

A Relational data model proposal for small firms in the retail sector of Bangladesh

```
SELECT Countries.Name AS "Country",  
States.Abbreviation AS "State",  
Teams.LeagueID AS "LeagueID" FROM  
(Teams INNER JOIN Cities ON Teams.  
INNER JOIN Countries ON Cities.  
INNER JOIN States ON Cities.  
WHERE Teams.ID = '$tid'  
ORDER BY States.Name ASC
```



Project report on:

“A Relational data model and database proposal for small firms
in the retail sector of Bangladesh”

Submitted To

Ahmed Imran Kabir

Lecturer

School of Business & Economics

United International University

Submitted By

Mohammad Mehedi Hossain Suvo

ID: 111 163 031

Department: BBA (Major in MIS)

School of Business & Economics

United International University

Date of Submission

5th June 2022

Letter of Transmittal

5th June, 2022

Ahmed Imran Kabir

Lecturer

School of Business and Economics

United International University

Subject: **Project report on “A Relational data model and database proposal for small firms in the retail sector of Bangladesh”.**

Dear sir,

As the final part of my undergraduate program of BBA, it is with my positive enthusiasm and outmost dedication to present you my research project “A Relational data model and database proposal for small firms in the retail sector of Bangladesh”. I had a truly educational and wonderful experience working on this project as it helped me to learn more about the small retail firms and the challenges, they face on converting their business from the traditional pen and paper management to managing fully on a digitalized database. My main focus on this work was to create and provide a data model that is not too complex, but enough reliable for the small firms to implement on their business properly and thrive on their business transactions as well as easy management of their administration.

Thank you for letting me choose this topic and work in depth on this topic. I did my best to experiment and create fulfilling research possible. I hope you find paper fulfilling.

Sincerely yours

Mohammad Mehedi Hossain Suvo

ID:111 163 031

Declaration of the student

I, Mohammad Mehedi Hossain Suvo declare that this project report titled “A Relational data model and database proposal for small firms in the retail sector of Bangladesh” was solely and uniquely made by me under the supervision of Ahmed Imran Kabir, Assistant lecturer of United International University, as a requirement for completion of my BBA program in United International University.

Additionally, I declare that this project report has not been used or submitted elsewhere. This was only for my academic purpose in this University and not for any other purpose.

Thank you.

Mohammad Mehedi Hossain Suvo

ID: 111 163 031

Submitted to:

Ahmed Imran Kabir

Assistant Lecturer

School of Business and Economics

United International University

Acknowledgement

I would like to express my gratitude to all those individuals who have provided me with immense support, motivation and confidence for completing the project I have been working on. I would firstly like to thank Almighty Allah for providing me the strength to complete the project in a complex free manner. I would also like to thank those friends and family members who have helped me on collecting crucial and analytical information that played a vital role in completion of the project I have worked on.

I would also like to thank my supervisor and honorary faculty of school of business administration, Ahmed Imran Kabir sir for providing me his value time and guidance throughout the entire process of the project.

Abstract

This project paper attempts on analyzing through market research and digital trends to create a data model and corresponding database for small firms in the retail sector of Bangladesh. In order to create a sustainable data model alongside its corresponding database, a relational data model was used in order to create the acceptable and sustainable database which in return will serve the small firms that cannot purchase and maintain an expensive and complex database. The key tables that were used to create a relational database are “city”, “address”, “staff”, “store”, “customer”, “payment”, “products”, “product_category”, “inventory”, and finally a supporting table named “category” to support the “product_category” table. The data model was created in way which will support expandability according a firm’s choice. Each of table consists of supporting attributes and one or multiple primary attribute keys are used to connect the relational data model. The software used to create data model and deploy the model to evolve into a database system is “MYSQL” software. “MYSQL Workbench” can the primary workbench used to retrieve, update, remove and/or enter data in the deployed database to sustain and/or manipulate the database. “MYSQL” uses a specific type of coding language, also called as ‘query’, that can be used to manipulate data according to the individual in charge of the database.

The relational database created will have a forward engineering codebase as well as the relational model which can be used by any organization, big or small. With the use of MySQL package (workbench, server, shell) the organization can create a database and data server of their business and use the workbench to alter the tables, create, update and delete data according the organization’s preference.

Table Of Contents

Letter of Transmittal	3
Declaration of the student	4
Acknowledgement.....	5
Abstract.....	6
Chapter 1	10
1.1 Introduction:	11
1.2 Problem Statement:.....	12
Chapter 2	13
2.1 Literature review.....	14
2.2 Database:.....	14
2.2.1 Data and Information:.....	14
2.2.2 What is a database:.....	15
2.2.2 What elements should a database consider:.....	15
2.2.2 What is an integrity constraint:	16
2.3 Database Management:	16
2.4 Database Security:	17
2.5 Integrity Control:	18
2.6 Enterprise Resource Planning.....	21
2.6 ER-Diagram	23
2.7 Relational Database Diagram	24

2.8 SQL Query and SQL vs No SQL	25
2.8.1 SQL queries and its implementations	25
2.8.2 SQL vs No SQL	26
Chapter 3	29
3.1 Introduction	30
3.2 Research Setting	30
3.3 methodological approach	32
3.3.1 Reason of choosing the tables and attributes	32
3.3.1.1 Customer related data	33
3.3.1.2 Inventory related data	35
3.3.1.3 The business aspects	37
3.3.2 The final relational data model	38
3.3.3 Database Deployment and forward engineering of the data model in the server .	40
3.3.4 Populating the database	47
Here the snaps/pictures taken of every table after populating with data entry on MySQL Workbench:	47
Address:	47
Chapter 4	53
4.1 Introduction	54
4.2 Query codes and pictures of the resulting outcome	54
Chapter 5	64
5.1 Introduction	65
5.2 Discussions of the outcome	65
5.3 Challenges faced	69
Chapter 6	70

6.1 Conclusions.....	71
6.2 Recommendations	71
References:	73

Chapter 1

- Introduction
- Problem statement

1.1 Introduction:

In the age of digitalization where all business sectors have altered their business model towards digital market rather than traditional brick and mortar market, managing the businesses and conducting business transactions has become much more convenient, secure and safe because of the emerging of e-commerce culture. This shift in digital market specially has profited many small firms inside a low-income country like Bangladesh where not every business can afford a physical retail shop to display their catalogues of products and/or services. These businesses however, can afford either a good website of their company and the associated products, advertise their products/services on social media platforms such as Facebook, Instagram, Tweeter or do both at the same time which is usually a much lower expense way of maintaining the sales and administration side of the business.

However, despite of having an existence in the digital market place, most of the small firms still conduct their administration side of the business in traditional pen and paper method. In Bangladesh, large super shops like Unimart, Meena bazaar, Agora etc. uses a dedicated database where they control both financial, sales and administration aspects of the business. From the attendance of the employees to information about customer transaction and even purchasing pattern of their membership customer club, these organizations store all the data in a dedicated database. These databases are created through relational data model freelanced to database designers and usually are very complex and huge.

This project aims to create and describe the advantages of using an open-source database. The main ones are being an easy-to-use database that is highly adjustable according to businesses' goal and demand and provide reliable outputs.

Moreover, there are also many advantages in using a relational database in general. These are speed, simplicity, security, accessibility, accuracy and multi user friendliness.

However, there are many cons of using an integrated database for the business. The main ones are cost, performance, physical storage, complexity, information loss, structure limitations.

In addition, the initial database primarily created and deployed was designed for a moderate sized superstore. The data model was created using MYSQL workbench which was also used to deploy the database. As it is an experimental research, some hypothetical data entries were created in order to demonstrate the MYSQL query system for a better conceptual view of how the entire process works.

1.2 Problem Statement:

This experimental research topic aims to address the scenario that what will be the biggest gain if an open-source relational data model and its significant database, briefly known as Relational Database Management System, can be developed and freelanced to small firms which can use it to achieve much more productivity and efficiency on running the business. There are some main advantages that drives a RDBMS (Relational Database Management System) being a game changer for not only the big firms but also for the small firms.

The advantages that might contribute firms using a RDBMS are “simplicity”, “ease of data retrieval”, “data integrity”, “flexibility” and “normalization”.

(<https://www.techwalla.com/articles/what-are-the-advantages-of-a-relational-database-model>) Furthermore, the properties that contribute to the uniqueness of RDBMS from a traditional pen and paper management system are “atomicity”, “consistency”, “isolation”, and “durability”. Ultimately, the following research question was developed:

Q: “What are the properties that contributes a Relational Database Model being advantageous than traditional pen and paper management system?”

Chapter 2

- **Literature review**

2.1 Literature review

The research study will now be presented from preexisting and available sources on this issue in this section of the paper; it would provide understanding of the current and ongoing research area on this specific subject. The literature evaluation will also assist us in identifying the information for the study that we generated previously, including the flaws in the experimental protocols and theoretical inconsistencies.

2.2 Database:

2.2.1 Data and Information:

Data constitute unstructured facts that make up the foundation of information. The database management system's heart is data. It is important to realize that not all of the information will be relevant. Data that has been processed yields useful information. To put it another way, data must be interpreted to gain information. The trick to creating good decisions is having good, timely, and necessary details. Organization needs depends on good prediction. Data are really a codified representation of information, concepts, or guidelines that may be communicated, interpreted, or processed by humans and machines. The information in a database system can be categorized into two kinds: the source of data needs of the organization and "metadata," which would be information about everything from the system. The most durable component of an organization's information system is data. Employees, offices, and paychecks must all be saved by a corporation. Data is the term for these bits of information. Persistent data is information stored indefinitely. Generally, we operate on number of data components in order to provide detailed information. A bookstore, for instance, keeps track of its membership, publications, deadlines, and fines (Sumathi & Esakkirajan, 2007).

2.2.2 What is a database:

Databases are very well gathering of data and is meaningfully related and can be retrieved in many logical sequences. Database systems are one where the processing and management of information have always been the most important features. As a result, database management systems are accompanied by strong quantities of data, the necessity for lengthy data storage, and control of the data by a wide range of users (Sumathi & Esakkirajan, 2007).

Denning et al. (1987) said that, a database is a collection of specified connections inside the relational paradigm. A relation's function can be analyzed directly as stored information or intellectually as a deriving rule that specifies how the information is generated. Malik and Patel (2016) said that, the information contained in databases is frequently arranged to facilitate procedures that require content search and processing. The database is where the majority of the data is kept.

2.2.2 What elements should a database consider:

Reiter (1992) said that, the standard view of analytic datasets is that they should be collections of first-order propositions. As being such, they might be thought of as assertions about the facts of some other universe, with the database serving as a symbolism of that reality. Almost all techniques to database proposal evaluation inside this rational database architecture handle questions from first algorithms, frequently with free factors whose bindings coming from the identification mode determine the query's responses. The other being Levesque's work, which advocates for a generic query language based on experiential analogical reasoning. In other words, questionnaires

should be able to address both factors in the external domain as reflected inside the database and elements of the databases itself, i.e., what the data understands.

2.2.2 What is an integrity constraint:

Reiter (1992) also said that, integrity constraints can be found in databases as well as Artificial representational languages. The core principle is that some database configurations are permissible and even an integrity constraint is being used to ensure that these legitimate states are followed. There are two types for integrity constraints: static and dynamic. A static constraint's applicability is only reliant upon its database's present state, not any of its previous states. A static restriction is the need that every worker has such a security number. Dynamic constraints are based on the state of two or even more databases. If, for instances, employee pay must never drop, an employee's compensation in a subsequent database state cannot be lower than it is now.

2.3 Database Management:

Sumathi and Esakkirajan (2007) also said that, a database management system is made up of a collection of linked data from a number of tools that allow you to access it. It is the database management software that helps you keep track of and use your data.

One of several main key benefits of the data structure and powerful query language out from 1970s was the idea of employing a DBMS to lighten the stress of data administration. A developer just needs to write a search query that indicates what data

they wish to retrieve when using this method. The DBMS then determines the most effective way of storing and retrieving data, as well as the safe interleaving of processes (Pavlo et al., 2017).

Jajodia and Meadows (1995) said that, database management is intended to make information search and processing more efficient. Because of their potency, when they're not correctly constructed to prevent illicit inferences, they would not only fail to stop them, but will considerably aid users in making them. However, inference difficulties in multidimensional databases have received little attention so far. This really is partly owing towards the problem's complexity and it's also apparent in the way that no way of attempting to control interpretations can be implemented until a fundamental issue of identifying how and where to retrieve relevant multidimensional data has been solved. (De Witt, 1979) said that, alternate programming architectures for non-numerical applications must be explored because datasets are constantly developing at a faster level than processor performance. Malik and Patel (2016) said that, a database management system (DBMS) is a software program that captures and analyzes data by interacting with the authenticated user, additional applications and the database infrastructure. Keeping indexes helps to arrange information for accurate processing and retrieval.

