Republic of Iraq
The Ministry of Higher Education
& Scientific Research
Al-Ma'arif University College
Department of Computer Science

Lecturer Name: Shihab Hamad Khaleefah
Academic Status: BhD. In Computer Science
Qualification: - Lecturer
Course Material: Database Management System
Grade Level: Second Stage, Lec. 4-5

DATA DEFINITION LANGUAGE OPERATORS (DDL) IN STRUCTURED QUERY LANGUAGE

Introduction to 'SQL' in Database Design and Management

SQL (Structured Query Language) is a database computer language designed for managing data in relational database management systems (RDBMS). SQL, is a standardized computer language that was originally developed by IBM for querying, altering and defining relational databases, using declarative statements.

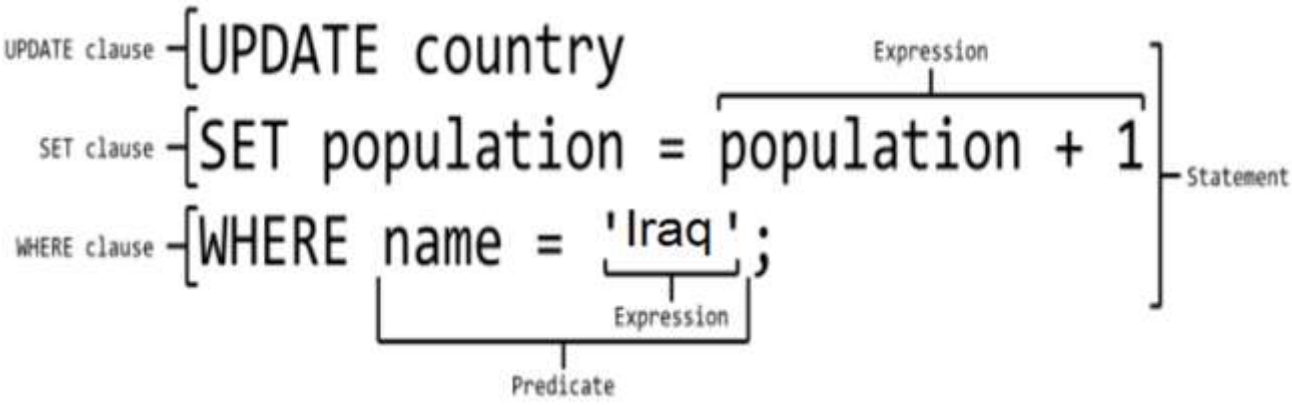
SQL – Structured Query language

A Database Computer Language designed for Managing Data in Relational Database Management Systems (RDBMS)

Query Examples:

```
• insert into STUDENT (Name , Number, SchoolId)
  values ('John Smith', '100005', 1)

• select SchoolId, Name from SCHOOL

• select * from SCHOOL where SchoolId > 100

• update STUDENT set Name='John Wayne' where StudentId=2

• delete from STUDENT where SchoolId=3
```

We have 4 different Query Types: **INSERT**, **SELECT**, **UPDATE** and **DELETE**



What can SQL do?

- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create new databases
- SQL can create new tables in a database
- SQL can create stored procedures in a database
- SQL can create views in a database
- SQL can set permissions on tables, procedures, and views

Even if SQL is a standard, many of the database systems that exist today implement their own version of the SQL language. In this document we will use the Microsoft SQL Server as an example.

There are lots of different database systems, or DBMS – Database Management Systems, such as:

• Microsoft SQL Server

o Enterprise, Developer versions, etc.

o Express version is free of charge

• Oracle

• MySQL (Oracle, previously Sun Microsystems) - MySQL can be used free of charge (open source license), Web sites that use MySQL: YouTube, Wikipedia, Facebook

• Microsoft Access

• IBM DB2

• Sybase

SQL Process

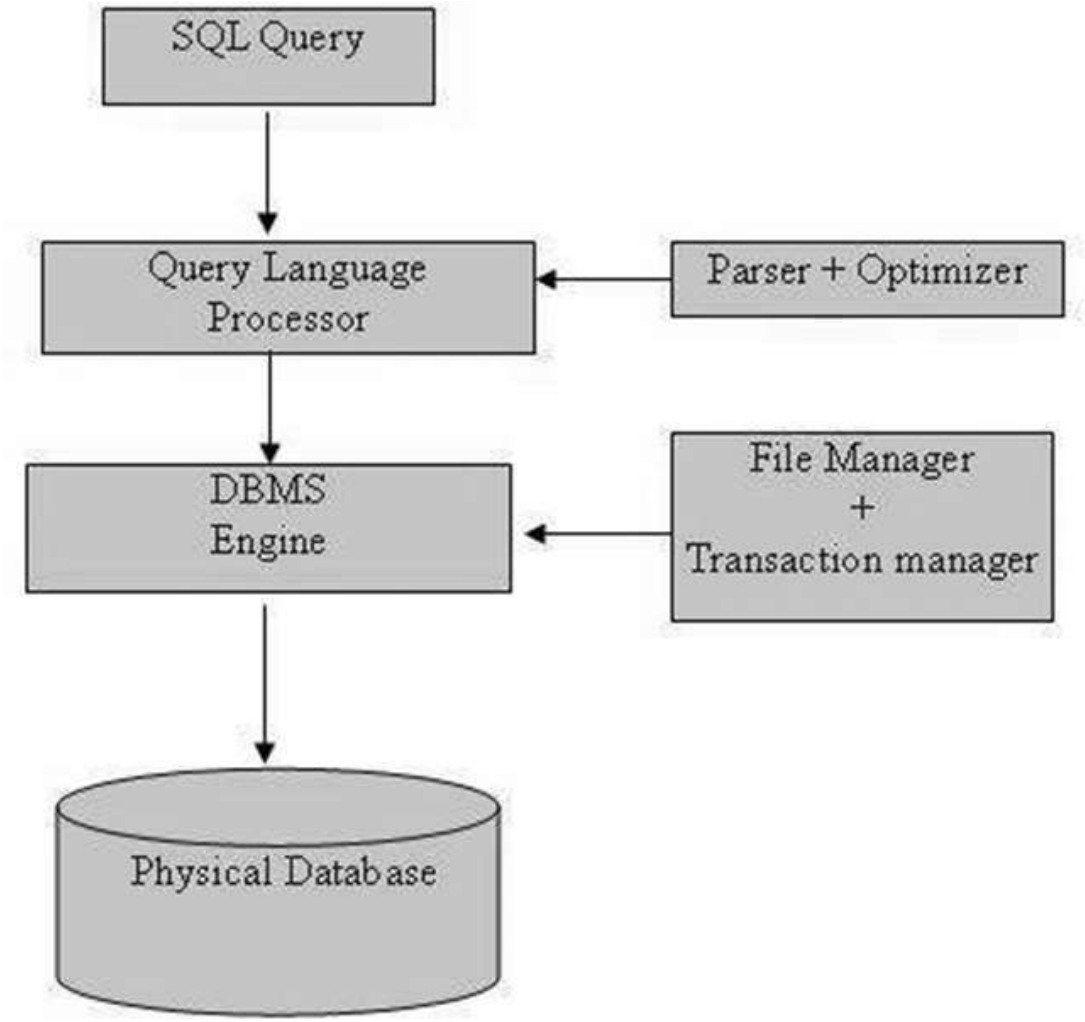
When you are executing an SQL command for any RDBMS, the system determines the best way to carry out your request and SQL engine figures out how to interpret the task. There are various components included in this process.

These components are :

- Query Dispatcher
- Optimization Engines
- Classic Query Engine
- SQL Query Engine, etc.

A classic query engine handles all the non-SQL queries, but a SQL query engine won't handle logical files.

Following is a simple diagram showing the SQL Architecture:



SQL Commands

The standard SQL commands to interact with relational databases are CREATE, SELECT, INSERT, UPDATE, DELETE and DROP. These commands can be classified into the following groups based on their nature:

1.1 Data Definition Language (DDL):

The Data Definition Language (DDL) manages table and index structure. The most basic items of DDL are the CREATE, ALTER, RENAME and DROP statements:

• CREATE creates an object (a table, for example) in the database.

• DROP deletes an object in the database, usually irretrievably.

• ALTER modifies the structure an existing object in various ways—for example, adding a column to an existing table.

1.2 Data Manipulation Language (DML):

The Data Manipulation Language (DML) is the subset of SQL used to add, update and delete data.

The acronym CRUD refers to all of the major functions that need to be implemented in a relational database application to consider it complete. Each letter in the acronym can be mapped to a standard SQL statement:

Operation	SQL	Description
Create	INSERT INTO	inserts new data into a database
Read (Retrieve)	SELECT	extracts data from a database
Update	UPDATE	updates data in a database
Delete (Destroy)	DELETE	deletes data from a database

1.3 DQL (Data Query Language):

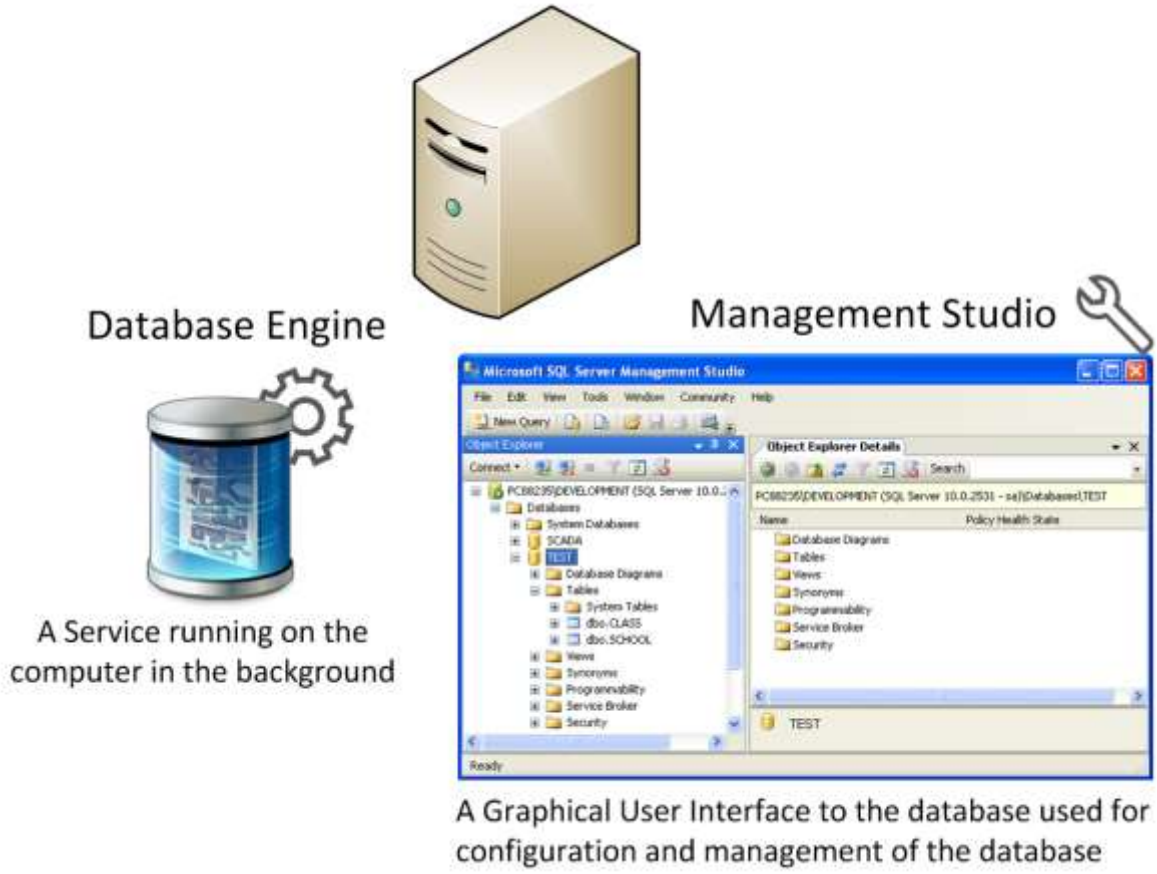
• SELECT

Introduction to SQL Server

Microsoft is the vendor of SQL Server. We have different editions of SQL Server, where SQL Server Express is free to download and use.

