κάθε πίνακας έχει N-διαστάσεις και οι τιμές των διαστάσεων αυτών έχουν τη μορφή μίας *πλειάδας (tuple)*, οι τιμές της οποίας αντιστοιχούν σε ένα ή περισσότερα γνωρίσματα του πίνακα. Ακόμα, οι τιμές του πίνακα εκτός από ακέραιες, δεκαδικές αλλά και αλφαριθμητικές μπορεί να είναι ένας εμφωλευμένος πίνακας. Για παράδειγμα, ένας πίνακας *myarray* 2-διαστάσεων 5χ5 όπου οι τιμές του είναι μία πλειάδα *(ακέραιος, δεκαδικός)* και τα ονόματα των χαρακτηριστικών του είναι A και B θα μπορούσε να ορισθεί ως εξής:

CREATE ARRAY myarray <A:integer, B: double> [I = 0:5, J = 0:5];

J \ I	[0]	[1]	[2]	[3]	[4]
[0]	(2, 0.7)	(5, 0.5)	(4, 0.9)	(2, 0.8)	(1, 0.2)
[1]	(5, 0.5)	(3, 0.5)	(5, 0.9)	(5, 0.5)	(5, 0.5)
[2]	(4, 0.3)	(6, 0.1)	(6, 0.5)	(2, 0.1)	(7, 0.4)
[3]	(4, 0.25)	(6, 0.45)	(6, 0.3)	(1, 0.1)	(0, 0.3)
[4]	(6, 0.5)	(1, 0.6)	(5, 0.5)	(2, 0.15)	(2, 0.4)

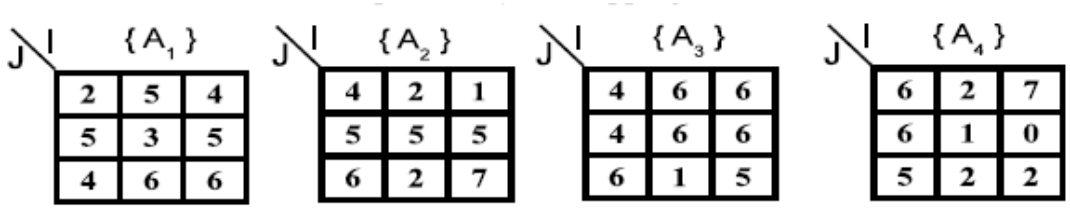
Εικόνα 13: Παράδειγμα πίνακα 2-Διαστάσεων όπου κάθε τιμή περιλαμβάνει μία πλειάδα με τιμές ακέραιο και δεκαδικό αριθμό (Brown, 2010).

Ένα μεγάλο πλεονέκτημα που παρέχει το σύστημα SciDB αφορά την επιλογή του διαχωρισμού του πίνακα τόσο κάθετα, δηλαδή ανά στήλες *(vertical partitioning)*, όσο και σε τμήματα ίσου μεγέθους *(chunking)* τα οποία μπορούν και να επικαλύπτονται. Αυτή η δυνατότητα προσφέρει επιπλέον ταχύτητα στην επεξεργασία των δεδομένων και ιδιαίτερα σε περιπτώσεις όπου αυτά είναι γειτονικά.

J \ I	{A}				
	2	5	4	2	1
	5	3	5	5	5
	4	6	6	2	7
	4	6	6	1	0
	6	1	5	2	2

J \ I	{B}				
	0.7	0.5	0.9	0.8	0.2
	0.5	0.5	0.9	0.5	0.5
	0.3	0.1	0.5	0.1	0.4
	0.25	0.45	0.3	0.1	0.3
	0.5	0.6	0.5	0.15	0.4

Εικόνα 14: Διαχωρισμός του πίνακα της Εικόνας 13 ανά στήλες σε 2 νέους πίνακες {A} και {B} (Brown, 2020).



Εικόνα 15: Διαχωρισμός του πίνακα {A} της Εικόνας 14 σε 4 "κομμάτια" ίσων διαστάσεων (Brown, 2010).

Τα δεδομένα που αποθηκεύονται στους πίνακες αυτούς μπορούν να επεξεργαστούν από δύο γλώσσες ερωτημάτων με την ονομασία AQL (Array Query Language) και AFL (Array Functional Language). Η πρώτη αφορά μια δηλωτική γλώσσα και θυμίζει αρκετά την SQL, ενώ η τελευταία περιλαμβάνει εντολές που εκτελούνται στον πίνακα υπό τη μορφή συνάρτησης.

3.4.3 Accumulo και μοντέλο κλειδιού-τιμής

Μοντέλο δεδομένων κλειδιού: τιμής

Πρόκειται για ένα από τα πιο συνηθισμένα μη σχεσιακά μοντέλα αποθήκευσης δεδομένων, το οποίο χαρακτηρίζεται από την απλότητά του αλλά και την ταχύτητα που προσφέρει όσον αφορά την εκμαίευση της πληροφορίας. Στην πιο απλή του μορφή αποτελείται από μία βάση η οποία χρησιμοποιεί ένα *κλειδί* ώστε να προσδιορίσει μοναδικά μία συγκεκριμένη *τιμή*. Στην πραγματικότητα, η βάση αυτή περιέχει πίνακες που συσχετίζονται μεταξύ τους (*associative arrays*), όπου ο πρώτος περιέχει τα κλειδιά και ο δεύτερος τις αντίστοιχες τιμές. Η διαφορά με το σχεσιακό μοντέλο έγκειται στο γεγονός ότι σε αυτή την περίπτωση δεν υπάρχει προκαθορισμένη δομή, καθώς η *τιμή* μπορεί να περιλαμβάνει οποιοδήποτε τύπο δεδομένων (π.χ. κείμενο, βίντεο, JSON αρχείο κ.τ.λ.). Μοναδική προϋπόθεση είναι τα κλειδιά να είναι μοναδικά μεταξύ τους εφόσον αποτελούν τα αναγνωριστικά των τιμών.

Σημαντικό πλεονέκτημα του μοντέλου κλειδιού-τιμής είναι η ιδιότητα της επεκτασιμότητας, δηλαδή η ικανότητά του να ανταποκρίνεται άριστα σε διαδικασίες εγγραφής και ανάκτησης δεδομένων των οποίων ο όγκος αυξάνεται διαρκώς. Επιπρόσθετα, πρόκειται για ένα μοντέλο με μεγαλύτερη ευελιξία και ταχύτητα από αυτή των σχεσιακών μοντέλων, τα οποία για την αναπαράσταση της ίδιας πληροφορίας απαιτούν μεγαλύτερο αποθηκευτικό χώρο και άρα περισσότερο χρόνο στην εκμαίευση των δεδομένων. Για παράδειγμα, σε ένα σχεσιακό μοντέλο οι προαιρετικές τιμές πρέπει οπωσδήποτε να αναπαρίστανται από ένα placeholder[20] (π.χ. null), κάτι το οποίο δε συμβαίνει στις ΒΔ που φιλοξενούν ζεύγη κλειδιού-τιμής (Karande, 2018).

[20] Αφορά μία τιμή η οποία χρησιμοποιείται για την προσωρινή αναπαράσταση δεδομένων.

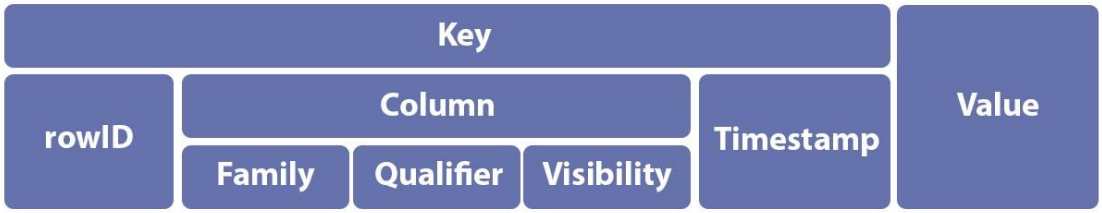
Key	Value
K1	AAA,BBB,CCC
K2	AAA,BBB
K3	AAA,DDD
K4	AAA,2,01/01/2015
K5	3,ZZZ,5623

Εικόνα 16: Πίνακες απεικόνισης κλειδιού-τιμής (https://en.wikipedia.org/wiki/Key%E2%80%93value_database).

Apache Accumulo

Το σύστημα Accumulo έχει υιοθετήσει το μοντέλο κλειδιού-τιμής με μερικές τροποποιήσεις, όσον αφορά το πεδίο του κλειδιού το οποίο και κατατάσσεται αυτόματα σε ταξινομημένη σειρά. Η δομή του αποτελείται από τα ακόλουθα στοιχεία:

i. **rowID:** Αναπαριστά το αναγνωριστικό της εκάστοτε γραμμής.
ii. **Column:** Το οποίο με τη σειρά του διαιρείται σε:
    a. **Family:** Ευρύτερη κατηγορία που ανήκει ένα κλειδί.
    b. **Qualifier:** Ειδικότερη κατηγορία που ανήκει ένα κλειδί.
    c. **Visibility:** Καθορίζει ποιοι χρήστες διαθέτουν πρόσβαση στην τιμή του κλειδιού αυτού.
iii. **Timestamp:** Παράγεται αυτόματα κατά την εισαγωγή του κλειδιού και της αντίστοιχης τιμής που προσδιορίζει.



Εικόνα 17: Αρχιτεκτονική ζεύγους κλειδιού-τιμής κατά το Accumulo σύστημα (Moreno et al., 2018).

Ένα ακόμη γνώρισμα του Accumulo είναι ότι το ζεύγος κλειδιού-τιμής αποθηκεύεται σε έναν πίνακα και όχι σε associative arrays. Το γεγονός ότι χρησιμοποιείται ένας πίνακας για την αποθήκευση των δεδομένων δε θα πρέπει να συγχέεται με την πεποίθηση ότι το Accumulo storage engine αφορά ένα σχεσιακό μοντέλο, καθώς τα δεδομένα που φιλοξενεί είναι ημιδομημένα και όχι δομημένα. Αξίζει

επίσης να αναφερθεί πως ο αρχικός πίνακας μπορεί να διαιρεθεί σε μικρότερους με βάση τα χαρακτηριστικά γνωρίσματα του κλειδιού. Οι νέοι αυτοί πίνακες ονομάζονται “tablets” και προσφέρουν μεγαλύτερη ταχύτητα στη διαδικασία ανάκτησης των δεδομένων (Sawyer et al., 2013).

4

Προσομοίωση έξυπνου ηλεκτρικού δικτύου

Στο παρόν κεφάλαιο θα πραγματοποιηθεί η παρουσίαση μίας μελέτης περίπτωσης σχετικά με την υλοποίηση ενός έξυπνου ηλεκτρικού δικτύου ή αλλιώς smart grid, χρησιμοποιώντας την υποδομή της εφαρμογής BigDAWG. Το συγκεκριμένο σενάριο μπορεί να αποτελέσει ρεαλιστικό παράδειγμα εφαρμογής ενός polystore συστήματος.

Μετά τον ορισμό του προβλήματος επακολουθεί η περιγραφή του συνόλου δεδομένων που θα χρησιμοποιηθεί για την υλοποίηση της εργασίας, το οποίο προέρχεται από την πλατφόρμα Kaggle. Στη συνέχεια τα δεδομένα αυτά θα μετασχηματιστούν εν μέρει έτσι ώστε να αντικατοπτρίζουν καλύτερα το σενάριο που έχει οριστεί, αλλά και να αποκτήσουν την κατάλληλη μορφή ώστε να εισαχθούν στις μηχανές αποθήκευσης του polystore.

Στο στάδιο της υλοποίησης θα γίνει λεπτομερής περιγραφή της διαδικασίας φόρτωσης των δεδομένων στο σύστημα, έχοντας πρώτα αποσαφηνίσει ποιο είναι το καταλληλότερο storage engine για συγκεκριμένους τύπους δεδομένων. Τέλος, το κεφάλαιο ολοκληρώνεται με την εκτέλεση ορισμένων queries τόσο για την εξακρίβωση της ορθής διαδικασίας εισαγωγής των δεδομένων, όσο και για λόγους επίδειξης των δυνατοτήτων του BigDAWG συμπλέγματος.

4.1 Ορισμός του προβλήματος

Όπως έχει διευκρινιστεί και σε προηγούμενη ενότητα, ο βασικός σκοπός της εργασίας αφορά την εισαγωγή δεδομένων σε ένα polystore σύστημα με την ονομασία BigDAWG. Για το λόγο αυτό κρίνεται αναγκαία η αναπαράσταση της παραπάνω διαδικασίας μέσα από την προσομοίωση ενός ρεαλιστικού σεναρίου το οποίο πληροί τις ιδιότητες ενός polystore. Ειδικότερα, πρόκειται για την υλοποίηση ενός έξυπνου ηλεκτρικού δικτύου το οποίο χαρακτηρίζεται από δεδομένα προερχόμενα από ετερογενείς πηγές. Πρόκληση αποτελεί όχι μόνο η εύρεση μεθόδου για την εμφάνιση των δεδομένων της μελέτης περίπτωσης στο BigDAWG, αλλά και η ανάθεσή τους στις κατάλληλες μηχανές αποθήκευσης

(PostgreSQL, SciDB και Accumulo) οι οποίες μπορούν να υποστηρίξουν τη δομή των δεδομένων με τον καλύτερο δυνατό τρόπο.

4.1.1 Smart Grid

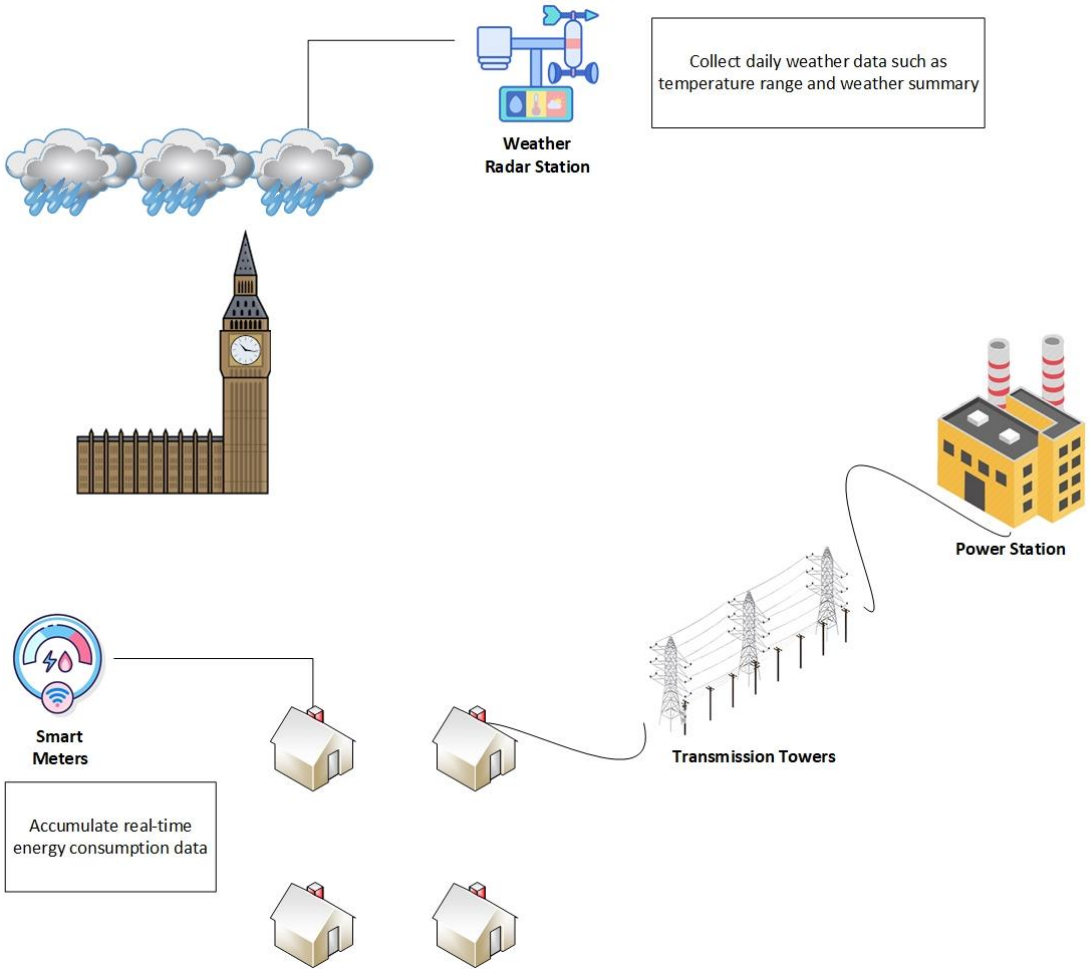
Ένα smart grid αξιοποιεί την τεχνολογία προκειμένου να ενσωματώσει την ηλεκτρική ενέργεια που παράγεται και καταναλώνεται, σε συνδυασμό με τον τεράστιο όγκο δεδομένων που την συνοδεύει. Εκείνο το γνώρισμα το οποίο μετατρέπει ένα συμβατικό ηλεκτρικό δίκτυο [21] σε "έξυπνο", είναι η παρουσία της τεχνολογίας του Διαδικτύου των Πραγμάτων μέσα από αισθητήρες οι οποίοι παράγουν, ανά τακτά χρονικά διαστήματα, δεδομένα (Big Data). Οι αισθητήρες αυτοί μεταφράζονται ως έξυπνοι μετρητές κατανάλωσης ενέργειας από κάθε νοικοκυριό ή επιχείρηση και μάλιστα σε πραγματικό χρόνο. Συνδυαστικά μπορούν να υπάρχουν μετρητές σχετικά με τις καιρικές συνθήκες, την ανίχνευση πιθανών διακυμάνσεων της ηλεκτρικής ενέργειας, όπως και άλλου είδους τεχνολογίες οι οποίες οδηγούν σε εναλλακτικές μορφές ενέργειας (π.χ. αιολική και ηλιακή). Ενδεικτικά, αναφέρονται τα εξής πλεονεκτήματα που προσφέρει ένα smart grid:

- Αποδοτικότερη παραγωγή ηλεκτρικής ενέργειας.
- Δυνατότητα αυτόματης αποκατάστασης του ηλεκτρικού δικτύου σε περιπτώσεις διακύμανσης της παραγόμενης ενέργειας.
- Μείωση κόστους μέσα από τη δυνατότητα διαχείρισης της κατανάλωσης ενέργειας των πελατών σε πραγματικό χρόνο.
- Πρόβλεψη πιθανών προβλημάτων που μπορούν να οφείλονται σε ακραία καιρικά φαινόμενα.

4.1.2 "Έξυπνοι μετρητές στο Λονδίνο"

Ένας τύπος έξυπνου ηλεκτρικού δικτύου φαίνεται να υπάρχει στην περιοχή του Λονδίνου, στα νοικοκυριά του οποίου έχουν εγκατασταθεί έξυπνοι μετρητές κατανάλωσης ενέργειας με σκοπό την αντιμετώπιση της κλιματικής αλλαγής, αλλά και τη γενικότερη αναβάθμιση του τοπικού ενεργειακού δικτύου. Η παρούσα εργασία θα εισάγει τα δεδομένα που παράγονται σε πραγματικό χρόνο από τους μετρητές αυτούς στο BigDAWG, ταυτόχρονα με περιγραφικά δεδομένα για κάθε αντίστοιχο νοικοκυριό αλλά και πληροφορίες, σχετικά με τις καιρικές συνθήκες που επικρατούν στις ημερομηνίες για τις οποίες συλλέχθηκαν τα δεδομένα κατανάλωσης ενέργειας.

