Comprehensive Database Design, Optimization, and Advanced Features – Task 10

RDBMS used : MySql

Business Scenario – eCommerce Platform

- **Schema Design**
  - Tables Created : Products, Orders, Customers, OrderDetails

- **Indexing**
  - Indexes are used to retrieve data from the database in a quickl manner.
  - The users cannot see the indexes, they are just used to speed up searches/queries.
  - **Syntax:**
    CREATE INDEX *index_name*
    ON *table_name* (*column1*, *column2*, ...);

```
CREATE INDEX idx_orders_customer ON orders(order_id);
CREATE INDEX idx_orderdetails_product ON orderdetails(prod_id);
CREATE INDEX idx_products_name ON products(prod_name);
```

- **Triggers**
  - A MySQL trigger is a stored program (with queries) which is executed automatically to respond to a specific event such as insertion, updation or deletion occurring in a table.
  - 6 types of triggers: Before Update, After Update, Before Insert, After Insert, Before Delete, After Delete.
  - **Syntax:**

```
DELIMITER //
CREATE TRIGGER trigger_name
{BEFORE | AFTER} {INSERT | UPDATE | DELETE}
ON table_name FOR EACH ROW
BEGIN
   -- SQL statements
END; //
```

- This trigger will **reduce inventory** when a new order is placed:

```
delimiter //
create trigger ReduceStock after insert on orderdetails
for each row
begin
        update products
        set stock = stock-new.quantity
        where prod_id = new.prod_id;
end //
```

➢ **Transactions**

- A database transaction is a series of operations executed as a single unit of work.
- Transactions allow grouping a set of operations as an inseparable single unit of operations, either all of which succeed or none of which does.
- This assures validity and consistency in the data.
- This transaction ensures that an **order and order details** are inserted together, or **rolled back** if an error occurs:

```
start transaction;
insert into orders(order_id, quantity) values(11,3);
SET @order_id = LAST_INSERT_ID();
insert into orderdetails (order_id, prod_id, quantity, subtotal)
VALUES (@OrderID, 1003, 3, 5000.00);

COMMIT;
```

➢ **Views**

- In SQL, a view is a virtual table based on the result-set of an SQL statement.
- A view contains rows and columns, just like a real table. The fields in a view are fields from one or more actual tables in the database.
- Creating a view to easily fetch the order summary:

```
CREATE VIEW OrderSummary AS
SELECT o.order_id, c.name AS Customer, o.Order_Date
FROM Orders o
JOIN Customers c ON o.order_id = c.order_id;

select * from OrderSummary;
```

order_id	Customer	Order_Date
2	Ramu	2020-07-17
3	Shyam	2021-10-08
1	Sheela	2020-04-28

- **Testing**

SELECT * FROM customers;

customer_id	name	address	items_qty	order_id
1	Ramu	chennai	2	2
2	Shyam	banglore	1	3
3	Sheela	trichy	2	1
NULL	NULL	NULL	NULL	NULL

SELECT * FROM products WHERE stock >=10;

prod_id	prod_name	category	price	color	size	gender	stock
1001	sneakers	casual	2500	dark brown	6	1	10
1003	boots	semi-formal	3000	black	5	0	12
1004	flip-flops	casual	500	blue	5	0	10
1005	boots	casual	500	black	8	1	20
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

CALL GetProdWithSize(5);

prod_id	prod_name	category	price
1003	boots	semi-formal	3000
1004	flip-flops	casual	500