Transforming a Multi-value Database System into a Relational Database System for Faster Querying

Submitted By

Md. Zahid Hasan

ID: 2013-2-96-010

Supervised By

Dr. Mohammad Rezwanul Huq

Assistant Professor, Department of CSE, EWU

A thesis

in

The Department of

Computer Science and Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Master of Science

(Computer Science and Engineering)

East West University

Dhaka, Bangladesh

October, 2016

Declaration

This thesis has been submitted to the department of Computer Science and Engineering, East West University in the partial fulfillment of the requirement for the degree of Master of Science in Computer Science and Engineering by me under the supervision of Dr. Mohammad Rezwanul Huq, Assistant Professor at Department of CSE at East West University under the course 'CSE 599' . I am also declaring that this thesis has not been submitted elsewhere for the requirement of any degree or any other purposes. This thesis complies with the regulations of this University and meets the accepted standards with respect to originality and quality. I hereby release this thesis to the public. I also authorize the University or other individuals to make copies of this thesis as needed for scholarly research.

Signature of the candidate

_____

Md. Zahid Hasan

2013-2-96-010

Letter of Acceptance

The thesis entitled "Transforming a Multi-value Database System into a Relational Database System for Faster Querying" submitted by Md. Zahid Hasan, ID 2013-2-96-010 to the department of Computer Science and Engineering, East West University, Dhaka 1212, Bangladesh is accepted as satisfactory for partial fulfillment of requirements for the degree of Master of Science(M.Sc) in Computer Science and Engineering on October, 2016.

Board of Examiners

1._____

Dr. Mohammad Rezwanul Huq — Supervisor
Assistant Professor
Department of Computer Science and Engineering
East West University, Dhaka, Bangladesh

2._____

Dr. Md. Mozammel Huq Azad Khan — Internal Member
Professor and Chairperson
Department of Computer Science and Engineering
East West University, Dhaka, Bangladesh

3._____

Dr. Abu Raihan Mostofa Kamal — External Member
Associate Professor
Department of Computer Science and Engineering
Islamic University of Technology, Gazipur, Bangladesh

Acknowledgments

During the course of this research I have learned time management, critical analysis, research techniques, how to work independently. Although there was time pressure, I have managed it as well.

Firstly, I would like to express my sincere gratitude to my advisor Dr. Mohammad Rezwanul Huq, Assistant Professor at Dept. of CSE for the continuous support during my thesis study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I heartily thank my advisors for helping me to complete this work.

I would like to thanks my employer Datasoft Systems Bangladesh Limited and Temenos (a world renowned Core Banking Solution) for their opportunity to gain knowledge on MVDBMS and allocate time for my research. I am also thankful to all my friends and colleagues during my master's program. We discussed and debate a lot on this topic which enrich my knowledge for this thesis.

I am also grateful to my primary and secondary school teacher who are my first teachers in my life and initiator of my basic knowledge.

Last but not the least; I would like to thank my parents and siblings for supporting me spiritually throughout writing this thesis. And at the very last thanks to the creator Allah for everything.

Abstract

Transforming a Multi-value Database System into a Relational Database System for Faster Querying

Md. Zahid Hasan

When the data is fluid and relationships among attributes are getting complex and are changing rapidly over time, a Multi-value Database Management System (MVDBMS) can provide a great deal of support to handle that. MVDBMSs can handle attributes taking a list of values unlike a Relational Database Management System (RDBMS) that can only accept single-valued attributes to conform first normal form (1NF). It is suitable to use multi-valued database in case where a list of values can be assigned to a single attribute in a particular record. In this case, MVDBMS can save a lot of storage space and it is quite efficient to retrieve a specific record. However, when it comes into retrieving data based on Select-Project-Join (SPJ) queries, a multi-valued database cannot handle these queries in a straightforward manner.

In this thesis, we propose techniques to transform a MVDBMS into a RDBMS for faster querying and retrieval. Therefore, we introduce two novel techniques: one of them will generate database schema conforming to Third Normal Form (3NF) and the other will be in First Normal Form (1NF). 3NF is a normal form that is used in normalizing a database design to reduce the duplication of data and ensure referential integrity. In 1NF the domain of each attribute contains only atomic (indivisible) values, and the value of each attribute contains only a single value from that domain. 1NF create a separate table for each set of related data, so a large number of attributes can be stored into one table but in a set of related data there is

a high possibility of data duplicity. In 3NF all the attributes in a table are determined only by the candidate keys of that table and not by any non-prime attributes; so data duplicity will be reduced but to establish referential integrity it requires more table. As the number of table increase, it will require more joining operation which will cost time.

In out thesis we have found that at the time of 3NF conversion, number of tables required to transform is dependent on some primary aspect of multi-value database systems like number of multi-value attributes, number of sub-value attributes and number of associated attributes. So as number of those types of attribute increase, number of tables required will be increased in a extent. On the other hand, at time of 1NF conversion number of table require for conversion is constant and is very negligible compared to 3NF conversion. We also found that data are more redundant in 1NF compared to 3NF which also conforms the property respectively. We also evaluate the performance of these two techniques for different types of queries. During the transformation, we carefully considered the trade-off in between execution time of queries and storage space consumed by data. By analyzing the outcomes of conversion we called 3NF conversion as 'space efficient conversion' and 1NF conversion as 'time efficient conversion'

Contents

List of Figures

List of Tables

Chapter 1

1 Introduction

In the real world, every moment whatever happening or happened is related to data. Someone is uploading image in facebook, someone has written poem with paper and pencil, all the environment variables of a airplane are storing into black box, HR information of an organization stored in computer memory, breaking news of military attack in news portal, past weather forecast of pacific region etc. all are typical example of data of interest. Data are stored in different format. Whatever the data type is or whatever the storage format is, most important thing is to know about the insight of data. It is important to know about the future trends or to make a business decision. When data is organized and analyzed data becomes information. As information is so important in most organizations, computer scientists have developed a large body of concepts and techniques for managing data. We will discuss on data, how it can be stored and how we can get inner view of those data.

1.1 Data and Information

Data is a set of values of qualitative or quantitative variables. Data is distinct pieces of information, usually formatted in a special way. Data can exist in a variety of forms as numbers or text on pieces of paper, as bits and bytes stored in electronic memory, or as facts stored in a person's mind. Data is measured, collected and reported, and analyzed, whereupon it can be visualized using graphs, images or other analysis tools. Data is collected by a huge range of organizations and institutions, ranging from businesses (e.g., sales data, revenue, profits, stock price), governments (e.g., crime rates, unemployment rates, literacy rates) and

non-governmental organizations (e.g., censuses of the number of homeless people by non-profit organizations).

Data as a general concept refers to the fact that some existing information or knowledge is represented or coded in some form suitable for better usage or processing. Data, information, knowledge and wisdom are closely related concepts, but each has its own role in relation to the other, and each term has its own meaning.

Pieces of data are individual pieces of information. Data is raw, unorganized facts that need to be processed. Data can be something simple and seemingly random and useless until it is organized. Data are simply facts or figures — bits of information, but not information itself. When data are processed, interpreted, organized, structured or presented so as to make them meaningful or useful, they are called information. Data only becomes information suitable for making decisions once it has been analyzed in some fashion. Data becomes information by interpretation. Knowledge is derived from extensive amounts of experience dealing with information on a subject. That is to say, data is the least abstract, information the next least, and knowledge the most abstract. Information provides context for data. For example, a list of dates — data — is meaningless without the information that makes the dates relevant (dates of holiday). "Data" and "information" are intricately tied together, whether one is recognizing them as two separate words or using them interchangeably, as is common today.

Examples of Data and Information:

- The history of temperature readings all over the world for the past 100 years is data. If this data is organized and analyzed to find that global temperature is rising, then that is information.

- The number of visitors to a website by country is an example of data. Finding out that traffic from the Dhaka is increasing while that from Chittagong is decreasing is meaningful information.
- Often data is required to back up a claim or conclusion (information) derived or deduced from it. For example, before a drug is approved by the FDA, the manufacturer must conduct clinical trials and present a lot of data to demonstrate that the drug is safe.

1.2 Database and DBMS

A database is an organized collection of data. It is the collection of schemas, tables, queries, reports, views, and other objects. The data are typically organized to model aspects of reality in a way that supports processes requiring information.

A database-management system (DBMS) is a collection of interrelated data and a set of programs to access those data. The collection of data, usually referred to as the database, contains information relevant to an enterprise. The primary goal of a DBMS is to provide a way to store and retrieve database information that is both convenient and efficient.

Formally, a "database" refers to a set of related data and the way it is organized. Access to these data is usually provided by a "database management system" (DBMS) consisting of an integrated set of computer software that allows users to interact with one or more databases and provides access to all of the data contained in the database (although restrictions may exist that limit access to particular data). The DBMS provides various functions that allow entry, storage and retrieval of large quantities of information and provides ways to manage how that information is organized. Because of the close relationship

between them, the term "database" is often used casually to refer to both a database and the DBMS used to manipulate it.

