CTU training solutions

0861 100 395 | www.ctutraining.co.za | enquiry@ctutraining.co.za

Gabriella Rakgotsoka

20232605

Question 1

Task 1: Database Design and Normalization (20 Marks)

Transform the conceptual design (ER diagram) into the relational model by converting the entities and relationships into appropriate tables. Check if your tables are normalized using the 1st, 2nd, and 3rd normal forms.

Clubs table:

Club_ID(PK)	Club_name	Stadium	Coach

Players table:

Player_ID(PK)	Player_name	Market_value	Position

Games table:

Game_ID(PK)	Date	Stadium

Game results table:

Game_ID(PK, FK)	Player_ID(FK)	Goals_scored	Coach

a) 1st Normal Form (1NF):
    a. All tables have atomic values in each cell.
    b. There are no repeating groups.
b) 2nd Normal Form (2NF):
    a. The tables are in 1NF.
    b. All non-key attributes are fully functional and dependent on the primary key.

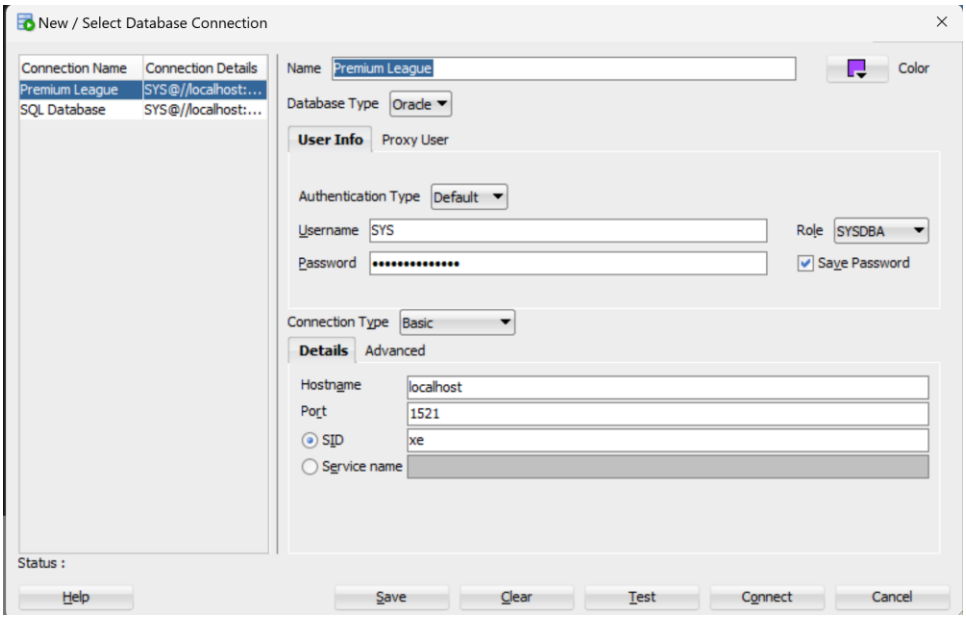
C) 3rd Normal Form (3NF):

    i. The tables are in 2NF.
    ii. There are no transitive dependencies.

This design shows that each table is normalized up to 3NF, which ensures that there are no repeating groups, and all dependencies are properly managed.

Task 2: Database Creation and Data Population (16 Marks)

I. In Oracle SQL Developer, create a database called "PremiumLeague."



Ii. Implement the tables specified in Task 1 using DDL (Data Definition Language) commands. Choose the appropriate data types, primary and foreign keys for the attributes. Provide detailed assumptions for any of your design decisions.

--Table containing attributes such as club names, stadiums where the clubs play and the coahes of each club

CREATE TABLE Clubs

(Club_ID INT PRIMARY KEY,

club_name VARCHAR(50) NOT NULL,

stadium VARCHAR(50) NOT NULL,

coach VARCHAR NOT NULL);

--Players table containing information for each player from the different clubs

CREATE TABLE Players

(Player_ID INT PRIMARY KEY,

player_name VARCHAR(50) NOT NULL,

```
market_value INT,
position INT);

--Games table. Containt data on the different games played and when.
CREATE TABLE Games
(Game_ID INT PRIMARY KEY,
date DATE,
stadium VARCHAR NOT NULL,
CONSTRAINT fk_Clubs FOREIGN KEY (stadium) REFERENCES Clubs(stadium)
);

--GameResults table contains data on the different games played by the clubs and the goals scored by each player
CREATE TABLE GameResults
(Game_ID INT,
Player_ID INT,
goals_scored INT,
CONSTRAINT fk_Games FOREIGN KEY (Game_ID) REFERENCES Games(Game_ID),
CONSTRAINT fk_Players FOREIGN KEY (Player_ID) REFERENCES Players(Players_ID),
PRIMARY KEY (Game_ID, Player_ID));
```

