RAMAIAH UNIVERSITY OF APPLIED SCIENCES

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

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:

1.1 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.

1.2 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

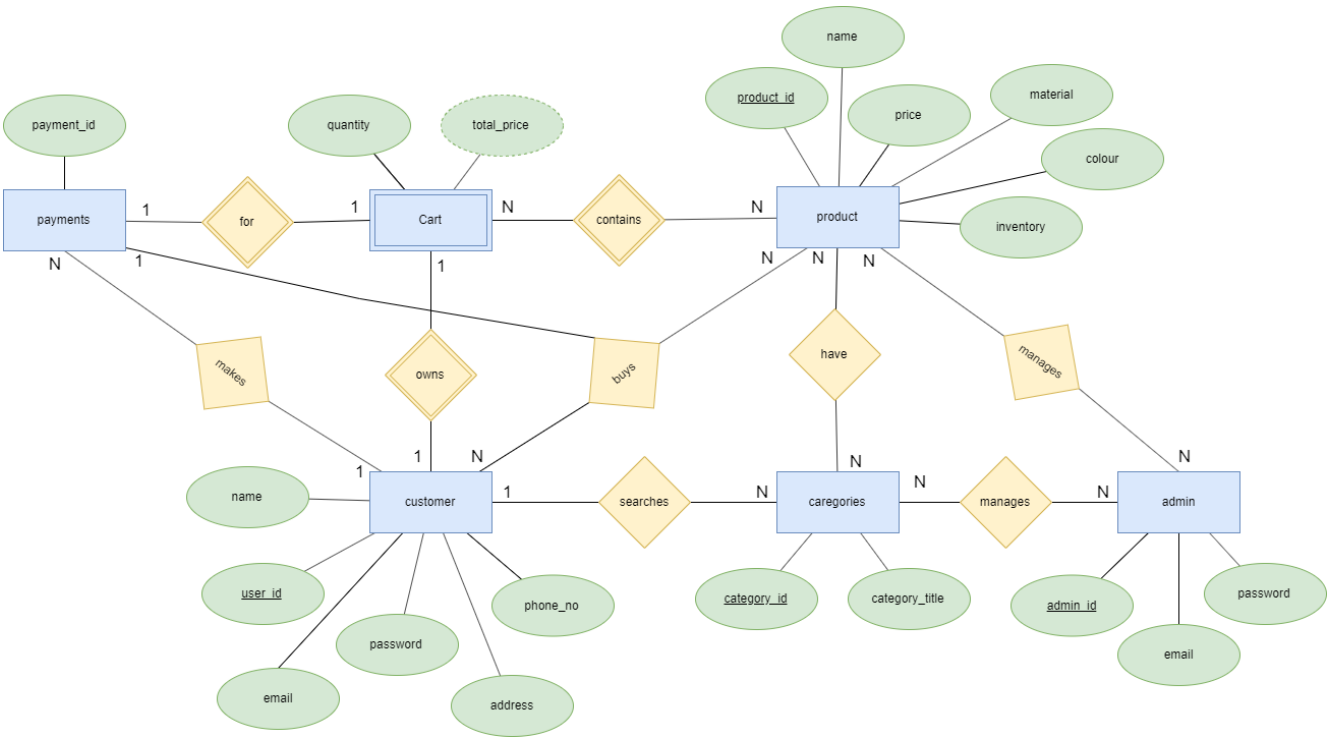


Figure 1 ER diagram

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

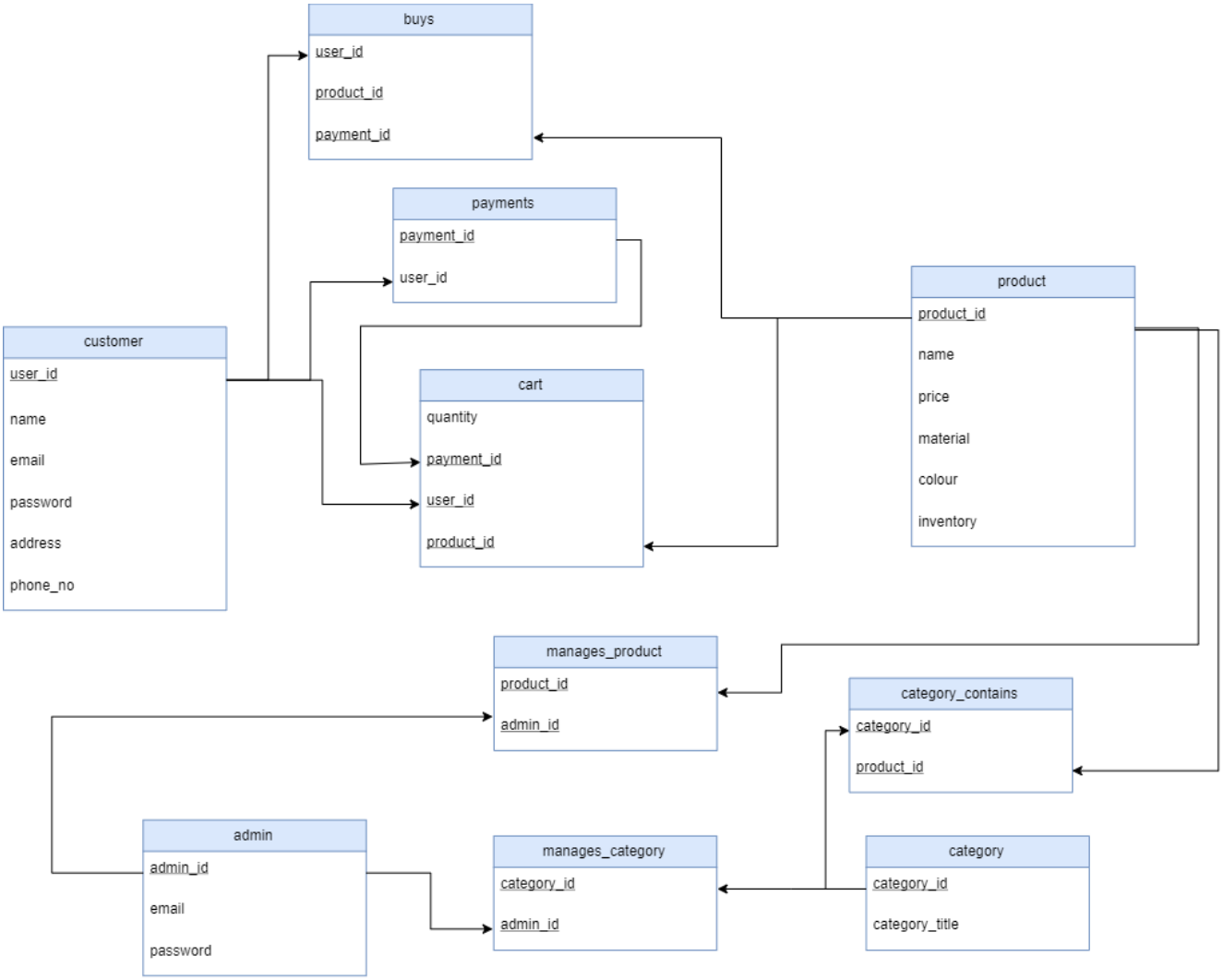


Figure 2 Database Schema

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

