Data Definition Language (DDL),
View
And Coding Conventions

Author: TrungDVQ (Fsoft-Academy)

Lesson Objectives

01 Introduction SQL
- ✓ What is SQL
- ✓ SQL process
- ✓ SQL components
- ✓ SQL data type
- ✓ SQL Operations

02 Data Definition Language
- ✓ Database
- ✓ Table
- ✓ Views

03 Coding Convention
- ✓ Naming convention
- ✓ Styles

# How to use MS SQL

How to use MS SQL

1. **Server type:** This is an option to select one out of four available MS SQL services option. We will be working on 'Database Engine' for creating and working with Database. Other Server type includes Analysis, Reporting & Integration Services.

2. **Server name:** This is Server's name where MS SQL Server is installed and need to establish the connection with that server. Generally, we use the server name as **"Machine name\Instance."** Here Instance is the name given to SQL Server instance while SQL server installation.

3. **Authentication:** This is defaulted to "Windows Authentication" if we use "Windows Authentication" during SQL Server Installation. Else, if we select 'Mixed Mode (Windows Authentication & Windows Authentication)' then Authentication will be defaulted to "SQL Server Installation."

4. **User name\Password:** If Authentication is selected other than "Windows Authentication" like "SQL server Installation" then these two fields will be required.

How to use MS SQL

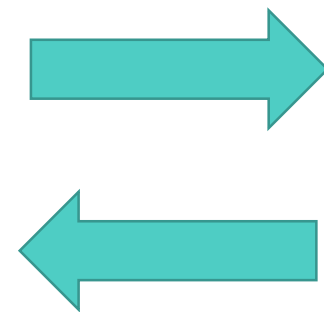**Databases in use**
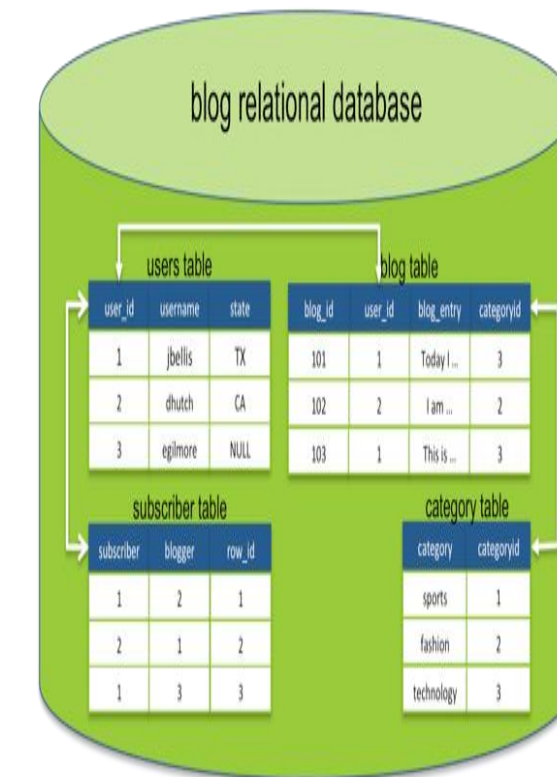
Execute button

Object Explorer

Query Editor

1
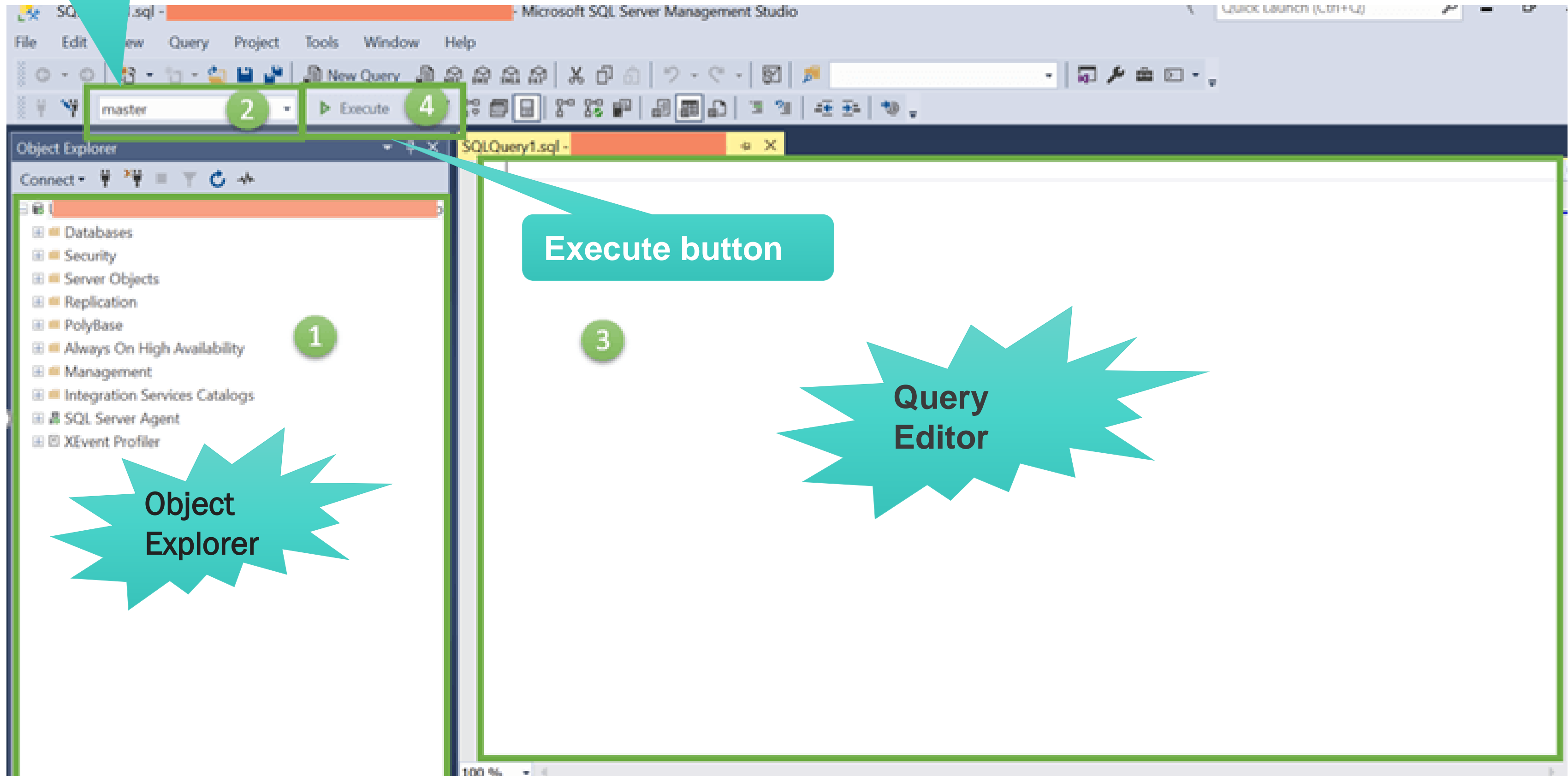
Introduction SQL

What is SQL?

Structured

Query

Language
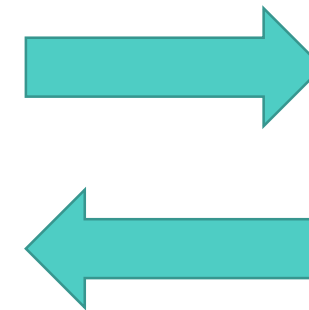
➢ **Which is a computer language for:**
- ✓ storing,
- ✓ manipulating
- ✓ retrieving data stored in relational database.

➢ SQL is the standard language for Relation Database System, like **MySQL, MS Access, Oracle, Sybase, Informix, PostgreSQL** and **SQL Server** use SQL as standard database language.

➢ SQL is an ANSI (American National Standards Institute) standard.

