**ASSIGNMENT 1**

|  |  |  |  |
| --- | --- | --- | --- |
| **Qualification** | **BTEC Level 5 HND Diploma in Computing and Systems Development** | | |
| **Unit number and title** | Unit 04: Database Design & Development | | |
| **Assignment due** |  | **Assignment submitted** |  |
| **Learner’s name** | Nguyễn Trọng Duy | **Assessor name** | Đỗ Duy Thảo |
| **Learner’s ID** | GCD17313 | **Submission number** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Learner declaration:**  I certify that the work submitted for this assignment is my own and research sources are fully acknowledged. | | | |
| **Learner signature** | Duy | **Date** |  |

**Grading grid**

|  |  |  |
| --- | --- | --- |
| P1 | M1 | D1 |
|  |  |  |
|  |  |  |

# Table of content

Contents

[Table of content 1](#_Toc106629489)

[Example of thing need Database ● The image illustrates the invoice / voucher that the App will manage● 3](#_Toc106629490)

[A. Overview of the Problem 3](#_Toc106629491)

[1. Data management tools and techniques 3](#_Toc106629492)

[2. About this project 7](#_Toc106629493)

[B. User Story 7](#_Toc106629494)

[\*\*Managers and Employees 7](#_Toc106629495)

[\*\*As a football team 7](#_Toc106629496)

[A. Analyze the requirements 8](#_Toc106629497)

[1. ERD Diagram 8](#_Toc106629498)

[Relationship: 9](#_Toc106629499)

[Figure 1: Entity Relationship Diagram of Soccer Match Management 9](#_Toc106629500)

[2. Table Description 9](#_Toc106629501)

[Figure 2: Players table 10](#_Toc106629502)

[Figure 3: Example of the Players table 10](#_Toc106629503)

[Figure 4: Clubs table 11](#_Toc106629504)

[Figure 5: Example of the Clubs table 11](#_Toc106629505)

[Figure 6: Matches table 11](#_Toc106629506)

[Figure 7: Example of the Matches table 11](#_Toc106629507)

[Figure 8: Stadiums table 12](#_Toc106629508)

[Figure 9: Example of the Stadiums table 12](#_Toc106629509)

[3. Database Analysis 12](#_Toc106629510)

[Table 1: Data information of Players table 12](#_Toc106629511)

[Table 2: Data information of Clubs table 13](#_Toc106629512)

[Table 3: Data information of Matches table 13](#_Toc106629513)

[B. Database design with explanations 14](#_Toc106629514)

[1. Database diagram 14](#_Toc106629515)

[D. Data validation 14](#_Toc106629516)

[E. Review whether the database is normalized 24](#_Toc106629517)

[Target 25](#_Toc106629518)

[Table 6: The second normal form (2NF) 28](#_Toc106629519)

[F. Code 28](#_Toc106629520)

[1. Create 28](#_Toc106629521)

[2. Insert values 29](#_Toc106629522)

[SQL Server 30](#_Toc106629523)

[Purpose of using SQL Server 31](#_Toc106629524)

[Components of SQL Server 31](#_Toc106629525)

[SQL Server versions 31](#_Toc106629526)

[Instance of SQL Server 32](#_Toc106629527)

[Benefits of installations 33](#_Toc106629528)

[SSMS 33](#_Toc106629529)

[MS SQL Server Management Studio interface when opened 35](#_Toc106629530)

[Database without Normalization 35](#_Toc106629531)

[1NF (First Normal Form) Rules 35](#_Toc106629532)

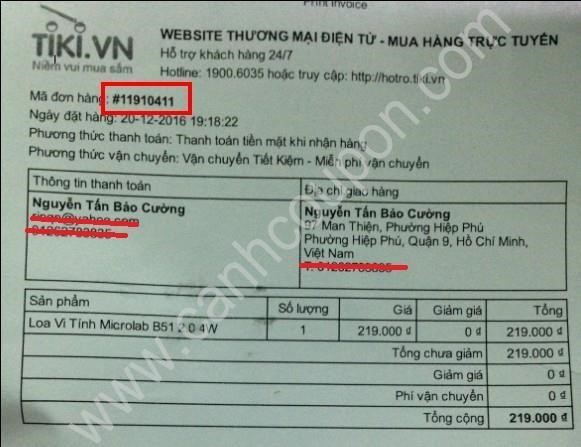
[2NF (Second Normal Form) Rules 36](#_Toc106629533)

[3NF (Third Normal Form) Rules 39](#_Toc106629534)

[Graphical Query Designer User Interface 45](#_Toc106629535)

[Command Type Text 45](#_Toc106629536)

|  |  |
| --- | --- |
| **Assignment Brief** |  |
| ElectroShop is a company who acquire and sell electronic devices to customers throughout Vietnam. They are looking to apply their current data store system with a relational database.  The company takes orders from customers, who can order any quantity of many items that ElectroShop have in their current catalogue. The catalogue includes TV, Phone, etc.  Each of ElectroShop’s suppliers supplies many items but each item is supplied by one main supplier.  ElectroShop would like to add all the customers to the database so that they can send a mailshot to them with any offers that are available, as well as the catalogue which is produced annually.  Salespersons at ElectroShop are paid a monthly bonus which is determined by the amount of sales they have made for each month. The company would like the bonus calculated |
| automatically by the system.  As well as the monthly bonus figures ElectroShop would like to be able to create comparative reports from the system i.e. sales figures for each month this year compared to last year. | |



# Example of thing need Database ● The image illustrates the invoice / voucher that the App will manage●

**I. Statements of user and system requirements (P1)**

## A. Overview of the Problem

### 1. Data management tools and techniques

**File Management System:**

A file management system is an application that's wont to administrate a system that stores, organizes, and accesses files stored on a disk or other memory device.

A file manager's primary function is to enable users to make, edit and save new files on a tool (laptop or desktop), examine all files that stored on the device, and arrange files by type. Folders, for instance, are easily categorizeable hierarchical structures.

The following examples are the elemental operations that a file management system may perform:

-Make a brand new document.

-Transfer files from one location to a different

-Show all files that are archived

-Basic metadata is also added and edited.

-Sort files by modification date, creation date, file size, file type, and other more factors.

**Network Data Model:**

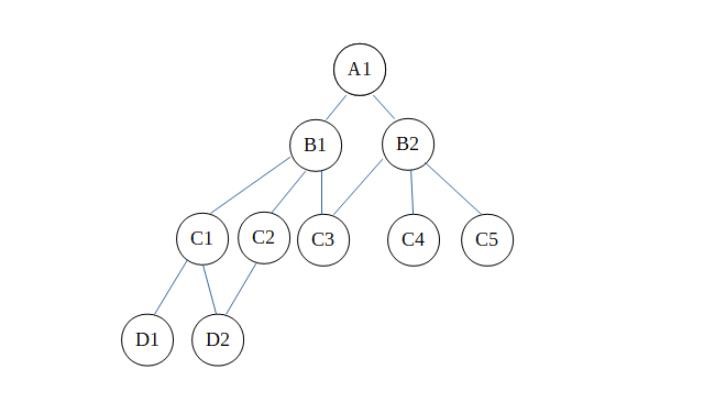
The Database Task Group defined this paradigm within the 1960s.

The hierarchical model could be a generalization of this approach.

To address the shortcomings of the hierarchical database model, the network database model was created.

This model can have many parent segments, which are divided into levels, yet there's a logical relationship between any level's segments.

Between any two segments, there's usually a many-to-many logical relationship. Graphs are made from logical connections between segments. As a result, this model substitutes a model graph-like structure for the hierarchical tree, yield more broad connections between nodes. (https://www.tutorialspoint.com/Network-Data-Model, n.d.)



