**Homework Assignment 2:**

In the first few HWs you will have an opportunity to apply the ideas and concepts of relational databases that you will learn in this course. You will need to design and implement a real-life relational database for a fictional company. In this process, you will first define company's business requirements and then design the database schema, showing in detail all the relations (tables), data types, relationships, and constraints to be used in the database. After the design phase is over, you will implement the schema in Oracle by creating the required tables with appropriate data types together with relationships and constraints, and populate all the relations with valid data. Next, you will use your database to retrieve required data using complex SQL queries.

In HW2, you need to include the following:

* A short description of the company (you can use just one department or business process if the company is large)

This is a website for coaching soccer. We know soccer is not a 90 minutes game. It needs lot of dedication and technique by individual and by teamwork. This website will contain the motion pictures on drill in different category. It would contain subscriber at different stages. It has lot of classification.

* Reasons why the company needs a database

First of all, this website contain subscriber with different skill set. A player can register for different type as like Goal-keeper, defense, Stricker. Every cataMoreover, it has lot of motion pictures which are categorized in different panel. It would become messy if it is not organized in a database.

* Detailed business requirements (business rules) that describe all entities, relationships, and constraints from the business perspective          (Note: **Do not include more than 4-5 entities** as it is your first draft! It is very important to design a database correctly and it is very difficult to do so with a large number of entities. Your database will be revised in the following HWs and you will have a chance to make it more complicated later on if needed.)

The goal of this website is to provide training for the player. A player can hesitate at the very beginning level about his playing position. As like, a player can be interested in either goal-keeper or defense or striker. Again, a player can be interested in all three types. Every type consists of three different level as like beginner, intermediate, and expert level. For better understanding the scenario, think about a player ‘John’, age 14. This player can be enrolled as Goal-keeper Beginner Level, Defense Expert Level and Striker Intermediate level simultaneously.

For every category of playing, there would be drill set for dribbling, passing and shooting. For instance, John has already enrolled for Goal-Keeper Beginner Level. This category can be consisting of 2 dribbling drills, 1 passing drills and 5 shooting drills.

In a summery, a player can enroll among 9 categories (Goal-Keeper, Defense, Stricker with Beginner, Intermediate, Expert level) as much as he wants. Every category consists of drill from 3 sets (Dribbling, Passing, Shooting). A drill can be only one type either dribbling or passing or shooting. One drill can not be part of two different sets.

Based on the business scenario, this website would contain many entities. I hope to come up with a perfect set of entities though out this course by brainstorming more and more. As an initiative step, I have figured out three entities. They are Player, Category and Drill.

A detailed description of each entity including the name, purpose, and attributes

In previous paragraph I have discussed the business scenario and the cause of choosing the entities. If we want to have a detail version of the entities, then we can end-up as like following:

PLAYER:

This entity is the very beginning step of this web for user-interaction. A user will open his account. Every player will posses an unique id. He can choose one or more category id among those 9 category\_ids. He will not be allowed to choose other than those.

player\_id: A unique id choose by user. It would be VARCHAR. Maximum 10 character will be allowed.

email: Mandatory attribute as it is necessary to contact with player. Type: VHARCHAR.

age: Number type

CATEGORY:

Every category would have id (1~9). For more detail description it would contain the attribute player\_type and player\_level. Moreover, it would also contain player\_id and drill\_id.

category\_id: (1~9) (Category Chart has been attached here)

player\_type: (Goalkeeper/Defense/Stricker)

player\_level: (Beginner/Intermediate/Expert)

|  |  |
| --- | --- |
| 1 | Goalkeeper-Beginner |
| 2 | Goalkeeper-Intermediate |
| 3 | Goalkeeper-Expert |
| 4 | Defense-Beginner |
| 5 | Defense -Intermediate |
| 6 | Defense -Expert |
| 7 | Striker-Beginner |
| 8 | Striker -Intermediate |
| 9 | Striker -Expert |

DRILL:

For drill, it should have unique id and type. It will also have duration of every drill in seconds.

drill\_id: VARCHAR (10)

drill\_type: (Dribbling/Passing/Shooting)

drill\_duration: number

PLAYERCATDRILL:

It is very difficult to implement many-to-many relationship in case of relational database. So, for making the relations among those three entities this PLAYCATDRILL entity is necessary. It has three attributes whose is not only as primary key but also as foreign key.

