A META DATA VAULT APPROACH FOR EVOLUTIONARY INTEGRATION OF BIG DATA SETS: CASE STUDY USING THE NCBI DATABASE FOR GENETIC VARIATION

Zaineb Naamane and Vladan Jovanovic

Department of Computer Science, Georgia Southern University, Statesboro, GA

ABSTRACT

A data warehouse integrates data from various and heterogeneous data sources and creates a consolidated view of the data that is optimized for reporting and analysis. Today, business and technology are constantly evolving, which directly affects the data sources. New data sources can emerge while some can become unavailable. The DW or the data mart that is based on these data sources needs to reflect these changes. Various solutions to adapt a data warehouse after the changes in the data sources and the business requirements have been proposed in the literature [1]. However, research in the problem of DW evolution has focused mainly on managing changes in the dimensional model while other aspects related to the ETL, and maintaining the history of changes has not been addressed. The paper presents a Meta Data vault model that includes a data vault based data warehouse and a master data management. A major area of focus in this research is to keep both history of changes and a "single version of the truth," through an MDM, integrated with the DW. The paper also outlines the load patterns used to load data into the data warehouse and materialized views to deliver data to end-users. To test the proposed model, we have used big data sets from the biomedical field and for each modification of the data source schema, we outline the changes that need to be made to the EDW, the data marts and the ETL.

KEYWORDS

Data Warehouse (DW), Enterprise Data Warehouse (EDW), Business Intelligence, Data Vault (DV), Business Data Vault, Master Data Vault, Master Data Management (MDM), Data Mart, Materialized View, Schema Evolution, Data Warehouse Evolution, ETL, Metadata Repository, Relational Database Management System (RDMS), NoSQL.

1. INTRODUCTION

The common assumption about a data warehouse is that once it is created, it remains static. This assumption is incorrect because the business environment needs to change and increase with time, and business processes are subject to frequent change. This change means that a new type of information requirement becomes necessary. Research in the data warehouse environment has not adequately addressed the management of schema changes to source databases. Such changes have significant implications on the metadata used for managing the warehouse and the applications that interact with source databases. In short, the data warehouse may not be consistent with the data and the structure of the source databases. For this reason, the data warehouse should be designed in away that allows flexibility, evolution, and scalability. In this thesis, we examine the implications for managing changes to the schema of the source databases and propose a prototype that will be able to perform the following: 1)- adapt to a changing business environment; 2)- support very large data; 3)- simplify design complexities; 4) - allow addition and removal of data sources with no impact on the existing design; 5)- keep all past data easily integrated with any future data using only additions; and 6)- allow tracking of history, evolution, and usage of data.

Designing such a flexible system motivated the focus of this thesis. The Data Vault methodology (DV 2.0), along with big data technologies, will be used to develop a Meta Data Vault approach for evolutionary integration of big data sets. The prototype developed will solve primary business requirements and address the evolving nature of the data model as well as issues related to the history and the traceability of the data.

2. Research Methodology

2.1. Problem

Building a data warehouse management system for scientific applications poses challenges. Complex relationships and data types, evolving schema, and large data volumes (in terabyte) are some commonly cited challenges with scientific data. Additionally, since a data model is an abstraction of a real world concept, the model has to change with changes in domain knowledge. For example, in biomedical long term research, as knowledge is gained from behavioral experiments, new tasks or behaviors might be defined or new signals might be introduced. This means that the data model and mappings must be redone again and again as the data models shift, therefore keeping all past data easily integrated with any future data using only additions is very desirable.

2.2. Research Requirements

A critical factor in the advancement of biomedical research is the ease in which data can be integrated, redistributed, tracked, and analyzed both within and across functional domains. My research methodology will be driven by meta-data requirements from this domain such as:

- the need to incorporate big data and unstructured data (research data includes free-text notes, images, and other complex data),
- keeping all past data easily integrated with any future data, using only additions,
- accessibility to and tracking of data or process elements in large volumes of scientific data,
- tracking where and when the data originated about a medical analysis (history, evolution, and usage of data is often as valuable as the results of the analysis itself), and
- allowing seamless integration of data when the model changes.

2.3. Validation Approach

The research of this paper centers upon the following question: Can the proposed Meta Data Vault prototype serve as a standard solution able to effectively track and manage changes for large-scale data sets while ensuring performance and scalability?

The solution developed will benefit from both relational and non-relational data design (RDBMS and NoSQL) by including meta-data approach (combining MDM concepts with DV concepts) for tracking and managing schema evolution, combined with NoSQL technology for managing big data sets.

To validate the proposed approach, a system prototype will be developed as a full proof of concept implementation / architecture with a performance base experiment on a real example from the biomedical field, which is the Single Nucleotide Polymorphism Database of Nucleotide Sequence Variation.

3. THE PROPOSED SOLUTION

3.1. ENVIRONMENT

Our solution is designed to handle the exponentially growing volume of data, the variety of semi-structured and unstructured data types, and the velocity of real-time data processing. To reach this goal, we will use SQL server 2016 because it represents the perfect choice for our database environment, especially with the new NoSQL feature introduced. This feature allows us to query across relational and non-relational sources like Hadoop. SQL Server Integration services (SSIS) will be used for data integration.

3.2. ARCHITECTURE SOLUTION

To reach our research goal and develop a flexible system which will be able to track and manage changes in both data and metadata, as well as their schemas, we will use a data vault based architecture (Figure 1).

