Text

Description automatically generated

**Activity based**

**Project 3 Report on**

**Database Management Systems**

**Submitted to Vishwakarma University, Pune**

**Under the Initiative of**

**Contemporary Curriculum, Pedagogy, and Practice (C2P2)**

**By**

**Alfiya Sayyed**

**SRN No : 202202232**

**Roll No : 58**

**Div : F**

**Second Year Engineering**

**Department of Computer Engineering**

**Faculty of Science and Technology**

**Academic Year**

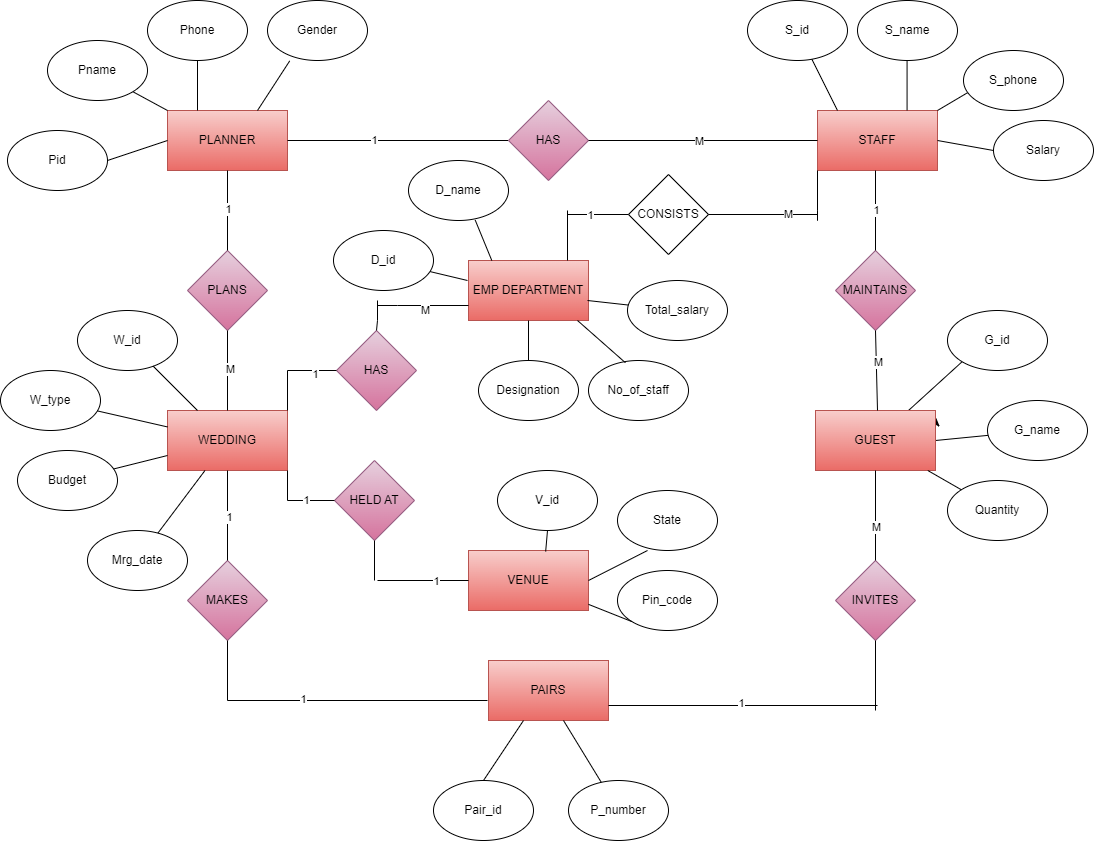
**2023-2024**

**Implement stored procedure, Cursor, trigger and views on Wedding Management System**

**Project Statement:**

Implement stored procedure, Cursor, trigger and views on Wedding Management System.

**ER diagram :**

****

*Figure 1-ER Diagram of Wedding Management System*

**Mention about stored procedure, Cursor, trigger and views commands applicable to given problem statement**

1. Trigger:

- A trigger is a database object that automatically executes in response to certain events on a particular table or view in a database.

- These events can include INSERT, UPDATE, DELETE, or SELECT operations.

- Triggers are typically used to enforce business rules, validate input data, maintain referential integrity, and audit changes to the database.

- There are two main types of triggers: BEFORE triggers and AFTER triggers, which indicate whether the trigger should execute before or after the event.

- Triggers are written in SQL and are associated with a specific table or view in the database.

2. Cursor:

- A cursor is a database object used to retrieve, manipulate, and process data row by row, especially in a procedural manner.

- Cursors are typically used in stored procedures, triggers, and other database programs to iterate over the result set of a query and perform operations on each row individually.

- Cursors provide a way to navigate through the result set sequentially, fetch data from the current row, and move to the next row until all rows have been processed.

