Basics of database systems

Project – Database design

Lappeenranta-Lahti University of Technology LUT
Software Engineering

Basics of database systems
Spring 2023

Atte Hiltunen 0565218
Aleksi Levanen 0613649

TABLE OF CONTENTS

1 DEFINITION

Game database

In project “Game database” database is being used in game to hold data and database is accessed via python interface.

In the game there will be player, enemies, rooms Inventory and enemies in rooms. Player can carry 2 items in its inventory and enemies can carry one item only. Database is to hold data of the game and it can be accessed and changed with python interface.

Database hold information about players, their attributes, rooms, enemies and their attributes, players items and enemies in certain room. These all are useful information used in game.

Players attributes and items must be accessible so they can be used in combat of enemies. Room level will determine enemies’ levels and when can player get into that room.

Game progress is determined with rooms and player is always in one room and if he beats enemy and has level for next room he can continue to next room.

Following database queries are implemented:

1. List of all rooms players has been in (room level is smaller than player level)
2. Players with at least some progress on the game
3. List of equipment on all players
4. List of enemies not holding anything
5. List of enemies in rooms (many to many relationships and two inner joins is used in query)

These are quite self-explanatory, and they explain what they show.

2 MODELING

2.1 Concept model

In Figure 1 Is the ER model of the designed database. There are five entities (Player, Room, Item and Enemy) in the model and four relationships (Include, Has twice and Is in). There is one N:M relationship in between Room and Enemy. There is one-to-one relationship between Enemy and Item. And one-to-many relationships between Player and Room.

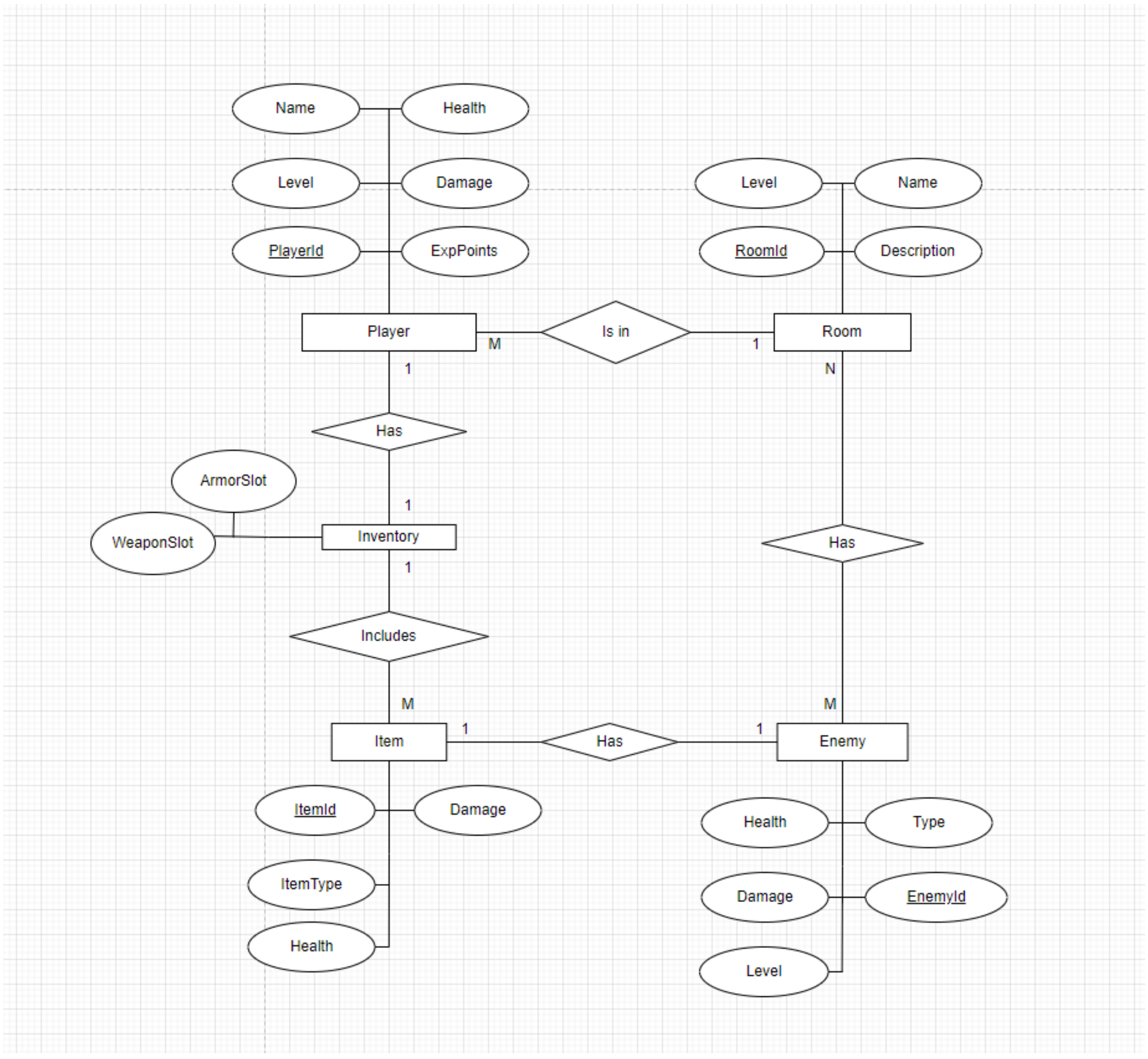

Figure 1: ER model

2.2 Relational model

Figure 2 shows the relational model has been created based on the ER model. Because of the N:M relationship, an interim relation was created between Rooms and Enemies. Inventory doesn't have primary key because every player has 1 inventory so you can find player inventory with his PlayerId.