[21] Παροχή ηλεκτρικής ενέργειας από έναν ηλεκτροπαραγωγό προς κατανάλωση από επιχειρήσεις και νοικοκυριά.



Εικόνα 18: Μελέτη περίπτωσης έξυπνου ηλεκτρικού δικτύου.

4.2 Περιγραφή δεδομένων

Το σετ δεδομένων που θα χρησιμοποιηθεί για την προσομοίωση της προαναφερόμενης μελέτης περίπτωσης προέρχεται από την πλατφόρμα Kaggle (https://www.kaggle.com/jeanmidev/smart-meters-in-london). Τα δεδομένα αυτά περιέχουν ένα δείγμα 5567 νοικοκυριών, συνοδευόμενα από την ενεργειακή κατανάλωσή τους (εκφρασμένη σε κιλοβατώρες *kWh*) κατά τη διάρκεια ολόκληρης της ημέρας, για το διάστημα μεταξύ Νοεμβρίου 2011 και Φεβρουαρίου 2014. Για την απεικόνιση της ενεργειακής κατανάλωσης των νοικοκυριών παρέχονται δύο τύποι αρχείων της μορφής ".csv", εκ των οποίων ο πρώτος αφορά συγκεντρωτικές μετρήσεις για το διάστημα κάθε μίας ημέρας ενώ ο δεύτερος μετρήσεις ανά μισάωρο μέσα στο ίδιο διάστημα. Ακολουθεί η προεπισκόπηση των αρχείων αυτών σε μορφή πινάκων:

	LCLid	day	energy_median	energy_mean	energy_max	energy_count	energy_std	energy_sum	energy_min
0	MAC000131	2011-12-15	0.4850	0.432045	0.868	22	0.239146	9.505	0.072
1	MAC000131	2011-12-16	0.1415	0.296167	1.116	48	0.281471	14.216	0.031
2	MAC000131	2011-12-17	0.1015	0.189812	0.685	48	0.188405	9.111	0.064
3	MAC000131	2011-12-18	0.1140	0.218979	0.676	48	0.202919	10.511	0.065
4	MAC000131	2011-12-19	0.1910	0.325979	0.788	48	0.259205	15.647	0.066

Πίνακας 1: Προεπισκόπηση του αρχείου "daily_dataset.csv" (Κώδικας1, Appendix).

Οι μεταβλητές του παραπάνω πίνακα ερμηνεύονται ως εξής:

- **LCLid:** Το μοναδικό αναγνωριστικό ενός νοικοκυριού.
- **day:** Η ημερομηνία μέτρησης της κατανάλωσης ενέργειας.
- **energy_median:** Η διάμεσος της κατανάλωσης ενέργειας.
- **energy_mean:** Ο μέσος της κατανάλωσης ενέργειας.
- **energy_max:** Η μέγιστη τιμή κατανάλωσης στο διάστημα της ημέρας.
- **energy_count:** Ο συνολικός αριθμός των συσκευών μέτρησης κατανάλωσης στη διάρκεια της ημέρας.
- **energy_std:** Η τυπική απόκλιση της κατανάλωσης ενέργειας.
- **energy_sum:** Η συνολική ενέργεια που καταναλώθηκε στο διάστημα της ημέρας.
- **energy_min:** Η ελάχιστη τιμή κατανάλωσης στο διάστημα της ημέρας.

	LCLid	tstp	energy(kWh/hh)
472492	MAC004828	2014-02-27 22:00:00.0000000	0
472493	MAC004828	2014-02-27 22:30:00.0000000	0.001
472494	MAC004828	2014-02-27 23:00:00.0000000	0.047
472495	MAC004828	2014-02-27 23:30:00.0000000	0.008

Πίνακας 2: Προεπισκόπηση του αρχείου "halfhourly_dataset/block_111.csv" (Κώδικας1, Appendix).

Οι μεταβλητές του παραπάνω πίνακα ερμηνεύονται ως εξής:

- **LCLid:** Το μοναδικό αναγνωριστικό ενός νοικοκυριού.
- **tstp:** Η ημερομηνία και ώρα (σε διαστήματα των 30 λεπτών) καταγραφής κατανάλωσης της ενέργειας.
- **energy(kWh/hh):** Η κατανάλωση ενέργειας εκφρασμένη σε κιλοβατώρες ανά 30 λεπτά.

Όσον αφορά τα νοικοκυριά, παρέχεται ένα σετ δεδομένων το οποίο χαρακτηρίζει κάθε ένα από αυτά με μία συγκεκριμένη κατηγορία νοικοκυριών.

	LCLid	stdorToU	Acorn	Acorn_grouped
0	MAC005492	ToU	ACORN-	ACORN-
1	MAC001074	ToU	ACORN-	ACORN-
2	MAC000002	Std	ACORN-A	Affluent
3	MAC003613	Std	ACORN-A	Affluent
4	MAC003597	Std	ACORN-A	Affluent

Πίνακας 3: Προεπισκόπηση του αρχείου "informations_households.csv" (Κώδικας1, Appendix.).

Οι μεταβλητές του παραπάνω πίνακα ερμηνεύονται ως εξής:

- **LCLid:** Το μοναδικό αναγνωριστικό ενός νοικοκυριού.
- **stdorToU:** Είδος φόρου (σταθερός ή δυναμικός) που επιβάλλεται σε κάθε νοικοκυριό.
- **Acorn:** Μεταβλητή κατηγοριοποίησης των νοικοκυριών.
- **Acorn_grouped:** Εναλλακτική μεταβλητή κατηγοριοποίησης των νοικοκυριών.

Με βάση τη μεταβλητή *"Acorn"*, οι τιμές της οποίας είναι της μορφής "Acorn-X", αναμένεται να γίνουν ορισμένες υποθέσεις σχετικά με τα νοικοκυριά, καθώς παρατηρείται έλλειψη περαιτέρω πληροφορίας για λόγους ανωνυμίας. Αναλυτικότερες πληροφορίες για την μεταβλητή "Acorn" δίνονται μέσα από τον επόμενο πίνακα:

	MAIN CATEGORIES	CATEGORIES	REFERENCE	ACORN-A	ACORN-B	ACORN-C	ACORN-D	ACORN-E	ACORN-F	ACORN-G	ACORN-H	ACORN-I	ACORN-J	A
0	POPULATION	Age	Age 0-4	77.0	83.0	72.0	100.0	120.0	77.0	97.0	97.0	63.0	119.0	
1	POPULATION	Age	Age 5-17	117.0	109.0	87.0	69.0	94.0	95.0	102.0	106.0	67.0	95.0	
2	POPULATION	Age	Age 18-24	64.0	73.0	67.0	107.0	100.0	71.0	83.0	89.0	62.0	104.0	
3	POPULATION	Age	Age 25-34	52.0	63.0	62.0	197.0	151.0	66.0	90.0	88.0	63.0	132.0	
4	POPULATION	Age	Age 35-49	102.0	105.0	91.0	124.0	118.0	93.0	102.0	103.0	76.0	111.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
821	LEISURE TIME	Holiday Destination/Type	Asia	171.0	137.0	94.0	220.0	196.0	69.0	122.0	95.0	72.0	97.0	
822	LEISURE TIME	Holiday Destination/Type	Activity / Outdoor Sports	298.0	278.0	138.0	119.0	93.0	96.0	113.0	96.0	84.0	96.0	
823	LEISURE TIME	Holiday Destination/Type	Cruise	272.0	295.0	272.0	44.0	44.0	70.0	70.0	70.0	70.0	70.0	
824	LEISURE TIME	Holiday Destination/Type	Package	196.0	186.0	166.0	49.0	49.0	101.0	101.0	121.0	106.0	101.0	
825	LEISURE TIME	Holiday Destination/Type	Self-catering	308.0	181.0	181.0	55.0	55.0	127.0	147.0	124.0	113.0	166.0	

Πίνακας 4: Προεπισκόπηση του αρχείου "acorn_details.csv" (Κώδικας1, Appendix.).

Η δεύτερη και τρίτη στήλη ("MAIN CATEGORIES", "CATEGORIES") σχετίζονται με κάποια χαρακτηριστικά τα οποία μπορούν να περιγράψουν ένα νοικοκυριό/σπίτι (π.χ. χρηματική αξία, αριθμός ενοίκων κ.α.). Η στήλη "ACORN" λαμβάνει 17 διαφορετικές τιμές από "A-Q". Η τιμή των στηλών αυτών μπορεί να επεξηγηθεί με ένα παράδειγμα. Αν δηλαδή παρατηρηθεί η τιμή 150 σε ένα συγκεκριμένο γκρουπ-Acorn, αυτό σημαίνει ότι υπάρχει 1.5 φορές μεγαλύτερη πιθανότητα η κατηγορία αυτή να

χαρακτηρίζεται από την τιμή της στήλης “CATEGORIES”. Οι ευρύτερες κατηγορίες όπου ανήκει η στήλη “CATEGORIES ”, δηλαδή οι μοναδικές τιμές της στήλης “MAIN CATEGORIES” διακρίνονται στην παρακάτω εικόνα:

```
array(['POPULATION', 'HOUSING', 'FAMILY', 'ECONOMY', 'EDUCATION',
       'HEALTH', 'TRANSPORT', 'MARKETING CHANNELS', 'FINANCE', 'DIGITAL',
       'SHOPPING', 'CONTACT', 'ENVIRONMENT', 'COMMUNITY SAFETY',
       'LEISURE TIME'], dtype=object)
```

Εικόνα 19: Μοναδικές τιμές της στήλης “MAIN CATEGORIES” (Κώδικας1, Appendix).

Με δεδομένο ότι δεν υπάρχουν περαιτέρω πληροφορίες για τα νοικοκυριά/σπίτια, αλλά και για λόγους πληρότητας της μελέτης περίπτωσης αποφασίστηκε η δημιουργία ενός νέου σετ δεδομένων με την ονομασία “house_info.csv” το οποίο περιέχει επιπλέον πληροφορίες. Για την υλοποίηση του αρχείου χρησιμοποιήθηκαν κάποιες υποθέσεις οι οποίες προέκυψαν από την αντιστοίχιση του γνωρίσματος “Acorn” *(ανήκει στο “informations_households.csv”)* κάθε σπιτιού με τα γνωρίσματα “CATEGORIES” και “REFERENCE” *(ανήκουν στο “acorn_details.csv”)*.

	houseId	tariff	acorn	type	tenure	value	members	annualIncome
0	3613	Std	affluent	Detached house	Owned outright	1200000	5	145000
1	4054	ToU	affluent	Bungalow	Mortgaged	780000	3	97500
2	3358	ToU	affluent	Detached house	Mortgaged	1380000	4	198000
3	1395	Std	affluent	Detached house	Owned outright	1530000	4	141000
4	3907	Std	affluent	Detached house	Owner occupied	855000	3	112000

Πίνακας 5: Προεπισκόπηση του αρχείου “house_info.csv” (Κώδικας1, Appendix).

Για τον παραπάνω πίνακα οι μεταβλητές περιγράφονται ως εξής:

- **houseId:** Το μοναδικό αναγνωριστικό ενός νοικοκυριού.
- **tariff:** Είδος φόρου (σταθερός ή δυναμικός) που επιβάλλεται σε κάθε νοικοκυριό.
- **acorn:** Οικονομική κατάσταση του νοικοκυριού.
- **type:** Ο τύπος της κατοικίας.
- **tenure:** Τύπος γαιοκτησίας.
- **value:** Αντικειμενική αξία της κατοικίας.
- **members:** Αριθμός ενοίκων που διαμένουν στην οικία.
- **annualIncome:** Το ετήσιο συνολικό εισόδημα του νοικοκυριού.

Τα καιρικά δεδομένα βρίσκονται στο αρχείο "weather_daily_darksky.csv" και ένα μέρος αυτών[22] διακρίνεται παρακάτω:

	temperatureMax	temperatureMaxTime	temperatureMin	summary
0	11.96	2011-11-11 23:00:00	8.85	Foggy until afternoon.
1	8.59	2011-12-11 14:00:00	2.48	Partly cloudy throughout the day.
2	10.33	2011-12-27 02:00:00	8.03	Mostly cloudy throughout the day.
3	8.07	2011-12-02 23:00:00	2.56	Partly cloudy throughout the day and breezy ov...
4	8.22	2011-12-24 23:00:00	3.17	Mostly cloudy throughout the day.

Πίνακας 6: Δεδομένα καιρικών συνθηκών για το αρχείο " weather_daily_darksky.csv " (Κώδικας1, Appendix).

Για τον παραπάνω πίνακα ισχύει ότι:

- **temperatureMax:** Η υψηλότερη καταγεγραμμένη θερμοκρασία τη συγκεκριμένη ημερομηνία.
- **temperatureMaxTime:** Η ημερομηνία και ώρα καταγραφής της υψηλότερης θερμοκρασίας.
- **temperatureMin:** Η χαμηλότερη καταγεγραμμένη θερμοκρασία για τη συγκεκριμένη ημερομηνία.
- **summary:** Σύνοψη καιρικού δελτίου.

Το τελευταίο σετ δεδομένων αφορά τις ημερομηνίες που αποτελούν και αργίες κατά το διάστημα μεταξύ Νοεμβρίου 2011 και Φεβρουαρίου 2014.

	Bank holidays	Type
0	2012-12-26	Boxing Day
1	2012-12-25	Christmas Day
2	2012-08-27	Summer bank holiday
3	2012-05-06	Queen?s Diamond Jubilee (extra bank holiday)
4	2012-04-06	Spring bank holiday (substitute day)

Πίνακας 7: Προεπισκόπηση του αρχείου "uk_bank_holidays.csv" (Κώδικας1, Appendix).

Οι μεταβλητές του Πίνακα 7 ερμηνεύονται ως εξής:

- **Bank holidays**: Ημερομηνία αργίας.
- **Type**: Τύπος αργίας.

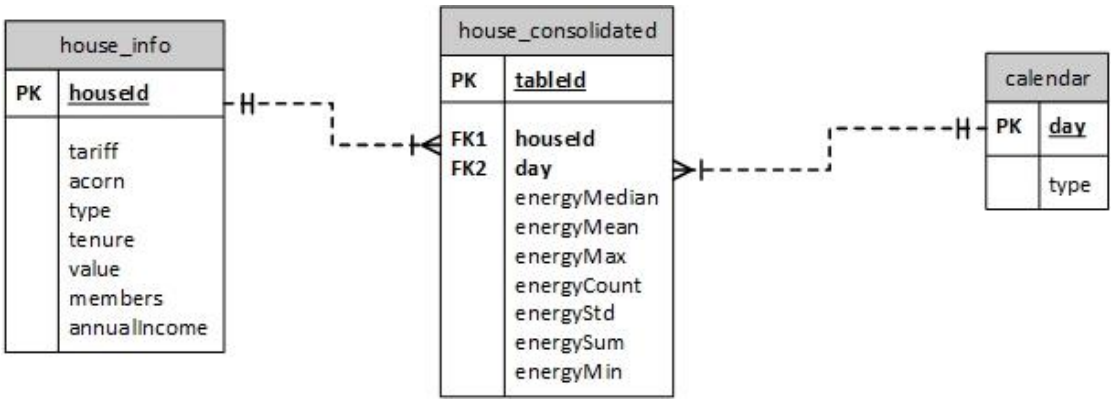
[22] Παρουσιάζεται μόνο το μέρος του αρχείου (από άποψη στηλών) το οποίο και θα εισαχθεί στο BigDAWG σύστημα.

4.2.1 Αντιστοιχία μεταξύ δεδομένων και μηχανών αποθήκευσης του BigDAWG

Το βασικό ερώτημα που χρειαζόταν να απαντηθεί μετά την περιγραφή των δεδομένων που προηγήθηκε, αφορά την αντιστοίχισή τους με τις κατάλληλες μηχανές αποθήκευσης τις οποίες παρέχει το σύμπλεγμα του BigDAWG. Ο σκοπός λοιπόν ήταν να επιλεγεί εκείνη η μηχανή που θα μπορέσει να ανταποκριθεί με τη μεγαλύτερη επιτυχία αλλά και ταχύτητα στις απαιτήσεις για αποθήκευση και επεξεργασία των δεδομένων, σε συνάρτηση πάντα με τα γνωρίσματα (*attributes*) αυτών. Παράλληλα, λήφθηκε υπόψιν ότι το παρόν σενάριο περιλαμβάνει ένα πολύ μικρό αριθμό δεδομένων σε σχέση με ένα πραγματικό, υπό την έννοια ότι η σωστή επιλογή ενός storage engine έγινε με γνώμονα και τη δυνητική αύξηση του όγκου των δεδομένων στο μέλλον. Προφανώς, προκειμένου να διατηρηθεί η ιδιότητα των polystore συστημάτων αξιοποιήθηκαν και οι τρείς, ετερογενείς μεταξύ τους, μηχανές αποθήκευσης.

4.2.1.1 Αντιστοίχιση με PostgreSQL

Το αρχείο “daily_dataset.csv” το οποίο περιλαμβάνει περιγραφικά στατιστικά μέτρα για την ημερήσια κατανάλωση κάθε σπιτιού και αποτελεί “προϊόν” των δεδομένων κατανάλωσης ανά μισή ώρα, χαρακτηρίζεται από δομημένα δεδομένα. Το ίδιο ισχύει και για το αρχείο “house_info.csv ” ο όγκος του οποίου είναι αμελητέος. Για το λόγο αυτό τα δύο σετ δεδομένων επιλέχθηκαν να αποθηκευτούν στη σχεσιακή ΒΔ της PostgreSQL. Ταυτόχρονα, στην ίδια βάση δεδομένων θα εισαχθεί και ένας νέος πίνακας με το όνομα “calendar”, ο οποίος προκύπτει από το αρχείο “ uk_bank_holidays.csv ” και περιλαμβάνει όλες τις πιθανές ημερομηνίες μεταξύ του ορισμένου χρονικού διαστήματος, αλλά και τους τύπους αργίας στο διάστημα αυτό.



Εικόνα 20: Σχεσιακό μοντέλο του PostgreSQL storage engine.

Το παραπάνω σχήμα αναπαριστά το απλουστευμένο σχεσιακό μοντέλο που θα υλοποιηθεί στην PostgreSQL βάση δεδομένων του συστήματος. Ο πίνακας “house_info” συμβολίζει τα δεδομένα που προέρχονται από το αρχείο “house_info.csv” και διαθέτει το γνώρισμα “houseId” ως πρωτεύον κλειδί, το

οποίο και συνδέεται μέσω ξένου κλειδιού με τον πίνακα “house_consolidated”. Ο τελευταίος πίνακας, αναπαριστά τα δεδομένα του αρχείου “ daily_dataset.csv”, ενώ ως πρωτεύον κλειδί διαθέτει τη στήλη “tableId” η οποία αναπαριστά έναν αύξοντα αριθμό με αρχική τιμή το 1. Και στους δύο προηγούμενους πίνακες η στήλη “LCLid” αντικαταστάθηκε με το όνομα “houseId”. Τέλος, ο πίνακας “house_consolidated” συνδέεται με τον calendar“ μέσω του ξένου κλειδιού “day”, το οποίο συμβολίζει τις ημερομηνίες καταγραφής κατανάλωσης ηλεκτρικής ενέργειας.

Η πληθικότητα απεικόνισης *(mapping cardinalities)* του προηγούμενου μοντέλου μεταφράζεται ως εξής:

- Για τη σχέση “house_info” – “house_consolidated”: *Ένα προς πολλά συσχέτιση*.
- Για τη σχέση “house_consolidated” – “calendar”: *Πολλά προς ένα συσχέτιση.*

Η βάση δεδομένων που θα χρησιμοποιηθεί για την υλοποίηση του προηγούμενου σχεσιακού μοντέλου έχει όνομα “households”. Τέλος, το ίδιο όνομα θα έχει και το schema της βάσης αυτής.