SQL Process
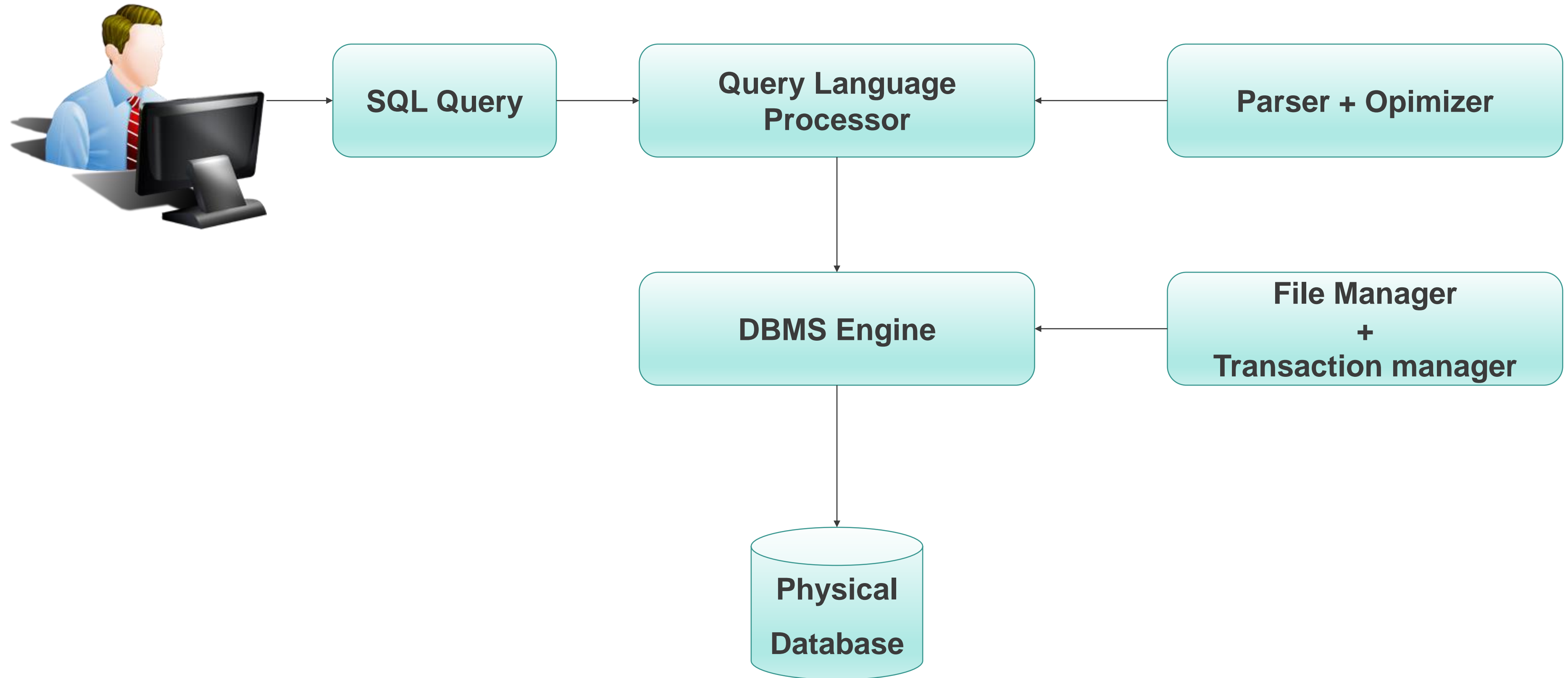
The Components of SQL

SQL consists of **three components:**

➢ Data Definition Language (DDL)

➢ Data Manipulation Language (DML)

➢ Data Control Language (DCL)
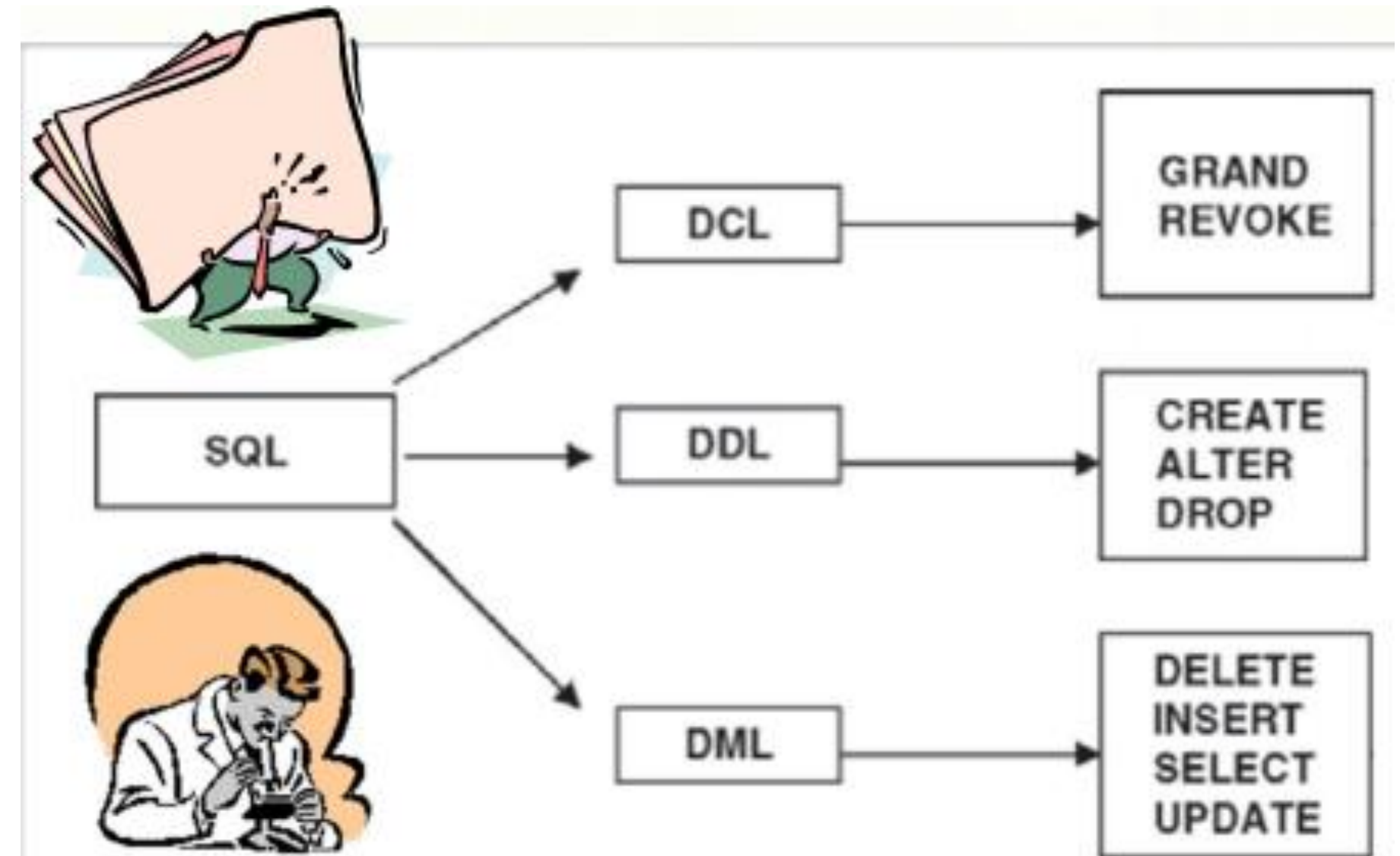
Ms SQL Server Data Types

Student:

- ✓ Name
- ✓ Birthday
- ✓ Gender
- ✓ Address
- ✓ Marks…

Storage

What type of data each of field ???..............

Ms SQL Server Data Types

SQL Server supports below data types. NULL is default value for most data type:

- ✓Exact Numerics
- ✓Approximate Numerics
- ✓Date and Time
- ✓Character Strings
- ✓Unicode Character Strings
- ✓Binary Strings
- ✓Other Data Types