Iii. Generate some data to populate your tables to simulate real-world scenarios.

```
INSERT INTO Clubs (Club_ID, club_name, stadium, coach) VALUES
  INSERT INTO Clubs (Club_ID, club_name, stadium, coach) VALUES
  (101, 'Supa Strikas', 'Peter Mokaba Stadium', 'Samuel Cambra'),
  (102, 'Cradle Moons', 'FNB Stadium', 'Dawie Du Plessis'),
```

```
(103, 'Hungry Lions', 'Moses Mabida Stadium', 'Isaac Murinyu'),
(104, 'El Matadores', 'Sagrada Stadium', 'Peter Scott'),
(105, 'Twisting Tigers', 'Pearl Bay Stadium', 'Quinton Nkhoma'),
(106, 'Dancing Rastas', 'Strikaland', 'Adam Bethelson'),
(107, 'Green Bay Giants', 'Goliath Stadium', 'Kuhle Nkonki'),
(108, 'Grinning Smalls', 'Olive Heights Stadium', 'David Thompson'),
(109, 'Swallows', 'Hogwarts Stadium', 'Brian Godwin'),
(110, 'Ravenclaws', 'Draco Potter Stadium', 'Daniel Garcia'),

INSERT INTO Players (Player_ID, player_name, market_value, position) VALUES
(201, 'Lloyd Wright', '80000', 'Striker'),
(202, 'David Jardim', '750000', 'Mid Fielder'),
(203, 'Daniele Minidio', '900000', 'Goal keeper'),
(204, 'Anesu Mandengu', '500000', 'Defender'),
(205, 'Troy Bent', '440000', 'Defender'),
(206, 'Angelo Rube', '290000', 'Attack'),
(207, 'Gustavo Nienov', '500000', 'Goal keeper'),
(208, 'Jared Naidoo', '700000', 'Goal keeper'),
(209, 'Jireh Rusagara', '780000', 'Center back'),
(210, 'Kushinga Mangena', '990000', 'Forward'),

INSERT INTO Games (Game_ID, date, stadium) VALUES
(301, '2024-02-01', 'Peter Mokaba Stadium'),
(302, '2024-02-10', 'Moses Mabida Stadium'),
(303, '2024-02-17', 'FND Stadium'),
(304, '2024-02-29', 'Strikaland')
```

(305, '2024-03-08', 'Goliath Stadium')

(306, '2024-03-17', 'Hogwarts Stadium')

(307, '2024-03-26', 'Olive Heights Stadium')

(308, '2024-03-31', 'Sagrada Stadium')

(309, '2024-04-03', 'Draco Potter Stadium')

(310, '2024-04-07', 'Pearl Bay Stadium')

INSERT INTO GameResults (Game_ID, Player_ID, goals_scored) VALUES

(301, 201, 6),

(302, 202, 10),

(303, 203, 2),

(304, 204, 15),

(305, 205, 4),

(306, 206, 7),

(307, 207, 0),

(308, 208, 1),

(309, 209, 20),

(310, 210, 3),

Task 3: SQL Queries (14 Marks)

I. Write an SQL query that returns the top 10 players in terms of market value and the clubs they play for.

```
SELECT
    p.player_name,
    p.market_value,
    c.club_name
FROM
    Players p
JOIN
    Clubs c ON p.position = c.Club_ID
ORDER BY
    p.market_value DESC
LIMIT 10;
```

Task 4: Database Security and Access Control (20 Marks)

i. Discuss the use of database roles and privileges to secure a database system.

Since they restrict access to data and features inside the database, database roles and privileges are essential to the security of a database system. The following is an explanation of how they improve security:

Access Control: You can manage who has access to what information and can carry out specific actions within the database by using roles and privileges. By doing this, it is made sure that the database and its objects can only be accessed by authorized people.

Role-Based Access Control (RBAC): Roles enable users to be logically grouped according to their duties or job functions. Privileges are not allocated to specific users, but rather to roles, which are subsequently bestowed upon users. This improves manageability and streamlines administration, particularly in big database systems with numerous users.

Granular Control: Administrators can fine-tune access rights depending on requirements by granting privileges at a granular level. Privileges can be provided, for instance, on views, tables, stored procedures, or even specific table columns.

Least Privilege Principle: Database administrators should adhere to this principle, which states that they should only provide users with the smallest number of privileges necessary for them to carry out their duties. This lowers the possibility of sensitive data being accidentally misused or accessed without authorization.

Data Integrity: Database roles and privileges contribute to the preservation of data integrity by limiting access to certain data and processes and preventing unauthorized users from making inadvertent changes to the data.

Enforcement of Security Policies: Administrators can enforce security policies and compliance requirements within the database system by using database roles and privileges. For instance, some jobs might only allow read-only access to particular rights, protecting sensitive information from being changed by unauthorized individuals.

Dynamic Privilege Management: Administrators can give or remove privileges as needed by using a dynamic approach to managing database roles and privileges. Because of this adaptability, access permissions can be changed in response to evolving business needs or security risks.