2.4 Database Security:

Murray (2010) said that, database security generally is based on a three-concept structure: confidentiality, integrity, and availability. Confidentiality or privacy makes reference to the encryption of information from security breaches, integrity means the protection of unapproved and inappropriate data modification and availability makes reference to the detection and correction from hardware and system errors along with suspicious data access that results in data defiance. Only authorized individuals do

permitted operations at authorized periods, according to database security. It refers to the system, policies, and procedures in place to keep a database safe from unauthorized access. Lunt and Fernandez (1990) said that, the capability of a computing device to implement firewall rules regulating the exposure, alteration or destruction of data is known as computer security. The security policy might be tailored to the company or it can be general. Basharat, Azam, and Muzaffar (2012) said that, the security measures include information security and privacy. Confidentiality, integrity, and availability are said to be the three basic forms of database security. Confidentiality places restrictions on retrieving safe data, preventing unauthorized access to the information. The term "integrity" refers to the fact that the information will not be contaminated in any way. Secure databases have the characteristic of reliable data availability. Malik and Patel (2016) said that, database permissions can be misused in a variety of ways. A user might misuse his or her privilege for nefarious purposes. Extensive privilege abuse, justifiable privilege abuse, and underutilized privilege abuse are all types of privilege abuse. This is the most hazardous sort of attack since authorized personnel are misusing data. These rights and protections can be exploited, putting people in danger.

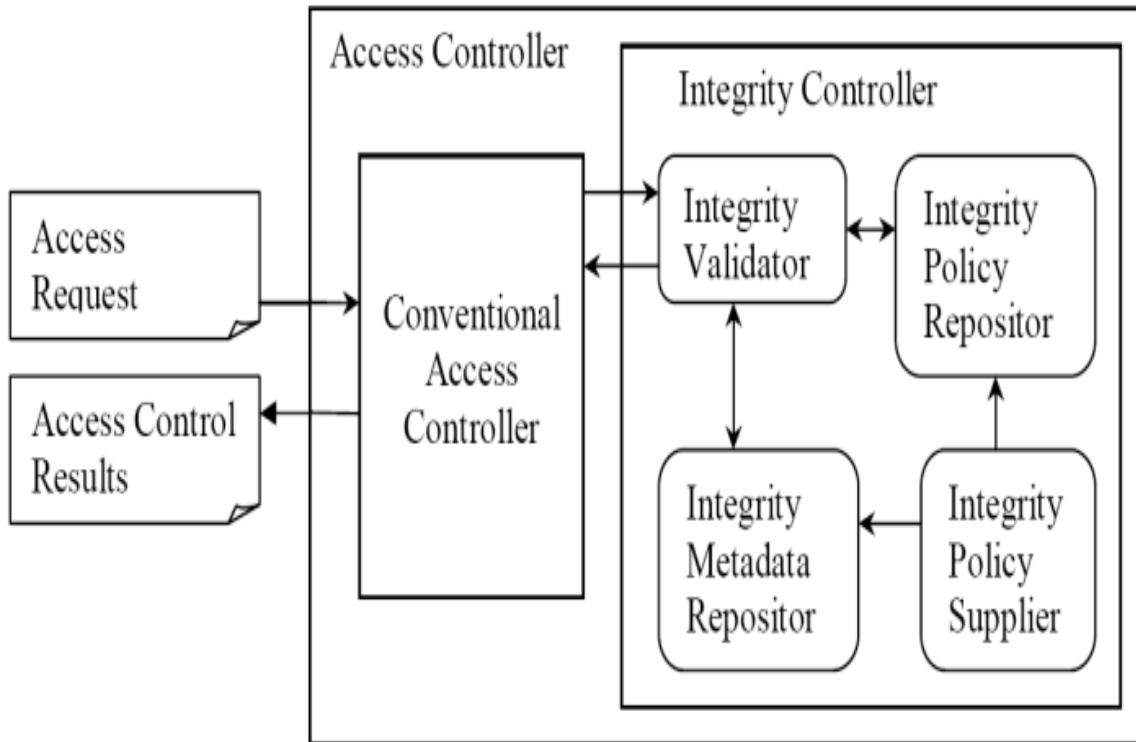
2.5 Integrity Control:

Grefen and Apers (1993) said that, the word integrity increases the reliability or authenticity of the information in the system as defined expressly by normalized as well as referential integrity. This means that neither fullness of both the database server nor the fullness of the representation of the real domain represented by the system is guaranteed generally. The completeness or accuracy of data is crucial in a DBMS. There are several ways that inaccurate data can appear in a database. The approach is specific in a straightforward top-down, declarative manner, and is built on the logical notions. The approach primarily is transaction-oriented, as that name suggests, and

provides the required functionality for execution.

Hong and Su (1981) said that, a platform's integrity could be harmed from three ways: (1) by insufficient scheduling, which might lead to a transaction interferences like "lost update" problem; (2) by failing to recognize semantically inaccurate information, which can happen as a consequence of corresponding query or system's incautious, improper, or harmful update; as well as (3) through failure to refresh other documents as one fraction of such dataset is changed. DBMS integrity control have historically relied on clear application domains with comprehensive information about the nature of the actual world. Data integrity throughout this context can be specified using integrity constraints, that also are first-order logical statements that describe database variables or chains of database states that are recognized to correspond into probable application domain event.

Bertino, Dai, Lim, and Lin (2008) said that, we developed a collection of applicable integrity requirements as well as a thorough structure building integrity control schemes. Data integrity has diverse meanings for different persons and applications. Coming up on the result, the first step in building any integrity management solution is to pinpoint the most critical criteria. There are various integrity models which have been investigated. These are Information flow control, data verification, prevention of fraud and errors and autonomous data integrity validation. The figure below shows an architecture for integrity management system as proposed by the author.



2.6 Enterprise Resource Planning

Moon (2007) said that, an Enterprise Resource Planning (ERP) system is an essential information system that integrates and optimizes a participate in business systems and processes. ERP is indeed a set of industry-driven strategies and systems that has gained widespread acceptance as a feasible option for achieving integrated business information systems. The academic and research program has taken numerous works in the area. Providing historic journal articles for social purpose is a common approach to the contribution to the subject.

The advantages of a well-chosen and executed ERP system can indeed be substantial. Inventory expenses can be decreased between 25 to 30 percent on average, and cost of raw materials by roughly 15 percent. Customers' lead times, production times, and cost of production can all be decreased. ERP implementations frequently fail, despite great expectations and large investments. "The New York Times" published an article titled "Software That Can Make a Grown Company Cry," which discussed ERP deployment issues. Allied Waste Businesses bought the ERP software program through SAP America, a prominent ERP software provider, but stopped using because it was just too difficult, too expensive, as well as lacked extra capability. According to many studies, the value an organization obtains from employing information technology (IT) is based just on organization's features. As a result, not all businesses will profit equally from utilizing having similar ERP software, and distinct Operating software packages might suit different businesses better. Other factors (like business deployment and customer integration) also influence ERP's performance (Arik Ragowsky, 2002).

Shehab, Sharp, Supramaniam, and Spedding (2004) said that, ERP is already widely seen as the cost of doing business operations and, at least from the perspective being, for connecting to other businesses in a digital industry, to enable "business to business" electronic transfers. Moreover, many multinational corporations limit their engagement to businesses using the same ERP software like the international corporation. ERP is

indeed a reality, and smaller businesses must modify their business models and approaches to match the processes and software used by larger businesses. Small and medium-sized businesses (SMEs) have had a difficult time adjusting to the expanding economy. Because they lack the financial stability of major corporations, SMEs must rely on IT and an integrated and comprehensive system to remain competitive and customer-focused.

An ERP solution is a business-wide synchronization mechanism for all organizational information systems. This integration allows businesses to respond quickly to competitive forces and market possibilities, provide more variable product combinations, minimize inventories and secure supply chain links.

Al-Mashari, Al-Mudimigh, and Zairi (2003) said that, there are two types of ERP advantages: tangible and intangible. Stock reduction, staff reduction, enhanced output, improved order planning, more accelerated closing of economic shocks, reduced IT and material cost, improved cash flows, enhanced sales and profits, reduced freight and distribution costs, lower operational maintenance, and enhanced on-time performance measurement on delivery are examples of tangible benefits. Increased awareness of corporate data, newer and better business processes, enhanced customer responsiveness, unexpected cost savings, stricter incorporation among systems, simplification of desktop applications, increased flexibility, worldwide information exchange, Y2K compliance, enhanced business effectiveness, and improved accessibility more into Supply chain management are examples of intangibles.

2.6 ER-Diagram

Btoush and Hammad (2015) said that, during database design, especially relational database, drawing an ER diagram is a crucial step. Tech's ER Studio, Microsoft's Visio, and Dia have all produced commercial programs for representing ER

models. Many studies likewise focus on building and implementing tools that build ER diagrams using various techniques. ER diagrams reflect the Entity relationship diagram that illustrate how data would be displayed and structured in the different components.

Thomas, Waugh, and Smith (2006) said that, we have divided our approach to interpreting diagrams into five stages: segmentation, assimilation, identification, aggregation, and integration. The very first two phases convert a raster to something like a collection of graphical representation components like boxes, lines, plus text. A drafting software's output, similar to the ones shown below for creating ER-diagrams, is essentially a collection of graphical representation primitive entities, one with its own number of characteristics, including unique coordinates. The employment of such a tool eliminates any inaccuracies, making the initial stages obsolete.

The accurate building of a relational architecture is the most important problem that prospective IT professionals must learn to tackle. To accomplish so, prospective intelligent developers must research the database's topic area in depth and create client needs. The end findings of this research should really be a dataset that is easy to navigate and correctly represents the dilemma, with no data redundancy. ER charts,

which are centered on the ideas of "Entities" and "Correlation," are being used to help in database architecture. This ER diagram is an illustration of a dataset that defines a specific topic area (Vakaliuk, Korotun, & Semerikov, 2020).

Hayes, Das, Odom, and Natarajan (2017) said that, from such a database standpoint, we suggest a way for determining patterns. We suggest in using Entity Relationship diagrams as the visualization techniques for a user to designate modes. The main idea would be that the patterns define how well the search gets carried out in the area of hypotheses. It can all be thought of as identifying the bits of something like the relational graphs that seem to be appropriate for the target idea from such a relational standpoint.

2.7 Relational Database Diagram

To discover the link among tables, any database must assess both primary and foreign keys. Following the determination of the connection between both the tables, an Entity Relational Diagram will be created. Each entity includes a set of key traits (primary and foreign keys) that might be used to differentiate 1 line from the others and characterize the entity's features. A relationship is the link that connects one entity to one another which would be represented in the ER diagram. Its relation cardinality can be used in connection to identify whether such a table contains a foreign key characteristic. It is possible to assess the connection between two entities by recognizing the status of both primary and foreign keys (Lubis & Zamzami, 2020).

2.8 SQL Query and SQL vs No SQL

2.8.1 SQL queries and its implementations

Egenhofer (1994) said that, spatial databases, that integrate traditional and spatially connected data, have recently received a lot of attention. Several distinct application sectors of spatial datasets, like Geographic Information Systems (GIS), picture databases, and remote sensing, have revealed necessity spatial programming (query) languages. The lack of proper language support to geographic data has hampered the use of common query languages to spatial data management. Instead of creating a different query language from its ground up, geographical ideas are added to the current mysql database language. SQL was adopted as something like the language specification, as well as final spatial query language. The graphical representation language, a sophisticated tool for describing graphical displays, rounds out the set. Spatial SQL's purpose provides greater layers for spatial data inside a query language through including concepts that are closer to how humans perceive about space.

The ability to stack query blocks to every degree has been SQL's most intriguing features among all the other features. SQL's performance is significantly limited without this functionality. However, present strategies for implementing this feature remain expensive in general and given the prevalence of SQL-like database languages, it is critical to create efficient methods for running nested queries. This study demonstrates that a query stacked to any depth is divided into five main types of layering and that out of the five basic kinds of nesting, four kinds (single row subquery, multiple row subquery, multiple column subqueries and correlated subqueries) are poorly understood, which accounts for the poor performance of layered, also known as

nested inquiries in existing systems. It creates algorithms that convert nested inquiries into similar, non-nested requests that query processing subsystems (also known as optimizers) can process more quickly. The techniques are based on different ways of understanding query operations that comprise the four forms for nesting which could often enhance the speed of nested searches by higher magnitudes (Kim, 1982).

For many types of users, such as advanced researchers, database administrators, as well as SQL developers, structured query languages are important and powerful solutions. However, because users must be adept in creating query languages and have a strong understanding of the framework, it is difficult and time-consuming for untrained users to submit structured queries which suit their query intention. When using SQL, however, individuals may run into comprehension issues, formulation issues, and ambiguous error signals. If they're lucky, they can get desired results by consulting instructions and frequently trying alternative SQL queries. The MySQL community has found a solution to the problem will be able to add keyword search within relational databases in terms of reducing the strain of posing queries. Users can utilize this search methodology to create keyword queries without needing to understand MYSQL database or Relational database syntax (Fan, Li, & Zhou, 2011).

2.8.2 SQL vs No SQL

The relational paradigm has been used in traditional database warehouses. These are sometimes referred to as SQL databases where query language is used to access them. Non-relational databases, on the other side of the coin, has exploded in popularity over the years. Such databases are usually referred to as NoSQL databases, which distinguishes them from standard SQL databases. NoSQL databases including

BigTable as well as HBase, are available in both commercial and fully accessible versions (Li & Manoharan, 2013).

Patil, Hanni, Tejeshwar, and Patil (2017) said that, non-relational databases are divided into four categories:

- *Key value: the idea is to utilize a hash table that assign an unique identifier and link to every data fragments with which data can be searched while using key.