Also:

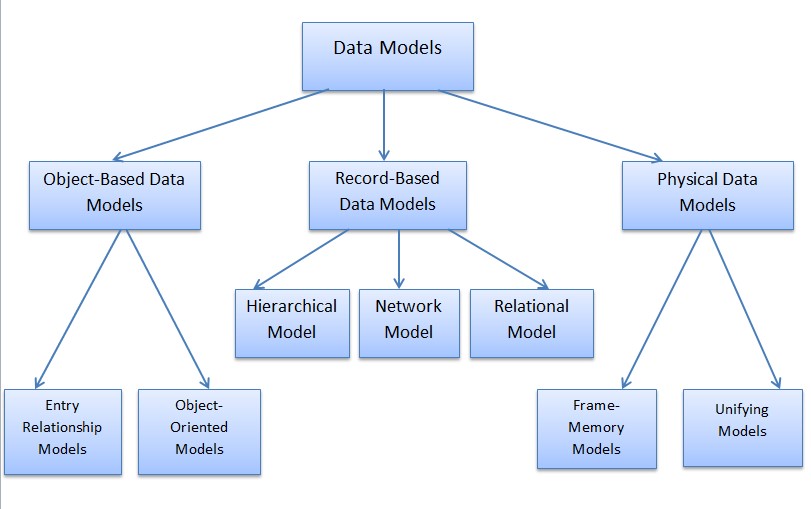
It can have a M:N connection meaning many-to-many relationship, which allows a record to own several parent segments.

A relationship is spoken as a collection during this context, and every set consists of a minimum of two of the subsequent forms of records:

-In a hierarchical paradigm, member records are like child records.

-In the hierarchical model, the owner record is that the same because the parent record.

(https://webeduclick.com/comparison-between-hierarchical-model-network-model-and-relational-model/, n.d.)



Disadvantages of Network Model:

Although it's regarded an improvement over the decentralized approach, the network model has significant drawbacks.

-The Network Model's Design & Structure don't seem to be user-friendly and are extremely complicated, therefore a programmer must thoroughly grasp it before implementing or modifying it.

-The Schema and Network Model is considerably more complicated than the Hierarchical Model.

-Because all records are preserved so far thanks to the usage of pointers for navigation, there are anomalies within the operation. As a result, it's difficult to handle and maintain, leading to complicated deployments.

-The Network approach, while more versatile than the Hierarchical model, still has limitations with flexibility.

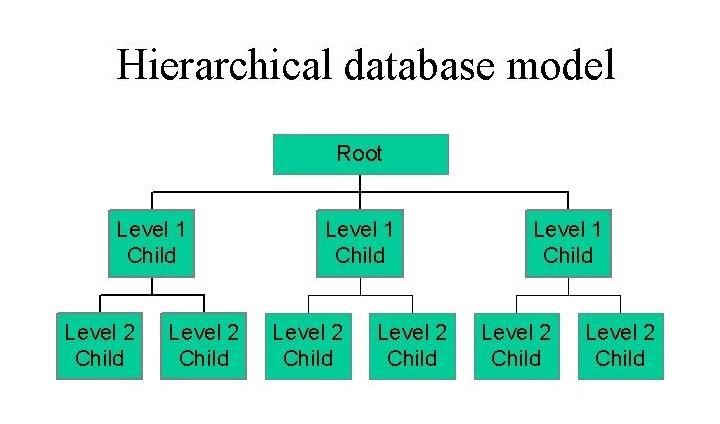
There is no automated query optimization coverage during this model.

Not all relationships is managed by using the owner and member attributes. (2018, Castro)

- Although the network database model is capable of obtaining data independence, this model doesn't accomplish structural independence.

**Hierarchical Data Model:**

(https://dataintegrationinfo.com/hierarchical-vs-relational-database/, n.d.)i



Entities are represented hierarchically during this data architecture, because the name implies. This pattern is represented by folders within folders! The model for The hierarchical database structure dictates that it's defined by its tree-like organization, within which there's typically a root "parent"

directory of knowledge stored as records that links to varied other subdirectory branches, and every subdirectory branch, or child record, may link to numerous other subdirectory branches. While a parent record can have multiple child records, each child record can only have one parent record. Fields are wont to hold data in records, and every field can only have one value.

In order to urge hierarchical data from a hierarchical database architecture, you want to first traverse the tree from the basis node. Impact: The hierarchical database model is most fitted to use cases where the first emphasis of knowledge collection is centered on a transparent

hierarchy, like multiple individual workers reporting to one section. This approach has some the subsequent disadvantage: -Missing updates anywhere will end in erroneous data. The hierarchical model somewhat solves this sort of redundancy. the matter of flat-file

redundancy is solved since the entries are organized in an exceedingly relevant table. this can be a case of information duplication and, as a result, redundancy.

-As a result, this model doesn't significantly minimize the redundancy problem. It causes redundancy and ambiguity. If we want to retrieve any data from this model, we must start at the highest and work our way down until we get the specified outcome.

**The relational model (RM):**

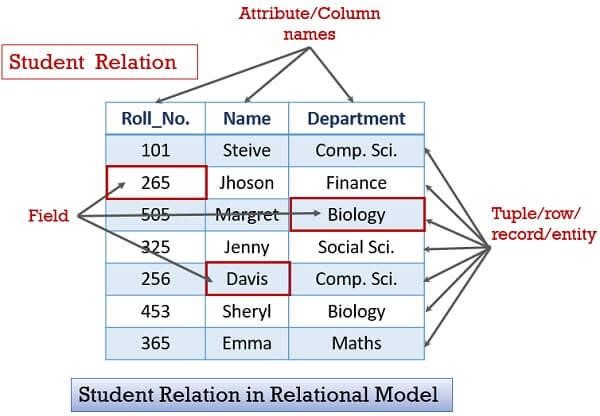
The Relational Model depicts how data is kept as a set of relations in an exceedingly on-line database.

A table of values is all that a relation is. Each table row represents a gaggle of connected data values. The table's rows each reflect a real-life entity or connection.

Data is stored in relational databases within the style of relationships (tables) with attributes.

The titles of the tables and columns help to elucidate what the numbers in each row signify. a group of relationships is employed to represent data. Tables are accustomed hold data within the relational paradigm. The physical storage of information, on the opposite hand, is unrelated to how the information is conceptually arranged.

(https://binaryterms.com/relational-data-model.html, n.d.)



**Big data management:** A lot of information due to its capacity to store and manage many alternative types of data

NoSQL databases management is usually employed in big data deployments to consult with the efficient processing, organization, or use of giant amounts of structured and unstructured data belonging to a company.

**Data warehouses and data lakes:**

Data warehouses and data lakes are two different data warehouses for handling analytical data.

Data Lake stores all data, no matter source or structure, whereas Data Warehouse stores data in numerical form with qualities.

A Data Lake may be a large collection of structured, semi-structured, and unstructured data, whereas an information Warehouse could be a set of technologies and components that allows data to be used strategically.

 **When working with data we should pay attention to the following:**

* Data integration:

the foremost common data integration approach is extract, transform, and cargo (ETL), which involves pulling data from a source system, converting it to a homogenous format, and so loading the combined data into an information warehouse or other destination system.

* Data modeling:

Data modelers build a spread of conceptual, logical, and physical data models so as to visually describe data sets and workflows and connect them to business needs for transaction processing and analysis.

