Database Manipulation with Prolog

Final Year Dissertation by Fatah Adan

12010183

Computer Science

Vladimir Rybakov

https://github.com/FatahAdan/Project.git

Declaration

No part of this project has been submitted in support of an application for any other degree or qualification at this or any other institute of learning. Apart from those parts of the project containing citations to the work of others, this project is my own unaided work.

Signed _____

Table of Contents

Chapter 1 - Introduction/Terms of reference

1.1 Learning Outcomes

To develop an understanding of the nature of databases and be able to develop, maintain and interrogate databases.

To develop knowledge of computer hardware and an understanding of how the selection of Hardware will affect the performance of an application.

To investigate the interaction between hardware and software and the influence of this

Interaction on the design of computer systems.

To study the fundamentals of computer network communications and communication protocols;
In addition, to study the management and security of networked systems.

1.2 Background

This project will consist of creating a database with the Prolog programming language. Prolog is a free software will allows the implementation of the programming language Prolog. It is commonly used for teaching and semantic web applications. It has a prosperous set of features, such as libraries for constraint logic programming. The other software needed to complete this project is Notepad++; I will be using this software to write my Prolog scripts. Notepad++ is a free source code editor that supports several languages. It is a simple code editing software, which is very user friendly and does everything needed. An example area in which a database can be used is a company, which store it employee and department information. It would consist of tables, which contain employee details (Name, Surname, Address etc…), and the different departments the company has such as (Department_ID, Department_Name). These are examples of the fields a database can have. There will be a design stage, which will be the process of deciding how many tables the database will have and what they will contain. The audience of this database can for example be the manager and the supervisors of the company, which want to find information on a particular employee. They can consult the database and find out everything they need to know about the employee. A database is a very useful way of storing data for businesses, because they keep track of everything such as employees, and transactions. This is a very beneficial factor because the database can be kept and archived. The data will always be available whenever it is needed.

1.3 Aim

The aim of this project is to provide users a way to store data in controlled area. Also the completion of the product, it must illustrate obtained Prolog scripts by application to a specific database. The database can be about anything I would like to do it on. For example a Hospital DB. Instruments to handle typical queries such as the ones used in Oracle database (select, insert) must be developed for a Prolog Database. I will be doing a database based on teachers and what subjects they teach. First things first I will have to learn the basics of the Prolog programming language and get comfortable with it. To do this

I will be going through online tutorials which I will implement in some practice Prolog files to familiarise myself with the way Prolog works. Then once I am comfortable with Prolog, I will be designing the database itself. This is the design stage of the database and processes such as creating the different tables and what they are going to store will take place. Once the designs are done, the database will be implemented with Prolog application. The final stage is when the database is created, and then there will be executions of Prolog queries, which will test the data the database contains. There will be queries, which will search for data in the database, such the 'Select' function in oracle SQL.

The Prolog commands to manipulate a database are very similar to SQL commands but they are named differently.

Prolog has a command called ‘assert’, which is used to input data into the database, this is similar to the 'Insert' command that SQL databases use. Prolog has two other method in which the user can input data, the first command is 'assertz', which places new at the end of the database, and 'asserta', which places new data at the beginning of the database. As well as inputting data into the database, Prolog has a way to remove data from it also. This command is called 'retract', when using this command the first occurrence of the argument will cause it to be removed.

1.4 Objectives

1. Learn basics of Prolog
2. Design the Database
3. Implement Prolog code to create the database
4. Create Prolog queries
5. Test the Database

1.5 Required Resources

Notepad++ - I will use this to write my Prolog scripts, the Prolog scripts I code will have an extension such 'name.pl' that will be recognised by the SWI Prolog software.

SWI Prolog - SWI Prolog runs on operating systems such as UNIX, Windows, and Macintosh and Linux platforms. I will be running it on a Windows 7 operating system; I will be using it to run the scripts I have coded.

Google Chrome - This a freeware web browser developed by Google. I will be using this for research and guidance throughout the course of this project.

GitHub - This is a web-based Git repository hosting services

Chapter 2 - Literature Review

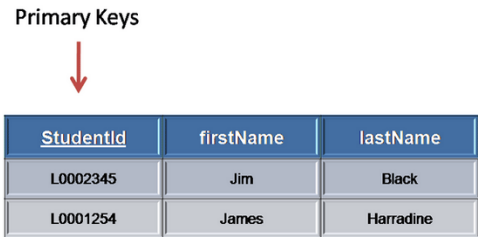
[James, Berrington (2014)], The use of a database provides methods in which enables you to collect and organise data in a suitable format. A database can be a simple list typed on paper or computer spreadsheet. This type of databases are

known as a flat file, which usually consists of one table with rows and columns of relevant data. An example of this would be a simple database created for a small business. The table would contain rows such as employee name and contact number. If the manager needs to contact an employee, the table can be scanned until the chosen employee name is found, then look at the associated contact number for that employee. The advantages of having a flat file are, all records are stored in one place, it's easy to understand and simple sorting and filtering can be carried out. A flat file also has limitations coming with it, example of this is potential data duplications, non-unique records and it is difficult to update.

