

DATABASE MANAGEMENT SYSTEMS

(10211DS207)

TASK : 12

TEAM DETAILS :

Team Leader : P koteswara Rao (VTU28129)

Reg No : (24UEDC0053)

Team members :

Names :

VTU :

Registrationno:

M. karthik reddy

(VTU28133)

(24UEDC0039)

G. Adithya reddy

(VTU28321)

(24UEDC0026)

N.Mohan sai

(VTU28583)

(24UEDC0046)

B. Maheshwar reddy

(VTU28585)

(24UEDC0007)

AIM :

To develop a microproject on Online Grocery System.

1. ER Diagram

The ER Diagram for an Online Grocery System represents how different components like customers, products, orders, carts, payments, and deliveries interact with each other. This model helps in designing a database that efficiently handles online ordering, payment, inventory, and delivery of grocery items.

Entities

1. Customer

- Attributes:

Customer_ID (PK), Name, Email, Phone, Address, Password

2. Product

- Attributes:

Product_ID (PK), Name, Category_ID (FK), Price, Description, Stock_Quantity

3. Category

- Attributes:

Category_ID (PK), Name, Description

4. Order

- Attributes:

Order_ID (PK), Customer_ID (FK), Order_Date, Total_Amount, Status

5. Order_Item

- Attributes:

OrderItem_ID (PK), Order_ID (FK), Product_ID (FK), Quantity, Subtotal

6. Cart

- Attributes:

Cart_ID (PK), Customer_ID (FK), Created_Date

7. Cart_Item

- **Attributes:**

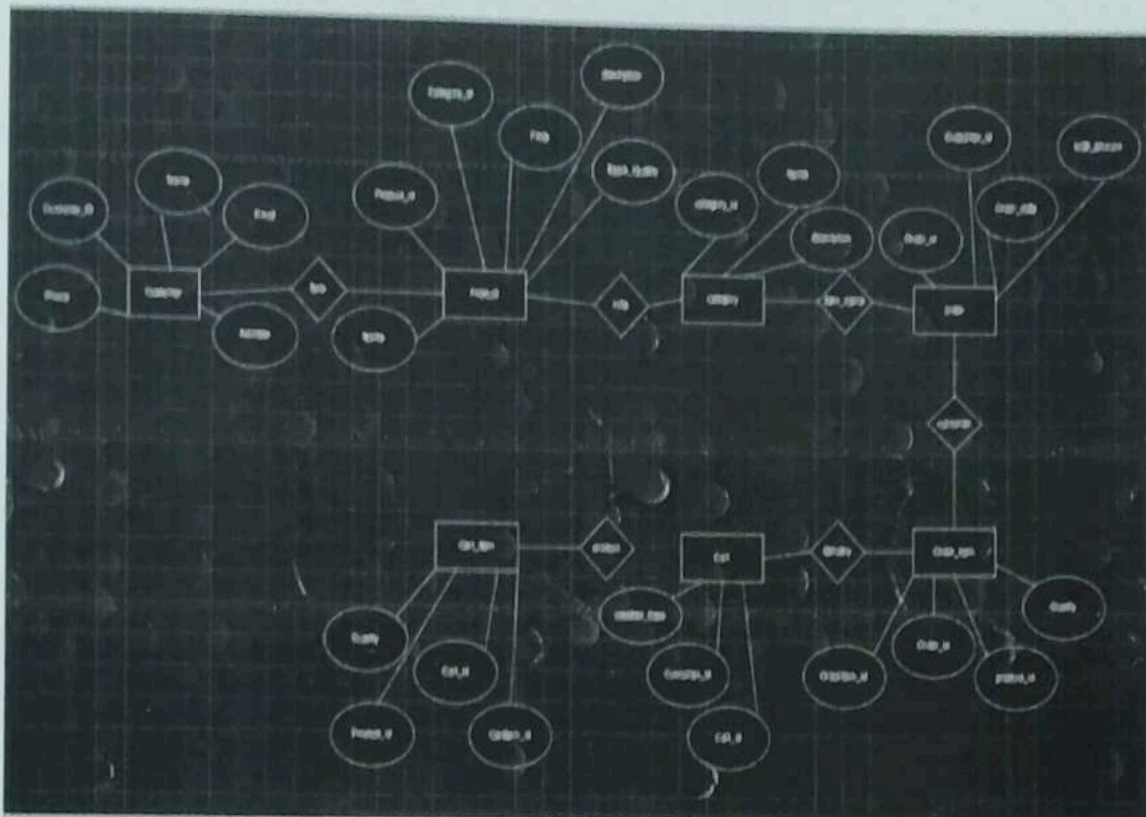
CartItem_ID (PK), Cart_ID (FK), Product_ID (FK), Quantity