4.2.1.2 Αντιστοίχιση με SciDB και Accumulo

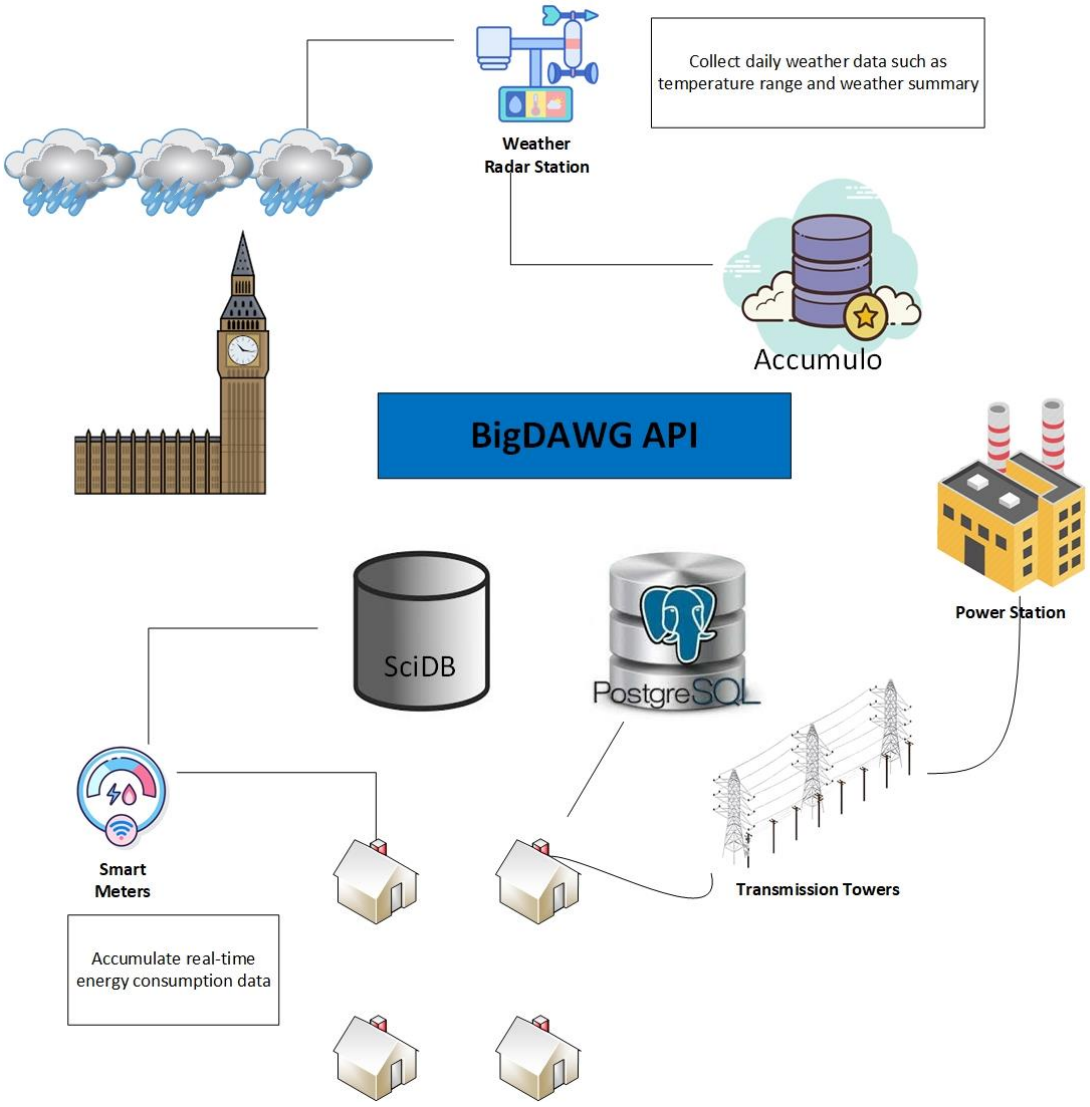
SciDB

Η βάση δεδομένων SciDB η οποία χρησιμοποιεί μαθηματικούς πίνακες για την αποθήκευση δεδομένων αποτελεί ιδανική λύση ώστε να φιλοξενήσει τα δεδομένα που παράγονται από τους έξυπνους μετρητές για κάθε σπίτι. Άλλωστε, τα δεδομένα αυτά μπορεί να είναι δομημένα, ωστόσο πρόκειται στην ουσία για χρονοσειρές *(time series)*. Καθίσταται συνεπώς προφανές, ότι ο όγκος των δεδομένων θα είναι εξαιρετικά μεγάλος και ταυτόχρονα θα πρέπει να παρέχεται γρήγορη πρόσβαση σε αυτά προκειμένου να πραγματοποιούνται πράξεις μαθηματικής ανάλυσης μεταξύ αυτών.

Accumulo

Η Accumulo βάση δεδομένων χρησιμοποιήθηκε για να φιλοξενήσει data που προέχονται από τις καιρικές συνθήκες για κάθε ημερομηνία. Στα δεδομένα αυτά πέρα από το εύρος της θερμοκρασίας της κάθε ημέρας, περιλαμβάνεται και ένα δελτίο καιρού σε μορφή ελεύθερου κειμένου. Καθώς η ελεύθερη μορφή κειμένου περιγράφει αδόμητα δεδομένα, το μοντέλο “κλειδιού-τιμής” όπου η τιμή περιλαμβάνει τη σύνοψη του δελτίου καιρού μπορεί να αναπαραστήσει βέλτιστα τα δεδομένα του αρχείου “ weather_daily_darksky.csv”.

Παρακάτω αποτυπώνεται εκ νέου το σχήμα της μελέτη περίπτωσης μετά την αντιστοίχιση των δεδομένων του σεναρίου με τις ΒΔ του BigDAWG.

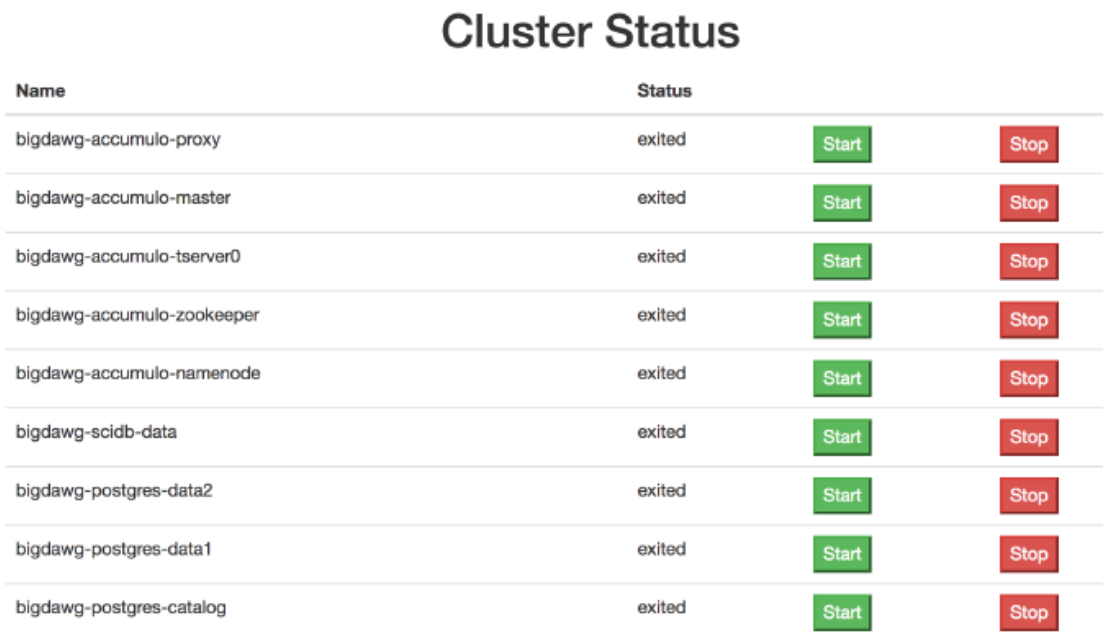


Εικόνα 21: Τελική μορφή μελέτης περίπτωσης μέσω της χρήσης του BigDAWG.

4.3 *Ενσωμάτωση δεδομένων στο BigDAWG*

4.3.1 *Ορισμός στρατηγικής*

Για να επιτευχθεί η εκχώρηση των δεδομένων στις βάσεις του συστήματος ακολουθήθηκαν συγκεκριμένα βήματα. Αρχικά, όπως έχει προαναφερθεί το σύμπλεγμα του BigDAWG στηρίζεται στη σύνθεση ορισμένων docker containers τα οποία εγκαθίστανται αυτόματα μαζί με το σύστημα *(για την εγκατάσταση του συστήματος βλέπε ενότητα 4.3.2)*.



Εικόνα 22: Τα docker containers του BigDAWG.

Όπως φαίνεται και από το παραπάνω στιγμιότυπο υπάρχουν εννέα docker containers εκ των οποίων τα δύο υλοποιούν ΒΔ σε PostgreSQL (BigDAWG Catalog [23]και households), ενώ αντίστοιχα υπάρχουν επιπλέον δύο για τις βάσεις δεδομένων SciDB και Accumulo.

Το πρώτο βήμα αφορά την εύρεση των κατάλληλων κοντέινερ και τη φόρτωση αυτών με τα δεδομένα. Η διαδικασία αυτή προϋποθέτει τη μεταφορά των αρχείων που περιέχουν τα δεδομένα στο περιβάλλον του αντίστοιχου container. Έπειτα, απαιτείται η παραμετροποίηση του κώδικα των αρχείων κάθε container, με τις αντίστοιχες εντολές στη γλώσσα κάθε τύπου ΒΔ έτσι ώστε να δημιουργηθούν τα νέα αντικείμενα (βάσεις δεδομένων, schemas και πίνακες). Η τελευταία κρίσιμη ενέργεια είναι αυτή της ενημέρωσης του καταλόγου του συστήματος για την ύπαρξη των νέων αντικειμένων που δημιουργήθηκαν. Η ενημέρωση του καταλόγου επιτρέπει στο ενδιάμεσο λογισμικό του συστήματος να έχει επίγνωση των νέων δεδομένων και άρα να έχει την ικανότητα πρόσβασης σε αυτά μέσω των queries που συντάσσονται από τους χρήστες.

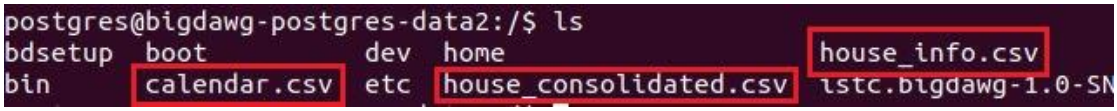
4.3.2 Παραμετροποίηση του κώδικα

Τα κοντέινερ για τα οποία θα γίνουν παραμετροποιήσεις με σκοπό την εισαγωγή των αντίστοιχων δεδομένων όπως ορίστηκε στην ενότητα 4.2.1 είναι τα παρακάτω:

[23] Υπενθυμίζεται ότι ο και ο κατάλογος του polystore είναι μία σχεσιακή βάση δεδομένων σε PostgreSQL.

- **bigdawg-postgres-data2**: Εισαγωγή δεδομένων σε PostgreSQL ΒΔ.
- **bigdawg-scidb-data**: Εισαγωγή δεδομένων σε SciDB ΒΔ.
- **bigdawg-accumulo-zookeeper**: Εισαγωγή δεδομένων σε Accumulo ΒΔ.
- **bigdawg-postgres-catalog**: Εισαγωγή μεταδεδομένων στον κατάλογο σε PostgreSQL ΒΔ.

Στο λειτουργικό σύστημα που χρησιμοποιήθηκε για την υλοποίηση της παρούσας εργασίας βρίσκεται ο φάκελος με την ονομασία "bigdawg2" *(διαθέσιμο στο https://github.com/bigdawg-istc/bigdawg)*, ο οποίος περιλαμβάνει καταλόγους και αρχεία για την αυτόματη εγκατάσταση της εφαρμογής. Στο αρχείο "setup_bigdawg_docker.sh"[24] προστέθηκε ο *Κώδικας2 (βλ. Appendix)* για τη μεταφορά των αρχείων δεδομένων στο περιβάλλον των κατάλληλων docker containers. Στα επόμενα στιγμιότυπα απεικονίζεται η επιτυχής εισαγωγή των αρχείων δεδομένων μέσα από το περιβάλλον κάθε κοντέινερ.

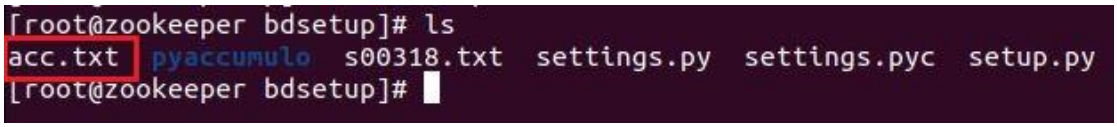


Εικόνα 23: Αρχεία στο περιβάλλον του bigdawg-postgres-data2 container (Κώδικας3, Appendix).

Στο αρχείο "house_info" το οποίο απεικονίζεται στην Εικόνα 23 συμπεριλήφθηκαν δεδομένα περιγραφής 15 διαφορετικών κατοικιών.

Εικόνα 24: Αρχεία στο περιβάλλον του bigdawg-scidb-data container (Κώδικας4, Appendix).

Το αρχείο "halfhourly.csv" της Εικόνας 24 περιέχει τα δεδομένα κατανάλωσης ηλεκτρικής ενέργειας που παράγονται ανά μισή ώρα για 15 διαφορετικές κατοικίες.



Εικόνα 25: Αρχεία στο περιβάλλον του bigdawg-accumulo-data container (Κώδικας5, Appendix).

Το αρχείο "acc.txt" που διακρίνεται στην προηγούμενη εικόνα περιλαμβάνει τα καιρικά δεδομένα για το διάστημα μεταξύ Νοεμβρίου 2011 – Φεβρουαρίου 2014.

[24] Πρόκειται για το αρχείο που εκκινεί τη διαδικασίας εγκατάστασης του BigDAWG.

bigdawg-postgres-data2

Για το κοντέινερ αυτό προστέθηκε ο Κώδικας6 (βλ. Appendix) στο αρχείο "setup.sh"[25]. Ως αποτέλεσμα δημιουργήθηκε η βάση δεδομένων αλλά και το schema της, και τα δύο με την ονομασία "households". Ακόμα, στη βάση αυτή προστέθηκαν οι τρείς πίνακες:

- house_info
- house_consolidated
- calendar

Τέλος, στους πίνακες αυτούς αντιγράφηκαν τα αντίστοιχα δεδομένα μέσω των αρχείων που βρίσκονται στο περιβάλλον του container. Επομένως, το σχεσιακό μοντέλο της Εικόνας 20 έχει πλήρως υλοποιηθεί στο σύστημα.

bigdawg-scidb-data

Στο αρχείο "setup.sh"[26] προστέθηκε ο Κώδικας7 (βλ. Appendix). Με τη χρήση εντολών σε γλώσσα AFL δημιουργήθηκε ένας μονοδιάστατος πίνακας με την ονομασία "halfhourly", κάθε γραμμή του οποίου αποτελείται από τις παρακάτω μεταβλητές:

- **houseId**: τύπου String.
- **day**: τύπου String.
- **time**: τύπου String.
- **energy**: τύπου double.

Στον ίδιο πίνακα εκχωρήθηκαν τα δεδομένα του αρχείου "halfhourly.csv".

bigdawg-accumulo-zookeeper

Για την εκχώρηση δεδομένων στο συγκεκριμένο κοντέινερ χρησιμοποιήθηκε ο Κώδικας8 (βλ. Appendix), που περιλαμβάνει εντολές σε γλώσσα Python και αξιοποίησε τη βιβλιοθήκη *pyaccumulo*, η οποία επιτρέπει τη διασύνδεση με την Accumulo ΒΔ. Ο κώδικας προστέθηκε στο αρχείο "setup.py" [27] και εισάγει τα δεδομένα του αρχείου "acc.txt" σε ένα νέο πίνακα με το όνομα "weather". Κάθε γραμμή του πίνακα έχει τη μορφή:

- **rowID**: day - ημερομηνία.
- **family:** Ένα String με τιμή "temperature range".
- **qualifier:** Το εύρος της θερμοκρασίας για την ημερομηνία που αναγράφεται στο rowID.

[25] Βρίσκεται στο μονοπάτι "bigdawg2/provisions/cluster_setup/postgres-data2/bdsetup".

[26] Βρίσκεται στο μονοπάτι "bigdawg2/provisions/cluster_setup/scidb-data/bdsetup".

[27] Βρίσκεται στο μονοπάτι "bigdawg2/provisions/cluster_setup/accumulo-data/bdsetup".

- **value:** Ελεύθερο κείμενο με την ανασκόπηση του δελτίου καιρού για τη συγκεκριμένη ημερομηνία.

bigdawg-postgres-catalog

Τα αρχεία προς παραμετροποίηση για τον κατάλογο του BigDAWG βρίσκονται στο μονοπάτι "bigdawg2/provisions/cluster_setup/postgres-catalog-bdsetup". Στο αρχείο "catalog_inserts.sql" εφαρμόστηκε ο Κώδικας9 (βλ. Appendix) προς ενημέρωση του καταλόγου για την ύπαρξη των αντικειμένων που δημιουργήθηκαν στα προηγούμενα βήματα. Επιπλέον, δημιουργήθηκε ένα νέο αρχείο με όνομα "households_schemas_ddl.sql" που περιλαμβάνει τον Κώδικα10 (βλ. Appendix) και υλοποιεί το schema και τους πίνακες της ΒΔ "households" από την πλευρά του καταλόγου. Τέλος, ο Κώδικας11 (βλ. Appendix) του αρχείου "setup.sh" ενσωματώνει το schema που δημιουργήθηκε προηγουμένως στα schemas που έχει πρόσβαση ο κατάλογος.

4.4 Εκκίνηση εφαρμογής και παρουσίαση ερωτημάτων

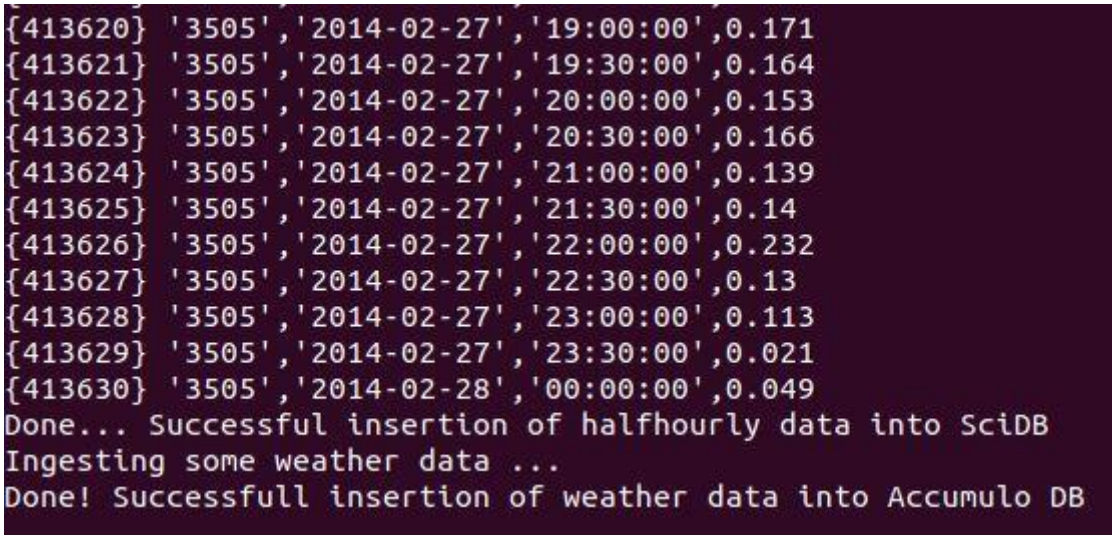
Η εγκατάσταση της εφαρμογής δεν παρουσιάζει δυσκολίες καθώς απαιτείται η εκτέλεση του παραμετροποιημένου αρχείου "setup_bigdawg_docker.sh", το οποίο αναλαμβάνει να εκτελέσει με τη σειρά του τα υπόλοιπα αρχεία που βρίσκονται στον φάκελο του bigdawg2.