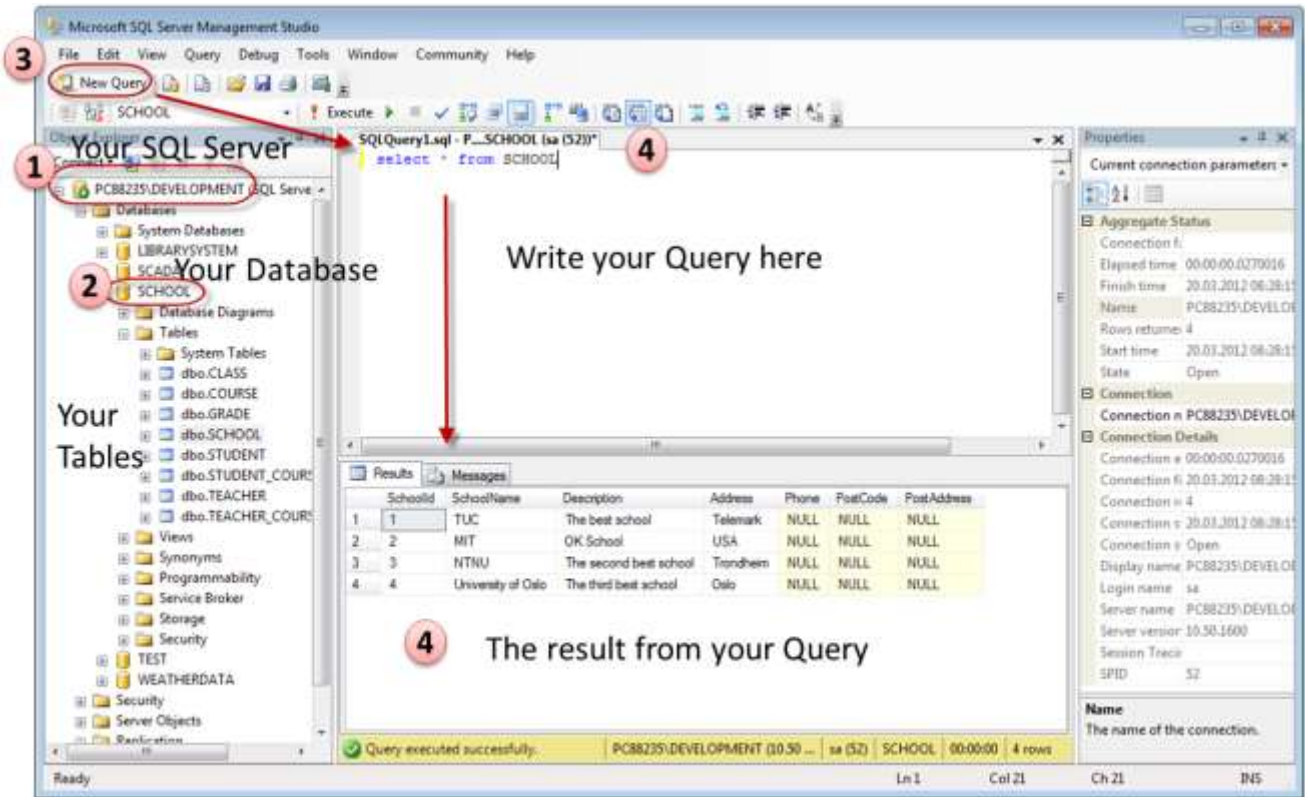
SQL Server uses T-SQL (Transact-SQL). T-SQL is Microsoft's proprietary extension to SQL. TSQL is very similar to standard SQL, but in addition it supports some extra functionality, builtin functions, etc. T-SQL expands on the SQL standard to include procedural programming, local variables, various support functions for string processing, date processing, mathematics, etc.

SQL Server consists of a **Database Engine** and a **Management Studio** (and lots of other stuff which we will not mention here). The Database engine has no graphical interface - it is just a service running in the background of your computer (preferable on the server). The Management Studio is graphical tool for configuring and viewing the information in the database. It can be installed on the server or on the client (or both).



SQL Server Management Studio

SQL Server Management Studio is a GUI tool included with SQL Server for configuring, managing, and administering all components within Microsoft SQL Server. The tool includes both script editors and graphical tools that work with objects and features of the server. As mentioned earlier, version of SQL Server Management Studio is also available for SQL Server Express Edition, for which it is known as SQL Server Management Studio Express.
A central feature of SQL Server Management Studio is the Object Explorer, which allows the user to browse, select, and act upon any of the objects within the server. It can be used to visually observe and analyze query plans and optimize the database performance, among others. SQL Server Management Studio can also be used to create a new database, alter any existing database schema by adding or modifying tables and indexes, or analyze performance. It includes the query windows which provide a GUI based interface to write and execute queries.

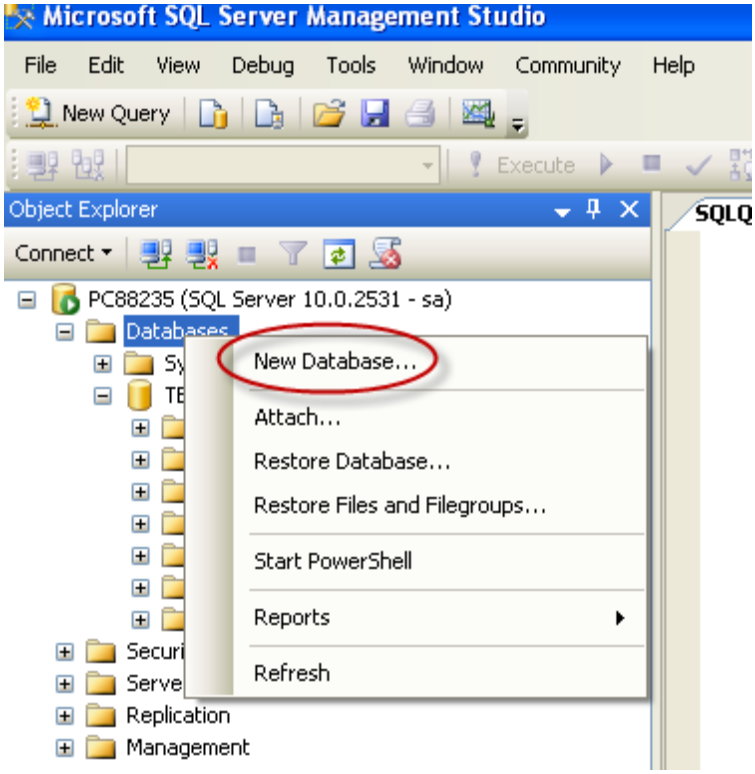


When creating SQL commands and queries, the "Query Editor" (select "New Query" from the Toolbar) is used (shown in the figure above).

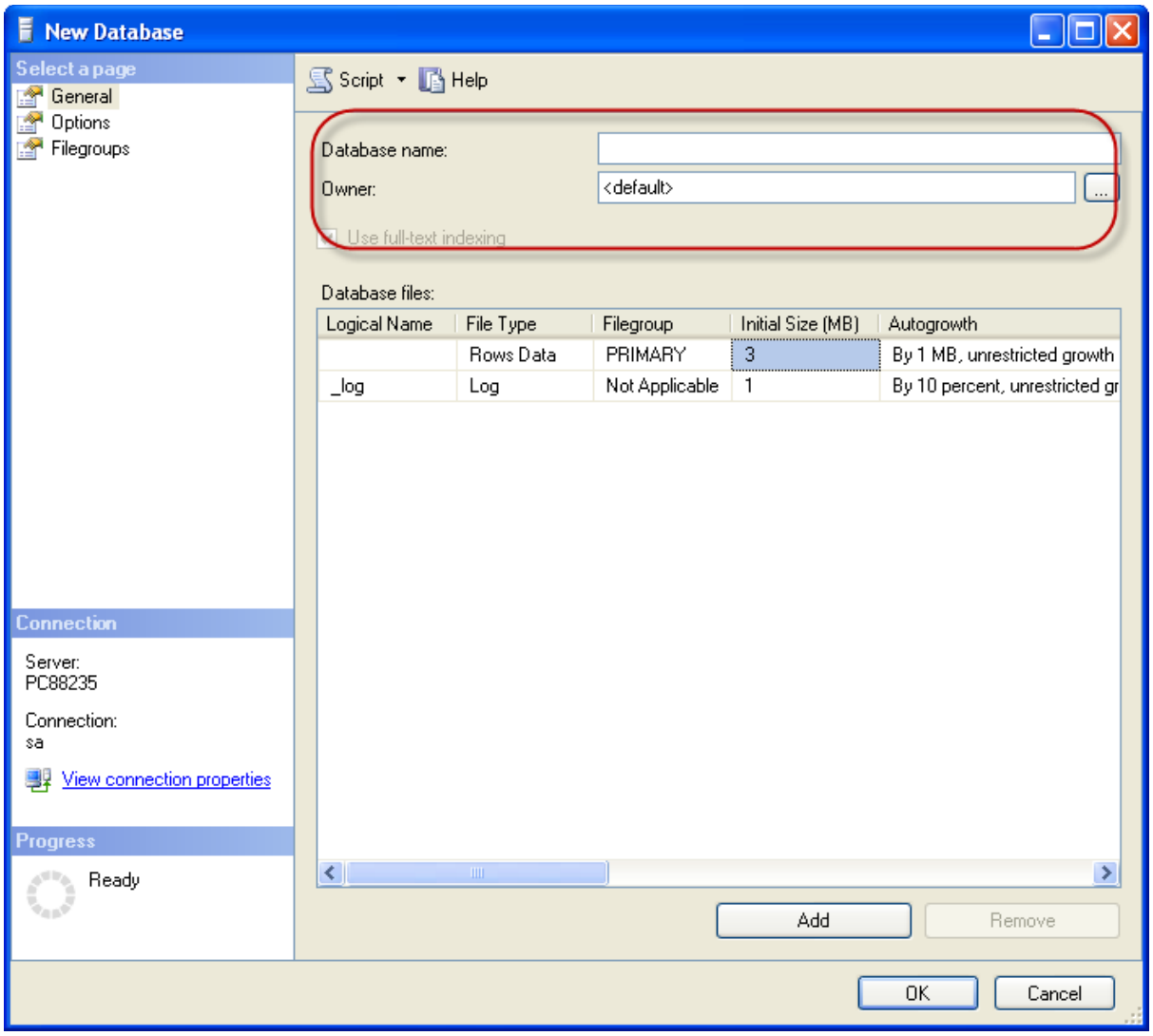
With SQL and the "Query Editor" we can do almost everything with code, but sometimes it is also a good idea to use the different Designer tools in SQL to help us do the work without coding (so much).