```
mysql> create database online_furniture_store;
Query OK, 1 row affected (0.03 sec)

mysql> use online_furniture_store;
Database changed
mysql>
```

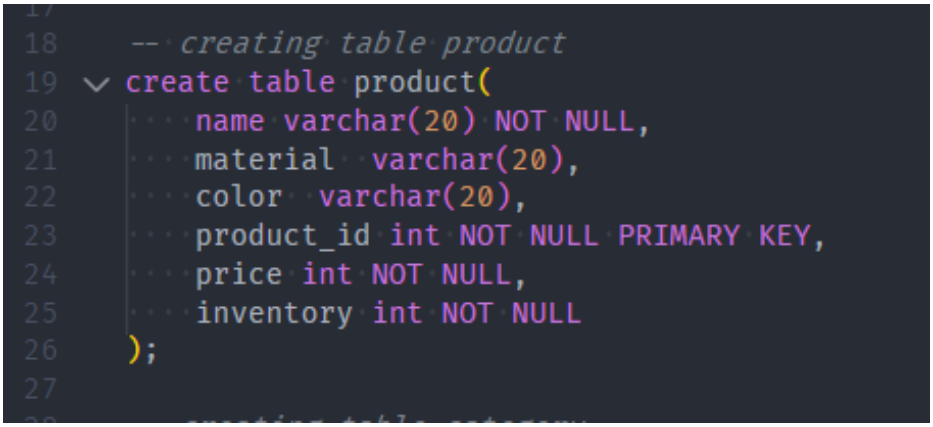
Figure 3 creating and using database in MySQL server

```
-- creating table customer
create table customer(
    name varchar(20),
    email varchar(20) UNIQUE,
    password varchar(20) NOT NULL,
    address varchar(200) NOT NULL,
    user_id int NOT NULL PRIMARY KEY,
    phone_no int NOT NULL UNIQUE
);
```