Numberics

Date & Time

String

...
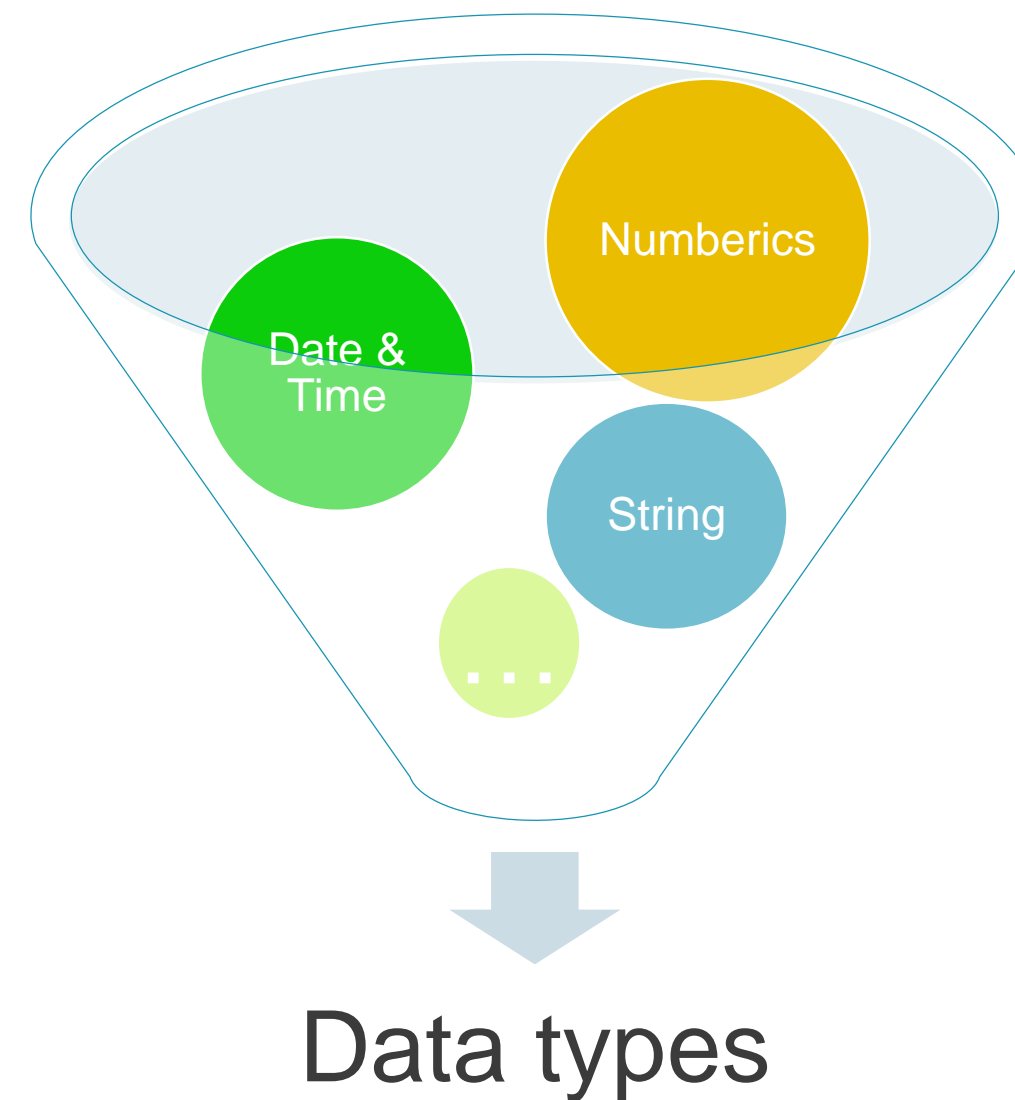
Data types

Exact Numbers

Interger-based data type

Data type	Size	Range of values
Bigint	8 Bytes	-2^{63} to $2^{63}-1$
Int	4 Bytes	-2^{31} to $2^{31}-1$
Smallint	2 Bytes	-2^{15} to $2^{15} - 1$
Tinyint	1 Byte	0 to 255
Bit	1 Bit	0 to 1

Exact Numbers

Exact decimal-based data type

Data type	Size	Range of values
Decimal(p,s)	5 - 17 Bytes (depending on precision)	- Varies based on precision setting. - Maximum values are -10^38 +1 through 10^38 -1
(p is the maximum number of all digits (both sides of the decimal point), s is the maximum number of digits after the decimal point)		
Numeric(p,s)	…	Identical to **Decimal type**
Smallmoney	4 Bytes	- 214,748.3648 to 214,748.3647
Money	8 Bytes	- 922,337,203,685,477.5808 To 922,337,203,685,477.5807

Data type	Size	Range of values
Float	8 Bytes	- 1.79E+308 to 1.79E+308
Float(n)	*Depends on the value of n* If 1 ≤ n ≤ 24: 4 Bytes (**Precision:** 7 digits) If 25 ≤ n ≤ 53: 8 Bytes (**Precision: 15** digits)	4 Bytes: - 3.40E + 38 to 3.40E + 38 8 Bytes: - 1.79E+308 to 1.79E+308
Real	…	- 3.40E + 38 to 3.40E + 38

Note: *SQL Server treats n as one of two possible values.*
- ✓ *If **1**<=n<=**24**, n is treated as **24**.*
- ✓ *f **25**<=n<=**53**, n is treated as **53**.*

Date and Time

Data Type	Description	Example
Date	Stores dates between January 1, 0001, and December 31, 9999	2008-01-15
Datetime	Stores dates and times between January 1, 1753, and December 31, 9999, with an accuracy of 3.33 milliseconds	2008-01-15 09:42:16.142
Datetime2	Stores date and times between January 1, 0001, and December 31, 9999, with an accuracy of 100 nanoseconds	2008-01-15 09:42:16.1420221
Datetimeoffset	Similar to the datetime2 data type, but also expects an offset designation of –14:00 to +14:00	2008-01-15 09:42:16.1420221 +05:00
Smalldatetime	Stores dates and times between January 1, 1900, and June 6, 2079, with an accuracy of 1 minute	2008-01-15 09:42:00
Time	Stores times with an accuracy of 100 nanoseconds	09:42:16.1420221

Character Strings

Non-Unicode string data types:

Data type	Description
Char(n)	- Fixed-length - Maximum length of 8,000 characters (1 ≤ n ≤ 8000)
Varchar(n)	- Variable-length - Maximum of 8,000 characters (1 ≤ n ≤ 8000)
Varchar(max)	- Variable-length - Maximum length of 2,147,483,647 characters
Text	- Variable-length - Maximum length of 2,147,483,647 characters - Use varchar(max) instead

Unicode Character Strings

Unicode string data types are **"double width":**

Data type	Description
Nchar(n)	- Fixed-length - Maximum specified length is 4,000 characters (1≤ n ≤ 4000)
Nvarchar(n)	- Variable-length - Maximum specified length is 4,000 characters (1≤ n ≤ 4000)
Nvarchar(max)	- Variable-length - Maximum length of 1,073,741,823 characters
Ntext	- Variable-length - Maximum length of 1,073,741,823 characters

Binary Strings

Data type	Description
Binary	- Fixed-length binary data - Maximum length of 8,000 bytes
Varbinary	- Variable length binary data - Maximum length of 8,000 bytes.
Image	- Variable length binary data - Maximum length of 2,147,483,647 bytes.

Other Data Types

Data Type	Description
Timestamp	Stores a database-wide unique number that gets updated every time a row gets updated
Hierarchyid	Special data type that maintains hierarchy positioning information
Uniqueidentifier	Stores a database-wide unique number that gets updated every time a row gets updated
Sql_variant	Stores values of various SQL Server-supported data types, except text, ntext, and timestamp
Xml	Stores XML data. You can store xml instances in a column or a variable (SQL Server 2005 only).
Table	Stores a result set for later processing

SQL Operators

What is an Operator in SQL?

➢ An operator is a reserved word or a character used primarily in an SQL statement's WHERE clause to perform operation(s), such as comparisons and arithmetic operations.

➢ Operators are used to specify conditions in an SQL statement and to serve as conjunctions for multiple conditions in a statement. Some types of most operators:

1   Arithmetic operators

2   Comparison operators

3   Logical operators.

SQL Arithmetic Operators

Here is a list of the Arithmetic operators available in SQL

Operator	Description	Example
+	Addition	a + b → 30
-	Subtraction	a - b → -10
*	Multiplication	a * b → 200
/	Division	b / a → 2
%	Modulus	b % a → 0

*(Assume variable **a** holds **10** and variable **b** holds **20**)*

SQL Comparison Operators

Here is a list of all the Comparison operators available in SQL

Operator	Description	Operator	Description
=	equal to	>=	greater than or equal to
!=, <>	not equal to	<=	less than or equal to
<	less than	!<	not less than
>	greater than	!>	not greater than

❑ Example

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

CUSTOMERS TABLE

*SQL: SELECT * FROM CUSTOMERS WHERE SALARY > 5000;*

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

SQL Logical Operators

Operator	Description
ALL	• Used to compare a value to all values in another value set.
AND	• Used when both conditions are included
ANY	• Used to compare a value to any applicable value in the list according to the condition
BETWEEN	• Used to limit the values in a range e.g.
EXISTS	• Used to search for the presence of a row in a specified table that meets certain criteria

SQL Logical Operators

| Operator	Description
IN	• Included in the list e.g.
LIKE	• Equal to some character (use quotes)
NOT	• Opposite of the logical value
OR	• Used when either of the condition is true
IS NULL	• This checks if the field has a null
UNIQUE	• Searches every row of a specified table for uniqueness

2

Data Definition Language (DDL)