Κατά την εγκατάσταση της εφαρμογής, είναι ευδιάκριτα τα μηνύματα ορθής εισαγωγής των δεδομένων της μελέτης περίπτωσης:

- Στον πίνακα "house_info" προστέθηκαν 15 γραμμές.
- Στον πίνακα "calendar" προστέθηκαν 791 γραμμές.
- Στον πίνακα "house_consolidated" προστέθηκαν 8646 γραμμές.
- Στον πίνακα "halfhourly" προστέθηκαν 413630 γραμμές.
- Στον πίνακα "weather" προστέθηκαν 882 γραμμές.

```
CREATE DATABASE
Creating database households
CREATE DATABASE
create schema households
CREATE SCHEMA
create table house_info, calendar and house_consolidated
CREATE TABLE
CREATE TABLE
CREATE TABLE
insert data into table house_info, calendar and house_consolidated
COPY 15
COPY 791
COPY 8646
```

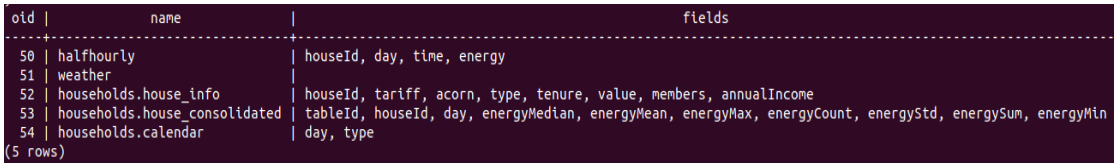
Εικόνα 26: Επιτυχής δημιουργία σχεσιακού μοντέλου συστήματος και εισαγωγή δεδομένων.



Εικόνα 27: Επιτυχής εισαγωγή δεδομένων στις SciDB και Accumulo ΒΔ.

4.4.1 Εκτέλεση Queries

Πριν τη διενέργεια ερωτήσεων προς το σύστημα χρειάζεται να ελεγχθεί αν ο κατάλογος εμπεριέχει στη σχεσιακή ΒΔ του τα αντικείμενα που δημιουργήθηκαν κατά τα προηγούμενα βήματα. Για το λόγο αυτό χρησιμοποιήθηκε ο Κώδικας12 (βλ. Appendix) ώστε να υπάρξει πρόσβαση στο docker container του καταλόγου *(bigdawg-postgres-catalog)* και συγκεκριμένα στη βάση με όνομα "bigdawg_catalog". Στη συνέχεια μέσω του Query0 (βλ. Appendix) επιβεβαιώνεται η ύπαρξη των αντικειμένων. Στο επόμενο στιγμιότυπο διακρίνονται οι νέοι πίνακες καθώς και τα γνωρίσματά τους:



Εικόνα 28: Αποτέλεσμα Query0.

Με δεδομένο ότι ο κατάλογος είναι ενήμερος για την ύπαρξη των νέων πινάκων, το ενδιάμεσο λογισμικό μπορεί να ανταποκριθεί σε ερωτήματα των χρηστών. Για την εκτέλεση οποιουδήποτε ερωτήματος μέσω του ενδιάμεσου λογισμικού πρέπει να ακολουθείται η παρακάτω σύνταξη:

$$\$\ curl - X\ POST - d\ \text{"<query>"}\ \text{localhost:8080/bigdawg/query/}$$

Η παραπάνω σύνταξη επιτρέπει την αποστολή του query σε μορφή String μέσω ενός Post αιτήματος στο μονοπάτι "bigdawg/query/". Εκεί βρίσκεται ο HTTP εξυπηρετητής Query Endpoint που αναλαμβάνει να μεταφέρει το ερώτημα στο ενδιάμεσο λογισμικό αλλά και να επιστρέψει πίσω στο χρήστη το

τελικό αποτέλεσμα. Η σύνδεση αυτή γίνεται μέσω του docker το οποίο χρησιμοποιεί το port 8080 στην τοπική IP διεύθυνση μηχανής με όνομα “localhost”.

4.4.1.1 PostgreSQL

Τα ερωτήματα PostgreSQL στο περιβάλλον του BigDAWG εσωκλείονται αυτούσια εντός της συνάρτησης:

$$bdrel(< PostgreSQL\ query >)$$

Query1 (βλ. Appendix)

Εμφάνιση των πρώτων 5 γραμμών του πίνακα “house_info”.

houseid	tariff	acorn	type	tenure	value	members	annualincome
3613	Std	affluent	Detached house	Owned outright	1200000.0	5	145000.0
4054	ToU	affluent	Bungalow	Mortgaged	780000.0	3	97500.0
3358	ToU	affluent	Detached house	Mortgaged	1380000.0	4	198000.0
1395	Std	affluent	Detached house	Owned outright	1530000.0	4	141000.0
3907	Std	affluent	Detached house	Owner occupied	855000.0	3	112000.0

Εικόνα 29: Αποτέλεσμα Query1.

Query2 (βλ. Appendix)

Εμφάνιση της συνολικής ενεργειακής κατανάλωσης όλων των νοικοκυριών ανά ημέρα, κατά αύξουσα σειρά ημερομηνίας *(από την παλαιότερη στην πιο πρόσφατη)*.

sum	day
4.186	2012-02-10
20.324	2012-02-11
28.225	2012-02-12
17.109	2012-02-13
14.868	2012-02-14

Εικόνα 30: Πρώτα πέντε αποτελέσματα του Query2.

241.073	2014-02-24
254.49399	2014-02-25
240.382	2014-02-26
229.312	2014-02-27
3.8209999	2014-02-28

Εικόνα 31: Τελευταία πέντε αποτελέσματα του Query2.

Query3 (βλ. Appendix)

Εμφάνιση του μέσου της ηλεκτρικής κατανάλωσης ενέργειας όλων των νοικοκυριών ανά κατηγορία "tenure".

tenure	avg
Owned outright	19.410387769280074
Social Renting	11.331155311115108
Shared Ownership	8.576849276142728
Owner occupied	17.557627893150944
Mortgaged	12.013653114008406

Εικόνα 32: Αποτέλεσμα Query3.

Query4 (βλ. Appendix)

Εμφάνιση κωδικού, οικονομικής κατάστασης, μέγιστης και ελάχιστης κατανάλωσης ενέργειας για τα σπίτια τα οποία έχουν εισόδημα μεγαλύτερο από 40000 την ημέρα των Χριστουγέννων του 2013.

houseid	acorn	annualincome	energymax	energymin	type
3358	affluent	198000.0	1.103	0.075	Christmas Day
1395	affluent	141000.0	1.943	0.214	Christmas Day
1969	Comfortable	60000.0	0.226	0.016	Christmas Day
4130	Comfortable	120000.0	1.495	0.201	Christmas Day
3796	Comfortable	77000.0	0.325	0.024	Christmas Day
3613	affluent	145000.0	3.631	0.219	Christmas Day
3907	affluent	112000.0	2.579	0.337	Christmas Day
3882	Comfortable	48500.0	0.095	0.006	Christmas Day
4054	affluent	97500.0	0.684	0.346	Christmas Day

Εικόνα 33: Αποτέλεσμα Query4.

Query5 (βλ. Appendix)

Εμφάνιση της συνολικής κατανάλωσης ηλεκτρικής ενέργειας ανά κατηγορία οικονομικής κατάστασης από το 2013 και μετά.

acorn	sum
affluent	50728.56
Comfortable	18375.99
Adversity	20725.285

Εικόνα 34: Αποτέλεσμα Query5.

Query6 (βλ. Appendix)

Εμφάνιση της μέγιστης ημερήσιας κατανάλωσης ηλεκτρικής ενέργειας, οικονομικής κατάστασης και αριθμού ενοίκων για τα σπίτια των οποίων η αντικειμενική αξία είναι μικρότερη ή ίση από 100000. Το αποτέλεσμα να αποτυπωθεί κατά φθίνουσα σειρά του κωδικού κάθε σπιτιού.

houseid	acorn	members	max
5364	Adversity	1	28.502
2717	Adversity	1	15.501
2305	Adversity	1	27.272

Εικόνα 35: Αποτέλεσμα Query6.

4.4.1.2 SciDB

Τα ερωτήματα SciDB στο περιβάλλον του BigDAWG εσωκλείονται αυτούσια εντός της συνάρτησης:

$$bdarray(< AFL\ query >)$$

Query7 (βλ. Appendix)

Εμφάνιση της συνολικής κατανάλωσης ηλεκτρικής ενέργειας για το σπίτι με houseId = 3613.

i	energy_sum
0	16724.290996799933

Εικόνα 36:Αποτέλεσμα Query7.

Query8 (βλ. Appendix)

Εμφάνιση των μετρήσεων κατανάλωσης ενέργειας του σπιτιού με houseId = 1395 για όλες τις ημέρες στο χρονικό διάστημα μεταξύ 17:30:00 - 18:00:00 το απόγευμα.

i	houseId	day	time	energy
177297	1395	2012-05-17	18:00:00	0.907
177345	1395	2012-05-18	18:00:00	0.301
177393	1395	2012-05-19	18:00:00	0.472
177441	1395	2012-05-20	18:00:00	0.703
177489	1395	2012-05-21	18:00:00	1.056
177537	1395	2012-05-22	18:00:00	0.43
177585	1395	2012-05-23	18:00:00	0.768
177633	1395	2012-05-24	18:00:00	0.621
177681	1395	2012-05-25	18:00:00	1.0700001
177729	1395	2012-05-26	18:00:00	0.346

Εικόνα 37: Πρώτα 10 αποτελέσματα του Query8.

208111	1395	2014-02-18	18:00:00	0.627
208159	1395	2014-02-19	18:00:00	0.723
208207	1395	2014-02-20	18:00:00	1.335
208255	1395	2014-02-21	18:00:00	0.521
208303	1395	2014-02-22	18:00:00	0.786
208351	1395	2014-02-23	18:00:00	1.068
208399	1395	2014-02-24	18:00:00	0.656
208447	1395	2014-02-25	18:00:00	0.954
208495	1395	2014-02-26	18:00:00	1.209
208543	1395	2014-02-27	18:00:00	0.641

Εικόνα 38: Τελευταία 10 αποτελέσματα του Query8.

Query9 (βλ. Appendix)

Εμφάνιση του μέσου όρου ενεργειακής κατανάλωσης όλων των νοικοκυριών, καθώς και της μέγιστης και ελάχιστης μέτρησης που καταγράφηκε για οποιοδήποτε χρονικό διάστημα εντός μισής ώρας.

i	energy_avg	energy_max	energy_min
0	0.30533807668866303	4.3130002	0.0

Εικόνα 39: Αποτέλεσμα Query9.

4.4.1.3 Accumulo

Τα ερωτήματα Accumulo στο περιβάλλον του BigDAWG εσωκλείονται αυτούσια εντός της συνάρτησης:

$$bdtext(< accumulo\ query >)$$

Query10 (βλ. Appendix)

Εμφάνιση όλων των εγγραφών του πίνακα “weather”.

2011-11-01	temperature range:9.68-15.57	Partly cloudy until evening.
2011-11-02	temperature range:8.88-15.19	Partly cloudy throughout the day.
2011-11-03	temperature range:12.79-17.41	Partly cloudy throughout the day.
2011-11-04	temperature range:11.53-15.54	Foggy overnight.
2011-11-05	temperature range:10.17-13.94	Foggy in the morning.

Εικόνα 40: Πρώτα πέντε αποτελέσματα του Query10.

```
2014-03-27      temperature range:2.71-10.04    Foggy overnight.

2014-03-28      temperature range:3.85-12.43    Foggy in the morning.

2014-03-29      temperature range:7.83-18.47    Clear throughout the day.

2014-03-30      temperature range:8.53-19.82    Mostly cloudy starting in the afternoon
2014-03-31      temperature range:10.98-16.37   Mostly cloudy until evening.
```

Εικόνα 41: Τελευταία πέντε αποτελέσματα του Query10.

Query11 (βλ. Appendix)

Εμφάνιση του εύρους θερμοκρασίας και της ανασκόπησης δελτίου καιρού για το πρώτο 15ήμερο του Ιανουαρίου για το έτος 2014.

```
2014-01-01      temperature range:6.59-11.34    Breezy until evening.

2014-01-02      temperature range:6.54-10.75    Partly cloudy until evening and breezy overnight.

2014-01-03      temperature range:7.01-10.49    Partly cloudy until evening and breezy in the evening.

2014-01-04      temperature range:4.06-9.58     Mostly cloudy until evening.

2014-01-05      temperature range:1.72-11.96    Mostly cloudy until evening and breezy starting in the evening.

2014-01-06      temperature range:9.34-12.57    Mostly cloudy until evening and breezy starting in the afternoon.

2014-01-07      temperature range:8.98-11.38    Breezy in the morning and partly cloudy until evening.

2014-01-08      temperature range:7.37-11.62    Mostly cloudy until evening.

2014-01-09      temperature range:4.86-12.01    Breezy in the morning and partly cloudy until evening.

2014-01-10      temperature range:2.68-10.14    Mostly cloudy until evening.

2014-01-11      temperature range:1.02-8.59     Partly cloudy until afternoon.

2014-01-12      temperature range:0.38-8.91     Overcast starting in the afternoon
2014-01-13      temperature range:5.42-9.72     Partly cloudy until evening.

2014-01-14      temperature range:2.05-6.82     Partly cloudy until evening.

2014-01-15      temperature range:6.41-11.19    Mostly cloudy until evening.
```

Εικόνα 42: Αποτελέσματα Query11.

5

Αξιολόγηση

Σε αυτό το κεφάλαιο θα παρουσιαστούν ορισμένες επιπλέον δυνατότητες αλλά και μερικά αδύνατα σημεία της εφαρμογής, ποιοτικά δηλαδή ευρήματα τα οποία προέκυψαν κατά το στάδιο της πειραματικής διαδικασίας, μέσα από την εκτέλεση queries προς το σύστημα. Στη συνέχεια, θα πραγματοποιηθεί συγκριτική αξιολόγηση μεταξύ της υλοποίησης της μελέτης περίπτωσης μέσω του BigDAWG, και μέσω μίας αυτόνομης σχεσιακής βάσεις δεδομένων.

5.1 Πλεονεκτήματα και μειονεκτήματα του BigDAWG

5.1.1 Migration δεδομένων εφαρμογής

Ενδεχομένως, μία από τις μεγαλύτερες καινοτομίες του BigDAWG σχετίζεται με τη δυνατότητα μεταφοράς *(migration)* των δεδομένων μεταξύ των διαφορετικών storage engines, καθώς όπως υποστηρίζεται από τη βιβλιογραφία, η απόδοση του χρόνου εκτέλεσης πανομοιότυπων ερωτημάτων διαφέρει ανάλογα με το storage engine που χρησιμοποιείται. Παρακάτω ακολουθούν δύο παραδείγματα:

Query12 (βλ. Appendix)

Μεταφορά των δεδομένων για το σπίτι με houseId = 3358 του πίνακα “halfhourly”, από τη ΒΔ SciDB σε έναν πίνακα με το όνομα “table12” στην PostgreSQL ΒΔ. Κατόπιν, μέσω PostgreSQL ερωτήματος, εμφάνιση του μέσου όρου κατανάλωσης ενέργειας ανά μισάωρο για το σπίτι αυτό, για το έτος 2013, ταξινομημένο κατά αύξουσα σειρά ημερομηνίας.

```
avg     day
0.2568958333333334      2013-01-01
0.207375        2013-01-02
0.18270833333333333     2013-01-03
0.12770833333333337     2013-01-04
0.10010416666666666     2013-01-05
```

Εικόνα 43: Πρώτα πέντε αποτελέσματα Query12.

```
0.2182291666666666      2013-12-27
0.19929166666666664     2013-12-28
0.17531249999999998     2013-12-29
0.21131250000000004     2013-12-30
0.22712499999999994     2013-12-31
```

Εικόνα 44: Τελευταία πέντε αποτελέσματα Query12.

Query13 (βλ. Appendix)

Μεταφορά των δεδομένων για το σπίτι με houseId = 2305 του πίνακα "house_info" από την PostgreSQL ΒΔ, σε ένα πίνακα με όνομα 'table' στην Accumulo ΒΔ. Κατόπιν, εμφάνιση όλων των διαθέσιμων πληροφοριών για το σπίτι αυτό.

```
5       1:houseid       2305
5       2:tariff        Std
5       3:acorn Adversity
5       4:type  Flat
5       5:tenure        Social Renting
5       6:value 77000.0
5       7:members       1
5       8:annualincome  19800.0
```

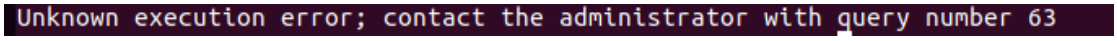
Εικόνα 45: Αποτέλεσμα Query13.

5.1.2 Αδυναμία εκτέλεσης ερωτημάτων

Κατά το στάδιο του πειραματισμού με την polystore εφαρμογή, παρατηρήθηκαν αδυναμίες του συστήματος, οι οποίες δεν επιτρέπουν την ορθή εκτέλεση ορισμένων queries. Στην περίπτωση της σχεσιακής βάσης δεδομένων, το βασικότερο πρόβλημα που εντοπίζεται αφορά την έλλειψη της δυνατότητας για μετονομασία μίας στήλης ενός query αποτελέσματος με τη χρήση ψευδώνυμου *(alias)*. Ως αποτέλεσμα, δεν μπορούν να πραγματοποιηθούν ταξινομήσεις αποτελεσμάτων, μέσα από στήλες οι οποίες προέρχονται από συναθροιστικές συναρτήσεις (π.χ. sum, count, average, min, max και standard deviation), μέσω εντολών όπως η "Order By". Η προηγούμενη παρατήρηση μπορεί να γίνει κατανοητή μέσω του ερωτήματος που ακολουθεί:

Query14 (βλ. Appendix)

Εμφάνιση των πέντε ημερομηνιών για τις οποίες σημειώθηκε η υψηλότερη ηλεκτρική κατανάλωση συνολικά για όλα τα νοικοκυριά.

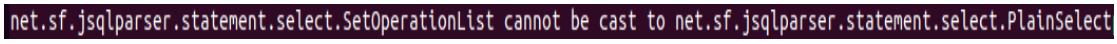


Εικόνα 46: Αποτυχία εκτέλεσης Query14.

Παράλληλα, για τις σχεσιακές βάσεις δεν υποστηρίζονται και "Union" εντολές:

Query15 (βλ. Appendix)

Εμφάνιση των σπιτιών για τα οποία η συνολική κατανάλωση ενέργειας είναι μικρότερη από 4200 κιλοβατώρες, καθώς και εκείνων των σπιτιών που ανήκουν στην οικονομική κατάσταση "Adversity".