Relationships

Relationship	Description	Type
Customer → Order	One customer can place many orders	1-to-N
Order → Order_Item	One order can contain multiple products	1-to-N
Product → Order_Item	One product can be in many order items	1-to-N
Customer → Cart	One customer has one cart	1-to-1
Cart → Cart_Item	One cart contains multiple items	1-to-N
Product → Cart_Item	One product can be added to many carts	1-to-N
Product → Category	One category has many products	1-to-N

- Each box represents an entity with its attributes inside.
- PK stands for Primary Key, which uniquely identifies each record in the table.
- FK stands for Foreign Key, which establishes a relationship between tables.
- The line connecting the entities represent the relationships between them.
- The notation "1" and "N" indicates the cardinality of the relationships.

ER DIAGRAM :



2. SQL Queries & Relational operations :

1. Select operation :

```
SELECT *  
FROM Product  
WHERE Price > 100;
```

Output:

ID	name	price	category
1	Wireless Headphones	150.00	Electronics
2	Office Chair	120.00	Furniture
3	Smartwatch	100.00	Electronics
7	Blender	110.00	Kitchen

2. Project operation :

```
SELECT Name, Price  
FROM Product;
```

Output :

name	price
Wireless Headphones	150.00
Laptop Stand	45.00
Office Chair	120.00
USB-C Hub	35.00
Smartwatch	100.00
Desk Lamp	29.00
Blender	110.00

3. Union :

```
SELECT Product_ID FROM Cart_Item  
UNION  
SELECT Product_ID FROM Order_Item;
```

Output :

product_id
1
3
5
5
7

4. Union all :

```
SELECT Product_ID FROM Cart_Item  
UNION ALL  
SELECT Product_ID FROM Order_Item;
```

Ouput :

product_id
1
3
5
5
7
3
4

5. intersect :

```
SELECT Customer_id FROM Orders  
INTERSECT  
SELECT Customer_id FROM Cart;
```

Output :

```
+-----+  
| customer_id |  
+-----+  
| 101         |  
| 203         |  
| 315         |
```

6. Sum :

```
SELECT SUM(Quantity) AS Total_Products_Sold  
FROM Order_Item;
```

Output :

```
+-----+  
| total_products_sold |  
+-----+  
| 342                 |
```

7. Count :

```
SELECT COUNT(*) AS Total_Customers  
FROM Customer;
```

Output :

```
+-----+  
| Total_Customers |  
+-----+  
| 250             |  
+-----+
```

8. AVG :

```
SELECT AVG(Price) AS Average_Product_Price  
FROM Product;
```

Output :

```
+-----+  
| Average_Product_Price |  
+-----+  
| 85.42 |
```

9. Minimum :

```
SELECT MIN(Price) AS Lowest_Product_Price  
FROM Product;
```

Output :

```
+-----+  
| Lowest_Product_Price |  
+-----+  
| 12.99 |
```

10. Maximum :

```
SELECT MAX(Price) AS Highest_Product_Price  
FROM Product;
```

Output :


```

+-----+
| highest_product_price |
+-----+
| 499.99 |
+-----+

```

11. Nested Querires :

Find products priced above the average price

```

SELECT Name, Price
FROM Product
WHERE Price > (
    SELECT AVG(Price)
    FROM Product
);

```

Output :

```

+-----+-----+
| Name           | Price |
+-----+-----+
| Wireless Headphones | 150.00 |
| Smartwatch      | 199.99 |
| Office Chair    | 120.00 |
+-----+-----+

```

12. PL SQL Procedure :

Check if a product is in stock

```

DECLARE
    v_stock NUMBER;
BEGIN
    SELECT Stock_Quantity INTO v_stock
    FROM Product
    WHERE Product_ID = 101;

    IF v_stock > 0 THEN
        DBMS_OUTPUT.PUT_LINE('Product is available.');

```

Output :

Product is out of stock.