Databases are widely used. Here are some representative applications:

- *Enterprise Information*
    - *Sales*: For customer, product, and purchase information.
    - *Accounting*: For payments, receipts, account balances, assets and other accounting information.
    - *Human resources*: For information about employees, salaries, payroll taxes, and benefits, and for generation of paychecks.
    - *Manufacturing*: For management of the supply chain and for tracking production
    - of items in factories, inventories of items in warehouses and stores, and orders for items.
- *Online retailers*: For sales data noted above plus online order tracking, generation of recommendation lists, and maintenance of online product evaluations.
- *Banking and Finance*
    - *Banking*: For customer information, accounts, loans, and banking transactions.
    - *Credit card transactions*: For purchases on credit cards and generation of monthly statements.
    - *Finance*: For storing information about holdings, sales, and purchases of financial instruments such as stocks and bonds; also for storing real-time market data to enable online trading by customers and automated trading by the firm.

- *Universities*: For student information, course registrations, and grades (in addition to standard enterprise information such as human resources and accounting).
- *Airlines*: For reservations and schedule information. Airlines were among the first to use databases in a geographically distributed manner.
- *Telecommunication*: For keeping records of calls made, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about the communication networks.

Over the course of the last four decades of the twentieth century, use of databases grew in all enterprises. In the early days, very few people interacted directly with database systems, although without realizing it, they interacted with databases indirectly—through printed reports such as credit card statements, or through agents such as bank tellers and airline reservation agents. Then automated teller machines came along and let users interact directly with databases. Phone interfaces to computers (interactive voice-response systems) also allowed users to deal directly with databases—a caller could dial a number, and press phone keys to enter information or to select alternative options, to find flight arrival/departure times, for example, or to register for courses in a university.

The Internet revolution of the late 1990s sharply increased direct user access to databases. Organizations converted many of their phone interfaces to databases into Web interfaces, and made a variety of services and information available online. For instance, when you access an online bookstore and browse a book or music collection, you are accessing data stored in a database. When you enter an order online, your order is stored in a database. When you access a bank Web site and retrieve your bank balance and transaction information, the information is retrieved from the bank's database system. When you access a

Web site, information about you may be retrieved from a database to select which advertisements you should see. Furthermore, data about your Web accesses may be stored in a database.

Thus, although user interfaces hide details of access to a database, and most people are not even aware they are dealing with a database, accessing databases forms an essential part of almost everyone's life today.

Existing DBMSs provide various functions that allow management of a database and its data which can be classified into four main functional groups:

- Data definition – Creation, modification and removal of definitions that define the organization of the data.
- Update – Insertion, modification, and deletion of the actual data.
- Retrieval – Providing information in a form directly usable or for further processing by other applications. The retrieved data may be made available in a form basically the same as it is stored in the database or in a new form obtained by altering or combining existing data from the database.
- Administration – Registering and monitoring users, enforcing data security, monitoring performance, maintaining data integrity, dealing with concurrency control, and recovering information that has been corrupted by some event such as an unexpected system failure.

1.3 Motivation

There are some systems which use multi-value database as their data storage; a type of NoSQL and multidimensional database. In this system a single field can store more than one value in a single record. These values are called multi-value of that field. It can be extend up-to sub-value for a specific field. Storing and updating a particular record in this method works fine in applications where the relationships are complex and can change rapidly and drastically over time. But when we consider the case of querying information by joining table or extracting information of sub-values/multi-values within same table; performance will degrade as number of records increase.

In multi-value database management systems there is no relationship between data sets or among multi-value/sub-value fields or between one table with other tables. Database native functionality like indexing, constraints can't be imposed in multi-value database systems. Each column data are independent with each other within a record as well as tables are also independent. In this scenario it is difficult to extract data.

Some business cases where multi-value database suits more than RDBMS for storing, retrieving (only a specific record) or updating record. If we efficiently query/extract data from more than one table of multi-value database systems then its searching issue would be resolved and it can be more usable. We are proposing a technique to convert multi-value database systems to RDBMS, so that we can define relationship between tables, convert multi-value/sub-value into a number of single values; so that we can search information by join table or retrieve information of multi-value, sub-value in efficient way. In our thesis we use MVDBMS and NoSQL as interchangeable.

1.4 Thesis Overview

Our main goal is to perform query faster for multi-value database management systems(MVDBMS). It is suitable to use multi-valued database in case where a list of values can be assigned to a single attribute in a particular record. In this case, MVDBMS can save a lot of storage space and it is quite efficient to retrieve a specific record. But when it comes into retrieving data based on Select-Project-Join (SPJ) queries, a multi-valued database cannot handle these queries in a straightforward manner. In our thesis we propose to convert MVDBMS to relational database to overcome scenario of SPJ and more complex query.

Our thesis has two main parts; first one is to transform multi-value database into relational database and second one is to compare performance of different conversion methods. We propose two technique to transform MVDBMS to RDBMS:

- Space Efficient Technique: In this technique we convert MVDBMS to RDBMS in such a way that it generates data in 3NF and load data accordingly. In this form data are in higher normal form, so it removes redundancy as much as possible; so it saves memory space comparatively but for large number of generated table it decrease the query performance in terms of time.
- Time Efficient Technique: In this technique we convert MVDBMS to RDBMS in such a way that it generates data in 1NF and load data accordingly. In this form data are in lesser normal form, so data redundancy is very common here and much more than first technique; so it takes more memory space compared to first technique but takes less time than first one to get insight data.

Later on we compare our two technique in respect of space and time at the time of getting same insight data. We get result as the name indicates "Space Efficient Technique" is space

efficient and time inefficient; on the other hand "Time Efficient Technique" is time efficient and space inefficient.

In this thesis, we use Amazon jBase as MVDBMS. The other multi-value databases can be studied in the same way.

Chapter 2

2 Background

We live in an information-centric world. As a society, we have a growing reliance on creating and consuming data, which must be available when and where it is needed. Data and related information services are enabled or provided via information technology services combining database applications, facilities, networks, servers, storage hardware, and software resources. We will concentrate on database on this chapter.

A database management system is important because it manages data efficiently and allows users to perform multiple tasks with ease. A database management system stores, organizes and manages a large amount of information within a single software application. Use of this system increases efficiency of business operations and reduces overall costs. Without database management, tasks have to be done manually and take more time. Data can be categorized and structured to suit the needs of the company or organization. Multiple users can use the system at the same time in different ways.

There are various types of database based on their data restore technique. We will concentrate only on relational database and multi-value database in our thesis.

2.1 Relational Database

A database can be understood as a collection of related files. How those files are related depends on the model used. The relational database model allowed files to be related by means of a common field. In order to relate any two files, they simply need to have a common field, which makes the model extremely flexible. The foundational idea underneath

relational databases is a simple but powerful structure. Each table is a set of sets, and within a single table all of these sets have the same data structure, containing a list of named fields and their values. SQL is a declarative language in which the expected result or operation is given without the specific details about how to accomplish the task. SQL, which is an abbreviation for Structured Query Language, is a language to request data from a database, to add, update, or remove data within a database, or to manipulate the metadata of the database. The steps required to execute SQL statements are handled transparently by the database.

2.1.1 Relational Model

The relational model (RM) for database management is an approach to managing data using a structure and language consistent with first-order predicate logic, first described in 1969 by Edgar F. Codd. In the relational model of a database, all data is represented in terms of tuples, grouped into relations. A database organized in terms of the relational model is a relational database. The purpose of the relational model is to provide a declarative method for specifying data and queries: users directly state what information the database contains and what information they want from it, and let the database management system software take care of describing data structures for storing the data and retrieval procedures for answering queries.

The relational model's central idea is to describe a database as a collection of predicates over a finite set of predicate variables, describing constraints on the possible values and combinations of values. The content of the database at any given time is a finite (logical) model of the database, i.e. a set of relations, one per predicate variable, such that all

predicates are satisfied. A request for information from the database (a database query) is also a predicate.