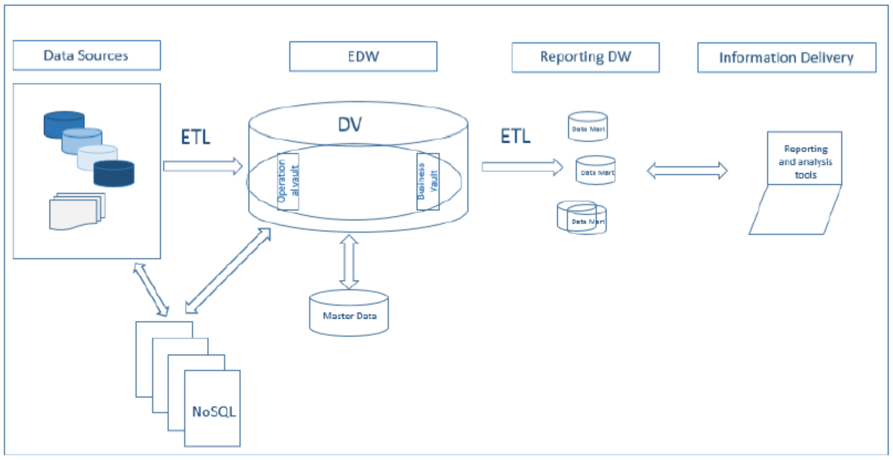


Figure 1. Solution Architecture

The proposed architecture consists of four parts:

- Data sources: A data warehouse is designed to periodically get all sorts of data from various data sources. One of the main objectives of the data warehouse is to bring together data from many different data sources and databases in order to support reporting needs and management decisions. Data sources include operational data stores and external sources.
- EDW: The enterprise data warehouse layer is modeled after the Data Vault methodology. It consists of a DV model [2] and a master data. The DV model is composed of two data models: one oriented towards the data source side (Operational Vault) and represents the system of record, and the other oriented towards the reporting side (Business Vault). The

master data is the metadata repository, which serves to integrate the operational vault and the business vault in the EDW.

- Reporting DW: Contains one or more information mart that depends on the enterprise data warehouse layer.
- Information Delivery: Allows users to perform their own data analysis tasks without involvement of IT.

3.2.1. DATA SOURCE

3.2.1.1. RELATIONAL MODEL

To test our prototype, we will use a business case from the genetic field which deals with a DW for the Nucleotide Sequence Variation.

Sequence variations exist at defined positions within genomes and are responsible for individual phenotypic characteristics, including a person's propensity toward complex disorders such as heart disease and cancer. As tools for understanding human variation and molecular genetics, sequence variations can be used for gene mapping, definition of population structure, and performance of functional studies [3]. For our data source, we will use the Single Nucleotide Polymorphism database (dbSNP), which is a public-domain archive for a broad collection of simple genetic polymorphisms. It has been designed to support submissions and research into a broad range of biological problems [3].The dbSNP Schema is very complex with well over 100 tables and many relationships among tables. One single ER diagram with all dbSNP tables will be too large to test our solution. The dbSNP schema has been modified and restricted to tables related to the particular implementation of the dbSNP SNP variation data. The build used for this study is the combination of different builds.

All data model examples are shown as diagrams using appropriate style-related variations based on the standard IDEF1X data -modeling notation. The reason for selecting the Idef1X is its very convenient closeness to the relational model, i.e. a direct visual representation of tables and foreign keys and minimal set of visual data modeling elements.

The dbSNP data source model in the figure below deals with three subject areas:

- Submitted Population: Contains information on submitted variants and information on the biological samples used in the experiments, such as the number of samples and the ethnicity of the populations used.
- SNP Variation: Contains information on the mapping of variants to different reference genomes.

Allele Frequency for Submitted SNP: Stores information on alleles and frequencies observed in various populations.

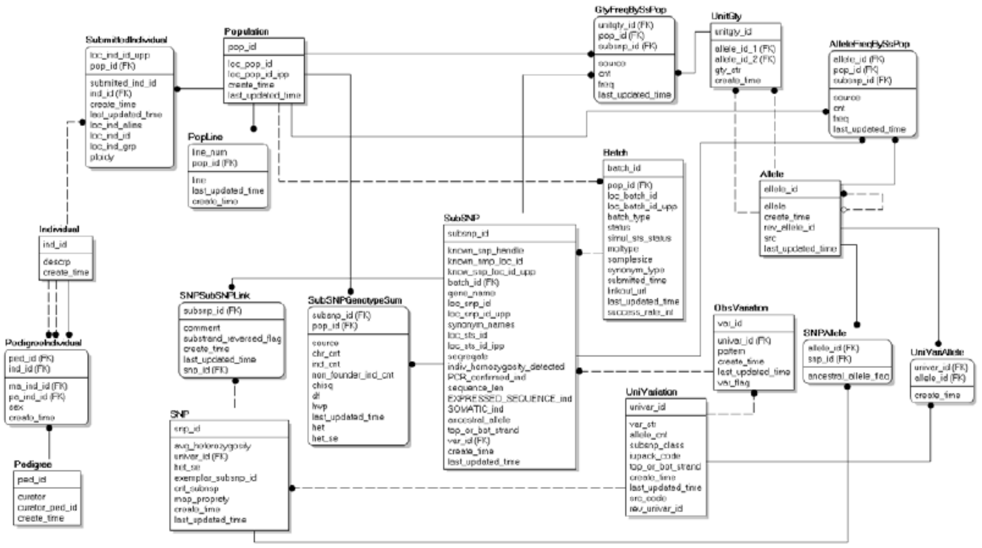


Figure 2. Data models for source databases

The model in Figure 2 has two major classes of data. The first class is submitted data, namely original observations of sequence variation, accessioned using a submitted SNP identifier (ss). The second class is generated during the dbSNP build cycle by aggregating data from multiple submissions, as well as data from other sources, and is identified with a "reference SNP" (rs) number. The rs identifier represents aggregation by type of sequence change and location on the genome if an assembled genome is available, or aggregation by common sequence if a genome is not available (Kitts et al, 2013).

