CSIS05I

Database Systems II

Lab (1)

General revision

Lab (1)

The Overview

The labs will cover all practical parts of the database systems module. Students will learn how to use Microsoft SQL Server to create and manipulate databases. As mentioned, we would be using SQL as our main language. SQL (Structured Query Language) is the database language used for data definition and manipulation. SQL is composed of **DDL** and **DML** statements. **Both** are needed to be able to develop and manipulate a database.

1. DDL (Data Definition Language):

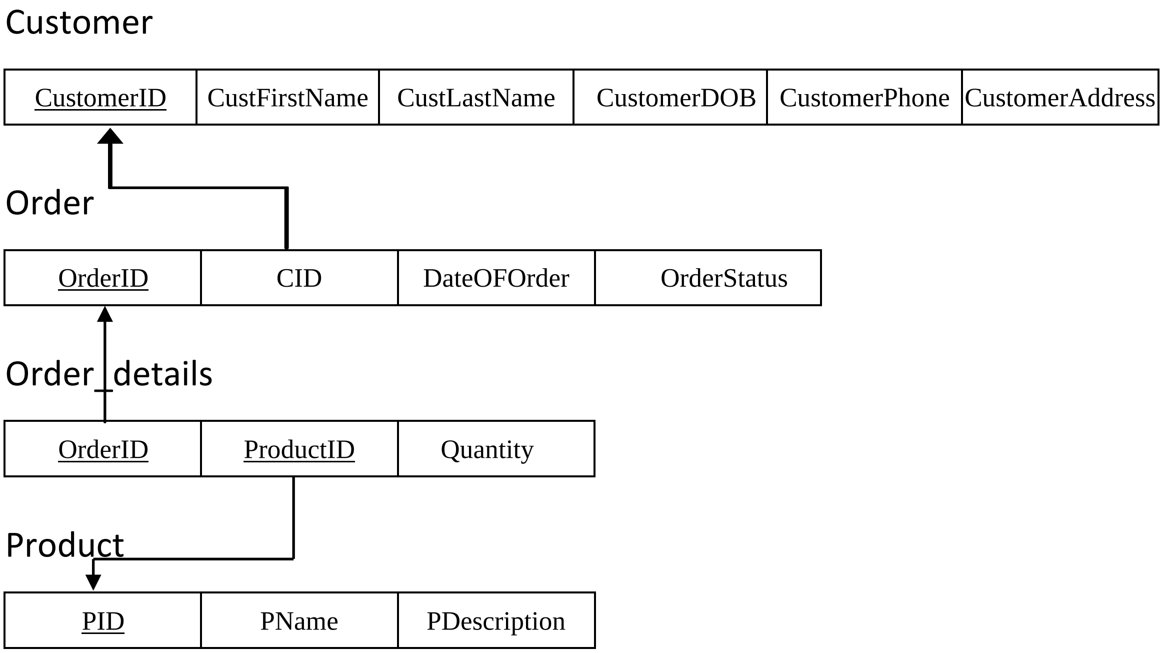
It includes statements that are used to create/update/delete tables as well as define the constraints of a database. When creating a table, the user must specify 3 things: Attributes, Data Types and Constraints (Ex. Primary Keys and Foreign Keys).

2. DML (Data Manipulation Language):

It includes the use of SQL statements that are used to insert data into relations, and can modify or delete them later. Also, DML allows a user to search for and retrieve specific data from the database.

In the lab, we will be revising on the SQL by building a database for a store. The idea is to remember the SQL structure and help you in the 2 lab exams.

Store Schema



Customer

CustomerID	CustFirstName	CustLastName	CustomerDOB	CustomerPhone	CustomerAddress

Order

OrderID	CID	DateOFOrder	OrderStatus

Order_details

OrderID	ProductID	Quantity

Product

PID	PName	PDescription

We will first start with the very basics, which are creating and deleting the database. In order to create anything within the database (including the database itself), we use the keyword **CREATE**. To delete anything within the database (including the database itself), we use the keyword **DROP.**

Create a new database schema called Store:

```
CREATE DATABASE Store;
```

To delete the database:

```
DROP DATABASE Store;
```

To create tables, you should have the table name, all the attributes and its data type then you should put constraints to this table like primary key and foreign key.