2.1 Relational Databases

However, in this time of technology not many companies use flat file databases. Most databases now are computerised; they are referred to as relational databases. To create and maintain a relational database you must obtain the experience using structured query language generally known as SQL. This sort of database uses multiple tables of data that are related and links them together using keys which is a common identifying code. An example of this is a employee code might serve as a key that links employee information and department tables together. This type of database prevents redundant data being entered or multiple entries of the same data and it provides fast sorting and numerous reporting opportunities.

Relational databases [James, Berrington (2014)], use primary and foreign keys in tables to link multiple tables together. Firstly, a primary key is a field most appropriate to be the main reference key for the table. As its name suggests, it is the primary key of reference for the table and is used all the way through the database to help establish relationships with other tables. The primary key must contain unique values, must never be null and uniquely identify each record in the table.

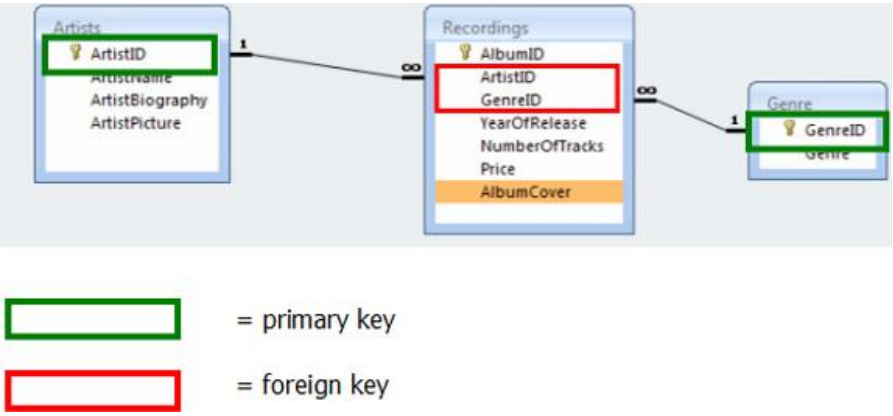


The design process of a database consists of using the ERD method. This is short for Entity Relationship Diagram. With ER-diagram to represent entities and their contacts is often used in database model design. What this diagram represents is the relationships between tables in a database, this essential to obtain a good database.

There are three basic elements in a ER-diagram, which the first element is Entities which are things for which we want to store information, such a person or place. The second element is Attributes which are data that's collected for an entity, for example a person name. The final element of this diagram is Relationships which describe the relations between entities. This will give you a overview of the database and everything it stores and what the relationships between them are.

Entities in database often have a relation to another entity in a different table. A relationship type defines an association among entity types. A relationship between entities has a degree that is the number of participating entity types. An example of this is a binary relationships and a ternary relationships. The relationship types can also have attributes, for example the StartDate attribute on a supervises relationship.

A foreign key is used to link tables together and create a relationship. It is a field in one table that is linked to the primary key in another table.



[Taylor, Allen G (1998)] A database management system is used to maintain a fully functioning relational database. A management system is important because it manages data efficiently and allows users to perform many tasks with no difficulty. A database management system stores, organises and manages a large quantity of information contained by a single software application. Use of this system increases efficiency of business operations and reduces overall costs. Without the use database management systems, tasks have to be done manually and take more time. The data can be categorised and structured to suit the needs of the company. The data is inputted into the system and accessed on a schedule basis by assigned users. Each user may have an assigned password to gain access to their part of the system which is relevant to the task they wish to complete. Multiple users can use the system at the same time in various ways. A good management system should reduce data redundancy, reduce updating errors to increase

consistency and greater data integrity for independence from application programs. The data security must be improved to protect the data. Although a management system has few benefits it brings, it also has disadvantages. The management systems are complex and time consuming to design. It will cost companies money for the hardware's and software's for moving from a flat file to a computerised database.

Databases can sometimes contain redundant data. The meaning of data redundancy is a condition created within a database. This is when the same piece of data is held in two different places, within the same database. When data is repeated, this constitutes data redundancy. This can happen by accident or can also be done deliberately for back up. The downside of having redundant data in your database is that it occupies more than required space. This will slow down the processing time, if there is a lot of redundant data. It is also difficult to update redundant data as there is more than one saved. This causes the data accuracy/consistency to decrease.