13. JOINS

A) INNER JOIN :

```
SELECT
    P.ProductID,
    P.Name AS ProductName,
    C.CategoryName,
    P.Price,
    P.Stock
FROM Product P
INNER JOIN Category C ON P.CategoryID = C.CategoryID;
```

Output :

ProductID	ProductName	CategoryName	Price	Stock
1	Apple iPhone 15	Electronics	999.99	150
2	Samsung TV 55"	Electronics	649.00	40
3	Nike Air Max	Footwear	120.00	200
4	Office Chair	Furniture	89.99	75

B) LEFT OUTER JOIN :

```
SELECT
    P.ProductID,
    P.Name AS ProductName,
    C.CategoryName
FROM Product P
LEFT OUTER JOIN Category C ON P.CategoryID = C.CategoryID;
```

Output :

ProductID	ProductName	CategoryName
1	Apple iPhone 15	Electronics
2	Nike Air Max	Footwear
3	Mystery Gadget	NULL

C) RIGHT OUTER JOIN :

```
SELECT
    C.CategoryID,
    C.CategoryName,
    P.ProductID,
    P.Name AS ProductName
FROM Product P
RIGHT OUTER JOIN Category C ON P.CategoryID = C.CategoryID;
```

Output :

CategoryID	CategoryName	ProductID	ProductName
1	Electronics	1	iPhone 15
2	Footwear	2	Nike Air Max
3	Furniture	NULL	NULL

D) FULL OUTER JOIN :

```
SELECT
    C.CustomerID,
    C.Name AS CustomerName,
    O.OrderID,
    O.OrderDate,
    O.TotalAmount
FROM Customer C
FULL OUTER JOIN "order" O ON C.CustomerID = O.CustomerID;
```

Output :

CustomerID	CustomerName	OrderID	OrderDate	TotalAmount
101	Alice Smith	5001	2025-09-01	150.00
102	Bob Johnson	NULL	NULL	NULL
NULL	NULL	6002	2025-09-10	200.00

4.Normalization :

Normalization is the process of organizing data in a database to reduce redundancy and improve data integrity.

We normally go through stages called **Normal Forms (1NF → 2NF → 3NF → BCNF)**.

For our **Online Grocery System**, we can start from a sample normalized relation and use **Griffith Normalization Tool** to automate the process.

1. First Normal Form (1NF)

A relation (table) is in 1NF if all its attributes contain atomic (indivisible) values, and each row is unique.

Key point: No repeating groups or arrays.

2. Second Normal Form (2NF)

A relation is in 2NF if it is in 1NF and every non-prime attribute is fully functionally dependent on the whole primary key.

Key point: No partial dependency on a part of a composite key.

3. Boyce-Codd Normal Form (BCNF)

A relation is in BCNF if it is in 2NF and every determinant is a superkey.

Key point: Even stricter than 2NF; eliminates anomalies caused by non-key attributes determining other attributes

• In this database we perform normalization using Griffith university normalization tool.

