**Gym Database Application**

**- Summary report -**

**Instructor**:

ECHIHABI Karima

**Teaching Assistants**:

ZEROUARI Hasnae

ABDENOURI Khaoula

**Group members**:

BENFELLAH Ikram

BENJELLOUN Elghali

BLALI Yassine

KABLY malak

**Introduction to Gym Database:**

Our Gym Database is designed to streamline gym management operations, ensuring data integrity, efficient access to information, and adherence to business rules.It is tailored to manage the operations, membership, and various facets of the gym’s facilities. It is structured to handle information related to customers, employees, training sessions, equipment management, and more, facilitating efficient day-to-day operations within the gym.

**Key Components:**

**Customer Management:**

* Tracks customer details, including personal information, registration dates, and birthdates.
* Manages customer memberships, linking them to specific disciplines and subscriptions.

**Employee Information:**

* Captures details of gym employees, such as names, salaries, and contact information.
* Manages administrative roles and responsibilities within the gym.

**Discipline and Training Sessions:**

* Organizes disciplines offered by the gym and associates them with training sessions.
* Links training sessions to specific spaces, ensuring effective scheduling.

**Equipment and Space Management:**

* Manages gym spaces and their capacities, enforcing constraints like space states and capacity checks.
* Tracks gym equipment, including types, brands, and maintenance schedules.

**Locker Room and Lounge Facilities:**

* Handles locker room facilities, tracking available lockers for rent and their pricing.
* Manages lounge spaces within the gym facility.

**Membership and Subscriptions:**

* Manages gym memberships, including the number of people involved and subscription durations.
* Associates memberships with customers, disciplines, and subscriptions.

**Feedback and Administration:**

* Captures customer feedback, linking them to both customers and administrative employees.
* Manages administrative roles, ensuring access control and responsibilities.

**Functionality Highlights:**

Triggers and Events, Views and Indexes, Roles, and Privileges. (see below for more details).

**Queries Overview:**

**DML commands (3hours):**

**🡪 SELECT STATEMENTS:**

1. ***Retrieve Customer Names and Rented Lockers:***

Uses a **JOIN** to retrieve customer names and details of their rented lockers.

2. ***Retrieve Feedback with Customer and Admin Names:***

Utilizes **LEFT JOIN** to retrieve feedback along with corresponding customer and admin names.

3***. Count Sessions for Each Discipline (Per Week):***

Performs a **LEFT JOIN** to count the number of sessions for each discipline per week.

4. ***Find Best Locker Room(s) with Minimal Price:***

Employs a **WHERE** clause to find the locker room(s) with the minimum available lockers and retrieves their prices.

5. ***Select Customers Renting Lockers or who are enrolled in a Discipline:***

Employes an **UNION** to select customers either renting a locker or enrolled in a discipline given its ID.

6. ***Order by Minimum Age for Each Membership:***

Uses **ORDER BY** to arrange results by membership ID, displaying the maximum minimum age required for each membership.

7. ***Distinct Query for Sessions:***

Applies **DISTINCT** to retrieve unique session details from the S\_session table, ordered by day and start time.

8. ***Aggregation Query for Employee Salaries:***

Utilizes **aggregate** functions (MIN, AVG, MAX) to find the minimum, average, and maximum salaries of employees.

9. ***Find min age needed to subscribe to a membership involving different disciplines:***

Uses **JOIN** operations to calculate the age needed to access a specific membership.

10. ***Select: Number of Disciplines in a Membership:***

This query retrieves the **count** of disciplines associated with each membership.

**🡪 UPDATE STATEMENTS:**

11. ***Update: Bonus for Trainer with Highest Sessions:***

Awards a bonus to the trainer with the highest number of conducted sessions.

12. ***Update: Bonus for Technicians with Equipment Maintenance:***

Technicians maintaining at least 10 pieces of equipment receive a bonus.

13. ***Update: Price Increase for Discipline with Most Customers:***

Increases the price for the discipline with the highest number of customers.

14. ***Update: Training Space Last Check Date:***

Updates the last check date for a training space with the specified space\_id.

**🡪 DELETE STATEMENTS:**

15. ***Delete: Remove Expired Subscriptions:***

Deletes subscriptions that have expired based on the duration from the registration date.

**🡪 INSERT STATEMENTS:**

16. ***Inserting Customers to a Membership:***

Example of inserting customers sharing the same membership involving one discipline.

17. ***Insert a new session:***

Example of inserting a session given a discipline and a space.

**Overall:**

The queries involve a mix of selecting, updating, and deleting records from various tables in a relational database.

The updates include awarding bonuses to employees, increasing discipline prices, and updating space-related information.

Insert statements demonstrate the addition of new customers to memberships, including scenarios with multiple customers sharing a membership.

For more details: view <https://github.com/yassine22-alt/um6p-cs-introdb-project1-KABLY/blob/main/Delivrable2/um6p-cs-introdb-project1-dml-log.txt>

**DDL SQL commands:(2hours)**

* **TRIGGERS:**

**Enforce\_max\_people\_membership:**

* Purpose: Enforces a maximum limit of 5 people in a membership.
* Event: Triggered before each insertion into the Subscribed table.
* Logic: Checks the current number of people in the membership being inserted. If it exceeds 5, it raises an error.