In the making of relational databases, a design process, which includes various models, must take place to decide what the database is going to include in its data and tables. In the next section, I will be discussing each one of these model, and describing how they work, and what benefits they bring to the database system.

The first model is the use case diagram, which overview the usage requirements for any given system or database. They are useful when you want to give a presentation on the overall requirements to a someone such as stakeholders. There are different things that go into a use case diagram. The first is the use cases themselves, they describe a sequence of actions that provide something of measurable value to an actor, and the shape used in the diagrams is horizontal ellipse. The second object in a use case diagram is Actors, which is a person or organisation that plays a role in one or more interactions with the operations in the system. The object used to represent an actor in the diagram is a stick figure. The next object that a use case diagram includes is Associations; these are associates between actors and use case. Their association is indicated using a solid line; an association exists whenever an actor is involved with an interaction described by a use case. Another feature the use case diagram includes is a System boundary, System boundary boxes are rarely used, but what they do is indicate the scope of the system. Anything within the box represents functionality that is in scope and anything placed outside the box is not functional.

The next model I will be discussing is the use case specifications. The first step of creating the model is to identify the use cases the database will include the use of use case specifications. These outline the pre-conditions, the flow of events that happen in the system. The post conditions and any alternative routes to the original flow of events. By doing this we can begin to plan the behaviour and the processes behind each actions that occurs in the system Taylor, Allen G (1998).

2.2 Structured Query Language – (SQL)

[Taylor, Allen G (1998)], SQL stands for Structured Query Language; it provides a way for users to communicate with databases. SQL is known to be the standard language for relational database management systems. The user must create SQL statements to perform tasks such as update or retrieve data from the database. Some of the more common database management systems, which use SQL, are Oracle and Microsoft SQL server. Although most database management system use the SQL language, some have their own unique extensions that can only be used on their specific system. An example of a SQL statement is the SELECT function, which is used to select data from a database.

SQL SELECT Syntax

```
SELECT column_name,column_name
FROM table_name;
```

and

```
SELECT * FROM table_name;
```

2.3 Prolog

[Kuo (1987)], Prolog is a free software will allows the implementation of the programming language Prolog. It is commonly used for teaching and semantic web applications; it has a prosperous set of features, such as libraries for constraint logic programming. Prolog has been created from research at the University of Aix-Marseille back in the late 60's and early 70's but most sources refer 1972 to be the birthdate of Prolog. Since 1972, Prolog has branched out into many different dialects. When a user is programming in Prolog and a code is loaded, it I referred to as consulting. Prolog has a prompt, which allows the user to

enter queries, unless the query is true and the result has been printed, if Prolog has no solution it will write 'no'.

2.3.1 SWI-Prolog

This a free implementation of the Prolog programming language, which is commonly used for teaching and semantic web applications. This program can be installed on different operating systems such as UNIX and Windows. The installation process is different for the two systems, which in UNIX, the SWI-Prolog is installed on to the system by default as 'swipl'. On UNIX the command line arguments of SWI-Prolog utilises programs that are documented using standard Unix man Pages. On the other hand, on Windows, when SWI-Prolog is installed on to the system, important new features are available to the user. The first feature is that a folder called 'swipl' is also installed, which contains all the libraries and executables, and no files are stored outside this directory. Another feature is a program called 'swipl-win.exe', which provides a window, which the user can interact with, to execute the coded programs [Kuo (1987)].

2.3.2 Facts, Rules and Queries

The Prolog programming language expressions are comprised of the following truth-functional symbols. The figure below shows the English to Prolog interpretation of the expressions used in Prolog codes.

English	PROLOG
And	,
Or	;
If	:-
Not	not

The next thing I will be discussing is the way Prolog uses variables and names in its syntax. In Prolog, variables begin with an uppercase letter. Therefore, predicate names, function names and object names must all begin with a lowercase letter, so the program does not confuse them with variables. An example of predicate names, function names and objects are as follows.

- mother_of
- male
- female
- greater_than

2.3.2.1 Facts

Now I am going to talking about how facts work within a Prolog program. A fact is a predicate which is an expression that makes a declarative statement about the problem domain. When a variable occurs in a Prolog program, it assumes that it is universally quantified. This is why all Prolog lines of code must end with a period. Below are examples of Prolog facts with the correct syntax.

- likes(ashraf, hope). Which means Ashraf likes Hope.
- likes(X, hope). Which means everyone likes Hope.
- likes(ashraf, Y). Which means Ashraf likes everyone.
- likes(ashraf, Y), likes(Y, ashraf). Which means Ashraf likes everyone and everyone likes Ashraf.
- likes(ashraf, hope); likes(ashraf, mary). Which means Ashraf like hope or Ashraf like Mary.
- Not(likes(ashraf, apples)). Which means Ashraf does not like apples.
- likes(ashraf, hope) :- likes(ashraf, mary). Which means Ashraf like Hope if Ashraf likes Mary.