Each row in the SNP table corresponds to a unique reference SNP cluster identified by snp_id (rs), as opposed to the table SubSNP that contains a separate row for each submission identified by subsnp_id. SNPsubSNPLNK is the central table that keeps the relationship between submitted SNP (ss) and reference SNP (rs).

Allele stores all unique alleles in the dbSNP. A submitted SNP variation could have two or more alleles. For example, a variation of A/G has two alleles: A and G. The alleles A and G are stored in the Allele table.

Submitted variation patterns are stored in ObsVariation table.

Alleles (Allele) typically exist at different frequencies (AlleleFreqBySsPop) in different populations (Population). GtyFreqBySsPop contains genotype frequency per submitted SNP and population. SubSNPGenotypeSum contains summary data for each SubSNP within a specific population.

3.2.1.2. NOSQL

Genomics produces huge volumes of data [4]; each human genome has about 25,000 genes comprised of 3 million base pairs. This amounts to 100 gigabytes of data. Consequently, sequencing multiple human genomes would quickly add up to hundreds of petabytes of data, and the data created by analysis of gene interactions multiplies those further [5].

NoSQL databases can store and process genomics data in real time, something that SQL is not capable of doing it when the volume of data is huge and the number of joins required to perform a query is significant.

Our NoSQL solution is not to replace the Relational Model as a whole, but only in some cases where there is a need for scalability and big data.

NoSQL databases allow greater scalability, cost-efficiency, flexibility, and performance when dealing with large data. The two tables that store information about a submitted SNP are subject to scale out with each submission and build, and they are the highest in terms of number of rows and size.

Using a document-based database (JSON) will increase performance as all the related data are in a single document which eliminates the join operation and speeds up the querying process. Moreover, the scalability cost is reduced because a document-based database is horizontally scalable, i.e. we can reduce the workload by increasing the number of servers instead of adding more resources when scaling in a relational database environment. The tables affected by denormalization are SubSNP, SNPSubLink.

3.2.2. ENTERPRISE DATA WAREHOUSE

Our enterprise data warehouse includes the raw data vault (RDV), which is data source oriented and contains unchanged copies of the originals from the data sources, and the business data vault (BDV), which is report oriented and aims to support business integration. In this project, we will implement the RDV and BDV as one single database, where the two models are connected using links. For visibility of the model, we have decoupled the entire model to RDV and BDV models, where hubs are the major redundant entities.

3.2.2.1. RAW DATA VAULT (RDV)

Data Vault design is focused on the functional areas of a business with the hubs representing business keys, links representing transactions between business keys, and the Satellites providing the context of the business keys. In addition to that, reference tables have been included in the data vault to prevent redundant storage of simple reference data that is referenced a lot. The data vault entities are designed to allow maximum flexibility and scalability while preserving most of the traditional skill sets of data modeling expertise [6].

Figure 3 below shows the logical DV model for our solution. Every object (hub, link, satellite, and reference tables) in the model stores two auditing attributes: The first load date (loadDTS), which is the first date when the specific line appeared, and the first load source (recSource), which is the first source where the specific line appeared. These attributes provide audit information at a very detailed level. For simplicity of the models, business keys are used instead of surrogate keys, as recommended by the DV methodology [7]. Hubs are represented in blue, links in red, satellites in yellow, and reference tables in gray. The reference tables are managed in the BDV model but are represented in the RDV model as redundant entities after separating the BDV from the RDV. Every hub has at least one satellite that contains descriptive attributes of the business key. Links represent the relationship between entities. Satellites that hang off the Link tables represent the changing information within that context.

One of the biggest advantages of the RDV model is maintaining history. For example, data related to the reference genome sequence, upon which SNPs are mapped, can change during time. We need to store valid values when a record was created. RefSNPs are operationally defined as a variation at a location on an ideal reference chromosome. These reference chromosomes are the goal of genome assemblies. However, since many of the genome projects are still in work, and since even the ‘finished’ genomes are updated to correct past annotation errors, a refSNP is defined as a variation in the interim reference sequence. Every time there is a genomic assembly update, the interim reference sequence may change, so the refSNPs must be updated [3]. Using a data vault based data warehouse, we will be able to recreate past data correctly and maintain the history of these changes at any point of the time.

The Raw Data Vault (RDV) model is an important piece in our prototype because it will keep the source system context intact. Data coming from different sources are integrated in the DV model below without undergoing transformations. Data are rapidly loaded in their raw format, including the date and the modification source. It is, therefore, possible to rebuild a source’s image at any point of time.