Figure 4 creating Customer table with its constraints

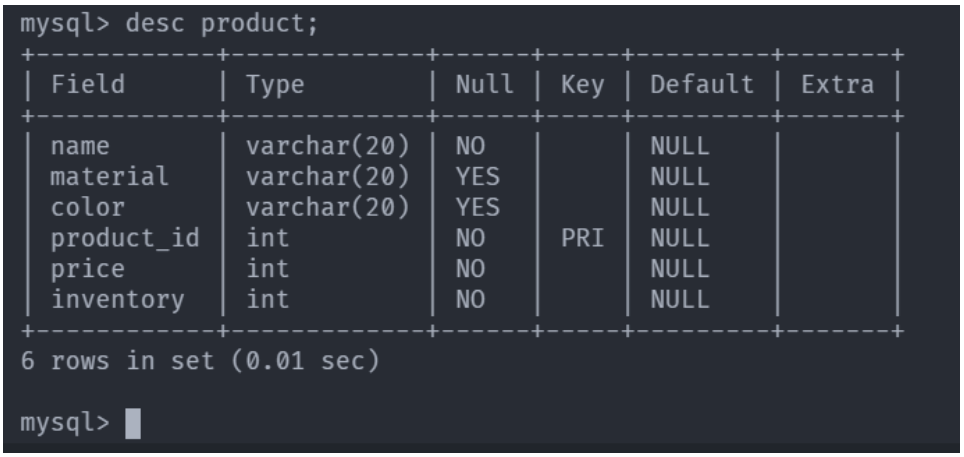
```
mysql> desc customer;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| name     | varchar(20)  | YES  |     | NULL    |       |
| email    | varchar(20)  | YES  | UNI | NULL    |       |
| password | varchar(20)  | NO   |     | NULL    |       |
| address  | varchar(200) | NO   |     | NULL    |       |
| user_id  | int          | NO   | PRI | NULL    |       |
| phone_no | int          | NO   | UNI | NULL    |       |
+----------+--------------+------+-----+---------+-------+
6 rows in set (0.00 sec)

mysql>
```

Figure 5 describe customer table



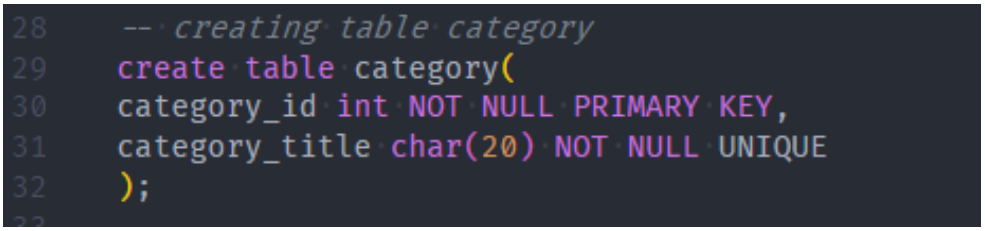
```
-- creating table product
create table product(
    name varchar(20) NOT NULL,
    material varchar(20),
    color varchar(20),
    product_id int NOT NULL PRIMARY KEY,
    price int NOT NULL,
    inventory int NOT NULL
);
```

Figure 6 creating product table with its constraints



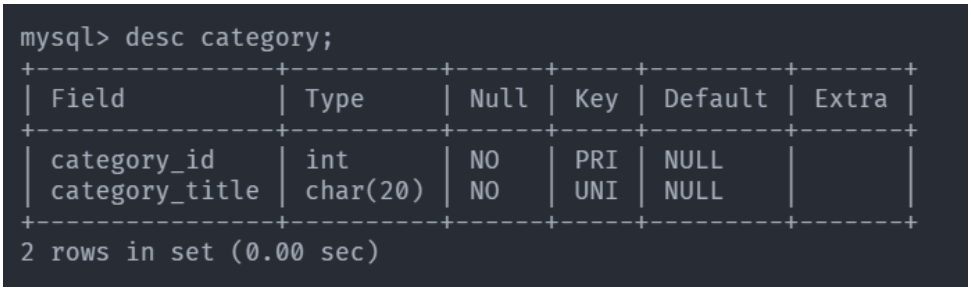
```
mysql> desc product;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| name       | varchar(20) | NO   |     | NULL    |       |
| material   | varchar(20) | YES  |     | NULL    |       |
| color      | varchar(20) | YES  |     | NULL    |       |
| product_id | int         | NO   | PRI | NULL    |       |
| price      | int         | NO   |     | NULL    |       |
| inventory  | int         | NO   |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
6 rows in set (0.01 sec)

mysql>
```