2.3.2.2 Rules

Now, I am going to be discussing how rules work in a Prolog program and how to use them to complete specific tasks. A rule in Prolog is a predicate expression that uses logical implications to describe a relationship or link among facts. A Prolog rule has a syntax format of 'left_hand_side :- right_hand_side.', this is interpreted as left hand side IF right hand side. The left hand side is restricted to a single, positive and literal, which means it must contain of a positive atomic expression. Therefore it cannot be changed and it cannot contain logical connectives [Jan Wielemaker].

Examples of valid rules are as follows:

- friends(X,Y) :- likes(X, Y), likes(Y, X). This interprets, as X and Y are friends if they like each other.
- Hates(X, Y) :- not(likes(X, Y)). This interprets, as X hates Y if X does not like Y.

Examples of invalid rules are as follows:

- left_of(X, Y) :- right_of(Y, X) – this rules is missing a period at the end of it making it invalid
- likes(X, Y), likes(Y, X) :- friends(X, Y). this is invalid because the left hand side is not a single literal.

2.3.2.3 Queries

Next, I am going to discussing how Queries are used within a Prolog program and how use them find what you are looking for in the database. SWI-Prolog responds to queries using the facts and rules saved in the database, and output the desired information to the user.

The database is expected to represent what is true about a particular problem domain. When you execute a Prolog query, you are essentially asking if it can prove, what you are asking is true. If the query is true, the compiler will display “Yes” and the variable bindings that it made while coming up with its answer. If the query is not true it will simply display “No”. The following example will be to demonstrate how queries can be used in a Prolog program to retrieve data from a given database [Jan Wielemaker].

Below are some simple facts a database could have:

- likes(dave, food). Which means dave likes food.
- likes(dave, wine). Which means dave likes wine.
- likes(hope, wine). Which means hope likes wine.
- likes(hope, john). Which means hope likes john.

From the facts above, the following queries can be used to retrieve specific answers

- |?- likes(dave, food). Which the answer for this query is true so the program displays “yes.”
- |?- likes(dave, wine). Which the answer for this query is true so the program displays “yes.”
- |?- likes(hope, food). Which the answer for this query is not true so the program displays “no.”.

2.4 Subject Area

The subject area I have chosen for my database is a booking system for a football training system, which football players can register to and book training sessions they desire to do. The main purpose of this database is to help a football coach keep all his training players in order, keeping track of the training sessions he has booked with them. I will want the database to have a registering player’s option so that the coach can see the amount of players under his training. The coach will also want a way the players can see all the different training sessions he is providing. He will also want a way In which his players can book a training session with him and saved on the program to keep everything safe.

2.5 Software’s

This section will be talking about all the different software's I will need to use in order to complete the given task, which is database manipulation using Prolog.

The first piece of software I will need is a text editing software, which I chose to use Notepad++. This a text editing software and can used to edit source codes on Microsoft Windows. I will use this to write my Prolog scripts, the Prolog scripts I code will have an extension such 'name.pl' that will be recognised by the SWI Prolog software. The location path of the file will be need, for that the program can compile it.

The next piece of software, I will need is a web browser. It can be any preferred browser, so I chose Google chrome because it was the one already Installed on my system so it was convenient for me to use. This browser combines a minimal design with sophisticated technology to make the web faster, safer and easier. My machines operating system is Windows so I found it suitable to use Google Chrome for my research I will be conducting.

Another piece of software I need to use is GitHub, I will using this to submit the work I have done. This is so the second reader can make a clone and have a look at it as well. GitHub is known to be the best place to share code with people you know or complete strangers. It is very popular, because statistics show that over eight million people use it to work on projects together.

Chapter 3 - Design

In this section, I will be discussing all the design procedures that have to in to making the completed design. The database model design is an extremely important link in the design of management information systems, and it is the basis of the applications. The database design model is good or bad will directly affect the success of the whole database. A good database design model design can make the application system high efficiency, maintenance simple and user friendly. It can also improve the application performance and life cycle. I will discuss the database model deign combined with a school's teacher database.

A data model is the data organisation to form a database. It is a model representation of all the data the database will contain and how to organise and give it a definition. A database not only represent what information is stored, but the most important thing is to represent the connection between various data by a certain data structure., which can reflect the associated information.