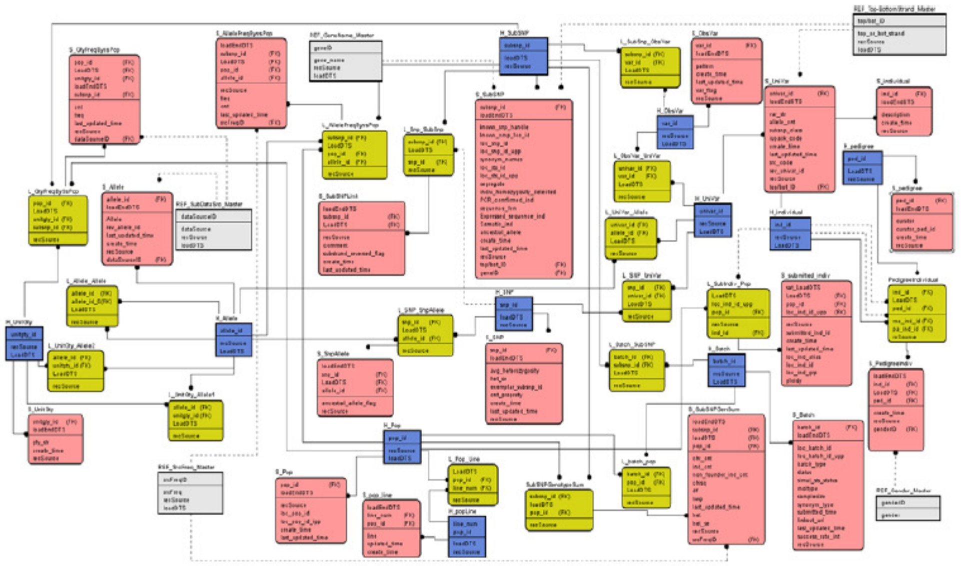


Figure 3. The Raw Data Vault (RDV) Model

3.2.2.3. BUSINESS DATA VAULT (BDV)

The business master data vault (BDV) is a component of our EDW that is oriented towards the reporting side. It is designed from the RDV by adding standardized master data and business rules in order to ensure business integration. Hubs in the BDV (Figure 4) are shown in light blue while hubs from the RDV are shown in dark blue. Every master hub in the BDV model is connected through links to the appropriate Hub from the RDV model. Every master Hub has at least one satellite and one same-as link. Same-as link SAL_SNP_MDM maps a record to other records within a H_SNP_Master hub.

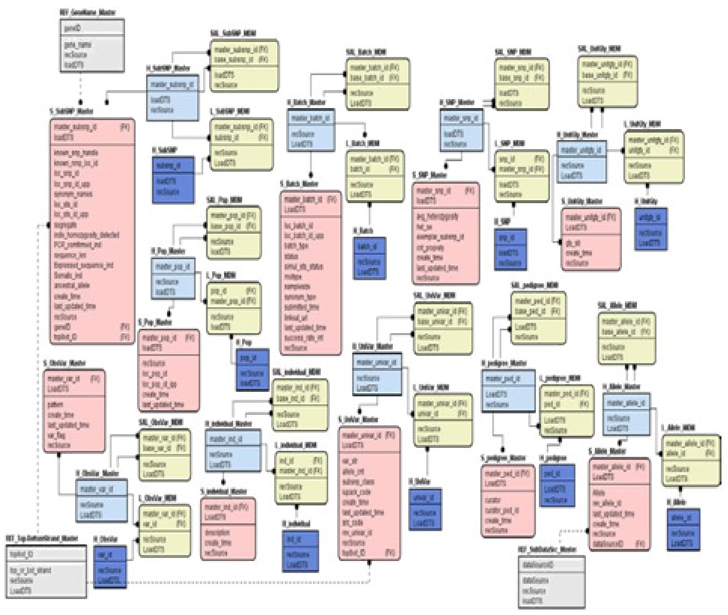


Figure 4.The Business Vault(BDV)Mode

The Business Data Vault (BDV) is also a mandatory piece in the EDW. It is the part where the business rules are executed, where integration is achieved, and where the result is stored. The data transformation in the BDV is typically a value-transformation and not so much a structure transformation used when creating Data Marts.

3.2.2.3. META DATA VAULT (MDV)

The Meta Data Vault (MDV) represents the metadata repository of our solution; the model is designed to preserve metadata, changes in metadata, and the history of metadata. The MDV is a data vault based model, which is used to integrate the Data warehouse metadata with the MDM metadata in one single model.

Figure 5 shows the MDV model developed for our EDW solution. For readability of the model, the DV meta-level attributes (recSource, LoadDTS) are not shown in the figure but are included in all the entities of the model. Hubs, links, satellites, attributes, domains, roles, rules, sources, resources, materialized views, and reference tables are represented as Hubs in the model (blue color). All hubs have at least two satellites (pink color). One contains descriptive attributes of the business key (name, description, type ...) and the other stores the business key. Every hub has a same -as link. This type of link is used to connect source and target business keys. The model also resolves the transformation rules from the data source to the RDV via ETL rules, transformation from the RDV into the BDV via the business, and master rules, as well as the transformation rules from the BDV to the materialized views. These rules are managed in the H_Rule, H_MV, H_Attribute, H_Source entities and their links. The security aspect was also treated in this model. H_Security is a hub linked to others hubs in the model to manage security and user access rights to entities, attributes, sources, resources, etc. This proposed model permanently stores and histories all the application metadata, because it is based on the Data Vault model. Hence, it will not only solve the problem of schema evolution but also the MDM evolution

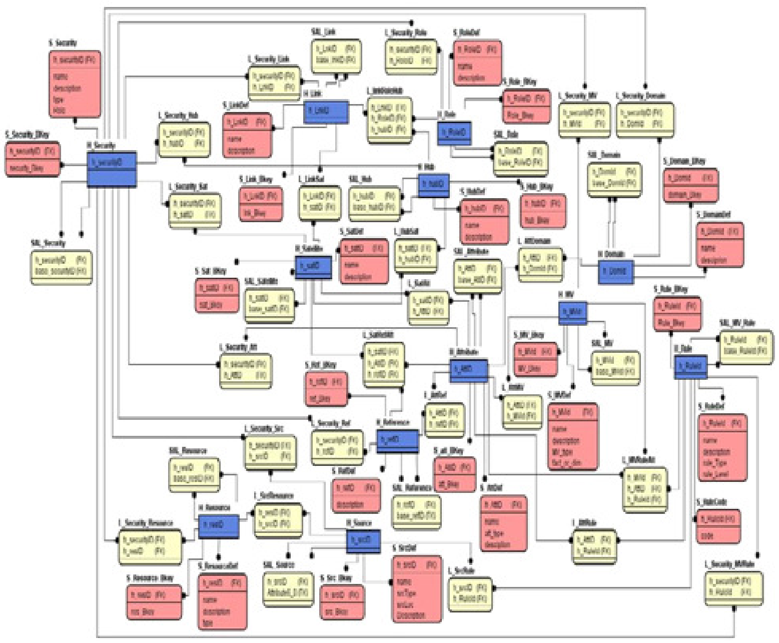


Figure 5. The Master Data Vault (MDV) Model

3.2.3. Reporting DW

3.2.3.1. DATA MART

MART MODEL

Dimensional modeling is still the best practice for analysis and reporting. However, it does not meet the performance expectation for an EDW system [8], [9]. Data Vault, on the other hand, is more suitable for the large Enterprise Data Warehouse, but not suitable for direct analysis and reporting. For this reason, we still need dimensional modeling to create data marts [10] in order to respond to business users' requirements. In order to test our solution, we considered the following requirements:

- Allele frequency and count by submitted SNP, allele and population.
- Genotype frequency and count by submitted SNP, genotype and population.

