1- Introduction

SQL (pronounced "ess-que-el") stands for Structured Query Language. SQL is used to communicate with a database. According to ANSI (American National Standards Institute), it is the standard language for relational database management systems. SQL statements are used to perform tasks such as update data on a database, or retrieve data from a database. Some common relational database management systems that use SQL are: Oracle, Sybase, Microsoft SQL Server, Access, Ingres, etc.

The SQL Standard has gone through a lot of changes during the years, which have added a great deal of new functionality to the standard, such as support for XML, triggers, regular expression matching, recursive queries, standardized sequences and much more.

The SQL language is based on several elements. For the convenience of SQL developers all necessary language commands in the corresponding database management systems are usually executed through a specific SQL command-line interface (CLI). These commands can be grouped in the following areas:

- Clauses - the clauses are components of the statements and the queries;
- Expressions - the expressions can produce scalar values or tables, which consist of columns and rows of data;
- Predicates - they specify conditions, which are used to limit the effects of the statements and the queries, or to change the program flow;
- Queries - a query retrieves data, based on a given criteria;
- Statements - using statements user can control transactions, program flow, connections, sessions, or diagnostics. In database systems the SQL statements are used for sending queries from a client program to a server

where the databases are stored. In response, the server processes the SQL statements and returns to the client program. This allows users to execute a wide range of fast data manipulation operations from simple data inputs to more complex queries.

1.2 Declaring tables

The first step when creating a new database application is the process of declaring the tables. In our example we have 4 tables to be created: Products, Orders, OrdersProducts and Customer. We will start by the creation of the table Products. The table Products records all the products that the company has in its catalogue. The script for the creation of this table is given below. The same script can be used to MySQL, SQL Server and Oracle DBMS.

```
create table Products
(
 cod_product integer,
 description varchar(50) NOT NULL,
 unit_price DECIMAL(10,2),
 available_stock integer,
 minimal_stock integer default 0,
 CONSTRAINT Products_pk PRIMARY KEY (cod_product)
);
```

The primary key is the "cod_products", which is declared in the last line. The description can has a maximum size of 50 characters and cannot be null. The field "unit_price" is declared has a decimal. The maximum number of digits of "unit_price" may be specified in the first parameter; the maximum number of digits to the right of the decimal point is specified in the second parameter. The field "minimal_stock" has a default value of 0, if this information is not given when the user inserts new data.

Then, we will create the table Customers. Like in previous example the same script can be used for all three databases (MySQL, SQL Server and Oracle).

```
create table Customers
(
 cod_customer integer,
 name varchar(50) NOT NULL,
 address varchar(95) Default 'Unknown',
 zip_code char(8),
 country varchar(40) Default 'Portugal',
 telephone varchar(15),
 CONSTRAINT Customers_pk PRIMARY KEY (cod_customer)
);
```

The primary key is the "cod_customer", which is declared in last line. The address is also a varchar type like name, but if no information is provided it will assume the default value of "unknown". The zip code is a char(8) and it will always reserve a char of size 8, independently of its content. The country of the customer will assume the default value of "Portugal". The others elements have a behavior similar to previous example.

Then, we will create the table Orders. Like before the script for both databases are the same.

```
create table Orders
```

```
(
 cod_order integer,
 date_order date,
 date_delivery date,
 cod_customer integer,
 CONSTRAINT Orders_pk PRIMARY KEY (cod_customer),
 CONSTRAINT Orders_Cust_fk FOREIGN KEY (cod_customer)
 REFERENCES Customers(cod_customer)
);
```

The primary key is the "cod_order". However, the Orders table also has a foreign key in the field "cod_customer". The foreign key is connected to the primary key of the table "Customer" by the field "cod_customer". It is also relevent to highlight the declaration of two variables of the date type. The date format is represented by "YYYY-MM-DD" and its supported range is from '1000-01-01' to '9999-12-31'.

