# Project: Sprint 3 - Database design and implementation

## General instructions:

This assignment must be performed in assigned project teams. You must update your project report by adding the sections relevant to the tasks below, using this (updated) [**project report template**](https://docs.google.com/document/d/1FHyQMjngGzt9kDd1qTOheFvQYS7qtCqW-FL3XW4_pEA/edit?usp=sharing).

Do not jump to the implementation or even the logical design and then reverse engineer your conceptual design! Work in the order specified.

Ensure that all entities/attributes/tables/columns are named in a meaningful way, using a consistent convention.

### *Sprint activities:*

### Part 1: Refine requirements [*try to complete this within the first 2-3 days of the Sprint*]

* If you still have user stories left to complete, choose a small subset of them to be included in Sprint 3.
* Refine your chosen subset of user stories as before.

See [project report template](https://docs.google.com/document/d/1FHyQMjngGzt9kDd1qTOheFvQYS7qtCqW-FL3XW4_pEA/edit?usp=sharing) for format.

|  |  |
| --- | --- |
| US1 | As a <Manager> I want to <create> a Fantasy Team for UEFA Champions League Tournament |
| US2 | As a Fantasy Team we will <gain> fantasy points from player statistics |
| US3 | As a Fantasy Team we will <earn> fantasy points from live matches in real life |
| US4 | As a <Player> I will help my team<earn> Fantasy Points by <making saves>. |
| US5 | As a <Player> I will help my team <earn> Fantasy Points by having <Shots on target>. |
| US6 | As a <Player> I will help my team <earn> Fantasy Points by <scoring goals>. |
| US7 | As a <Player> I will help my team <earn> Fantasy Points by <making assist>. |
| US8 | As a <Manager> you can search<player positions> to look up players to buy |
| US9 | As a <Player> background information about my <country > will be revealed to the public |

### Part 2: Perform conceptual design

Derive an updated conceptual design to accommodate the stories to be considered in this sprint and/or correct issues from previous sprints. You may change existing components of your original conceptual design or add new components. As part of this, do the following:

* Concretize the list of entities. For each entity, identify attributes and determine their types (i.e., simple / composite, single-valued / multi-valued, stored / derived). Some entities may have natural primary keys. Identify such primary keys.
* Concretize relationships among the entities identified above. For each relationship, identify cardinality and participation constraints. Only include relationships that will require some form of representation in your database.

See [project report template](https://docs.google.com/document/d/1FHyQMjngGzt9kDd1qTOheFvQYS7qtCqW-FL3XW4_pEA/edit?usp=sharing) for format.

Entity: **Managers**

Attributes:

Manager Name

Entity: **Fantasy Team**

Attributes:

Team Code

Player Names

Team Points

Team Budget

Entity: **Player Statistics**

Attributes:

Shots on target points

Assist points

Saves points

Player points

Fantasy points

Player Code

Goals Scored points

Entity: **Players**

Attribute:

Team Id

Player Price

Player Name

Player Code

Player Country

Entity: **Player Positions**

Attribute:

Goalkeeper

Forwards

Position Id

Entity: **Country**

Attribute:

Country Id

Country

Entity: **Player Country**

Attribute:

Player Code

Country Id

Relationship: **Manager** creates **Fantasy Team**

**Fantasy Team** buys **Players**

**Player Statistics** creates **Fantasy Points**

**Players** are added to **Fantasy Team**

**Player Statistics** boost position of **Fantasy Team** points

**Manager** searches for specific players **Player Position**

**Manager** can look up **Player’s Country**

**Player** is from a **Country**

Cardinality:

<One > to <One >

<One > to <Many>

<Many> to <Many>

<Many> to <One >

<Many> to <One >

<One > to <One >

<One > to <Many>

<Many> to <One >

Participation:

Managers has <partial > participation

Fantasy Team has <total> participation

Players has < partial > participation

Players Statistics has <total> participation

Player Position <partial > participation

Country <partial > participation

### Part 3: Perform logical design, with normal form identification

Update your logical design (i.e., relational model) to match the updated conceptual design derived in [Part 2](#_6k50oc4b9q9g) above. As part of this, do the following:

* Map all the entities (both strong and weak), attributes and relationships that you have identified so far into relational tables.
* For each table
  + indicate the **primary key**;
  + indicate all **foreign keys** and mention what attribute of which table each foreign key refers to.
* In situations where you have multiple options for mapping (e.g., foreign key approach versus cross-reference approach), briefly **justify** your choice.
* For every new or updated table in your logical design, derive and specify the highest level of normalization. If the level is below fourth normal form (4NF), either normalize the table or justify why you choose to leave the table below 4NF. *You are not required to check for normalization levels higher than 4NF*. [**This**](http://www.rlvision.com/blog/method-for-determining-candidate-keys-and-highest-normal-form-of-a-relation-based-on-functional-dependencies/) resource may be useful to you when determining what the highest level of normalization is.

Table: **Managers**

Columns:

pk\_1:

Manager’s Name

No index

Table: **Fantasy Team**

Columns:

pk\_2: Team Code

Player Name’s

Team Budget

Team Points

Player Country

*Justification:* Team Code is referencing Players so that players are on a team earning points.

The index that should be created is Players Name. The index will be non-clustered. The reason for using a non-clustered is because the index columns are in not in order.

Third Normal Form – the table is in second normal form and all of its columns are not transitively dependent on the primary key

4NF

Table: **Player Statistics**

Columns:

pk\_2: Player Code

Shots on target Points

Assist Points

Saves Points

Goals Scored Points

Fantasy Points

Player Code [foreign key; references **Player Code** of **Players**]

The index that should be created is Players Code. The index will be non clustered. The reason for using a clustered is because the index columns are not repeating values

*Justification:* Player Code is referencing Fantasy Team, Players and Player Positions and so that is used to add the points to the Fantasy Points in the Table entity.

2NF:

The table is in first normal form and all the columns depend on the table’s primary key.

Adding indexes on columns

Table: **Players**

Columns:

pk\_2: Player Code

Player Name

Player Price

Player Country

Team Id: [foreign key; references **Team Code** of **Fantasy Team**]

The index that should be created is Players Price. The index will be clustered. The reason for using a clustered index is because the index columns has some repeating values

*Justification:* Player Code is referencing Player Statistics and Fantasy Team and Player Positions and Player Country so a name and code is attached to statistics retrieved.

2NF:

The table is in first normal form and all the columns depend on the table’s primary key.

Table: **Player Positions**

Columns:

Pk 2:

Goalkeepers

Forwards

Position Id [foreign key; references **Player Code** of **Players**]

The index that should be created is Players Id. The index will be non-clustered. The reason for using a non-clustered index is because there are no repeating values.

4th Normalization Form

Table: **Country**

Columns:

Pk 2: Country Id

Countries

*Justification:* Country Id is referencing Player Country table so that countries of players can be tracked down.

2NF

The table is in first normal form and all the columns depend on the table’s primary key.

The index is countries, and it is clustered because it can repeat values.

Table: **Player** **Country**

Columns:

Pk 2:

Player Code [foreign key; references **Player Code** of **Players**]

Country Id [foreign key; references **Country Id** of **Country**]

*Justification:* No primary key

2NF

The index is non-clustered because values are not unique and can’t be repeated.

### Part 4: Indexes

For each table in your logical design, identify what indexes should be created (*material on indexes will be covered in the week of 11/19, so wait until then to work on this part*). For each index that you choose to create:

* Specify whether the index will be **clustered** or **nonclustered**.
* Briefly justify your choice. Since indexes are primarily used to speed up query performance, index design is done based on expected queries. So, in your justification, you must essentially explain what types of queries are common/most likely on a particular table/set of tables and justify your index choices in that context.