1.1 Create a new Database

It is quite simple to create a new database in Microsoft SQL Server. Just right-click on the "Databases" node and select "New Database…"



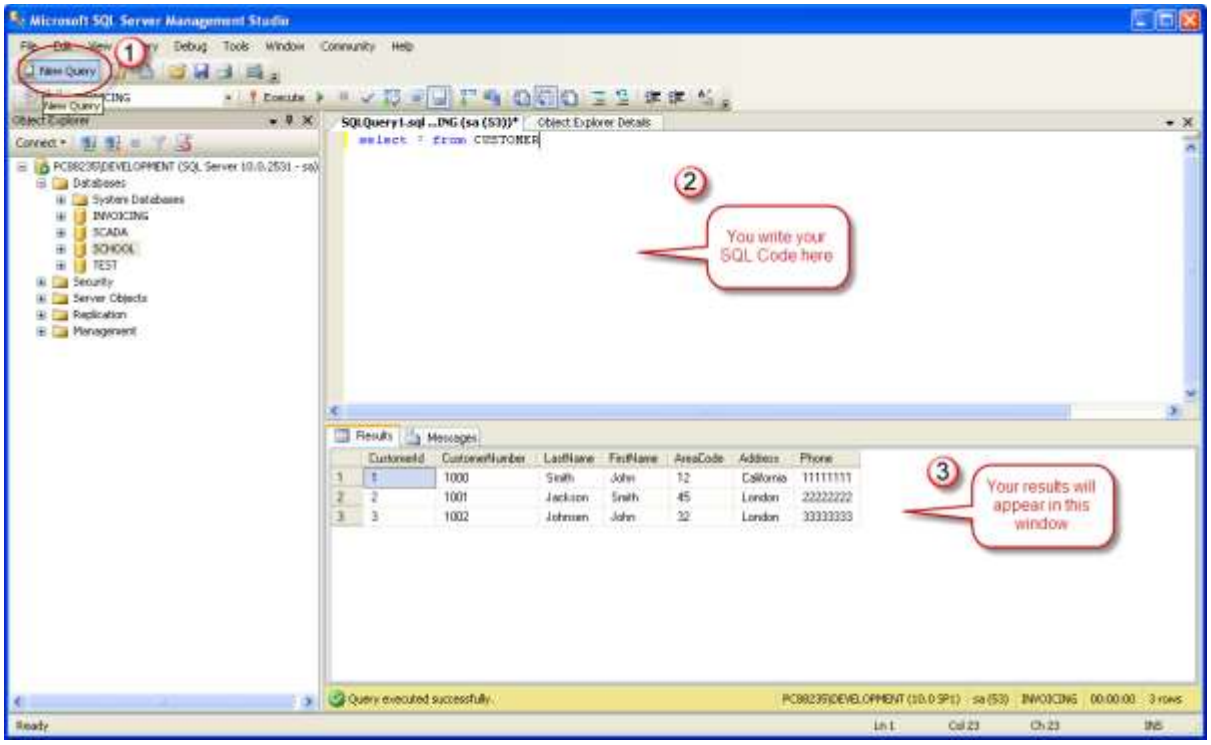
There are lots of settings you may set regarding your database, but the only information you must fill in is the name of your database:



You may also use the SQL language to create a new database, but sometimes it is easier to just use the built-in features in the Management Studio.

1.2 Queries

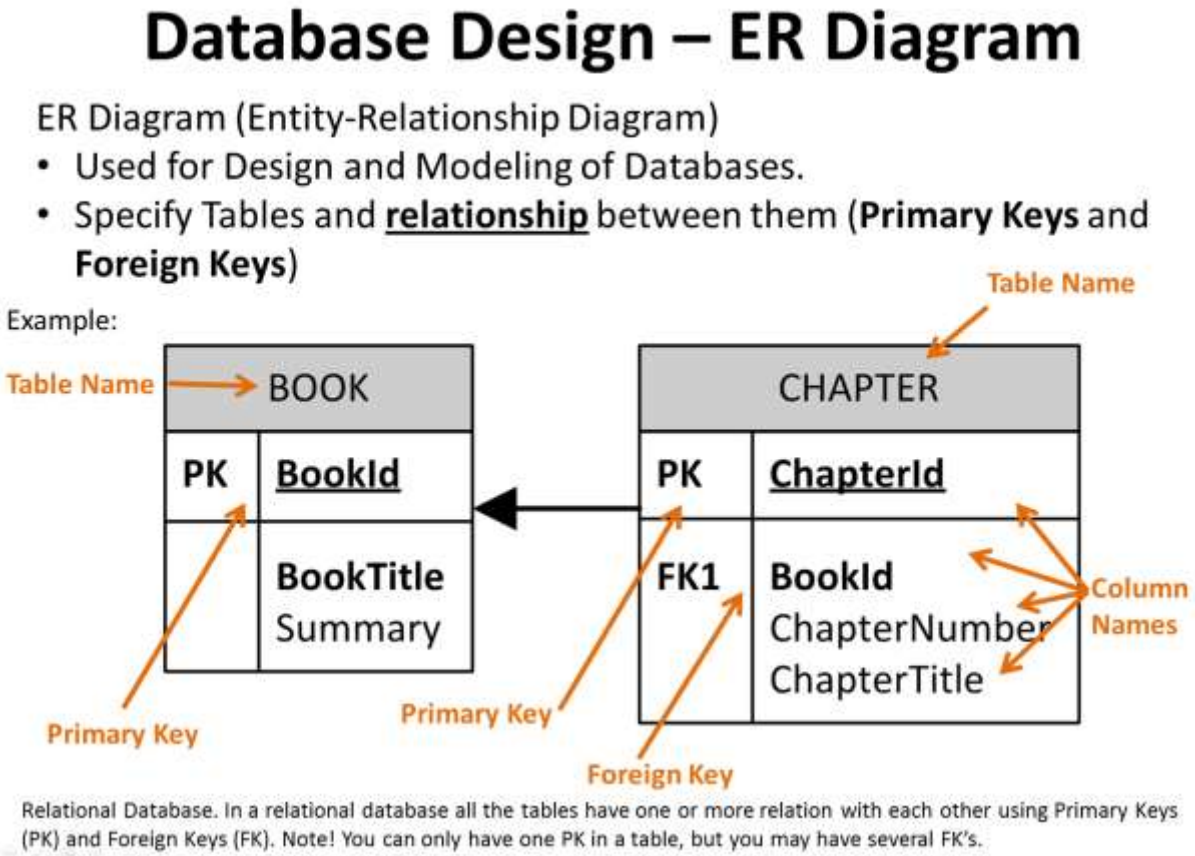
In order to make a new SQL query, select the "New Query" button from the Toolbar.



Here we can write any kind of queries that is supported by the SQL language.

1.3 CREATE TABLE

Before you start implementing your tables in the database, you should always spend some time design your tables properly using a design tool like, e.g., ERwin, Toad Data Modeler, PowerDesigner, Visio, etc. This is called Database Modeling.



The **CREATE TABLE** statement is used to create a table in a database.

Syntax:

```
CREATE TABLE table_name
(
column_name1 data_type,
column_name2 data_type,
column_name3 data_type,
....
)
```

The data type specifies what type of data the column can hold.

You have special data types for numbers, text dates, etc.
Examples:

- Numbers: **int**, **float**
- Text/Stings: **varchar(X)** – where X is the length of the string
- Dates: **datetime**
- etc.

Example:

We want to create a table called "CUSTOMER" which has the following columns and data types:

Column Name	Data Type	Allow Nulls
CustomerId	int	☐
CustomerNumber	int	☐
LastName	varchar(50)	☐
FirstName	varchar(50)	☐
AreaCode	int	☑
Address	varchar(50)	☑
Phone	varchar(20)	☑
		☐

```
CREATE TABLE CUSTOMER
(
CustomerId int IDENTITY(1,1) PRIMARY KEY,
CustomerNumber int NOT NULL UNIQUE,
LastName varchar(50) NOT NULL,
FirstName varchar(50) NOT NULL,
AreaCode int NULL,
Address varchar(50) NULL,
Phone varchar(50) NULL,
)
```

Best practice:

When creating tables you should consider following these guidelines:

- Tables: Use upper case and singular form in table names – not plural, e.g.,

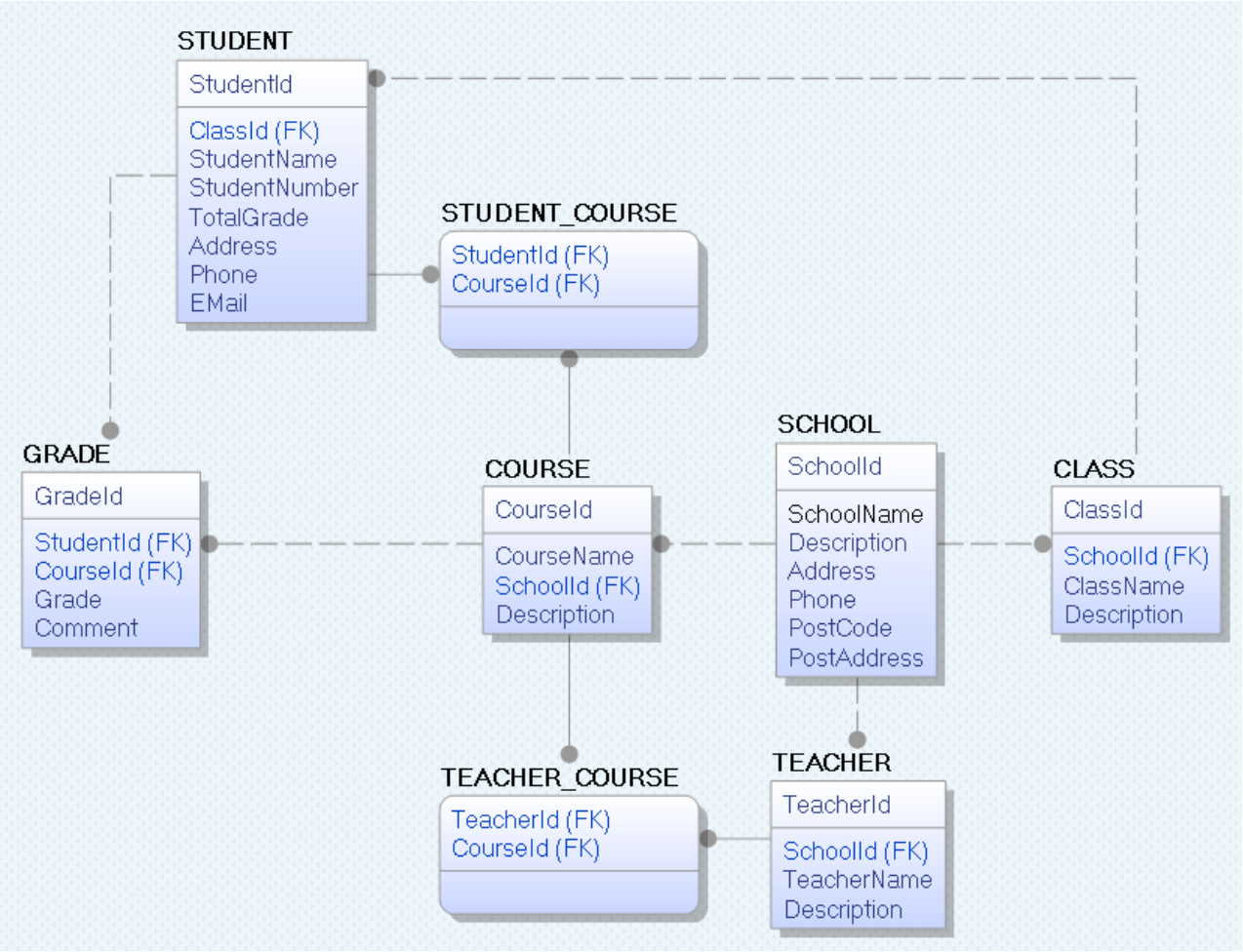
"STUDENT" (not students)

- Columns: Use Pascal notation, e.g., "StudentId"
- Primary Key:
  - If the table name is "COURSE", name the Primary Key column "CourseId", etc.
  - "Always" use Integer and Identity(1,1) for Primary Keys. Use UNIQUE constraint for other columns that needs to be unique, e.g. RoomNumber
- Specify Required Columns (NOT NULL) –
  i.e., which columns that need to have data or not
- Standardize on few/these Data Types: int, float, varchar(x), datetime, bit
- Use English for table and column names
- Avoid abbreviations! (Use RoomNumber – not RoomNo, RoomNr, ...)

2.1 Database Modelling

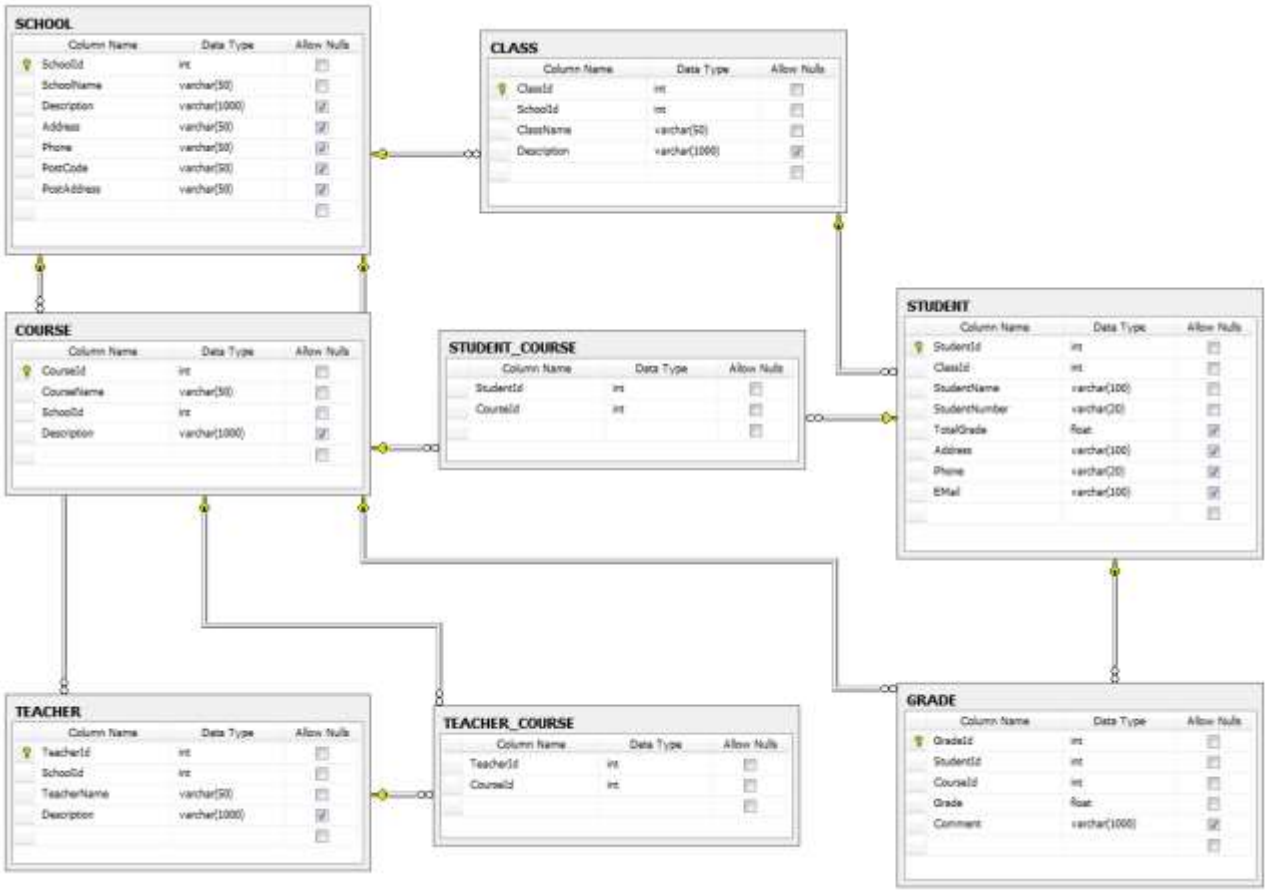
As mention in the beginning of the lecture, you should always start with database modelling before you start implementing the tables in a database system.

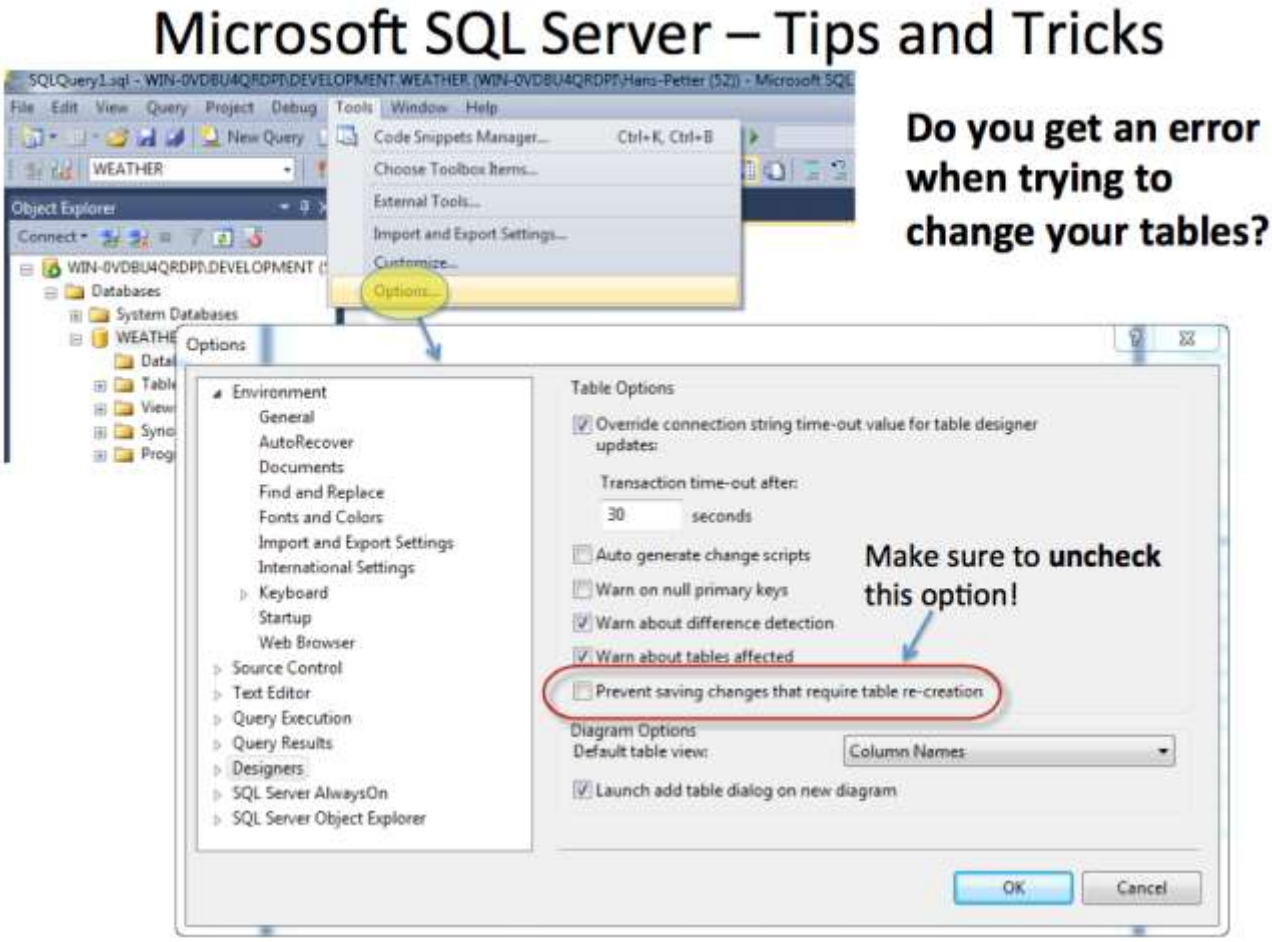
Below we see a database model in created with ERwin.



With this tool we can transfer the database model as tables into different database systems, such as e.g., SQL Server. CA ERwin Data Modeler Community Edition is free with a 25 objects
limit. It has support for Oracle, SQL Server, MySQL, ODBC and Sybase.

Below we see the same tables inside the design tool in SQL Server.


