

|  |  |
| --- | --- |
| **ASSIGNMENT** | |
| **Course Code** | 19CSC302A |
| **Course Name** | Database Systems |
| **Programme** | B. Tech. |
| **Department** | Computer Science and Engineering |
| **Faculty** | Faculty of Engineering & Technology |

#### 

|  |  |
| --- | --- |
| **Name of the Student** | Subhendu Maji |
| **Reg. No** | 18ETCS002121 |
| **Semester/Year** | 5TH semester / 2018 batch |
| **Course Leader/s** | A. Prabhakar |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Declaration Sheet** | | | | | | | | |
| Student Name | Subhendu Maji | | | | | | | |
| Reg. No | 18ETCS002121 | | | | | | | |
| Programme | B. Tech. | | | | | Semester/Year | 5th sem / 2018 batch | |
| Course Code | 19CSC302A | | | | | | | |
| Course Title | Database Systems | | | | | | | |
| Course Date |  | | to | |  | | | |
| Course Leader | A. Prabhakar | | | | | | | |
| **Declaration**  The assignment submitted herewith is a result of my own investigations and that I have conformed to the guidelines against plagiarism as laid out in the Student Handbook. All sections of the text and results, which have been obtained from other sources, are fully referenced. I understand that cheating and plagiarism constitute a breach of University regulations and will be dealt with accordingly. | | | | | | | | |
| Signature of the Student | |  | | | | | Date |  |
| Submission date stamp  (by Examination & Assessment Section) | |  | | | | | | |
| Signature of the Course Leader and date | | | | Signature of the Reviewer and date | | | | |
|  | | | |  | | | | |

# **Contents**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

[**Declaration Sheet** ii](#_Toc59142077)

[**Contents** iii](#_Toc59142078)

[Marking Scheme 4](#_Toc59142079)

[**Question No. 1** 5](#_Toc59142080)

[1.1 Merits and demerits of relational and graph databases 5](#_Toc59142081)

[1.2 Justification of the stance taken and conclusion 6](#_Toc59142082)

[**Question No. 2** 8](#_Toc59142083)

[2.1 List of functional and data requirements 8](#_Toc59142084)

[2.2 Discussion on the entities, attributes, and relationships 11](#_Toc59142085)

[2.3 ER diagram 13](#_Toc59142086)

[2.4 Identification of any requirement that is not possible to model using ER diagram 13](#_Toc59142087)

[**Question No. 3** 14](#_Toc59142088)

[3.1 Design of database schema 14](#_Toc59142089)

[3.2 Discussion on the constraints 15](#_Toc59142090)

[3.3 Implementation using SQL commands 17](#_Toc59142091)

[3.4 Update operations violating the schema constraints 24](#_Toc59142092)

[Bibliography 25](#_Toc59142093)

# 

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Assignment - 01 | | | | | | | | |
| Register No. | | | 18ETCS002121 | Name of Student | | Subhendu maji | | |
| Sections |  | Marking Scheme | | | Max Marks | | First Examiner Marks | Second Examiner Marks |
| Part A | A.1 | Merits and demerits of relational and graph databases | | | 02 | |  |  |
| A.2 | Justification of the stance taken and conclusion | | | 03 | |  |  |
|  | **Part-A Max Marks** | | | **05** | |  |  |
| Part B1 | B1.1 | List of functional and data requirements | | | 03 | |  |  |
| B1.2 | Discussion on the entities, attributes, and relationships | | | 02 | |  |  |
| B1.3 | ER diagram | | | 02 | |  |  |
| B1.4 | Identification of any requirement that is not possible to model using ER diagram | | | 03 | |  |  |
|  | **B1 Max Marks** | | | **10** | |  |  |
| Part B2 | B2.1 | Design of database schema | | | 03 | |  |  |
| B2.2 | Discussion on the constraints | | | 02 | |  |  |
| B2.3 | Implementation using SQL commands | | | 02 | |  |  |
| B2.4 | Update operations violating the schema constraints | | | 03 | |  |  |
|  | **B2 Max Marks** | | | **10** | |  |  |
|  | **Total Assignment Marks** | | | | **25** | |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Course Marks Tabulation** | | | | |
| **Component- 1(B)Assignment** | **First Examiner** | **Remarks** | **Second Examiner** | **Remarks** |
| A |  |  |  |  |
| **Marks (out of 10)** |  |  |  |  |
| Signature of First ExaminerSignature of Second Examiner | | | | |

# **Question No. 1**

**Solution to Question No. 1:**

## Merits and demerits of relational and graph databases

**Relational Database**

**Advantages:**

1. Ease of use: The revision of any information as tables consisting of rows and columns is much easier to understand.

2. Flexibility: Different tables from which information has to be linked and extracted can be easily manipulated by operators such as project and join to give information in the form in which it is desired.