Before starting the implementation of the product, the model of the system must be constructed. This process consists of identifying what actions will take place, and how the data is transmitted throughout the database by those actions. A popular method that is used is the use of Unified Modelling Language or known as simply (UML). What UML focuses on is an object-oriented approach to the systems analysis and design, which will provide a well-structured system to implement once finished. The completion of designing a model of the system, only then we can work how the data will affect each other's behaviours, such as the transactions that occur from the user queries. You will have a clear idea of the data works together to fulfil the user's actions [Taylor, Allen G (1998)].

3.1 Models

The first step of creating the model is to identify the use cases the database will include the use of use case specifications. These outline the pre-conditions, the flow of events that happen in the system. The post conditions and any alternative routes to the original flow of events. By doing this we can begin to plan the behaviour and the processes behind each actions that occurs in the system Taylor, Allen G (1998).

3.1.1 Use Case Specifications

The first use case for this database is as follows

Use case Name
Register

Description
Player registers to the database

Pre-conditions
Player is new to the program and wants to register

Flow of events

- Player accesses the user Interface of the Prolog program
- Player chooses the register option
- Player types desired username
- Player types desired password
- Inputs checked for duplicates

- Player username and password inserted into the database

Alternative flow of events
Duplicate – If the username and password matches, an existing combination the player is shown an error message stating them to use a different combination.

Post Conditions
The player registering was successful and they can login to the system.

Use case Name
Login

Description
Player logging into the database

Pre-conditions
Player must be registered to the football training system

Flow of events
- Player accesses the user Interface of the Prolog program
- Player types in their username
- Player types their password
- Player hits Enter
- The inputted username and password combination is queried against the data in the database
- Player successfully logs into the system.

Alternative flow of events
Invalid User – If the username and password does not match a combinations saved on the database the player will have a login-failed message and prompting them to try again.

User is not registered - If the username and password does not recognise them, the player will be prompted with an error message asking to double check their login details or register if they have not yet done so.

Post Conditions
The user login was successful and they can access the rest of the program features
The user fails logging and redirected to login again or registered

Use case Name
Browse training sessions

Description
Player browses different training sessions

Pre-conditions
Player must be registered and logged in to the football training system

Flow of events
- Player accesses the user Interface of the Prolog program
- Player queries database for training sessions
- Training sessions are displayed
- Player chooses desired session

Alternative flow of events
Invalid User – If the username and password does not match a combinations saved on the database the player will have a login-failed message and prompting them to try again.

User is not registered - If the username and password does not recognise them, the player will be prompted with an error message asking to double check their login details or register if they have not yet done so.

Post Conditions
The player finds desired session and moves on to book it.

Use case Name
Booking

Description
Player books a training session

Pre-conditions
Player must be registered and logged in to the football training system

Flow of events
- Player accesses the user Interface of the Prolog program
- Player access the booking features
- Player chooses the date
- Player chooses the time
- Player saves changes
- Player enters his booking

Alternative flow of events
Invalid User – If the username and password does not match a combinations saved on the database the player will have a login-failed message and prompting them to try again.

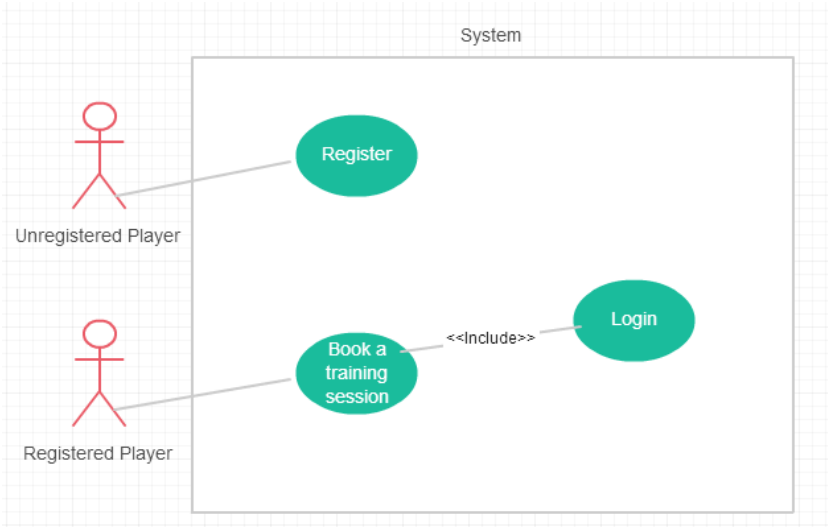
User is not registered - If the username and password does not recognise them, the player will be prompted with an error message asking to double check their login details or register if they have not yet done so.

Post Conditions

The player booking is successful and saved for the coach.

