Basics of database systems

**Project – Database design**

Lappeenranta-Lahti University of Technology LUT

Software Engineering

Basics of database systems

Spring 2023

Atte Hiltunen

Aleksi Levanen

Table of contents

[Table of contents 1](#_Toc92697309)

[1 Definition 2](#_Toc92697310)

[2 modeling 4](#_Toc92697311)

[2.1 Concept model 4](#_Toc92697312)

[2.2 Relational model 5](#_Toc92697313)

[3 database implementation 6](#_Toc92697314)

[4 discussion 7](#_Toc92697315)

# Definition

Describe the need of your work, for what the database is developed. Describe the problem area in such detail that it can be modelled into a database and it can be used to critically compare the concept analysis during evaluation. Finally, describe the five queries/views you are going to create.

**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, enemy and rooms, also player can carry 2 items one armour and one hand weapon. 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.

Players attributes and items must be accessible so they can be used in combat of enemies. Room level will determine enemies’ levels.

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. Getting player stats with player name
2. List of all rooms player has been in (room level is smaller than player level)
3. List of all items in player inventory
4. Find rooms enemy and what item it is holding
5. List of enemies in rooms (many to many relationship and two inner joins is used in query)

**Example text: Staff database**

In project ‘Staff database,’ database is developed for a customer who manages a small company that deals with outsourced employees (provides employees to different companies). Within the database, the staff information is stored as well as the companies they are working with and the tasks they can perform to be able to offer the correct staff members to the right companies. The database also stores the information of the immediate family members in case of an emergency Important staff information are name, address, date of birth, and contact information.

It is also important to be able to find the start date of all staff memebers and be able to search based on a specific start date. In addition, the tasks the staff members can perform need to be readily available for quick use.

The following database queries have to be implemented: (1) List the information of a specific staff member, their tasks and work status. (2) List all staff members working for X company. (3) List all X staff members that can perform Y task(s). (4) Show the close family members of X staff member(s). (5) List the employees that have started working during a specific time period as well as the companies they are working in.

# modeling

## Concept model

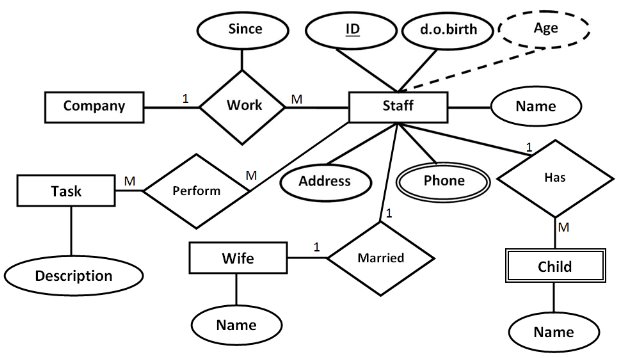
Database design begins with concept analysis. The use of concept analysis leads to database design decisions that are independent of the data and implementation.

At this point of the project, the aim is to describe the conceptual model of the database that has been developed. Use the ER-modeling notation.

Represent at least: Entities (concepts), relationships (the connection between concepts) and the cardinalities of the relationships (one-to-one 1:1, one-to-many 1:N, many-to-many N:M), and properties (attributes). Describe the ER model so that you point out the most important parts that may be altered or go through modifications during the transformation to a releational model.

**Example text:**

In Figure 1 is the ER model of the designed database. There are five entities in the model and four relationships. There is one N:M relationship in between Staff and Task. The Staff entity has a derived attribute ‘age’ which is removed during the database implementation. Additionally, there is a multivalued attribute ‘phone’ that may be reduced to a single value, multiple fields or an additional relation to accommodate multiple values.



**Figure 1:** ER model

## Relational model

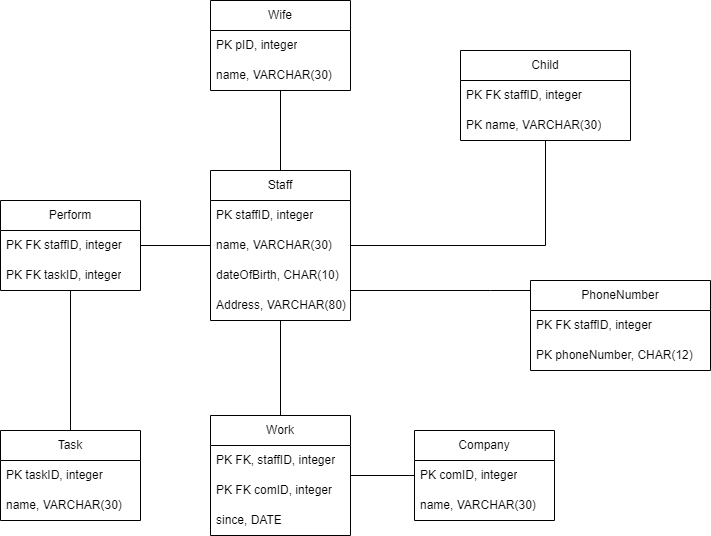
After the ER-modelling, transform it to a relational form using the appropriate transformation rules. Represent the result either as a relational diagram or a freeform UML-style table structure.

Each entity should have the necessary primary and foreign keys. The definition of accepted value range should be done for each attribute: length, type.

Relationships have their own integrity constraints when necessary. If a relationship has attributes, it is described as an entity.

**Example text:**

Figure 2 shows the relational model that has been created based on the ER model. Due to the N:M relationship, an interim relation was created between Staff and Task entities. The derived attribute ‘age’ was removed and the multivalued attribute 'Phone’ was separated into an additional entity. Finally, the ‘work’ relationship was created as a relation because of the related attribute.



**Figure 2:** Relational modelfrom the ER model

# Database implementation

During the implementation, you have to develop the different integrity constraints as well as indices. Describe the constraints and indices you have created for your database. You can decide what kind of format you use for describing them. The example shows each relation in a list and the constraints in them, you can use the same format or use tables or whatever seems best for you.

If you have created a Python interface, describe that here as well.

* **Player**
  + PlayerID, health, damage, experience points and level cannot be null (NOT NULL)
  + Unique composite key that no player has same name character with same ID
  + 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)

**Example text**

During implementation, the following constraints are created for the relations:

* **Staff**:
  + Name, date of birth and address cannot be null (NOT NULL)
  + Date of birth has to be at least 18 years ago (CHECK)
* **Wife**:
  + Foreign key reference to staff.
  + Name cannot be null (NOT NULL)
  + ON DELETE CASCADE
* **Child**:
  + Foreign key reference to staff
  + unique composite key of staffID and child name so that the same staff member doesn’t have children with the same name.
* **Perform**:
  + Foreign key reference to staff and task
  + ON UPDATE CASCADE
* **Task**:
  + Name cannot be null (NOT NULL)
* **Work**:
  + Foreign key reference to staff and company.
  + Date cannot be null and defaults to current date (NOT NULL, DEFAULT)
  + ON UPDATE CASCADE
* **Company**:
  + Name cannot be null (NOT NULL)
* **PhoneNumber**:
  + Foreign key reference to staff
  + unique phone number so that there aren’t any others with the same phone number. (UNIQUE)
  + ON UPDATE CASCADE
  + ON DELETE RESTRICT

In addition to the integrity constraints listed above, the database will also implement two indices; One based on the Task name, another based on the Work since. These indices are to allow quickly search who perform the same tasks as well as to search for employees that have started working during a specific time period.

# discussion

If you want to mention something that has not been discussed in the previous chapters, you can discuss them here.

**Example**

Nothing to discuss.