3. Precision: The usage of relational algebra and relational calculus in the manipulation of the relations between the tables ensures that there is no ambiguity, which may otherwise arise in establishing the linkages in a complicated network type database.

4. Security: Security control and authorization can also be implemented more easily by moving sensitive attributes in a given table into a separate relation with its own authorization controls. If authorization requirement permits, a particular attribute could be joined back with others to enable full information retrieval.

5. Data Independence: Data independence is achieved more easily with normalization structure used in a relational database than in the more complicated tree or network structure.

6. Data Manipulation Language: The possibility of responding to query by means of a language based on relational algebra and relational calculus e.g SQL is easy in the relational database approach. For data organized in other structure the query language either becomes complex or extremely limited in its capabilities.

**Disadvantages:**

1. Performance: A major constraint and therefore disadvantage in the use of relational database system is machine performance. If the number of tables between which relationships to be established are large and the tables themselves effect the performance in responding to the sql queries.

2. Physical Storage Consumption: With an interactive system, for example an operation like join would depend upon the physical storage also. It is, therefore common in relational databases to tune the databases and in such a case the physical data layout would be chosen so as to give good performance in the most frequently run operations. It therefore would naturally result in the fact that the lays frequently run operations would tend to become even more shared.

3. Slow extraction of meaning from data: if the data is naturally organized in a hierarchical manner and stored as such, the hierarchical approach may give quick meaning for that data.

**Graph Database**

**Advantages:**

* Processing unstructured data. Graph databases can save a complex object in one operation
* Simple query language and visual result.
* Ease of change due to the flexible data structure.
* A large increase in productivity when working with interrelated data, compared with relational databases

**Disadvantages**

* Performance is lower on simple queries.
* Limited functionality and customization options
* The complexity of the search for specialists working with graph databases.

## Justification of the stance taken and conclusion

A graph database is a data management system software. The building blocks are vertices and edges. To put it in a more familiar context, a relational database is also a data management software in which the building blocks are tables. Both require loading data into the software and using a query language or APIs to access the data.

Relational databases boomed in the 1980s. Many commercial companies (i.e. Oracle, Ingres, IBM) backed the relational model (tabular organization) of data management. In that era, the main data management need was to generate reports.

Graph databases didn't see a greater advantage over relational databases until recent years, when frequent schema changes, managing explosives volume of data, real-time query response time, and more intelligent data activation requirements make people realize the advantages of the graph model.

Hence, I think graph databases can replace relational database technologies.

* Relational database is having tables with lots of columns and a few of these columns are used by rows. Data can have lots of very different attributes and only a few of them can be meaningful for some data items. In contrast to relational database, graph database, stores only meaningful attributes for the related node and adding data for only used attributes for the related node increase efficiency
* Relational databases are more mature and secure as compared to graph databases, but its schema is fixed, which makes it difficult to extend these databases and less suitable to manage schemas that evolve over time.
* Another criterion is that relational database requires a predefined schema before adding any data to the system while graph database provide adding data to the system without needing any predefined schema.
* The graph databases retrieve the results of the set of predefines query faster than relation databases.
* Graph database is a very powerful tool to annotate resource and create data models for the repositories of the different types of resources.

**Conclusion**

In conclusion, we see many advantages of native graph databases managing big data that cannot be worked around by traditional relational databases. However, as any new technology is replacing old technology, there are still obstacles in adopting graph databases. One is that there are fewer qualified developers in the job market than the SQL developers. Another is the non-standardization of the graph database query language. There's been a lot of marketing hype and incomplete offerings that have led to subpar performance and subpar usability, which slows down graph model adoption in the needed enterprises.

In the phase of deciding which database model is most suitable for specific domain, data should be investigated by considering basic criteria. if data has lots of many-to-many relationships, using graph model can be very efficient. Graph database traverse data very efficiently by using relationship entities while relational database traverse database has to use many complex and expensive join operations.

# **Question No. 2**

**Solution to Question No. 2:**

## 2.1 List of functional and data requirements

Functional Requirements

|  |  |
| --- | --- |
| Requirement Tag | FR1 |
| Requirement Description | the system should allow new users to register and existing users to login |
| Dependent on | DR1, DR2 |
| User/System interacting with the requirement | user, admin |

|  |  |
| --- | --- |
| Requirement Tag | FR2 |
| Requirement Description | the system should allow users to search furniture by category. |
| Dependent on | DR4 |
| User/System interacting with the requirement | User |

|  |  |
| --- | --- |
| Requirement Tag | FR3 |
| Requirement Description | the software should display furniture description |
| Dependent on | DR6, DR4 |
| User/System interacting with the requirement | user |

|  |  |
| --- | --- |
| Requirement Tag | FR4 |
| Requirement Description | the system should allow user to see the stock available for the furniture |
| Dependent on | DR4, FR3 |
| User/System interacting with the requirement | User,admin |

|  |  |
| --- | --- |
| Requirement Tag | FR5 |
| Requirement Description | the system should allow users to buy multiple furniture at a time by adding to cart. |
| Dependent on | DR5, FR1 |
| User/System interacting with the requirement | user |

|  |  |
| --- | --- |
| Requirement Tag | FR6 |
| Requirement Description | the system should allow users to buy furniture directly without adding to cart |
| Dependent on | DR1, DR2, FR1 |
| User/System interacting with the requirement | user |

|  |  |
| --- | --- |
| Requirement Tag | FR7 |
| Requirement Description | the system should allow admin to add or update furniture details. |
| Dependent on | DR1, DR2 |
| User/System interacting with the requirement | admin |