SCHOOL
Column Name Data Type Allow Nulls
SchoolId int
SchoolName varchar(50)
Description varchar(1000)
Address varchar(50)
Phone varchar(50)
PostCode varchar(50)
PostAddress varchar(50)
CLASS
Column Name Data Type Allow Nulls
ClassId int
SchoolId int
ClassName varchar(50)
Description varchar(1000)
COURSE
Column Name Data Type Allow Nulls
CourseId int
CourseName varchar(50)
SchoolId int
Description varchar(1000)
STUDENT_COURSE
Column Name Data Type Allow Nulls
StudentId int
CourseId int
STUDENT
Column Name Data Type Allow Nulls
StudentId int
ClassId int
StudentName varchar(100)
StudentNumber varchar(20)
TotalGrade float
Address varchar(100)
Phone varchar(20)
EMail varchar(100)
TEACHER
Column Name Data Type Allow Nulls
TeacherId int
SchoolId int
TeacherName varchar(50)
Description varchar(1000)
TEACHER_COURSE
Column Name Data Type Allow Nulls
TeacherId int
CourseId int
GRADE
Column Name Data Type Allow Nulls
GradeId int
StudentId int
CourseId int
Grade float
Comment varchar(1000)

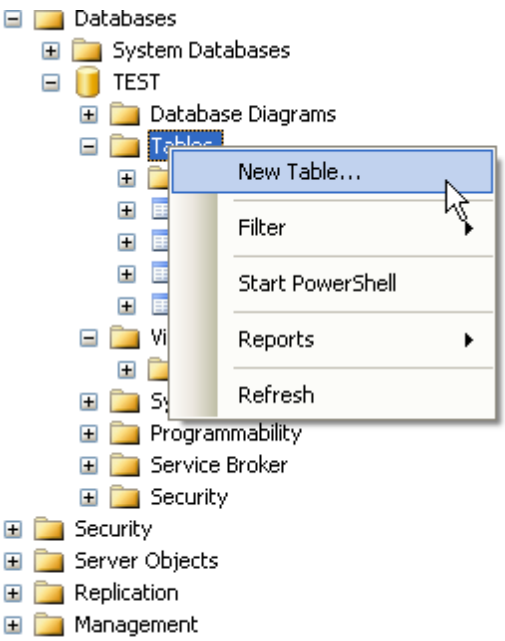


2.2 Create Tables using the Designer Tools

Even if you can do "everything" using the SQL language, it is sometimes easier to do it in the designer tools in the Management Studio in SQL Server.

Instead of creating a script you may as well easily use the designer for creating tables.

Step1: Select "New Table …":



Step2: Next, the table designer pops up where you can add columns, data types, etc.

Column Name	Data Type	Allow Nulls
CustomerId	int	☐
CustomerNumber	int	☐
LastName	varchar(50)	☐
FirstName	varchar(50)	☐
AreaCode	int	☑
Address	varchar(50)	☑
Phone	varchar(20)	☑
		☐

In this designer we may also specify Column Names, Data Types, etc.

Step 3: Save the table by clicking the Save button.

2.3 SQL Constraints

Constraints are used to limit the type of data that can go into a table.

Constraints can be specified when a table is created (with the CREATE TABLE statement) or after the table is created (with the ALTER TABLE statement).

Here are the most important constraints:

- PRIMARY KEY
- NOT NULL
- UNIQUE
- FOREIGN KEY
- CHECK
- DEFAULT
- IDENTITY

In the sections below we will explain some of these in detail.

2.3.1 PRIMARY KEY

The PRIMARY KEY constraint uniquely identifies each record in a database table.

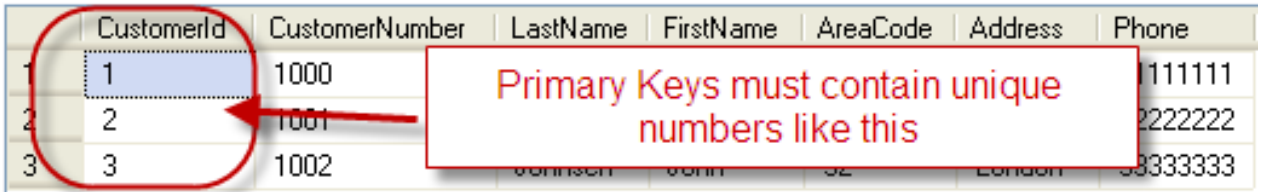
Primary keys must contain unique values. It is normal to just use running numbers, like 1, 2, 3, 4, 5, … as values in Primary Key column. It is a good idea to let the system handle this for you by specifying that the Primary Key should be set to **identity(1,1)**. IDENTITY(1,1) means the first value will be 1 and then it will increment by 1.

Each table should have a primary key, and each table can have only ONE primary key.

If we take a closer look at the CUSTOMER table created earlier:

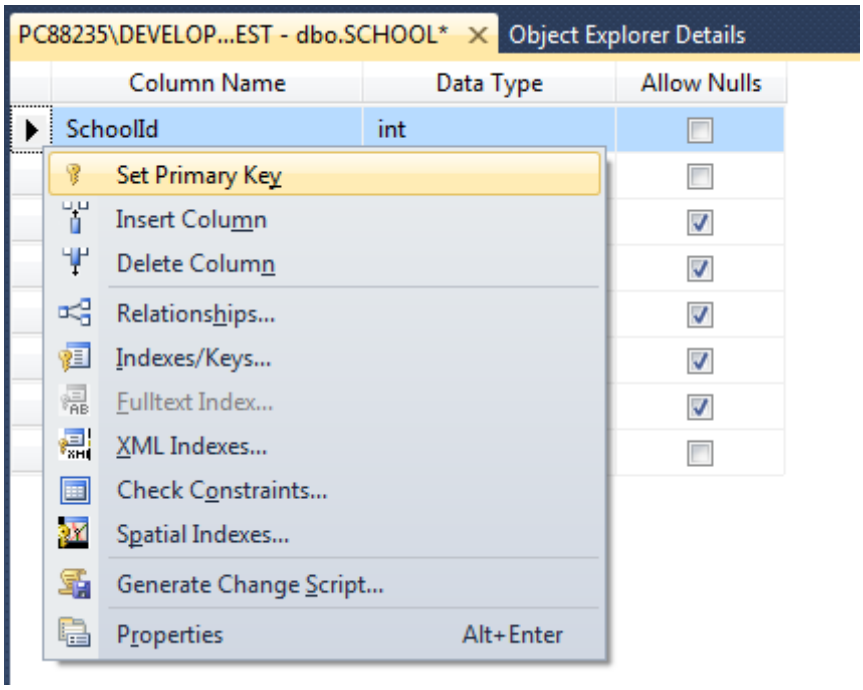
```
CREATE TABLE [CUSTOMER]
(
CustomerId int IDENTITY(1,1) PRIMARY KEY,
CustomerNumber int NOT NULL UNIQUE,
LastName varchar(50) NOT NULL,
FirstName varchar(50) NOT NULL,
AreaCode int NULL,
Address varchar(50) NULL,
Phone varchar(50) NULL,
)
```

As you see we use the "Primary Key" keyword to specify that a column should be the ***Primary Key***.



Setting Primary Keys in the Designer Tools:

If you use the Designer tools in SQL Server, you can easily set the primary Key in a table just by right-click and select "Set primary Key".

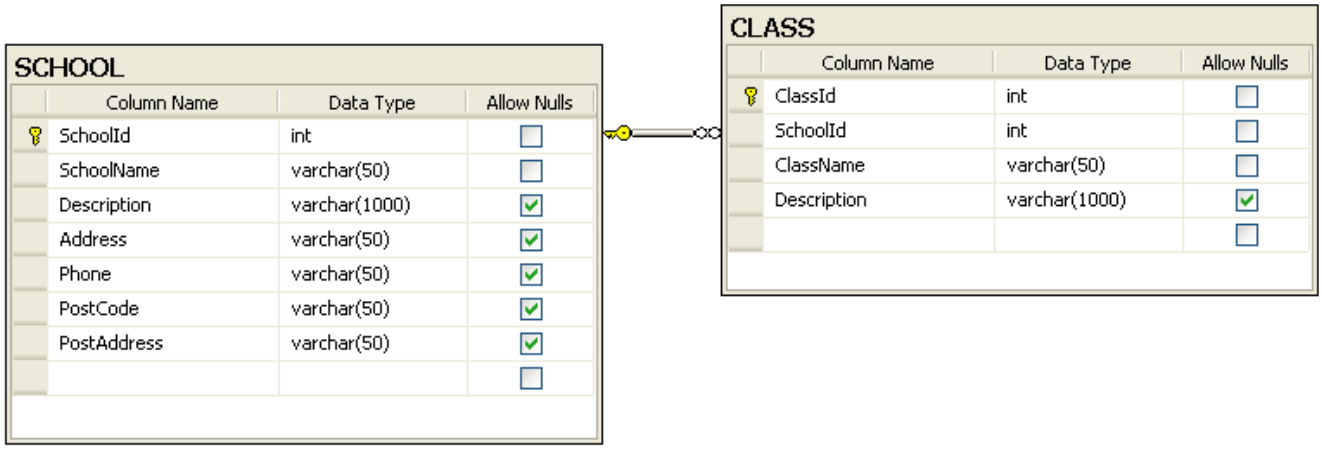


The primary Key column will then have a small key in front to illustrate that this column is a Primary Key.

2.3.2 FOREIGN KEY

A FOREIGN KEY in one table points to a PRIMARY KEY in another table.

Example:



We will create a CREATE TABLE script for these tables:

SCHOOL:

```
CREATE TABLE SCHOOL
(
SchoolId int IDENTITY(1,1) PRIMARY KEY,
SchoolName varchar(50) NOT NULL UNIQUE,
Description varchar(1000) NULL,
Address varchar(50) NULL,
Phone varchar(50) NULL,
PostCode varchar(50) NULL,
PostAddress varchar(50) NULL,
)
```

CLASS:

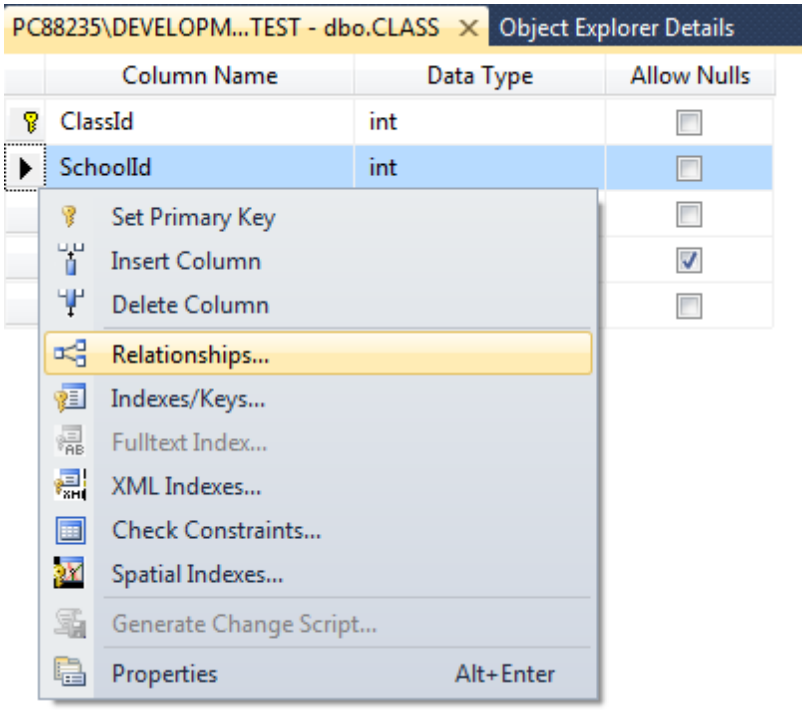
```
CREATE TABLE CLASS
(
ClassId int IDENTITY(1,1) PRIMARY KEY,
SchoolId int NOT NULL FOREIGN KEY REFERENCES SCHOOL(SchoolId),
ClassName varchar(50) NOT NULL UNIQUE,
Description varchar(1000) NULL,
)
```

The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables.