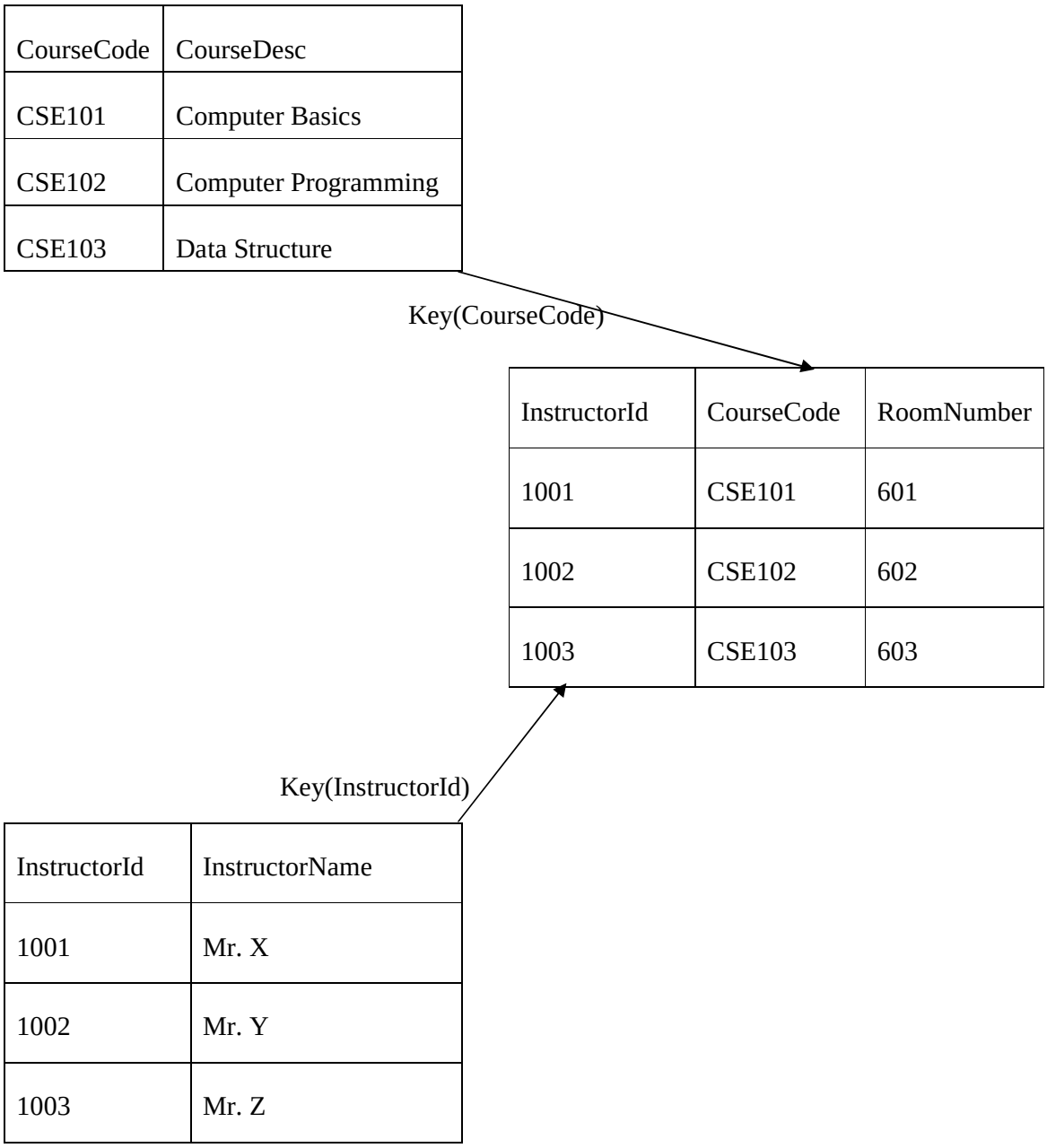

Figure 2.1: An example database according to the relational model

Relational data model is the primary data model, which is used widely around the world for data storage and processing. This model is simple and it has all the properties and

capabilities required to process data with storage efficiency. The relational data model is based on a collection of tables. The user of the database system may query these tables, insert new tuples, delete tuples, and update (modify) tuples. There are several languages for expressing these operations. Some important terms in relational data models:

Tables − In relational data model, relations are saved in the format of Tables. This format stores the relation among entities. A table has rows and columns, where rows represents records and columns represent the attributes.

Tuple − A single row of a table, which contains a single record for that relation is called a tuple.

Relation instance − A finite set of tuples in the relational database system represents relation instance. Relation instances do not have duplicate tuples.

Relation schema − A relation schema describes the relation name (table name), attributes, and their names.

Relation key − Each row has one or more attributes, known as relation key, which can identify the row in the relation (table) uniquely.

Attribute domain − Every attribute has some pre-defined value scope, known as attribute domain.

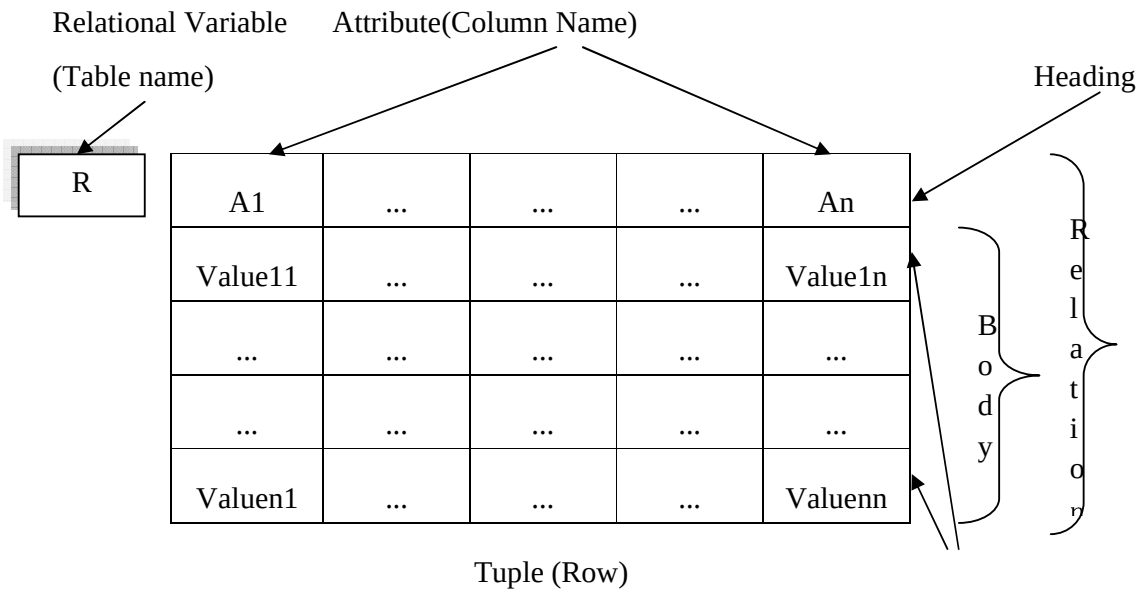

Figure 2.2: Relational model concepts

2.1.2 Database Schema

A database schema of a database system is its structure described in a formal language supported by the database management system (DBMS). A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data. The term "schema" refers to the organization of data as a blueprint of how the database is constructed (divided into database tables in the case of relational databases). The formal definition of a database schema is a set of formulas called integrity constraints imposed on a database. These integrity constraints ensure compatibility between parts of the schema. All constraints are expressible in the same language. This describes how real-world entities are modeled in the database.

A database schema specifies, based on the database administrator's knowledge of possible applications, the facts that can enter the database, or those of interest to the possible

end-users. Thus a schema can contain formulas representing integrity constraints specifically for an application and the constraints specifically for a type of database, all expressed in the same database language. A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It's the database designers who design the schema to help programmers understand the database and make it useful. In a relational database, the schema defines the tables, fields, relationships, views, indexes, packages, procedures, functions, queues, triggers, types, sequences, materialized views, synonyms, database links, directories, XML schemas, and other elements.

A database schema can be divided broadly into two categories −

- **A physical data model** is a representation of a data design as implemented, or intended to be implemented, in a database management system. In the lifecycle of a project it typically derives from a logical data model, though it may be reverse-engineered from a given database implementation. A complete physical data model will include all the database artifacts required to create relationships between tables or to achieve performance goals, such as indexes, constraint definitions, linking tables, partitioned tables or clusters. Analysts can usually use a physical data model to calculate storage estimates; it may include storage allocation details for a given database system.
- **Logical Database Schema** − It defines all the logical constraints that need to be applied on the data. It defines tables, views, and integrity constraints. Logical data models represent the abstract structure of a domain of information. They are often diagrammatic in nature and are most typically used in business processes that seek to

capture things of importance to an organization and how they relate to one another. Once validated and approved, the logical data model can become the basis of a physical data model and form the design of a database.