- Cursors can be forward-only, scrollable, or dynamic, depending on the requirements of the application.

- While cursors can be useful for processing data row by row, they can also have performance implications, especially when dealing with large result sets, so they should be used judiciously.

- Views are virtual database objects that represent the result set of a SELECT query, providing a customized and simplified view of the underlying data.

- They offer an abstraction layer over complex data structures, enabling users to interact with data in a more intuitive and efficient manner.

-Views can join multiple tables, apply filters, and perform calculations, allowing users to retrieve specific subsets of data without directly accessing the underlying tables.

- Views enhance database security by restricting access to sensitive columns or rows, ensuring that users only see the data they are authorized to access.

- Views simplify application development and maintenance by encapsulating complex queries and business logic, reducing the need to rewrite queries across multiple applications or queries.

3. Stored Procedure:

- A stored procedure is a precompiled collection of one or more SQL statements and procedural logic that is stored in the database and can be executed as a single unit.

- Stored procedures can accept input parameters, perform database operations, and return output parameters or result sets to the calling program.

- They are often used to encapsulate business logic, implement complex database operations, improve performance by reducing network traffic, and enforce security by controlling access to database objects.

- Stored procedures are written in a procedural language supported by the database system, such as PL/SQL for Oracle, T-SQL for SQL Server, or PL/pgSQL for PostgreSQL.

- They are compiled and stored in the database, making them reusable and portable across different applications and clients.

In summary, triggers, cursors, stored procedures and views are powerful database constructs used to automate tasks, process data, enforce business rules, and improve the efficiency and security of database operations. Each serves a specific purpose and can be leveraged to achieve various database-related goals.

**Problem Description:**

1. **Stored Procedure**:
   * A stored procedure named **GetWeddingDetails** is created to encapsulate multiple SELECT statements retrieving data from various tables such as Planner, Wedding, Emp\_Department, Venue, Pairs, Staff, and Guest.
   * This procedure can be called to fetch detailed information about weddings and related entities.
2. **Cursor**:
   * A cursor-based stored procedure named **CalculateTotalBudget** is defined to calculate the total budget for all weddings in the database.
   * It iterates through the Wedding table using a cursor, accumulating the budget of each wedding.
   * Cursors provide a mechanism to traverse the result set of a query row by row, allowing for more complex data processing.
3. **Trigger**:
   * A trigger named **UpdateGuestQuantity** is created to enforce business rules when inserting new records into the Guest table.
   * This trigger ensures that the quantity of guests for a wedding does not exceed the maximum allowed guests specified in the Emp\_Department table for the corresponding wedding.
   * Triggers are automatically executed in response to specific database events and are used to enforce data integrity and implement business logic.
4. **View**:
   * Although not explicitly mentioned in the script, views can be used to simplify complex queries by encapsulating them into a virtual table.
   * Views can enhance data security by restricting access to certain columns or rows of a table.
   * They can also improve query performance by predefining commonly used joins or aggregations.

Overall, these database objects and features contribute to the functionality, maintainability, and performance of the database system by encapsulating logic, automating tasks, and providing simplified access to data.

**Project stage- 3 details:**

-- Inline Views: Example: Retrieve the details of weddings along with the average budget for each wedding type.

SELECT W.W\_id, W.W\_type, W.Budget, W.Mrg\_date, AvgBudget.WeddingTypeAvgBudget

FROM Wedding W

JOIN (

SELECT W\_type, AVG(Budget) AS WeddingTypeAvgBudget

FROM Wedding

GROUP BY W\_type

) AS AvgBudget ON W.W\_type = AvgBudget.W\_type;

-- Procedure

DELIMITER //

CREATE PROCEDURE GetWeddingDetails()

BEGIN

-- Select all from Planner table

SELECT \* FROM Planner;

-- Select all from Wedding table

SELECT \* FROM Wedding;

-- Select all from Emp\_Department table

SELECT \* FROM Emp\_Department;

-- Select all from Venue table

SELECT \* FROM Venue;

-- Select all from Pairs table

SELECT \* FROM Pairs;

-- Select all from Staff table

SELECT \* FROM Staff;

-- Select all from Guest table

SELECT \* FROM Guest;

END //

DELIMITER ;

-- Cursor

DELIMITER //

CREATE PROCEDURE CalculateTotalBudget()

BEGIN

DECLARE total DECIMAL(10, 2);

DECLARE done INT DEFAULT FALSE;

DECLARE wedding\_id INT;

DECLARE budget DECIMAL(10, 2);

-- Declare cursor for selecting budgets from Wedding table

DECLARE budget\_cursor CURSOR FOR

SELECT W\_id, Budget FROM Wedding;

-- Declare handler for cursor

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

-- Initialize total

SET total = 0;

