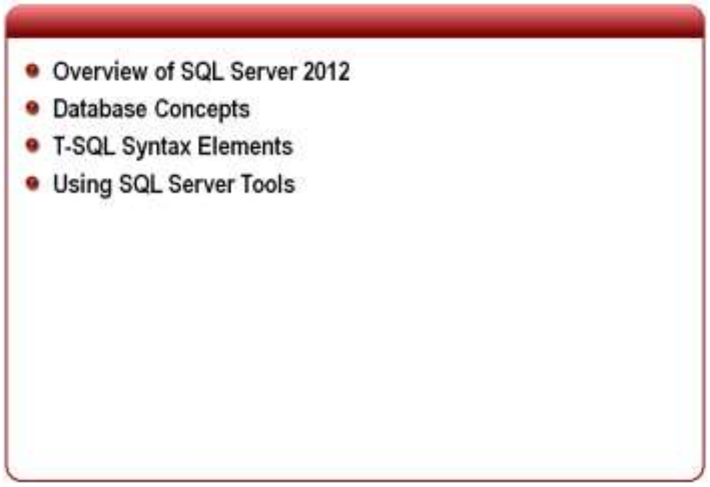
# Module 1: Fundamentals of Database and T-SQL

## Contents

## Module Overview



- Overview of SQL Server 2012
- Database Concepts
- T-SQL Syntax Elements
- Using SQL Server Tools

Microsoft SQL Server is considered one the most commonly used system for data base management in the world. His popularity has been gained by high degree of stability, security, and business intelligence and integration functionality. As in earlier versions, Microsoft SQL Server 2012 Database Engine is a database server that accepts queries from clients, evaluates them and then internally executes them, to deliver results to the client. The client is an application that produces queries, and through a database provider and communication protocol sends requests to the server, and retrieve the result for client side processing and/or presentation.

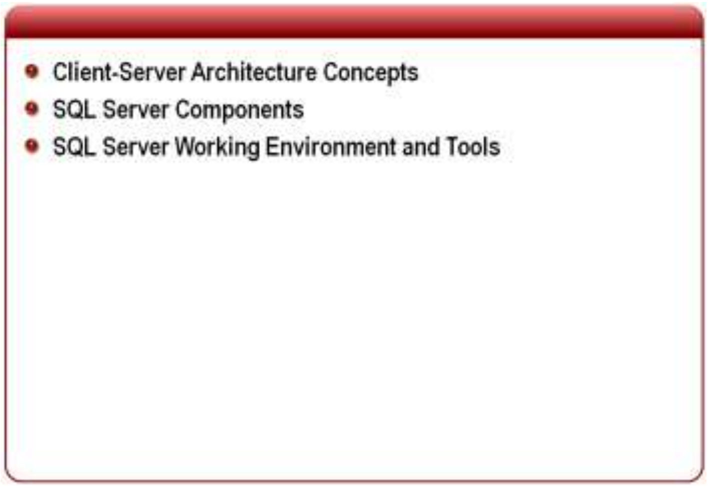
Before starting to write queries and work with Microsoft SQL Server 2012 it’s a good idea to gain a good understanding of how the software works. With a good understanding of the product and its mechanics you’ll be able to write more efficient queries and get results much faster.

### Objectives

After completing this module, you will be able to:

- Understand what SQL Server 2012 is
- Understand the basics of Database Concepts
- Understand the basics of T-SQL Syntax Elements
- Better utilize SQL Server Tools

## Lesson 1: Overview of SQL Server 2012



- Client-Server Architecture Concepts
- SQL Server Components
- SQL Server Working Environment and Tools

When writing queries it’s important to understand that the interaction between the tool of choice and the database based on a client-server architecture, and the processes that are involved. It’s also important to understand which components are available and what functionality they provide. While Microsoft SQL Server 2012 host is a relational database engine it offers a variety of components such as an OLAP engine, ETL engine, a reporting environment, and much more.