**Prevent\_future\_dates\_customer\_birth\_date:**

* Purpose: Prevents the insertion of future dates into the birth\_date column of the customer table.
* Event: Triggered before each insertion into the customer table.
* Logic: Check if the birth date to be inserted is in the future. If true, it raises an error.

**Prevent\_future\_dates\_employee\_first\_day\_work:**

* Purpose: Prevents the insertion of future dates into the first\_day\_work column of the employee table.
* Event: Triggered before each insertion into the employee table.
* Logic: Check if the first day of work to be inserted is in the future. If true, it raises an error.

**Prevent\_member\_access**:

* Purpose: Prevents access to a membership for customers whose age is less than the specified minimum age requirement.
* Event: Triggered before each insertion into the customer table.
* Logic: Calculates the age of the member based on the birth date. Checks the maximum age requirement among all disciplines associated with the membership. If the member's age doesn’t respect the discipline age requirement, it raises an error.

------> These triggers use conditions and logic to ensure data integrity and enforce business rules during data insertion events in our tables.

For more details view: <https://github.com/yassine22-alt/um6p-cs-introdb-project1-KABLY/blob/main/Delivrable2/um6p-cs-introdb-project1-ddl-log.txt>

* **Indexes:**

**idx\_customer\_lex\_order** on Customer:

* Purpose: Sorts the Customer table based on names in lexicographical order.
* Logic: Creates an index on the c\_name column of the Customer table, optimizing queries that involve sorting or searching based on customer names

**idx\_session\_start\_time\_day** on S\_session:

* Purpose: Sorts the S\_session table based on start time and date of the session.
* Logic: Creates a composite index on the start\_time and s\_day columns of the S\_session table. This index enhances the performance of queries involving sorting or filtering sessions by both start time and day.

**idx\_employee\_lex\_order** on Employee:

* Purpose: Sorts the Employee table based on names in lexicographical order.
* Logic: Creates an index on the e\_name column of the Employee table, optimizing queries that involve sorting or searching based on employee names.

---> These indexes contribute to faster query performance by providing efficient access paths for specific columns used in sorting or filtering conditions.

**DCL Commands:(2hours)**

**Create Roles:**

* Purpose: Creates new roles (EMPLOYEE1, IT1, CEO1, TECHNICIAN1) to assign specific privileges.
* Logic: Roles are created to logically group users based on their responsibilities and access levels.

**Create Users and Grant Roles:**

* Purpose: Establishes users with associated roles and assigns passwords.
* Logic: Users ('IT'@'%', 'EMPLOYEE'@'%', 'CEO'@'%', "TECHNICIAN"@"%") are created and granted corresponding roles (IT1, EMPLOYEE1, CEO1, TECHNICIAN1).

**Grant Privileges to Technicians:**

* Purpose: Grants specific privileges to technicians, including updating the last\_check\_date column on Equipment\_details.
* Logic: Technicians are permitted to update specific columns in the Equipment\_details table.

**Create Technician View, Grant Select:**

* Purpose: Creates a view (technician\_view) for technicians based on certain conditions then Grants the SELECT privilege to the TECHNICIAN role.
* Logic: Technicians are allowed to query and retrieve information from the technician\_view which we created to present information to technicians, showing details based on specific date conditions.

**Grant All Privileges to IT and CEO:**

* Purpose: Grants all privileges on the entire gym database to roles IT1 and CEO1.
* Logic: IT and CEO roles are given full access to the gym database.

**Create Employee View and Grant Select:**

* Purpose: Creates a view (employee\_view) for employees showing specific attributes and grants SELECT privilege to EMPLOYEE1.
* Logic: Employees are provided with a view containing restricted information, and they are granted permission to query that view.

**CHALLENGES:**

The development of the Gym Database was marked by several challenges, each demanding thoughtful solutions to ensure the robustness and effectiveness of the database system. Some of the key challenges faced during the project include:

**Data Generation and Integrity(7hours):**

*Challenge*: Generating realistic and meaningful data, especially when establishing foreign key relationships and ensuring functional dependencies, was the hardest task.

**Error Handling and Modifications(4hours):**

*Challenge*: Dealing with errors in the database design or logic required frequent modifications, impacting the overall development process which slowed the process.

**Normalization and Denormalization(3hours):**

*Challenge*: Striking a balance between normalization for data integrity and denormalization for query performance presented a challenge.

**Membership and Subscription Management:**

*Challenge*: Managing memberships involving multiple individuals, linking them to disciplines, and handling payment also posed a challenge that obliged us to make modifications within the ERD and the tables.

**Access Control and Role Definition:**

*Challenge*: Defining secure access control to protect sensitive information while accommodating various user roles (e.g., IT, Employee, CEO).

**Documentation (3hours):**

*Challenge*: writing the report and ensuring comprehensive documentation for the database structure and logic.

Through addressing these challenges, our Gym Database has evolved into a more adaptable system, capable of meeting the diverse needs of a gym facility while navigating the complexities inherent in database development.