Data Requirements:

|  |  |
| --- | --- |
| Requirement Tag | DR1 |
| Item Name | Login\_id |
| Item type | integer |
| Item Description (Where/How used) | To be used while login as username by user/admin |
| User/System interacting with the requirement | User, admin |

|  |  |
| --- | --- |
| Requirement Tag | DR2 |
| Item Name | password |
| Item type | char |
| Item Description (Where/How used) | To authenticate user/admin |
| User/System interacting with the requirement | User, admin |

|  |  |
| --- | --- |
| Requirement Tag | DR3 |
| Item Name | address |
| Item type | char |
| Item Description (Where/How used) | To deliver the order |
| User/System interacting with the requirement | user |

|  |  |
| --- | --- |
| Requirement Tag | DR4 |
| Item Name | Category\_name |
| Item type | char |
| Item Description (Where/How used) | To search the product from categories |
| User/System interacting with the requirement | user |

|  |  |
| --- | --- |
| Requirement Tag | DR5 |
| Item Name | quantity |
| Item type | integer |
| Item Description (Where/How used) | To order the quantity of a product to be ordered |
| User/System interacting with the requirement | user |

|  |  |
| --- | --- |
| Requirement Tag | DR6 |
| Item Name | Product\_name |
| Item type | char |
| Item Description (Where/How used) | To search a specific product |
| User/System interacting with the requirement | user |

## 2.2 Discussion on the entities, attributes, and relationships

* **Entity**: a thing with distinct and independent existence.
* **Attributes**: these are properties of entity.
* **Relationship**: the number of occurrences in one entity that is associated with the number of occurrences in another entity

|  |  |
| --- | --- |
| Entities of online furniture shopping system | |
| CUSTOMER | It is a strong entity. It is a person who deals with the system and who select the products and buy the products and interact with entire online Furniture system |
| PRODUCT | It is a strong entity. It is the item or thing that the customer buys |
| PAYMENTS | It is a strong entity; it associates a user to the product or cart he has purchased |
| CART | It is a weak entity. A cart only exists if a user exists, because each user has his own cart |
| CATOGERIES | It is a strong entity. The products are classified into different categories for the user to search |
| ADMIN | It is a strong entity. Admin manages the products and categories. |

|  |  |
| --- | --- |
| Relationship of online furniture shopping system | |
| Category\_contains | Binary relationship between category and product entity. It is n:n relationship, because n products can have n categories |
| Searches | It is a binary relationship between customer and categories, it is a 1:n relationship, because a customer can search for n categories |
| Buys | It is ternary relationship between customer, product and payment. It is n:n:1, relationship, as n customers can buy n products and make 1 payment for it |
| Makes\_payment | It’s a binary relationship between, a customer and payment. It’s a 1:n relationship, as 1 customer can make n payments |
| Owns | It’s a weak relationship between a user and a cart. This means if the user exists, then he owns a cart, it’s a 1:1 relationship |
| Cart\_contains | It’s a weak relationship between a cart and a product. If a cart exists, then it can contain multiple products, 1: n relation |
| Payment\_for\_cart | It’s a weak relationship between a car and a payment. If a cart exists, then a payment can be made for it, 1:1 relation |
| Manages\_product | Binary relationship between admin and product entity. It is n:n relationship, because n admins can manage n products |
| Manages\_admin | Binary relationship between admin and category entity. It is n: n relationship, because n admins can manage n categories |

|  |  |
| --- | --- |
| Attributes of online furniture shopping system | |
| CUSTOMER | **name** – name of the customer  **email** – email id of the customer, it has to be unique  **password** – password of the user, required to login or signup  **address** – the delivery address of the user  **user\_id** – the user\_id of the user, also used as the login id. It is used to uniquely identify each user (primary key)  **phone\_no** – the users contact number |
| PRODUCT | **name** – name of the product  **material** – material that the product is made-up of  **color** – the color of the product  **product\_id** – the primary key used to uniquely id the product  **price** – the cost of a single unit of the product  **inventory** – the current stock or the number of units to be sold |
| PAYMENTS | **payment\_id** – the primary key used to uniquely identify each transaction  **user\_id** – the foreign key used to id the user associated to the payment |
| CART | **quantity** – the units of product ordered  **payment\_id** – the payment associated with the cart  **user\_id** – the user that owns the cart  **product\_id** – the product that is added to the cart |
| CATOGERIES | **category\_title** – the title of the category  **category \_id** – the primary key of each category |
| ADMIN | **email** – the admins email address  **password** – the password used by the admin to login or sign-up  **admin \_id** – the primary key or the login id of the admin |