First Normal Form :

Attributes in Table

1 Separate attributes using a comma (,)

customer_id, customer_name, cart, cart_item, order, order_item, product, product_id, product_name

Functional Dependencies

customer_id ×



customer_name × cart ×

cart_item × order ×

order_item × product ×

product_id × product_name ×

Delete

Check Normal Form



2NF

The table is in 2NF



3NF

The table is in 3NF



BCNF

The table is in BCNF

Show Steps



2NF

find all candidate keys. The candidates keys are { customer_id }, The set of key attributes are: { customer_id }
for each non-trivial FD, check whether the LHS is a proper subset of some candidate key or the RHS are not
all key attributes
checking FD: customer_id →
customer_name, cart, cart_item, order, order_item, product, product_id, product_name

3NF

find all candidate keys. The candidates keys are { customer_id }, The set of key attributes are: { customer_id }
for each FD, check whether the LHS is superkey or the RHS are all key attributes
checking functional dependency customer_id →
customer_name, cart, cart_item, order, order_item, product, product_id, product_name

BCNF

A table is in BCNF if and only if for every non-trivial FD, the LHS is a superkey.

Normaliza to 2NF :

Normalize to 2NF

Attributes

customer_id customer_name cart cart_item order order_item product product_id product_name

Functional Dependencies

customer_id → customer_name
customer_id → cart
customer_id → cart_item
customer_id → order
customer_id → order_item
customer_id → product
customer_id → product_id
customer_id → product_name

Show Steps



First, find the minimal cover of the FDs, which includes the FDs

customer_id → customer_name
customer_id → cart
customer_id → cart_item
customer_id → order
customer_id → order_item
customer_id → product
customer_id → product_id
customer_id → product_name

Initially rel[1] is the original table:

Round1: checking table rel[1]

***** The table is in 2NF already, send it to output *****

Normalize to 3NF :

1NF to 3NF

Attributes

customer_id customer_name cart cart_item order order_item product product_id product_name

Functional Dependencies

customer_id → customer_name

customer_id → cart

customer_id → cart_item

customer_id → order

customer_id → order_item

customer_id → product

customer_id → product_id

customer_id → product_name

Show Steps



Table already in 3NF

Normalize to BCNF :

Normalize to BCNF

Attributes

customer_id product_name customer_name cart cart_item order order_item product product_id

Functional Dependencies

customer_id product_name → customer_name cart cart_item order order_item product product_id

Show Steps

Table already in BCNF, return itself.

Thus, the normalization to 1NF, 2NF, 3NF, BCNF is completed successfully.

5.Document database using MONGODB :

CRUD, Which stands for Create, Read, Update, and Delete, represents a set of fundamental operations used to insert with and manipulate data stored in a database. These operations serve as the building blocks upon which countless applications, from simple to highly complex, rely.

Create :


```

db.createcollection('customers');
db.customers.insertmany([
{
  customer_id: 'C001',
  name: 'koti',
  contact_no: '8956495236',
  address: 'chennai'
},
{
  customer_id: 'C002',
  name: 'shivaya',
  contact_no: '9999999999',
  address: 'kasi'
}
]);

```

Output:

```

{
  "acknowledged" : true,
  "insertedIds" : [
    ObjectId("68f785e0196f2fe7a371c0ac"),
    ObjectId("68f785e0196f2fe7a371c0ad"),
    ObjectId("68f785e0196f2fe7a371c0ae")
  ]
}
{ "_id" : ObjectId("68f785e0196f2fe7a371c0ac"), "empId" : 1, "name" : "Clark", "dept" : "Sales" }
{ "_id" : ObjectId("68f785e0196f2fe7a371c0ad"), "empId" : 2, "name" : "Ava", "dept" : "Sales" }

```

Read :

```
db.customers.find().pretty();
```

output :

```

{
  "_id" : ObjectId("68efbd7a10027ef18f9360dc"),
  "customer_id" : "C001",
  "name" : "koti",
  "contact_no" : "8956495236",
  "address" : "Chennai"
}

```

Update :


```
db.customers.updateOne(  
  { customer_id: "C001" },  
  { $set: { email: "koti@gmail.com" } }  
);
```

Output :

```
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
```

Delete :

```
db.customers.deleteOne({ customer_id: "C001" });
```

output :

```
{ "acknowledged" : true, "deletedCount" : 1 }
```

6. Graph Database using MONGODB(using neo4j online compiler)

a) Create a graph database :

```
CREATE (1:customer{id:"C001",name:"koti"})
```

Output :