- *column-oriented: this database type was created with the goal of processing big data across several servers. A patchy, shared, multifunctional sorted map has been used to store information. Each entry can have a varying number of rows and columns; keys remain, although they point across numerous columns.

- *graph-stored: to describe data, this database architecture is characterized like a system of relationships between such pieces of graph architecture including such nodes as well as edges.

- *Document-oriented: unlike relational database systems, this type of non-relational database store data as docs rather than columns and rows. They create an appropriate framework for modeling data that strongly matches object-oriented design. In object-oriented design, every item is considered an entity. Within document databases, the idea of schema is adaptable: each document may include various fields.

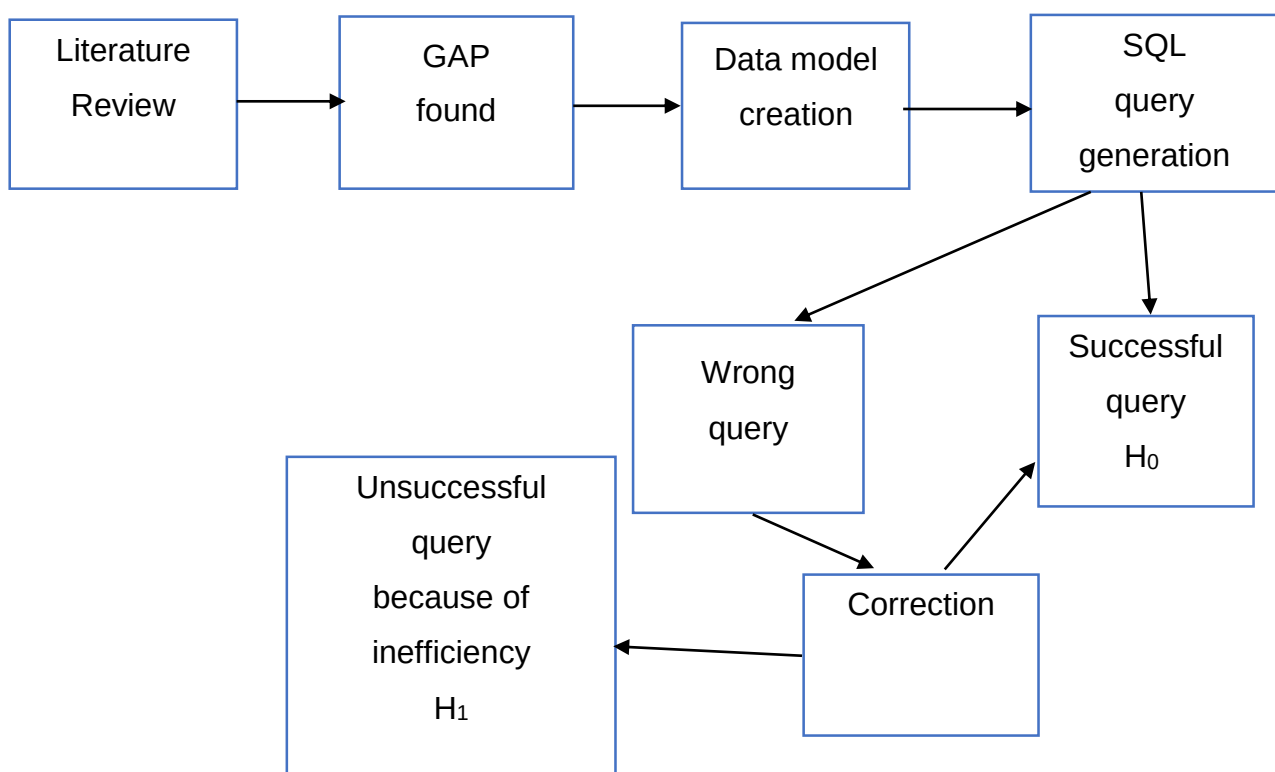
From the following literature review, there was a gap found which was that there was no prominent journal found that elaborates and discusses about the usage of a relational database approach in order to create efficiency which is also an open-source SQL relational database whose data model was created on MySQL workbench.

After the literature review, the researcher has come up with two hypotheses. These are:

H₀ - Using a relational database approach creates efficiency.

H₁ - using a relational database approach doesn't create efficiency.

The following framework was created after concluding the hypothesis which is shown in the next page. The framework denotes the creation of the “Hutbazaar” relational model creation which can be used by any organization as an open-source initial relational model which can be independently modified according the organization's need.



Chapter 3

- Research Methodology

3.1 Introduction

In this part of the paper, the methodology of the research has been discussed. It describes the research strategy, data collection methods and data analysis procedures. Through this section of the paper, we will come to know the variables, the data model design approach, the data keys used as well as the usage of SQL workbench as the platform with which the database can be organized, controlled and stored consequently.

3.2 Research Setting

I have conducted the research through using a hypothetical data. A relational data model of the company “Hutbazaar” which is both a physical super shop as well as online e-commerce super shop that provides groceries to the customers. The customers can either buy online as well as through visiting physically to the outlet. The relational data model consists of three major segments which was focused on creating the relational model. These three segments were: “Customer Data”, “Inventory” and “The business Aspect”. The platform of choice was MySQL workbench where the relational model as well as database deployment and Relational Database Management System (RDMS) was initialized. This brings to an important question given below:

“Why will an organization implement database and why my superstore uses the database approach?”

To answer this question, the advantages of using a database approach is explained on the following paragraph.

Database approach aids in much more efficient way of running businesses which deal with inventory, customer and the supplier constantly. These are the following advantages of using a database approach:

1. Control of data redundancy
2. Data consistency
3. More information from the same amount of data
4. Sharing of data
5. Improved data integrity
6. Improved security
7. Enforcement of standards
8. Economy of scale
9. Balance conflicting requirements
10. Improved data accessibility and responsiveness
11. Increased productivity
12. Improved maintenance through data independence
13. Increased concurrency
14. Improved backup and recovery services

The relational data model that I have designed for my super shop “Hutbazaar” is an open-source SQL relational model which means any organization with the willingness of converting its business from pen and paper management to Relational Database Management System can use the relational data model as well as modify to their business needs which I built for free.

This describes the following research question that has risen from the paper:

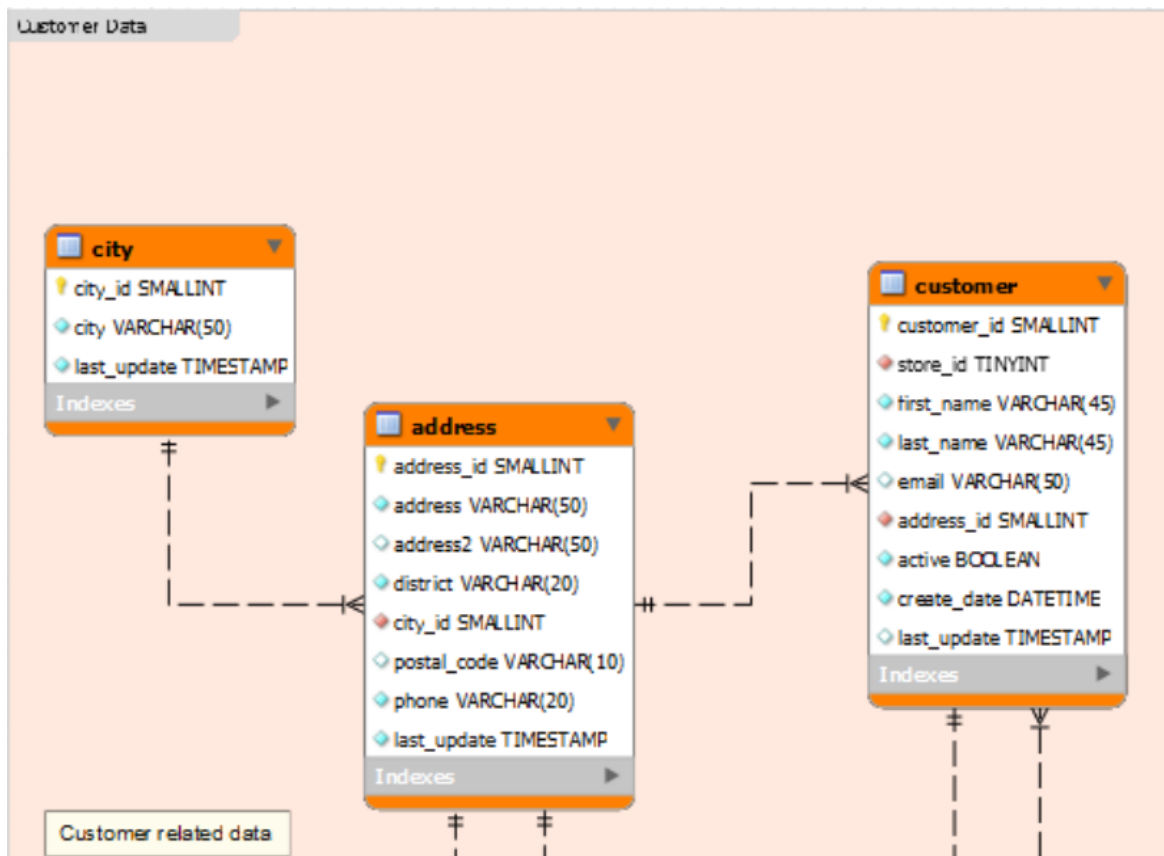
Q. Is a relational database efficient for managing the big dataset of a superstore?

3.3 methodological approach

3.3.1 Reason of choosing the tables and attributes

The relational model of my organization which is known as “Hutbazaar” is segmented into three aspects which has been discussed previously. These are “customer data”, “inventory” and the business aspect. I will discuss each and every segment thoroughly discussing and enlightening about the reason of choosing the proper datatypes or domains.

3.3.1.1 Customer related data



Customer data section consists of three tables named “city”, “address” and “customer”. These tables consist of multiples attributes. The “city” table consists of city id, city name and last update where the city id is the primary key. The city id has a domain of small integer because any attribute which denotes itself as id will subsequently has to have numerical whole numbers that too from 0 to 1 which is why small integer is declared. City name domain is VARCHAR(50) which means upto 50 alphabet, upper case letters, lower case letters, signs and numbers can be used to enter and entry of a city name.

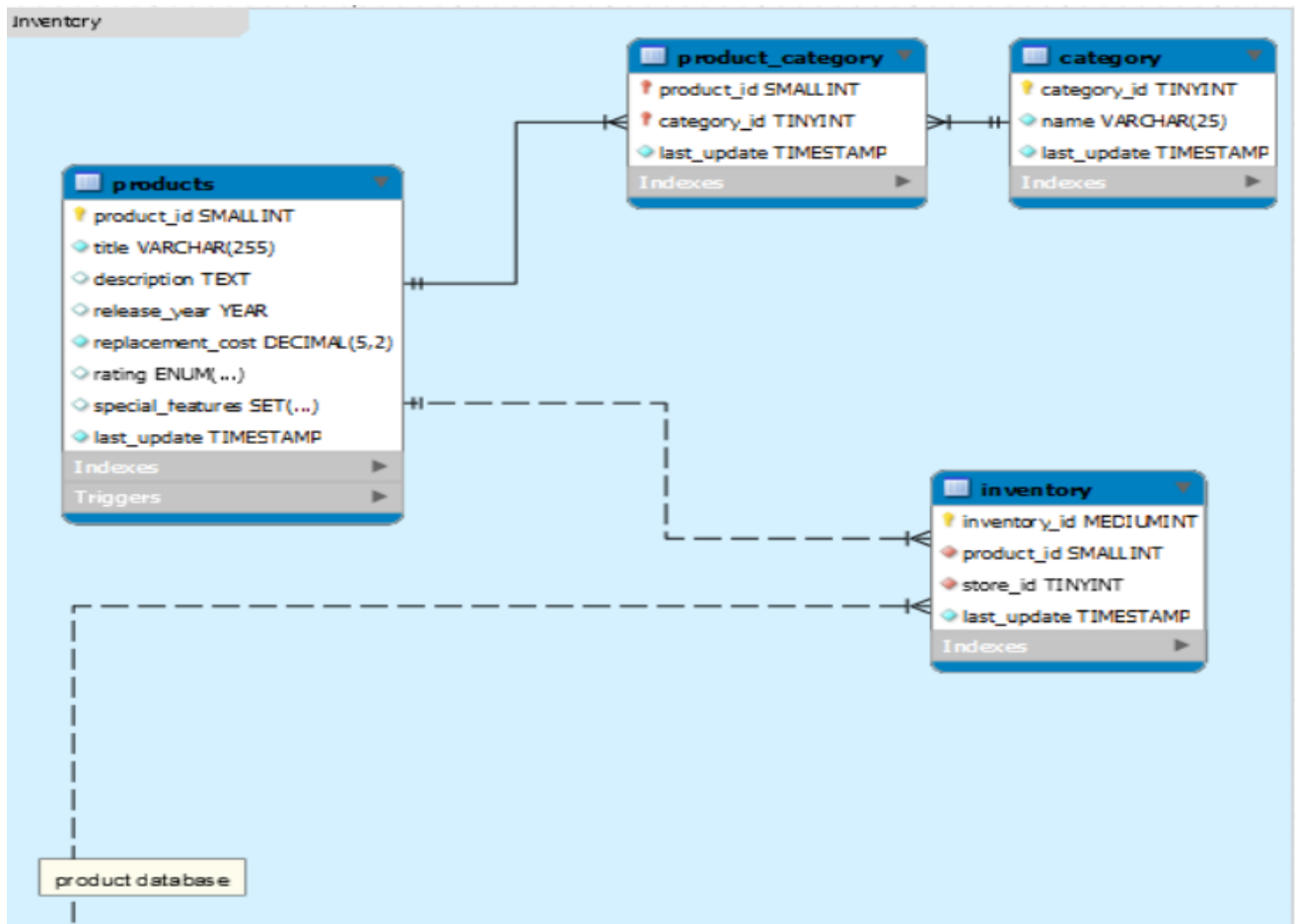
Last update attribute is TIMESTMAP because the entry will be denoted as a time period consisting of timing and dates.