Εικόνα 47: Αποτυχία εκτέλεσης Query15.

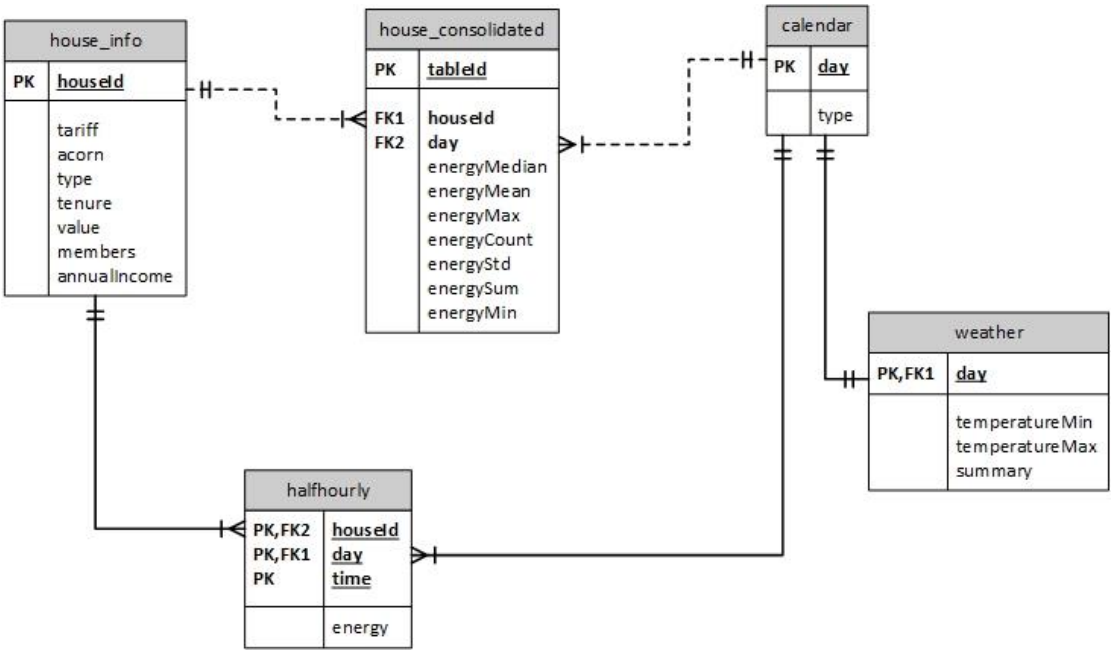
5.2 Ορισμός παραμέτρου και μεθόδου αξιολόγησης

Σκοπός της αξιολόγησης, εκτός από την εύρεση δυνατών και αδύνατων σημείων της εφαρμογής, είναι ο ποσοτικός έλεγχος της απόδοσης του BigDAWG συστήματος ως προς την εκτέλεση ερωτημάτων, σε σχέση με την εκτέλεση αντίστοιχων queries από την πλευρά ενός αυτόνομου storage engine. Η μετρική που θα χρησιμοποιηθεί για την απόδοση ενός query είναι αυτή του χρόνου εκτέλεσής του, ο οποίος εκφράζεται σε δευτερόλεπτα *(seconds)*.

Πρέπει να τονιστεί ότι ο έλεγχος της απόδοσης των queries για το BigDAWG θα πραγματοποιηθεί για σενάρια χρήσης μίας μηχανής αποθήκευσης για κάθε ερώτημα *(single storage engine scenario)*, δίχως δηλαδή διαδικασίες migration των δεδομένων μεταξύ των ΒΔ του συστήματος. Σε πρώτο στάδιο, η αυτόνομη βάση δεδομένων που θα χρησιμοποιηθεί για τη σύγκριση της απόδοσης των ερωτημάτων θα είναι ένα στιγμιότυπο της PostgreSQL ΒΔ. Τέλος, θα εξεταστεί και ο επιπρόσθετος χρόνος εκτέλεσης μεμονωμένων ερωτημάτων και πάλι από το BigDAWG, σε αντιδιαστολή με τις SciDB και Accumulo μηχανές αποθήκευσης.

5.3 Οργάνωση πειράματος ελέγχου απόδοσης συστημάτων

Αρχικά, για την οργάνωση των πειραμάτων ελέγχου απόδοσης, δημιουργήθηκε μία ξεχωριστή PostgreSQL βάση δεδομένων, η οποία περιλαμβάνει όλα τα δεδομένα της μελέτης περίπτωσης υπό την μορφή συσχετισμένων πινάκων. Το όνομα της βάσης αυτής είναι "test_households", για την οποία ορίζεται το schema "households". Παρακάτω ακολουθεί η αναπαράσταση του σχεσιακού μοντέλου για τη νέα ΒΔ "test_households":



Εικόνα 48: Σχεσιακό μοντέλο αναπαράστασης των δεδομένων της μελέτης περίπτωσης από ένα στιγμιότυπο PostgreSQL.

Συγκριτικά με το σχεσιακό σχήμα της Εικόνας 20, προστέθηκαν οι δύο πίνακες "halfhourly" και "weather", οι οποίοι στην περίπτωση του BigDAWG ανήκουν στις μηχανές αποθήκευσης SciDB και Accumulo αντίστοιχα. Αναλυτικότερα για τους δύο νέους πίνακες ισχύει ότι:

- **halfhourly**: Ως πρωτεύον κλειδί ορίζεται ο συνδυασμός των στηλών "houseId", "day" και "time". Η ένωση με τους πίνακες "house_info" και "calendar" επιτυγχάνεται μέσα από τις αντίστοιχες σχέσεις πρωτεύοντος και ξένου κλειδιού.
- **weather:** Ως πρωτεύον κλειδί για αυτόν τον πίνακα ορίζεται η στήλη "day", μέσω της οποίας επιτυγχάνεται η ένωση με τον πίνακα "calendar".

Τα ερωτήματα βάσει των οποίων θα γίνει ο έλεγχος της απόδοσης παρουσιάστηκαν στην ενότητα 4.3 (Query1 – Query11). Αξίζει να σημειωθεί, ότι τα Query7 έως Query11 που είναι συμβατά με SciDB και Accumulo storage engines, μετασχηματίστηκαν αναλόγως έτσι ώστε να εκτελεστούν στη γλώσσα PostgreSQL της ανεξάρτητης βάσης δεδομένων.

Παράλληλα, για τον εμπλουτισμό των συμπερασμάτων ως προς την απόδοση του polystore, δημιουργήθηκαν δύο διακριτές και ξεχωριστές από το BigDAWG βάσεις δεδομένων σε SciDB και Accumulo, για τις οποίες προστέθηκαν τα αντίστοιχα δεδομένα της μελέτης περίπτωσης (*ο πίνακας "halfhourly" στην SciDB και ο πίνακας "weather" στην Accumulo*).

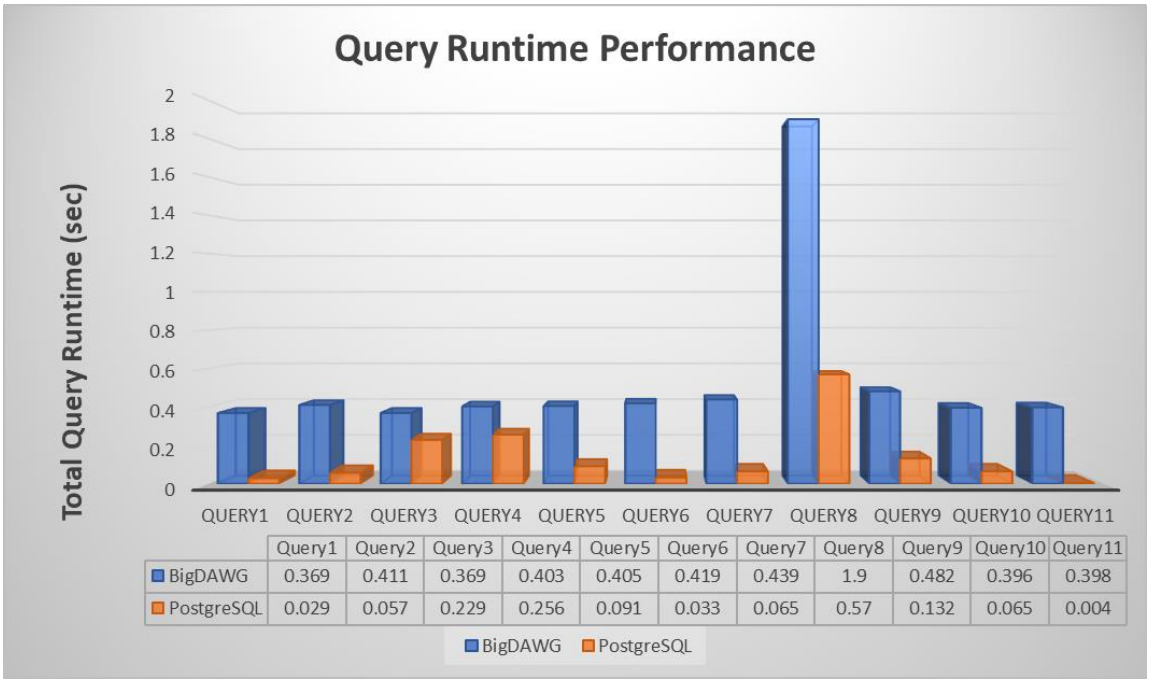
Στο σημείο αυτό θα πρέπει να τονιστεί ότι η εκτέλεση κάθε ερωτήματος προσομοιώθηκε πολλαπλές φορές, ώστε να δημιουργηθεί ένας αντιπροσωπευτικός μέσος όρος του χρόνου εκτέλεσής του. Επιπλέον, ο αριθμός των ερωτημάτων που εκφράστηκαν μέσα από τις βάσεις δεδομένων SciDB και

Accumulo είναι περιορισμένος, λόγω της αδυναμίας του BigDAWG να ενσωματώσει πλήρως τη λειτουργικότητα κάθε μηχανής αποθήκευσης.

5.4 Αποτελέσματα ελέγχου

5.4.1 Σύγκριση BigDAWG και PostgreSQL

Στο διάγραμμα που ακολουθεί αποτυπώνονται οι χρόνοι που καταγράφηκαν μετά την εκτέλεση των ερωτημάτων:

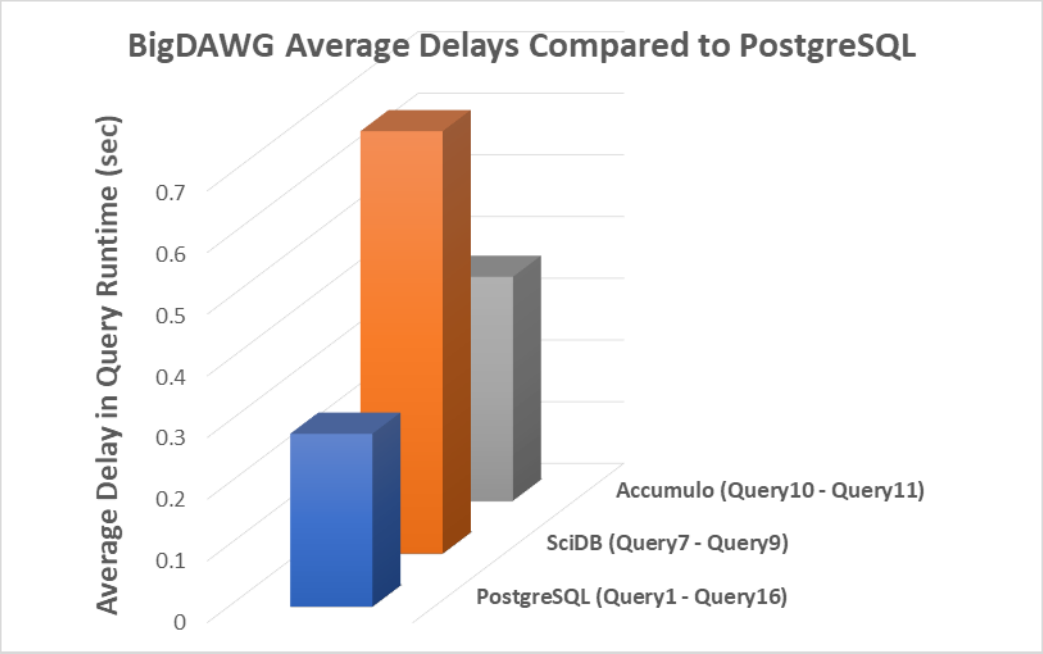


	Query1	Query2	Query3	Query4	Query5	Query6	Query7	Query8	Query9	Query10	Query11
BigDAWG	0.369	0.411	0.369	0.403	0.405	0.419	0.439	1.9	0.482	0.396	0.398
PostgreSQL	0.029	0.057	0.229	0.256	0.091	0.033	0.065	0.57	0.132	0.065	0.004

Διάγραμμα 1: Χρόνοι εκτέλεσης ερωτημάτων μεταξύ BigDAWG και PostgreSQL.

Σύμφωνα με το Διάγραμμα 1, προκύπτει ότι ο χρόνος εκτέλεσης των ερωτημάτων μέσω του BigDAWG, είναι μεγαλύτερος, περίπου κατά 0.405 δευτερόλεπτα κατά μέσο όρο συγκριτικά με το PostgreSQL στιγμιότυπο. Η διαπίστωση θεωρείται αναμενόμενη, καθώς μεταξύ του χρήστη που θέτει τα ερωτήματα και του storage engine, παρεμβάλλεται το ενδιάμεσο λογισμικό της εφαρμογής. Λαμβάνοντας υπόψιν ότι η μονάδα μέτρησης εκφράζεται σε δευτερόλεπτα, οι αποκλίσεις που σημειώθηκαν μεταξύ των δύο περιπτώσεων δεν θεωρούνται μεγάλες, καθώς είναι της τάξης του millisecond.

Στο επόμενο διάγραμμα, διαφαίνεται ο μέσος χρόνος καθυστέρησης της υλοποίησης των ίδιων ερωτημάτων από την polystore εφαρμογή σε σύγκριση με την PostgreSQL, ανά μηχανή αποθήκευσης:



Διάγραμμα 2:Μέσος χρόνος καθυστέρηση της εκτέλεσης ερωτημάτων του BigDAWG σε σχέση με την PostgreSQL ΒΔ, ανά μηχανή αποθήκευσης.

Το Διάγραμμα 2, υποδεικνύει ότι η μεγαλύτερη διαφορά στην εκτέλεση των ερωτημάτων κατά μέσο όρο, προέρχεται από queries που απευθύνονται στη SciDB ΒΔ, η οποία εμφανίζεται πιο αργή κατά 0.68 δευτερόλεπτα. Ακολουθούν η Accumulo βάση δεδομένων με καθυστέρηση κατά μέσο όρο 0.36 δευτερολέπτων, ενώ η μικρότερη απόκλιση στην απόδοση σημειώθηκε για τη σύγκριση μεταξύ της ενσωματωμένης PostgreSQL στο BigDAWG και της αντίστοιχης ανεξάρτητης ΒΔ με 0.28 δευτερόλεπτα.

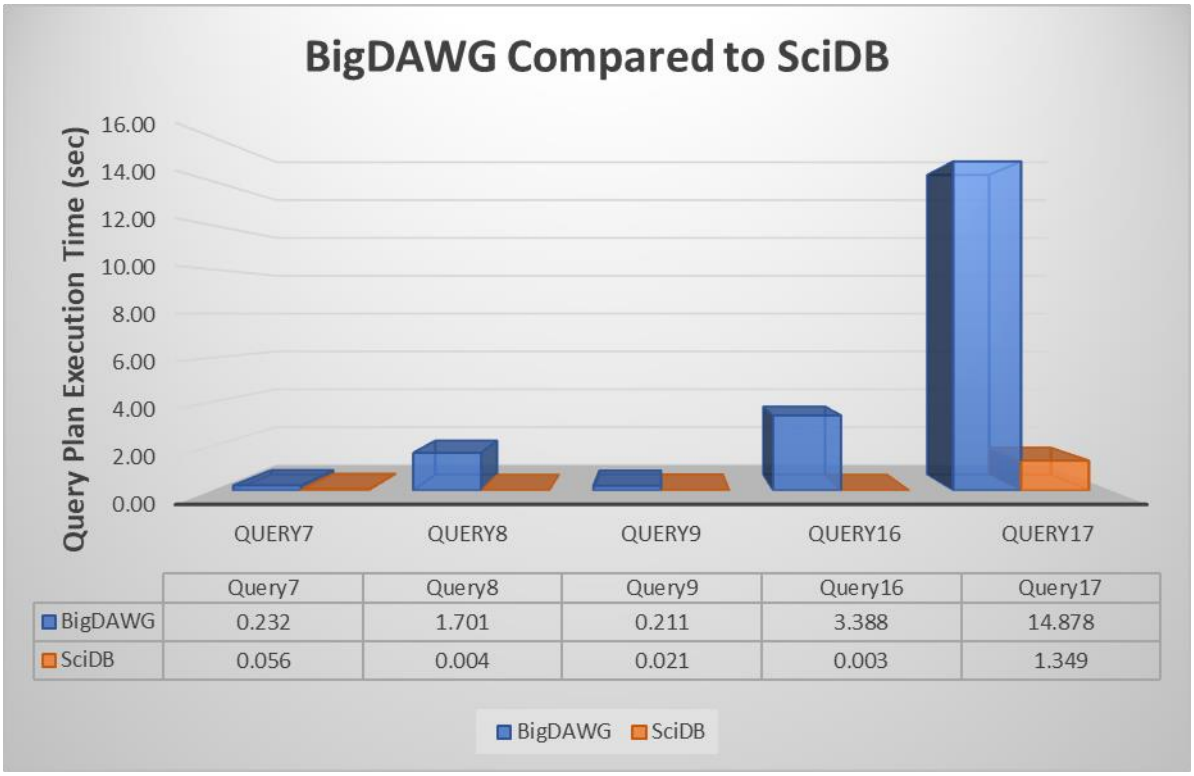
Όπως αναφέρεται και στην ενότητα 3.2, το ενδιάμεσο λογισμικό του BigDAWG αποτελείται από επιμέρους εργαλεία, κάθε ένα από τα οποία καταναλώνει συγκεκριμένο χρόνο για την εκτέλεση των διεργασιών του, ενώ το χρονικό άθροισμα που προκύπτει αντανακλά τον ολικό χρόνο εκτέλεσης ενός query. Στον πίνακα που ακολουθεί διακρίνεται η ανάλυση των ερωτημάτων του πειράματος μέσω της εκτέλεσης από το BigDAWG, διασπασμένη ως προς το χρόνο που αναλογεί για:

- Τη σχεδίαση του πλάνου εκτέλεσης ερωτήματος *(planning time)*.
- Την πραγματική εκτέλεση του ερωτήματος μέσα από το πλάνο εκτέλεσης *(execution time)*.
- Άλλες διεργασίες όπως είναι η καταγραφή του χρόνου εκτέλεσης παρελθοντικών ερωτημάτων από το monitor λογισμικό *(other)*.