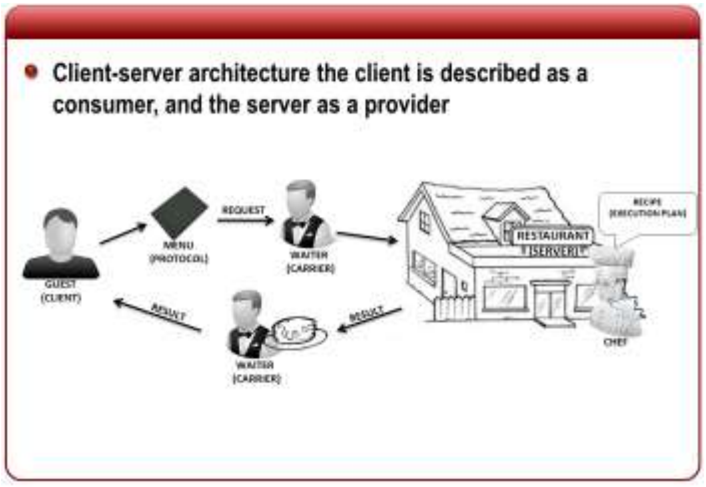
With a broader understanding of the full product and its components and tools, you’ll be able make better use of its functionality, and also benefit from using the right tool for specific jobs.

### Objectives

After completing this lesson, you will be able to:

- Understand Client-Server Architecture Concepts
- Understand SQL Server Components
- Understand SQL Server Working Environment and Tools

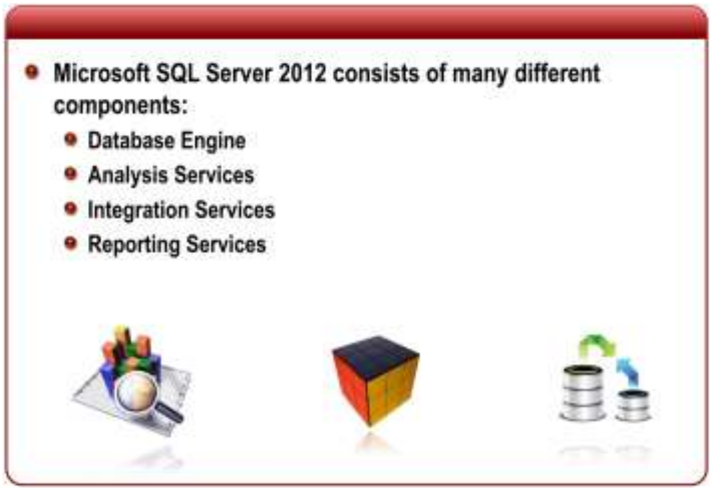
## Client-Server Architecture Concepts



- Client-server architecture the client is described as a consumer, and the server as a provider

In a client-server architecture the client is described as a consumer, and the server as a provider. To compare an application (use) with a dinner at a fine restaurant the guest (client), visits the restaurant (server), for having a dinner (application). The restaurant consists of a variety of endpoints, protocols and internal processes. To send a request we need a waiter who acts as the carrier between the client and the server, and a mutual protocol, the menu. When we order pancakes the waiter carries the request to the kitchen and put the request in a queue. Neither the waiter nor the guest needs to know what pancakes consist of, or how they are made. It is an internal process of the server. When the chef is ready he retrieves the request from the worker queue, evaluates and translates the request to an execution plan, the recipe. He then evaluates the best method and starts to allocate resources, ingredients and tools, and starts cooking. Depending if the client is connected or disconnected the waiter either waits for the dish of pancakes, or starts serving new guests. When the dish is ready the waiter then delivers it to the guest, this is returning the result to the client.