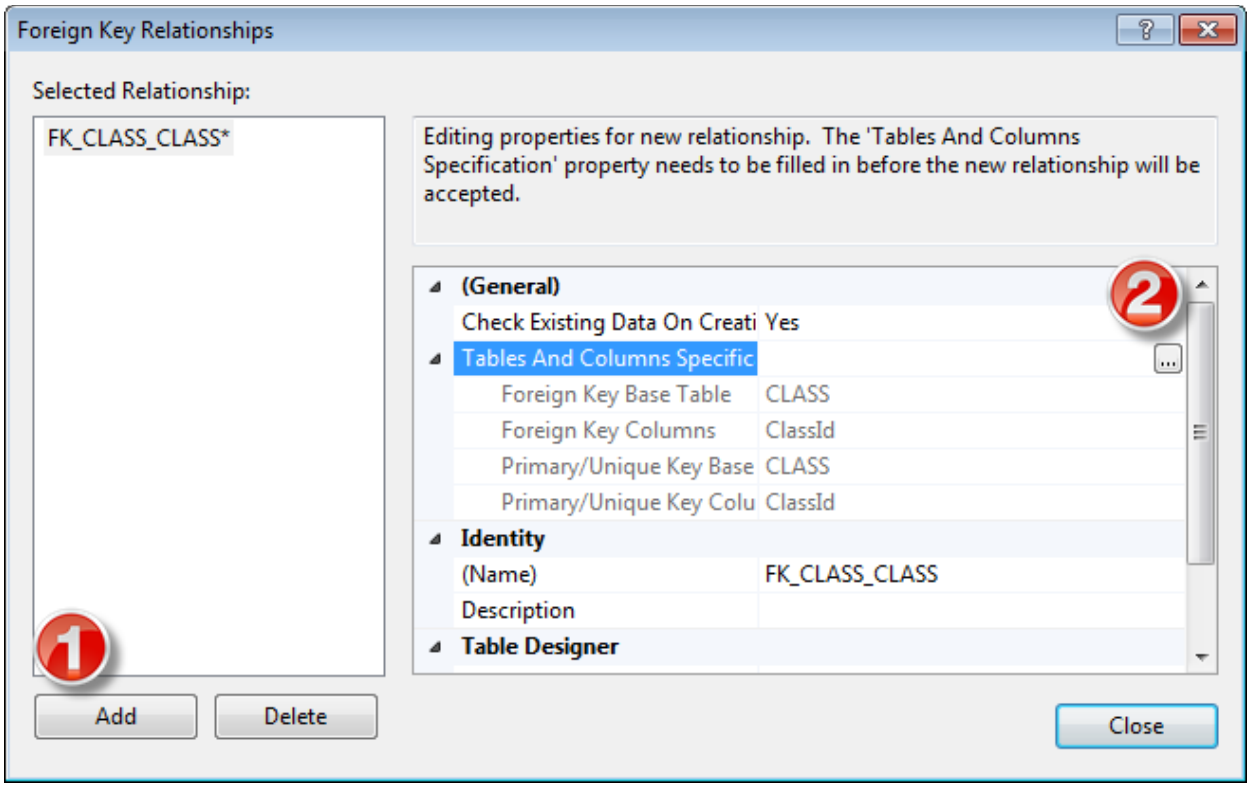
The FOREIGN KEY constraint also prevents that invalid data from being inserted into the foreign key column, because it has to be one of the values contained in the table it points to.

Setting Foreign Keys in the Designer Tools:

If you want to use the designer, right-click on the column that you want to be the Foreign Key and select "**Relationships…**":

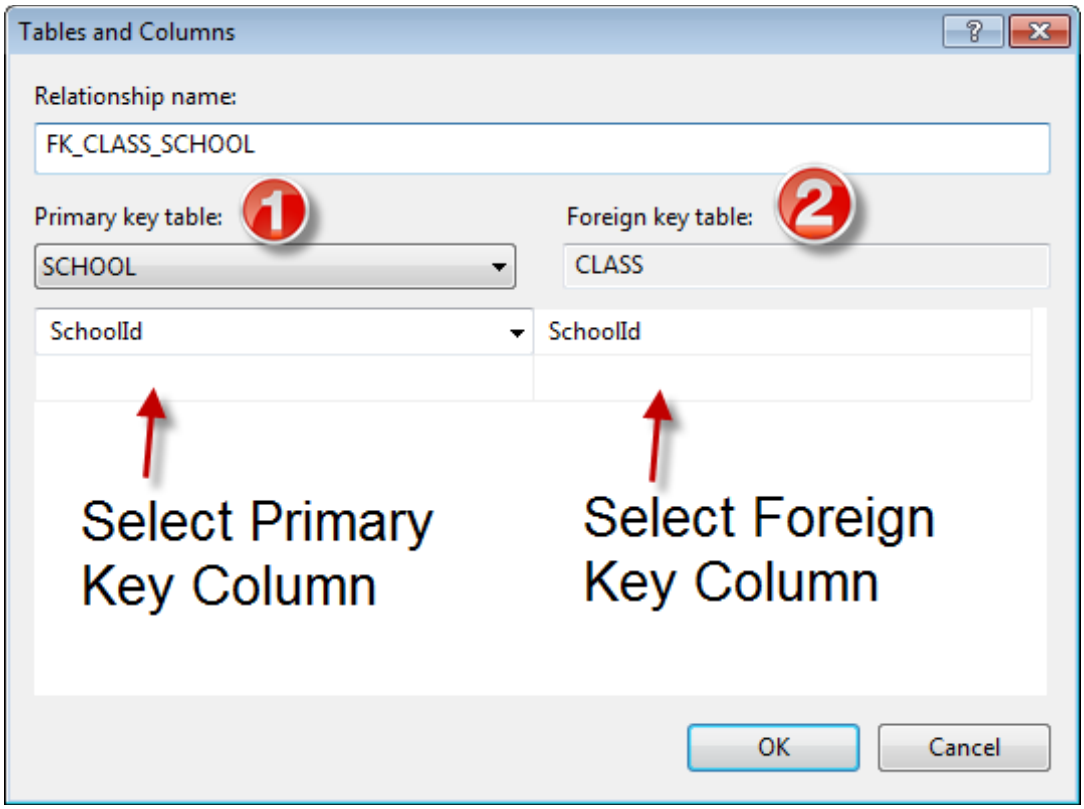


The following window pops up (Foreign Key Relationships):



Click on the "Add" button and then click on the small "…" button. Then the following window

pops up (Tables and Columns):



Here you specify the primary Key Column in the Primary Key table and the Foreign Key Column in the Foreign Key table.

2.3.3 NOT NULL / Required Columns

The NOT NULL constraint enforces a column to NOT accept NULL values.

The NOT NULL constraint enforces a field to always contain a value. This means that you cannot insert a new record, or update a record without adding a value to this field.

If we take a closer look at the CUSTOMER table created earlier:

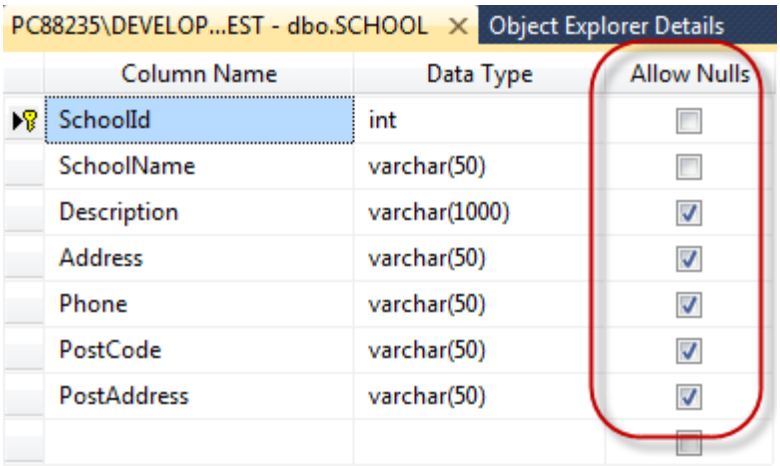
```
CREATE TABLE [CUSTOMER]
(
CustomerId int IDENTITY(1,1) PRIMARY KEY,
CustomerNumber int NOT NULL UNIQUE,
LastName varchar(50) NOT NULL,
FirstName varchar(50) NOT NULL,
AreaCode int NULL,
Address varchar(50) NULL,
Phone varchar(50) NULL,
)
```

We see that "CustomerNumber", "LastName" and "FirstName" is set to "NOT NULL", this means these columns needs to contain data. While "AreaCode", "Address" and "Phone" may be left empty, i.e, they don't need to filled out.

Note! A primary key column cannot contain NULL values.

Setting NULL/NOT NULL in the Designer Tools:

In the Table Designer you can easily set which columns that should allow NULL or not:



2.3.4 UNIQUE

The **UNIQUE** constraint uniquely identifies each record in a database table. The UNIQUE and PRIMARY KEY constraints both provide a guarantee for uniqueness for a column or set of columns.

A PRIMARY KEY constraint automatically has a UNIQUE constraint defined on it.

Note! You can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.