Similarly, address table consists of the following attributes given in the diagram. Note that both postal code and the telephone attributes has a VARCHAR domain given because the telephone no. and the postal code no. can be recorded with a “-” or hyphen sign.

Finally, the customer table has the given attributes as shown in the diagram. There are some new attributes as well as domains. Email, active and create date are new attributes that are new attributes. The BOOLEAN domain denoted in the active schema means that the attribute that will entered in the schema will only be declared as exactly two elements like “false” or “true” and 0 and 1. Active schema will only declare whether a particular customer is active or not. Create date has a DATETIME domain meaning that the entry will be only be valid if it resembles the day/month/year date format.

The city table is connected with the address table in a 1:n or one to many database relationship. This means that one city can have multiple addresses but one address cannot have multiple cities. Similarly, the address table is connected with the customer table in a 1:n fashion meaning that there can be many customers of similar addresses but one address cannot consist many customers. The unique keys are denoted as the “key” sign as shown in the diagram. The diamond shapes which are colorful means that these attributes cannot be left blank or null. They have to have an entry apart from the colorless diamonds which denotes that those attributes can be left blank or null.

3.3.1.2 Inventory related data



Inventory related data consists of four tables named “products”, “product_category”, “category” and “inventory”. This paragraph will steadily describe each of the table alongside the attributes, domains and relationships used.

The “products” table as given contains the above-mentioned attributes. Among all the attributes, **replacement_cost**, **rating** and **special_features** are some new attributes. The attributes describes means what it exactly spells like. However the domains are to be discussed. The TEXT domain in the description schema means on that specific schema, there can be a descriptive textual attribute. The YEAR domain denotes a year attribute. The DECIMAL(5,2) domain means there can be attributes of numerical value upto 5 digits and 2 decimal place. ENUM domain in the rating schema means there are

some predefined values given and the user has to set a value from the given values. The null and not null values are defined by the colorful and colorless diamonds as discussed previously.

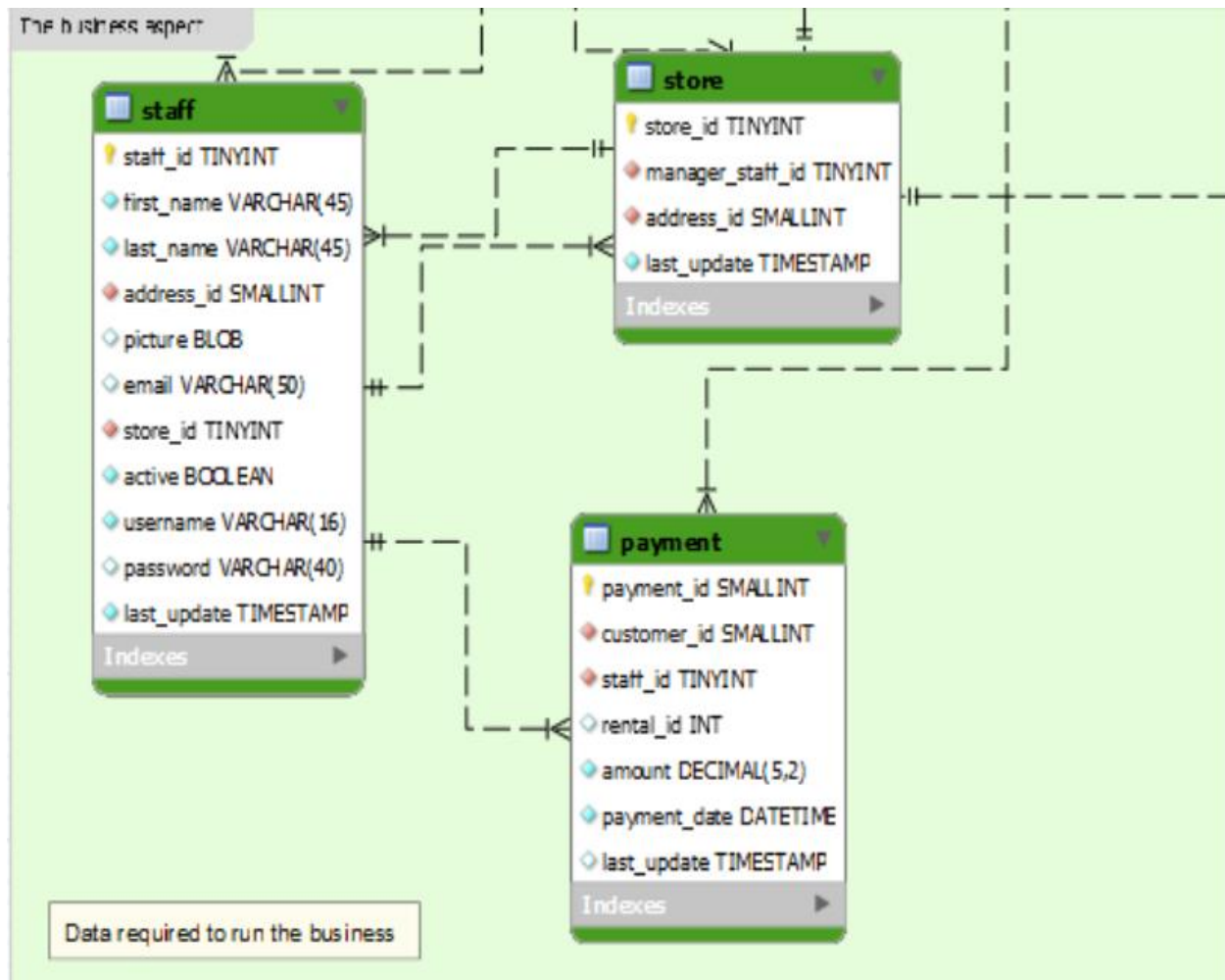
Product_category table has the above given attributes where all the domains have been discussed earlier except tiny integer. This is an incredibly tiny integers that can also be either signed as well as unsigned. The allowed range, if annotated, comprises -128 to 127. If the value is unsigned, the spectrum is 0 to 255. A breadth of up to four digits can be specified.

The category table has category_id, name and last_update attributes. The domains have been discussed earlier.

Finally, the inventory table consists of inventory_id, product_id, store_id and last_update. All these domains have been discussed except for MEDIUMINT or medium integer. A medium-sized integer defines as MEDIUMINT. MEDIUMINT SIGNED has a significance level value of -8388608 and 8388607, respectively. The minimal and highest UNSIGNED values fall within the ranges of 0 to 16777215, respectively. Each row comprises upto 3 bytes long.

The relationships between the tables are all one to many or 1:n relationships. Product is connected in one to many fashion with the product category meaning there can be multiple category of products which will be connected with all the single products not the other way around. The category table is connected with product category in one to many fashion because of the same reason. The category_id acts as primary key towards the category_id in the product_category table. The product table is connected with the inventory table in one to many fashion meaning there can be many products in the inventory rather than one product being in the inventory only.

3.3.1.3 The business aspects



The business-related data section contains three main tables which are “staff”, “store” and “payment”.

The staff table contain all the attributes which describes the details about the staff of the organization. All the attributes and domains have been discussed earlier.

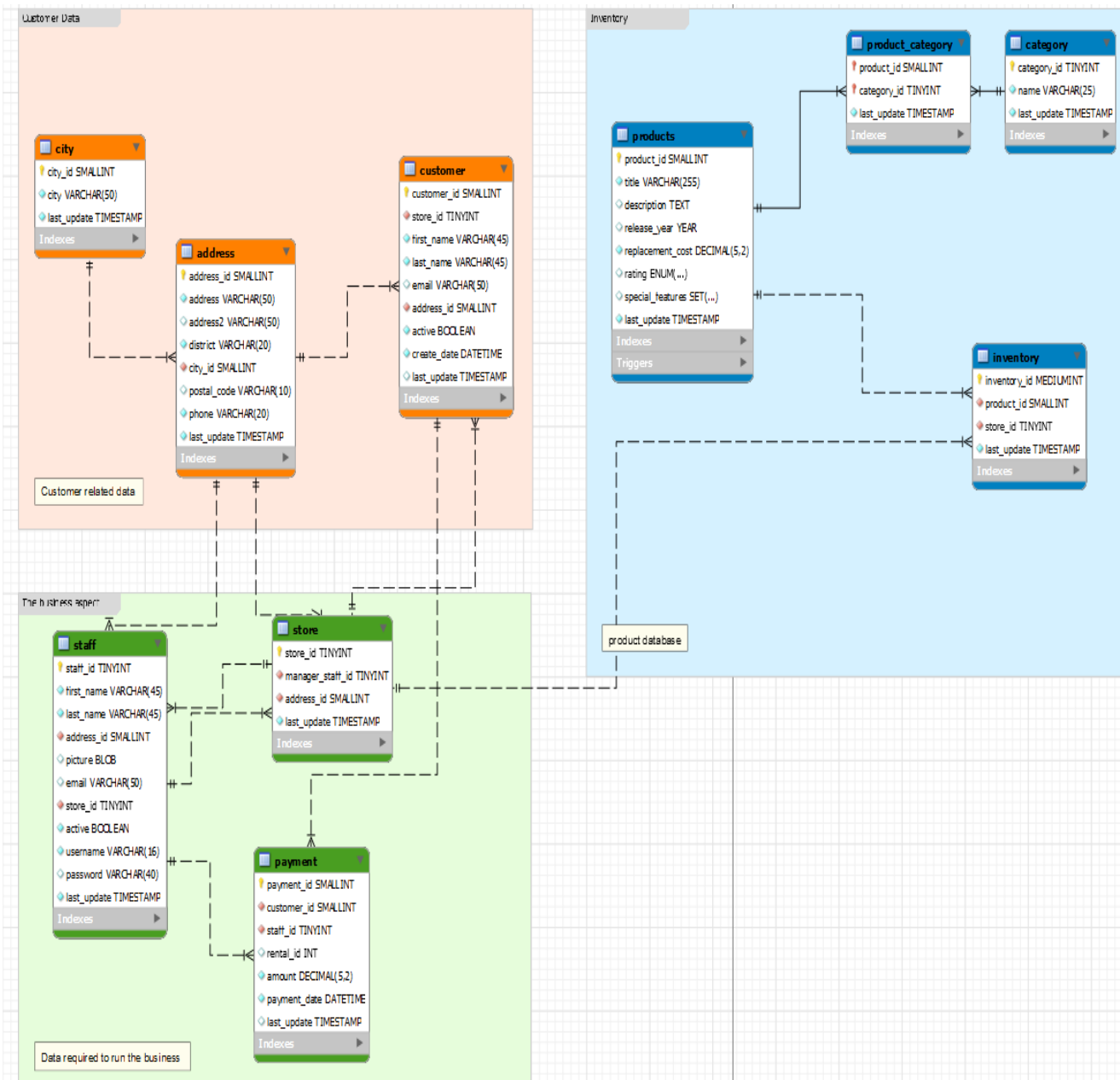
The store table contains the information about the store and the attributes required to denote the information about the store as there can be multiple outlet of the store. The attributes and the domain have been discussed earlier.

The payment table contains the payment information emphasizing both on the staff, customer and the supplier as well as the expenses required to run the store. The attributes speaks for itself and the domains have been discussed earlier.

All the tables are also connected in a one to many or 1:n fashion. Staff to store, store to staff and staff to payment are all in 1:n relationships. Staff to store and store to staff are both connected in a 1:n fashion because one staff might have duty on multiple store as well as one store might have multiple staffs. Incase of staff to payment, there might be multiple payments including incentive , commission, salary of one staff. Each staff has a different amount and variations of payroll.