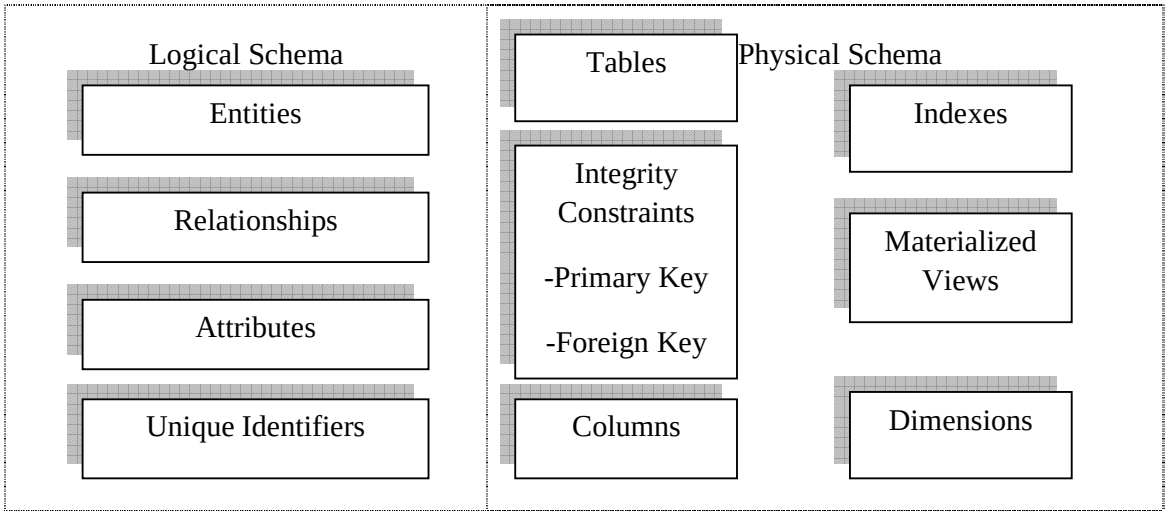

Figure 2.3: Logical Design Compared with Physical Design

2.1.3 RDBMS and SQL

RDBMS stands for Relational Database Management System. A relational database management system (RDBMS) is a database management system (DBMS) that is based on the relational model as invented by E. F. Codd, of IBM's San Jose Research Laboratory.

RDBMS data is structured in database tables, fields and records. Each RDBMS table consists of database table rows. Each database table row consists of one or more database table fields. RDBMS store the data into collection of tables, which might be related by common fields (database table columns). RDBMS also provide relational operators to manipulate the data stored into the database tables. A relational database management system (RDBMS) is a program that lets you create, update, and administer a relational database.

Most commercial RDBMS's use the Structured Query Language (SQL) to access the database.

SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in relational database.SQL is the standard language for Relation Database System. All relational database management systems like MySQL, MS Access, Oracle, Sybase, Informix, postgres and SQL Server use SQL as standard database language.

Originally based upon relational algebra and tuple relational calculus, SQL consists of a data definition language, data manipulation language, and Data Control Language. The standard SQL commands to interact with relational databases are CREATE, SELECT, INSERT, UPDATE, DELETE and DROP. These commands can be classified into groups based on their nature:

DDL - Data Definition Language:

Command	Description
CREATE	Creates a new table, a view of a table, or other object in database
ALTER	Modifies an existing database object, such as a table.
DELETE	Deletes an entire table, a view of a table or other object in the database.

Table 2.1: Data Definition Language

DML - Data Manipulation Language:

Command	Description
SELECT	Retrieves certain records from one or more tables
INSERT	Creates a record
UPDATE	Modifies records
DELETE	Deletes records

Table 2.2: Data Manipulation Language

DCL - Data Control Language:

Command	Description
GRANT	Gives a privilege to user
REVOKE	Takes back privileges granted from user

Table 2.3: Data Control Language

2.2 Multi-value Database

MVDBMSs can handle attributes taking a list of values unlike a Relational Database Management System (RDBMS) that can only accept single-valued attributes. It is suitable to use multi-valued database in case where a list of values can be assigned to a single attribute in a particular record. The first multi-value databases were designed in the 1960s and it was developed for the army. multi-value databases are a type of NoSQL and multidimensional databases that understand 3 dimensional data directly. MV databases can work with relatively low-powered servers and achieve great performances. They are primarily giant strings that are perfect for manipulating HTML and XML strings directly and they differ from relational databases as they encourage using attributes that can take a list of values rather than just single-values.

2.2.1 Introduction to MVDBMS

In case of relational database, it is not allowed to keep more than one instance of value in any column, so we have to keep each instance separately against same record id. It will make data redundant which can be solved using multi-valued database. The biggest benefit is that it means you don't need to maintain more tables like RDBMS to fulfill the full operational process of a business model. In case of MVDBMS, in one table you can

store/maintain large number of fields; sometimes all field values of a record in a single table. As MVDBMS store data in flat file, it doesn't establish relationship between them. So it easy to store, retrieve, update that particular record. For this, sometimes refer to them as record based database systems. jBASE, Revelation, Ladybridge, InterSystems are some examples of multi-valued database. In a Multi-Value database system:

- a database or schema is called an "account"
- a table or collection is called a "file"
- a column or field is called a field or an "attribute", which is composed of "multi-value attributes" and "sub-value attributes" to store multiple values in the same attribute.
- a row or document is called a "record" or "item"

For example, assume there's a file (table) called "CUSTOMER". In this file, there is an attribute called "eMailAddress". The eMailAddress field can store a variable number of email address values in a single record. The list [a@example.com, b@example.net, c@example.org] can be stored and accessed via a single query when accessing the associated record.

Achieving the same (one-to-many) relationship within a traditional relational database system would include creating an additional table to store the variable number of email addresses associated with a single "CUSTOER" record.

Data is stored using two separate files: a file to store raw data and a dictionary to store the format for displaying the raw data. Note that in case of MVDBMS term 'file' and 'table' are interchangeable.

2.2.2 Multi-value, Sub-value and Associated Value

In a multi-value database systems, more than one value may exists in an attribute. Such multiple values within a single attribute/field is called multi-value. For example in a 'TRANSACTION' table(record id: account number), against a single account record multiple transaction date may exists like "1001"^"20160901::2060902::20160903". Here 1001 is account number as record id of "TRANSACTION" table and it has multiple value 20160901,2060902,20160903 as transaction date.

Same way in multi-value database systems, more than one value may exists in a single multi-value of an attribute. Such multiple values within a single multi-value under a single attribute/field is called sub-value. For example in a 'TRANSACTION' table(record id: account number), against a single account record multiple transaction date may exists as well as multiple transaction amount may exists against each transaction date like "1001"^"20160901::2060902::20160903"^"10~-5::20~10~-15::-10~5". Here 1001 is account number as record id of "TRANSACTION" table and it has multiple value 20160901, 2060902, 20160903 as transaction date. And on transaction date 20160901, 20160902, 20160903 it has transaction amount 10 and -5, 20 and 10 and -15, -10 and 5 respectively. So each transaction amount is sub-value against corresponding transaction date.

In multi-value there are basically 3 levels of data storage within a record:

1	Attributes	Separated by field delimiter (Here it is ^)
2	Multi-Values	Separated by multi-value delimiter (Here it is ::)
3	Sub-Values	Separated by sub-value delimiter (Here it is ~)

Table 2.4: Different types of delimiter

If more than one attribute/fields are grouped/associated with each other than that set of fields are called associated field set and each field is called associated field. This property

is very common to multi-value database systems. For example in a 'TRANSACTION' table(record id: account number), against a single account record multiple transaction date and transaction number may exists like "1001"^"20160901::2060902::20160903"^"2::3::2". Here 1001 is account number as record id of "TRANSACTION" table and it has multiple transaction date 20160901,2060902,20160903 and 2,3,2 are number of transaction for each transaction date respectively. So here transaction date and transaction number are associated field. On our sub-value example transaction date and transaction amount are also associated.

2.2.3 Mechanism in MVDBMS

MVDBMS just simply use different methods to store values. It uses a set of delimiters to separate the values, multi-values and sub-values of the record. Suppose we have meta data of a table ‘Account’ under multi value database system is:

FIELD_NAME	LEN	MULTI_MARK	ASSOC
Id	20		
account_title	35		
email_id	35	M	
mobile_no	20	M	
txn_date	8	M	Y
no_of_txn	2	M	Y
txn_type	2	S	Y
txn_amt	20	S	Y

Table 2.5: Dictionary file of Account table

Suppose there is a record with ID 10001 of ‘Account’ file:

10001

ZahidHasan,a@yahoo.com::b@yahoo.com,8801921033322::8801721033322,20141220::201

41221,1::2,CR::CR~DR,10::20~30

Data will be mapped against dictionary file as follows:

FIELD_NAME	FIELD_VALUES
Id	10001
account_title	Zahid Hasan
email_id	a@yahoo.com::b@yahoo.com
mobile_no	8801921033322::8801721033322
txn_date	20141220::20141221
no_of_txn	1::2
txn_type	CR::CR~DR
txn_amt	10::20~30

Table 2.6: Data representation of Account table

Here email_id, mobile_no, txn_date, no_of_txn are multi-value fields and separated by '::' delimiter. txn_type, txn_amt are sub-value fields and separated by '~' delimiter.