Introduction to DDL Statements

DDL stands for **D**ata **D**efinition **L**anguage

Define data structures in SQL Server as creating, altering, and dropping
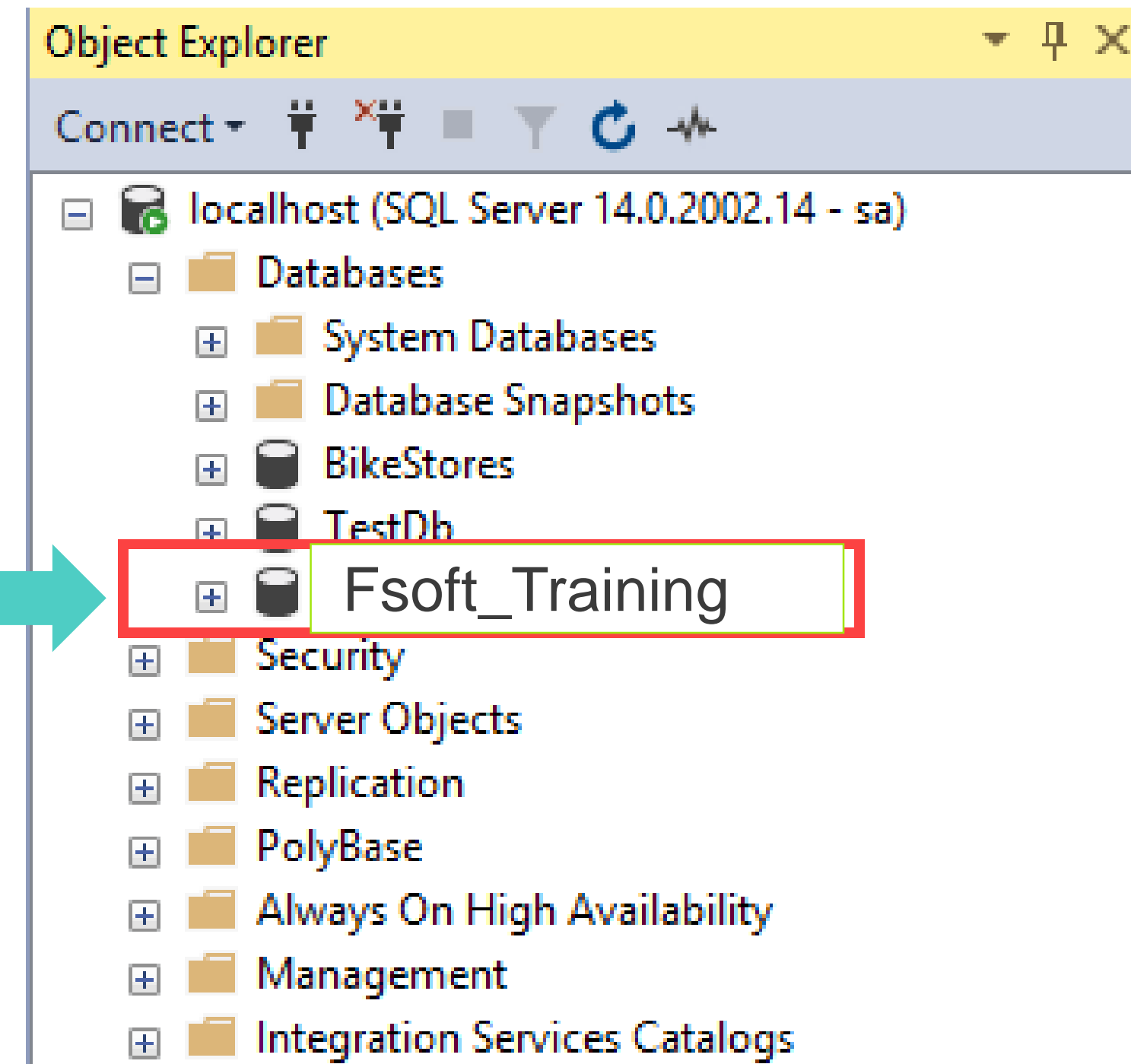
tables and establishing constraints…