The Work Information Package (WIP) is constructed for two business metrics within the SNP Data Mart. The two fact tables represent each of the two measures above. Table 1 represents the user requirements glossary.

Fact	Dimensions	Measures	History
Allele frequency and chromosome count in the allele	Allele; Population; Popline; SubSnp	Frequency of the allele in the population Number of chromosomes observed for the allele	10 years
Genotype frequency and individual count for the genotype	Genotype; SubSnp; Population; Popline	Frequency of the allele in the population by genotype Number of individuals having the genotype identified	10 years

Table 1. User requirements glossary

Based on the requirement analysis, we derived the following data mart model.

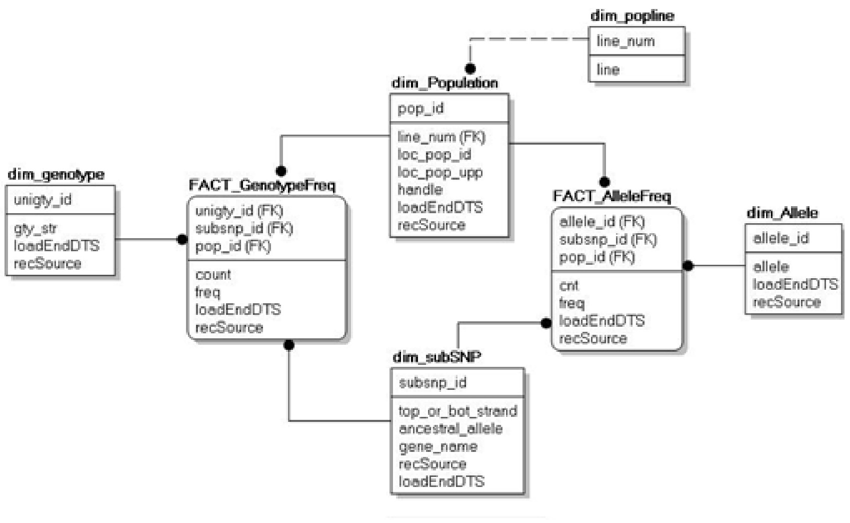


Figure 6. Data Mart Model

3.2.3.2. MATERIALIZED VIEWS

In data warehouses, materialized views can be used to pre-compute and store aggregated data such as Avg. sum etc. Materialized views in a data warehouse system are typically referred to as summaries since they store summarized data. They can also be used to pre-compute joins with or without aggregations. Thus, a materialized view is used to eliminate overhead associated with expensive joins or aggregations for a large or important class of queries [11].

For an EDW running analytical queries directly against the huge raw data volume of a data warehouse, results in unacceptable query performance. The solution to this problem is storing materialized views in the warehouse, which pre-aggregate the data and thus avoid raw data access and speed up queries [12].
In our solution, materialized views are derived from the BDV model and used to perform analysis and reporting on data.

All dimensions are built by joining of a Hub and a Satellite (figure 7)

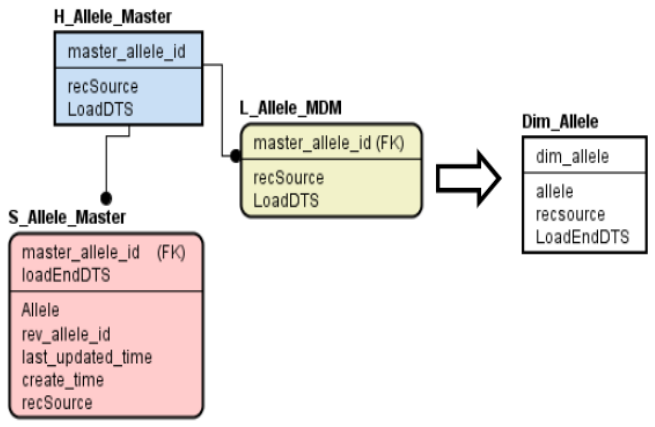


Figure 7. Derived Allele Dimension Table

Hubs, links and satellites are used to derive fact tables (figure 8):

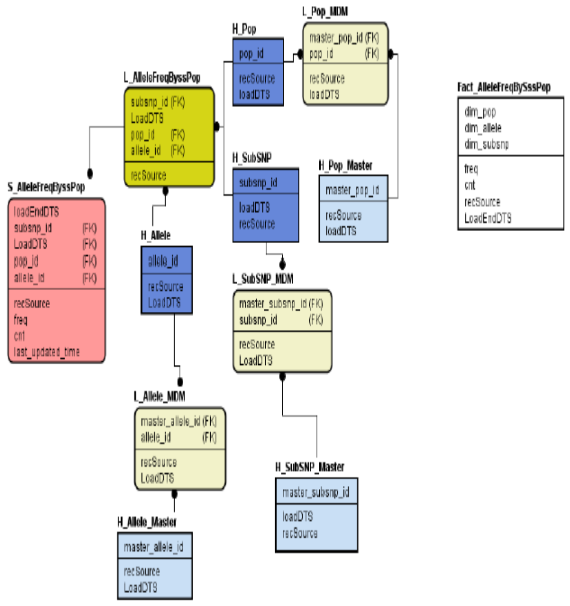


Figure 8. Derived Allele Frequency Fact Table

3.2.3.3. ETL

In this section, we will present the load patterns used to load the data from the data source to the EDW. The Raw Data Vault (RDV) will be built from the data sources, the Business Data Vault (BDV) will be built from the RDV, and finally the DM will be built from the BDV.