3.3.2 The final relational data model

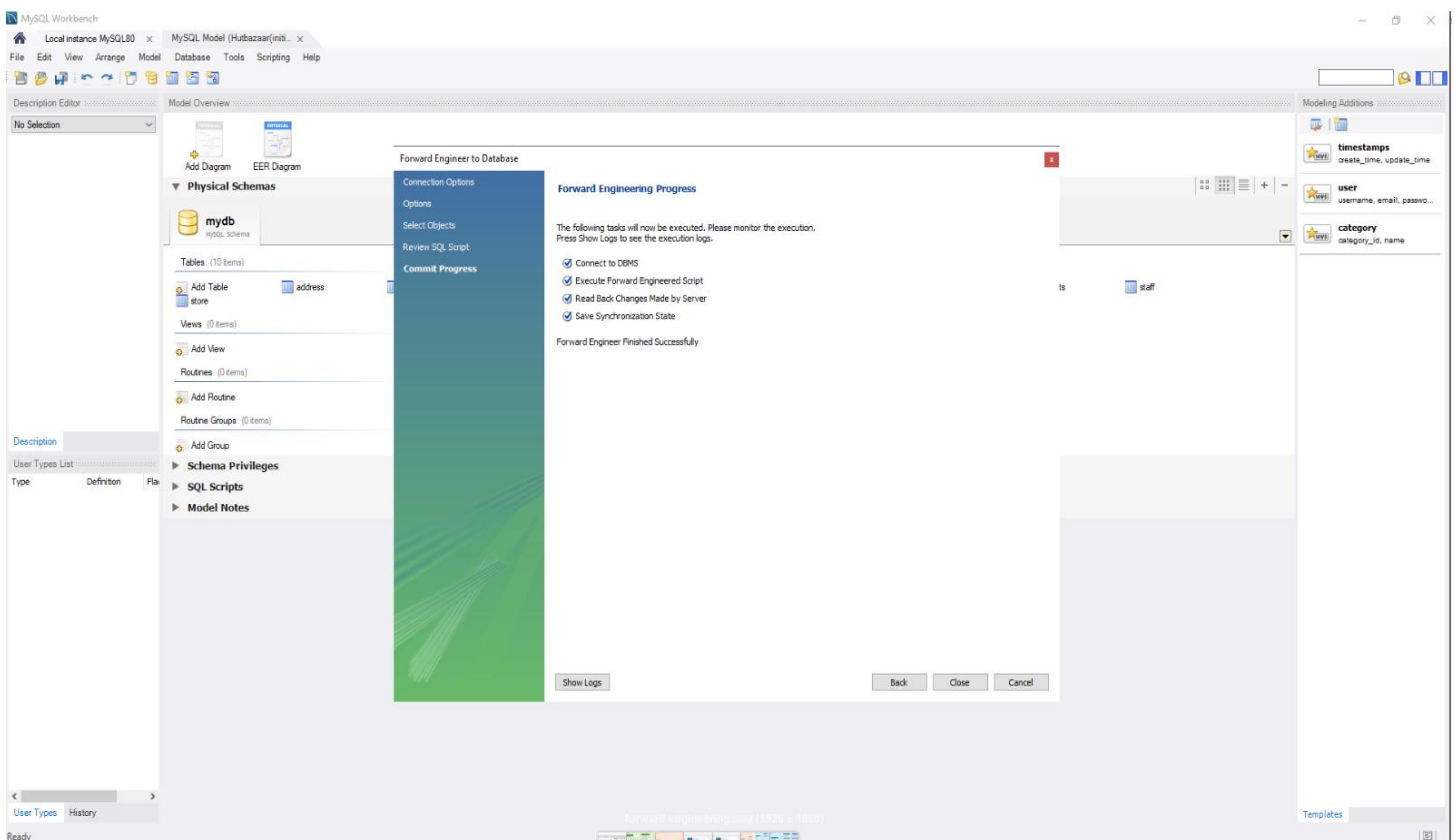
After connecting every sub aspect of the relational table we get the following table given on the next page.



The relational model the company “Hutbazaar” given above is connected in a one to many fashion where the address table and the customer table is connected with the staff table, store table and the payment table of the business related portion whereas the inventory table of the inventory related data is connected solely with the store table of the business related portion in the relational data model creating a interconnected relationship with all of the three aspects of “Hutbazaar” data model.

3.3.3 Database Deployment and forward engineering of the data model in the server

The following attachment is the deployment and forward engineering of the database of “Hutbazaar”.



The code that was generation during forward engineering procedure and database deployment of the server is included below.

```
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-----

-- Schema mydb
-----

-- Schema mydb
-----

CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;
USE `mydb` ;

-----

-- Table `mydb`.`city`
-----

CREATE TABLE IF NOT EXISTS `mydb`.`city` (
  `city_id` INT NOT NULL,
  `city` VARCHAR(45) NOT NULL,
  `last_update` TIMESTAMP NOT NULL,
  PRIMARY KEY (`city_id`))
ENGINE = InnoDB;

-----

-- Table `mydb`.`address`
-----

CREATE TABLE IF NOT EXISTS `mydb`.`address` (
  `address_id` INT NOT NULL,
  `address` VARCHAR(45) NOT NULL,
  `address2` VARCHAR(45) NULL,
  `district` VARCHAR(45) NOT NULL,
  `city_id` VARCHAR(45) NOT NULL,
  `postal_code` VARCHAR(10) NULL,
  `phone` VARCHAR(20) NOT NULL,
```

```

`last_update` TIMESTAMP NOT NULL,
`city_city_id` INT NOT NULL,
PRIMARY KEY (`address_id`),
INDEX `fk_address_city_idx` (`city_city_id` ASC) VISIBLE,
CONSTRAINT `fk_address_city`
  FOREIGN KEY (`city_city_id`)
    REFERENCES `mydb`.`city` (`city_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-----
-- Table `mydb`.`customer`
-----

```

```

CREATE TABLE IF NOT EXISTS `mydb`.`customer` (
  `customer_id` INT NOT NULL,
  `store_id` INT NOT NULL,
  `first_name` VARCHAR(45) NOT NULL,
  `last_name` VARCHAR(45) NOT NULL,
  `email` VARCHAR(45) NULL,
  `address_id` INT NOT NULL,
  `create_date` DATETIME NOT NULL,
  `last_update` TIMESTAMP NULL,
  `address_address_id` INT NOT NULL,

```

```

  PRIMARY KEY (`customer_id`),
  INDEX `fk_customer_address1_idx` (`address_address_id` ASC) VISIBLE,
  CONSTRAINT `fk_customer_address1`
    FOREIGN KEY (`address_address_id`)
      REFERENCES `mydb`.`address` (`address_id`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-----
-- Table `mydb`.`store`
-----

```

```

CREATE TABLE IF NOT EXISTS `mydb`.`store` (
  `store_id` INT NOT NULL,
  `manager_staff_id` INT NOT NULL,

```

```

`address_id` INT NOT NULL,
`last_update` TIMESTAMP NOT NULL,
`address_address_id` INT NOT NULL,
`staff_staff_id` INT NOT NULL,
PRIMARY KEY (`store_id`),
INDEX `fk_store_address1_idx` (`address_address_id` ASC) VISIBLE,
INDEX `fk_store_staff1_idx` (`staff_staff_id` ASC) VISIBLE,
CONSTRAINT `fk_store_address1`
  FOREIGN KEY (`address_address_id`)
    REFERENCES `mydb`.`address` (`address_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
CONSTRAINT `fk_store_staff1`
  FOREIGN KEY (`staff_staff_id`)
    REFERENCES `mydb`.`staff` (`staff_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-----
-- Table `mydb`.`staff`
-----

CREATE TABLE IF NOT EXISTS `mydb`.`staff` (
  `staff_id` INT NOT NULL,
  `first_name` VARCHAR(45) NOT NULL,
  `last_name` VARCHAR(45) NOT NULL,
  `address_id` INT NOT NULL,
  `picture` BLOB NULL,
  `email` VARCHAR(45) NULL,
  `store_id` INT NOT NULL,
  `username` VARCHAR(45) NOT NULL,
  `password` VARCHAR(50) NULL,
  `last_update` TIMESTAMP NOT NULL,
  `address_address_id` INT NOT NULL,
  `store_store_id` INT NOT NULL,
  PRIMARY KEY (`staff_id`),
  INDEX `fk_staff_address1_idx` (`address_address_id` ASC) VISIBLE,
  INDEX `fk_staff_store1_idx` (`store_store_id` ASC) VISIBLE,
  CONSTRAINT `fk_staff_address1`

```

```

FOREIGN KEY (`address_address_id`)
REFERENCES `mydb`.`address` (`address_id`)
ON DELETE NO ACTION
ON UPDATE NO ACTION,
CONSTRAINT `fk_staff_store1`
FOREIGN KEY (`store_store_id`)
REFERENCES `mydb`.`store` (`store_id`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;

-----

-- Table `mydb`.`payment`
-----

CREATE TABLE IF NOT EXISTS `mydb`.`payment` (
  `payment_id` INT NOT NULL,
  `customer_id` INT NOT NULL,
  `staff_id` INT NOT NULL,
  `amount` DECIMAL(5,2) NULL,
  `payment_date` DATETIME NULL,
  `last_update` TIMESTAMP NULL,
  `staff_staff_id` INT NOT NULL,
  `customer_customer_id` INT NOT NULL,
  PRIMARY KEY (`payment_id`),
  INDEX `fk_payment_staff1_idx` (`staff_staff_id` ASC) VISIBLE,
  INDEX `fk_payment_customer1_idx` (`customer_customer_id` ASC) VISIBLE,
  CONSTRAINT `fk_payment_staff1`
    FOREIGN KEY (`staff_staff_id`)
      REFERENCES `mydb`.`staff` (`staff_id`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION,
  CONSTRAINT `fk_payment_customer1`
    FOREIGN KEY (`customer_customer_id`)
      REFERENCES `mydb`.`customer` (`customer_id`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION)
ENGINE = InnoDB;

-----

-- Table `mydb`.`products`

```

```

-----
CREATE TABLE IF NOT EXISTS `mydb`.`products` (
  `product_id` INT NOT NULL,
  `title` VARCHAR(45) NOT NULL,
  `description` TEXT NULL,
  `replacement_cost` DECIMAL(5,2) NOT NULL,
  `rating` VARCHAR(45) NULL,
  `last_update` TIMESTAMP NOT NULL,
  PRIMARY KEY (`product_id`))
ENGINE = InnoDB;
-----

-- Table `mydb`.`category`
-----

CREATE TABLE IF NOT EXISTS `mydb`.`category` (
  `category_id` INT NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  `last_update` TIMESTAMP NOT NULL,
  PRIMARY KEY (`category_id`))
ENGINE = InnoDB;
-----

-- Table `mydb`.`product_category`
-----

CREATE TABLE IF NOT EXISTS `mydb`.`product_category` (
  `product_id` INT NOT NULL,
  `category_id` INT NOT NULL,
  `last_update` TIMESTAMP NOT NULL,
  `products_product_id` INT NOT NULL,
  `category_category_id` INT NOT NULL,
  PRIMARY KEY (`product_id`, `category_id`, `products_product_id`, `category_category_id`),
  INDEX `fk_product_category_products1_idx` (`products_product_id` ASC) VISIBLE,
  INDEX `fk_product_category_category1_idx` (`category_category_id` ASC) VISIBLE,
  CONSTRAINT `fk_product_category_products1`
    FOREIGN KEY (`products_product_id`)
      REFERENCES `mydb`.`products` (`product_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_product_category_category1`
    FOREIGN KEY (`category_category_id`)

```

```

REFERENCES `mydb`.`category` (`category_id`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;

-----

-- Table `mydb`.`inventory`
-----

CREATE TABLE IF NOT EXISTS `mydb`.`inventory` (
  `inventory_id` INT NOT NULL,
  `product_id` INT NOT NULL,
  `store_id` INT NOT NULL,
  `last_update` TIMESTAMP NOT NULL,
  `store_store_id` INT NOT NULL,
  `products_product_id` INT NOT NULL,
  PRIMARY KEY (`inventory_id`),
  INDEX `fk_inventory_store1_idx` (`store_store_id` ASC) VISIBLE,
  INDEX `fk_inventory_products1_idx` (`products_product_id` ASC) VISIBLE,
  CONSTRAINT `fk_inventory_store1`
    FOREIGN KEY (`store_store_id`)
      REFERENCES `mydb`.`store` (`store_id`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION,
  CONSTRAINT `fk_inventory_products1`
    FOREIGN KEY (`products_product_id`)
      REFERENCES `mydb`.`products` (`product_id`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION)
ENGINE = InnoDB;

SET SQL_MODE=@OLD_SQL_MODE;

SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;

```

Finally, the tables of the database have been populated with hypothetical data entry in order to demonstrate the uses of query language. The following section will show snaps/pictures of hypothetical population of every table of the database server.