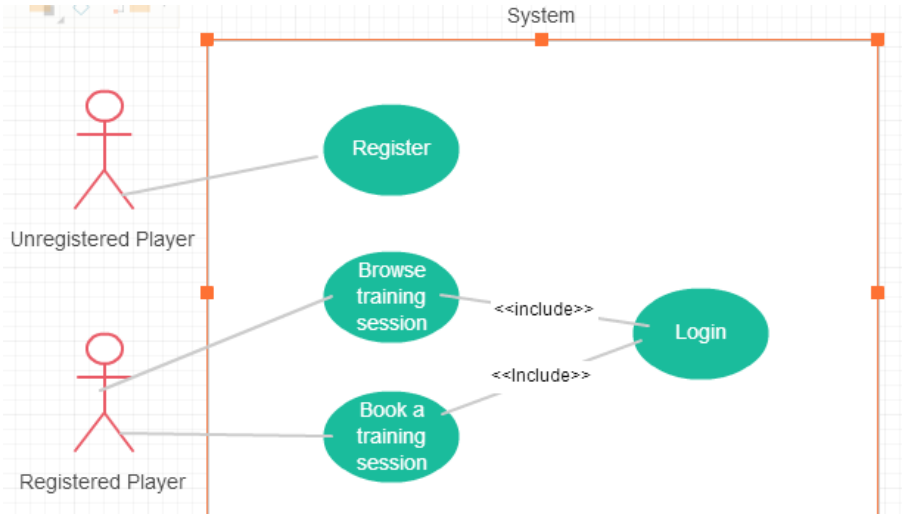
3.1.2 Use Case Diagram

Now, that I have outlined and explained the different use cases that are going to include in the system, I can now build a use case diagram which graphically displays the system. This is includes the actors such as the player and the coach; this is represented by the figure below.



(The first draft of the use case diagram for the database system)

As you can see the use, case for this system is very simple and easy to understand, which is why I thought it was incomplete but it represents everything the system is supposed to do. Firstly, as you can the unregistered player has the only option to register and nothing else. This is because to be able to access the database system you must first register. Once, the registering process has been successful, the player details are inputted into the system and given the ability to make a booking for a football training session. As you can see, from the diagram, the book a training session use case includes login use case and there is no way to use the booking option without it. However, I have decided to add an option where the player can query the Prolog database to browse different training sessions available. Therefore, I have considered this new idea and added a use case for the registered player to be able to browse football-training sessions. As you can see the figure below, this is the final use case diagram I think represent the processes, which take place in the system.



(Final use case diagram design for the database system)

3.2 The user Interface and Database

When the user executes the Prolog file, I want the user interface on swi-prolog to prompt the player to enter their username and password in order to move forward. When the player fills in the required information, the program will compare the combination to the saved one that are in the database. If it finds a match for that specific combination, the player is granted access to the system and do things such as bookings. However, if the login process fails because the program could not find a match in the Prolog database, the player is either given the option to try again due to a mistake they could have made, or given the option to register. The interface will allow the user to query the database using Prolog data manipulation methods to retrieve a list of training sessions provided by the coach. I want the Prolog to have an interface, which includes a menu, for example a register page, which the player can register to in order to able to access the system. Once registered I want the Prolog program to input the register details of the player username and password into a list in the program. This list will be used to compare future login from players in order to authenticate their login username and password combinations. This will give the program to be able to reject any unrecognised input combination and only letting registered player's access to the system. If the program does not recognise the username and password, I want it to prompt the player to try the login again or register. Once logged in, I want the user interface to include a menu of options they would like to do, such as browsing the list of training sessions available to book. After, the player finds the desired training session, I want to an option to book that particular session and inputted into the database. The tables of this database will include a username and password table, which keeps a record of all the registered players; this is also the table the program will consult in the authentications of player logins. Another table I want is one, which holds information on the different kinds of training sessions available to the players. The program will consult this table when the player queries the database for a list of training sessions. The final table will have the purpose of holding information on bookings players have made. The coach can consult this

table to see the bookings dates and times the players have made. As described before in my subject area I have chosen to do, the database will be made for a football training facility, which allows players to register, and book training sessions they desire to go through. However, the database will created using the Prolog programming languages, which uses similar methods as does a MySQL database does, but just a little different. The Prolog database will be created on a text editor such notepad++, and compiled and tested using Swi-Prolog. These two programs are both free sources I found online and are required to do this. The Prolog database will be populated with lists of facts and rules applied to them to fulfil a particular query.

The following tables will have to implemented into a Prolog program, using facts to represent them. The first table is the table which holds all the username and passwords of the players. The SQL database representation of this table will be shown in an example below. It shows a table with two columns called username and password, which what they store is the inputs entered by the player at registration. The program will insert this in to a list to keep them on record.