Figure 7 describe product table



```
-- creating table category
create table category(
category_id int NOT NULL PRIMARY KEY,
category_title char(20) NOT NULL UNIQUE
);
```

Figure 8 creating category table with its constraints



```
mysql> desc category;
+----------------+----------+------+-----+---------+-------+
| Field          | Type     | Null | Key | Default | Extra |
+----------------+----------+------+-----+---------+-------+
| category_id    | int      | NO   | PRI | NULL    |       |
| category_title | char(20) | NO   | UNI | NULL    |       |
+----------------+----------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Figure 9 describe category table

```sql
-- creating table admin
create table admin(
    email varchar(20) UNIQUE,
    password varchar(20) NOT NULL,
    admin_id int  NOT NULL PRIMARY KEY
);

```

Figure 10 creating admin table with its constraints

```
mysql> desc admin;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| email    | varchar(20) | YES  | UNI | NULL    |       |
| password | varchar(20) | NO   |     | NULL    |       |
| admin_id | int         | NO   | PRI | NULL    |       |
+----------+-------------+------+-----+---------+-------+
3 rows in set (0.01 sec)

mysql>
```

Figure 11 describe admin table

```sql
-- creating table payments
create table payments(
    payment_id int NOT NULL PRIMARY KEY ,
    user_id int  NOT NULL,
    FOREIGN KEY (user_id) REFERENCES customer(user_id)
);

```

Figure 12 creating payments table with its constraints

```
mysql> desc payments;
+------------+------+------+-----+---------+-------+
| Field      | Type | Null | Key | Default | Extra |
+------------+------+------+-----+---------+-------+
| payment_id | int  | NO   | PRI | NULL    |       |
| user_id    | int  | NO   | MUL | NULL    |       |
+------------+------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql>
```

Figure 13 describe payments table

```sql
-- creating table cart
create table cart(
    quantity int NOT NULL,
    product_id int  NOT NULL UNIQUE,
    user_id int  NOT NULL,
    payment_id int NOT NULL,
    FOREIGN KEY (payment_id) REFERENCES payments(payment_id),
    FOREIGN KEY (product_id) REFERENCES product(product_id),
    FOREIGN KEY (user_id) REFERENCES customer(user_id)
);
```

Figure 14 creating cart table with its constraints

```
mysql> desc cart;
+------------+------+------+-----+---------+-------+
| Field      | Type | Null | Key | Default | Extra |
+------------+------+------+-----+---------+-------+
| quantity   | int  | NO   |     | NULL    |       |
| product_id | int  | NO   | PRI | NULL    |       |
| user_id    | int  | NO   | MUL | NULL    |       |
| payment_id | int  | NO   | MUL | NULL    |       |
+------------+------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql>
```

Figure 15 describe cart table

```sql
-- creating table buys
create table buys(
    product_id int  NOT NULL,
    user_id int  NOT NULL,
    payment_id int NOT NULL UNIQUE,
    FOREIGN KEY (product_id) REFERENCES product(product_id),
    FOREIGN KEY (user_id) REFERENCES customer(user_id),
    FOREIGN KEY (payment_id) REFERENCES payments(payment_id)
);
```

Figure 16 creating buys table with its constraints

```
mysql> desc buys;
+------------+------+------+-----+---------+-------+
| Field      | Type | Null | Key | Default | Extra |
+------------+------+------+-----+---------+-------+
| product_id | int  | NO   | MUL | NULL    |       |
| user_id    | int  | NO   | MUL | NULL    |       |
| payment_id | int  | NO   | PRI | NULL    |       |
+------------+------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql>
```

Figure 17 describe buys table

```
-- creating table category_contains
create table category_contains(
    product_id int  NOT NULL,
    category_id int  NOT NULL,
    FOREIGN KEY (product_id) REFERENCES product(product_id),
    FOREIGN KEY (category_id) REFERENCES category(category_id)
    );
```

Figure 18 creating category_contains table

```
mysql> desc category_contains;
+-------------+------+------+-----+---------+-------+
| Field       | Type | Null | Key | Default | Extra |
+-------------+------+------+-----+---------+-------+
| product_id  | int  | NO   | MUL | NULL    |       |
| category_id | int  | NO   | MUL | NULL    |       |
+-------------+------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql>
```

Figure 19 describe category contains

```
-- creating table manages_category
create table manages_category(
    admin_id int  NOT NULL,
    category_id int  NOT NULL,
    FOREIGN KEY (admin_id) REFERENCES admin(admin_id),
    FOREIGN KEY (category_id) REFERENCES category(category_id)
);
```

Figure 20 creating manages_category with its constraints

