CSIS05I

Database Systems II

Lab (3)

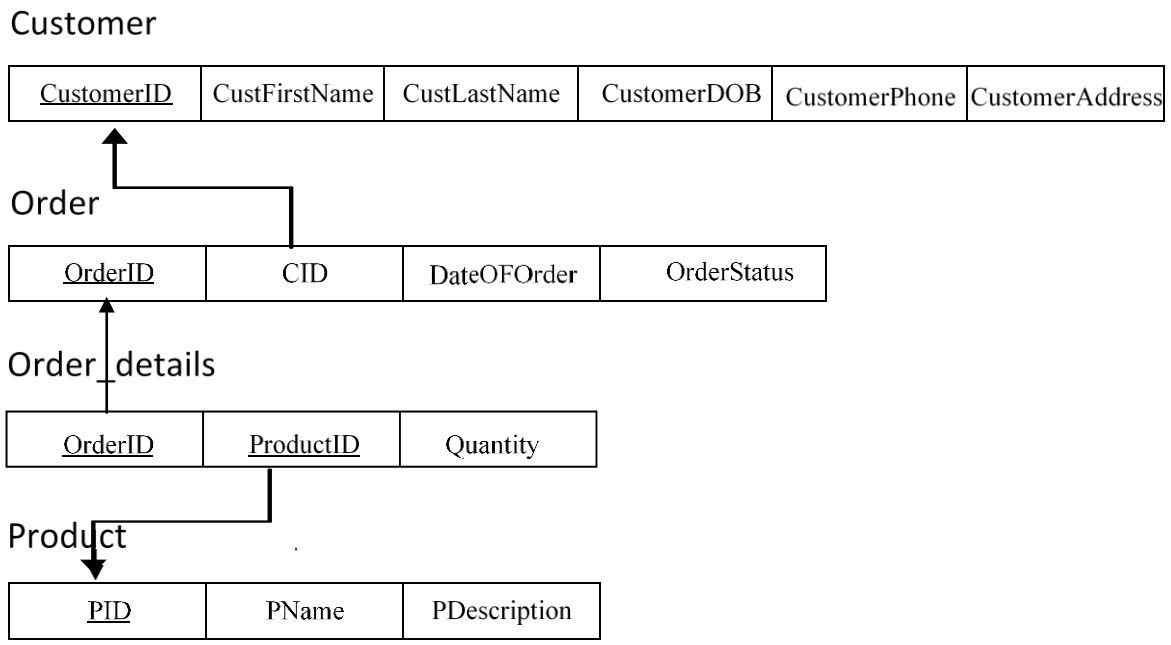
INDEXING

Lab (3)

The Overview

In this lab we will learn the indexing and how to create and manipulate it. But before that, we will learn what is auto increment and how to make it.

Store Schema:



Auto Increment Feature:

Auto-increment feature allows a unique number to be generated automatically when a new record is inserted into a table. Often this is the primary key field that we would like to be created automatically every time a new record is inserted.

Syntax:

The SQL Server uses the IDENTITY keyword to perform an auto-increment feature. In the DDL (within the Table Creation), we add the IDENTITY keyword to the targeted attribute definition AS:

```
IDENTITY (Starting Value, Increment Rate)
```

Example:

The following SQL statement defines the "CustomerID" attribute to be an auto-increment primary key field in the "Customer" table:

```
CREATE TABLE Customer (
CustomerID          INT         identity(1,1),
CustFirstName       VARCHAR(30),
CustLastName        VARCHAR(30),
CustomerDOB         varchar(30),
CustomerPhone       VARCHAR(30),
CustomerAddress     VARCHAR(30),

CONSTRAINT Customer_pk PRIMARY KEY (CustomerID)
);
```

In the example above, the starting value for IDENTITY is 1, and it will increment by 1 for each new record. ***Note:*** To specify that the "CustomerID" column should start at value 100 and increment by 5, change it to **IDENTITY(100,5).**

To insert a record into the "Customer" table, we will NOT have to specify a value for the "CustomerID" column (a unique value will be added automatically):

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

Indexing:

An index is a data structure that optimizes searching and accessing the data. One of the ways that will optimize your database searching and accessing is having indexes on the columns.

Advantages

- Indexes are used to retrieve data from the database very fast.
- The users cannot see the indexes, they are just used to speed up searches/queries.

We will first start with the very basics, which are creating and deleting the index. 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**.

Single-Column Indexes syntax

```
CREATE INDEX index_name
ON table_name (column);
```

Example 1:

Create an index named "idx_lastname" on the " CustLastName " column in the"Customer" table.

```
CREATE INDEX idx_lastname
ON Customer (CustLastName);
```

To create an index on a combination of columns, you have to list the column names within the parentheses separated by commas.

Composite Indexes syntax

```
CREATE INDEX index_name
ON table_name (column1, column2,……);
```

Example 2:

Creates an index named "idx_Fullname" on the "CustFirstName" column and "CustLastName" column in the "Customer" table.

```
CREATE INDEX idx_Fullname
ON Customer (CustFirstName,CustLastName);
```

The rename Index syntax

```
EXEC sp_rename
N'table_name.index_name',
N'new_index_name',
N'INDEX';
```

Example 3:

Rename the index idx_lastname to idx_LName

```
EXEC sp_rename
N'Customer.idx_lastname',
N' idx_LName',
N'INDEX';
```

The disable Index syntax

```
Alter Index index_name
ON table_Name
DISABLE;
```

Example 4:

Disable the 'idx_LName' on table customer

```
Alter Index idx_LName
ON Customer
DISABLE;
```

The enable Index syntax

```
Alter Index index_name
ON table_name
REBUILD;
```

Example 5:

Enable the disabled index 'idx_LName' on table customer

```
Alter index idx_LName
ON Customer
REBUILD;
```

The drop Index syntax

```
DROP INDEX table_name.index_name;
```

Example 6:

Drop the index that retrieve the full name in customer table.

```
DROP INDEX Customer.idx_Fullname;
```

There are several methods to find indexes on a table. The methods include using system stored procedures;such as sp_helpindex.

Find Indexes on a table using SP_HELPINDEX

"sp_helpindex" is a system stored procedure which lists the information of all the indexes on a table or view. This is the easiest method to find the indexes in a table. sp_helpindex returns the name of the index, description of the index and the name of the column on which the index was created.

SP_HELPINDEX Syntax

```
EXEC sp_helpindex 'TABLE-NAME'
GO
```

Example 7:

Retrieve all the information about the index in table customer.

```
EXEC sp_helpindex 'Customer'
GO
```

Exercises

1. Create index that have all customer ID from customer relation.
2. Create index that have the full name of the customer with its date of birth and ID.
3. Create index that have the product name along with its description.
4. Create index that have the order ID and the Product ID.
5. Retrieve all the index information from table customer.
6. Rename the index that contain the customer ID.
7. Delete the index that contains the customer ID.
8. Disable the index on product table.
9. Enable the index on product table.