## SQL Server Components



- Microsoft SQL Server 2012 consists of many different components:
  - Database Engine
  - Analysis Services
  - Integration Services
  - Reporting Services

Microsoft SQL Server 2012 consists of many different components to serve a variety of organizational needs of their data platform. Some of these are:

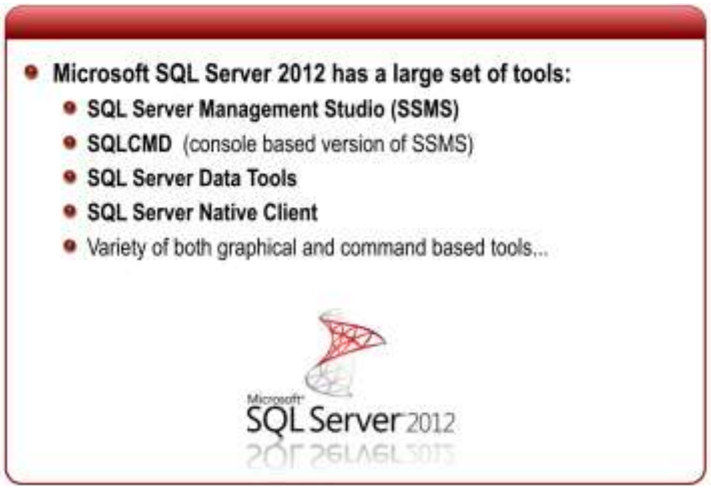
**Database Engine** is the relational database management system (RDBMS), which hosts databases and processes queries to return results of structured, semi-structured, and non-structured data in online transactional processing solutions (OLTP).

**Analysis Services** is the online analytical processing engine (OLAP) as well as the data mining engine. OLAP is a way of building multi-dimensional data structures for fast and dynamic analysis of large amounts of data, allowing users to navigate hierarchies and dimensions to reach granular and aggregated results to achieve a comprehensive understanding of business values. Data mining is a set of tools used to predict and analyze trends in data behavior and much more.

**Integration Services** supports the need to extract data from sources, transform it, and load it in destinations (ETL) by providing a central platform that distributes and adjusts large amounts of data between heterogeneous data destinations. .

**Reporting Services** is a central platform for delivery of structured data reports and offers a standardized, universal data model for information workers to retrieve data and model reports without the need of understanding the underlying data structures.

## SQL Server Working Environment and Tools



- Microsoft SQL Server 2012 has a large set of tools:
  - SQL Server Management Studio (SSMS)
  - SQLCMD (console based version of SSMS)
  - SQL Server Data Tools
  - SQL Server Native Client
  - Variety of both graphical and command based tools...

Microsoft SQL Server 2012

On the client side of Microsoft SQL Server 2012 is a large set of tools to help us benefit from all aspects of the product.

**SQL Server Management Studio (SSMS)** is the main tool for the management and development, of the database engine, and an administrative interface of all main components of the product. It delivers a designer to T-SQL queries with features as solutions, projects, intellisense, templates, and much more.

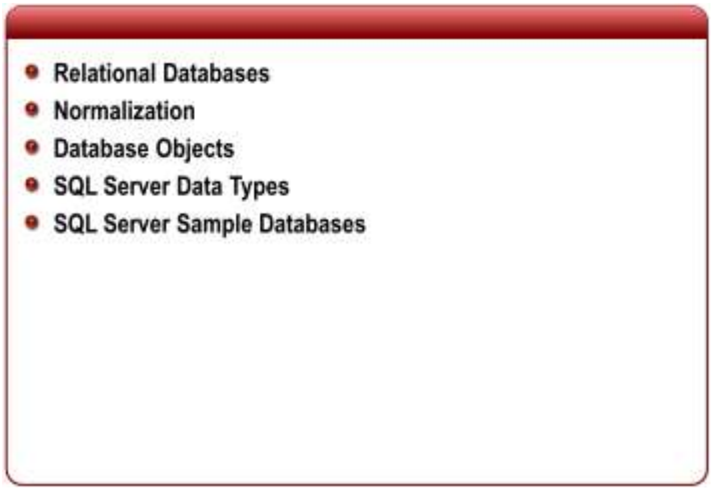
**SQLCMD** is a console based version of SQL Server Management Studio (SSMS) used for administration and developer tasks

**SQL Server Data Tools** is the business intelligence development tool that offers designers and wizard's templates to build SSIS (SQL Server Integration Services) packages, OLAP cubes, reports, data mining structures and much more.

**SQL Server Native Client** provides connectivity between applications and the server platform.

Besides these tools there is a variety of both graphical and command based tools to utilize Microsoft SQL Server 2012.

