Inland Norway University of Applied Sciences

Databases and Network course

Mock exam (2h)

By. Dr. M.R.YAGOUBI

# I. Theoretical Questions

1. Explain the types of relationships in relational databases (one-to-one, one-to-many, and many-to-many) and provide an example of how each might be used in a game's database.
2. If we want to complement a relational structure with a NoSQL structure in multiplayer gaming, which NoSQL database is the most suitable and why?
3. Draw a simple diagram of asymmetric encryption and explain the main difference between symmetric and asymmetric encryption.
4. Explain the strength and drawbacks of NoSQL document oriented databases against the classical relational structure.

# II. Labs

## Game Development Scenario

We are developing a game with the following premise:

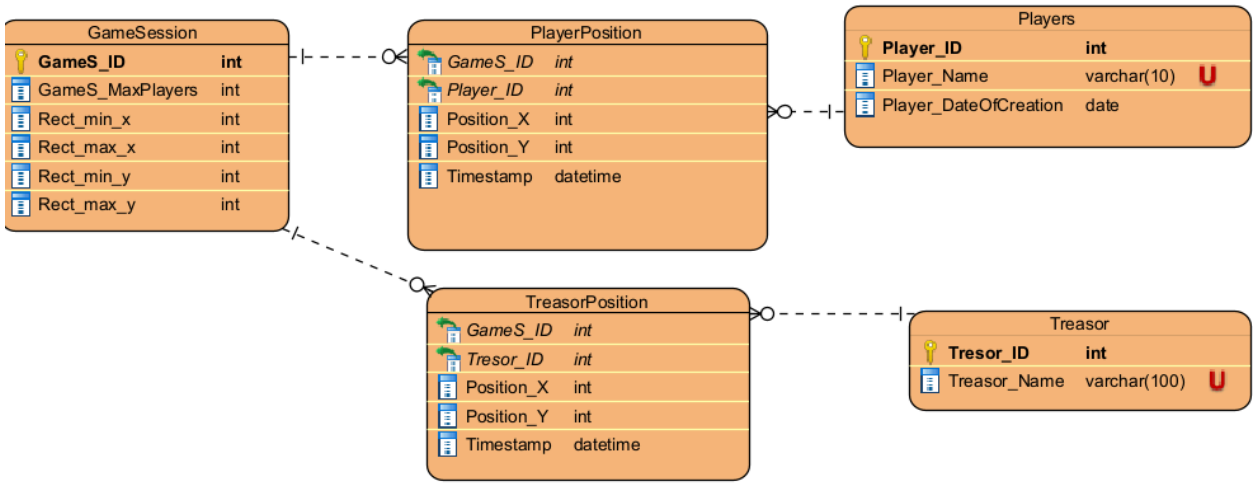
- **Session Initialization:** A session is initiated with defined rectangular boundaries and a maximum number of players allowed to join.
- **Treasure Placement:** Upon each session initiation, a treasure is randomly positioned within the boundaries.
- **Gameplay:** Players move within the area until one player finds the treasure, who is then declared the winner.

Our goal is to maximize computational efficiency by leveraging the database. Consider the following schema designed for SQL Server:

Inland Norway University of Applied Sciences

## Schema Overview

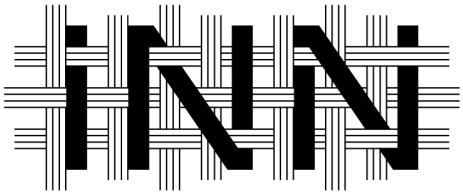
- **GameSession:** Defines each game session's boundaries (as a rectangle) and the player limit.
- **Players:** Stores information about individual players, including a unique identifier and creation date.
- **Treasure:** Stores information about each treasure, with a unique name and identifier. The treasure position is defined randomly after initiation of a new session.
- **PlayerPosition:** Tracks the position of each player in a specific game session. When a player changes position, an update is performed on this table.
- **TreasurePosition:** Defines the position of each treasure in a specific game session. The position of the treasure is set upon the initialization of a new GameSession.




## Data Query Language

1. **Write the correct SQL statement to:**

a. Show all players sorted in ascending order of their creation date (from oldest to newest).
b. Display the game session ID, the number of players involved, and the name of the winner.
c. Show all players' names and the number of sessions they have played so far.
d. Create a view that shows all players and the number of sessions they have won.
e. Create a view to show the datetime when the session is created (Starting time) and the datetime when the session is finished (Close time).
f. ***(Optional)*** Show the average area of rectangles defined in game sessions.



Inland Norway University of Applied Sciences

## B. SQL Transactional

1. Write a **transaction** inside an SQL **stored procedure** that creates a new session and initializes a random position for a treasure.
   *(Input parameters: `min_X`, `max_X`, `min_Y`, `max_Y`)*
2. Propose and implement a solution to save the entire history of players' positions.
3. Write a trigger on the `PlayerPosition` table that corrects a player's position if they move beyond the boundary.
   *For example, if `new_x > max_x`, then set `new_x = max_x`.*
4. ***(Optional)*** Write stored procedure that gives the average velocity of a player in a session (Assume that we store position on meter) *(Input parameters: `Player_ID, GameS_ID)`*
5. ***(Optional)*** Write a view that shows the average number of updates to player positions in each session until a player finds the treasure.

## C. NoSQL (Document-Oriented)

1. Write the collection data for `GameSession` with `ID=2` if we want to translate it from the relational schema to a NoSQL document-oriented schema.

**NB:**

```
SELECT FLOOR(RAND()*(10-5+1)+5); –Return a random integer between 5 and 10
```