The ETL jobs to load a data vault based data warehouse (The RDV and BDV both based on the data vault methodology) typically run in two steps: In the first step, all hubs are loaded in parallel. In the second step, all links and satellites are loaded in parallel.

3.2.3.3.1. ETL For The Raw Data Vault (RDV)

The individual ETL operations for each type of the RDV objects are:

- Hubs: The business key must appear only once in the hub; insert with a lookup on the business key
- Links: The association must appear only one time in the link; insert with a lookup on the association of surrogate key
- Satellites: The loading of the object is the most complex; it depends on the historization mode. Either the changed attributes are updated, or a new version must be inserted.

3.2.3.3.2. ETL For The Business Data Vault (BDV)

Data captured in the RDV remains in its raw form (unchanged). Once the data enters the RDV, it is no longer deleted or changed. The BDV, on the other hand, is report oriented. It is where the business rules are implemented and only the "golden copy" of the records is stored [1]. The issue with the DBSNP database is that data can be replicated between submitters, which make the notion of "golden key" often lost. The BDV is designed to consolidate all the duplicated records coming from the RDV into one master business key. The RDV only contains the highest quality information on which the reporting applications can depend.

The individual ETL operations for each type of the BDV objects are:

- Master Hubs: Hubs in the business Data Vault will receive new surrogate keys. This is better than reusing the keys that were distributed in the BDV because these keys now have different meanings. The surrogate key must only appear once in the Master Hub; insert with a lookup on the surrogate key.
- MDM Links: The relationship between the original records in the RDV and the cleansed records in BDV has to be maintained for audit purposes. The MDM Link shows the relationship between the original copy and the golden copy of data. The association must appear one time only in the link; insert with a lookup on the association of surrogate key.
- Master Satellites: Master satellites in the BDV hold the descriptive attributes of the master business key. The loading of the object is the most complex; it depends on the historization mode. Either the changed attributes are just updated, or a new version must be inserted.

4. Experiments

4.1. Motivation And Overall Perspective

In this experiment, we will evaluate the effects of schema changes to the data source on the modeled data. The motivation here is to experiment how our solution is affected when there is a need of:

4.1.1. Data Sources Schema Changes

Business needs to change and increase with time, and business processes are subject to frequent change. Due to this change, a new type of information requirement that requires different data becomes necessary. These changes have important implications; first on the metadata (MDM) used for managing the warehouse, and second, on the applications that interact with the source databases. In short, the data warehouse may not be consistent with the data and the structure of the source databases. We expect our solution to allow seamless integration, flexibility, evolution, and scalability.

4.1.2. Traceability

The data warehouse team faces difficulties when an attribute change. They need to decide whether the data warehouse keeps track of both the old and new value of the attribute or not, what to use for the key, and where to put the two values of the changed attribute?

We expect our solution to ensure the preservation of history and provenance, which enables tracking the lineage of data back to the source. Data coming from different sources is not transformed before it is inserted into the MDV, thus enabling a permanent system of records.

4.1.3. Auditability

One other typical requirement for a data warehouse is the ability to provide information about source and extraction time of data stored in the system. One reason for that is to track down potential errors and try to understand the flow of the data into the system. Since Data Vault keeps a comprehensive history, including the ability to record where the data came from, we expect that our solution of data vault to be the perfect choice for any system where keeping an audit trail is important.

4.2. Expectations

In this thesis, we examine the implications of changes to the schema of the source databases on the Enterprise data warehouse, as well as the system ability in terms of traceability and auditability. We expect that proposed prototype will be able to:

1. Adapt to a changing business environment;
2. Support very large data, simplify design complexities;
3. Allow addition and removal of data sources without impacting the existing design;
4. Keep all past data easily integrated with any future data using additions only; and
5. Allow tracking of history, evolution and usage of data.

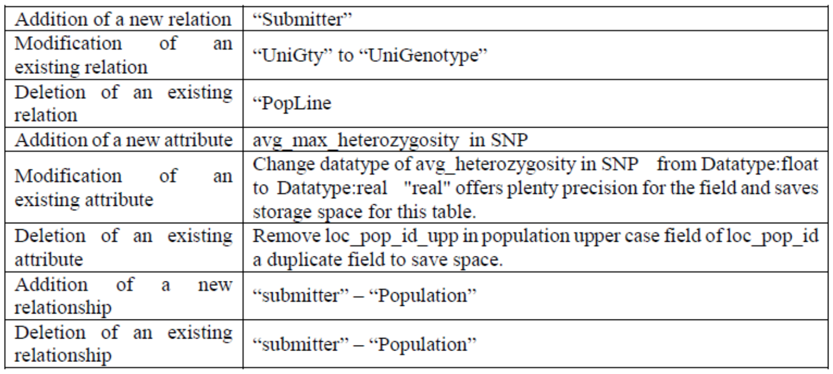
To summarize, the prototype developed will solve primary business requirements, addressing the evolving nature of the data model as well as issues related to the history, and the traceability of the data.

4.3. Experiment

In this experiment, we will measure the effects of data source changes on the raw data vault (RDV) model, the business data vault (BDV) model, the data marts (materialized views), and the ETL package that loads data from the data source to the RDV and from the RDV to the BDV and from the BDV to the DM. We will also evaluate how the system responds when there is a need for traceability and auditability.