Auditing and Monitoring: To keep tabs on user activity and make sure security regulations are being followed, database roles and privileges can be audited and observed. Audit logs can be used to spot unusual activity or unauthorized access attempts, which can aid in the detection and mitigation of security breaches.

Taking these factors into account, database roles and privileges are crucial parts of database security because they offer a strong framework for limiting access to data and safeguarding the confidentiality and integrity of the data kept in the database system.

Ii. Discuss the available grant options and how they enable the database administrator to control access. Provide examples of granting permissions on the database created in Tasks 1 and 2.

The database administrator can manage who has access to certain database objects and features by using one of the several grant options in Oracle Database. Among these grant choices are:

GRANT: The GRANT statement gives the database administrator the ability to provide users or roles particular rights on database objects. SELECT, INSERT, UPDATE, DELETE, EXECUTE, ALL rights, and more are among these rights.

WITH ADMIN OPTION: The grantee (user or role) who receives privileges with the WITH ADMIN OPTION clause has the option to extend those same privileges to additional users or roles. It can be helpful to divide out administrative responsibilities while keeping centralized control using this delegation of authority.

GRANT OPTION: This option lets the grantee provide additional users or roles the same privilege. It offers some flexibility in controlling database access permissions.

ROLE: Users can be bestowed with roles, which entitles them to acquire the corresponding privileges. As a result, privilege management is made simpler because roles are given privileges, and users are then awarded those roles.

SYSTEM PRIVILEGES: Rather than granting access to particular objects, system privileges provide extensive permissions that impact the entire database system. CREATE SESSION, CREATE TABLE, CREATE VIEW, CREATE PROCEDURE, and ALTER USER are a few instances of system privileges.

OBJECT PRIVILEGES: Permissions on particular database objects, like tables, views, sequences, and procedures, are granted by object privileges. SELECT, INSERT, UPDATE, DELETE, EXECUTE, and REFERENCES are a few examples.

Granting SELECT permission on the Players table to a user named 'user1':

GRANT SELECT ON Players TO user1;

Granting INSERT and UPDATE permissions on the Clubs table to a role named 'coach':

GRANT INSERT, UPDATE ON Clubs TO coach;

Granting SELECT permission on the Games table to a user named 'user2' with the ability to further grant the privilege:

GRANT SELECT ON Games TO user2 WITH GRANT OPTION;

Granting EXECUTE permission on a stored procedure named 'calculate_stats' to a role named 'stats_admin':

GRANT EXECUTE ON calculate_stats TO stats_admin;

Granting the 'coach' role to a user named 'user3':

GRANT coach TO user3;

Granting the 'stats_admin' role to a user named 'user4' with the ability to further grant the role:

GRANT stats_admin TO user4 WITH ADMIN OPTION;

Iii. Discuss the role of views in controlling database access.

Because views provide an abstract layer over the underlying tables, they are important for managing database access. They function as virtual tables that display information in a way that is specific to one or more tables. The following is how views help manage database access:

Selective Data Access: Views let database managers show users just a subset of a table's rows or columns. This implies that access to private or sensitive data kept in the underlying tables may be restricted for users.

Data Abstraction: By providing a streamlined and customized presentation of the data, views can conceal the intricacy of the underlying table structures. This makes it possible for users to engage with the database more successfully without having to comprehend all of the minute aspects of the database schema.

Column Masking: Views can be used to conceal some columns, such as email addresses or social security numbers, which are considered personally identifiable information (PII), while allowing users to view other non-sensitive data.

Joining Tables: Views can use joins to bring together data from several tables so that users can easily access related information without having to handle laborious join operations themselves. In addition to guaranteeing that customers receive consistent and correct data, this upholds data integrity.

Security Enforcement: By granting access to views rather than directly to tables, administrators can use views to enforce security regulations. This makes it possible to regulate access more precisely because permissions can be adjusted according to the description of the view.

Dynamic Data Filtering: Filters can be applied to views to limit the amount of data that users see to only that which meets predetermined standards. One way to display data is by creating a view that only displays records for a particular department, area, or time frame.

Query Simplification: By encapsulating intricate SQL queries, views enable users to access the necessary data more quickly and easily without having to create difficult queries themselves. This increases database usability and query performance.

Data Standardization: Views are a useful tool for ensuring consistency in data presentation across various user interfaces and applications. This lowers the possibility of data errors or conflicts and helps preserve data quality.

All things considered, views are an effective tool for managing database access since they give users a flexible and adaptable way to see data while upholding security guidelines and preserving data integrity. They are essential for maintaining database performance, safeguarding sensitive data, and ensuring that users have access to the appropriate information at the appropriate time.

Completed Declaration of Authenticity

I Gabriella Rakgotsoka _ hereby

(FULL NAME)

declare that the contents of this assignment PRG522_FA is entirely my own work except for the following documents: (List the documents and page numbers of work in this portfolio

that were generated in a group)

Activity	Date

Signature: Date: _2024/04/08