## 2.3 ER diagram

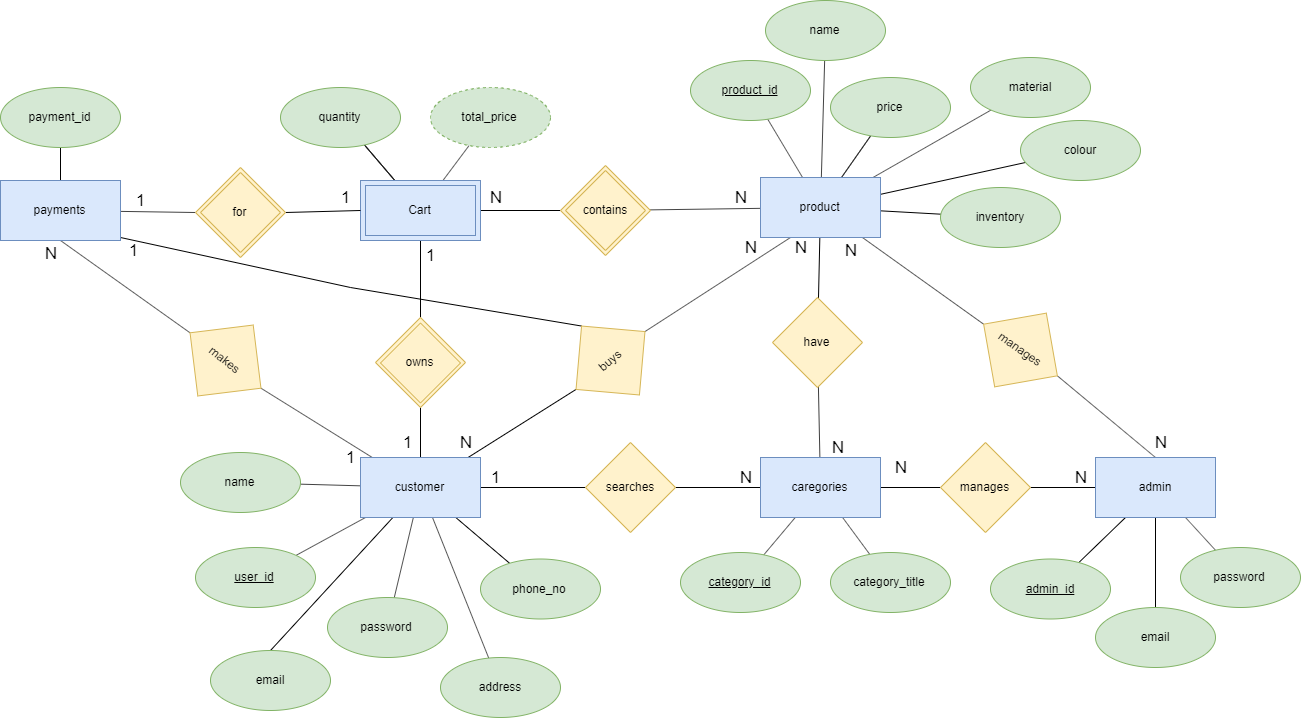


Figure 1 ER diagram

## 2.4 Identification of any requirement that is not possible to model using ER diagram

Identified problem is the application will not show if the product is out of stock and it will show an error hence it can be resolved using the conceptual data model, since data model is a model focuses on identifying the data used in the business but not its processing flow or physical characteristics.