* Data governance, data quality, and MDM:

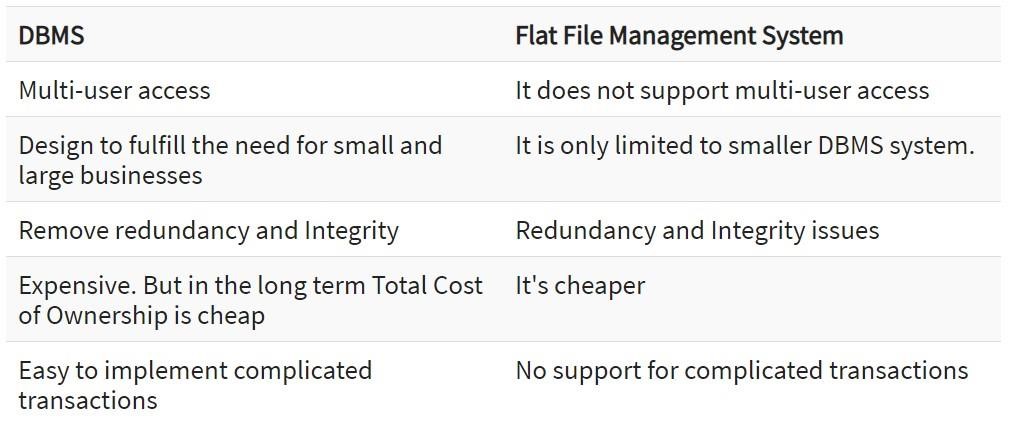
Data governance is basically an organizational process; software tools which will assist within the administration of information management programs are an optional component.

The Relational model, specifically is that the most prevalent having the subsequent benefits:

* Simplicity: in a very management system, the relational data model is less complicated than the hierarchical and network models. o Data independence: A relational database's structure could also be modified without requiring any application changes.
* Structural independence: Relational databases care solely about data and not about structure. This might help the model perform better.
* Scalable: to enhance usability, the database must be scalable in terms of the number of records, or rows, and a few fields. o Easy to use: The relational model in DBMS is easy to use since row and column tables are intuitive. o Query capabilities: It aids high-level query languages like SQL in avoiding time-consuming database exploration.

This work is also accomplished using two direction systems:

SQL Server and file Management System. Here I could be a table comparing the two differents: (https://www.facebook.com/computergyan5/posts/database-part-1-docs/737814773418971/, n.d.)



We judged the Relational model and DBMS to be better suited for this project because of these advantages and the preceding comparison qualities

Than I picked this software.

### 2. About this project

This project was born to solve the problem of the Football Club . Currently, 1 database is needed to manage the system more easily and effectively including Footballers, Club names, Matches, CHart Stats Data and

football locations

## B. User Story

**My system will have 2 roles in this program:**

### \*\*Managers and Employees

* As a manager, I'd like to have Footballers, Club names, Matches, CHart Stats Data and football locations visible so I can report specifics back to the football team manager.
* As a manager, I want to manage the Team's calendar and specific work so that I can organize them effectively.
* As a manager, I like to add, remove and update data to suit the needs of the client and manage the team to check for errors

### \*\*As a football team

* As a football team, I want to see how many yellow card red card fouls have been committed
* As a football team, I want to check the location of my football venue to check if my information in the system is correct.

**The system must include CRUD (Create/Read/Update/Delete) actions and at least 5 advanced statistics actions.**

**Function:**

* Create: Create a new invoice when a customer places an order.
* Read: Read product information in the shop.
* Update: Update product information or order information.
* Delete: Delete the entered information.

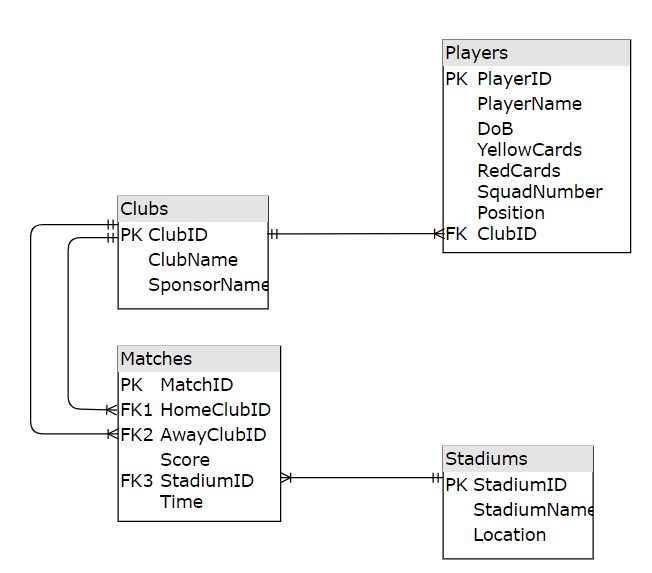
#### 5 Advanced Statistical Actions

* Statistics All players to add more products
* Statistics Number of cards to penalize non-compliant players.
* Team Statistics for easy control and maintenance of the match
* Statistics How many points scored in the match
* Statistics of locations, where the next match will take place

**II. Design the relational database system**

## A. Analyze the requirements

### 1. ERD Diagram



//Have 4 Boards//

Club: Club

-Club ID: Club ID number

-Club Name: Club name

-Sponsor Name: Sponsor's name

\*\* Matches: Matches

-Match ID: Match ID number

-HomeCLub ID: The ID number of the Home Club

-AwayClub ID: Guest Club ID number

Stadium: Stadium

-Stadium ID: The ID number of the stadium

-Stadium name: The name of the field

-Location: The location of the yard

Players: Player

-PlayerID: Player ID number

-PlayerName: Player name

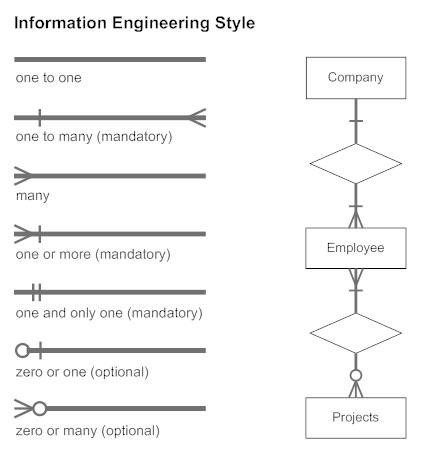
-Dob (Date of Birth): The player's date of birth

-YellowCards, redcards: Green, red penalty cards

-Club ID

### Relationship:

(https://www.smartdraw.com/entity-relationship-diagram/, n.d.)



One to Many Example:

1 club (have Multiple Players), 1 player 'can only join' 1 club

Club ID --> One to Many --> (HomeCLub)(AwayClub

Club ID /From Clubs/--> One to Many --> Club ID /From Players/

Stadium ID --> One to Many --> Stadium time

Diagram Database Objectives:

-1 multi-player club (Players)

-In 1 match (Matches) there will be 2 clubs (Home and Away)-HomeClub & AwayClub

//The club has a home club (HomeCLub), a guest club (AwayClub) //

-1 match (Matches) takes place on 1 stadium (Stadiums)

-1 leaderboard (Ranking) with many clubs (Clubs)

-Position is the position a player holds when playing, like a goalkeeper or a striker (Defender, Midfielder, Striker)

### Figure 1: Entity Relationship Diagram of Soccer Match Management

### 2. Table Description

- Players table: store information of the players. A player belongs to only one host club. Players table includes the following information:

+ PlayerID: **the primary key** of the Player table, used to ensure players are unique and not duplicated

+ PlayerName: the name of the player

+ DoB: player's date of birth

+ YellowCards: the number of yellow cards that the player is penalized for

+ RedCards: the number of red cards the player is penalized for

+ SquadNumber: the number is written on the player's shirt when wearing it on the field

+ Position: the position of the player held on the field

+ ClubID: the code of the club - where the player joins



### Figure 2: Players table



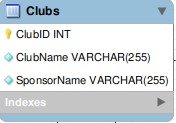
### Figure 3: Example of the Players table

- Clubs table: save information of clubs. A club with many players. The Clubs table includes the following information:

+ ClubID: the primary key of the Club table, used to ensure that the club is unique and not duplicated

+ ClubName: the name of the club

+ SponsorName: the name of the main sponsor for the club



### Figure 4: Clubs table



### Figure 5: Example of the Clubs table

- Matches table: save information of matches. A match will have two clubs (home and away). A match is played in a stadium. The Matches table includes the following information:

+ MatchID: the primary key of the Match table, used to ensure that the match is unique and not duplicated