USERNAME	PASSWORD
PlayerFooty12	qwerty
ShootingExpert	123456

Following the same principle, the other two tables will have a similar format to this one. The second table will store the different training sessions the coach offers to his students, and the third table will store the bookings the players make. However, they will have different attribute and entities. What I must do is interpret these tables as Prolog language using the correct syntax.

Chapter 4 - Implementation and Testing

This section I will be talking about the implementation and test process in order to complete the Prolog database. I managed to do very basic code implementation, due finding it difficult to grasp the way the Prolog programming language works. With this being the case I have populated the code file with facts and rules and queries.

The facts which were included were facts such as the player names. I applied some rules on them to make them have a meaning. An example is working out if two given players are partners and friends with each other. These rules use the football training pairs facts. The way it works is that if two given players are both in pairs, they will be friends. The logic behind this is that if they are in pairs in both occasions they are likely to be friends. An example of these tasks are shown below.

```
pairs(ashraf, sisoko).

pairs(sisoko, ashraf).

pairs(milly, sisoko).

pairs(sisoko, milly).

pairs(ashraf, milly).

pairs(claire, sisoko).


partners(X,Y):-
        pairs(X,Y),
        pairs(Y,X).


friends(X, Y) :-
        pairs(X, Y);
        pairs(Y, X).
```

There is a list which shows the player football training schedules they booked, showing their name, exercise type and the dat. Using these facts, it can yield queries such as to see which players are better at a particular exercise opposed to another player. I done this by using the correct operators to represent the argument.

An example of this is shown below

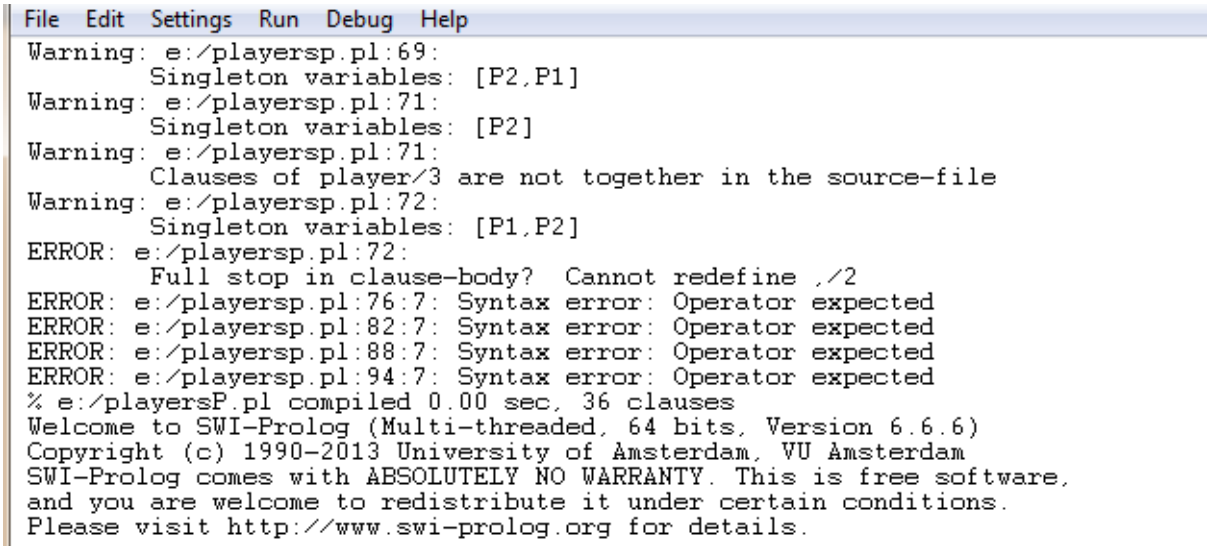
```
/*Rules who has better dribbling */


player(ashraf, dribbling, everyday).
```

```
player(sisoko, dribbling, monday).


better_than(ashraf, sisoko) :-
        player(ashraf, dribbling, everyday).
        player(sisoko, dribbling, monday)
        write(ashraf), write(' has better dribbling than '), write(sisoko), nl.
```