Another important property of MVDBMS is field association. In dictionary file example note that txn_date, no_of_txn, txn_type, txn_amt fields are associated (ASSOC column is Y). We can observe the impact of this property on data part example. For Account id 10001 there are two transaction date; one on 20/12/2014 with transaction type 'Credit' with transaction amount 10. For second multi-value of txn_date all other association can be understood. For associated field set, values are grouped together. So for an associated field set, no of multi-value for each member field under association must be same. Same way no of sub-value for each sub-valued member field under association must be same. In example txn_date, no_of_txn, txn_type, txn_amt all have 2 multi-values and under each multi-value of a sub-valued field, no of sub-value is also equal. In example there are two sub-valued field txn_type, txn_amt under association; for each field, for first multi-value set, no of sub-value under this is 1 and for second multi-value set, no of sub-value under this are 2. We will discuss the mechanism to transform MVDBMS to RDBMS for efficient query by joining tables as well as extract multi-value, sub-value data to overcome the limitation of MVDBMS on data query.

Chapter 3

3 Related Work

Our area of interest include multi value database. In [1] [5] it states for today's interactive web and mobile applications the importance of flexibility and scalability in data model can't be over-stated and NoSQL databases are currently being used by Google, Amazon, Facebook and many other major organizations operating in the era of Web 2.0. In addition to SQL databases, cloud applications usually store their data in NoSQL databases as in[2]. It [4] shows every day people and companies generate enormous amounts of data and this data may be structured, unstructured, semi-structured or a combination of all. This has called [5] for the need to design NoSQL databases which can store this type and volume of data.

In the circumstances like today's informative world where data comes from various source like social media, news portal, renowned poet/novelist, financial institutes, search engine, mechanical/electrical/electronic device in form of audio, video, image, number and character, text etc. To simply store and retrieval of data is not easy as those data volume is very huge and data are coming very fast. Companies are facing these challenges in a climate where they have the ability to store anything and they are generating data like never before in history. Storing those data in relational form is quite difficult for its volume, velocity and variety of data. Nosql/multi-value is the ultimate choice to store those data.

However, in [4] shows the NoSQL solution to this Big Data problem has also lead to several other problems. Storing or retrieving a particular record is good enough in multi-value database but when it comes to query a data by join operation, most multi-value databases do

not support join operations[2]. Therefore, the traditional relational data modeling approach is not suitable for NoSQL databases, and it makes data query more complex in NoSQL databases[2]. The challenges of NoSQL databases specially in [2] [4] [6] the field of 'Business Intelligence and Analytics' can't be overlooked.

NoSQL databases have evolved to meet the scaling demands of modern Web 2.0 applications. Consequently, most of their feature set is oriented toward the demands of these applications. However, data in an application has value to the business that goes beyond the insert-read update-delete cycle of a typical Web application. Businesses mine information in corporate databases to improve their efficiency and competitiveness, and business intelligence (BI) is a key IT issue for all medium to large companies. NoSQL databases offer few facilities for ad-hoc query and analysis[2]. Even a simple query requires significant programming expertise, and commonly used BI tools do not provide connectivity to NoSQL.

It is very important to get the insight of data for making business decision. Business Intelligence is a concept that typically involves the delivery and integration of relevant and useful business information in an organization. As such, companies use business intelligence to detect significant events and identify/monitor business trends in order to adapt quickly to their changing environment or scenario. At the end of the day it is important to be able to query data easily, reliably, accurately and declaratively. Existing relational database technology experience tells us that declarative database query languages are beneficial for different reasons, including application independence from access path considerations and decades of sophisticated query optimization technologies [7]. To resolve the SPJ query on multi-value data source, transforming it into relational model can be a good choice. Transforming multi-value database into RDBMS also tends to build resourceful data

warehouse which lead to achieve business intelligence. An argument can be made that SQL is actually going to be an important success factor for NoSQL databases [7].

Most NoSQL databases are extremely painful for their users when querying data: in order to retrieve data, users must write a program in their favorite programming language and execute it (rinse and repeat for every query). Most NoSQL databases expose only a limited programming language interface[6][7][11], but usually not a declarative query language like SQL. It is therefore not surprising that SQL compliance has been one of the most requested additions to the Google App Engine platform [10].It shows [6][9] how different NoSQL databases uses third party API for ad-HOC query generation ; even some don't have any option to do that [11] but the level of programming expertise in writing queries needs to be much higher than that of a relational database. Following table shows the comparison between different NoSQL databases in respect of performing Ad-HOC query.

Database Tool	**Ad-HOC query**
DynamoDB	Built in API
Riak	CorrugatedIron
Voldermort	NO
Tokyo Cabinet	NO
CouchDB	Cloudant, Lucene
MongoDB	BSON based format
RavenDB	Built in, Limited
Cassandra	HIVE, PIG
Hbase	HIVE, PIG
Neo4j	Chyper

Table 3.1: A Comparison Of Different NoSQL Databases

From the comparison table we can see that some of the NoSQL database even don't have any Ad-HOC query facility; in addition some of them have their different API which are distinct. Each have been built based on their own API, it requires programming language skill to execute Ad-HOC query. Also for different NoSQL different programming language skill required. However, their lack of SQL support represents a major hurdle for a wider adoption.

This arises at different levels due to the prevalence of SQL as the standard, widely mastered and efficient query language for databases. A large number of tools and middleware coupled to SQL have been developed and matured over the years and are currently at the basis of most application development frameworks[10].

It will be very handy to build a common SQL interface which can give all multi-value database for a standard query processing platform. As SQL has a common programming language, it will diminish the need of different programming language skill or different platform for different NoSQL databases in respect of query processing. In this scenario converting NoSQL/ multi-value database into a relational database can be a unique Ad-HOC query solution for all multi-value databases. Relational database has some other features like expertise, administration, support, maturity level over NoSQL databases[4].

Chapter 4

4 Architecture

Incoming data can come in various form; from external source or internal source. Data sources can be database/csv files/text files/flat file. Whatever source it can be; it will must have field marker, value marker and sub-value marker to distinguish between field value, multi-value and sub-value. Those marker/delimiter is very significant on transformation process. On the other hand to identify attribute property (like single valued/multi-valued/sub-valued field, associated or not, association group, field details etc.) it require a metadata of source file. It can be found from existing multi-value database systems or from existing work flow. Metadata and data source handed over to STD unit, which stands for Scan-Transform-Database unit. STD unit's output is required relational schema with data.

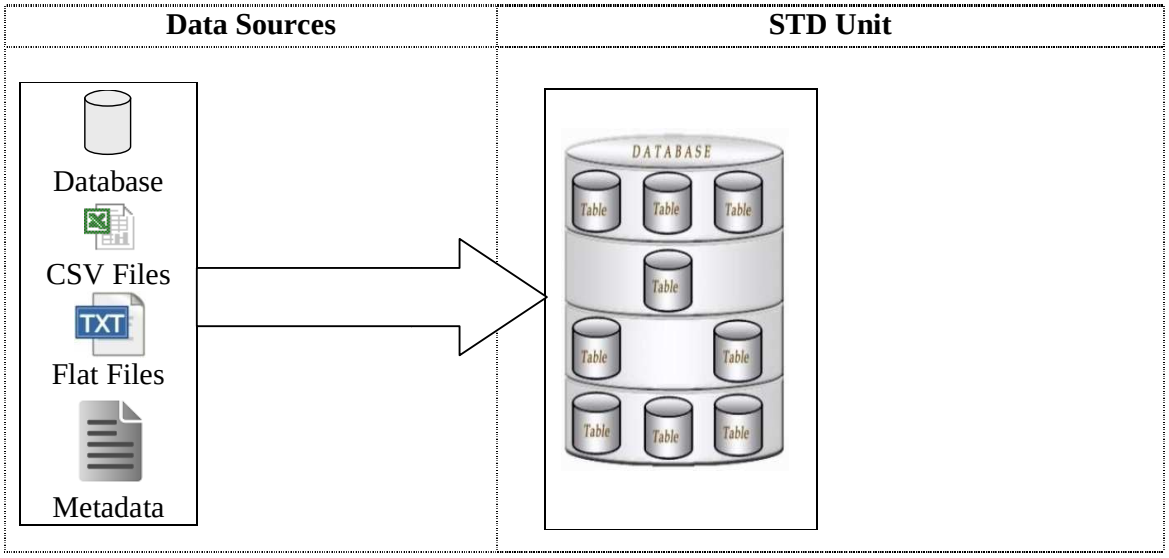

Figure 4.1: Transformation mechanism

4.1 STD Unit

STD unit consists of three units:

- **Scan unit**: This unit reads the metadata of source data to find-out the table property as well as parse incoming data to segregate field value, multi-value and sub-value.
- **Transform unit**: It takes input of scan unit. This unit prepares strings for table schema according to metadata. It also prepare data string from source data for inserting into the respective table.
- **Database unit** : It takes strings as input from transform unit. It crate table according to table schema prepared by transform unit; it also insert table rows according to data string prepared by transform unit.