If we take a closer look at the CUSTOMER table created earlier:

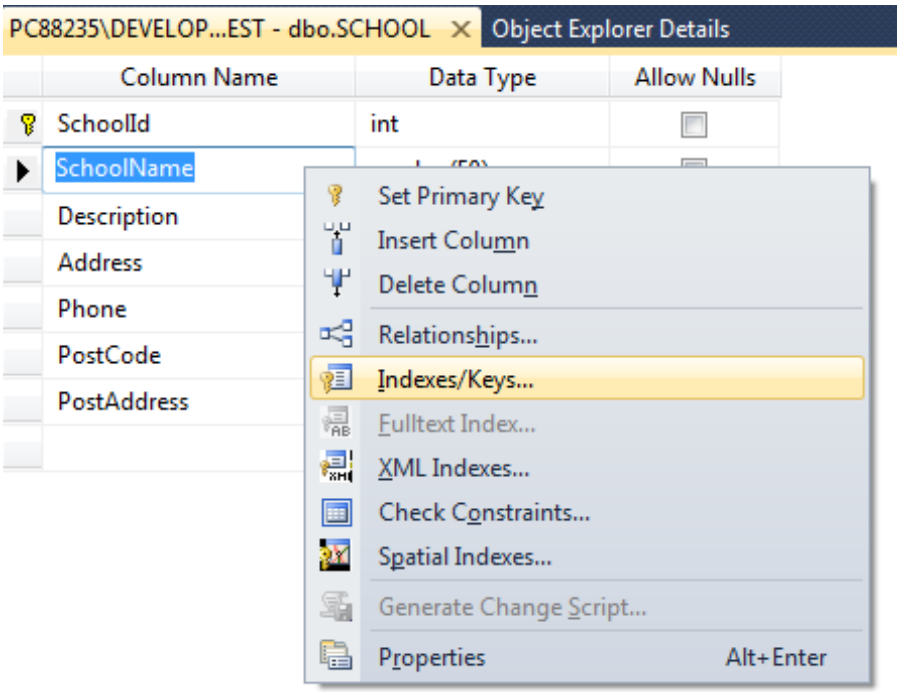
```
CREATE TABLE [CUSTOMER]
(
CustomerId int IDENTITY(1,1) PRIMARY KEY,
CustomerNumber int NOT NULL UNIQUE,
LastName varchar(50) NOT NULL,
FirstName varchar(50) NOT NULL,
AreaCode int NULL,
Address varchar(50) NULL,
Phone varchar(50) NULL,
)
```

We see that the "CustomerNumber" is set to UNIQUE, meaning each customer must have a unique Customer Number. Example:

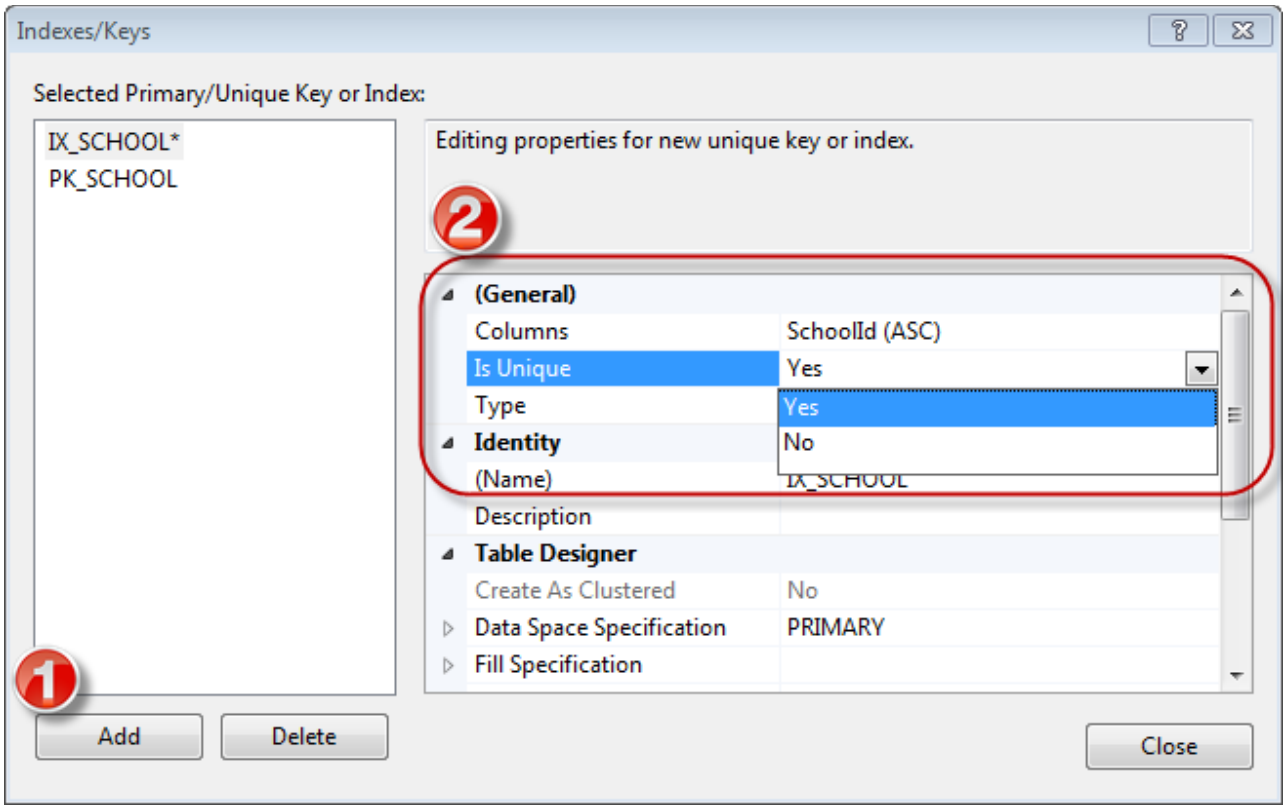
	CustomerId	CustomerNumber	LastName	FirstName	AreaCode	Address	Phone
1	1	1000	Smith	John	12	California	11111111
2	2	1001	Jackson	Smith	45	London	22222222
3	3	1002	Johnsen	John	32	London	33333333

Setting UNIQUE in the Designer Tools:

If you want to use the designer, right-click on the column that you want to be UNIQUE and select "**Indexes/Keys…**":



Then click "Add" and then set the "Is Unique" property to "Yes":



2.3.5 CHECK

The CHECK constraint is used to limit the value range that can be placed in a column.

If you define a CHECK constraint on a single column it allows only certain values for this column.

If you define a CHECK constraint on a table, it can limit the values in certain columns based on values in other columns in the row.

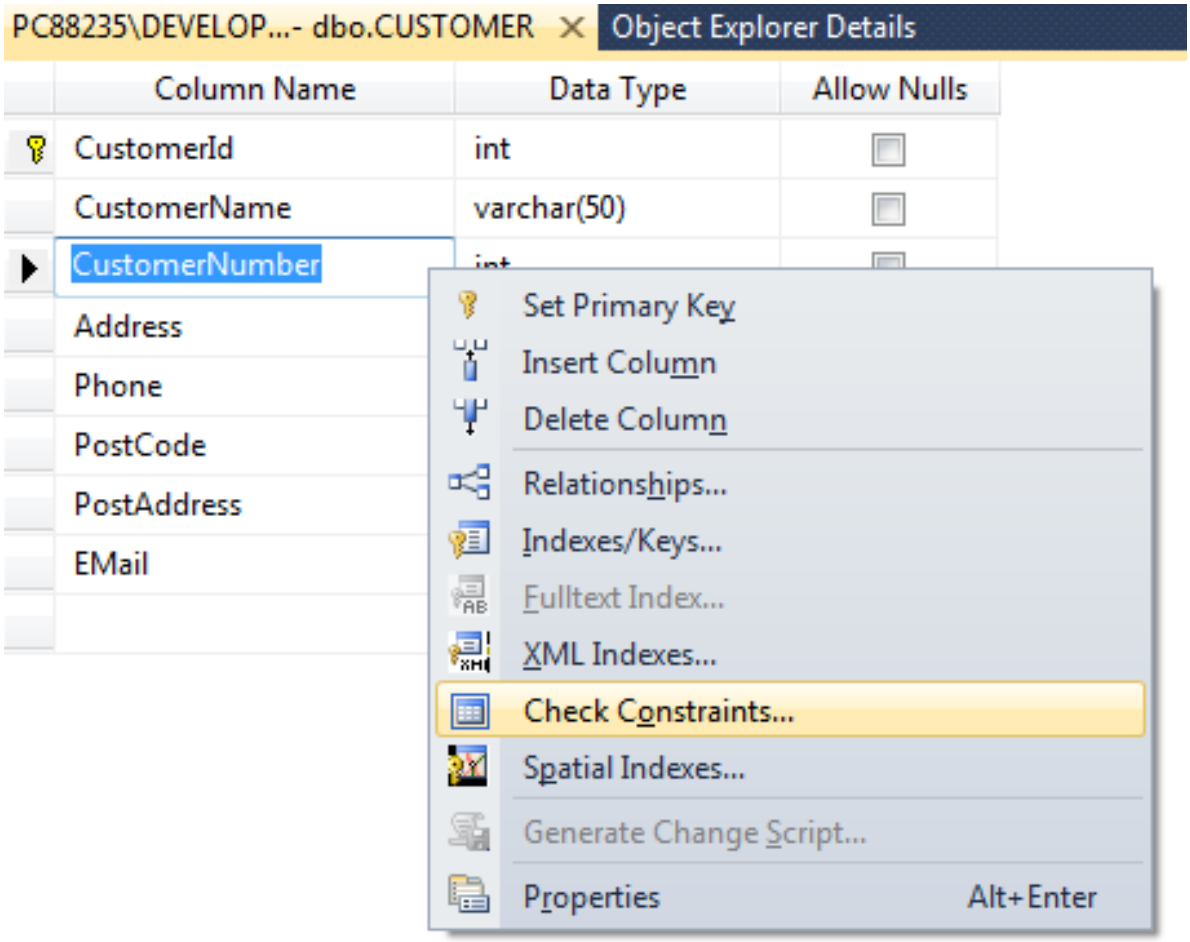
Example:

```
CREATE TABLE [CUSTOMER]
(
CustomerId int IDENTITY(1,1) PRIMARY KEY,
CustomerNumber int NOT NULL UNIQUE CHECK(CustomerNumber>0),
LastName varchar(50) NOT NULL,
FirstName varchar(50) NOT NULL,
AreaCode int NULL,
Address varchar(50) NULL,
Phone varchar(50) NULL,
)
```