In order to determine the changes that should be applied to the EDW, we consider the possible changes in the data source schema. We analyze how these data source schema changes (Table 2) are propagated to the BDV, RDV, ETL, and DM.

Table 2. Set of changes from the data source schema



Addition of a new relation	"Submitter"
Modification of an existing relation	"UniGty" to "UniGenotype"
Deletion of an existing relation	"PopLine
Addition of a new attribute	avg_max_heterozygosity in SNP
Modification of an existing attribute	Change datatype of avg_heterozygosity in SNP from Datatype:float to Datatype:real "real" offers plenty precision for the field and saves storage space for this table.
Deletion of an existing attribute	Remove loc_pop_id_upp in population upper case field of loc_pop_id a duplicate field to save space.
Addition of a new relationship	"submitter" – "Population"
Deletion of an existing relationship	"submitter" – "Population"

4.4. Experiment Results

From the results of the experiment, we were able to demonstrate that the proposed solution resolves issues of tracking the data back to the sources, the history of changes, and the loss of information. In this experiment, we have validated that:

- Any changes to the data source are implemented through additions only. No information or data is ever deleted, which means that the loss of information is avoided and any question about data traceability can be answered.
- The solution provides a solid basis for the audit process due to its ability to record when the data was inserted and where is it coming from. Each row in the RDV and BDV is accompanied by record source and load date information. Using these two attributes, we can answer any questions related to auditability at any given point of time.
- Data Vault dependency minimizations makes it simpler to load tables in parallel, which accelerates the load process while still maintaining the referential integrity. In the data vault methodology, the dependency does not go deeper than three levels, with hubs being on top of the hierarchy, dependent links and hub satellites dependent on hubs and the last link satellites dependent on links.
- The use of the materialized view for data mart has significantly affected the performance of the system, especially when querying a huge amount of data.

4.4.1. Effect Of Changes On The EDW

Table 3 shows a summary of the effects of changes to the data sources on the EDW.

Table 3. Effect of changes in the existing data source schemas on the RDV, BDV, MV, and ETL

Source Schema change	Example	Effect on RDV	Effect on BDV	Effect on DM/MV	Effect on ETL
Addition of a new relation	"Submitter"	- Addition of new hub, link and satellite	- Addition of new master hub, MDM link, master satellite and SAL link	- Creation of new MV for the dimension. (Simple extension of the model)	- Addition of an ETL transformation to loads data from the data source to the RDV new entities (hub, link and satellites). - Addition of an ETL transformation to load data from RDV to BDV new entities.
Modification of an existing relation	"UniGty" to "UniGenotype"	- No action required	- No action required	- No action Required	- Update OLE DB source for the ETL transformation that loads the data from the data source to the RDV (Hub, link and sat)
Deletion of an existing relation	"PopLine	- No deletions, update only. - Addition of new validity satellite for end-dating to hub and its links	- Addition of a new validity satellite to the master hub entity. - No deletions	- No deletions to the DM. - Alteration of the MV population dimension to remove the reference to the Popline dimension. - If the deleted relation is a regular dimension in the DM, an alteration of the Fact table view is required. - If the deleted relation is being used in a fact table, the alteration of the fact view is required to updated the measures.	- Addition of an ETL transformation to load the validity satellite in the RDV. - Addition of an ETL transformation to load the validity satellite in the BDV.
Addition of a new attribute	avg_max_heterozygosity in SNP	- Addition of new satellite for the new attribute or combine it with the old satellite.	- Addition of new master satellite for the new attribute or combine it with the old master satellite.	- A new fact table view will be created with the numeric attribute avg_max_heterozygosity as its plausible measure. - If the attribute is not numeric then addition of this attribute to a pre-existing dimension view is required.	- If new satellite is created for the attribute, an addition of two ETL transformations to load the new satellite in the RDV and BDV is required.

Modification of an existing attribute	Change datatype of avg_heterozygosity in SNP from Datatype:float to Datatype:real	- Addition of a new satellite to store the attribute with the new datatype. - No deletions	- Addition of a new master satellite to store the attribute with the new datatype. - No deletions	- Alteration of view to include the changed attribute is required.	- Addition of an ETL transformation to load the new satellite in the RDV. - Addition of an ETL transformation to load the new master satellite in the BDV
Deletion of an existing attribute	Remove loc_pop_id_upp in population	- No deletion. - Addition of anew validity satellite for end-dating to the population Hub	- No deletions - Addition of a new validity satellite for the end-dating to the population Master Hub	- Alteration of the dimension population(MV) to exclude this attribute.	-No deletions, only additions -Addition of an ETL transformation to load the new validity satellite in the RDV -Addition of an ETL transformation to load the new validity satellite in the BDV
Addition of a new relationship	"submitter" – "Population"	- Simple extension of the model - Addition of a new link (submitter – Population)	-Simple extension of the model -Addition of a new MDM link (submitter – Population)	- No action required	- Addition of a new ETL transformation to load the new link in the RDV. - Addition of a new ETL transformation to load the new MDM link in the BDV.
Deletion of an existing relationship	"submitter" – "Population"	-Creation of a new validity satellite for end-dating on the link population-submitter	- Creation of a new validity satellite for end-dating on the MDM link population - submitter	-No action required.	- Addition of a new ETL transformation to load the new validity satellite to the RDV. - Addition of a new ETL transformation to load the new validity satellite to the BDV.

4.4.2. Evaluation Of Traceability

Data vault is considered as a system of record for the EDW. It ensures the preservation of history and provenance, which enables tracking the lineage of data back to the source [1]. Data coming from different sources are inserted in the RDV model without any transformations to ensure data traceability; the RDV enables permanent system of records, which means that copies of the originals are kept permanently in RDV. This way, the loss of information is avoided and any question about data traceability is answered. In this experiment, we have validated that any changes to the data source are implemented through additions-only (Table 4):

- Historical changes are captured by inserting new links and satellites.
- Tracking change is tolerably complex due to a highly normalized structure.
- Provides the most detailed and auditable capture of changes.