# **Question No. 3**

**Solution to Question No. 3:**

## 3.1 Design of database schema

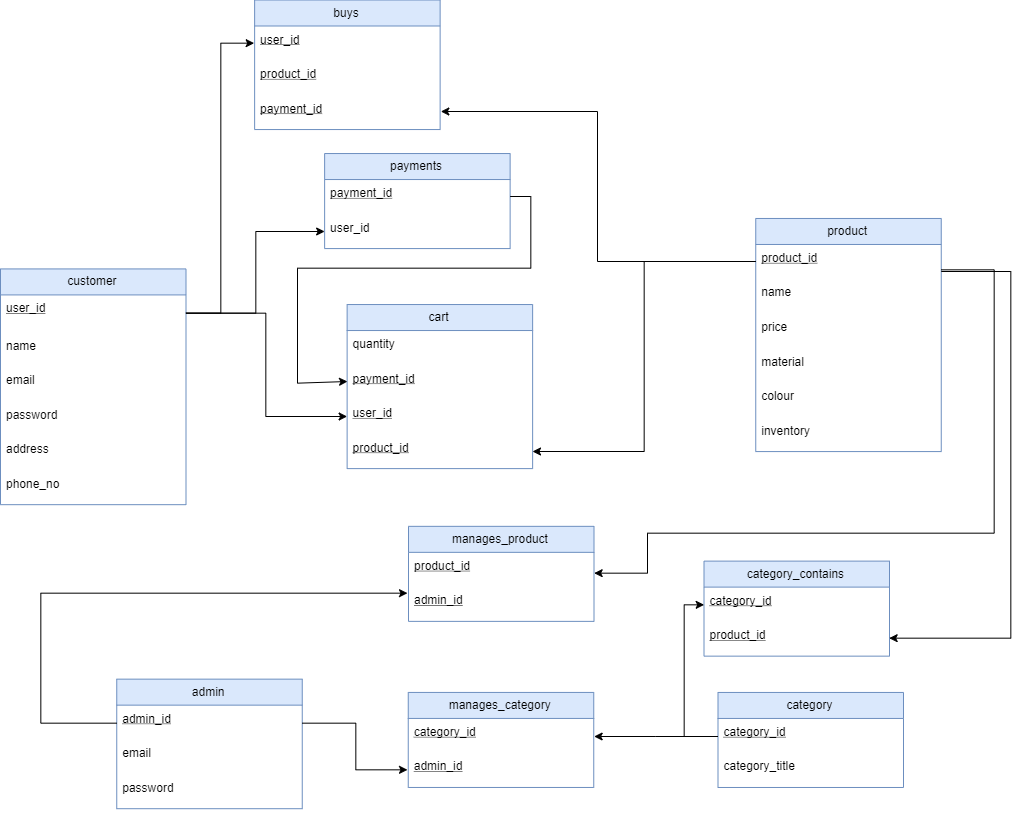


Figure 2 Database Schema

## 3.2 Discussion on the constraints

|  |  |
| --- | --- |
| Customer table | |
| Domain constraint | name, email, password, address – char  user\_id, phone\_no – int |
| Entity Integrity Constraint | user\_id, password cannot be null, required for login  address, phone\_no cannot be null, required for delivery |
| Relational Integrity Constraint | - |
| Key Constraint | user\_id, is primary key so it must be unique  email, phone\_no, must be unique |

|  |  |
| --- | --- |
| Product table | |
| Domain constraint | name, material, color – char  product\_id, price, inventory – int |
| Entity Integrity Constraint | product\_id cannot be null, primary key  name, price, inventory, cannot be null required to place order |
| Relational Integrity Constraint | - |
| Key Constraint | product\_id, is primary key so it must be unique |

|  |  |
| --- | --- |
| Category table | |
| Domain constraint | category\_title – char  category \_id – int |
| Entity Integrity Constraint | category \_id cannot be null, primary key  category\_title, cannot be null required to search |
| Relational Integrity Constraint | - |
| Key Constraint | category \_id, is primary key so it must be unique  category\_title, must be unique, used for search |

|  |  |
| --- | --- |
| Admin table | |
| Domain constraint | email, password– char  admin \_id – int |
| Entity Integrity Constraint | admin\_id, password cannot be null, required for login |
| Relational Integrity Constraint | - |
| Key Constraint | admin\_id, is primary key so it must be unique  email, must be unique |

|  |  |
| --- | --- |
| Payments table | |
| Domain constraint | payment\_id, user\_id – int |
| Entity Integrity Constraint | payment\_id, user\_id, cannot be null |
| Relational Integrity Constraint | user\_id are foreign ids |
| Key Constraint | payment \_id, is primary key so it must be unique |

|  |  |
| --- | --- |
| Cart table | |
| Domain constraint | quantity, payment\_id, user\_id, product\_id – int |
| Entity Integrity Constraint | quantity, payment\_id, user\_id, product\_id cannot be null |
| Relational Integrity Constraint | payment\_id, user\_id, product\_id are foreign ids |
| Key Constraint | product\_id is unique |

|  |  |
| --- | --- |
| Buys table | |
| Domain constraint | product\_id, user\_id, payment\_id – int |
| Entity Integrity Constraint | product\_id, user\_id, payment\_id cannot be null |
| Relational Integrity Constraint | product\_id, user\_id, payment\_id are foreign ids |
| Key Constraint | payment\_id is unique |

|  |  |
| --- | --- |
| category\_contains table | |
| Domain constraint | product\_id, category \_id – int |
| Entity Integrity Constraint | product\_id, category \_id, cannot be null |
| Relational Integrity Constraint | product\_id, category \_id are foreign ids |
| Key Constraint | - |

|  |  |
| --- | --- |
| Manages\_category table | |
| Domain constraint | admin\_id, category \_id – int |
| Entity Integrity Constraint | admin \_id, category \_id, cannot be null |
| Relational Integrity Constraint | admin \_id, category \_id are foreign ids |
| Key Constraint | - |

|  |  |
| --- | --- |
| Manages\_products table | |
| Domain constraint | admin\_id, product\_id – int |
| Entity Integrity Constraint | admin \_id, product \_id, cannot be null |
| Relational Integrity Constraint | admin \_id, product \_id are foreign ids |
| Key Constraint | - |