+ HomeClubID: the code of the host club

+ AwayClubID: guest club code

+ Score: score of the match with the format Home team score - Away team score

+ StadiumID: code of the stadium

+ StartTime: the start time of the match



### Figure 6: Matches table



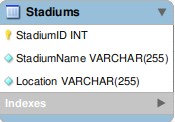
### Figure 7: Example of the Matches table

- Stadiums table: save information of stadiums. A stadium is a place where many football matches can take place. The Stadiums table includes the following information:

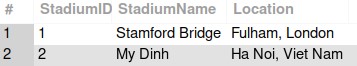
+ StadiumID: the primary key of the Match table, used to ensure the match is unique and not duplicated

+ StadiumName: the name of the stadium

+ Location: the location of the stadium



### Figure 8: Stadiums table



### Figure 9: Example of the Stadiums table

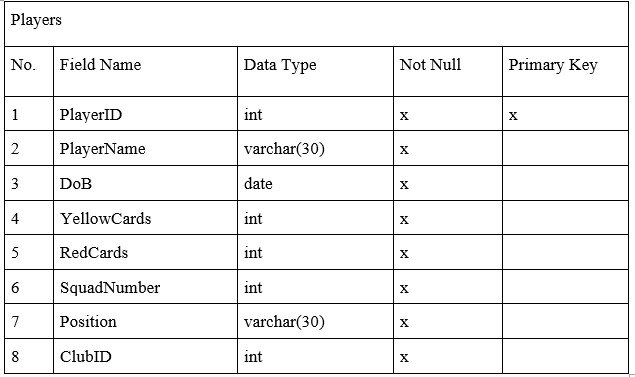
- Charts table: save information ranking of the charts. A leaderboard ranking has many clubs. The Charts table includes the following information:

+ Ranking: the primary key of the Chart table, used to ensure that the order is unique and not duplicated + ClubID: club code

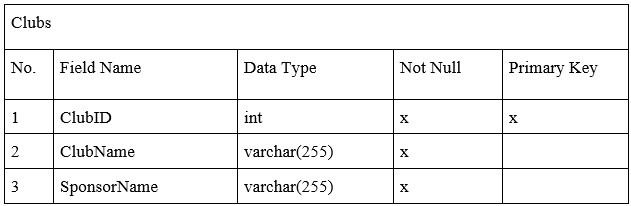
+ MatchesPlay: number of matches participated

+ GoalDiff: difference of total goals minus total goals conceded + Point: the score achieved by the club

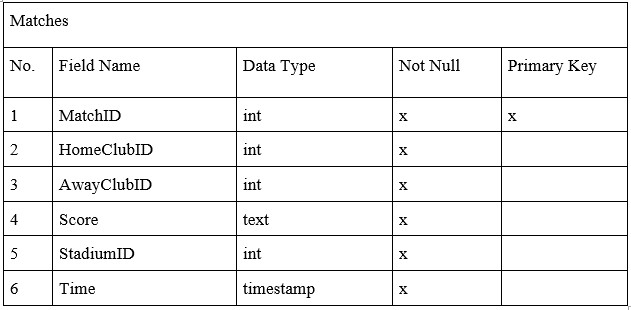
### 3. Database Analysis



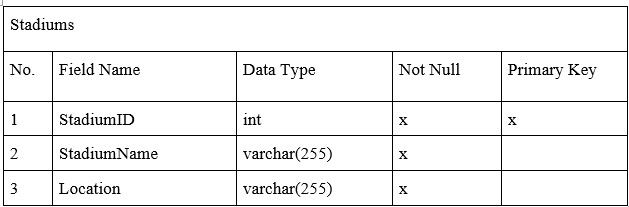
### Table 1: Data information of Players table



### Table 2: Data information of Clubs table



### Table 3: Data information of Matches table



**Table 4: Data information of Stadiums table**

## B. Database design with explanations

### 1. Database diagram

*Figure 12: Database Diagram*

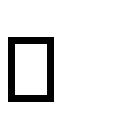
ERD diagrams and database diagrams both have tables, attributes, and relationships. However, ERD diagrams clearly show primary and foreign keys than database diagrams. From there, it helps readers easily visualize the system

=> The database diagram will retain all the important elements of the ERD diagram

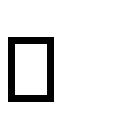
### D. Data validation

#### 1. Data type

In this article I have used the following this data types:

 INT: An integer is represented by the character INT. An integer can be written without a fractional component, for example, 1, 100, 4, -10, but not 1,2, 5/3, and so on. An integer can be 0 or a number that is both positive and negative. (SQL Server INT, 2021, sqlservertutorial)

|  |  |  |  |
| --- | --- | --- | --- |
| **Data type** | **Range** |  | **Storage** |
| **INT** | -231 (-2,147,483,648) to 231-  1 (2,147,483,647) | 4 Bytes |  |

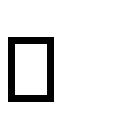
 **NVARCHAR:** The NVARCHAR data type in SQL Server is used to hold variable-length Unicode string data. The syntax of NVARCHAR is as follows:

NVARCHAR(n) is a kind of variable

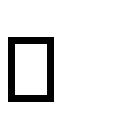
(sqlservertutorial, SQL Server NVARCHAR, 2021)

In this syntax, **n** defines a string length between 1 and 4,000. If you do not specify a string length, its default value is 1.

|  |  |
| --- | --- |
|  | **NVARCHAR** |
| Character Data Type | Variable-length, both Unicode and nonUnicode characters such as Japanese, Korean, and Chinese. |
| Maximum Length | Up to 4,000 characters |
| Character Size | Takes up 2 bytes per Unicode/Non-Unicode character |
| Storage Size | 2 times Actual Length (in bytes) |
| Usage | Due to storage only, used if you need Unicode support such as the Japanese Kanji or Korean Hangul characters. |

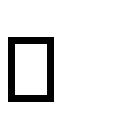
 **Text:** The text data type can be used to store non-Unicode data on the server's code page.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Data**  **Type** |  | **Lower**  **limit** | **Upper limit** | **Memory** |
| text | 0 chars |  | 2,147,483,647 chars | n bytes + 4 bytes |

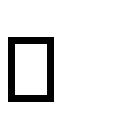
 **Float:** The approximate numeric data type is used to hold floating-point data. In scientific computations, they are often utilized.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Data Type** | **Lower limit** | **Upper limit**  1.79E+308 | **Memory** | **Precision** |
| float(n) | −1.79E+308 | Depends on the value of n | 7 Digit |

**Date:** The SQL Server DATE data type is used to hold date data in the database. The single component of the DATE data type is the date. A DATE value can be anywhere between January 1, 1 CE (000101-01) and December 31, 9999 CE (9999-12-31). A DATE value is stored in three bytes.



**2.**



(sqlservertutorial, SQL Server DATE, 2021)

The default literal string format of a DATE value is as follows: **YYYY-MM-DD** In this format:

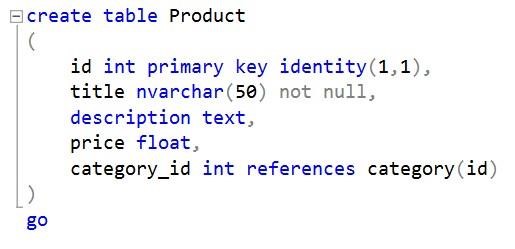
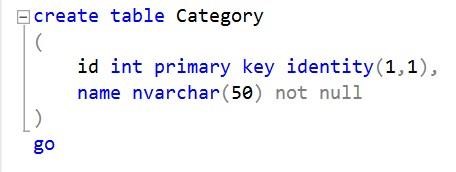
* **YYYY** is four digits that represent a year, which ranges from 0001 to 9999. o **MM** is two digits that represent a month of a year, which ranges from 01 to 12.
* **DD** is two digits that represent a day of the specified month, which ranges from 01 to 31, depending on the month. **Presence**