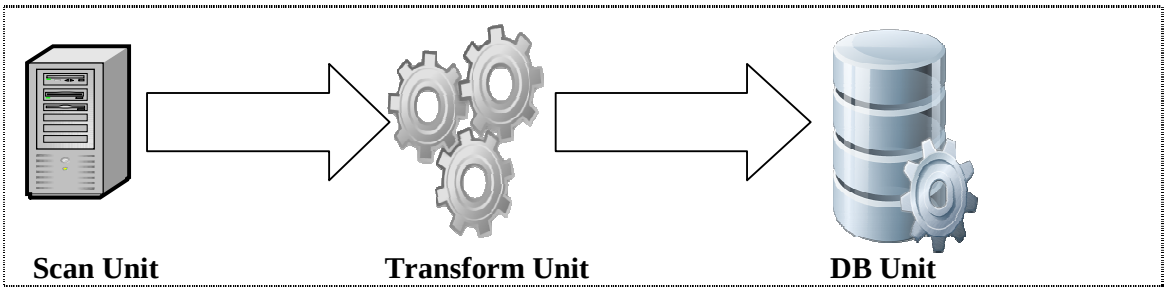

Figure 4.2: STD unit

4.2 Workflow of STD unit

Workflow of STD unit consists of following steps respectively:

1. First it scan unit scan metadata of received file to read the properties of table.
2. Transform unit capture table properties from scan unit; then prepare table schema accordingly.
3. DB unit creates table according to table schema prepared by transform unit.

4. Then scan unit parse the incoming data to find out distinct values, multi-values, sub-values, associated fields.
5. Transform unit prepare string for insert command for each row according to scan unit parsing.
6. Finally DB unit insert each row according to insert command prepared by transform unit.

Building block of STD unit:

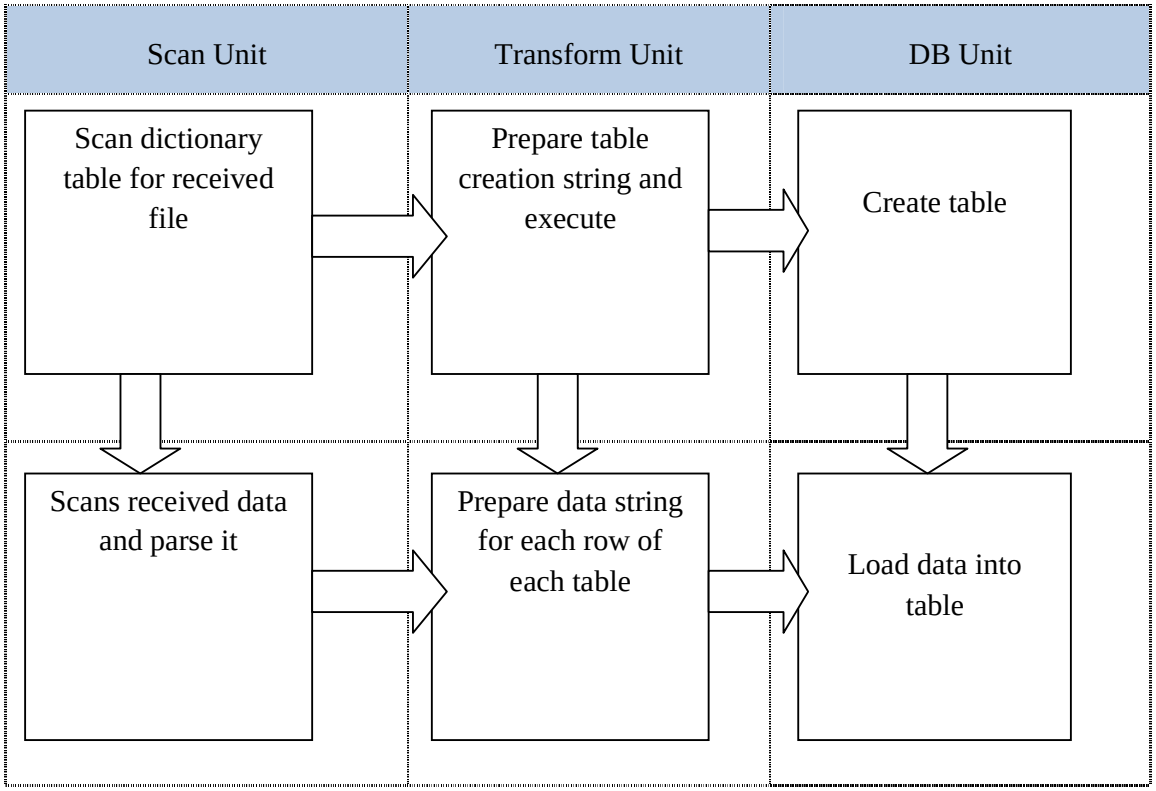

Figure 4.3: Block diagram of STD Unit

Chapter 5

5 Working Principle

Our objective is to convert data from multi value database systems to RDBMS so that data can be searched more efficiently. We propose two evaluation criteria to transform the system:

A. Space efficient transformation

B. Time efficient transformation

A. Space efficient transformation

This technique transform source data into normal form as higher as possible. In this technique source file generally split into many tables into RDBMS. In file, it will have data consisting field values, multi values and sub values of records; values can be found in association or non association mode. If is association found then a set of values are related/grouped to each other within that associated group. In association it may have only multi value set or mixed of multi value and sub value set. If multi value columns are not associated then to normalize each multi value column of a records it will require a separate table. Also in case of associated multi value set of column if it has no sub value then for that whole associated set it will require one table. So it will require ***n*** number of separate columns to normalize ***n*** number of associated multi value set of columns. Additionally columns which are single valued; means which are not multi valued or sub valued, for all single valued columns it will require one table. Now consider case when an associated value set consists of multi value and sub value set. In this scenario for multi value data set it require one table as

well as for sub value data set it will require one more table to normalize the associated data set. So we can list out the cases how data values can form a record:

Case-1: Data values will have some single-valued columns.

Case-2: Data values may have one or more non associated number of multi- valued columns.

Case-3: Data values may have one or more associated number of columns where association exists only within multi-valued columns.

Case-4: Data values may have one or more associated number of columns where association exists with mixed of multi-valued and sub-valued columns.

Note that if a sub value column is exists in any data position it must be under association of a multi value column. A sub value column will never exist separately out of association. In this technique we have taken first multi value column as its identity. In the same way multi value column will never exists without a single value column but unlike sub value it may come without association. Actually this multi value column is associated with the first single value column (first column of a record, normally id of the record). So in case of sub value column it will require record id along with first multi value column of its association to fully identify it.

So we can set an expression for number of tables requires normalizing multi value data systems. Suppose,

A=Number of non associated multi-valued column.

B= Number of associated value set which contain only multi-valued column set.

C= Number of associated value set which contain both multi- valued and sub-valued column set.

Total number of table require in this technique,

$$X = 1 + A + B + 2C \qquad (5.1)$$

Suppose we have Meta data of a table ‘account’ under MVDBMS is:

FIELD_NAME	LEN	MULTI_MARK	ASSOC
account_id	20		
Category	5		
first_name	35		
email_id	35	M	
txn_date	8	M	Y
txn_type	2	S	Y
txn_amt	20	S	Y
int_rate	6		
doc_name	20	M	Y
doc_id	20	M	Y
branch_id	20		

Table 5.1: Metadata of Account file

In this method, number of table required as follows:

Number of non associated multi value column, *A* = 1(Column email_id)

Number of associated value set which contain only multi value column set, *B* = 1(Columns doc_name, doc_id)

Number of associated value set which contain both multi value and sub value column set, *C* = 1(txn_date, txn_type, txn_amt)

Single value columns are account_id, category, first_name, int_rate, branch_id.

So total number of tables require to normalize according to this method,

$$X = 1 + A + B + 2C = 5$$

The table structure will be as follows:

For all single value columns, it create table account as:

```
CREATE TABLE account
(
  account_id    varchar2(35),
  category      number(3),
  first_name    number(4),
  int_rate      varchar2(1),
  branch_id     varchar2(1)
)
```

Figure 5.1: Schema-A1

For non associated multi value column, it create table named account_email_id as follows:

```
CREATE TABLE account_email_id
(
  account_id      varchar2(35),
  email_id        varchar2(35)
)
```

Figure 5.2: Schema-A2

Note that all table (except single value columns) table name convention will be name of the columns one after another prefixed by main table but maximum length will not exceed 30.