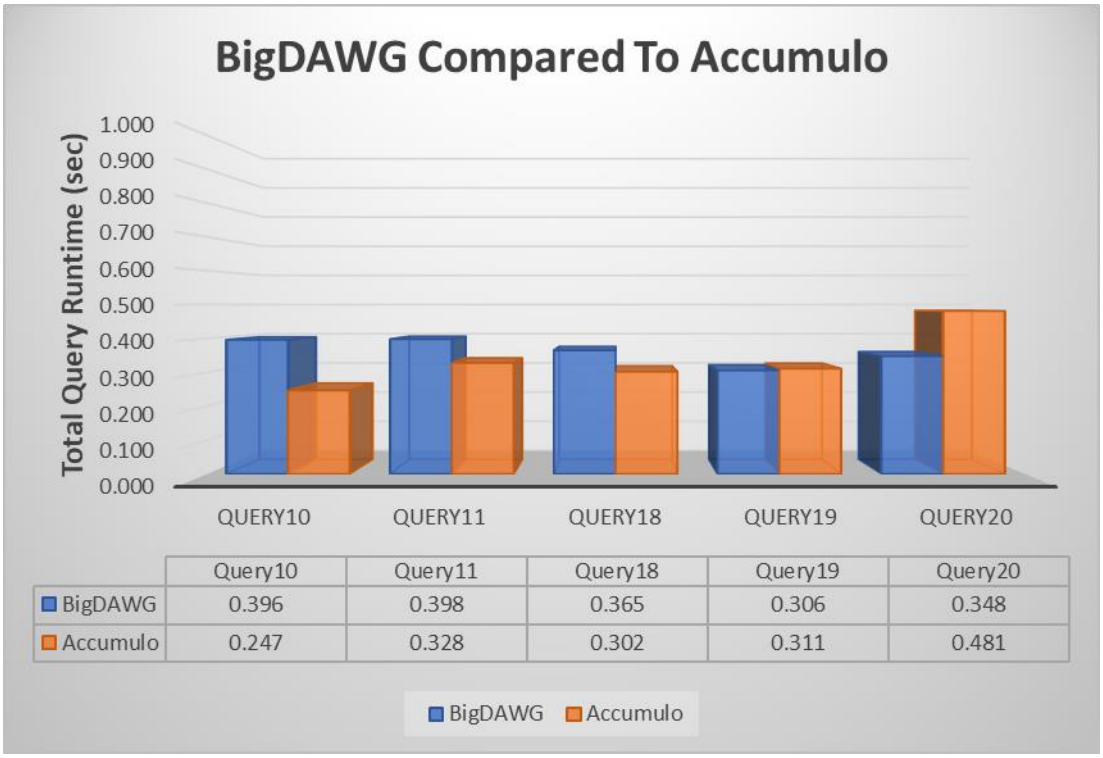
Query	Planning	Execution	Other	Total
Query1	0.162	0.119	0.088	0.369
Query2	0.212	0.085	0.114	0.411
Query3	0.177	0.103	0.089	0.369
Query4	0.231	0.106	0.066	0.403
Query5	0.233	0.077	0.095	0.405
Query6	0.252	0.09	0.077	0.419
Query7	0.123	0.232	0.084	0.439
Query8	0.156	1,701	0.043	1.9
Query9	0.217	0.211	0.054	0.482
Query10	0.175	0.172	0.049	0.396
Query11	0.16	0.118	0.055	0.398

Πίνακας 8: Ανάλυση χρόνου εκτέλεσης ερωτημάτων από το BigDAWG.

Η τελευταία σύγκριση, αφορά τους χρόνους εκτέλεσης των ερωτημάτων μεταξύ της polystore εφαρμογής και των απομονωμένων βάσεων δεδομένων SciDB και Accumulo. Για τη SciDB ΒΔ, η σύγκριση αφορά τον *execution* χρόνο εκτέλεσης καθώς μόνο αυτή η πληροφορία είναι διαθέσιμη από το περιβάλλον της, ενώ για την Accumulo βάση θα γίνει σύγκριση του ολικού χρόνου εκτέλεσης των ερωτημάτων. Τα αποτελέσματα των μετρήσεων είναι διαθέσιμα στα δύο διαγράμματα που ακολουθούν:



Διάγραμμα 3: Σύγκριση query execution time μεταξύ BigDAWG και SciDB.



Διάγραμμα 4: Σύγκριση εκτέλεσης ερωτημάτων μεταξύ BigDAWG και Accumulo (total runtime time).

Αν και πρόκειται για περιορισμένο αριθμό ερωτημάτων, είναι εμφανής η μεγάλη διαφορά στους χρόνους εκτέλεσης μεταξύ του BigDAWG και του SciDB storage engine, όσον αφορά το χρόνο εκτέλεσης των ερωτημάτων. Χαρακτηριστικό παράδειγμα αποτελεί το Query17, το οποίο στην αυτόνομη SciDB μηχανή αποθήκευσης υλοποιήθηκε ταχύτερα κατά περίπου 13.5 δευτερόλεπτα. Αντίθετα, για τη σύγκριση με την Accumulo ΒΔ, οι αποκλίσεις στην απόδοση της εκτέλεσης των ερωτημάτων δεν παρουσιάζουν μεγάλες διαφορές, ενώ παρατηρήθηκαν δύο ερωτήματα (Query19, Query20), όπου το polystore σύστημα απέδωσε ταχύτερους χρόνους.

5.5 Σύνοψη συμπερασμάτων αξιολόγησης

Μετά την εκτέλεση των ερωτημάτων αλλά και την παρουσίαση των ευρημάτων που προέκυψαν, μπορούν να εξαχθούν τα παρακάτω συμπεράσματα:

1. Η σημαντικότερη ίσως παροχή του BigDAWG, η οποία εμπίπτει στα πλαίσια της θεωρίας των polystore συστημάτων, είναι αυτή της δυνατότητας για μεταφορά των δεδομένων μεταξύ των μηχανών αποθήκευσης του συστήματος. Αν και δεν πραγματοποιήθηκαν έλεγχοι απόδοσης για σενάρια πολλαπλής χρήσης μηχανών αποθήκευσης υπό το πρίσμα του ίδιου ερωτήματος, είναι βέβαιο ότι καμία αυτοτελής ΒΔ δεν μπορεί να μιμηθεί αυτή τη συμπεριφορά.
2. Το σύμπλεγμα του BigDAWG δεν έχει καταφέρει να ενσωματώσει πλήρως όλο το εννοιολογικό πλαίσιο *(semantics)* των storage engines που υποστηρίζει. Ως αποτέλεσμα, η εφαρμογή

αδυνατεί να υλοποιήσει βασικές εντολές ερωτημάτων (π.χ. Group By ‘alias’, Union) όσον αφορά την PostgreSQL. Το ίδιο ισχύει και για τις υπόλοιπες δύο βάσεις δεδομένων, των οποίων οι δυνατότητες δεν έχουν αξιοποιηθεί ακόμα στο έπακρο.

3. Όπως ήταν αναμενόμενο, από ποσοτικής άποψης, υπήρξε διαφορά ανάμεσα στο χρόνο εκτέλεσης των ερωτημάτων της μελέτης περίπτωσης μεταξύ του BigDAWG και του στιγμιότυπου της PostgreSQL. Κατά μέσο όρο η διαφορά αυτή υπολογίστηκε στα 0.405 δευτερόλεπτα *(405 ms)*, ενώ δε θεωρείται μεγάλη.
4. Κατά την περαιτέρω ανάλυση του χρόνου εκτέλεσης ερωτημάτων από την εφαρμογή, παρατηρήθηκε ότι ο χρόνος για την πραγματική εκτέλεση των ερωτημάτων (query plan execution time), είναι αρκετά μεγαλύτερος από ότι ο αντίστοιχος μέσω της SciDB. Όπως έχει προαναφερθεί και στο θεωρητικό υπόβαθρο για τη SciDB ΒΔ, αλλά και όπως παρατηρήθηκε από τις μετρήσεις που έγιναν, πρόκειται για μία ταχύτατη μηχανή αποθήκευσης όσον αφορά την επεξεργασία των δεδομένων της. Επομένως, η σύνδεση του BigDAWG με τη συγκεκριμένη βάση δεδομένων, χρήζει μεγάλα περιθώρια βελτίωσης.
5. Για την Accumulo μηχανή αποθήκευσης, οι χρόνοι εκτέλεσης των ερωτημάτων μέσω του BigDAWG θεωρούνται παραπλήσιοι. Ωστόσο, και σε αυτή την περίπτωση δεν έχει επιτευχθεί πλήρης συμβατότητα του polystore με όλες τις γλωσσικές εντολές της συγκεκριμένης βάσης δεδομένων.

6

Πρακτική άσκηση στον οργανισμό Nestlé Ελλάς

Στα πλαίσια του μεταπτυχιακού προγράμματος του τμήματος "Διοικητικής Επιστήμης και Τεχνολογίας" του Οικονομικού Πανεπιστημίου Αθηνών, παρέχεται η δυνατότητα προς τους σπουδαστές να συμμετάσχουν σε πρόγραμμα πρακτικής άσκησης με σκοπό την εξοικείωση και ένταξή τους στην αγορά εργασίας. Στο παρόν κεφάλαιο θα παρουσιαστούν πληροφορίες σχετικά με την πρακτική άσκηση που πραγματοποιήθηκε στην εταιρία "Nestlé Ελλάς".

6.1 Περιγραφή τμήματος εργασίας

Η εκπόνηση της πρακτική άσκησης, πραγματοποιήθηκε στο τμήμα Πληροφορικής του οργανισμού – "GR Information Technology", και πιο συγκεκριμένα στην ομάδα του "Business Analytics/BA". Ο στόχος της ομάδας, όπως και αυτός του τμήματος της Πληροφορικής, δεν είναι άλλος από την εφαρμογή και αξιοποίηση προηγμένων τεχνολογικών λύσεων προς το σύνολο της εταιρίας, με σκοπό την δημιουργία ανταγωνιστικού πλεονεκτήματος το οποίο θα βοηθήσει τον οργανισμό να επικρατήσει στην αγορά. Οι προσφερόμενες αυτές λύσεις κατευθύνουν τον οργανισμό προς τον ψηφιακό μετασχηματισμό, ενώ παράλληλα στα πλαίσια της εταιρικής κοινωνικής ευθύνης ιδιαίτερη βαρύτητα δίνεται και στην υποστήριξη πρωτοβουλιών σχετικά με την περιβαλλοντική βιωσιμότητα.

Για την επίτευξη των ανωτέρω στόχων, η ομάδα του Business Analytics έχει επωμιστεί με την ευθύνη της σχεδίασης και ανάπτυξης λύσεων αναλυτικής δεδομένων, κάνοντας χρήση εργαλείων επεξεργασίας δεδομένων όπως είναι το περιβάλλον του "SQL Server Management Studio (SSMS)", ενσωμάτωσης δεδομένων όπως το "SQL Server Integration Services (SSIS)", αλλά και αναπαράστασης αυτών με χαρακτηριστικό παράδειγμα το πρόγραμμα "Microsoft Power BI". Μέσα από την αξιοποίηση των εργαλείων αυτών, εξάγονται χρήσιμα συμπεράσματα προς τα άμεσα ενδιαφερόμενα μέρη της εταιρίας, υποστηρίζοντας με αυτόν τον τρόπο μία "data-driven" διαδικασία λήψης επιχειρηματικών αποφάσεων. Τέλος, παρέχονται και υπηρεσίες τεχνικής υποστήριξης προς υπόλοιπα μέλη της εταιρίας, σχετικά με τα εργαλεία οπτικοποίησης των δεδομένων.

6.2 Περιγραφή θέσης εργασίας και δραστηριοτήτων

6.2.1 Business Intelligence Internship

Ο ρόλος μου στην εταιρία είχε τον τίτλο του "Business Intelligence Specialist" και αφορούσε το πεδίο της επιχειρηματικής ευφυΐας. Ειδικότερα, οι απαιτήσεις της θέσης αυτής διακρίνονται ως εξής:

- Στη σχεδίαση, ανάπτυξη και υλοποίηση λύσεων αναλυτικής δεδομένων μέσω των εργαλείων "SQL Server Management Studio" και "Microsoft Power BI".
- Στο μετασχηματισμό ανεπεξέργαστων δεδομένων *(raw data)* σε χρήσιμη και μετρήσιμη γνώση προς τον οργανισμό.
- Στη συμμετοχή σε εταιρικές συνεδριάσεις με άλλα μέλη της εταιρίας, με σκοπό την κατανόηση προβλημάτων και από κοινού ανταλλαγή ιδεών για την σχεδίαση και παροχή λύσεων.

Οι δεξιότητες που απαιτούνται για τη θέση αυτή, επεκτείνονται τόσο σε τεχνικό όσο και σε επιχειρησιακό *(business)* επίπεδο. Αρχικά, χρειάζεται κατανόηση της business ορολογίας του οργανισμού αλλά και επικοινωνιακές δεξιότητες, προκειμένου μέσα από εταιρικές συνεδριάσεις, να γίνονται αντιληπτές οι απαιτήσεις και ανάγκες των ενδιαφερόμενων μερών. Όσον αφορά το τεχνικό επίπεδο, είναι αναγκαία η βαθιά γνώση της θεωρίας των δεδομένων κυρίως μέσα από σχεσιακά μοντέλα, και κατ' επέκταση η εφαρμογή αυτής μέσω των προγραμματιστικών εργαλείων που χρησιμοποιούνται από το τμήμα Πληροφορικής. Ταυτόχρονα, σημαντικό μέρος της διαδικασίας υλοποίησης ενός έργου, είναι η εύρεση του αποδοτικότερου τρόπου εκτέλεσής του, αλλά και η κατόπιν συνεχής εξέλιξή του μέσα από προτεινόμενες βελτιώσεις.

6.2.2 Έργα, δραστηριότητες και προγραμματισμός στο τμήμα BA

Κατά τη διάρκεια της πρακτικής άσκησης, παρεχόταν συνεχής εκπαίδευση σε θέματα εσωτερικών κανονισμών και συμπεριφοράς, προκειμένου να διασφαλιστεί η ισότητα μεταξύ των εργαζομένων αλλά και η ομαλή λειτουργία της εταιρίας. Παράλληλα, η εκπαίδευση κάλυπτε και ένα ευρύ φάσμα τεχνικών και επιχειρηματικών γνώσεων, μεταξύ των οποίων περιοχές σχετικές με ανάλυση δεδομένων, κυβερνοασφάλεια, ψηφιακό μάρκετινγκ, πωλήσεις, περιβαλλοντική βιωσιμότητα και πολλά άλλα.

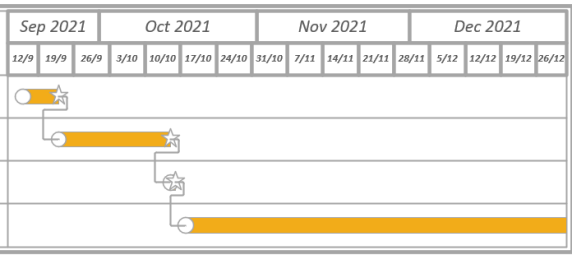
Επιπλέον, στο ίδιο διάστημα συμμετείχα σε τρία έργα, τα στάδια εκτέλεσης των οποίων μπορούν να διακριθούν στους παρακάτω πίνακες Gantt Chart.

Pricing Report

Το έργο αυτό αποτελεί ένα εβδομαδιαίο report οπτικοποίησης δεδομένων τιμοληψίας για τα προϊόντα της Nestle και του ανταγωνισμού. Με αυτόν τον τρόπο, ο οργανισμός αποκτά μία βελτιωμένη εικόνα σχετικά με τις διακυμάνσεις των τιμών των προϊόντων για την τελευταία εβδομάδα που προηγήθηκε.

Με βάση αυτή την πληροφόρηση ο οργανισμός μπορεί να αντιδράσει άμεσα στις αλλαγές που επικρατούν στην αγορά, αποφασίζοντας να επανεξετάσει τις τιμές των δικών του προϊόντων. Τέλος, η ανάλυση των δεδομένων είναι διαθέσιμη μέσα από ομαδοποιήσεις που μπορούν να γίνουν σε επίπεδο καταστημάτων λιανικής πώλησης, προϊοντικών κατηγοριών αλλά και ημερομηνιών.

ID	Pricing Project	Start	Finish	Duration
1	Collect Business Requirements (Planning & Analysis)	9/16/2021	9/22/2021	5d
2	Power BI Implementation (Design & Implementation)	9/23/2021	10/14/2021	16d
3	Project Presentation to Business Partners	10/15/2021	15/10/2021	1d
4	Evaluation and Enhancements (Maintenance)	18/10/2021	12/23/2022	310d



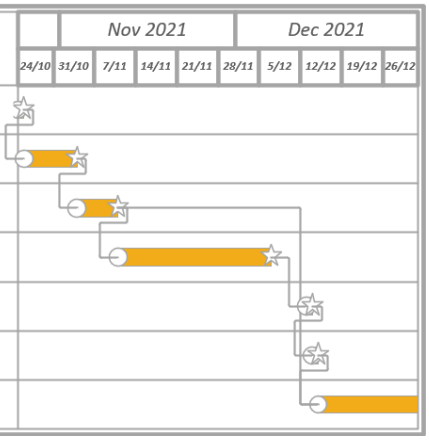
Πίνακας 9: Πίνακας Gantt-Chart για το έργο "Pricing Report".

Το αρχικό στάδιο της διαδικασίας υλοποίησης, περιλάμβανε τον καθορισμό των απαιτήσεων που τέθηκαν για την πληροφορία που απαιτείται να παρέχει το έργο. Για την υλοποίηση του έργου χρησιμοποιήθηκε το εργαλείο Power BI, μέσα από το οποίο δημιουργήθηκαν οι κατάλληλες μετρικές, συναθροίσεις και γραφήματα που καθορίστηκαν από το αίτημα των business partners του οργανισμού. Μετά και την επιτυχή παρουσίαση του πρώτου παραδοτέου σε όλα τα εμπλεκόμενα με το έργο μέλη της εταιρίας, πραγματοποιούνται διαδικασίες επανελέγχου και βελτιστοποίησης του project. Τέλος, μέρος των αρμοδιοτήτων μου έγκειται και στην εβδομαδιαία ανανέωση του έργου μέσα από επικοινωνίες με την εταιρία-πάροχο των ακατέργαστων δεδομένων.

Stock Cover Project

Το Stock Cover Project αφορά ένα αίτημα αυτοματοποίησης της διαδικασίας παρακολούθησης του αποθέματος από το τμήμα εφοδιαστικής αλυσίδας της εταιρίας "Supply Chain/SC". Στόχος είναι η παρακολούθηση του αποθέματος να μην πραγματοποιείται χειροκίνητα από το αρμόδιο τμήμα και επαναληπτικά για κάθε κατηγορία πρώτων υλών μέσω του εργαλείου "SAP", αλλά να παρέχεται αυτόματα με τη μορφή report από το Power BI. Το έργο αυτό, αναμένεται ότι εξοικονομεί στο τμήμα της εφοδιαστικής αλυσίδας περίπου πέντε ώρες ανά μήνα (δεδομένου του χρόνου που χρειαζόταν για το reporting από τα μέλη της ομάδας).

ID	Stock Cover Project	Start	Finish	Duration
1	Collect Business Requirements (Planning & Analysis)	10/25/2021	10/25/2021	1d
2	Bookmarks Mapping & Total Demand Value Validation (per Category)	10/26/2021	11/3/2021	7d
3	Stock Cover Calculation per Category	11/4/2021	11/10/2021	5d
4	Final Validation	11/11/2021	12/6/2021	18d
5	Presentation to Supply Chain Team	12/13/2021	12/13/2021	1d
6	Go Live Phase (Cloud Publication)	12/14/2021	12/14/2021	1d
7	Continuous Evaluation and Enhancements	12/15/2021	3/11/2022	63d



Πίνακας 10: Πίνακας Gantt Chart για το έργο "Stock Cover Project".