Database

DATABASE

CREATE

ALTER

DROP
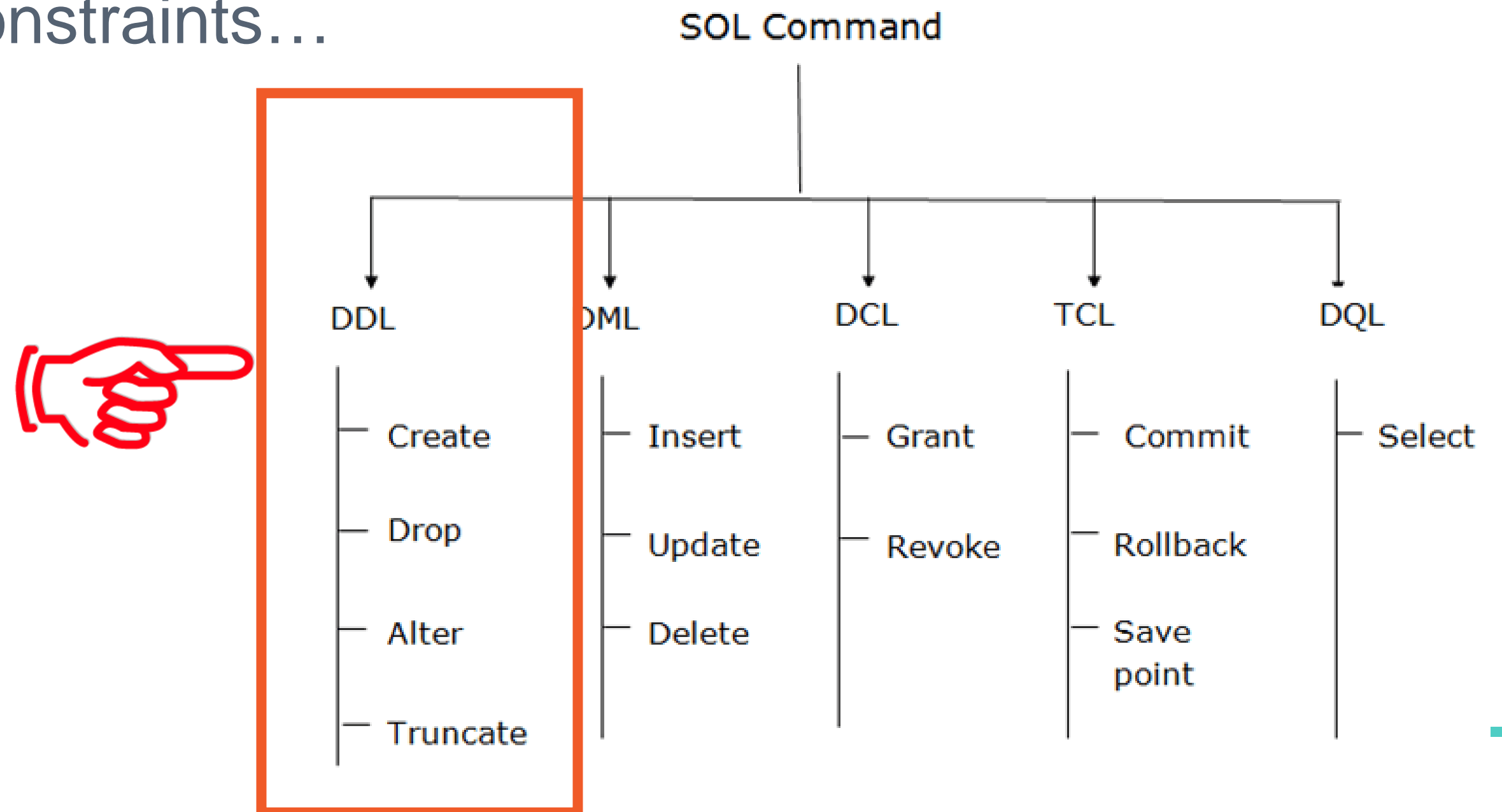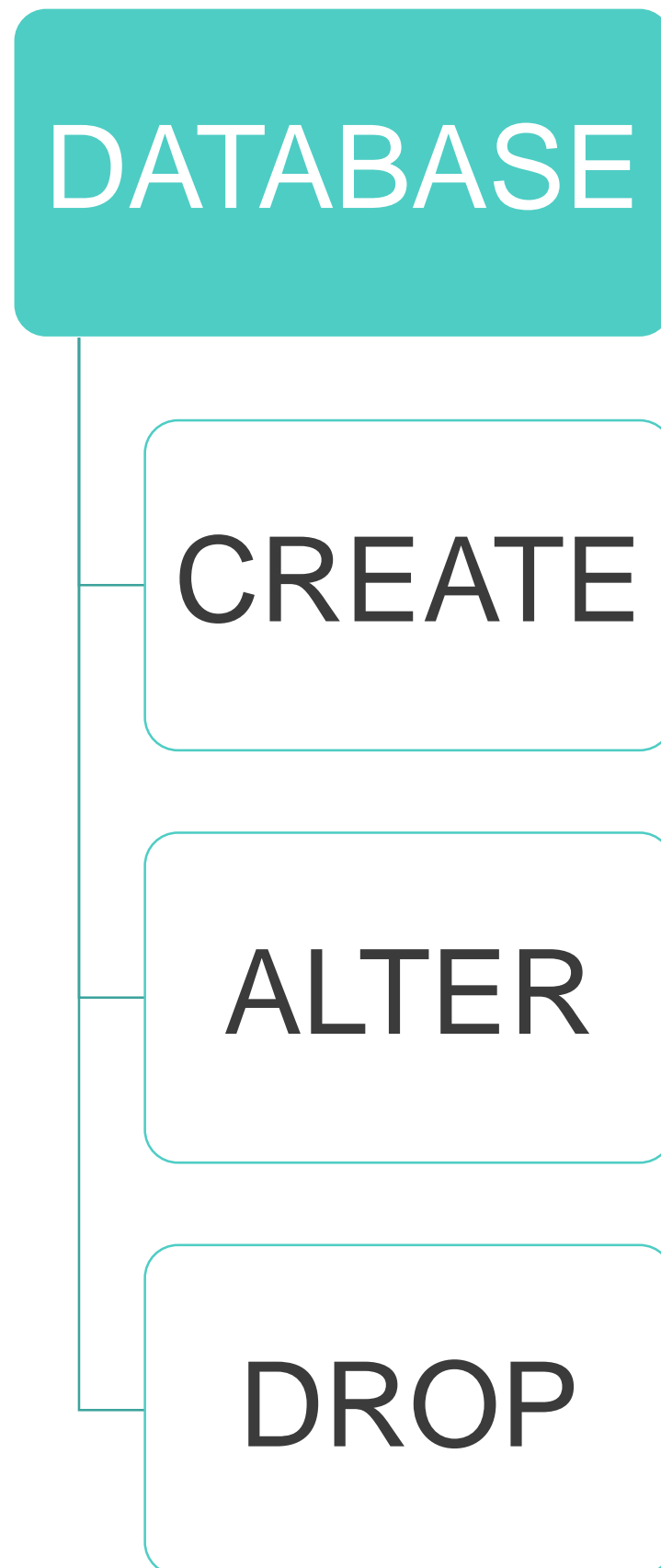
CREATE
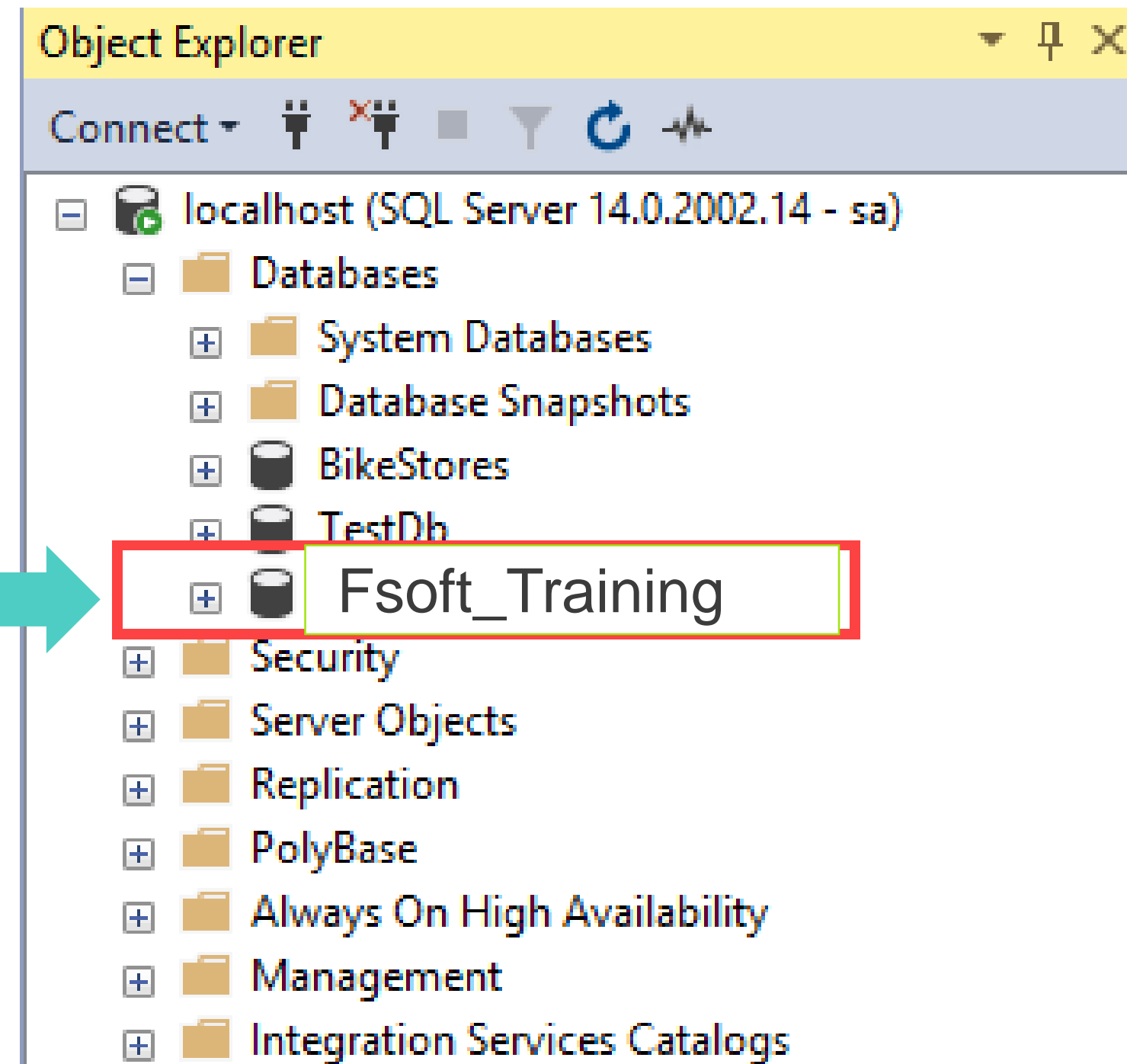
CREATE DATABASE <Database_Name>
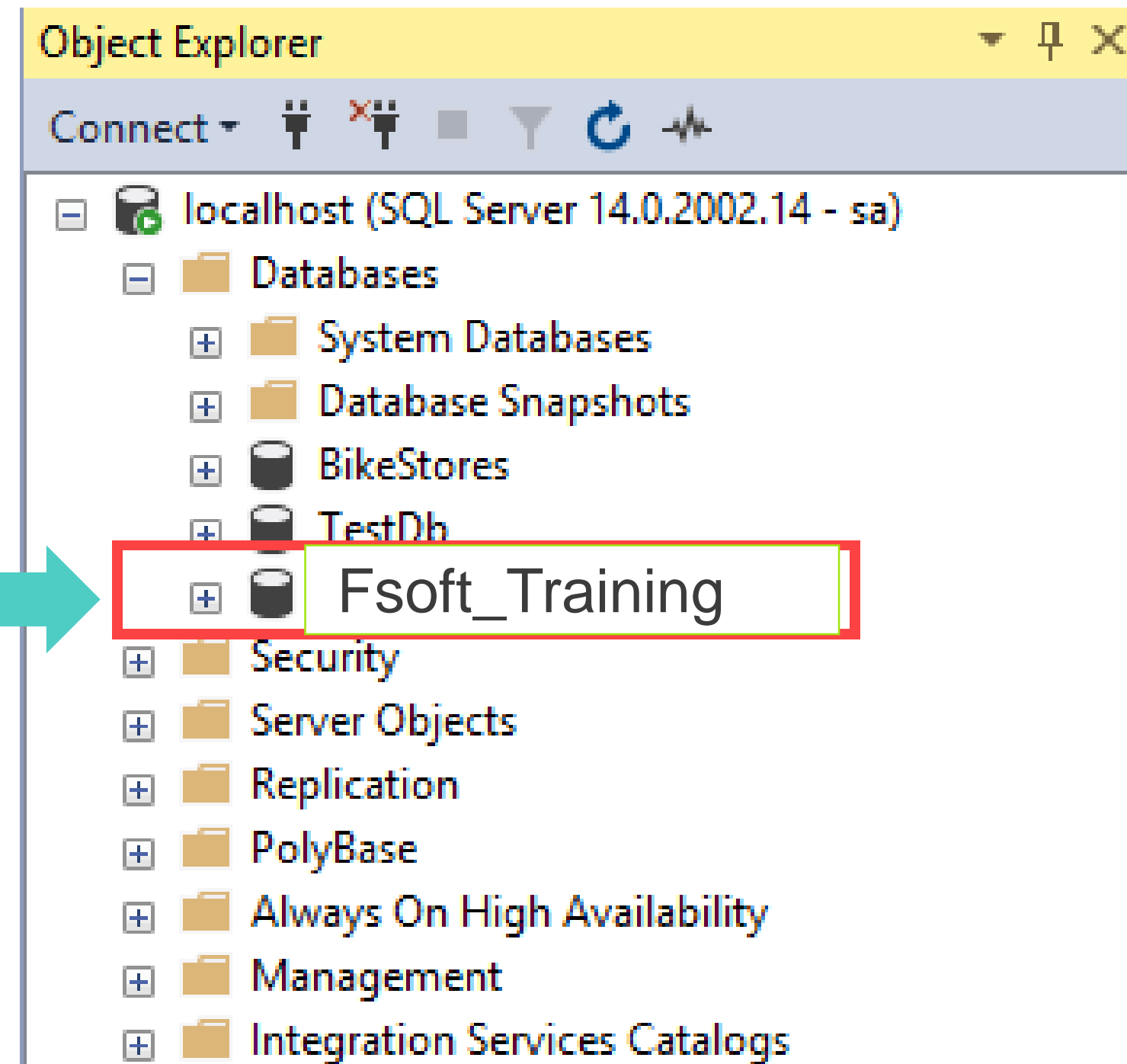
CREATE DATABASE Fsoft_training