NOT NULL Constraint in SQL

The columns can have NULL values by default. A NOT NULL constraint in SQL is used to prohibit NULL values from being inserted into a given column, as it is considered an invalid value for that column. This implies that in the INSERT or UPDATE operations, you should give that column a valid SQL NOT NULL value because the column will always have data. (Yaseen, 2017)

* For example, we have the following simple create table statement that is used to define the Category table. This table contains only two columns, id, and name. In the name column definition statement, the SQL NOT NULL column-level constraint is enforced, treating the name column as a required column that must be given a valid SQL NOT NULL value. The other case for the id column can be omitted in the INSERT statement, with the possibility of giving it a NULL value. If the null possibility is not specified while defining the column, it will accept NULL value by default:
* For example, in the Product table, in the title column definition statement, the SQL NOT NULL column-level constraint is enforced, treating the title column as a required column that must be given a valid SQL NOT NULL value. The other case for other columns can be omitted in the INSERT statement, with the ability to give it a NULL value. If the null possibility is not specified while defining the column, it will accept NULL value by default:

17

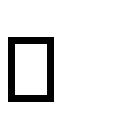


* For example, in the Staff table, in the fullname column definition statement, the SQL NOT NULL column-level constraint is enforced, treating the fullname column as a required column that must be given a valid SQL NOT NULL value. The other case for other columns can be omitted in the INSERT statement, with the ability to give it a NULL value. If the null possibility is not specified while defining the column, it will accept NULL value by default:

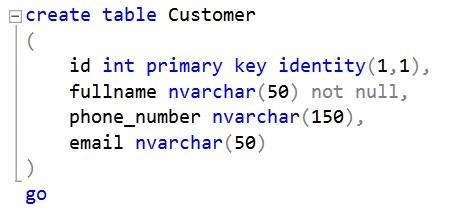
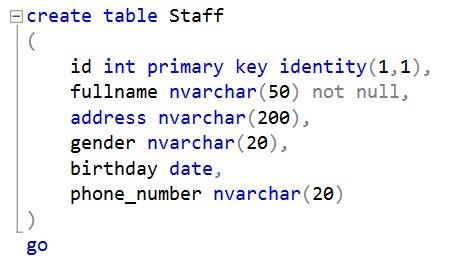
The PRIMARY KEY constraint uniquely identifies each record in a table. Primary keys must contain UNIQUE values, and cannot contain NULL values. A table can have only ONE primary key; and in the table, this primary key can consist of single or multiple columns (fields). (W3Schools, 2021) o The following SQL creates a PRIMARY KEY on the "id" column when the "Customer" table is created:

**3.**

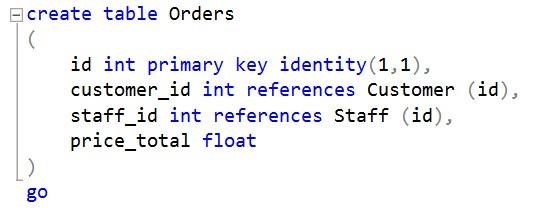
**Unique**



SQL PRIMARY KEY Constraint



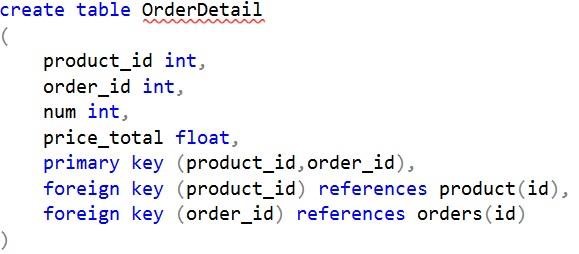
* The following SQL creates a PRIMARY KEY on the "id" column when the "Orders" table is created:



#### 4. Identity

In SQL Server, we create an identity column to auto-generate incremental values. It generates values based on predefined seed (Initial value) and step (increment) values. (Gupta, 2019)

For example, in id int Identity (1,1) in OrderDetail table, the first 1 means the starting value of the ID and the second 1 means the increment value of the ID. It will increment like 1,2,3,4... If it's (5,2), then it starts from 5 and increments to 2 like 5,7,9,11, ...



#### 5. Data integrity

**CASCADE OPERATIONS:** Cascading allows when a change is made to a certain entity, this change to apply to all related entities.

In this project, I have used Cascade Delete. A foreign key with Cascade Delete means that if a record in the parent table is deleted, the corresponding record in the child table will also be deleted automatically.

In this foreign key example, we created the parent table as the Category table. The products table has a primary key that includes a category\_id field.

Next, we created a second table called the Product table which will be the child table in this foreign key with cascading delete example. We used the CREATE TABLE statement to create a foreign key on the products table. The foreign key establishes the relationship between the category\_id column in the Category table and the category\_id column in the Product table.

For this foreign key, we have specified an ON DELETE CASCADE clause that tells SQL Server to delete the corresponding records in the child table when the data in the parent table is deleted. So in this example, if the category\_id value is removed from the Category table, the corresponding records in the products table using this category\_id will also be deleted.

Besides that, I also use Cascade Updates because:

-

**Classification of Data Integrity**

•

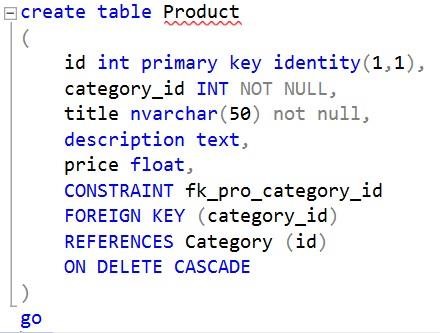
System/Pre Defined Integrity

•

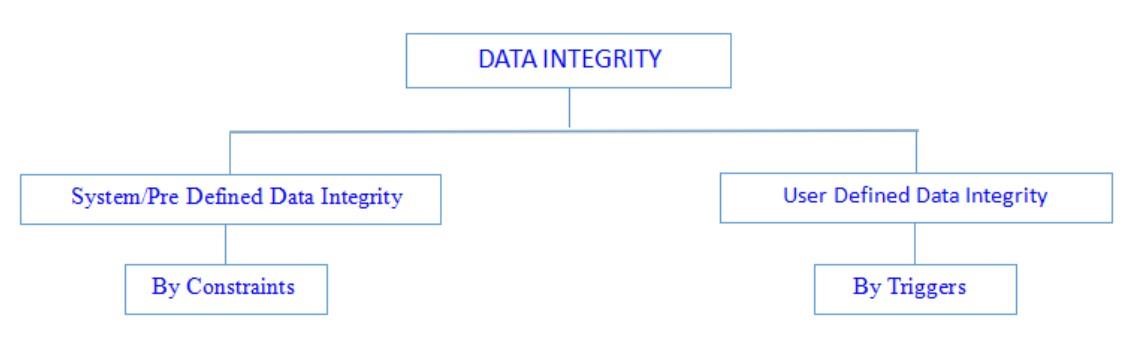
User

-

Defined Integrity

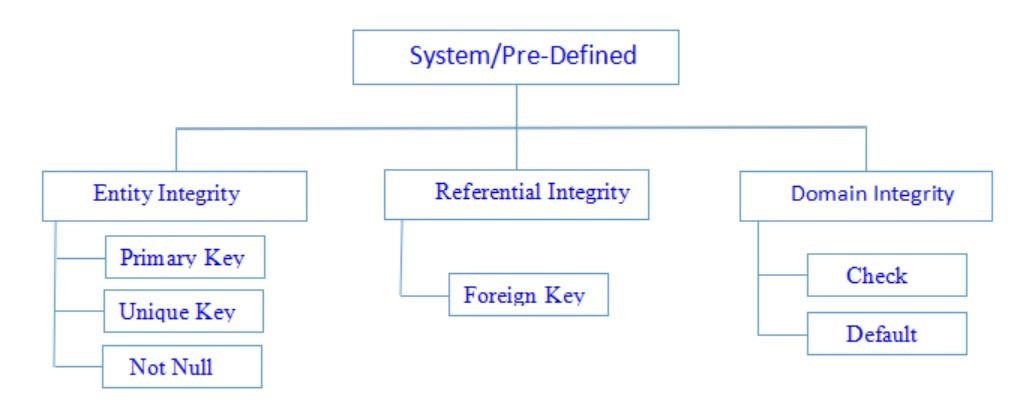


The primary key is not identity (not auto-incremented) and therefore it can be changed - Each table has a unique constraint.