3.3.4 Populating the database

Here the snaps/pictures taken of every table after populating with data entry on MySQL Workbench:

Address:

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane displays a tree view of the 'mydb' database, with the 'address' table selected. The main window shows the 'address' table with the following data:

address_id	address	address2	district	city_id	postal_code	phone	last_update	city_city_id
1	Badda	Neturbarazar	Dhaka	1	1207	880-1717990022	2014-09-25 22:30:27	1
2	Gulshan	Dhanmondi	Dhaka	2	1205	880-171619966	2014-09-25 22:30:09	2
3	Bahari	Pakirpara	Dinagpur	3	1210	880-1717003322	2015-09-26 21:20:32	3
4	Hoghbazar	Mugda	Dhaka	4	1220	880-1765468954	2015-09-26 19:20:22	4
5	Mirpur	Gabrail	Dhaka	5	1205	880-1956476315	2016-05-23 10:20:27	5

The bottom pane shows the 'Output' window with the following message:

```
1 22:24:55 SELECT * FROM mydb.address LIMIT 0, 1000
5 row(s) returned
0.015 sec / 0.000 sec
```

category:

The screenshot shows the MySQL Workbench interface with a query executed on the 'category' table. The query is `SELECT * FROM mydb.category;`. The result grid displays 5 rows of data. The output pane shows the execution details.

category_id	name	last_update
1	Beverage	2014-09-25 22:30:27
2	Cheese	2015-10-25 12:18:12
3	Kitchen Utensils	2014-05-18 12:20:45
4	Fisheries	2014-05-14 13:18:30
5	Rice	2017-10-20 17:25:38

Output:

#	Time	Action	Message	Duration / Fetch
1	22:24:55	SELECT * FROM mydb.address LIMIT 0, 1000	5 row(s) returned	0.015 sec / 0.000 sec
2	22:28:54	SELECT * FROM mydb.category LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec

City:

The screenshot shows the MySQL Workbench interface with a query executed on the 'city' table. The query is `SELECT * FROM mydb.city;`. The result grid displays 5 rows of data. The output pane shows the execution details.

city_id	city	last_update
1	Dhaka	2006-02-15 04:45:25
2	Dhaka	2005-06-17 05:26:37
3	Dagpur	2008-09-26 18:15:40
4	Bangura	2007-05-28 20:16:40
5	Rangpur	2008-06-25 18:16:55

Output:

#	Time	Action	Message	Duration / Fetch
1	22:24:55	SELECT * FROM mydb.address LIMIT 0, 1000	5 row(s) returned	0.015 sec / 0.000 sec
2	22:28:54	SELECT * FROM mydb.category LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
3	22:30:38	SELECT * FROM mydb.city LIMIT 0, 1000	5 row(s) returned	0.016 sec / 0.000 sec

Customer:

The screenshot shows the MySQL Workbench interface with the 'customer' table selected in the 'mydb' database. The table structure is displayed on the left, and the query results are shown in the main window.

Table: customer

Columns:	customer_id	first_name	last_name	email	address_id	create_date	last_update	address_address_id
customer_id	int	PK						
first_name	varchar(45)							
last_name	varchar(45)							
email	varchar(45)							
address_id	int							
create_date	datetime							
last_update	timestamp							
address_address_id	int							

Query Results:

customer_id	store_id	first_name	last_name	email	address_id	create_date	last_update	address_address_id
1	1	Shiro	Hossein	shiro1757@gmail.com	1	2006-02-14 22:04:36	2009-12-12 11:11:50	1
2	2	Jamul	Idan	jamul757@gmail.com	2	2006-02-14 22:00:10	2009-12-12 11:11:50	2
3	3	Pronay	Hossein	pronayhossein123@gmail.com	3	2006-02-14 22:00:10	2009-11-11 00:00:00	3
4	4	Tamir	Dani	tamir1757@gmail.com	4	2006-02-14 22:00:10	2009-11-11 00:00:00	4
5	5	Abnar	Promit	promit1757@gmail.com	5	2006-02-14 22:00:10	2009-11-11 00:00:00	5

Action Output:

#	Time	Action	Message	Duration / Fetch
1	22:24:55	SELECT * FROM mydb.address LIMIT 0, 1000	5 row(s) returned	0.015 sec / 0.000 sec
2	22:28:54	SELECT * FROM mydb.category LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
3	22:30:38	SELECT * FROM mydb.city LIMIT 0, 1000	5 row(s) returned	0.016 sec / 0.000 sec
4	22:31:52	SELECT * FROM mydb.customer LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec

Inventory:

The screenshot shows the MySQL Workbench interface with the 'inventory' table selected in the 'mydb' database. The table structure is displayed on the left, and the query results are shown in the main window.

Table: inventory

Columns:	inventory_id	product_id	store_id	last_update	store_store_id	products_product_id
inventory_id	int	PK				
product_id	int					
store_id	int					
last_update	timestamp					
store_store_id	int					
products_product_id	int					

Query Results:

inventory_id	product_id	store_id	last_update	store_store_id	products_product_id
1	1	1	2006-02-14 22:04:36	1	1
2	2	2	2006-02-14 22:04:36	2	2
3	3	3	2006-02-14 22:04:36	3	3
4	4	4	2006-02-14 22:04:36	4	4
5	5	5	2006-02-14 22:04:36	5	5

Action Output:

#	Time	Action	Message	Duration / Fetch
1	22:24:55	SELECT * FROM mydb.address LIMIT 0, 1000	5 row(s) returned	0.015 sec / 0.000 sec
2	22:28:54	SELECT * FROM mydb.category LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
3	22:30:38	SELECT * FROM mydb.city LIMIT 0, 1000	5 row(s) returned	0.016 sec / 0.000 sec
4	22:31:52	SELECT * FROM mydb.customer LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
5	22:33:36	SELECT * FROM mydb.inventory LIMIT 0, 1000	5 row(s) returned	0.015 sec / 0.000 sec

Payment:

The screenshot shows the MySQL Workbench interface with the 'payment' table selected in the 'Schemas' pane. The table structure is displayed in the 'Table: payment' pane, showing columns: payment_id (int PK), customer_id (int), staff_id (int), amount (decimal), payment_date (datetime), last_update (timestamp), staff_staff_id (int), and customer_customer_id (int). The 'Result Grid' shows the query results for the 'payment' table, limited to 1000 rows. The 'Output' pane shows the execution log for the query.

payment_id	customer_id	staff_id	amount	payment_date	last_update	staff_staff_id	customer_customer_id
1	1	1	450.00	2008-02-14 22:04:36	2008-02-14 22:04:36	1	1
2	2	2	50.00	2007-02-14 22:04:36	2008-02-14 22:04:36	2	2
3	3	3	123.00	2008-02-14 22:04:36	2008-02-14 22:04:36	3	3
4	4	4	789.00	2008-02-14 22:04:36	2008-02-14 22:04:36	4	4
5	5	5	456.00	2010-02-14 22:04:36	2008-02-14 22:04:36	5	5

Output:

Time	Action	Message	Duration / Fetch
2 22:29:54	SELECT * FROM mydb.category LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
3 22:30:38	SELECT * FROM mydb.city LIMIT 0, 1000	5 row(s) returned	0.016 sec / 0.000 sec
4 22:31:52	SELECT * FROM mydb.customer LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
5 22:33:36	SELECT * FROM mydb.inventory LIMIT 0, 1000	5 row(s) returned	0.015 sec / 0.000 sec
6 22:35:13	SELECT * FROM mydb.payment LIMIT 0, 1000	5 row(s) returned	0.016 sec / 0.000 sec

Product Category:

The screenshot shows the MySQL Workbench interface with the 'product_category' table selected in the 'Schemas' pane. The table structure is displayed in the 'Table: product_category' pane, showing columns: product_id (int PK), category_id (int PK), last_update (timestamp), products_product_id (int), and category_category_id (int). The 'Result Grid' shows the query results for the 'product_category' table, limited to 1000 rows. The 'Output' pane shows the execution log for the query.

product_id	category_id	last_update	products_product_id	category_category_id
1	1	2006-02-15 05:03:42	1	1
2	5	2006-02-15 05:03:42	2	5
3	4	2006-02-15 05:03:42	3	4
4	2	2006-02-15 05:03:42	4	2
5	3	2006-02-15 05:03:42	5	3

Output:

Time	Action	Message	Duration / Fetch
3 22:30:38	SELECT * FROM mydb.city LIMIT 0, 1000	5 row(s) returned	0.016 sec / 0.000 sec
4 22:31:52	SELECT * FROM mydb.customer LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
5 22:33:36	SELECT * FROM mydb.inventory LIMIT 0, 1000	5 row(s) returned	0.015 sec / 0.000 sec
6 22:35:13	SELECT * FROM mydb.payment LIMIT 0, 1000	5 row(s) returned	0.016 sec / 0.000 sec
7 22:36:54	SELECT * FROM mydb.product_category LIMIT 0, 1000	5 row(s) returned	0.031 sec / 0.000 sec

Products:

The screenshot shows the MySQL Workbench interface. The 'Schemas' pane on the left lists the 'mydb' database with various tables. The 'Table: products' is selected, showing its columns: product_id (INT PK), title (VARCHAR(45)), description (TEXT), product_cost (DECIMAL(5,2)), rating (VARCHAR(45)), and last_update (TIMESTAMP). The 'Result Grid' displays the query results for 'SELECT * FROM mydb.products;'. The table has 6 columns: product_id, title, description, product_cost, rating, and last_update. The 'Action Output' pane at the bottom shows the execution of the query, indicating 5 rows returned.

product_id	title	description	product_cost	rating	last_update
1	Cocacola	A soft drink	35.00	Good	2006-02-15 09:03:42
2	Bismark...	Starch-rich food	45.00	Great	2006-02-15 09:03:42
3	Silvercup ...	Fish from the river	150.00	average	2006-02-15 09:03:42
4	Lays	Chips made by a r...	70.00	Great	2006-02-15 09:03:42
5	Vimbar	A deli washing soap	30.00	Good	2006-02-15 09:03:42

Staff:

The screenshot shows the MySQL Workbench interface. The 'Schemas' pane on the left lists the 'mydb' database with various tables. The 'Table: staff' is selected, showing its columns: staff_id (INT PK), first_name (VARCHAR(45)), last_name (VARCHAR(45)), address_id (INT), picture (BLOB), email (VARCHAR(45)), store_id (INT), username (VARCHAR(45)), password (VARCHAR(50)), last_update (TIMESTAMP), address_address_id (INT), and store_store_id (INT). The 'Result Grid' displays the query results for 'SELECT * FROM mydb.staff;'. The table has 12 columns: staff_id, first_name, last_name, address_id, picture, email, store_id, username, password, last_update, address_address_id, and store_store_id. The 'Action Output' pane at the bottom shows the execution of the query, indicating 5 rows returned.

staff_id	first_name	last_name	address_id	picture	email	store_id	username	password	last_update	address_address_id	store_store_id
1	Meraz	Hossan	6		meraz	1	meraz	ae0f2224	2006-02-14 22:04:36	6	1
2	Saber	Alien	7		cabr	2	cabr	232466	2006-02-14 22:04:36	7	2
3	Jabid	Tommy	8			1	tommy	464466	2006-02-14 22:04:36	8	1
4	Manzar	Hossan	9			2	manzar	789789	2006-02-14 22:04:36	9	2
5	Fardin	Latif	10		fardin	1	fardin	ee9ee9e	2006-02-14 22:04:36	10	1

Store:

The screenshot displays the MySQL Workbench interface. On the left, the 'SCHEMAS' pane shows the 'mydb' database with various tables and views. The 'store' table is selected, and its structure is shown in the 'Table: store' pane. The table has columns: store_id (INT, PK), manager_staff_id (INT), address_id (INT), last_update (TIMESTAMP), address_address_id (INT), and staff_staff_id (INT). The 'Result Grid' shows the following data:

store_id	manager_staff_id	address_id	last_update	address_address_id	staff_staff_id
1	1	8	2006-02-14 22:04:36	8	1
2	2	9	2006-02-14 22:04:36	9	2
3	3	10	2006-02-14 22:04:36	10	3
4	4	11	2006-02-14 22:04:36	11	4
5	5	12	2006-02-14 22:04:36	12	5

The 'Action Output' pane at the bottom shows the execution of several queries, including SELECT statements with LIMIT clauses, and their execution times.

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Chapter 4

- query generation and testing

4.1 Introduction

This chapter will demonstrate the usage of SQL query language to create, update and delete data on the MySQL server using MySQL Workbench. The queries will be discussion on the later chapter which will explain why the following query languages have been used to alter the particular data on the data server. This part of the chapter will contain 10 queries along with the codes and pictures of the resulting outcome from the particular query entry.