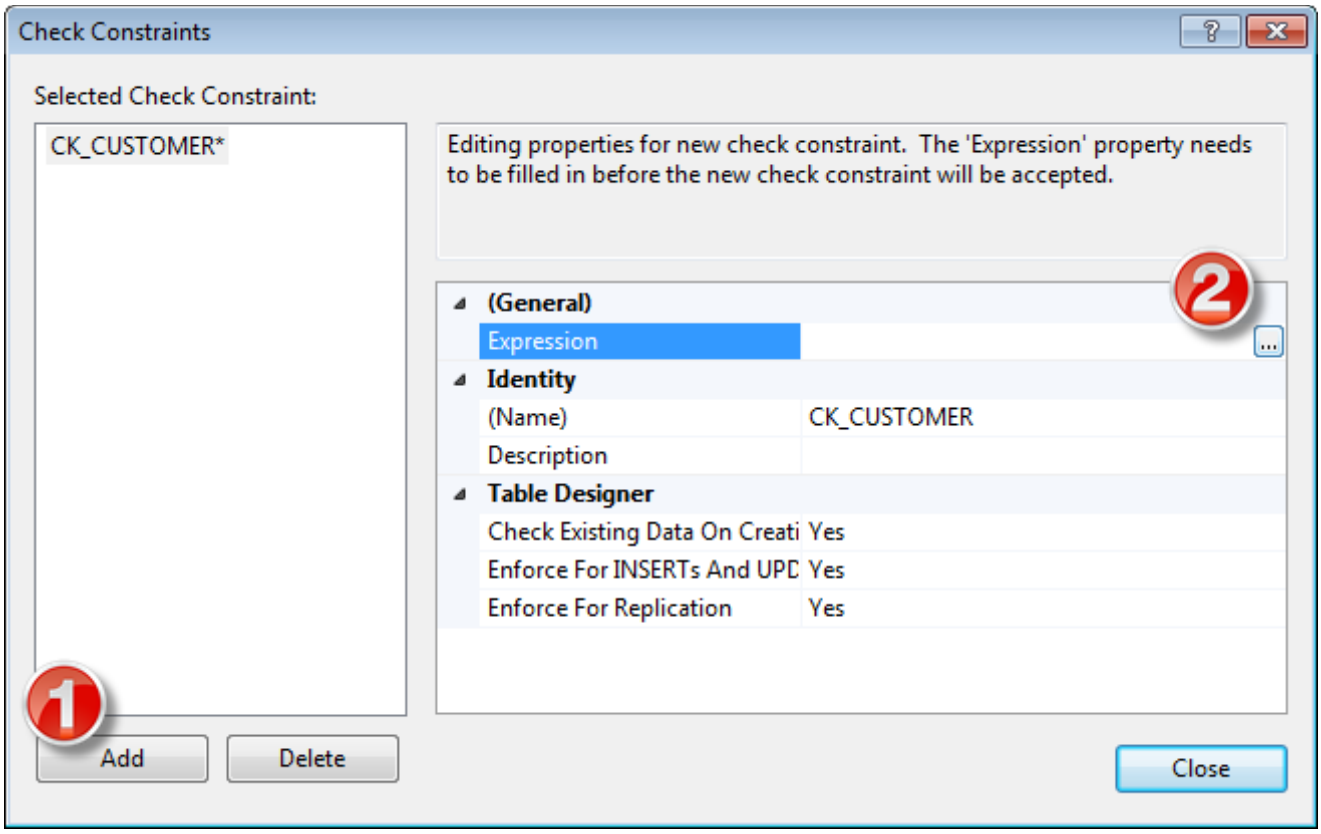
In this case, when we try to insert a Customer Number less than zero we will get an error message.

Setting CHECK constraints in the Designer Tools:

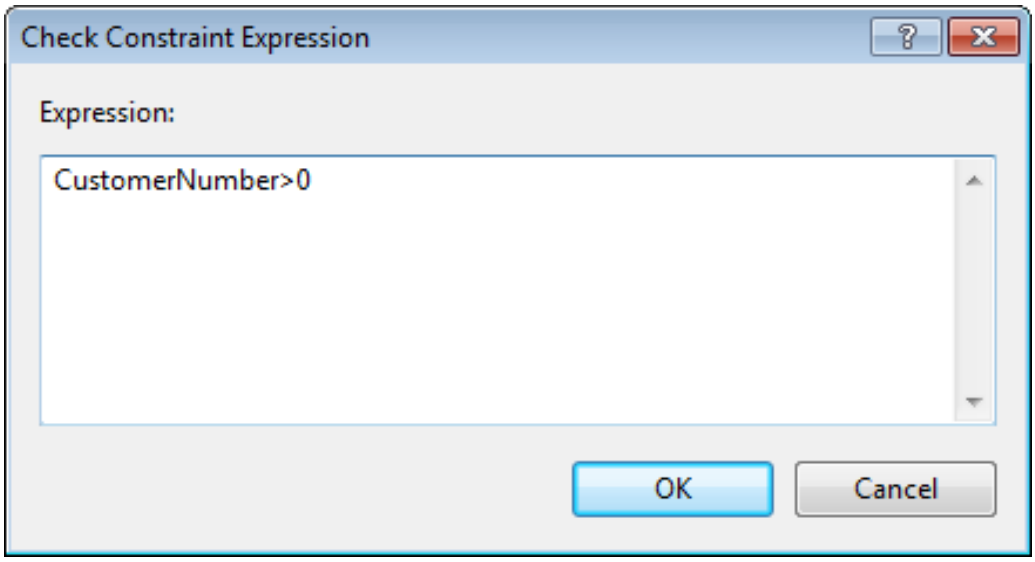
If you want to use the designer, right-click on the column where you want to set the constraints and select "**Check Constraints…**":



Then click "Add" and then click "…" in order to open the Expression window:



In the Expression window you can type in the expression you want to use:



2.3.6 DEFAULT

The DEFAULT constraint is used to insert a default value into a column.

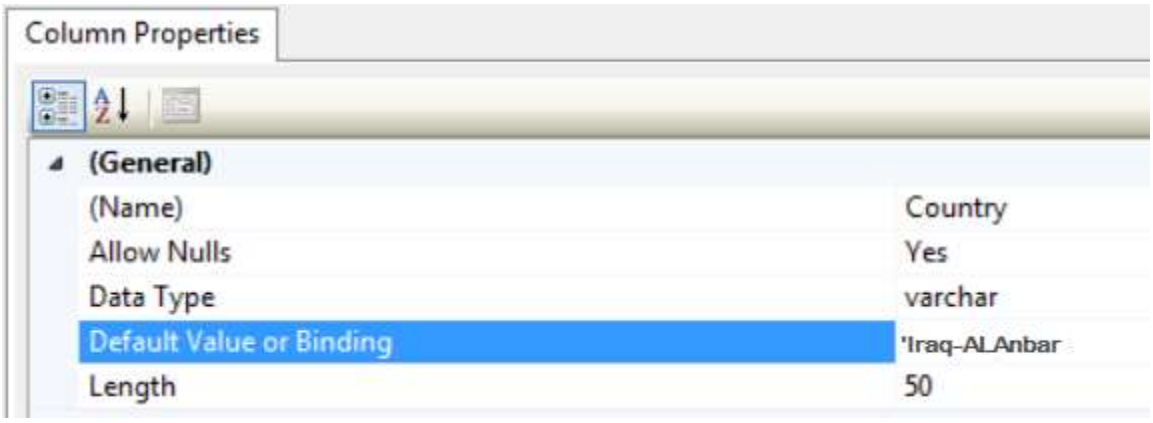
The default value will be added to all new records, if no other value is specified.

Example:

```
CREATE TABLE [CUSTOMER]
(
CustomerId int IDENTITY(1,1) PRIMARY KEY,
CustomerNumber int NOT NULL UNIQUE,
LastName varchar(50) NOT NULL,
FirstName varchar(50) NOT NULL,
Country varchar(20) DEFAULT 'Iraq – AL Anbar',
AreaCode int NULL,
Address varchar(50) NULL,
Phone varchar(50) NULL,
)
```

Setting DEFAULT values in the Designer Tools:

Select the column and go into the "Column Properties":



2.3.7 AUTO INCREMENT or IDENTITY

Very often we would like the value of the primary key field to be created automatically every time a new record is inserted.

Example:

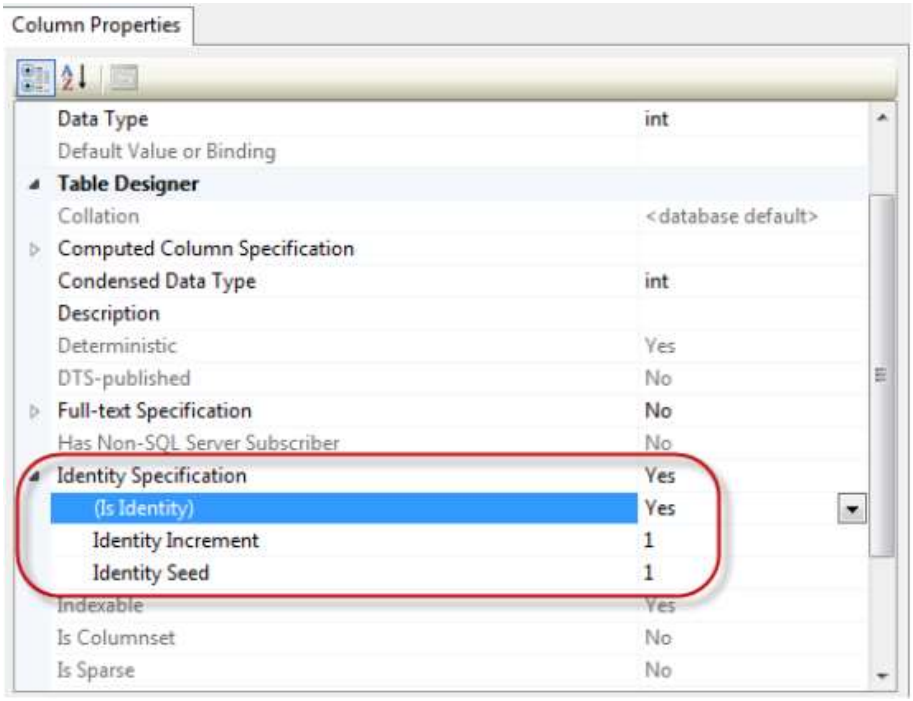
```
CREATE TABLE CUSTOMER
(
CustomerId int IDENTITY(1,1) PRIMARY KEY,
CustomerNumber int NOT NULL UNIQUE,
LastName varchar(50) NOT NULL,
FirstName varchar(50) NOT NULL,
AreaCode int NULL,
Address varchar(50) NULL,
Phone varchar(50) NULL,
)
```

As shown below, we use the IDENTITY() for this. IDENTITY(1,1) means the first value will be 1 and then it will increment by 1.

Setting identity (1,1) in the Designer Tools:

We can use the designer tools to specify that a Primary Key should be an identity column that is automatically generated by the system when we insert data in to the table.

Click on the column in the designer and go into the Column Properties window:



2.3.8 ALTER TABLE

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.

To add a column in a table, use the following syntax:

```
ALTER TABLE table_name
ADD column_name datatype
```

To delete a column in a table, use the following syntax (notice that some database systems don't allow deleting a column):

```
ALTER TABLE table_name
DROP COLUMN column_name
```

To change the data type of a column in a table, use the following syntax:

```
ALTER TABLE table_name
ALTER COLUMN column_name datatype
```

Summary

In this lecture,

The student's will be able to learn how to make the definition of database terminology from tables, queries, reports and forms. also will be able to create and manage databases, use data stored in the databases. The most important point is to understanding the data models of database, three levels of the database, the physical data structure, three factors of the data model with entire example illustrates the work of this function.