##### System/Pre Defined Integrity

We can implement this using constraint. This is divided into three categories.



## E. Review whether the database is normalized

**purpose of database normalization**

The database should be normalized to reduce redundancy of (data duplication) and guarantee that only relevant data is maintained in each table.

It also eliminates any issues that may arise as a result of database changes such as “inserts, deletes, and updates”. Normal form refers to the phases of organization.

### Target

-Less waste memory

-Faster access and accuracy

- No anomalies occurred during data update

**How normalized in database works:**

In 1972, E.F Codd give out 3 normal standard form

Raymond F. Boyce and Edgar F. Codd created BCNF in 1974 to address some sorts of anomalies not covered by 3NF as initially established.



Relational normalization in databases consists of 5 main stages: (1) organization, (2) identity tables,

(3) primary keys, (4) foreign keys, (5) data types and form identifiers

First, it's easy to see that all the created tables: Players, Clubs, Stadiums, Matches and Charts all satisfy the 1NF (First Standard Form) rule. Because of:

* Each cell in the table contains only 1 value
* Each record is unique

Next, the above 5 tables also satisfy the standard 2NF (Second Normal Form) rule. Because of:

* The above relationship has met the 1NF . standard
* Each primary key has only 1 attribute and no candidate key. Thus, every primary key identifies all non-key attributes in each table

|  |  |
| --- | --- |
| **2NF** | |
|  |  |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

### Table 6: The second normal form (2NF)

Finally, considering the 3NF standard (Third Normal Form), the above tables also satisfy the rules of the 3NF standard. Because of:

* The above relationship has met 2NF . standard
* Don't have any transitive dependency properties on every key

## F. Code

### 1. Create

CREATE DATABASE soccer\_match;

USE soccer\_match;

CREATE TABLE Clubs(

ClubID int PRIMARY KEY,

ClubName varchar(255) NOT NULL,

SponsorName varchar(255) NOT NULL

);

CREATE TABLE Players(

PlayerID int PRIMARY KEY,

PlayerName varchar(30) NOT NULL,

DoB date NOT NULL,

YellowCards int NOT NULL,

RedCards int NOT NULL,

SquadNumber int NOT NULL,

Position varchar(30) NOT NULL,

ClubID int NOT NULL,

FOREIGN KEY (ClubID) REFERENCES Clubs(ClubID)

);

CREATE TABLE Stadiums(

StadiumID int PRIMARY KEY,

StadiumName varchar(255) NOT NULL,

Location varchar(255) NOT NULL

);

CREATE TABLE Matches(

MatchID int PRIMARY KEY,

HomeClubID int NOT NULL,

AwayClubID int NOT NULL,

Score text NOT NULL,

StadiumID int NOT NULL,

StartTime timestamp NOT NULL,

FOREIGN KEY (HomeClubID) REFERENCES Clubs(ClubID),

FOREIGN KEY (HomeClubID) REFERENCES Clubs(ClubID),

FOREIGN KEY (StadiumID) REFERENCES Stadiums(StadiumID)

);

### 2. Insert values

INSERT INTO Clubs

VALUES (1, 'Chelsea', 'Yokohama'),

(2, 'MU', 'Chevrolet'),

(3, 'Real Madrid', 'Fly Emirates'),

(4, 'Viet Nam', 'So1HungVuong'),

(5, 'Barcelona', 'Rakuten');

INSERT INTO Players

VALUES (1, 'Eden Hazard', '1991-01-07', 0, 0, 10, 'Tien Dao', 1),

(2, 'Maxwell Nguyen', '1998-05-30', 2, 1, 11, 'Thu Mon', 1),

(3, 'Mewcos Hiep', '1998-01-06', 0, 1, 12, 'Tien Ve', 1),

(4, 'Gary Cahill', '1985-12-19', 1, 1, 13, 'Hau Ve', 1),

(5, 'Adnan Januzaj', '1995-02-05', 0, 0, 09, 'Tien Ve', 1),

(6, 'Luke Xuan Phu', '1998-08-18', 0, 0, 06, 'Tien Ve', 2),

(7, 'David De', '1990-01-10', 0, 0, 09, 'Thu Mon', 2),

(8, 'Cristiano', '1998-11-23', 2, 0, 07, 'Tien Dao', 2),

(9, 'Ronado', '1996-02-07', 2, 1, 10, 'Hau Ve', 2),

(10, 'Doan Van Hau', '1992-01-01', 1, 0, 01, 'Hau Ve', 4),

(11, 'Phan Van Duc', '1991-01-27', 0, 0, 02, 'Hau Ve', 4),

(12, 'Nguyen Quang Hai', '1995-10-07', 1, 1, 05, 'Thu Mon', 4),

(13, 'Nguyen Van Toan', '1999-12-01', 1, 1, 10, 'Tien Dao', 4),

(14, 'Toni', '1993-11-07', 0, 0, 08, 'Tien Dao', 5);

INSERT INTO Stadiums

VALUES (1, 'Stamford Bridge', 'Fulham, London'),

(2, 'My Dinh', 'Ha Noi, Viet Nam');

INSERT INTO Matches

VALUES (1, 1, 2, '2-0', 1, '2021-01-01 10:00:00'),

(2, 1, 3, '2-5', 2, '2021-02-01 01:00:00'),

(3, 2, 1, '0-3', 2, '2021-03-01 13:45:00'),

(4, 3, 4, '1-0', 1, '2021-11-01 15:30:00'),

(5, 3, 5, '2-2', 2, '2021-12-01 02:15:00');

# SQL Server

**What is SQL Server?**

●Software developed by Microsoft based on RDBMS. Also an ORDBMS (Object Relational Database Management System). An independent platform. The software uses both command line interface and GUI interface. SQL language support (formerly SEQUEL - structured English query language) - which is IBM's product.

## Purpose of using SQL Server

●Create database.

●Maintain the database.

●Data analysis by SSAS - SQL Server Analysis Services.

●Create report by SSRS - SQL Server Reporting Services.

●Perform ETL (Extract-Transform-Load) process with SSIS - SQL Server Integration Services.

## Components of SQL Server

●SQL Server operates on a client-server model, so there are two components: Workstation and Server:

●Workstation is installed on any device or on the device of the server operator. These are software interfaces to interact with the Server, such as SSMS, SSCM, Profiler, BIDS or SQLEM ...

●Server is installed on centralized server. These are services like SQL Server, SQL Server Agent, SSIS, SSAS, SSRS, SQL Browser, SQL Full Text Search ...

### SQL Server versions

|  |  |  |
| --- | --- | --- |
| **Versions** | **Release year**  1995 1996  1998 | **Code Name**  SQL95  Hydra  Sphinx |
| 6.0 |
| 6.5 |
| 7.0 |
| 8.0 (2000) | 2000 2005  2008 2010  2012 | Shiloh  Yukon  Katmai  Kilimanjaro  Denali |
| 9.0 (2005) |
| 10.0 (2008) |
| 10.5 (2008 R2) |
| 11.0 (2012) |
| 12 (2014) | 2014 | Hekaton (Original Name), SQL 14 (Current Name) |

### Instance of SQL Server

●Is a SQL Server installation.

●An exact copy of the software.

●Installing "n" times will create a "n" installation.

●There are two types of SQL Server installations: Default - Named - Named.

●Only one default installation is supported on one Server.

●Many new installations are supported on a Server.

●The default installation will take the server name as an Instance name.

●The name of the default installation is MSSQLSERVER.

●SQL Server 2000 version supports 16 installations.