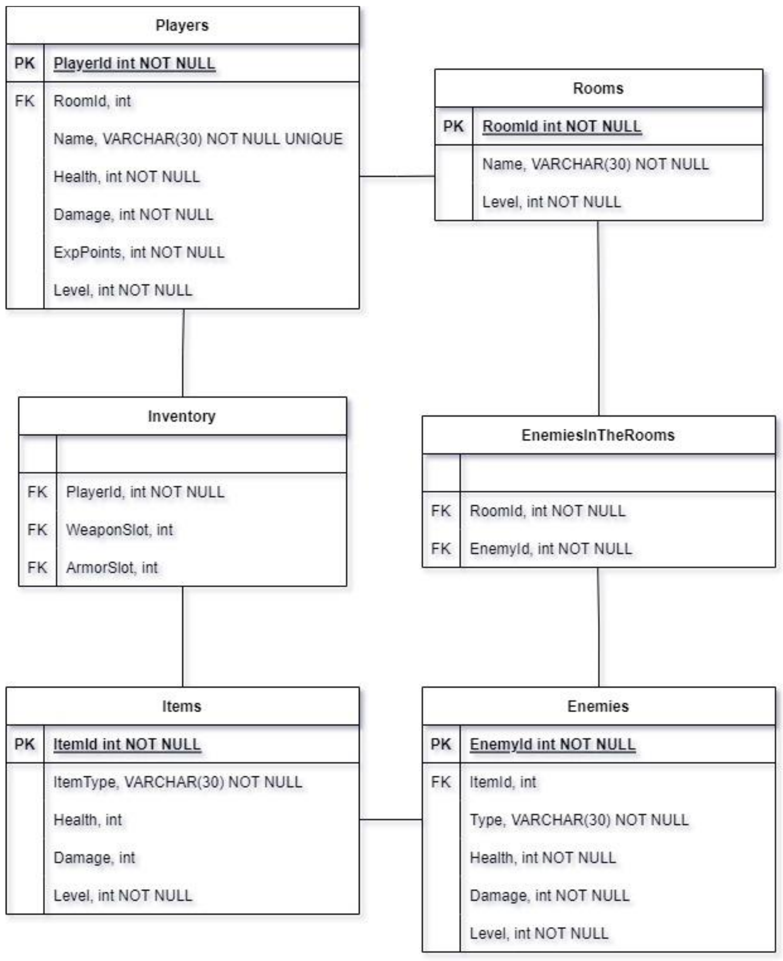

Figure 2: Relational model from the ER model

3 DATABASE IMPLEMENTATION

- **Player**
    - PlayerID, health, damage, experience points and level cannot be null (NOT NULL)
    - PlayerID has autoincreament. (AUTOINCREAMENT)
    - RoomId, Health, Damage, ExpPoints and Level all default to respective values if not given. (DEFAULT)
    - Unique key name that no player has the same name on character. (UNIQUE)
    - RoomID default to first room and id can't be null also check for player level to be same or higher with room level (DEFAULT, NOT NULL, CHECK)
    - Foreign key reference to Rooms
    - ON UPDATE CASCADE
- **Room**s
    - RoomID, name and level cannot be null (NOT NULL)
- **Enemies**
    - EnemyID, itemid, type, health, damage and level cannot be null (NOT NULL)
    - foreign key reference to Items
    - ON UPDATE CASCADE
- **Enemies in the rooms**
    - Room id and enemy id cannot be null (NOT NULL)
    - foreign key references to Rooms and Enemies
- **Inventory**
    - Foreign key reference to Player and Items
    - ON UPDATE CASCADE

- **Items**
    - ItemID, item type and level cannot be null (NOT NULL)

In addition to the integrity constraints listed above, the database will also implement two indices. One based on the ItemType from Items list, and another based on Name from Players list. These indices can be used for finding players or items from their tables by their name faster than before.

Python Interface:

1. See all rooms players have cleared.
2. Players with progress
3. Equipment on players
4. Enemies with nothing in hand
5. Enemies in each room
6. Search players stats with name
7. Insert, update or delete items
    a. Insert
    b. Update
    c. Delete
8. Quit (Option 0)

Selections 1-5 are queries presented in definition part. Users can print them with selection. Selection 6 is to search for a player from Players table and it prints the players' information for user. (PlayerId is left out on purpose, because it is only on database side)
Selection 7 is to modify players' information. Users can create a new player with an insert. Users can modify or delete an already existing player. Modifying option gives user format which has to be used so data gets updated properly.

4 DISCUSSION

Our database is going to be used in terminal type game which we are aiming to get done in the summer. We used the python template provided on moodle to get started with our interface. Python interface is done in a way that it should not crash on small errors and user can keep on using and can try again functions he wanted. Crash handling is done with try-except clauses and few if statements.