4.2 Query codes and pictures of the resulting outcome

Here are the following 10 queries codes alongside the pictures of the resulting outcome of the queries in MySQL workbench:

1. USE mydb;

```
SELECT customer_id, concat(first_name," ",last_name) AS full_name
```

```
FROM customer;
```

Result:

The screenshot displays the MySQL Workbench interface. The left sidebar shows the 'SCHEMAS' tree with 'mydb' selected. The central pane shows the SQL editor with the following query:

```
1 USE mydb;
2 SELECT customer_id, concat(first_name," ",last_name) AS full_name
3 FROM customer;
```

The 'Result Grid' shows the following data:

customer_id	full_name
1	Shiryo Hassan
2	Jamal Islam
3	Pronay Hassan
4	Tanner Doris
5	Abner Prout

The bottom panel shows the 'Action Output' with the following log entries:

#	Time	Action	Message	Duration / Feas
8	22:38:10	SELECT * FROM mydb.products LIMIT 0, 1000	5 row(s) returned	0.016 sec / 0.000 sec
9	22:39:20	SELECT * FROM mydb.staff LIMIT 0, 1000	5 row(s) returned	0.016 sec / 0.000 sec
10	22:40:32	SELECT * FROM mydb.store LIMIT 0, 1000	5 row(s) returned	0.016 sec / 0.000 sec
11	22:57:02	USE mydb	0 row(s) affected	0.000 sec
12	22:57:02	SELECT customer_id, concat(first_name," ",last_name) AS full_name FROM customer LIMIT 0, 1000	5 row(s) returned	0.031 sec / 0.000 sec

2. USE mydb;

SELECT last_name, first_name, store_id, username

FROM staff

WHERE last_name LIKE '%SS%'

ORDER BY last_name, first_name, store_id, Username;

Result:

The screenshot shows the MySQL Workbench interface. The 'Schemas' pane on the left lists the 'mydb' database. The 'Query Editor' in the center contains the following SQL query:

```
1. USE mydb;
2. SELECT last_name, first_name, store_id, username
3. FROM staff
4. WHERE last_name LIKE '%SS%'
5. ORDER BY last_name, first_name, store_id, Username;
6.
```

The 'Result Grid' below the query shows the results of the query:

last_name	first_name	store_id	username
Hossan	Manzar	2	manzar

The 'Output' pane at the bottom shows the execution log:

#	Time	Action	Message	Duration / Fetch
12	22:57:02	SELECT customer_id, concat(first_name, ' ', last_name) AS full_name FROM customer LIMIT 0, 1000	5 row(s) returned	0.031 sec / 0.000 sec
13	23:02:55	USE mydb	0 row(s) affected	0.000 sec
14	23:02:55	SELECT last_name, first_name FROM staff WHERE last_name LIKE '%SS%' ORDER BY last_name, first_name LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
15	23:04:11	USE mydb	0 row(s) affected	0.000 sec
16	23:04:11	SELECT last_name, first_name, store_id, username FROM staff WHERE last_name LIKE '%SS%' ORDER BY last_name, first_name, store_id, Username	2 row(s) returned	0.000 sec / 0.000 sec

3. USE mydb;

SELECT address, address2, city_id,district

FROM address

WHERE district IN ('Dhaka');

Result:

The screenshot shows the MySQL Workbench interface. The 'Schemas' pane on the left lists the 'mydb' database. The 'Query Editor' in the center contains the following SQL query:

```
1 USE mydb;
2 SELECT address, address2, city_id,district
3 FROM address
4 WHERE district IN ('Dhaka');
```

The 'Result Grid' pane displays the query results in a table with the following data:

address	address2	city_id	district
Badda	Nurulbazar	1	Dhaka
Gulshan	Dharmond	2	Dhaka
Moghbazar	Mugda	4	Dhaka
Mirpur	Gabrail	5	Dhaka

The 'Output' pane at the bottom shows the execution log with the following entries:

Time	Action	Message	Duration / Fetch
15 23:04:11	USE mydb	0 row(s) affected	0.000 sec
16 23:04:11	SELECT last_name, first_name,store_id,username FROM staff WHERE last_name LIKE 'S%' ORDER BY last_name, first_name,store_id,Usen...	2 row(s) returned	0.000 sec / 0.000 sec
17 23:26:41	SELECT** FROM mydb.address LIMIT 0, 1000	5 row(s) returned	0.016 sec / 0.000 sec
18 23:29:28	USE mydb	0 row(s) affected	0.000 sec
19 23:29:28	SELECT address, address2, city_id,district FROM address WHERE district IN ('Dhaka') LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec

4. USE mydb;

SELECT first_name, last_name, SUM(amount)

FROM staff s

INNER JOIN payment p

ON s.staff_id = p.staff_id

GROUP BY p.staff_id

ORDER BY last_name ASC;

Result:

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
1 USE mydb;
2 SELECT first_name, last_name, SUM(amount)
3 FROM staff s
4 INNER JOIN payment p
5 ON s.staff_id = p.staff_id
6 GROUP BY p.staff_id
7 ORDER BY last_name ASC;
```

The Results window displays the following data:

first_name	last_name	SUM(amount)
Saleh	Alan	50.00
Manzar	Hossain	480.00
Manzar	Hossain	789.00
Fardin	Latif	496.00
Isabel	Trenney	123.00

The Action Output window shows the execution log:

#	Time	Action	Message	Duration / Fetch
19	23:29:28	SELECT address, address2, city_id, district FROM address WHERE district IN (Dhaka) LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
20	23:34:12	USE Sakila	0 row(s) affected	0.016 sec
21	23:34:12	SELECT first_name, last_name, SUM(amount) FROM staff s INNER JOIN payment p ON s.staff_id = p.staff_id GROUP BY p.staff_id ORDER BY...	2 row(s) returned	0.156 sec / 0.000 sec
22	23:34:55	USE mydb	0 row(s) affected	0.000 sec
23	23:34:55	SELECT first_name, last_name, SUM(amount) FROM staff s INNER JOIN payment p ON s.staff_id = p.staff_id GROUP BY p.staff_id ORDER BY...	5 row(s) returned	0.016 sec / 0.000 sec

5. USE mydb;

SELECT last_name, first_name, SUM(amount)

FROM payment p

INNER JOIN customer c

ON p.customer_id = c.customer_id

GROUP BY p.customer_id

ORDER BY last_name ASC;

Result:

The screenshot displays the MySQL Workbench interface. The SQL editor at the top contains the following query:

```
1 * USE mydb;
2 SELECT last_name, first_name, SUM(amount)
3 FROM payment p
4 INNER JOIN customer c
5 ON p.customer_id = c.customer_id
6 GROUP BY p.customer_id
7 ORDER BY last_name ASC;
```

The Results tab below the editor shows a table with the following data:

last_name	first_name	SUM(amount)
Doris	Tamir	789.00
Hossein	Shuvo	450.00
Hossein	Promay	123.00
Idan	Samuel	90.00
Promit	Abnar	456.00

The bottom pane shows the Action Output log with the following entries:

#	Time	Action	Message	Duration / Fetch
21	23:34:12	SELECT first_name, last_name, SUM(amount) FROM staff s INNER JOIN payment p ON s.staff_id = p.staff_id GROUP BY p.staff_id ORDER BY...	2 row(s) returned	0.156 sec / 0.000 sec
22	23:34:55	USE mydb	0 row(s) affected	0.000 sec
23	23:34:55	SELECT first_name, last_name, SUM(amount) FROM staff s INNER JOIN payment p ON s.staff_id = p.staff_id GROUP BY p.staff_id ORDER BY...	5 row(s) returned	0.016 sec / 0.000 sec
24	23:42:33	USE mydb	0 row(s) affected	0.000 sec
25	23:42:33	SELECT last_name, first_name, SUM(amount) FROM payment p INNER JOIN customer c ON p.customer_id = c.customer_id GROUP BY p.customer...	5 row(s) returned	0.047 sec / 0.000 sec

6. USE mydb;

UPDATE customer

SET first_name =

CASE

WHEN first_name = 'Tanvir'

THEN 'Samiul'

ELSE 'Samiul Ameen'

END

WHERE customer_id =4;

Result:

The screenshot displays the MySQL Workbench interface. The 'SQL Editor' window contains the following SQL query:

```
1 USE mydb;
2 UPDATE customer
3 SET first_name =
4 CASE
5 WHEN first_name = 'Tanvir'
6 THEN 'Samiul'
7 ELSE 'Samiul Ameen'
8 END
9 WHERE customer_id =4;
```

The 'Result Grid' shows the state of the 'customer' table before the update:

customer_id	store_id	first_name	last_name	email	address_id	create_date	last_update	address_address_id
1	1	Shuvo	Hossain	shuvo777@gmail.com	1	2006-02-14 23:00:10	2008-12-12 11:11:50	1
2	2	Jamul	Islam	jamul777@gmail.com	2	2006-02-14 23:00:10	2008-12-12 11:11:50	2
3	3	Pranay	Hossain	pranayhossain123@gmail.com	3	2006-02-14 23:00:10	2009-11-11 00:00:00	3
4	4	Samiul Ameen	Dorzi	ameen123@gmail.com	4	2006-02-14 23:00:10	2009-11-11 00:00:00	4
5	5	Abrar	Pront	prontfrozen@gmail.com	5	2006-02-14 23:00:10	2009-11-11 00:00:00	5

The 'Output' window shows the execution log:

Time	Action	Messages	Duration / Tech
30 23:57:29	UPDATE customer SET first_name = CASE WHEN first_name = 'Tanvir' THEN 'Ameen' ELSE 'Samiul Ameen' END WHERE customer_id =4	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.062 sec
31 23:57:33	SELECT * FROM mydb.customer LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
32 23:57:59	USE mydb	0 row(s) affected	0.000 sec
33 23:57:59	UPDATE customer SET first_name = CASE WHEN first_name = 'Tanvir' THEN 'Samiul' ELSE 'Samiul Ameen' END WHERE customer_id =4	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.063 sec
34 23:58:10	SELECT * FROM mydb.customer LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec

7. USE mydb;

SELECT rating, title, last_update, description

FROM products

WHERE rating='Good';

Result:

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
1 USE mydb;
2 SELECT rating, title, last_update, description
3 FROM products
4 WHERE rating='Good';
```

The query is executed, but an error occurs. The output pane shows the following error messages:

Time	Action	Message	Duration / Fetch
48 00:05:42	SELECT rating, title, last_update, description WHERE rating='Good'	Error Code: 1054. Unknown column 'rating' in 'field list'	0.031 sec
49 00:05:44	USE mydb	0 row(s) affected	0.000 sec
50 00:05:44	SELECT rating, title, last_update, description WHERE rating='Good'	Error Code: 1054. Unknown column 'rating' in 'field list'	0.000 sec
51 00:06:14	USE mydb	0 row(s) affected	0.015 sec
52 00:06:14	SELECT rating, title, last_update, description FROM products WHERE rating='Good' LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec

The result grid shows two rows of data:

rating	title	last_update	description
Good	Cocacola	2006-02-15 05:03:42	A soft drink
Good	Veibler	2006-02-15 05:03:42	A dish washing soap

8. USE mydb;

SELECT store.store_id, SUM(amount)

FROM store

INNER JOIN staff

ON store.store_id = staff.store_id

INNER JOIN payment p

ON p.staff_id = staff.staff_id

GROUP BY store.store_id

ORDER BY SUM(amount);

Result:

The screenshot displays the MySQL Workbench interface. The 'SQL Editor' window contains the following query:

```
1 USE mydb;
2 SELECT store.store_id, SUM(amount)
3 FROM store
4 INNER JOIN staff
5 ON store.store_id = staff.store_id
6 INNER JOIN payment p
7 ON p.staff_id = staff.staff_id
8 GROUP BY store.store_id
9 ORDER BY SUM(amount);
10
```

The 'Result Grid' shows the output of the query:

store_id	SUM(amount)
1	3029.00

The 'Output' window shows the execution log:

#	Time	Action	Message	Duration / Fetch
71	00:25:45	SELECT * FROM mydb.staff LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
72	00:26:01	SELECT * FROM mydb.customer LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
73	00:26:05	SELECT * FROM mydb.payment LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
74	00:26:33	USE mydb	0 row(s) affected	0.000 sec
75	00:26:33	SELECT store.store_id, SUM(amount) FROM store INNER JOIN staff ON store.store_id = staff.store_id INNER JOIN payment p ON p.staff_id = staff.staff_id GROUP BY store.store_id ORDER BY SUM(amount)	2 row(s) returned	0.000 sec / 0.000 sec

9. USE mydb;

ALTER TABLE customer

ADD COLUMN middle_name VARCHAR(45);

Result:

The screenshot shows the MySQL Workbench interface. The 'Query 1' window contains the following SQL commands:

```
1 USE mydb;
2 ALTER TABLE customer
3 ADD COLUMN middle_name VARCHAR(45);
```

The 'Result Grid' displays the data for the 'customer' table after the command execution. The table has 5 rows and 9 columns: customer_id, store_id, first_name, middle_name, last_name, email, address_id, create_date, and last_update.

customer_id	store_id	first_name	middle_name	last_name	email	address_id	create_date	last_update
1	1	Shirley		Hossein		1	2006-02-14 22:04:36	2008-12-12 11:11:50
2	2	Jamul		Islam	jamulj7575@gmail.com	2	2006-02-14 23:00:10	2008-12-12 11:11:50
3	3	Pronay		Hossein	pronayhossein123@gmail.com	3	2006-02-14 23:00:10	2009-11-11 09:00:00
4	4	Samuel		Abraham		4	2006-02-14 23:00:10	2009-11-11 09:00:00
5	5	Abir		Promit	promitrozeen@gmail.com	5	2006-02-14 23:00:10	2009-11-11 09:00:00

The 'Output' window shows the execution log:

#	Time	Action	Message	Duration / Fetch
4	00:57:47	USE mydb	0 rows(s) affected	0.000 sec
5	00:57:47	ALTER TABLE customer ADD COLUMN middle_name	Error Code: 1064: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to u...	0.000 sec
6	00:58:21	USE mydb	0 rows(s) affected	0.000 sec
7	00:58:21	ALTER TABLE customer ADD COLUMN middle_name VARCHAR(45)	0 rows(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.234 sec
8	00:58:31	SELECT * FROM mydb.customer LIMIT 0, 1000	5 rows(s) returned	0.016 sec / 0.000 sec

10. USE mydb;

ALTER TABLE customer

DROP COLUMN middle_name;

The screenshot shows the MySQL Workbench interface. The 'Query' tab is active, displaying three SQL statements:

```
1 USE mydb;
2 ALTER TABLE customer
3 DROP COLUMN middle_name;
```

The 'Results' grid shows the output of the queries. The first query, 'USE mydb;', returned 0 rows. The second query, 'ALTER TABLE customer DROP COLUMN middle_name;', returned 0 rows. The third query, 'SELECT * FROM mydb.customer LIMIT 0, 1000;', returned 5 rows of customer data.

customer_id	store_id	first_name	last_name	email	address_id	create_date	last_update	address_address_id
1	1	Shiro	Hossan	shiro@hossan.com	1	2008-02-14 22:04:36	2008-12-12 11:11:00	1
2	2	Jenai	Hossan	jenai@hossan.com	2	2008-02-14 23:00:10	2008-12-12 11:11:00	2
3	3	Pronay	Hossan	pronay@hossan.com	3	2008-02-14 23:00:10	2009-11-11 00:00:00	3
4	4	Samuel	Amsen	amsen@hossan.com	4	2008-02-14 23:00:10	2009-11-11 00:00:00	4
5	5	Abner	Pronay	pronay@hossan.com	5	2008-02-14 23:00:10	2009-11-11 00:00:00	5

The 'Output' tab shows the execution log:

#	Time	Action	Message	Duration / Fetch
7	00:58:21	ALTER TABLE customer ADD COLUMN middle_name VARCHAR(45)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.234 sec
8	00:58:31	SELECT * FROM mydb.customer LIMIT 0, 1000	5 row(s) returned	0.016 sec / 0.000 sec
9	01:01:46	USE mydb	0 row(s) affected	0.000 sec
10	01:01:46	ALTER TABLE customer DROP COLUMN middle_name	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.044 sec
11	01:01:54	SELECT * FROM mydb.customer LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec

Chapter 5

- Outcome and discussions
- Challenges faced

5.1 Introduction

This chapter will describe and elaborate briefly about the query language used and why the specific commands were used to find out the specific queries alongside the inner joins between multiple tables. The following section will describe each of the queries generated and tested in a comprehensive manner.