●The SQL Server version 2005 and later supports 50 installations.

### Benefits of installations

●Install many different MS SQL versions on the same machine.

●Cut down the costs.

●Maintain separate production, development and testing environments. ●Minimize temporary problems on the database.

●Separate security privileges.

●Maintain backup server.

# SSMS

●SQL Server Management Studio is a tool in SQL Server if you choose when installing. This tool helps to connect and manage SQL Server on a graphical interface instead of using the command line.

●To connect to a remote SQL Server installation, you will also need this software or another similar software. Management Studio is often used by administrators, developers, testers ....

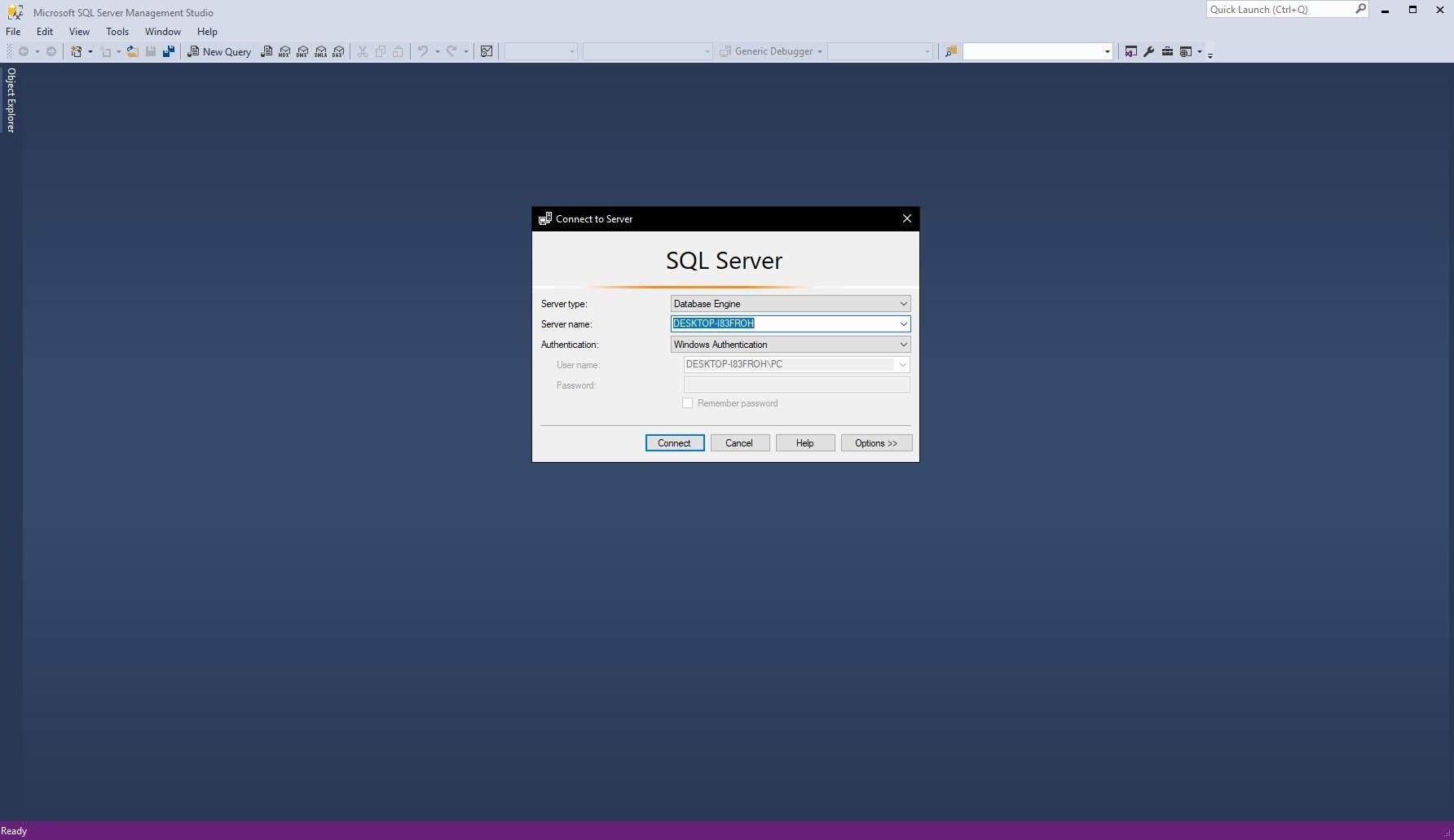
●These are common ways to open SQL Server Management Studio.

●Method 1:

Open Start> All Programs> MS SQL Server 2012> SQL Server Management Studio.

●Method 2:

Open Run and type SQLWB (with 2005 version) or SSMS (with version 2008 and later), then click Enter. SQL Server Management Studio will open as shown in the image below.

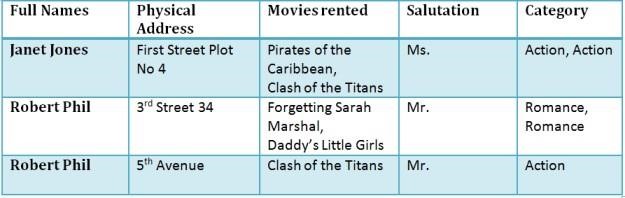


## MS SQL Server Management Studio interface when opened

**M1. Produce a comprehensive design for a fully functional system which includes interface and output designs, data validations and data normalisation.**

●The sequence of steps to design Database, from 1NF to 3NF:

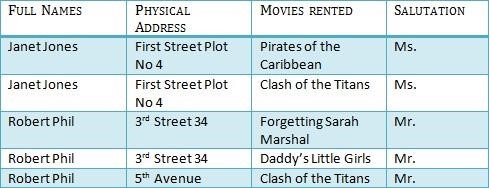
## Database without Normalization



## 1NF (First Normal Form) Rules

* Each table cell should contain a single value
* Each record needs to be unique

**1NF Example**



**What is a Composite Key?**

A composite key is a primary key composed of multiple columns used to identify a record uniquely

In our database, we have two people with the same name Robert Phil, but they live in different places.

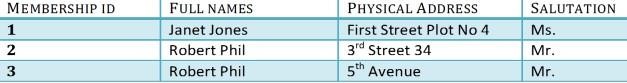
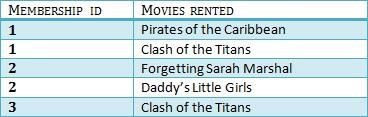


We require both Full Name and Address to identify a record uniquely. That is a composite key.

## 2NF (Second Normal Form) Rules

* Rule 1 - Be in 1NF
* Rule 2 - Single Column Primary Key

It is clear that we can't move forward to make our simple database in 2nd Normalization form unless we partition the table above.

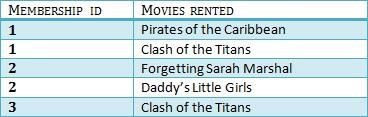


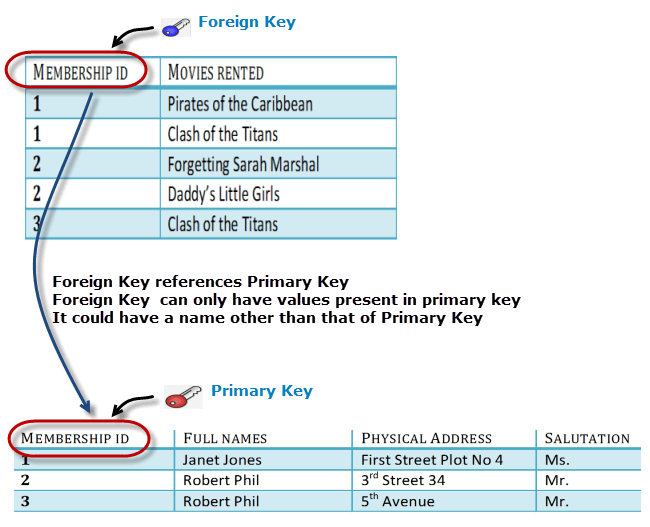
We have divided our 1NF table into two tables viz. Table 1 and Table2. Table 1 contains member information. Table 2 contains information on movies rented.