CREATE

```sql
CREATE DATABASE Fsoft_Training
ON PRIMARY
(
    NAME = N'Fsoft_Training ',
    FILENAME = N'C:\Program Files\Microsoft SQL Server\MS
SQL14.SQL2017\MSSQL\DATA\Fsoft_Training.mdf',
    SIZE = 1024MB,
    FILEGROWTH = 256MB)
LOG ON
(
    NAME = N' Fsoft_Training_log',
    FILENAME = N'C:\Program Files\Microsoft SQL Server\MS
SQL14.SQL2017\MSSQL\DATA\Fsoft_Training_log.ldf',
    SIZE = 512MB,
    FILEGROWTH = 125MB)
```

Object Explorer

Connect ▾

localhost (SQL Server 14.0.2002.14 - sa)
  Databases
    System Databases
    Database Snapshots
    BikeStores
    TestDb
    Fsoft_Training
    Security
    Server Objects
    Replication
    PolyBase
    Always On High Availability
    Management
    Integration Services Catalogs

CREATE

```sql
CREATE DATABASE Fsoft_Training
ON PRIMARY
(
    NAME = N'Fsoft_Training ',
    FILENAME = N'C:\Program Files\Microsoft SQL Server\MS
SQL14.SQL2017\MSSQL\DATA\Fsoft_Training.mdf',
    SIZE = 1024MB,
    FILEGROWTH = 256MB)
LOG ON
(
    NAME = N' Fsoft_Training_log',
    FILENAME = N'C:\Program Files\Microsoft SQL Server\MS
SQL14.SQL2017\MSSQL\DATA\Fsoft_Training_log.ldf',
    SIZE = 512MB,
    FILEGROWTH = 125MB)
```

Object Explorer

Connect ▾

- localhost (SQL Server 14.0.2002.14 - sa)
  - Databases
    - System Databases
    - Database Snapshots
    - BikeStores
    - TestDb
    - Fsoft_Training
  - Security
  - Server Objects
  - Replication
  - PolyBase
  - Always On High Availability
  - Management
  - Integration Services Catalogs

ALTER

ALTER DATABASE <OLD_NAME> MODIFY NAME = <NEW_NAME> ;

ALTER DATABASE Fsoft_training MODIFY NAME = Fsoft_training_new ;

ALTER

```
-- SQL Server Syntax
ALTER DATABASE { database_name | CURRENT }
{
        MODIFY NAME = new_database_name
        | COLLATE collation_name
        | <file_and_filegroup_options>
        | SET <option_spec> [ ,...n ] [ WITH <termination> ]
}
```

https://docs.microsoft.com/en-us/sql/t-sql/statements/alter-database-transact-sql?view=sql-server-ver15#syntax

DROP

DROP DATABASE database_name

↓

DROP DATABASE 'Fsoft_Training'

Tip 1

Lists all databases in the SQL Server:

EXEC sp_databases;

OR

```
SELECT
    name
FROM
    master.sys.databases
ORDER BY
    name;
```

Tip 2

```sql
-- Delete Database if this exists
IF EXISTS (SELECT * FROM sys.databases WHERE Name LIKE 'Fsoft_Training')
DROP DATABASE Fsoft_Training

GO
-- Create Database by using statement
CREATE DATABASE Fsoft_Training

GO
-- Use this database on query
USE Fsoft_Training

GO
```

SQL Naming convention

The first character must be one of the following:

➢A letter as defined by the Unicode Standard 3.2. The Unicode definition of letters includes Latin characters from a through z, from A through Z, and also letter characters from other languages.

➢The underscore (_), at sign (@), or number sign (#).

➢Some Transact-SQL functions have names that start with double at signs (@@). **To avoid confusion with these functions, you should not use names that start with @@.**

SQL Naming convention

Subsequent characters can include the following:
- ➢ Letters as defined in the Unicode Standard 3.2.
- ➢ Decimal numbers from either Basic Latin or other national scripts.
- ➢ The at sign, dollar sign ($), number sign, or underscore.

The identifier must **not be** a Transact-SQL reserved word. SQL Server reserves both the **uppercase** and **lowercase** versions of reserved words.

Embedded spaces or special characters are not allowed.

Table, PK, FK and Constraints

TABLE

Table is a repository for data, with items of data grouped in one or more columns
- ➢ Data types
- ➢ Constraints
- ➢ Index

	EmployeeID	NationalIDNumber	ManagerID	Title	BirthDate	MaritalStatus	Gender	HireDate
1	1	14417807	16	Production Technician - WC60	1972-05-15 00:00:00.000	M	M	1996-07-31 00:00:00.000
2	2	253022876	6	Marketing Assistant	1977-06-03 00:00:00.000	S	M	1997-02-26 00:00:00.000
3	3	509647174	12	Engineering Manager	1964-12-13 00:00:00.000	M	M	1997-12-12 00:00:00.000
4	4	112457891	3	Senior Tool Designer	1965-01-23 00:00:00.000	S	M	1998-01-05 00:00:00.000
5	5	480168528	263	Tool Designer	1949-08-29 00:00:00.000	M	M	1998-01-11 00:00:00.000
6	6	24756624	109	Marketing Manager	1965-04-19 00:00:00.000	S	M	1998-01-20 00:00:00.000
7	7	309738752	21	Production Supervisor - WC60	1946-02-16 00:00:00.000	S	F	1998-01-26 00:00:00.000
8	8	690627818	185	Production Technician - WC10	1946-07-06 00:00:00.000	M	F	1998-02-06 00:00:00.000
9	9	695256908	3	Design Engineer	1942-10-29 00:00:00.000	M	F	1998-02-06 00:00:00.000

CREATE

CREATE TABLE <table_name> (
    column1   data_type[(size)],
    column2   data_type[(size)],
...);

```sql
-- Create table
CREATE TABLE Employee(
    EmployeeID          int,
    FirstName           nvarchar(100),
    LastName            nvarchar(100),
    AddressDetail       varchar(100),
    City                nvarchar(50),
    BirthDate           nvarchar(10),
    IsDeletedFlag       bit
)
```

Table Constraints

Table Constraints: Are used to limit the type of data that can go into a table.

We will focus on the following constraints:

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

Table Constraints

SQL constraints can be applied at:

- **Table level**
  - ✓Are declared independently from the column definition
  - ✓declare table-level constraints at the end of the CREATE TABLE statement
- **Column level:**
  - ✓Are declared when define columns for the table.
  - ✓It is applied particularly to the column where it attached to

Table Constraints

Column level

```
CREATE TABLE table_name (
    column1 datatype constraint,
    column2 datatype constraint,
    column3 datatype constraint,
    ....
);
```

Table level

```
CREATE TABLE table_name (
    column1 datatype ,
    column2 datatype ,
    column3 datatype ,
    ....

    CONSTRAINT <Name> CHECK( constraint)
    CONSTRAINT <PK_Name> PRIMARY KEY (colum
n1, ,Column2));
```

Primary key?

A primary key is a combination of fields which uniquely specify a row.

This is a special kind of unique key, and it has implicit NOT NULL constraint. It means, Primary key values cannot be NULL.

Foreign key

A **foreign key** is one table which can be related to the primary key of another table.

Relationship needs to be created between two tables by referencing foreign key with the primary key of another table.

A **FOREIGN KEY** provides a way of enforcing referential integrity within SQL Server.
In simple words, foreign key **ensures** values in **one table** must be **present in another table**.

Foreign key

UNIQUE and PRIMARY KEY

UNIQUE: Enforce the uniqueness of the values in a set of columns

- **Synstax:**

    CONSTRAINT unique_name UNIQUE (col_names)

PRIMARY KEY: Specify primary key of table.