Στον Πίνακα 10 παρουσιάζονται τα βήματα υλοποίησης του Stock Cover Project. Μετά τον καθορισμό των απαιτήσεων από το SC, σχεδιάστηκε η στρατηγική υλοποίησης του έργου, η οποία στοχεύει στην άρτια σύνδεση μεταξύ του SAP BW και του Power BI, προκειμένου να δημιουργηθούν οι κατάλληλες μετρικές και τα γραφήματα απεικόνισης του αποθέματος. Η δυσκολία του εγχειρήματος σχετίζεται με τη συχνά ασταθή σύνδεση μεταξύ των δύο προγραμματιστικών εργαλείων, αλλά και την αντιστοίχιση συγκεκριμένων πεδίων (π.χ. κωδικός κατηγορίας προϊόντος, κωδικός εργοστασίου κ.τ.λ.) με τις κατηγορίες ενδιαφέροντος των προϊόντων που είχαν οριστεί κατά το στάδιο καθορισμού των απαιτήσεων. Στα επόμενα βήματα της διαδικασίας, απαιτούνταν η σύγκριση των μετρήσεων ανά κατηγορία προϊόντος μεταξύ του SAP BW και Power BI, τόσο σε επίπεδο μέτρησης της ζήτησης, όσο και αποθέματος, προκειμένου να επαληθευτεί η άρτια σύνδεση μεταξύ των δύο εργαλείων.

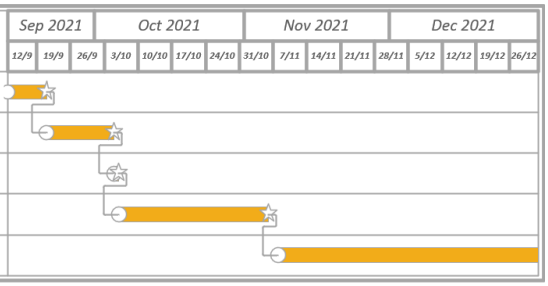
Ολοκληρώνοντας και τις τελικές επαληθεύσεις, το project παρουσιάστηκε στην ομάδα του SC, και εν συνεχεία έγινε διαθέσιμο μέσω της cloud πλατφόρμας του Power BI. Τέλος, συνεχίζονται εργασίες συντήρησης και βελτιώσεων, με βάση την ανατροφοδότηση που προέρχεται από το τμήμα εφοδιαστικής αλυσίδας.

Trade Promo Monitoring

Πρόκειται για ένα project το οποίο ακολουθεί μία κλασική ETL διαδικασία, στόχος της οποίας είναι η απόδοση μίας δυαδικής τιμής *(0/1)* σε κάθε προϊόν της εταιρίας αλλά και του ανταγωνισμού, στην περίπτωση για την οποία πραγματοποιήθηκε ή όχι κάποια προωθητική ενέργεια για το προϊόν αυτό, για συγκεκριμένες εβδομάδες και καταστήματα λιανικής πώλησης. Ως αποτέλεσμα, παράγεται ένας πίνακας με δυαδικές τιμές ο οποίος στη συνέχεια θα χρησιμοποιηθεί από το τμήμα "Demand and Supply Planning/DSP", και θα εισαχθεί στο εργαλείο "SAS Forecast Studio", με σκοπό την εκτέλεση προβλεπτικών μοντέλων. Μέσω της εκτέλεσης αυτών των μοντέλων, αφενός αξιολογείται με ακρίβεια η επίδραση των προωθητικών ενεργειών στις πωλήσεις των προϊόντων της εταιρίας και αφετέρου γίνεται

πρόβλεψη της εξέλιξης των πωλήσεων. Επιπροσθέτως, η πληροφορία που παράγεται από την προβλεπτική αυτή ανάλυση καθορίζει και τον σχεδιασμό της προωθητικής στρατηγικής του οργανισμού.

ID	Trade Promo Monitoring	Start	Finish	Duration
1	Collect Business Requirements (Planning & Analysis)	9/13/2021	9/20/2021	6d
2	Power BI Implementation (Design & Implementation)	9/21/2021	10/4/2021	10d
3	Project Presentation to Demand and Supply Planning Team	10/5/2021	10/5/2021	1d
4	Testing Period	10/6/2021	11/5/2021	23d
5	Project Population (Addition of More Businesses)	11/8/2021	3/11/2022	90d



Πίνακας 11: Πίνακας Gantt-Chart για το έργο "Trade Promo Monitoring".

Όπως και στα δύο προηγούμενα έργα, έτσι και σε αυτό, για τη διαδικασία της ολοκλήρωσής του περιλαμβάνονται κατά σειρά: ο καθορισμός των απαιτήσεων, το στάδιο της υλοποίησης στο Power BI, η παρουσίαση στην ομάδα που αιτήθηκε το έργο (σε αυτήν την περίπτωση DSP), αλλά και η περίοδος ελέγχου για την εύρεση πιθανών αστοχιών και βελτίωση των διεργασιών. Τέλος, για το έργο αυτό δημιουργούνται διαφορετικές εκδόσεις ανάλογα με την κατηγορία των προϊόντων ενδιαφέροντος (π.χ. ελληνικός καφές, δημητριακά, κάψουλες καφέ κ.α.), ενώ συχνά επαναπροσδιορίζονται και οι αντίστοιχες απαιτήσεις.

7

Επίλογος

Στο κεφάλαιο αυτό πραγματοποιείται η σύνοψη της εργασίας, ανακεφαλαιώνοντας τα βασικά συμπεράσματα που προέκυψαν από τα στάδια της υλοποίησης και αξιολόγησης. Επιπλέον, διατυπώνονται πιθανές μελλοντικές επεκτάσεις.

7.1 Σύνοψη και συμπεράσματα

Αναγνωρίζοντας την πρόσφατη εμφάνιση μίας πολλά υποσχόμενης λύσης με το όνομα polystores στον τομέα της ενοποίησης ετερογενών δεδομένων *(data integration)*, η παρούσα εργασία επιχείρησε, αφενός να διαλευκάνει και να εξερευνήσει σε θεωρητικό επίπεδο τα προτερήματά τους έναντι των προγενέστερων προσεγγίσεων, και αφετέρου να εξοικειώσει τον αναγνώστη με ένα συγκεκριμένο τύπο τέτοιου συστήματος που ονομάζεται BigDAWG. Για το λόγο αυτό, και καθώς πρόκειται για μία ιδιαίτερα πολύπλοκη εφαρμογή διαχείρισης ετερογενών βάσεων δεδομένων, αναλύθηκαν οι τεχνολογίες (polystores, PostgreSQL, SciDB, Accumulo, Docker) που την απαρτίζουν, η κατανόηση των οποίων είναι απαραίτητη για τον οποιοδήποτε πειραματισμό με αυτή. Στη συνέχεια, παρουσιάστηκε μία μεθοδολογία εκχώρησης των δεδομένων του χρήστη στην εφαρμογή, μέσω μίας μελέτης περίπτωσης, το σενάριο της οποίας είναι ιδανικό για την υλοποίηση μέσω ενός polystore συστήματος.

Μετά την υλοποίηση της μεθοδολογίας, αλλά και την επιβεβαίωση της ορθής εκτέλεσής της μέσω ερωτημάτων προς το σύστημα, καταγράφηκαν θετικά αλλά και αρνητικά ποιοτικά στοιχεία προς συζήτηση. Ειδικότερα, παρατηρήθηκε ότι δεν έχουν ενσωματωθεί πλήρως οι δυνατότητες τις οποίες προσφέρουν μεμονωμένα οι μηχανές αναζήτησης που περιλαμβάνονται στο BigDAWG. Για παράδειγμα, για την PostgreSQL ΒΔ, δεν παρέχεται η δυνατότητα εκτέλεσης εντολών union, αλλά και αυτή της μετονομασίας της στήλης ενός πίνακα-αποτελέσματος μέσω ψευδωνύμου *(alias)*,γεγονός το οποίο δεν επιτρέπει την εκτέλεση όλου του φάσματος ερωτημάτων που μπορούν να τεθούν.

Για λόγους αξιολόγησης του συστήματος, η μελέτη περίπτωσης εφαρμόστηκε και σε μία αυτοτελή PostgreSQL ΒΔ, και ως μετρική απόδοσης χρησιμοποιήθηκε ο συνολικός χρόνος εκτέλεσης των

ερωτημάτων. Για το σενάριο αυτό, οι μετρήσεις έδειξαν ότι το BigDAWG εκτελεί τα ίδια ή πανομοιότυπα ερωτήματα, με μία μέση καθυστέρηση της τάξης των 0.405 δευτερολέπτων συγκριτικά με την PostgreSQL μηχανή αποθήκευσης. Εν μέρει, η καθυστέρηση αυτή είναι δικαιολογημένη, και οφείλεται στο γεγονός ότι το ενδιάμεσο λογισμικό του BigDAWG παρεμβάλλεται μεταξύ της εκάστοτε βάσης δεδομένων και του χρήστη, έχοντας το ρόλο του διαμεσολαβητή. Η αξιολόγηση επεκτάθηκε, προκειμένου να προσδιοριστεί η διαφορά εκτέλεσης στο χρόνο των ερωτημάτων μεταξύ και των υπόλοιπων δύο storage engines. Κατά τη νέα αυτή σύγκριση, εντοπίστηκαν μεγάλες διαφορές στην απόδοση του BigDAWG και της SciDB. Αντίθετα, σε σχέση με την Accumulo ΒΔ υπήρξε σύγκλιση του χρόνου εκτέλεσης, ενώ παρατηρήθηκαν και δύο ερωτήματα όπου το polystore σύστημα ήταν ταχύτερο.

Το μεγάλο πλεονέκτημα του BigDAWG έγκειται στη δυνατότητα υλοποίησης migration διαδικασιών, δηλαδή στη μεταφορά των δεδομένων μεταξύ των ετερογενών ΒΔ του συστήματος. Σύμφωνα με τη βιβλιογραφία, κάθε ετερογενής τύπος μηχανής αποθήκευσης μπορεί να επεξεργαστεί τα ίδια δεδομένα, σε διαφορετικό χρόνο ή ακόμα και να υποστηρίξει εντελώς διαφορετικούς τύπους ερωτημάτων.

Συνεπώς, η τεχνολογία των polystore συστημάτων, μπορεί να προσφέρει εναλλακτικές λύσεις σε σχέση με τη χρήση ενός μοντέλου, το οποίο υλοποιείται μέσω μίας και μόνο μηχανής αποθήκευσης. Μία τέτοια περίπτωση όπως η τελευταία, δεν μπορεί να παρέχει την ευελιξία της μεταφοράς δεδομένων σε ετερογενείς ΒΔ, αλλά ούτε και να διαχειριστεί με την ίδια ευελιξία Big Data που προέρχονται από ετερογενείς πηγές, μέσω ενός interface το οποίο επικοινωνεί απευθείας με τις πηγές αυτές στο δικό τους μοντέλο. Όσον αφορά το BigDAWG, αυτό παρουσιάζει μεγάλες προοπτικές, ωστόσο καθώς βρίσκεται σε πειραματικό στάδιο, υπάρχουν ακόμα αρκετά περιθώρια βελτίωσης προκειμένου να θεωρηθεί μία αξιόπιστη λύση στον τομέα του Big Data Integration.

7.2 Μελλοντικές επεκτάσεις

Η συγκεκριμένη εργασία αποτελεί μία προκαταρκτική μελέτη και ανάλυση της τεχνολογίας των polystore συστημάτων και των δυνατοτήτων τους. Συγκεκριμένα, για τον πειραματισμό με την εφαρμογή BigDAWG χρησιμοποιήθηκαν πραγματικά δεδομένα κατανάλωσης ηλεκτρικής ενέργειας από 15 νοικοκυριά, για το διάστημα μεταξύ Νοεμβρίου 2011 και Φεβρουαρίου 2014. Τα δεδομένα αυτά αντιστοιχούν σε 8646 εγγραφές για τον πίνακα “house_consolidated” και 413630 εγγραφές για τον πίνακα “halfhourly”. Επομένως, καθώς η περιοχή της εργασίας άπτεται σε αυτή των Big Data, μία αρχική επέκταση αφορά τον πειραματισμό με ακόμα περισσότερα σπίτια (στο μέγεθος μίας μικρής πόλης), έτσι ώστε να παραχθεί ένας ικανοποιητικός αριθμός δεδομένων εισόδου στο σύστημα, που θα οδηγήσει σε ποιοτικότερα ευρήματα αξιολόγησης της συμπεριφοράς και απόδοσης του polystore.

Επιπροσθέτως, επισημαίνεται ότι για την αξιολόγηση του συστήματος *(query runtime performance)*, χρησιμοποιήθηκαν single storage engine scenarios, δηλαδή περιπτώσεις όπου η εκτέλεση των ερωτημάτων εκτελέστηκε σε μία μόνο μηχανή αποθήκευσης χωρίς την μεταφορά των δεδομένων μεταξύ των

ετερογενών ΒΔ. Η επόμενη μελλοντική επέκταση λοιπόν, αφορά multiple storage engine scenarios, με σκοπό την αξιολόγηση του ισχυρισμού της βιβλιογραφίας, πως ενδεχομένως η απόδοση της ταχύτητας εκτέλεσης ενός ερωτήματος μπορεί να βελτιωθεί μέσα από διαφορετικές μηχανές αποθήκευσης. Τέλος, η SciDB βάση δεδομένων δεν αξιοποιήθηκε σε περιπτώσεις γραμμικής άλγεβρας, δηλαδή για την εκτέλεση μαθηματικών πράξεων που την χαρακτηρίζουν. Συνεπώς, μέσα από τη δημιουργία νέων πινάκων εντός της SciDB μπορούν να προκύψουν νέα ενδιαφέροντα ευρήματα από αυτή.

8

Βιβλιογραφία

[1] Alotaibi, R., Cautis, B., Deutsch, A., Latrache, M., Manolescu, I. and Yang, Y. 2020. ESTOCADA: Towards Scalable Polystore Systems. *Proceedings of the VLDB Endowment* 13(12): 2949-2952. doi: https://doi.org/10.14778/3415478.3415516.

[2] Azevedo, L., Soares, E., Souza, R. and Moreno, M. 2020. Modern Federated Database Systems: An Overview. *Proceedings of the 22nd International Conference on Enterprise Information Systems* 1: 276-283. doi: 10.5220/0009795402760283.

[3] Bondiombouy C., Kolev, B., Levchenko, O. and Valduriez, P. 2016. Multistore Big Data Integration with CloudMdsQL. In Hameurlain, A., Kung, J, Wagner, R. and Chen, Q (eds.). *Transactions on Large-Scale Data- and Knowledge-Centered Systems.* 28. Berlin Heidelberg: Springer-Verlag, 48-74.

[4] Bondiombouy, C. and Valduriez, P. 2016. Query Processing in Multistore Systems: an overview. *International Journal of Cloud Computing (IJCC)* 5(4): 309. doi: 10.1504/IJCC.2016.080903.

[5] Brown, P. 2010. Overview of sciDB: large scale array storage, processing and analysis. *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data:* 963-968. doi: https://doi.org/10.1145/1807167.1807271.

[6] Calvanese, D., Giacomo, G., Lembo, D., Lenzerini, M. and Rosari, R. 2017. Ontology-Based Data Access and Integration. *Encyclopedia of Database Systems.* doi: https://doi.org/10.1007/978-1-4899-7993-3_80667-1.

[7] Cito, J., Schermann, G., Wittern, J., Leitner, P., Zumberi, S. and Gall, H. 2017. An Empirical Analysis of the Docker Container Ecosystem on GitHub. *IEEE/ACM 14th International Conference on Mining Software Repositories:* 323-333. doi: 10.1109/MSR.2017.67.

[8] Clarke, G. The Register. 2010. SciDB: Relational daddy answers Google, Hadoop, NoSQL. https://www.theregister.com/2010/09/13/michael_stonebraker_interview

[9] Dong, X. and Srivastava, D. 2013. Big Data Integration. *Proceedings of the VLDB Endowment* 6(11):1245-1248. doi: 10.1109/ICDE.2013.6544914.

[10] Duggan, J., Elmore, A., Stonebraker, M., Balazinska, M., Howe, B., Kepner, J., Madden, S., Maier, D., Mattson, T. and Zdonik, S. 2015. The BigDAWG Polystore System. *ACM SIGMOD Record* 44(2): 11-16. doi: 10.1145/2814710.2814713.

[11] Euzenat, J. and Shvaiko, P. 2013. *Ontology Matching*. 2nd ed. Berlin Heidelberg: Springer

[12] Fink, J., Gobert, M. and Cleve, A. 2020. Adapting Queries to Database Schema Changes in Hybrid Polystores. IEEE *Proceedings of the 20th International Working Conference on Source Code Analysis and Manipulation (SCAM)* 1:127-131. doi: 10.1109/SCAM51674.2020.00019.

[13] Gadepally, V., Chen, P., Duggan, J., Elmore, A., Haynes, B., Kepner, J., Madden, S., Mattson, T. and Stonebraker, M. 2016. The BigDAWG Polystore System and Architecture. *IEEE High Performance Extreme Computing Conference*: 1-6. doi: 10.1109/HPEC.2016.7761636.

[14] Ghosh, D. 2010. Multiparadigm Data Storage for Enterprise Applications. *IEEE Software* 27(5): 57-60. doi: https://doi.org/10.1109/MS.2010.87.

[15] Hurwitz, J., Nugent, A., Halper, F. and Kaufman, M. 2013. *Big Data For Dummies.* Hoboken, N.J.: Wiley.

[16] Jananthan, H., Zhou, Z., Gadepally, V., Hutchison, D., Kim, S. and Kepner, J. 2017. Polystore Mathematics of Relational Algebra. *IEEE International Conference on Big Data.* doi: 10.1109/BigData.2017.8258298.

[17] Jirkovsky, V. and Obitko, M. 2014. Semantic Heterogeneity Reduction for Big Data in Industrial Automation. *CEUR Workshop Proceedings.* 1214. http://ceur-ws.org/Vol-1214/z1.pdf.

[18] Karande, N. 2018. A Survey Paper on NoSQL Databases: Key-Value Data Stores and Document Stores. *International Journal of Research in Advent Technology* 6(2): 1-5. http://www.ijrat.org/downloads/Vol-6/feb-2018/paper%20ID-62201812.pdf.

[19] Kharlamov, E., Mailis, T., Bereta, K., Bilidas, D., Brandt, S., Ruiz, E., Lamparter, S., Neuenstadt, C., Özcep, Ö., Soylu, A., Svingos, C., Xiao, G., Zheleznyakov, D., Calvanese, D., Horrocks, I., Giese, M., Ioannidis, Y., Kotidis, Y., Moller, R., Waaler, A. 2016. A semantic approach to polystores. *Proceedings of the IEEE International Conference on Big Data.* 2565-2573. doi: 10.1109/BigData.2016.7840898.

[20] Kolovos, D., Medhat, F., Paige, R., Ruscio, D., Storm, T., Scholze, S. and Zolotas, A. 2019. Domain-Specific Languages for the Design, Deployment and Manipulation of Heterogeneous Databases. *2019 IEE/ACM 11TH International Workshop on Modelling in Software Engineering (MISE): 89-92.* doi: https://doi.org/10.1109/MiSE.2019.00021.

[21] Laney, D. 2001. 3D Data Management: Controlling Data Volume, Velocity, and Variety. Application Delivery Strategies by *META Group Inc.* 949. http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf.

[22] Leclerq, E. and Savonnet, M. 2018. TDM: A Tensor Data Model for Logical Data Independence in Polystore Systems. *VLD Workshop on Data Management and Analytics for Medicine and Healthcare.* 39-56. doi: 10.1007/978-3-030-14177-6_4.