The structure of the table creation is

```
CREATE TABLE Customer
(
CustomerID          INT                 NOT NULL,
CustFirstName       VARCHAR(30),
CustLastName        VARCHAR(30),
CustomerDOB         DATE,
CustomerPhone       VARCHAR(30),
CustomerAddress     VARCHAR(30)         NULL,

CONSTRAINT Customer_pk PRIMARY KEY (CustomerID),

CONSTRAINT unique_name unique (CustFirstName)

);

CREATE TABLE Order
(
OrderID             INT                 NOT NULL,
CID                 INT                 NOT NULL,
DateOFOrder         DATE,

OrderStatus         VARCHAR(30),

CONSTRAINT Order_pk PRIMARY KEY (OrderID),

CONSTRAINT Customer_Order_fk
FOREIGN KEY (CID) REFERENCES Customer (CustomerID)

);
```

Note that:

*PK and Unique constraints are called **Entity integrity**. While **Referential integrity** is the relationship between the tables (FKs). There is also **Domain integrity** which will be explained in the upcoming labs.*

Modification / Deletion of Relations:

To add/delete an attribute in a relation (table):

Example:

We want to add a new attribute to a relation:

```
ALTER TABLE Customer
ADD IsMarried CHAR;
```

We no longer need the IsMarried attribute of Customer, so to remove it:

```
ALTER TABLE Customer
DROP COLUMN IsMarried;
```

To add/delete a constraint to a relation:

Example:

We want to add a primary key to a table that was created without one:

```
ALTER TABLE Customer
ADD CONSTRAINT Cust_PK PRIMARY KEY (CustomerID);
```

To delete a constraint from a relation:

```
ALTER TABLE Customer
DROP CONSTRAINT Cust_PK;
```

Insert a new row into a relation:

In order to insert values for all the attributes, the following SQL statement is used

Example:

```
INSERT INTO Customer VALUES ('1', 'Mark', 'Smith', '26/01/1993',
'01001589678', 'New Cairo');
```

To retrieve specific information from a database by using a certain format of command to provide one or more conditions for the retrieval is called Querying.

Writing a Query

This is the basic form of a query:

```
SELECT <Attribute List>
FROM <Table List>
WHERE <Condition>
```

<Attribute List>: is the list of attribute names whose values we want to retrieve from the database.

<Table List>: is a list of the relation(s) name(s) required to process the query.

<Condition>: is the condition specified by the user to retrieve relevant records ONLY.

Also, we may add extra conditions to the statement to have more specific retrieval with different format like order by and Group by.

```
SELECT     <Attribute List>
FROM       <Table List>
WHERE      <Condition>
GROUP BY <Condition>
HAVING     <Condition>
ORDER BY <Condition>
```

Example 1:

Retrieve the name of every Customer in the Customer relation.

```
SELECT DISTINCT CustFirstName + ' ' + CustLastName As
'Customer Name'
FROM Customer;
```

Example 2:

Retrieve the CustomerID of every Customer in the Customer relation arranged with their Last Name in descending order.

```
SELECT CustomerID
FROM Customer
ORDER BY CustLastName DESC;
```

Example 3:

Retrieve the birth date and address of the Customer whose first name starts with letter "M", or his last name ends with "H".

```
SELECT CustomerDOB, CustomerAddress
FROM Customer
WHERE CustFirstName like 'M%' OR CustLastName like '%H';
```

Example 4:

Retrieve number of customers in customer relation according to their date of birth.

```
SELECT COUNT (CustomerID), CustomerDOB
FROM Customer
GROUP BY CustomerDOB;
```

Note that: *there are other aggregate functions like sum(), avg(), min(), max().*

Example 5:

Retrieve the first and last name of all Customers along with their orders id and the status of the order.

```
SELECT CustFirstName, CustLastName, OrderID, OrderStatus
FROM Customer ,Order
WHERE CustomerID=CID;
```

OR

```
SELECT CustFirstName, CustLastName, OrderID, OrderStatus
FROM Customer JOIN Order
ON CustomerID=CID;
```

<u>Note that:</u> there are other join types like self-join, outer join (left outer join, right outer join, and full outer join).

Example 6:

Retrieve customer IDs along with the total number of their orders; only retrieve the customers who have made less than two orders.

```
SELECT CustomerID, COUNT (OrderID) AS 'Total Number of Orders'
FROM Customer , Orderr
WHERE CustomerID = CID
GROUP BY CustomerID
HAVING COUNT (OrderID) < 2;
```

Exercises:

1. Create database with Store name.

2. Create the tables and its relationships.

3. Insert 3 rows for each of the following relations:

   A. Customer
   B. Order
   C. Order_Details
   D. Product

4. Retrieve all the customers information.

5. Retrieve the products name, ID and its quantity.

6. Retrieve all the Customers ID and first name that have their order status is delivered.