Chapter 21: CREATE TABLE

Parameter	Details
tableName	The name of the table
columns	Contains an 'enumeration' of all the columns that the table have. See **Create a New Table** for more details.

The CREATE TABLE statement is used create a new table in the database. A table definition consists of a list of columns, their types, and any integrity constraints.

Section 21.1: Create Table From Select

You may want to create a duplicate of a table:

```
CREATE TABLE ClonedEmployees AS SELECT * FROM Employees;
```

You can use any of the other features of a SELECT statement to modify the data before passing it to the new table. The columns of the new table are automatically created according to the selected rows.

```
CREATE TABLE ModifiedEmployees AS
SELECT Id, CONCAT(FName," ",LName) AS FullName FROM Employees
WHERE Id > 10;
```

Section 21.2: Create a New Table

A basic `Employees` table, containing an ID, and the employee's first and last name along with their phone number can be created using

```
CREATE TABLE Employees(
    Id int identity(1,1) primary key not null,
    FName varchar(20) not null,
    LName varchar(20) not null,
    PhoneNumber varchar(10) not null
);
```

This example is specific to Transact-SQL

`CREATE TABLE` creates a new table in the database, followed by the table name, `Employees`

This is then followed by the list of column names and their properties, such as the ID

```
Id int identity(1,1) not null
```

Value	Meaning
`Id`	the column's name.
`int`	is the data type.
`identity(1,1)`	states that column will have auto generated values starting at 1 and incrementing by 1 for each new row.
`primary key`	states that all values in this column will have unique values
`not null`	states that this column cannot have null values

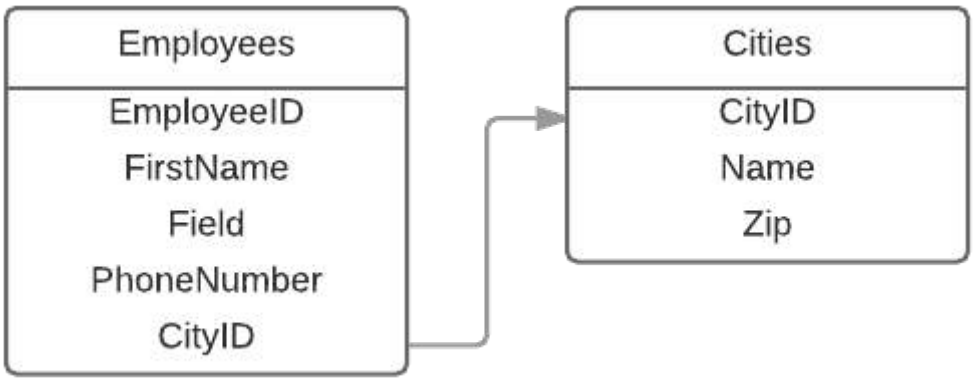
Section 21.3: CREATE TABLE With FOREIGN KEY

Below you could find the table `Employees` with a reference to the table `Cities`.

```
CREATE TABLE Cities(
    CityID INT IDENTITY(1,1) NOT NULL,
    Name VARCHAR(20) NOT NULL,
    Zip VARCHAR(10) NOT NULL
);

CREATE TABLE Employees(
    EmployeeID INT IDENTITY (1,1) NOT NULL,
    FirstName VARCHAR(20) NOT NULL,
    LastName VARCHAR(20) NOT NULL,
    PhoneNumber VARCHAR(10) NOT NULL,
    CityID INT FOREIGN KEY REFERENCES Cities(CityID)
);
```

Here could you find a database diagram.



The column `CityID` of table `Employees` will reference to the column `CityID` of table `Cities`. Below you could find the syntax to make this.

```
CityID INT FOREIGN KEY REFERENCES Cities(CityID)
```

Value	Meaning
`CityID`	Name of the column
`int`	type of the column
`FOREIGN KEY`	Makes the foreign key *(optional)*
`REFERENCES Cities(CityID)`	Makes the reference to the table `Cities` column `CityID`

Important: You couldn't make a reference to a table that not exists in the database. Be source to make first the table `Cities` and second the table `Employees`. If you do it vise versa, it will throw an error.

Section 21.4: Duplicate a table

To duplicate a table, simply do the following:

```
CREATE TABLE newtable LIKE oldtable;
INSERT newtable SELECT * FROM oldtable;
```

Section 21.5: Create a Temporary or In-Memory Table

PostgreSQL and SQLite

To create a temporary table local to the session:

```
CREATE TEMP TABLE MyTable(...);
```

SQL Server

To create a temporary table local to the session:

```
CREATE TABLE #TempPhysical(...);
```

To create a temporary table visible to everyone:

```
CREATE TABLE ##TempPhysicalVisibleToEveryone(...);
```

To create an in-memory table:

```
DECLARE @TempMemory TABLE(...);
```