We have introduced a new column called Membership\_id which is the primary key for table 1. Records can be uniquely identified in Table 1 using membership id

**Database - Forgein Key**

In Table 2, Membership\_ID is the Foreign Key

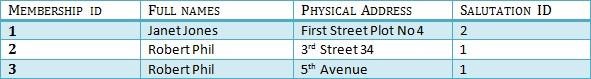


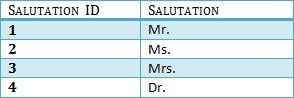
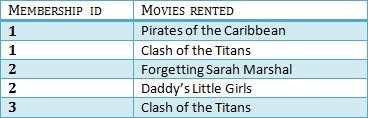


## 3NF (Third Normal Form) Rules

* Rule 1- Be in 2NF
* Rule 2- Has no transitive functional dependencies

**3NF Example**





Create table Salutation CREATE TABLE Salutation(

SalutationID int NOT NULL PRIMARY KEY,

Salutation varchar(255) NOT NULL

);

Create Table Membership CREATE TABLE Membership (

MembershipID int IDENTITY(1, 1) PRIMARY KEY,

FullName varchar(255),

PhysicalAddress varchar(255),

SalutationID int ,

CONSTRAINT fk\_salutation\_membership

FOREIGN KEY (SalutationID)

REFERENCES Salutation (SalutationID) );

Create Table Movie Rented

CREATE TABLE Movie(

MembershipID int,

MoviesRented varchar(255),

CONSTRAINT PK\_Movie

PRIMARY KEY (MembershipID, MoviesRented),

CONSTRAINT FK\_movie

FOREIGN KEY (MembershipID)

REFERENCES Membership (MembershipID) );

**D1. Assess the effectiveness of the design in relation to user and system requirements.**

●A database is a collection of organized data, often stored and accessed electronically from a computer system. When databases are more complex, they are often developed using formal design and modeling techniques.

●Database management system (DBMS) is software that interacts with end users, applications and databases to collect and analyze data. DBMS software includes core utilities provided for database administration. A total of databases, DBMS and related applications can be called "database systems". Usually the term "database" is also used to refer to any DBMS, database system or application associated with the database. ●Computer scientists can classify database management systems according to the database models they support. The relational database became dominant in the 1980s. These model data are in the form of rows and columns in a series of tables and mostly use SQL to write and query data. In the 2000s, non-relational databases became popular, called NoSQL because they used other query languages.



(HẢI)

**Source: https://hainh2k3.com/tim**[**-hieu-ve-with-nolock-trong-ms-sql-server/**](https://hainh2k3.com/tim-hieu-ve-with-nolock-trong-ms-sql-server/)

●Officially, a "database" refers to a set of related data and organization. Access to this data is often provided by a "database management system" (DBMS) that includes an integrated computer software suite that allows users to interact with one or more databases and grant access to all data contained in the database (although restrictions may exist limited access to specific data). DBMS provides various functions that allow the import, storage and retrieval of large amounts of information and provide ways to manage how that information is organized.

Due to their close relationship, the term "database" is often used casually to refer to both the database and the DBMS used to process and query it.

Outside the world of professional information technology, the term database is often used to refer to any set of related data (such as spreadsheets or index cards) because of the size and normal usage requirements. must use database management system.

The current DBMS provides different functions that allow database and data management to be categorized into four main functional groups:

Data definition - Create, modify and remove defined data organization definitions.

Update - Insert, modify and delete actual data.

Retrieval - Provides information in a form that can be used directly or for further processing by other applications. Retrieved data can be provided in the same basic form as stored in a database or in a new form obtained by changing or combining existing data from the database.

Administration - Register and monitor users, enforce data security, monitor performance, maintain data integrity, handle concurrency control and recover information that has been corrupted by some events as an unexpected system error.

Both the database and its DBMS comply with the principles of a specific database model."Database system" is collectively referred to as database model, database management system and database.

Physically, the database server is a specialized computer that contains a real database and only runs DBMS and related software. Database servers are usually multi-processor computers, with generous memory and RAID disk arrays used for stable storage. RAID is used to recover data if any disk fails. Hardware database accelerator, which is connected to one or more servers via a high-speed channel, is also used in high-volume transaction processing environments. DBMS is found at the heart of most database applications. DBMSs can be built around a custom multitasking kernel that supports integrated networks, but modern DBMSs often rely on a standard operating system to provide these functions.

Because DBMSs are an important market, computer and storage providers often take into account DBMS requirements in their own development plans.

Databases and DBMSs can be classified according to the database model they support (such as relational or XML), the type of computer they run on (from the server cluster to mobile phones), Query language (s) is used to access databases (such as SQL or XQuery) and their internal techniques, affecting performance, scalability, resilience, and security.

## Graphical Query Designer User Interface

This graphical query designer supports three types of query commands: Text, StoredProcedure, or TableDirect. Before you create a query for your dataset, you must select a command type option on the Query page of the Dataset Properties dialog box.

The following options are available for query type:

Text Supports standard Transact-SQL query text for relational database data sources, including data processing extensions for Microsoft SQL Server and Oracle.

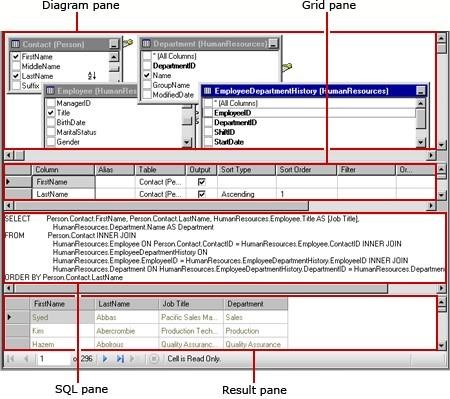
TableDirect Selects all the columns from the specified table. For example, for a table named Customers, this is the equivalent of the Transact-SQL statement SELECT \* FROM Customers.

StoredProcedure Supports calls to stored procedures on the data source. To use this option, you must have been granted Execute permissions on the stored procedure by the database administrator on the data source.

The default command type is Text.

### Command Type Text

In **Text** type, the graphical query designer presents four areas, or panes. You can specify columns, aliases, sort values, and filter values for a Transact-SQL query. You can view the query text generated from your selections, run the query, and view the result set. The following figure shows the four panes.



Source: [https://docs.microsoft.com/en-us/sql/reporting-services/report-data/graphical-query-designer-user-interface?view=sqlserver2017](https://docs.microsoft.com/en-us/sql/reporting-services/report-data/graphical-query-designer-user-interface?view=sql-server-2017)

The following table describes the function of each pane.

|  |  |
| --- | --- |
| **Pane** | **Function** |
| Diagram | Displays graphic representations of the tables in the query. Use this pane to select fields and define relationships between tables. |
| **Pane** | **Function** |
| Grid | Displays a list of fields returned by the query. Use this pane to define aliases, sort order, filters, groups, and parameters. |
| SQL | Displays the Transact-SQL query represented by the diagram and Grid panes. Use this pane to write or update a query using Transact-SQL. |
| Result | Displays the results of the query. To run the query, right-click in any pane, and then click **Run**, or click the **Run** button on the toolbar. |

● Evaluate the responsiveness of Database to user requirements

-Useful for people working in Database, easy to use and fast interface.

●About DBMS(DataBase Management System)

- Advantages of DBMS:

* Ensure consistency of data.
* Managing redundant data.
* Ensure data integrity.
* More data sharing.

- Disadvantages of DBMS:

* Takes up a lot of memory.
* Quite complicated.
* Common DBMS is usually slow
* Depending on the environment and functions, there are different prices