For associated value set which contain only multi value column set, it create table account_doc_name_doc_id as:

```
CREATE TABLE account_doc_name_doc_id
(
  account_id      varchar2(35),
  doc_name        varchar2(20),
  doc_id          varchar2(20)
)
```

Figure 5.3: Schema-A3

For associated value set which contain both multi value and sub value column set, it will generate two tables. One named as account_txn_date as follows:

```
CREATE TABLE account_txn_date
(
  account_id    varchar2(35),
  txn_date      varchar2(8)
)
```

Figure 5.4: Schema-A4

Another one named as account_txn_date_txn_type_txn_amt as follows:

```
CREATE TABLE account_txn_date_txn_type_txn_amt
(
  account_id      varchar2(35),
  txn_date        varchar2(8),
  txn_type        varchar2(2),
  txn_amt         varchar2(20)
)
```

Figure 5.5: Schema-A5

Now concentrate on how data is going to be parsed and get populate into table in this technique. Consider the case that already discussed above(Table-5.1)

Suppose value of the following column account_id, category, first_name, email_id, txn_date, txn_type, txn_amt, int_rate, doc_name, doc_id, branch_id is stored like this:

0004121000001,6001,A,a@yahoo.com::b@yahoo.com,20141220::20141221::20141223,CR::CR~DR::DR,10::20~30::40,5.00,PASSPORT_NO::NID::BIRTH_REG,A1230::123450::98760,BD0010004

Here ‘,’ is field value separator, ’::’ is multi value separator, ‘~’ is sub value separator.

Columns which have only single value will get populated into single value column table (as per SCHEMA-A1); Table 5.2: TABLE-A1.

So it populate data columns of single value as follows:

account_id	category	first_name	int_rate	branch_id
0004121000001	6001	A	5.00	BD0010004

Table 5.2: TABLE-A1

Columns which are non associative multi value will get populated into separate table TABLE-A2 as SCHEMA-A2. In this case number of inserted rows for that record will be number of multi value of that column. So it populate data columns of non associative multi value set as follows:

account_id	email_id
0004121000001	a@yahoo.com
0004121000001	b@yahoo.com

Table 5.3: TABLE-A2

Columns which are associated with only multi value set will get populated into a separate table TABLE-A3 as SCHEMA-A3. So if there is more than one associated multi value set than it will get populated on separate table. On both cases number of inserted rows for that record will be number of multi value of any of the associated column. Note that no of multi value will be equal for each column that are associated with each other.

So it populate data columns of associative multi value set as follows:

account_id	doc_name	doc_id
0004121000001	PASSPORT_NO	A1230
0004121000001	NID	123450
0004121000001	BIRTH_REG	98760

Table 5.4: TABLE-A3

Columns which are associated with both multi value and sub value set will get populated into a two different table TABLE-A4 and TABLE-A5 as per SCHEMA-A4 and SCHEMA-A5. One table to relate multi value data with record id say main_multi_set and another to relate sub value data with record id as well as multi value say main_first_multi_sub_set.

In main_multi_set table, columns which are multi-valued among the association get populated. If there is more than one associated value set like this then it will populated on

same way on separate main_multi_set table. In all cases number of inserted rows for that record will be number of multi value of any of the associated column. Number of multi value will be equal for each multi value column that are associated with each other.

In main_first_multi_sub_set table, columns which are sub value among the association get populated. In addition the first multi value of that association will be also taken to relate this sub value. So if there is more than one associated value set like this then it will get populated on same way on separate main_first_multi_sub_set table. In all cases no of inserted rows for that record will be sum of sub values of each multi value of the associated column. Number of sub value will be equal for each sub value column that are associated with each other.

It populate data columns of associative multi value set as:

account_id	txn_date
0004121000001	20141220
0004121000001	20141221
0004121000001	20141223

Table 5.5: TABLE-A4

account_id	txn_date	txn_type	txn_amt
0004121000001	20141220	CR	10
0004121000001	20141221	CR	20
0004121000001	20141221	DR	30
0004121000001	20141223	DR	40

Table 5.6: TABLE-A5

B. Time efficient transformation

In this technique we concentrate on how much time it will take to process a query after loading into data warehouse. In this technique simply we will create only two tables.

One for all the single value columns. Another one for all the multi-value and sub-value columns whether it is associated or not. Number of table require on this technique is 2.

Considering scenario of Table 5.1, it will create one table for single value columns named as account_main as follows:

```
CREATE TABLE account_main
(
  account_id    varchar2(20),
  category      varchar2(5),
  first_name    varchar2(35),
  int_rate      varchar2(6),
  branch_id     varchar2(20)
)
```

Figure 5.6: Schema-B1

Another one for all the multi value and sub value columns whether it is associated or not as account_multi_sub as follows:

```
CREATE TABLE account_multi_sub
(
  account_id      varchar2(20),
  mv_seq          number(3),
  sv_seq          number(3),
  email_id        varchar2(35),
  txn_date        varchar2(8),
  txn_type        varchar2(20),
  txn_amt         varchar2(20),
  doc_name        varchar2(20),
  doc_id          varchar2(20)
)
```

Figure 5.7: Schema-B2

Now concentrate on how data is going to be parsed and get populate into table in this technique. We will use same data set as used in previous technique.

Columns which have only single value will get populated into account_main table as per SCHEMA-B1.

account_id	category	first_name	int_rate	branch_id
0004121000001	6001	A	5.00	BD0010004

Table 5.7: TABLE-B1

Columns which have multi value and sub value set will get populated into account_multi_sub as per SCHEMA-B2.

account_id	mv_seq	sv_seq	email_id	txn_date	txn_type	txn_amt	doc_name	doc_id
0004121000001	1	1	a@yahoo.com	20141220	CR	10	PASSPORT_NO	A1230
0004121000001	2	1	b@yahoo.com	20141221	CR	20	NID	123450
0004121000001	2	2	b@yahoo.com	20141221	DR	30	NID	123450
0004121000001	3	3		20141223	DR	40	BIRTH_REG	98760

Table 5.8: TABLE-B2

In account_multi_sub table, there will be a row for each multi-value position as well as for each sub-value position under that multi-value position of a particular record.

Consider multi-value set of a record, for each multi-value element there will be a separate row. So in this case actually it will insert maximum number of multi-value for any of the fields on that record. Say if there are 3 multi-valued fields on a value set; first one has 1 multi-value, second has 3 and third has 2. When considering whole value set, it create 3 rows to separately identify multi values and identify relationship between them.

Now consider case where there is sub-valued field among the whole value set (on each multi-value, it may have more than one sub-value). Suppose there are two sub-valued fields on a value set, where first one has 1 multi-value and under it 2 sub-values, it will require 2(maximum number of sub-value for its only multi-value position) rows to be inserted. Second has 2 multi-value where it has 2 and 3 number of sub-values under its first and second multi-value respectively. For first multi-value position maximum number of sub-value is 2 and for second multi-value position is 3. So for first multi-value position of the value-set it will require 2(maximum number of sub-value for first multi-value position) rows

to be inserted and for second multi-value position of the value-set it will require 3(maximum number of sub-value for second multi-value position) rows to be inserted.

Say n is the number of multi-value element of a field among a value set and $S_1, S_2, S_3, \cdots\cdots\cdots\cdots, S_n$ are the maximum number of sub-value for 1st, 2nd, ……, nth multi-value element respectively. So total no of rows inserted for each record will be:

$$X = S_1 + S_2 + S_3 + \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots + S_n = \sum_{i=1}^{n} S_i \qquad (5.2)$$

Chapter 6

6 Implementation

Working principle has been implemented using following:

- Java as programming language
- jdk-7 as development kit
- netbeans-7.4 as java editor
- oracle as RDBMS
- Toad as oracle editor

The flow of implementation is as follows:

- First we install the required software packages
- Then we take the metadata of our sample dataset and create a table in oracle database to store it.
- After that we develop our programs to implement our proposed techniques.
- After successful run of the programs, it creates and populates table as discussed in both the techniques.

We develop the following classes to implement the proposed methods:

- ReadTextFileTest: It is the main class which controls the flow of programs
- ReadMetaData: It parses the metadata of the file for further program actions.
- DbOperation: It controls all the database operations
- ReadTextFile: It implements the principle of space efficient technique
- TimeEffective: It implements the principle of time efficient technique

Chapter 7

7 Performance Evaluation

We measure space and time on both space and time efficient case and evaluate space versus time to evaluate performance.

First we take a look back on how tables created and inserted data in space efficient technique. By applying this method we will get schemas in third normal form; so except foreign key we will not get data duplication. On the other hand in case of time efficient method only two tables will be created. By applying this method we will get schemas in first normal form; so in table there will be huge data duplication can be found.

Suppose there is a subvalue field which have 9 multivalues $m_1, m_2, m_3, \cdots\cdots\cdots m_9$ and multivalues have 2, 4, 6, 8, 10, 12, 14, 16, 18 subvalues respectively. And another subvalue field (not associated) have 9 multivalues $n_1, n_2, n_3, \cdots\cdots\cdots, n_9$ and multivalues have 18, 16, 14, 12, 10, 8, 6, 4, 2 subvalues respectively. Now we will calculate number of rows inserted by each technique.