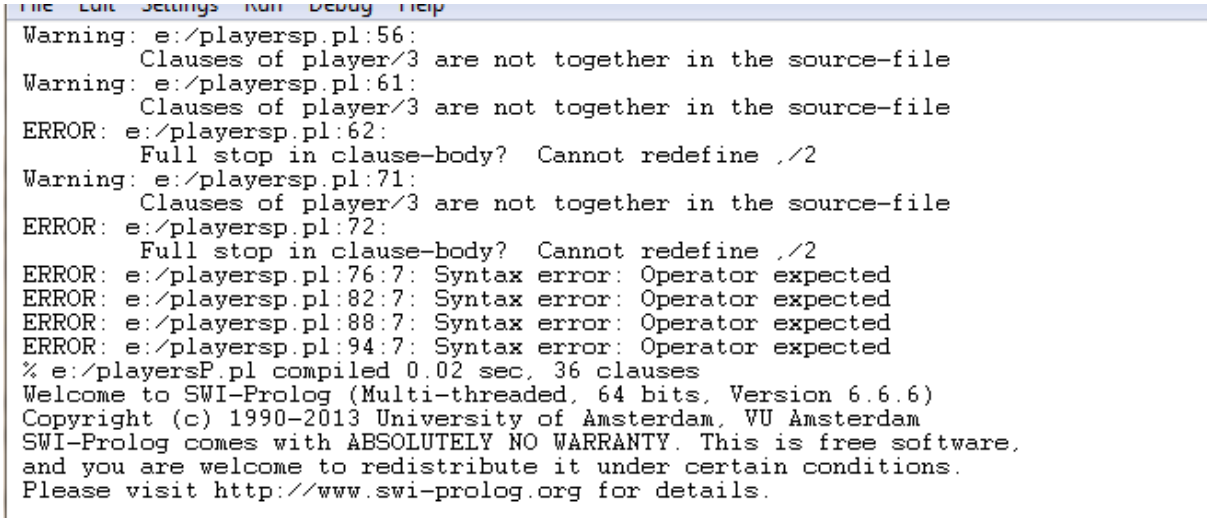
I added a section which represented the training sessions of players. My initial design states a booking feature however I do not know how implement this using Prolog. It keeps a record of the training, day, time and the players it involves. The example below shows my attempt at this.

```
/*Training sessions*/


Session(
        training,
        day(monday),
        time(10:00, 11:30),
        player(ashraf, sisoko)).


Session(
        training,
        day(tuesday),
        time(11:00, 12:30),
        player(milly, claire)).
```

When I came to running some queries on the database, I have faced singleton variable error and other syntax errors. The singleton errors showed in the screen shot below were telling me that some of the variables were only mentioned once in the rule it appears in. Such variables are often but not always spelling mistakes. As you can see below the variables 'P1' and 'P2' were causing errors, which I then fixed by replacing them with the correct variable names.



The second error problems I had was just simply the syntax errors, these are errors which do not comply with the Prolog syntax layout. The output gave the location of the errors and what it expected. The screen shot below shows proof of the error message that the Prolog interface displayed. I was not able to fix some these errors, so the basic code contains them.



Chapter 5 – Evaluation

The proposed objectives at the start of the project were as follows:

1. Learn basics of Prolog
2. Design the Database
3. Implement Prolog code to create the database
4. Create Prolog queries
5. Test the Database

I can say that I have learned the basics of the Prolog language and I understand it, considering that I have never used or heard of Prolog before this. I have done the design section of this database and used different modelling methods to come up with model to represent the data that will be included into the database. I have attempted the implementation of the design in Prolog code, but I found it difficult to know how to create things such as the menu and the login features of the program. These features were the desired things I wanted my program to include but did not manage to get to this stage. I ran simple queries on the facts and rules I have managed to and tested it to show if the queries showed the right results.

Chapter 6 – Conclusion

With the desired application partially completed, however it needs a lot work to be done to meet all the design requirements. But for now, I will be looking at the aim and learning outcomes that were outlined in previous chapter. And derive what was achieved, what wasn't achieved, what was learnt, what could be improved and what could be done in the future. The original The aim of this project is to provide users a way to store data in controlled area. Also the completion of the product, it must illustrate obtained Prolog scripts by application to a specific database.

Firstly what was achieved, was a completed design of the desired database, containing everything that was to be included into it, and a basic attempt at the code itself in the implementation. What wasn't achieved was a completed fully function database problem. What I learnt is the basics of the Prolog programming language and how facts, rules and queries work. The one thing that could improved was the code implementation, by maybe making the code more complex by including different functions, such as a menu for the user. What could be done in the future is maybe making a design that can implemented by a beginner with the Prolog language such as myself. In conclusion, I have tried my best to do the Prolog database manipulation and test it. I have been given the opportunity to base the

database on any subject area I desired, and I chose to do it for a football training coach to be able to keep track of his players.

References

James, Berrington (2014), Databases, Royal College of Anaesthetists CPD matrix, Electronic Article.

Kuo (1987), Implementation Of Prolog Database System, Journal of information processing, Electronic Article.

Taylor, Allen G (1998), SQL dummies, Foster City, IDG books 3rd edition, Book

Jan Wielemaker, Leslie De Koninck, Thom Fruehwirth, Markus Triska, Marcus Uneson, “SWI Prolog Reference Manual 7.1 ”, BoD – Books on Demand, 2014.