In addition, separating the RDV that is a data source oriented from the BDV that is report oriented means separating reversible and irreversible transformations. Reversible transformations represent ETL patterns used to load data from a data source into the RDV, and it is possible to trace back their effects, in order to obtain the system-of records [1].

4.4.3. Evaluation Of Auditability

Data vault methodology provides a solid basis for the audit process due to its ability to record when the data was inserted and where is it coming from. Each row in the RDV and BDV is accompanied by record source and load date information. Using these two attributes, we are able to answer any questions related to auditability at any given point of time. Thus, we are able to know where a specific information is extracted from, when was it extracted, and what was the process used to extract it.

4.4.4. Etl Performance

In the data vault methodology, the dependency does not go deeper than three levels, with hubs being on top of the hierarchy and then links and hub satellites depended on hubs and the last link satellites dependent on links. Data Vault dependency minimizations makes it easier to load tables in parallel, hence speed up the load process while still maintaining the referential integrity. Moreover, one of the big advantages of DV 2.0 was the replacement of the standard integer surrogate keys with hash-based primary keys. Hashing allows a child “key” to be computed in parallel to the parent “key” and be loaded independently of each other, which highly increases load performance of an Enterprise Data Warehouse especially when working with large volumes of data. To conclude, Data Vault enables greatly simplified ETL transformations in a way not possible with traditional data warehouse designs.

5. Conclusions

Nowadays, The DW environment is constantly changing. However, adapting the DW to these changes is a time-consuming process and still needs more in-depth research. In this thesis, we examine the implications of managing changes to the schema of the operational databases in the context of data warehouses. We propose a Meta Data Vault approach for evolutionary integration of big data sets from the biomedical field. The approach proposed uses an integrated DW and MDM data repository. The DW includes the Raw Data Vault (RDV), that contains the actual copies of the originals from the data sources; and the Business Data Vault (BDV) that stores the data resulted from the execution of business rules.

The BDV is a component of our EDW that is oriented towards the reporting side. It is expanded from the RDV by adding standardized master data and business rules in order to ensure business integration. The main purpose of expanding the RDV with the BDV is to separate the storing and data warehousing portion from the interpretation of the data. This is accomplished by pushing "business rules" downstream after RDV, towards BDV and reporting DW. Furthermore this arrangement enabled evolution of business rules localized to the BDV alone having as a consequence more flexible maintenance also easier to implement using independent parallel tasks.

The RDV and BDV are integrated via the Metadata Repository Data Vault (MDV). A data vault based data warehouse needs an additional layer to deliver analysis reports to the users. Therefore, data marts were derived from the RDV using materialized views. From the results of the experiment, we were able to demonstrate that the proposed solution resolves issues of tracking the data back to the sources, the history of changes, and loss of information. The DV approach also allows fast parallel loading and preservation of the DW integrity. The Meta data vault repository is a complex subject that still needs additional work and research in order to be formalized and standardized even though, the MDV was out of scope in the experiment. The suggested models (RDV, BDV), along with the load patterns used, have proved its flexibility and easy adaptation to the data source schema changes, while preserving the history of these changes using additions only. The model has also proved its efficiency in terms of traceability, auditability, and performance on the overall data warehouse system.

REFERENCES

[1] D.Subotic, V. Jovanovic, and P. Poscic, Data Warehouse and Master Data management Evolution – a Meta-Data-Vault Approach. Issues in Information Systems, 15(Ii), 14–23, 2014.
[2] D.Linstedt, SuperCharge Your Data Warehouse: Invaluable Data Modeling Rules to Implement your Data Vault. Create Space Independent Publishing Platform, USA, 2011.
[3] A.Kitts, L. Phan, W. Minghong, and J. B. Holmes, The database of short genetic variation (dbSNP), The NCBI Handbook, (2nd), 2013. Available at http://www.ncbi.nlm.nih.gov/books/NBK174586/.
[4] H.Afify, M. Islam, and M. Wahed. DNA Lossless Differential Compression Algorithm Based on Similarity of Genomic Sequence Database, International Journal of Computer Science & Information Technology (IJCSIT), 2011.
[5] B.Feldman, E. Martin, and T. Skotnes. Genomics and the role of big data in personalizing the healthcare experience, 2013. Available at https://www.oreilly.com/ideas/genomics-and-the-role-of-big-data-in-personalizing-the-healthcare-experience.
[6] V.Jovanovic, and I. Bojicic. Conceptual Data Vault Model, Proceedings of the Southern Association for Information Systems Conference, Atlanta, GA, USA, 2012.
[7] D. Linstedt, and M. Olschimke, Building a scalable data warehouse with Data Vault 2.0, 2016.
[8] Z. Naamane, and V. Jovanovic, Effectiveness of Data Vault compared to Dimensional Data Marts on Overall Performance of a Data Warehouse System. IJCSI International Journal of Computer Science Issues, Volume 13, Issue 4, 2016
[9] N.Rahman, An empirical study of data warehouse implementation effectiveness. International Journal of Management Science and Engineering Management, 2017
[10] R.Kimball, and M. Ross, The data warehouse toolkit, 3rd edition, John Wiley, 2013.
[11] B.Shah, K. Ramachandran, and V. Raghavan, A Hybrid Approach for Data Warehouse View Selection.International Journal of Data Warehousing and Mining (IJDWM), 2006.
[12] M.Teschke, and A. Ulbrichl, Using materialized views to speed up data warehousing. University Erlangen-Nuremberg (IMMD VI), 1–16 , 1997.
Available at http://www.ceushb.de/forschung/downloads/TeUl97.pdf