```
mysql> desc manages_category;
+-------------+------+------+-----+---------+-------+
| Field       | Type | Null | Key | Default | Extra |
+-------------+------+------+-----+---------+-------+
| admin_id    | int  | NO   | MUL | NULL    |       |
| category_id | int  | NO   | MUL | NULL    |       |
+-------------+------+------+-----+---------+-------+
2 rows in set (0.01 sec)

mysql>
```

Figure 21 describe manages_category table

```
88    -- creating table manages_product
89    create table manages_product(
90        admin_id int NOT NULL,
91        product_id int NOT NULL,
92        FOREIGN KEY (admin_id) REFERENCES admin(admin_id),
93        FOREIGN KEY (product_id) REFERENCES product(product_id)
94    );
95
```

Figure 22 creating manages_product with its constraints

```
mysql> desc manages_product;
+------------+------+------+-----+---------+-------+
| Field      | Type | Null | Key | Default | Extra |
+------------+------+------+-----+---------+-------+
| admin_id   | int  | NO   | MUL | NULL    |       |
| product_id | int  | NO   | MUL | NULL    |       |
+------------+------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql>
```

Figure 23 describe manages_product

```
-- inserting dummy data in the tables
insert into customer values('subhendu','s@gmail.com','subh@pass','delhi',1,987654);
insert into customer values('kenny','k@gmail.com','kenn@pass','punjab',2,789456);
insert into customer values('rohit','r@gmail.com','rohi@pass','bangalore',3,987564);


insert into product values('chair1','wood','brown',100, 1500,50);
insert into product values('chair2','plastic','red',101, 700,120);
insert into product values('table1','wood','black',200, 2000,70);
insert into product values('table2','plastic','orange',201, 1600,86);

insert into category values(1001,'chairs');
insert into category values(1002,'tables');

insert into admin values('a1@gmail.com','a1@pass',10001);
insert into admin values('a2@gmail.com','a2@pass',10002);

insert into payments values(501,1);
insert into payments values(502,2);
insert into payments values(503,3);
insert into payments values(504,1);

insert into cart values(5,100,1,501);
insert into cart values(8,201,1,501);

insert into buys values(100,2,502);
insert into buys values(101,1,504);

insert into category_contains values(100,1001);
insert into category_contains values(200,1002);

insert into manages_category values(10001,1001);
insert into manages_category values(10002,1002);

insert into manages_product values(10001,100);
insert into manages_product values(10002,200);
```

Figure 24 adding dummy values in all the tables

```
mysql> select * from customer;
+----------+-------------+-----------+-----------+---------+----------+
| name     | email       | password  | address   | user_id | phone_no |
+----------+-------------+-----------+-----------+---------+----------+
| subhendu | s@gmail.com | subh@pass | delhi     |       1 |   987654 |
| kenny    | k@gmail.com | kenn@pass | punjab    |       2 |   789456 |
| rohit    | r@gmail.com | rohi@pass | bangalore |       3 |   987564 |
+----------+-------------+-----------+-----------+---------+----------+
3 rows in set (0.01 sec)

mysql>
```

Figure 25 viewing customer table

3.4 Update operations violating the schema constraints

```sql
-- Update operations violating the schema constraints

update customer set phone_no='adm' where user_id=1; -- domain constraint
update customer set password=NULL where user_id=1; -- entity integrity constraint
update customer set user_id=1 where name='kenny'; -- key constraint

update product set price='abcd' where product_id=200; -- domain constraint
update product set price=NULL where product_id=200; -- entity integrity constraint
update product set product_id=101 where name='chair1'; -- key constraint

update category set category_id='jai' where category_title='chairs'; -- domain constraint
update category set category_id=NULL where category_title='tables'; -- entity integrity constraint
update category set category_id=1001 where category_title='tables'; -- key constraint

update payments set payment_id='abc' where user_id=1; -- domain constraint
update payments set payment_id=NULL where user_id=1; -- entity integrity constraint
update payments set user_id=5 where payment_id=502; -- relational integrity constraint

```

Figure 26 Examples of update commands which are violating schema constraints

```
mysql> update customer set phone_no='adm' where user_id=1;
ERROR 1366 (HY000): Incorrect integer value: 'adm' for column 'phone_no' at row 1
mysql>
```

Figure 27 example of violating domain constraint (ERROR)

```
mysql> update customer set password=NULL where user_id=1;
ERROR 1048 (23000): Column 'password' cannot be null
mysql>
```

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

```
mysql> update customer set user_id=1 where name='kenny';
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails (`online_furniture_store`.`buys`, CONSTRAINT `buys_ibfk_2` FOREIGN KEY (`user_
id`) REFERENCES `customer` (`user_id`))
mysql>
```

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

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