## 3.3 Implementation using SQL commands

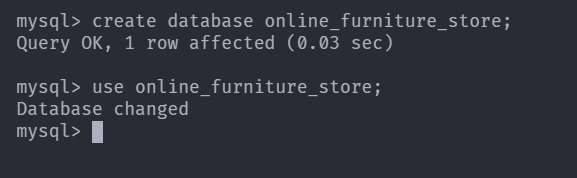


Figure 3 creating and using database in MySQL server

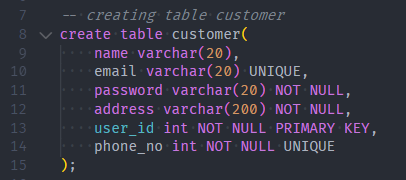


Figure 4 creating Customer table with its constraints

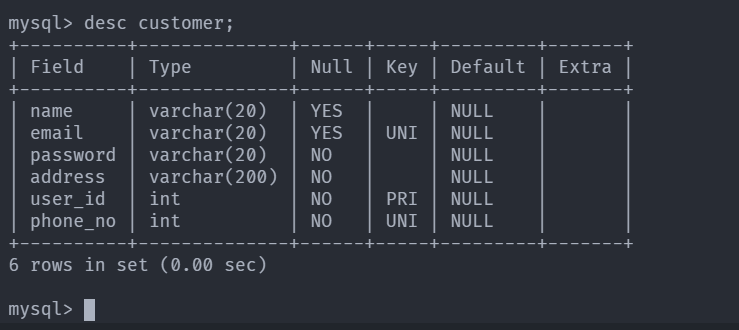


Figure 5 describe customer table

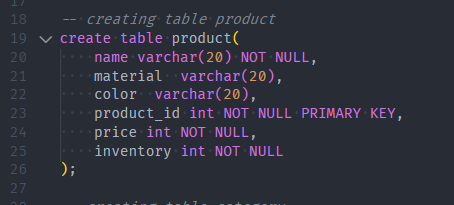


Figure 6 creating product table with its constraints

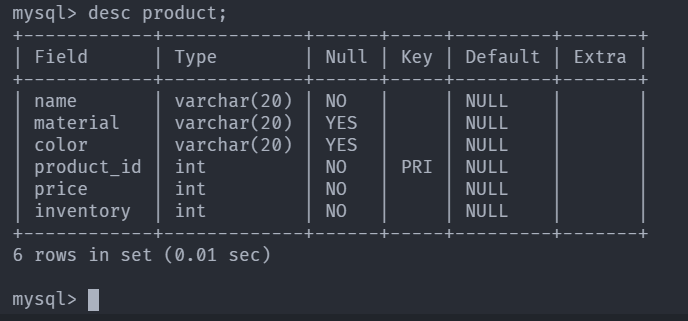


Figure 7 describe product table

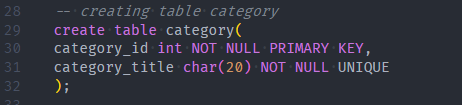


Figure 8 creating category table with its constraints

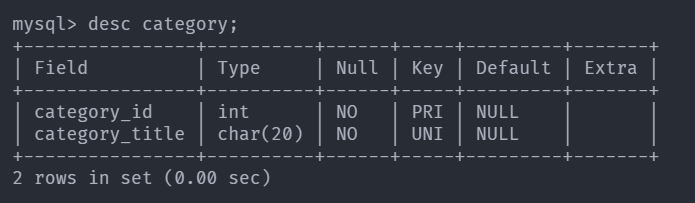


Figure 9 describe category table

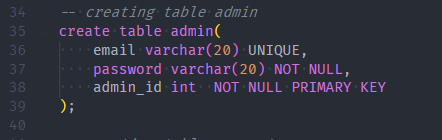


Figure 10 creating admin table with its constraints

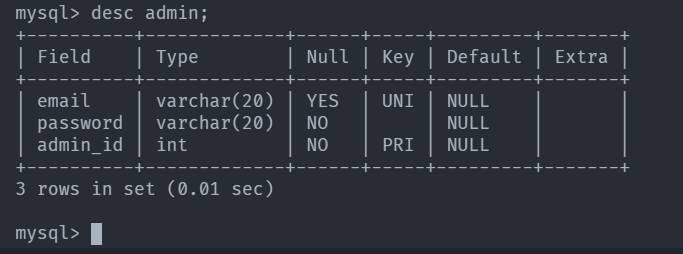


Figure 11 describe admin table

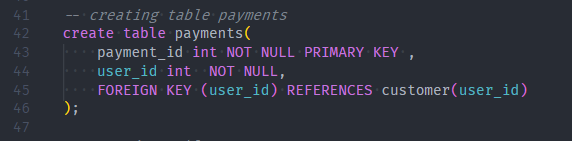


Figure 12 creating payments table with its constraints