Created 1 node, set 2 properties, added 1 label

```
CREATE (a1:Animal{id:"A001",breed:"jersey",gender:"Female",age:4})
```

Output :

Created 1 node, set 4 properties, added 1 label

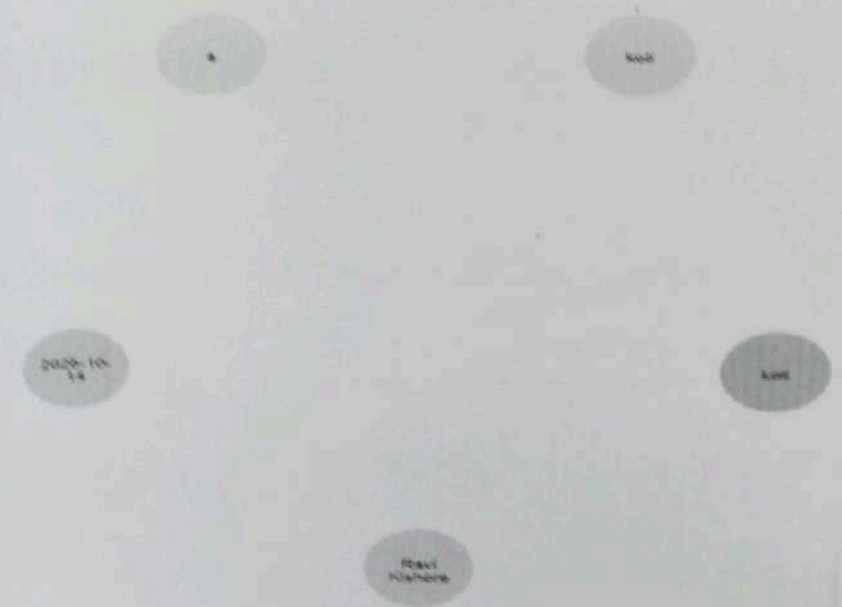
```
CREATE (m1:MilkProduction {id: "MP001", quantity: 15, date: "2025-10-14"})
```

Output :

Created 1 node, set 3 properties, added 1 label

Match(n) return(n)

Output :

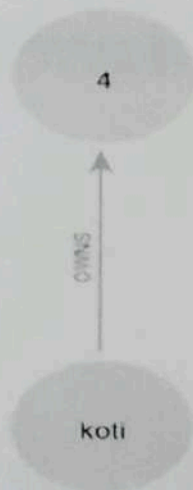


```

MATCH (c:customer), (a:Animal)
WHERE c.name = 'koti' AND a.id = 'A001'
CREATE (c)-[:OWNS]->(a)
RETURN c, a

```

Output :



b) Delete a node from Farmer :

Syntax :

MATCH (c:customer)

WHERE c.name = 'koti'

DELETE c

Output : Deleted 1 node, deleted 1 relationship

4

koti

Thus, the document database and graph database by using mongodb is implemented.

VEL TECH	
EX NO.	12
PERFORMANCE (5)	5
RESULT AND ANALYS'S (5)	8
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	20
SIGN WITH DATE	

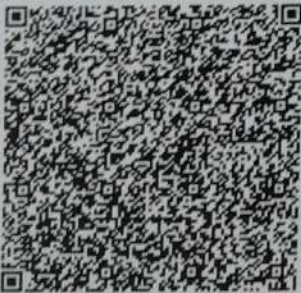
4
25/10/20

Result :

Thus, micro project for Online Grocery System was developed and implemented successfully.

COURSE COMPLETION CERTIFICATE

The certificate is awarded to
GUDA ADITHYA REDDY
for successfully completing the course
Data Management
on October 15, 2025



Infosys | Springboard

Congratulations! You make us proud!

Issued on: Thursday, October 16, 2025
To verify, scan the QR code at <https://verify.onwingspan.com>

Satheesha B.N.
Satheesha B. Nanjappa
Senior Vice President and Head
Education, Training and Assessment
Infosys Limited