[23] Manousis, P., Vassiliadis, P., Zarras, A. and Papastefanatos, G. 2016. Schema Evolution for Databases and Data Warehouses. In Zimanyi, E. and Abello, A. (eds.). *Business Intelligence.*Springer, Cham. 1-31. doi: 10.1007/978-3-319-39243-1_1.

[24] Mattson, T., Gadepally, V., She, Z., Dziedzic A. and Parkhurst. J. 2017. Demonstrating the BigDawg Polystore System for Ocean Metagenomic Analysis. *CIDR.* Santa Cruz, USA, January 8-11, 1-9. http://cidrdb.org/cidr2017/papers/p120-mattson-cidr17.pdf

[25] Mondal, A., Gupta, H., Srivastava, J., Reddy, P. and Somayajulu, D. 2018. *Big Data Analytics*. Springer International Publishing. doi: 10.1007/978-3-030-04780-1.

[26] Moreno, J., Fernadez, E., Medina, E. and Serrano, M. 2018. A Security Pattern for Key-Value NoSQL Database Authorization. *Proceedings of the 23rd European Conference on Pattern Languages of Programs:* 1-4. doi: https://doi.org/10.1145/3282308.3282321.

[27] Patidar, R., Shrestha, S. and Bhalla, S. 2018. Polystore Data Management Systems for Managing Scientific Data-sets in Big Data Archives. *Lecture Notes In Computer Science* 11297: 217-227. doi: 10.1109/ICRITO.2018.8748325.

[28] Sabtu, A., Azmi, N., Sjarif, N., Ismail, S. Yusop, O. Sarkan, H. and Chuprat, S. 2017. The Challenges of Extract, Transform and Loading (ETL) System Implementation For Near Real-Time Environment. 1-5. doi: 10.1109/ICRIIS.2017.8002467.

[29] Sawyer, S., O'Gwynn, D., Tran, A. and Yu, T. 2013. Understanding Query Performance in Accumulo. *2013 IEEE High Performance Extreme Computing Conference (HPEC).* Waltham, MA, USA 10-12 September, 1-6. doi: 10.1109/HPEC.2013.6670330.

[30] Stonebraker, M. 2015. *The Case for Polystores*. https://wp.sigmod.org/?p=1629.

[31] Stonebraker, M. and Cetintemel, U. 2005. One size fits all: an idea whose time has come and gone. *Proceedings of the 21st International Conference on Data Engineering*. 2-11. doi: 10.1109/ICDE.2005.1.

[32] Stonebraker, M., Brown, P., Poliakov A. and Raman S. 2011. The Architecture of SciDB. *Lecture Notes in Computer Science* 6809: 1-16. doi: https://doi.org/10.1007/978-3-642-22351-8_1.

[33] Suwanmanee, S., Benslimane, D., Champin, P. and Thiran, P. 2005. Wrapping And Integrating Heterogeneous Databases With Owl. *Advanced Information Networking and Applications.AINA 2005. 19th International Conference* 1: 145-150. https://liris.cnrs.fr/Documents/Liris-1626.pdf.

[34] Tan, R., Chirkova, R., Gadepally, V. and Mattson, T. 2017. Enabling Query Processing across Heterogeneous Data Models: A Survey. *Proceedings of the IEEE International Conference on Big Data (Big Data)*. 3211-3220. doi: 10.1109/BigData.2017.8258302.

[35] Vogt, M., Stiemer, A. and Schuldt, H. 2018. Polypheny-DB: Towards a Distributed and Self-Adaptive Polystore. *IEEE International Conference on Big Data (Big Data)*: 3364-3373. doi: 10.1109/BigData.2018.8622353.

[36] Wang, J., Baker, T., Balazinska, M., Halperin, D., Haynes, B., Howe, B., Hutchison, D., Shrainik, J., Maas, R., Mehta, P., Moritz, D., Myers, B, Ortiz, J., Suciu, D, Whitaker, A. and Xu, S. 2017. The Myria Big Data Management and Analytics System and Cloud Servises. *CIDR.* Chaminade, California January 8-11. http://cidrdb.org/cidr2017/papers/p37-wang-cidr17.pdf.

[37] Wang, L. 2017. Heterogeneous Data and Big Data Analytics. *Automatic Control and Information Sciences* 3(1): 8-15. doi: 10.12691/acis-3-1.

[38] Widom, J. 1995. Research Problems in Data Warehousing. *CIKM '95: Proceedings of the fourth international conference on Information and knowledge management*: 25-30. doi: https://doi.org/10.1145/221270.221319.

Appendix

Επισημαίνεται ότι για την υλοποίηση της παρούσας εργασίας, χρησιμοποιήθηκε το περιβάλλον Ubuntu Linux 21.10 μέσω της εικονικής μηχανής Oracle VM Virtualbox. Για την αρχική περιγραφή των δεδομένων χρησιμοποιήθηκαν απλές εντολές της Python γλώσσας (διανομή μέσω της σουίτας Anaconda και εκτέλεση των εντολών στο προγραμματιστικό περιβάλλον ανάπτυξης Jupyter Notebook), ενώ η ίδια γλώσσα αξιοποιήθηκε και για την εισαγωγή δεδομένων στην Accumulo μηχανή αποθήκευσης. Επιπλέον, χρησιμοποιήθηκαν εντολές στις γλώσσες PostgreSQL (SQL), SciDB (AFL, AQL) και Accumulo για την επεξεργασία των δεδομένων στο BigDAWG σύστημα. Τέλος, για την αλληλεπίδραση με το polystore σύστημα, χρησιμοποιήθηκαν εντολές Unix, Docker αλλά και BigDAWG στη γραμμή εντολών του Linux συστήματος.

Όσον αφορά τις εικόνες για τις οποίες δεν περιλαμβάνεται κάποια πηγή, αυτές υλοποιήθηκαν μέσω του Microsoft Visio. Τέλος, μέσα από το ίδιο λογισμικό υλοποιήθηκαν τα σχεσιακά μοντέλα, αλλά και οι πίνακες Gantt-Chart.

Κώδικας

Κώδικας1

```
1. import pandas as pd

2.

3. daily = pd.read_csv(r"C:\Users\Παύλος\Desktop\archive\daily_da-
   taset.csv\daily_dataset.csv")

4. halfhourly = pd.read_csv(r"C:\Users\Παύλος\Desktop\archive\half-
   hourly_dataset\halfhourly_dataset\block_111.csv")

5. house_information = pd.read_csv(r"C:\Users\Παύλος\Desktop\archive\in-
   formations_households.csv", usecols = ['LCLid','stdor-
   ToU','Acorn','Acorn_grouped'])

6. acorn =  pd.read_csv(r"C:\Users\Παύλος\Desktop\archive\acorn_de-
   tails.csv", encoding='latin1')

7. weather_daily =  pd.read_csv(r"C:\Users\Παύλος\Desktop\ar-
   chive\weather_daily_darksky.csv", usecols = ['temperatureMax','tem-
   peratureMaxTime','temperatureMin','summary'])
```

```
holidays = pd.read_csv(r"C:\Users\Παύλος\Desktop\archive\uk_bank_holidays.csv")

house_info = pd.read_csv(r"C:\Users\Παύλος\Desktop\Feed\house_info.csv")

daily.head()

halfhourly.tail()

house_information.head()

acorn.head()

acorn['MAIN CATEGORIES'].unique()

weather_daily.head()

holidays.head()

house_info.head()
```

Κώδικας2

```
# postgres-data2

docker cp cluster_setup/postgres-data2/bdsetup bigdawg-postgres-data2:/

docker cp mimic2_flatfiles.tar.gz bigdawg-postgres-data2:/bdsetup/

# Insert csv to container

docker cp ~//Desktop//Feed//house_info.csv bigdawg-postgres-data2:/house_info.csv

docker cp ~//Desktop//Feed//house_consolidated.csv bigdawg-postgres-data2:/house_consolidated.csv

docker cp ~//Desktop//Feed//calendar.csv bigdawg-postgres-data2:/calendar.csv

docker exec --user=root bigdawg-postgres-data2 /bdsetup/setup.sh

# scidb

docker cp cluster_setup/scidb-data/bdsetup bigdawg-scidb-data:/home/scidb/

docker cp a40001_000001.dat bigdawg-scidb-data:/home/scidb/bdsetup
```

```
12. docker cp a40001_000001.hea bigdawg-scidb-data:/home/scidb/bdsetup
13. # Move halfhourly.csv to SciDB container
14. docker cp ~//Desktop//Feed/halfhourly.csv bigdawg-scidb-data:/home/scidb/scidb_data/000/0/halfhourly.csv
15. docker exec bigdawg-scidb-data /home/scidb/bdsetup/setup.sh
16. # accumulo
17. docker cp cluster_setup/accumulo-data/bdsetup bigdawg-accumulo-zookeeper:/
18. docker cp s00318.txt bigdawg-accumulo-zookeeper:/bdsetup/
19. # Move acc.txt to accumulo-container
20. docker cp acc.txt bigdawg-accumulo-zookeeper:/bdsetup/
21. docker exec bigdawg-accumulo-zookeeper python /bdsetup/setup.py
```

Κώδικας3

```
1. docker exec -it bigdawg-postgres-data2 bash
2. ls
```

Κώδικας4

```
1. docker exec -it bigdawg-scidb-data bash
2. cd /home/scidb/scidb_data/000/0
3. ls
```

Κώδικας5

```
1. docker exec -it bigdawg-accumulo-zookeeper bash
2. cd bdsetup
3. ls
```

Κώδικας6

```
1. echo "Creating database households"
2. psql -c 'create database households;'
```

```
3. echo "create schema households"
4. #psql inventory -c 'create schema inventory;'
5. psql households -c 'create schema households;'
6. echo "create table house_info, calendar and house_consolidated"
7. psql households -c 'create table households.house_info(houseId inte-
   ger PRIMARY KEY, tariff varchar(40), acorn varchar(40), type var-
   char(40), tenure varchar(40), value real, members integer, annual-
   Income real);'
8. psql households -c 'create table households.calendar(day date PRIMARY
   KEY, type varchar(80));'
9. psql households -c 'create table households.house_consoli-
   dated(tableId integer PRIMARY KEY, houseId integer, day date
   REFERENCES households.calendar(day), energyMedian real, energyMean
   real, energyMax real, energyCount integer, energyStd real, energySum
   real, energyMin real, FOREIGN KEY(houseId) REFERENCES house-
   holds.house_info(houseId));'
10.  echo "insert data into table house_info, calendar and house_consol-
   idated"
11.  psql households -c "\copy households.house_info FROM
   '//house_info.csv' delimiter ';' csv header;"
12.  psql households -c "\copy households.calendar FROM '//calendar.csv'
   delimiter ';' csv header;"
13.  psql households -c "\copy households.house_consolidated FROM
   '//house_consolidated.csv' delimiter ';' csv header;"
```

Κώδικας7

```
1. echo "Creating array 'halfhourly'"
2. /opt/scidb/14.12/bin/iquery -aq "CREATE ARRAY halfhourly <
   houseId:string, day:string, time:string, energy:double >
   [i=0:*,1000000,0];"
3.
4. echo "Inserting values to halfhourly array..."
5. /opt/scidb/14.12/bin/iquery -aq "load(halfhourly, 'half-
   hourly.csv');"
6. echo "Done... Successful insertion of halfhourly data into SciDB"
```

Κώδικας8

```
1. from pyaccumulo import Accumulo, Mutation, Range
2. import settings
3. import sys
4. sys.path
5. sys.path.append('/bdsetup')
6.
7. table = "weather"
8.
9. conn = Accumulo(host=settings.HOST, port=settings.PORT, user=set-
   tings.USER, password=settings.PASSWORD)
10.
11.  if conn.table_exists(table):
12.      conn.delete_table(table)
13.
14.  conn.create_table(table)
15.  wr = conn.create_batch_writer(table)
16.
17.  print "Ingesting some weather data ..."
18.  f = open("/bdsetup/acc.txt", "r")
19.  for i in range(882):
20.      line = f.readline()
21.      mylist = line.split(',')
22.      label = line.split(',')[0]
23.      cq1 = line.split(',')[1]
24.      val1 = line.split(',')[2]
25.      mut = Mutation(label)
26.      mut.put(cf='temperature range', cq =cq1,val=val1)
```

```
27.      #mut.put(cf='cf_%s'%label, cq='cq1', val=line)
28.      #mut.put(cf='cf_%s'%label, cq='cq2', val=line)
29.      wr.add_mutation(mut)
30.      i += 1
31.  wr.close()
32.
33.  print "Done! Successfull insertion of weather data into Accumulo
     DB"
34.
35.  conn.close()
```

Κώδικας9

```
1. -- adding own data - scidb
2. insert into catalog.objects values(50, 'halfhourly', 'houseId, day,
   time, energy',2,6);
3. -- adding own data accumulo
4. insert into catalog.objects values(51, 'weather','', 7,7);
5. -- adding households objects
6. insert into catalog.objects values(52, 'households.house_info',
   'houseId, tariff, acorn, type, tenure, value, members, annualIncome',
   9,9);
7. insert into catalog.objects values(53, 'households.house_consolidat-
   ed', 'tableId, houseId, day, energyMedian, energyMean, energyMax, en-
   ergyCount, energyStd, energySum, energyMin', 9,9);
8. insert into catalog.objects values(54, 'households.calendar', 'day,
   type', 9,9);
9. select setval('catalog.objects_oid_seq'::regclass, 54);
```

Κώδικας10

```
1. CREATE SCHEMA IF NOT EXISTS households;
2. -- see plain.sql
```

```
3. CREATE TABLE IF NOT EXISTS households.house_info (houseId integer
   PRIMARY KEY, tariff varchar(40), acorn varchar(40), type varchar(40),
   tenure varchar(40), value real, members integer, annualIncome real);

4. CREATE TABLE IF NOT EXISTS households.calendar (day date PRIMARY KEY,
   type varchar(80));

5. CREATE TABLE IF NOT EXISTS households.house_consolidated (tableId in-
   teger PRIMARY KEY, houseId integer, day date REFERENCES house-
   holds.calendar(day), energyMedian real, energyMean real, energyMax
   real, energyCount integer, energyStd real, energySum real, energyMin
   real, FOREIGN KEY(houseId) REFERENCES house-
   holds.house_info(houseId));
```

Κώδικας11

```
1. psql -f households_schemas_ddl.sql -d bigdawg_schemas
```

Κώδικας12

```
1. docker exec -it bigdawg-postgres-catalog bash

2. psql

3. \l

4. \c bigdawg_catalog
```

Query

Query0

```
1. SELECT * FROM CATALOG.objects WHERE oid >= 50
```

Query1

```
1. curl -X POST -d "bdrel(select * from households.house_info limit 5;)"
   localhost:8080/bigdawg/query/
```

Query2

```
1. curl -X POST -d "bdrel(select sum(energySum) from house-
   holds.house_consolidated group by day order by day)" lo-
   calhost:8080/bigdawg/query/
```

Query3

```
1. curl -X POST -d "bdrel(select hi.tenure, avg(hc.energySum) from
   households.house_info as hi inner join households.house_consolidated
   as hc on hi.houseId = hc.houseId  group by hi.tenure)" lo-
   calhost:8080/bigdawg/query/
```

Query4

```
1. curl -X POST -d "bdrel(select distinct hi.houseId, hi.acorn, hi.an-
   nualIncome, hc.energyMax, hc.energyMin, c.type from house-
   holds.house_info as hi inner join households.house_consolidated as hc
   on hi.houseId = hc.houseId inner join households.calendar as c on
   c.day = hc.day where hi.annualIncome > 40000 and c.day = '2013-12-
   25')" localhost:8080/bigdawg/query/
```

Query5

```
1. curl -X POST -d "bdrel(select hi.acorn, SUM(hc.energySum) from house-
   holds.house_info as hi inner join households.house_consolidated as hc
   on hi.houseId = hc.houseId inner join households.calendar as c on
   c.day = hc.day where c.day >='2013-01-01' group by hi.acorn)" lo-
   calhost:8080/bigdawg/query/
```

Query6

```
1. curl -X POST -d "bdrel(select hi.houseId,hi.acorn, hi.members,
   MAX(hc.energySum) from households.house_info as hi inner join house-
   holds.house_consolidated as hc on hi.houseId = hc.houseId inner join
   households.calendar as c on c.day = hc.day where hi.value <=100000
   group by hi.houseId order by houseId DESC)" localhost:8080/big-
   dawg/query/
```

Query7

```
1. curl -X POST -d "bdarray(aggregate(filter(halfhourly, houseId =
   '3613'), sum(energy)));" localhost:8080/bigdawg/query/
```

Query8

```
1. curl -X POST -d "bdarray(filter(halfhourly, houseId = '3505' AND time
   = '18:00:00'));" localhost:8080/bigdawg/query/
```

Query9

```
1. curl -X POST -d "bdarray(aggregate(halfhourly, avg(energy), max(en-
   ergy), min(energy)));" localhost:8080/bigdawg/query/
```

Query10

```
1. curl -X POST -d "bdtext({ 'op' : 'scan', 'table' : 'weather' });" lo-
   calhost:8080/bigdawg/query/
```

Query11

```
1. curl -X POST -d "bdtext({ 'op' : 'scan', 'table' : 'weather', 'range'
   : { 'start': ['2014-01-01','',''], 'end' : ['2014-01-16','','']} });"
   localhost:8080/bigdawg/query/
```

Query12

```
1. curl -X POST -d "bdrel(SELECT AVG(energy) FROM bdcast( bdarray(fil-
   ter(halfhourly,houseId ='3358')), table12,'(i bigint, houseId var-
   char(10),  day varchar(20), energy real)', relational) WHERE day >=
   '2013-01-01' AND day <='2013-12-31' GROUP BY day ORDER BY day)" lo-
   calhost:8080/bigdawg/query/
```

Query13

```
1. curl -X POST -d "bdtext({ 'op' : 'scan', 'table' : 'bdcast(bdrel(se-
   lect * from households.house_info where houseId = 2305), res, '',
   text)'})" localhost:8080/bigdawg/query/
```

Query14

```
1. curl -X POST -d "bdrel(select c.day, SUM(hc.energySum) as totalCon-
   sumption from households.house_info as hi inner join house-
   holds.house_consolidated as hc on hi.houseId = hc.houseId inner join
   households.calendar as c on c.day = hc.day group by c.day order by
   totalConsumption desc limit 5;)" localhost:8080/bigdawg/query/
```

Query15

```
1. curl -X POST -d "bdrel(select houseId from households.house_consoli-
   dated group by houseId having sum(energySum) < 4200 UNION select
   houseId from households.house_info where acorn = 'Adversity')" lo-
   calhost:8080/bigdawg/query/
```

Query16

```
1. curl -X POST -d "bdarray(filter(halfhourly, houseId = '2305'));" lo-
   calhost:8080/bigdawg/query/
```

Query17

```
1. curl -X POST -d "bdarray(sort(halfhourly, houseId));" lo-
   calhost:8080/bigdawg/query/
```

Query18

```
1. curl -X POST -d "bdtext({ 'op' : 'scan', 'table' : 'weather', 'range'
   : { 'start': ['2013-06-01','',''] } });" localhost:8080/bigdawg/query/
```

Query19

```
1. curl -X POST -d "bdtext({ 'op' : 'scan', 'table' : 'weather', 'range'
   : { 'end': ['2012-12-31','',''] } });" localhost:8080/bigdawg/query/
```

Query20

```
1. curl -X POST -d "bdtext({ 'op' : 'scan', 'table' : 'weather', 'range'
   : { 'start': ['2011-11-11','',''], 'end' : ['2014-02-16','',''] } });"
   localhost:8080/bigdawg/query/
```