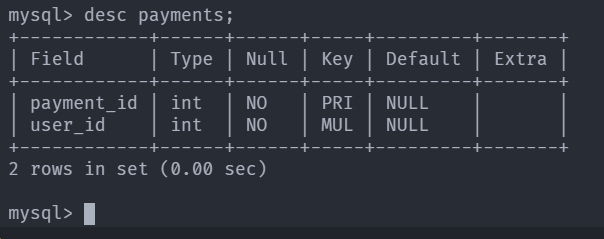


Figure 13 describe payments table

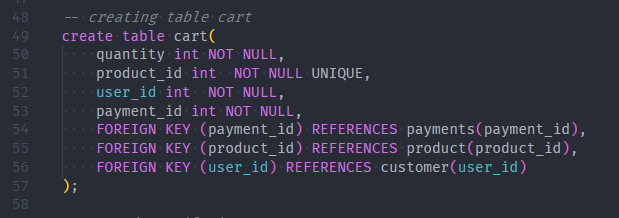


Figure 14 creating cart table with its constraints

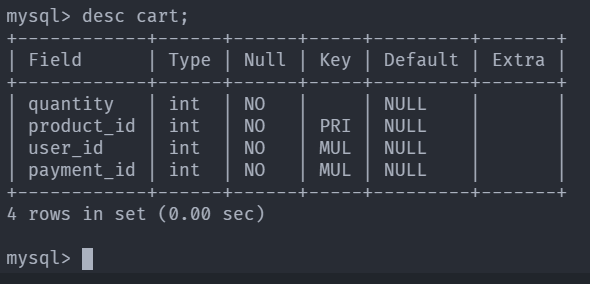


Figure 15 describe cart table

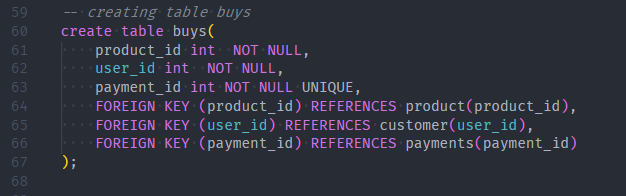


Figure 16 creating buys table with its constraints

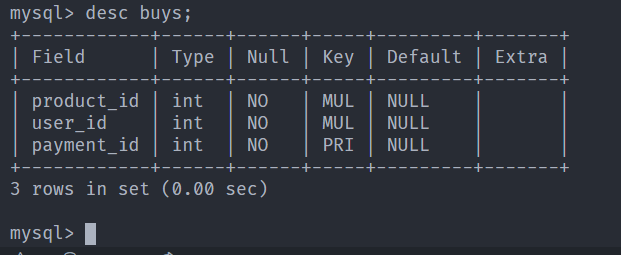


Figure 17 describe buys table

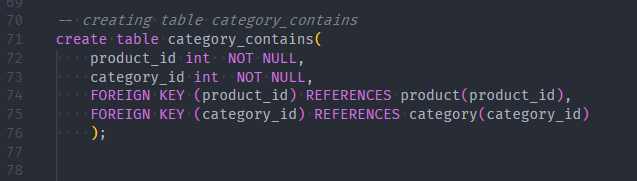


Figure 18 creating category\_contains table

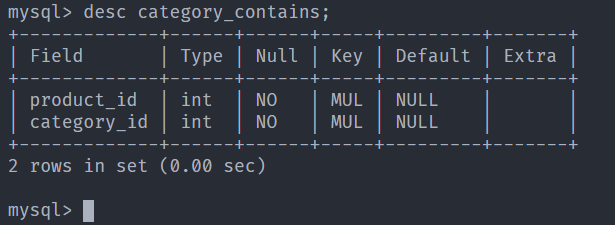


Figure 19 describe category contains

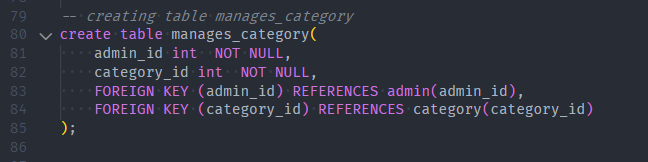


Figure 20 creating manages\_category with its constraints

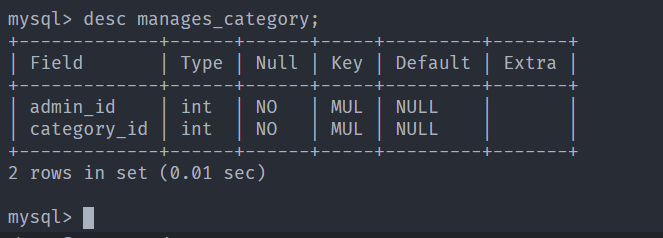


Figure 21 describe manages\_category table

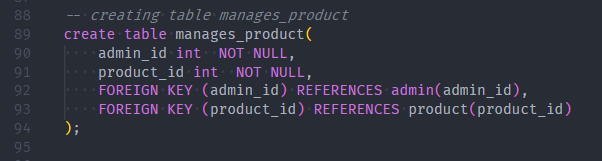


Figure 22 creating manages\_product with its constraints

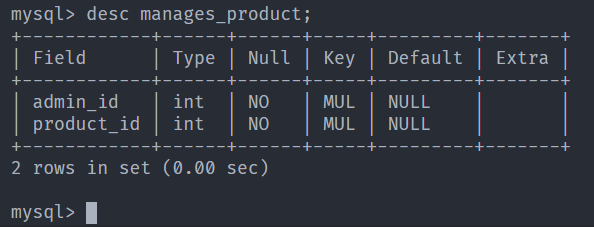