- **Syntax:**

    [CONSTRAINT *PK_Name*]
    PRIMARY KEY [col_names*]

Table Constraint

Column Name	Data Type	Allow Nulls
🔑 ProductID	int	☐
ProductName	nvarchar(50)	☑
Description	nvarchar(100)	☑
RetailPrice	float	☑
WholeSalePrice	nvarchar(100)	☑

Table Constraint

```sql
-- PRIMARY KEY Constraint
CREATE TABLE Product(
    ProductID int
,   ProductName nvarchar(50)
,   Description nvarchar(100)
,   RetailPrice float
,   WholeSalePrice nvarchar(100)
,   PRIMARY KEY(ProductID)
)
```

```sql
-- PRIMARY KEY Constraint
CREATE TABLE Product(
    ProductID int
,   ProductName nvarchar(50)
,   Description nvarchar(100)
,   RetailPrice float
,   WholeSalePrice nvarchar(100)
,   PRIMARY KEY(ProductID)
)
```

FOREIGN KEY

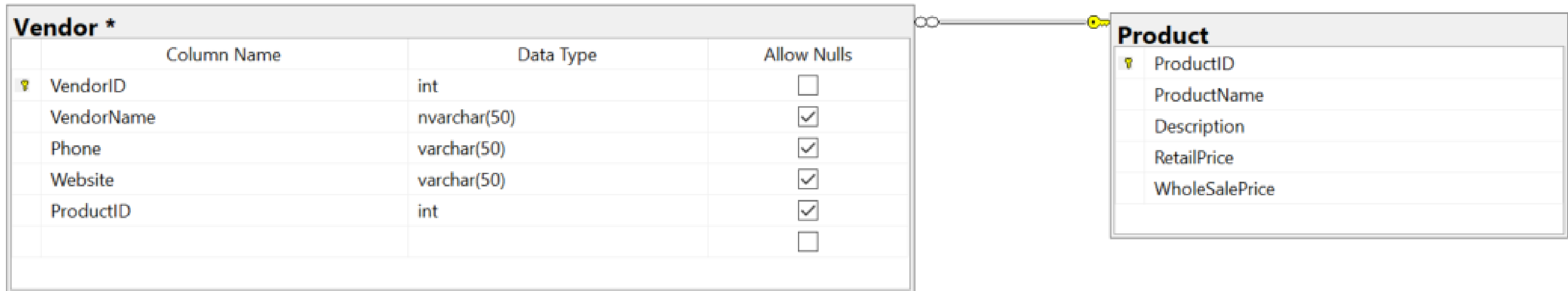FOREIGN KEY: Used to define relationships between tables in the database.

- **Syntax:**

  [CONSTRAINT *FK_Name]*
  FOREIGN KEY [*(col_names)]*
  REFERENCES *reference_table(col_names)*

DEFAULT: Defaults specify what values are used in a column if you do not specify a value for the column when you insert a row.

FOREIGN KEY

Vendor *		
Column Name	**Data Type**	**Allow Nulls**
🔑 VendorID	int	☐
VendorName	nvarchar(50)	☑
Phone	varchar(50)	☑
Website	varchar(50)	☑
ProductID	int	☑
		☐

Product	
🔑 ProductID	
ProductName	
Description	
RetailPrice	
WholeSalePrice	

FOREIGN KEY

```sql
-- FOREIGN KEY constraint
CREATE TABLE Vendor(
    VendorID     int
,   VendorName   nvarchar(50)
,   Phone        varchar(50)
,   Website      varchar(50)
,   ProductID    int
,   PRIMARY KEY(VendorID)
,   FOREIGN KEY (ProductID) REFERENCES Product(ProductID)
)
```

NOT NULL and CHECK

NOT NULL: Specifies that the column does not accept NULL values.

CHECK: Enforce domain integrity by limiting the values that can be put in a column.

- **Syntax**:

  [CONSTRAINT *constraint_name]*

  CHECK (*condition)*

NOT NULL and CHECK

```sql
CREATE TABLE Timesheet(
    PayrollDate datetime
,   WorkingHours int DEFAULT 8
,   CONSTRAINT IsGreaterThan0 CHECK (WorkingHours>0)
)
```
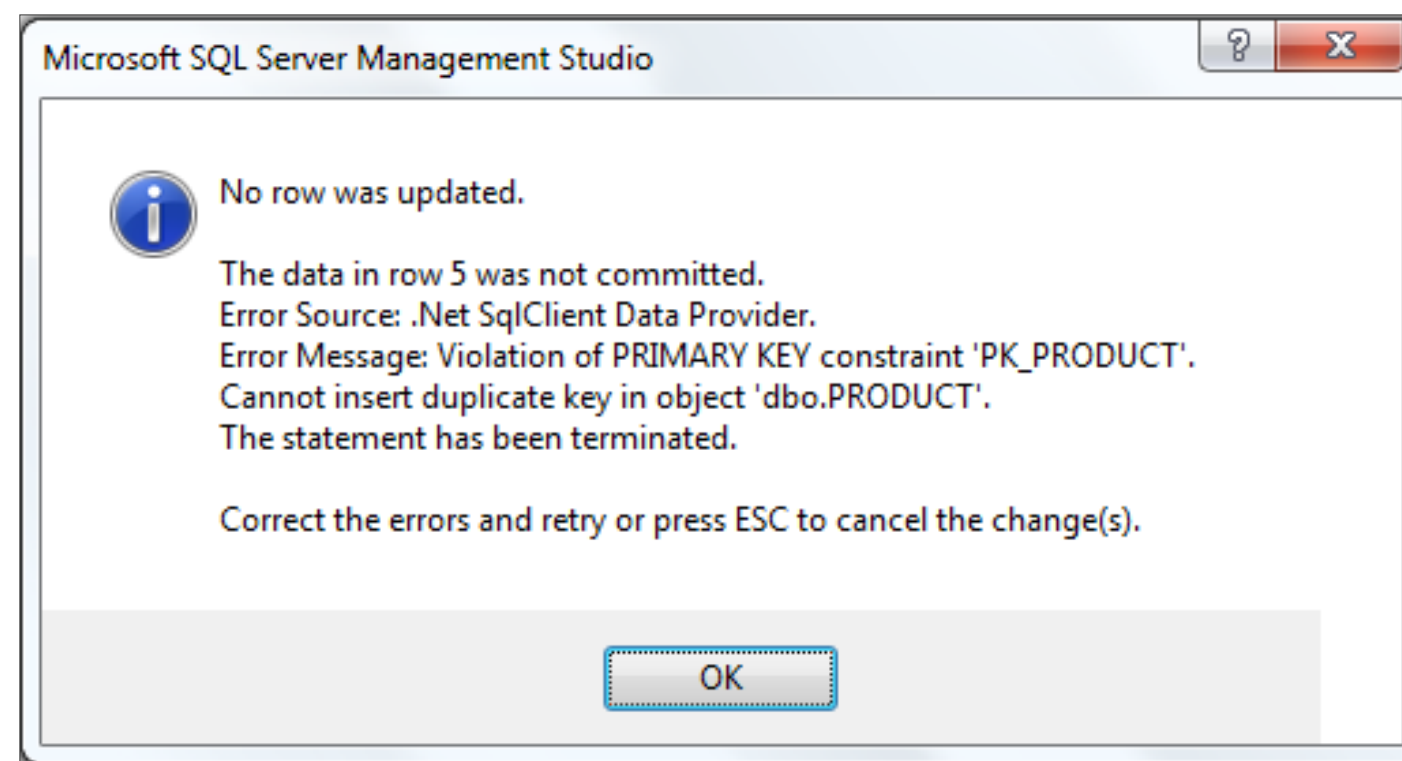
Identity

Identity

Identity has:

- A seed
- An increment

Seed is the initial value

Increment is the value by which we need to skip to fetch the next value

For example:

- Identity(1,2) will generate sequence numbers 1,3,5,7…

Identity(1,1)
1
2
3
4
5
…

Identity(1,3)
1
4
7
10
13
…

Identity(1,2)
1
3
5
7
9
…

Identity

```sql
CREATE TABLE Persons (
    Personid int IDENTITY(1,1) PRIMARY KEY,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int
);
```

Identity

```sql
CREATE TABLE Persons (
    Personid int NOT NULL AUTO_INCREMENT,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    PRIMARY KEY (Personid)
);
```

Computed columns

```sql
CREATE TABLE persons
(
    person_id   INT PRIMARY KEY IDENTITY,
    first_name  NVARCHAR(100) NOT NULL,
    last_name   NVARCHAR(100) NOT NULL,
    dob         DATE
);
```

full_name = first_name + ' ' + last_name

```sql
CREATE TABLE persons
(
    person_id   INT PRIMARY KEY IDENTITY,
    first_name  NVARCHAR(100) NOT NULL,
    last_name   NVARCHAR(100) NOT NULL,
    full_name   AS (first_name + ' ' + last_name),
    dob         DATE
);
```

```sql
ALTER TABLE persons
ADD full_name
    AS (first_name + ' ' + last_name);
```

Drop table

Sometimes, you want to remove a table that is no longer in use.

DROP TABLE *table_name*

ALTER table

```
ALTER TABLE Persons
ADD PRIMARY KEY (ID);

            ALTER TABLE Persons
            ADD CONSTRAINT PK_Person PRIMARY KEY (ID,LastName);

                        ALTER TABLE Persons
                        ADD Name VARCHAR (255) NOT NULL

                ALTER TABLE Persons
                DROP COLUMN Name1, Name2;

    ALTER TABLE sales.price_lists
    DROP CONSTRAINT ck_positive_price;
```

Truncate statement

➢ Removes all rows in a table.

➢ Table structure and its columns, constraints, indexes, …remain.

    ✓   Resets the identity value.

    ✓   Releases the memory used.