Add index details to your **logical design** using the format shown in the [project report template](https://docs.google.com/document/d/1FHyQMjngGzt9kDd1qTOheFvQYS7qtCqW-FL3XW4_pEA/edit?usp=sharing).

**View**: <doublePoints >

Goal: <What the view contains is the Assist Points column from the Player Statistics table. What the users would use this for is to select all rows from the Player Statistics table where Assist Points are located.

**View**: <budget >

Goal: <What the view contains is the Players Price column from the Players table. What the users would use this for is to select all rows from the Players table where Player Price are located.

**View**: <playerstats >

Goal: <What the view contains is the Shots On Target column from the Player Statistics table. What the users would use this for is to select all rows from the where players have shots on target points located.

### Part 5: Stored programs and views

As you have already seen from [this discussion](http://www.mysqltutorial.org/introduction-sql-views.aspx) on views, views present several advantages (simplify complex queries, limit data access, provide security, etc.).

As you have also learned, it is possible to use stored programs (i.e., stored procedures, stored functions, triggers and events) to perform some repetitive / automatic / scheduled operations on your database.

As part of this sprint, you must identify and write **views** and **stored programs** that are relevant to your application. *Note: in Sprints 1 and 2, you were asked to write key SQL queries. Consider converting these queries into views or stored programs* ***when relevant***.

Demonstrate the working of your views/stored programs from your user interface (*a simple button to trigger the view/stored program is sufficient - there is no need to implement actual features*).

//\*change to dynamic and not directly

**View**: <playerprofile>

Goal: <What the view contains is the Assist Points column from the Player Statistics table. What the users would use this for is to select all rows from the Player Statistics table where Assist Points are located.

**Stored procedure**: < Statistic>

Parameters: < playerCode, IN >

Goal: <What the store procedure does is runs the query in which the keyword is stated in the call and the user calls. The users calls the procedure Statistics with a pass in 2 in the parameter. 2 represents the Player Code of Players Table. >

**Stored procedure**: < Biography>

Parameters: <birthplace, IN >

Goal: < What the store procedure does is runs the query in which the keyword is stated in the call and the user calls the Biography procedure with “Brazil” passed in the parameters so that player names are selected for whoever is from Brazil>

See [project report template](https://docs.google.com/document/d/1FHyQMjngGzt9kDd1qTOheFvQYS7qtCqW-FL3XW4_pEA/edit?usp=sharing) for format.

### Part 6: Implement and deploy database

Modify your database to reflect all changes made in this sprint. In other words:

* Create (or update) new (or existing) tables according to your updated, normalized logical design.
* In all tables (old and new):
  + ensure that appropriate data types are chosen for all columns;
  + identify and set up any NOT NULL and UNIQUE constraints;
  + set up its primary key;
  + set up all foreign keys;
  + create all the indexes identified.
* Populate your tables with relevant, meaningful data, updating old data as needed.
* Create all your views and stored programs within your database.

Note: As before, **do** **not** attempt to manually update your Sprint 2 database. Instead,

* export your Sprint 2 schema as a SQL script;
* make updates to the script to reflect the updated tables/data;
* create a new Sprint 3 schema;
* import your updated script into this new Sprint 3 schema;
* add views/stored programs as needed;
* add indexes as needed.

## Submission:

Create a **compressed** (.zip) file containing the following:

* your updated project report, adhering to the updated [project report template](https://docs.google.com/document/d/1FHyQMjngGzt9kDd1qTOheFvQYS7qtCqW-FL3XW4_pEA/edit?usp=sharing);
* a MySQL dump of your database.

**One** member of each project team must submit the zip file via **Canvas**.

The primary key will be an index by default.

Do you want to create another index that also needs to use the primary key as the index? No. It is not necessary, since the primary key is an index by default.

You can refer to our slides. You could simply write and execute a SQL statement

CREATE INDEX index\_name ON table\_name(column\_name);.

Or you could use the index tab of MySQL workbench by

moving your mouse to table name in the schema panel

clicking the information button

select a column