In space efficient technique, number of inserted rows will be=2+4+6+8+10+12+14+16+18=50*2=100

In time efficient technique, number of inserted rows will be=18*9=162. Also other multi values and sub values data will get duplicated for same multi value and sub value position. So it is clear that first method will take less space than second one.

Now take a look on time efficiency. Space efficient method generates schema in 3NF; number of tables on this method is good in number (depends on dictionary file). So it requires

frequent joining operation to get expected query result. On the contrary time efficient method generates schema in 1NF; number of tables on this method is only 2. So it requires very less joining operation to get expected query result. Number of tables created in space efficient method against a table follows:

No of non associated MV column(A)	No of associated MV column set(B)	No of associated set which contain both MV and SV column set(C)	$X = 1 + A + B + 2C$
1	1	1	5
2	2	2	9
3	3	3	13
4	4	4	17
5	5	5	21

Table 7.1: Simulation of number of table required

As the number of joining operation increase, time require executing query also increase. It summarize that in space efficient technique it save space over time efficient technique; first one is space efficient. But time efficient technique save time over space efficient technique; second one is time efficient.

Sample data of MVDBMS taken and convert it into both techniques, calculate occupied space. Then we perform various queries to get expected result set and calculate elapsed time.

Space and time measurement for both techniques are as follows:

Sample No	Space(MB) X		Time(ms) Y	
	Space Efficient Technique	Time Efficient Technique	Space Efficient Technique	Time Efficient Technique
1	0.43	1.28	162	9
2	2.59	7.74	345	23
3	59.91	180.33	540	227
4	99.45	300.34	2005	655
5	199.59	604.75	3107	904
6	1087.06	3304.66	33060	2137
7	3025.01	9150.51	52903	4007

Table 7.2: Space versus Time table

By plotting values obtained from Table 7.2, we get the following space versus time graph:

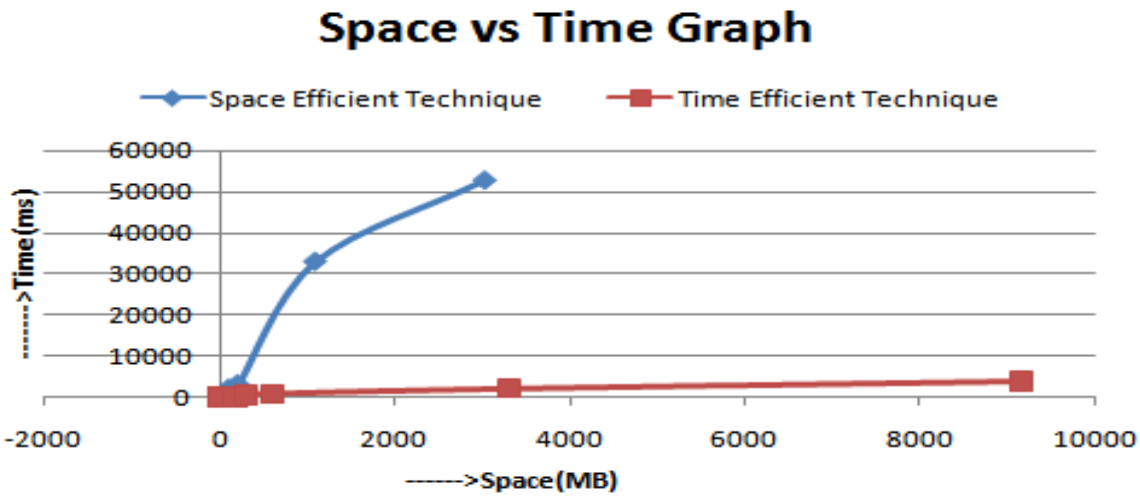

Figure 7.1: Space versus Time graph

Chapter 8

8 Conclusion And Future Work

In this thesis, we have identified that a multi-valued database cannot handle complex queries well. Therefore, we have proposed and then elaborated two approaches to transform multi-value database systems into relational database systems for faster querying and supporting complex SPJ queries. One of the proposed approach transform multi-value source data into 3NF and works well when only limited amount of storage space is available; hence it is called "Space Efficient Technique". While the other approach transform multi-value source data into 1NF and executes queries faster than the former which makes it more suitable to online or streaming data platform; hence it called "Time Efficient Technique".

In future, we would like to extend this work and want to build an intelligent system that will be capable of analyzing the current system and data characteristics to determine the appropriate platform, i.e. either MVDBMS or RDBMS, as well as to transform MVDBMS into RDBMS (if required) using which of these two proposed approaches. We would like to also investigate the possibility of building a hybrid approach which can offer us the goods of both of these proposed approaches. We would also like to develop a common API which can be used on any NoSQL/multi-value database to transform into relational database for its query processing. In addition to that a user interface can be built where user will just select options of query and API will generate query against that selection to make complex data retrieval more easy to person who don't know SQL scripting.

References

[1] Karamjit Kaur, Rinkle Rani: Modeling and querying data in NoSQL databases. In: IEEE International Conference on Big Data, 2013

[2] Xiang Li, Zhiyi Ma, Hongjie Chen: QODM: A query-oriented data modeling approach for NoSQL databases. In: IEEE Workshop on Advanced Research and Technology in Industry Applications (WARTIA), 2014

[3] Anderson Chaves Carniel, Aried de Aguiar Sá, Vinícius Henrique Porto Brisighello, Marcela Xavier Ribeiro, Renato Bueno, Ricardo Rodrigues Ciferri , Cristina Dutra de Aguiar Ciferri: Query processing over data warehouse using relational databases and NoSQL. In: XXXVIII Conferencia Latinoamericana En Informatica (CLEI), 2012

[4] Kudakwashe Zvarevashe, Tatenda Trust Gotora: A Random Walk through the Dark Side of NoSQL Databases in Big Data Analytics. In: International Journal of Science and Research (IJSR), ISSN (Online): 2319-7064, Volume 3 Issue 6, June 2014

[5] Mohamed A. Mohamed, Obay G. Altrafi, Mohammed O. Ismail: Relational vs. NoSQL Databases: A Survey. In: International Journal of Computer and Information Technology, (ISSN: 2279 – 0764) , Volume 03 – Issue 03, May 2014

[6] Biswajeet Sethi, Samaresh Mishra, Prasant ku. Patnaik: A Study of NoSQL Database. In: International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181, Vol. 3 Issue 4, April - 2014

[7] Christoph Bussler: SQL for NoSQL Databases: Déjà Vu. In: 7th Biennial Conference on Innovative Data Systems Research (CIDR'15), Asilomar, California, USA, January 4-7, 2015

[8] Kavita Ozarkar, Rakesh Rajani: Optimization Technique for Efficient Dynamic Query Forms with NoSQL. In: International Journal of Science and Research (IJSR), Volume 3 Issue 11, November 2014

[9] Haleemunnisa Fatima, Kumud Wasnik: Comparison of Sql, Nosql and New Sql Databases in Light of Internet Of Things – A Survey. In: Proceedings of 48th IRF International Conference, Pune, India, ISBN: 978-93-85973-21-5, 31st January 2016

[10] Ricardo Vilaca, Francisco Cruz, Jose Pereira, Rui Oliveira: An effective scalable SQL engine for NoSQL databases. In: 13th IFIP WG 6.1 International Conference, DAIS 2013, Held as Part of the 8th International Federated Conference on Distributed Computing Techniques, DisCoTec 2013, Florence, Italy, pp 155-168, June 3-5, 2013

[11] Ramon Lawrence: Integration and Virtualization of Relational SQL and NoSQL Systems Including MySQL and MongoDB. In: International Conference on Computational Science and Computational Intelligence (CSCI), 10-13 March 2014

[12] http://www.jbase.com/products/jbase-multivalue-database/ (accessed on 19/09/16)

[13] Y. Zhao, K. Ramasamy, K. Tufte, J.F. Naughton: Array-based evaluation of multi-dimensional queries in object-relational database systems. In:14th International Conference on Data Engineering, pp- 241-249, 1998

[14] R. Kimball: Data Warehousing Toolkit. In: John Wiley & Sons, 1995

[15] M. Gyssens, L. V.S. Lakshmanan: A Foundation for Multi-Dimensional Databases. In: VLDB 1997, pp.106 -115, 1997

[16] R. Agrawal, A. Gupta, S. Sarawagi: Modeling Multidimensional Databases. In: ICDE 97, pp.232 -243, 1997

[17] Suchitra Reyya, T. Summallika, G.V.M. Vasuki: An approach to store spatial big-data using multi-valued database. In: International Journal of Research in Engineering & Technology (IMPACT: IJRET), ISSN(E): 2321-8843, ISSN(P): 2347-4599, Vol. 1, Issue 6, Nov 2013