5.2 Discussions of the outcome

The queries in this section have been discussed in an orderly fashion.

1. . USE mydb;

```
SELECT customer_id, concat(first_name," ",last_name) AS full_name  
FROM customer;
```

Discussion: the queries use the USE command to declare the particular database , in this case “mydb”. It uses SELECT to identify the columns intended to be shown and/or used which is the customer_id as the base information. IT then uses concat to combine the first_name and the last_name to be showed as FULL_name of the customer with each customer_id from the customer table.

2. USE mydb;

```
SELECT last_name, first_name,store_id,username  
FROM staff  
WHERE last_name LIKE '%SS%'  
ORDER BY last_name, first_name,store_id,Username;
```

Discussion: Here, most of the information are known but only WHERE command is used to display only those last_name which will have SS on their names. Finally, ORDER is used to show the result as the intended orderly fashion.

3. USE mydb;

```
SELECT address, address2, city_id,district
```

```
FROM address
```

```
WHERE district IN ('Dhaka');
```

Discussion: Here, the information is the same except a WHERE command is used to determine that information of the attribute which only resembles the district of Dhaka.

4. USE mydb;

```
SELECT first_name, last_name, SUM(amount)
```

```
FROM staff s
```

```
INNER JOIN payment p
```

```
ON s.staff_id = p.staff_id
```

```
GROUP BY p.staff_id
```

```
ORDER BY last_name ASC;
```

Discussion: Here, the main goal of the query is to show the payment handled by each staff. SUM(amount) is used for the addition of the total amount and INNER JOIN of payment p is used link up the staff_id of the staff table with the staff_id of the payment table. Finally, GROUP BY used to group the outcome of the final result using the staff_id of the payment ASC stands for the “ascending order”.

5. USE mydb;

```
SELECT last_name, first_name, SUM(amount)
```

```
FROM payment p
```

```
INNER JOIN customer c
```

```
ON p.customer_id = c.customer_id
```

```
GROUP BY p.customer_id
```

```
ORDER BY last_name ASC;
```

Discussion: Same as previous, only here the goal is to find out the amount spent by each customer from the payment table. The methods are same as no. 4.

```
6. USE mydb;
UPDATE customer
SET first_name =
CASE
WHEN first_name = 'Tanvir'
THEN 'Samiul'
ELSE 'Samiul Ameen'
END
WHERE customer_id =4;
```

Discussion: Here an entry has been updated on an existing entry using UPDATE on the customer table. SET is used to alter or change the specific column of the table. CASE is used to bring out a condition dynamics and a return value (for example, if then else) if condition is met. WHEN describes the situation of the condition, which in this case WHEN first_name='Tanvir'. THEN only 'samiul' will be the new entry. ELSE is used if it doesn't return the specific intended value, the first_name will be changed to 'Samiul Ameen'. END is used to END the CASE. The customer_id has to be defined using WHERE command.

```
7. USE mydb;

SELECT rating, title, last_update, description
FROM products
WHERE rating='Good';
```

Discussion: Same elements discussed earlier. Here only the information of the product Which only has a rating declared as 'Good' will be showed.

8. USE mydb;

```
SELECT store.store_id, SUM(amount)
```

```
FROM store
```

```
INNER JOIN staff
```

```
ON store.store_id = staff.store_id
```

```
INNER JOIN payment p
```

```
ON p.staff_id = staff.staff_id
```

```
GROUP BY store.store_id
```

```
ORDER BY SUM(amount);
```

Discussion: Here, the main goal was to find out how much money each store brought or earned from the business. INNER JOIN was used on store table to connect with both staff and payment table to display the total amount of money on a GROUP BY basis of store_id of the store table and ORDER BY basis of the highest SUM amount of the corresponding stores.

9. USE mydb;

```
ALTER TABLE customer
```

```
ADD COLUMN middle_name VARCHAR(45);
```

Discussion: Here the ALTER TABLE command is used to alter the column of the table, in this case customer table and ADD COLUMN to add a new column named middle_name on the table. The domain has been declared as VARCHAR(45) meaning alphabets, number and even signs can be used with a limit upto 45 characters.

10. USE mydb;

```
ALTER TABLE customer
```

```
DROP COLUMN middle_name;
```

Discussion: Here the mostly the same type of query is used except that DROP COLUMN command is used to totally drop the intended column specified by the user. Which, in this case the middle_name column.

5.3 Challenges faced

There was a challenge of coming up with the specific attributes as well as table which should be connected with each other in order to create the database. As the relational model is somewhat hypothetical but usable, the main limitations were creating the data model of the company “Hutbazaar” which is a super shop mostly comprised of grocery products. Which means, the tables also have to be specified accordingly to the type of real-life elements which will be implemented on such as customers, staff, stores, inventories, products and many more.

Next, finding out the accurate relationships within the tables was a big challenge as the user has to define whether one attribute of a particular table should influence only one or multiple attribute(s) of another table and vice versa.

Finally , during the population procedure of the database which is mostly hypothetical, but by all means necessary in order to show and describe the query generation and testing methodology, there was an error of foreign key constraints specifically named The MySQL ERROR 1452 which was solved by using a code given below:

```
SET FOREIGN_KEY_CHECKS=0;
```

This solved the problem by disabling the continuous checking of foreign key constraints of one table connected with other tables during the beginning of the population where every row have null value which is the obvious.

Chapter 6

- Conclusions and recommendations

6.1 Conclusions

The research topic was aroused due to the long gap of digitalization of small shops and companies using traditional pen and paper transactional and administrative methods. This led to the initial build up of the hypothetical data model as well as the relational database model of the company “Hutbazaar” super shops. The model is open source which can be used by even the smallest grocery shops with a computer.

MySQL platform enabled the ease of creating, implementing and deploying a data server-based business environment. It created efficiency while also saving a heap load of time. This time management enables or will enable the organizations wanting to use the system to focus and conduct business in an effective manner and focus on the other aspects of the business-like real life physical interaction with the business such as customer management, feedback, employee training, generating new ideas to amplify the business of the organization to the public.

In conclusion, the answer to this experimental research question “What are the properties that contributes a Relational Database Model being advantageous than traditional pen and paper management system” has been identified in the paper with database model creation and implementation. Queries generation with codes and pictures of the implemented code on MySQL Workbench has been showed with outstanding results.

6.2 Recommendations

This paper identified the advantages of using a relational database management system rather than the traditional pen and paper management system. However, during creation of the relational model, there was an issue in building the model where a user must know about the actual proper way of using the specific domains and attributes and no. of tables that should be used. The problem might arise that a user might not know all these procedures.

A future research of simplified data model creation with simplified user interface program can be developed in order to ease out the confusing aspects during creation of

the data model. Also a simplified query user interface that resembles a video game which contains a tutorial at the beginning can be developed and implicated as a stand alone software or extended on the MySQL workbench.

References:

- Al-Mashari, M., Al-Mudimigh, A., & Zairi, M. (2003). Enterprise resource planning: A taxonomy of critical factors. *European journal of operational research*, 146(2), 352-364.
- Arik Ragowsky, T. M. S. (2002). Enterprise resource planning. *Journal of Management Information Systems*, 19(1), 11-15.
- Basharat, I., Azam, F., & Muzaffar, A. W. (2012). Database security and encryption: A survey study. *International Journal of Computer Applications*, 47(12).
- Bertino, E., Dai, C., Lim, H.-S., & Lin, D. (2008). *High-assurance integrity techniques for databases*. Paper presented at the British National Conference on Databases.
- Btoush, E. S., & Hammad, M. M. (2015). Generating ER diagrams from requirement specifications based on natural language processing. *International Journal of Database Theory and Application*, 8(2), 61-70.
- De Witt, D. J. (1979). DIRECT—A multiprocessor organization for supporting relational database management systems. *IEEE Transactions on Computers*, 100(6), 395-406.
- Denning, D. E., Akl, S. G., Heckman, M., Lunt, T. F., Morgenstern, M., Neumann, P. G., & Schell, R. R. (1987). Views for multilevel database security. *IEEE Transactions on Software Engineering*(2), 129-140.
- Egenhofer, M. J. (1994). Spatial SQL: A query and presentation language. *IEEE Transactions on knowledge and data engineering*, 6(1), 86-95.
- Fan, J., Li, G., & Zhou, L. (2011). *Interactive SQL query suggestion: Making databases user-friendly*. Paper presented at the 2011 IEEE 27th International Conference on Data Engineering.
- Grefen, P. W., & Apers, P. M. (1993). Integrity control in relational database systems—an overview. *Data & Knowledge Engineering*, 10(2), 187-223.
- Hayes, A. L., Das, M., Odom, P., & Natarajan, S. (2017). *User friendly automatic construction of background knowledge: Mode construction from er diagrams*. Paper presented at the Proceedings of the Knowledge Capture Conference.
- Hong, Y., & Su, S. Y. (1981). Associative hardware and software techniques for integrity control. *ACM Transactions on Database Systems (TODS)*, 6(3), 416-440.
- Jajodia, S., & Meadows, C. (1995). Inference problems in multilevel secure database management systems. *Information Security: An integrated collection of essays*, 1, 570-584.
- Kim, W. (1982). On optimizing an SQL-like nested query. *ACM Transactions on Database Systems (TODS)*, 7(3), 443-469.
- Li, Y., & Manoharan, S. (2013). *A performance comparison of SQL and NoSQL databases*. Paper presented at the 2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM).
- Lubis, J. H., & Zamzami, E. M. (2020). *Relational database reconstruction from SQL to Entity Relational Diagrams*. Paper presented at the Journal of Physics: Conference Series.

- Lunt, T. F., & Fernandez, E. B. (1990). Database security. *ACM Sigmod Record*, 19(4), 90-97.
- Malik, M., & Patel, T. (2016). Database security-attacks and control methods. *International Journal of Information*, 6(1/2), 175-183.
- Moon, Y. B. (2007). Enterprise Resource Planning (ERP): a review of the literature. *International journal of management and enterprise development*, 4(3), 235-264.
- Murray, M. C. (2010). Database security: What students need to know. *Journal of information technology education: Innovations in practice*, 9, IIP-61.
- Patil, M. M., Hanni, A., Tejeshwar, C., & Patil, P. (2017). *A qualitative analysis of the performance of MongoDB vs MySQL database based on insertion and retrieval operations using a web/android application to explore load balancing—Sharding in MongoDB and its advantages*. Paper presented at the 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC).
- Pavlo, A., Angulo, G., Arulraj, J., Lin, H., Lin, J., Ma, L., . . . Quah, I. (2017). *Self-Driving Database Management Systems*. Paper presented at the CIDR.
- Reiter, R. (1992). What should a database know? *The Journal of Logic Programming*, 14(1-2), 127-153.
- Shehab, E., Sharp, M., Supramaniam, L., & Spedding, T. A. (2004). Enterprise resource planning: An integrative review. *Business process management journal*.
- Sumathi, S., & Esakkirajan, S. (2007). *Fundamentals of relational database management systems* (Vol. 47): Springer.
- Thomas, P., Waugh, K., & Smith, N. (2006). *Using patterns in the automatic marking of ER-diagrams*. Paper presented at the Proceedings of the 11th annual SIGCSE conference on Innovation and technology in computer science education.
- Vakaliuk, T., Korotun, O., & Semerikov, S. (2020). *The selection of cloud services for ER-diagrams construction in IT specialists databases teaching*.