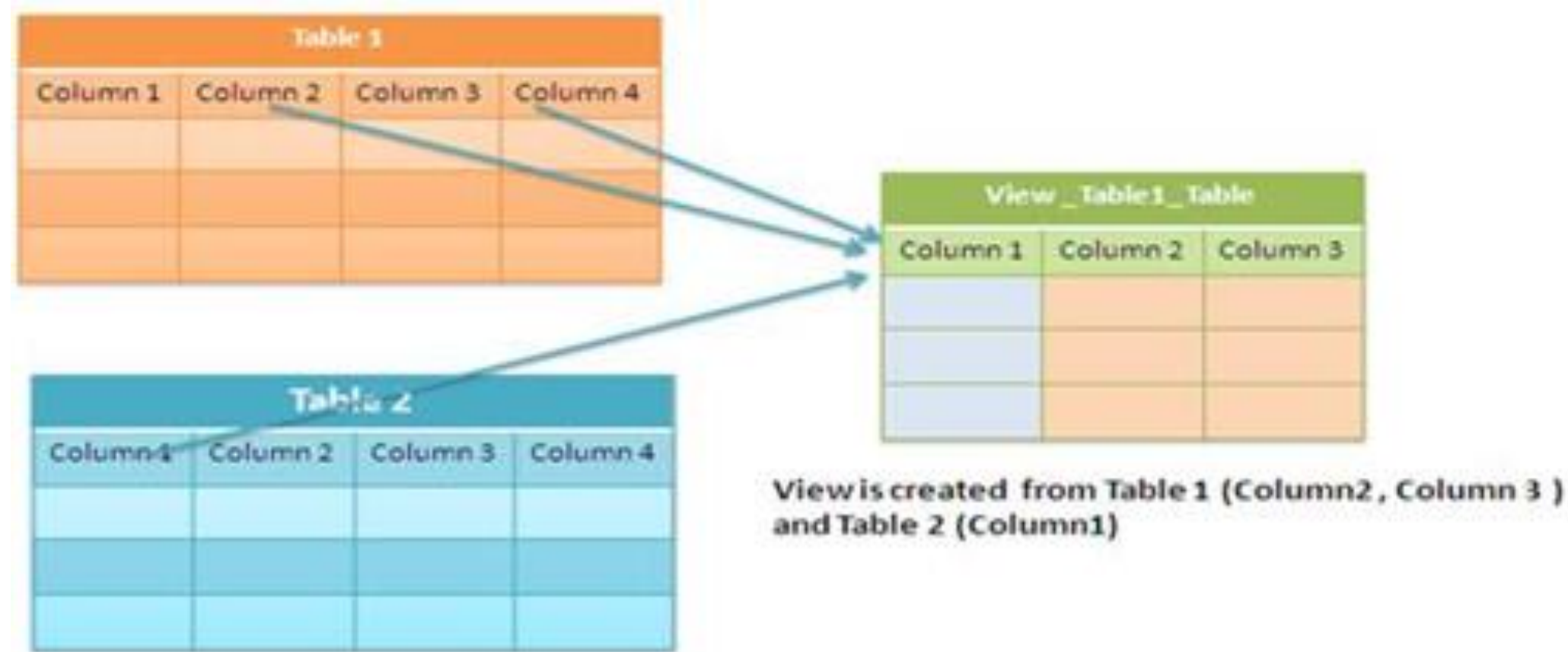# TRUNCATE TABLE *table_name;*

VIEWS

What is a view?

A View is a logical or virtual table. The fields in a view are fields from one or more real tables in the database.

There are two major reasons you might want to use views:

Views allow you to limit the data users can access

Views reduce complexity for end users.

View is created from Table 1 (Column2 , Column 3) and Table 2 (Column1)

Creating a view

CREATE VIEW View_Name [list of column names]

AS

SELECT…

Example:

```
CREATE VIEW view_EmployeeByDpt
AS
SELECT  ID, NAME, AGE, DEPT_NAME
FROM    EMP, DEPARTMENT
WHERE   EMP.DEP_ID = DEPARTMENT.DEPT_ID
```

SELECT * FROM view_EmployeeByDpt

Table: EMP

ID	NAME	AGE	DEP_ID
1	John	25	3
2	Mike	30	2
3	Parm	25	1
4	Todd	23	4
5	Sara	35	1
6	Ben	40	3

Table: DEPARTMENT

DEPT_ID	DEPT_NAME
1	IT
2	Payroll
3	HR
4	Admin

view_EmployeeByDpt

ID	NAME	AGE	DEPT_NAME
1	John	25	HR
2	Mike	30	Payroll
3	Parm	25	IT
4	Todd	23	Admin
5	Sara	35	IT
6	Ben	40	HR
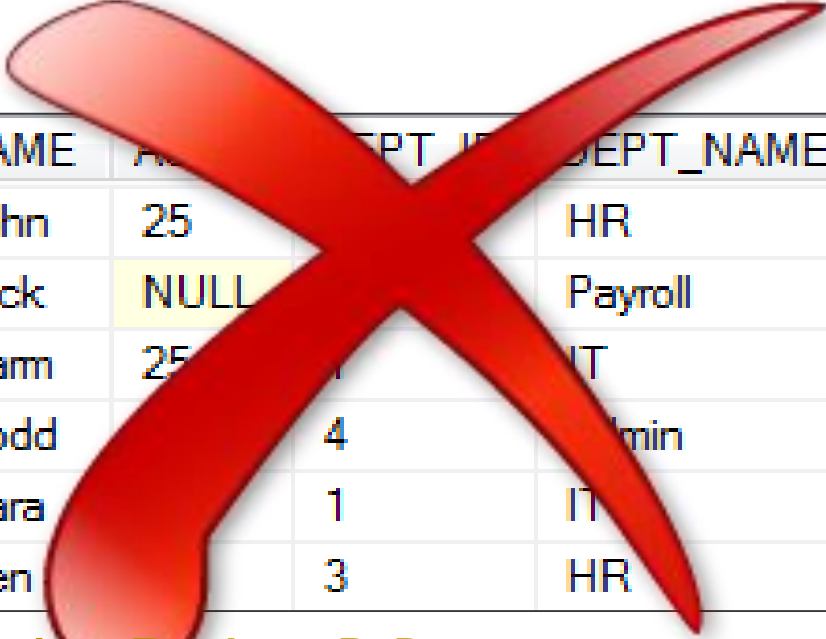
Deleting a view

Syntax:

> DROP   VIEW   View_Name

Example:

> DROP VIEW view_EmployeeByDpt

ID	NAME	A...	...PT_I...	DEPT_NAME
1	John	25		HR
2	Nick	NULL		Payroll
3	Parm	25		IT
4	Todd		4	...min
5	Sara		1	IT
6	Ben		3	HR

view_EmployeeByDpt

3

Naming convention and styles

NAMING CONVENTION AND STYLES

1. Use UPPER CASE for all T-SQL constructs, excepts Types

Correct:

```
SELECT MAX([Salary]) FROM dbo.[EmployeeSalary]
```

Incorrect:

```
SELECT max([Salary]) from dbo.[EmployeeSalary]
```

2. Use lower case for all T-SQL Types and Usernames

Correct:

```
DECLARE @MaxValue int
```

Incorrect:

```
DECLARE @MaxValue INT
```

3. **Use Pascal casing for all UDO's**

Correct:

```sql
CREATE TABLE dbo.EmployeeSalary

(

        EmployeeSalaryID  INT

)
```

Incorrect:

```sql
CREATE TABLE dbo.Employeesalary

(

        EmployeesalaryID  int

)
```

4.Avoid abbreviations and single character names

> **Correct:**
>
> ```
> DECLARE @RecordCount int
> ```
>
> **Incorrect:**
>
> ```
> DECLARE @Rc int
> ```

5.UDO naming must confer to the following regular expression ([a-zA-Z][a-zA-Z0-9]).

> Do not use any special or language dependent characters to name objects. Constraints can use the underscore character.
>
> **Correct:**
>
> ```
> CREATE TABLE dbo.[EmployeeSalary]
> ```
>
> **Incorrect:**
>
> ```
> CREATE TABLE dbo.[Employee Salary]
> ```

6. Use the following prefixes when naming objects

usp_: User stored procedures

ufn_: User defined functions

view_: Views

IX_: Indexes

usp_: User stored procedures

DF_: Default constraints

PK_: Primary Key constraints

FK_: Foreign Key constraints

CHK_: Check constraints

UNI_: Unique constraints

Correct:

```
CREATE PROCEDURE dbo.usp_EmployeeSelectAll
```

Incorrect:

```
CREATE PROCEDURE dbo.EmployeeSelectRetired --without
preffixed
```

7.Name tables in the singular form

    Correct:

```
CREATE TABLE dbo.[Employee]
```

    Incorrect:

```
CREATE TABLE dbo.[Employees]
```

8.Tables that map one-to many, many-to-many relationships should be named by concatenating the names of the tables in question, starting with the most central table's name.

    Correct:

```
CREATE TABLE dbo.[EmployeeSalary]
```

Thank you!

Any questions?