Figure 23 describe manages\_product



Figure 24 adding dummy values in all the tables

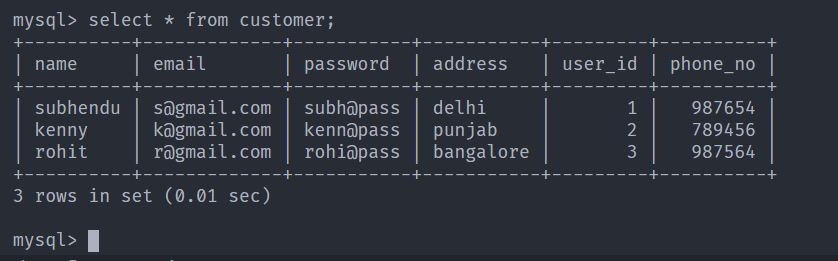


Figure 25 viewing customer table

## 3.4 Update operations violating the schema constraints

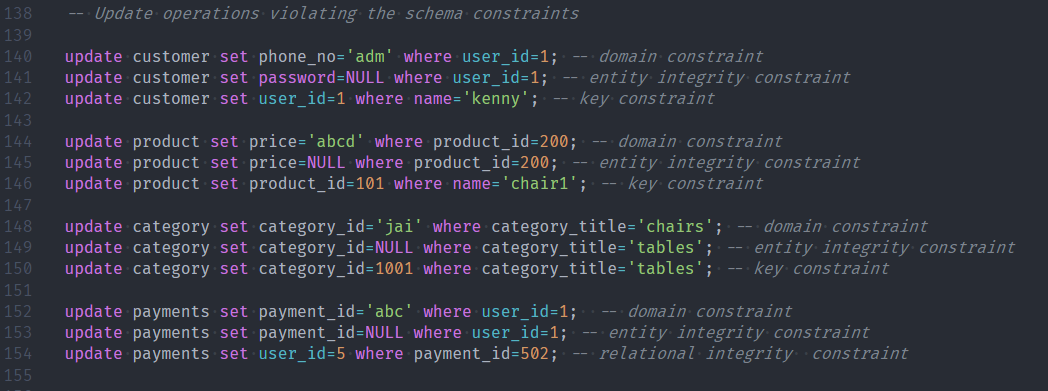


Figure 26 Examples of update commands which are violating schema constraints

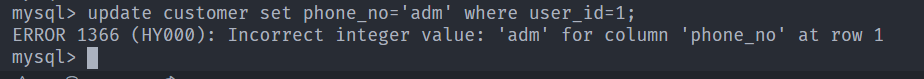


Figure 27 example of violating domain constraint (ERROR)

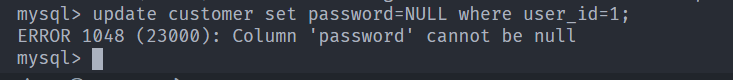


Figure 28 example of violating entity integrity constraint in customer table (ERROR)

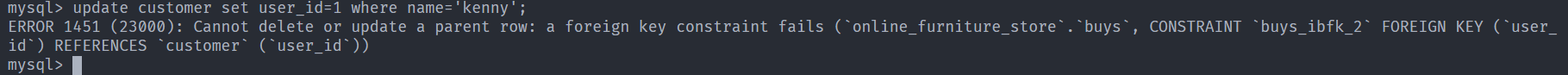


Figure 29 example of violating key constraint in customer table (ERROR)

# Bibliography

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. <https://neo4j.com/developer/graph-db-vs-rdbms/>
2. <https://medium.com/@mtbuzzerseo/graph-database-vs-relational-database-e5798281f6ef>