player\_id: VARCHAR

category\_id: (1~9)

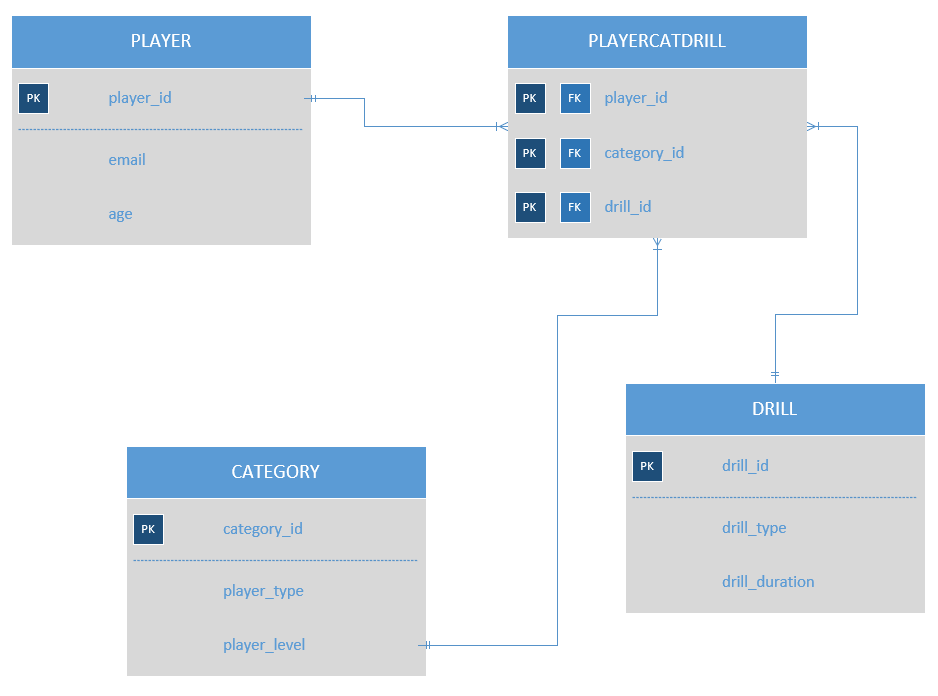
drill\_id: VARCHAR

* Description of relationships between entities (make sure to evaluate each relationship in both directions)

I would like to depict the relationship in a table with evaluation.

|  |  |  |  |
| --- | --- | --- | --- |
| Entity 1 | Entity 2 | Relationship | Cause |
| PLAYER | CATEGORY | Many-to-Many | A player can be enrolled in many categories again one category can contain many players |
| PLAYER | DRILL | Many-to-Many | A player can practice many drills and one drill can contain many players |
| DRILL | CATEGORY | Many-to-Many | Every category is build with drill set of many drills. Similarly, every drill can be part of many categories. |
| PLAYER/CATEGORY/DRILL | PLAYERCATDRILL | One-to-Many | One instance from any of these three entities (PLAYER/CATEGORY/DRILL) is related to many instances of PLAYCATDRILL entity. |

* Entity relation diagram:



* A list of different users for the database you proposed. Define user categories and describe different types of applications each user would need. Explain what type of interface each user would need.

As it is a web application it can have three types of user.

* 1. App-User: They are the end user of this system. But an immutable design will be applied for them so that they cannot change any data. They can only use the online drill for training purpose.
  2. App-Developer: They are responsible for making the interaction between end-user and the system database. They will develop the program so that user can access the database and use the data. But this App-Developer are not allowed to modify the schema.
  3. Database-Administrator: They have the rights to change the data. This type of account cannot connect from the application. They are the developer account. They can do everything including changing the schema.
* Explain which DBMS architecture would you choose from section 2.5 of the Fundamentals of Database Systems textbook and why. Why would the other architectures not be a good choice?

I choose three types of user for this system. And there are many business logic based on the player’s category. Everybody is not allowed to access all the drills. Players are allowed to access specific drill based on their enrolled categories. So, for maintaining the business logic we need to apply mid layer between end user and the database. So, our architecture would be a three-tier architecture. Where App-user are the client and Web-server, or application developed by the App-developer would be the mid layer and Database server would be the last layer of this system. I do not prefer to give access the client directly to the data as there is many logic involves. So, I think three layer architecture is better than others.