Finally, the script for OrdersProducts is presented. Here the script below only works in MySQL due to the int() type.

```
create table OrdersProducts
(
 cod_product integer,
 cod_order integer,
 quantity int(2),
 CONSTRAINT OrdersProducts_pk PRIMARY KEY (cod_product,
cod_order),
 CONSTRAINT OrdersProducts_Prod_fk FOREIGN KEY (cod_product)
 REFERENCES Products(cod_product),
 CONSTRAINT OrdersProducts_Orders_fk FOREIGN KEY (cod_order)
 REFERENCES Orders(cod_order)
);
```

The primary key is composed by the fields "cod_product" and "cod_order". The quantity is declared an integer where the maximum number of digits is specified in parenthesis. Finally there are two foreign keys: the first one connected to the table Products; and the last one associated to the Orders table. The fields "cod_product" and "cod_order" are simultaneously primary and foreign keys.

1.3 Insert Data to the data base table

The insertion of new data in a database can be done by following two different, but similar, approaches:

- The first form does not specify the column names where the data will be inserted, only their values;
- The second form specifies both the column names and the values to be inserted.

For the introduction of data in tables "Products" and "Customers" we will adopt the second approach.

```
Insert Into Products (cod_product, description, unit_price)
Values (1, 'Eggs', 2.49);
Insert Into Products (cod_product, description, unit_price)
Values (2, 'Ice Cream', 3.99);
Insert Into Products (cod_product, description, unit_price)
```

```
Values (3, 'Soda', 0.65);
Insert Into Products (cod_product, description, unit_price)
Values (4, 'Cheese', 2.89);
Insert Into Products (cod_product, description, unit_price)
Values (5, 'Pork Meat', 3.10);
Insert Into Customers (cod_customer, name)
Values (1, 'Anne');
Insert Into Customers (cod_customer, name)
Values (2, 'Peter');
Insert Into Customers (cod_customer, name)
Values (3, 'Elena');
Insert Into Customers (cod_customer, name)
Values (4, 'Shirley');
Insert Into Customers (cod_customer, name)
Values (5, 'John');
```

In table "Products" we will introduce data related to its code of product, description and unit price. On the other side, we only introduce information regarding the code of customer and its name in table "Customers". In any situation it is mandatory to fill the primary key fields. The other fields will assume the default value if it was declared in the creation table process.

In order to place new data in table "Orders" we will use the script below that gives data values in all fields.

```
Insert Into Orders Values (1, '2015-12-21', '2015-12-21', 1);
Insert Into Orders Values (2, '2015-12-22', '2015-12-23', 1);
Insert Into Orders Values (3, '2015-12-22', '2015-12-27', 2);
Insert Into Orders Values (4, '2015-12-27', '2015-12-30', 3);
Insert Into Orders Values (5, '2015-12-30', '2015-12-31', 3);
```

The customer with code "1" and "3" will have 2 orders recorded in the database. The customer with code "2" will only have one order. Finally the customers with code "4" and "5" don't have any order. It is also important to highlight that the primary field must be always unique.

Finally, we will adopt the script below to include information regarding each order. This information is stored in table "OrdersProducts".

```
Insert Into OrdersProducts Values (1, 1, 1);
Insert Into OrdersProducts Values (2, 1, 1);
Insert Into OrdersProducts Values (1, 2, 2);
Insert Into OrdersProducts Values (5, 3, 7);
Insert Into OrdersProducts Values (4, 3, 4);
Insert Into OrdersProducts Values (3, 3, 5);
Insert Into OrdersProducts Values (2, 3, 5);
Insert Into OrdersProducts Values (1, 4, 8);
Insert Into OrdersProducts Values (2, 4, 2);
Insert Into OrdersProducts Values (1, 5, 3);
Insert Into OrdersProducts Values (2, 5, 3);
Insert Into OrdersProducts Values (4, 5, 5);
```

The Order with code "1" and "4" have two products; the Order with code "5" has three products; the Order with code "3" has 4 products; finally the Order with code "2" has only one product.

HW

By using SQL instructions create hospital database with two tables; doctor(doc_id, doc_name, doc_age, doc_spec, doc_address, doc_tel) paitient(pait_id, pait_name, doct_id, pait_room, pait_age, pait_address, entry_date)

And add at less six records for each creating table.