-- Open cursor

OPEN budget\_cursor;

-- Loop through cursor

read\_loop: LOOP

FETCH budget\_cursor INTO wedding\_id, budget;

IF done THEN

LEAVE read\_loop;

END IF;

-- Accumulate total budget

SET total = total + budget;

END LOOP;

-- Close cursor

CLOSE budget\_cursor;

-- Output total budget

SELECT total AS Total\_Budget;

END //

DELIMITER ;

-- Trigger

DELIMITER //

CREATE TRIGGER UpdateGuestQuantity BEFORE INSERT ON Guest

FOR EACH ROW

BEGIN

DECLARE max\_guests INT;

DECLARE wedding\_id\_val INT;

-- Get the Wedding\_id for the given Pairs\_id

SELECT Wedding\_id INTO wedding\_id\_val

FROM Pairs

WHERE Pair\_id = NEW.Pairs\_id;

-- Get the maximum number of guests allowed for the wedding

SELECT No\_of\_staff INTO max\_guests

FROM Emp\_Department

WHERE Wedding\_id = wedding\_id\_val;

-- Check if the new quantity exceeds the maximum allowed guests

IF NEW.Quantity > max\_guests THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Cannot exceed maximum number of guests allowed for this wedding';

END IF;

END;

//

DELIMITER ;

-- Call the stored procedure GetWeddingDetails()

CALL GetWeddingDetails();

-- Execute the cursor CalculateTotalBudget()

CALL CalculateTotalBudget();

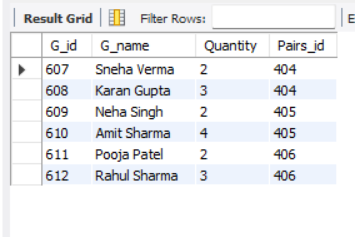
-- run trigger

-- Insert a new guest into the Guest table, where the quantity exceeds the maximum allowed guests for the corresponding wedding.

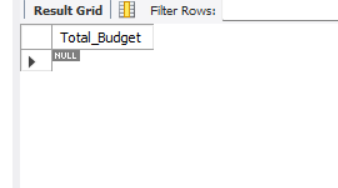
INSERT INTO Guest (G\_id, G\_name, Quantity, Pairs\_id) VALUES (613, 'Test Guest', 15, 404);

**OUTPUT:**

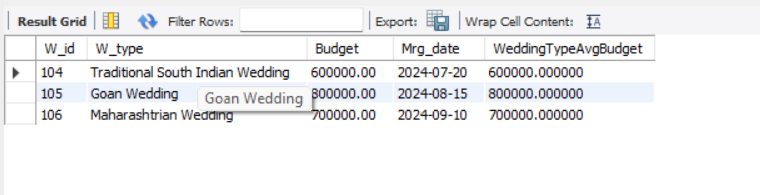
**Procedure**

****

**Cursor (navigates 1 by 1 )**

****

**View**

****

**Conclusion :**

In conclusion, the provided SQL script outlines the database schema for managing wedding-related information, including planners, weddings, departments, venues, pairs, staff, and guests. The script demonstrates the utilization of various SQL features and constructs to define the database structure, insert data, perform queries, and implement business logic:

1. \*\*Database Structure\*\*:

- The script creates tables for different entities involved in wedding planning, such as Planner, Wedding, Emp\_Department, Venue, Pairs, Staff, and Guest.

- Relationships between tables are established using foreign keys, ensuring data integrity and enforcing referential integrity constraints.

2. \*\*Data Manipulation\*\*:

- Sample data is inserted into the tables to populate the database with relevant information, including details of planners, weddings, departments, venues, pairs, staff, and guests.

- Select queries are executed to retrieve data from the database, allowing users to view information about weddings, departments, staff, and guests.

3. \*\*Advanced SQL Features\*\*:

- The script showcases the use of advanced SQL features such as set operations (UNION, INTERSECT, EXCEPT), subqueries (single-row, multiple-row, correlated, nested, and derived), and joins (LEFT JOIN, INNER JOIN, OUTER JOIN) to perform complex data manipulations and retrievals.

- Stored procedures, cursors, triggers, and views are utilized to encapsulate logic, automate tasks, enforce business rules, and simplify data access.

4. \*\*Data Integrity and Business Logic\*\*:

- Triggers are implemented to enforce business rules, such as restricting the number of guests for a wedding to the maximum allowed quantity specified in the Emp\_Department table.

- Cursors are used to perform calculations, such as calculating the total budget for all weddings in the database.

Overall, the SQL script provides a comprehensive database schema and demonstrates the use of advanced SQL features and constructs to manage wedding-related information effectively. It serves as a foundation for building a robust and efficient wedding planning application, ensuring data integrity, scalability, and maintainability.