## Lesson 2: Database Concepts



- Relational Databases
- Normalization
- Database Objects
- SQL Server Data Types
- SQL Server Sample Databases

In order to write queries against a database it is important to understand the concept of relational databases, and what kind of objects they contain. Microsoft SQL Server 2012 Database Engine is a host for relational databases that stores and manages structured, semi-structured, and non-structured data as columns in tables.

Normalization is a methodology of structuring tables to organize data in more efficient structures. Without normalization tables would look more like spreadsheets with risk of slow performance, un-necessary redundancy of data, and many other negative effects.

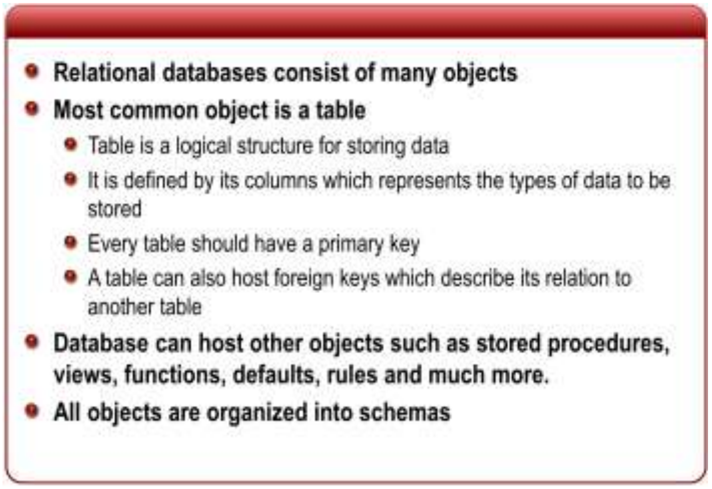
A relational database can consist of many tables and columns with many relations between them. To write queries against the database you need to have a thorough understanding of its data model and as the model complexity grows it will be more important to know how to navigate its schema.

### Objectives

After completing this lesson, you will be able to:

- Describe a relational database
- Have a basic understanding of normalization
- Describe different kinds of database objects
- Describe and use SQL Server Data Types
- Navigate through the SQL Server sample databases

## Relational Databases



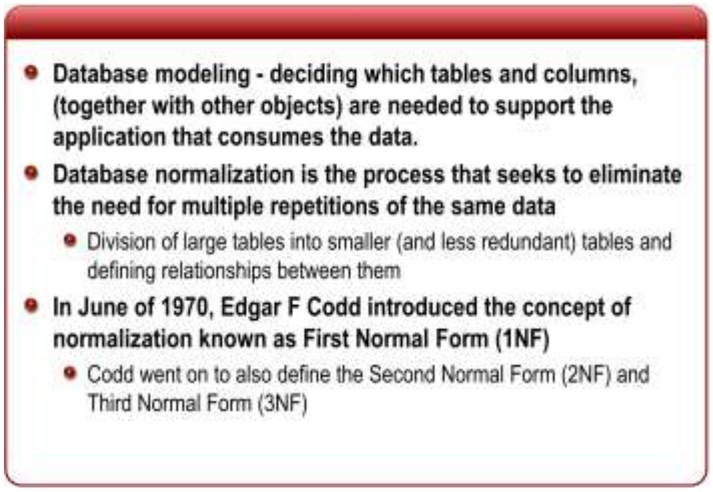
- Relational databases consist of many objects
- Most common object is a table
  - Table is a logical structure for storing data
  - It is defined by its columns which represents the types of data to be stored
  - Every table should have a primary key
  - A table can also host foreign keys which describe its relation to another table
- Database can host other objects such as stored procedures, views, functions, defaults, rules and much more.
- All objects are organized into schemas

Relational databases hosted by Microsoft SQL Server 2012 consist of many objects, the most common object is a table. A table is a logical structure for storing and retrieving rows of data into columns. It is defined by its columns which represents the types of data to be stored.

Every table should have a primary key that is a unique representation of a single row. It is usually a single column that either hosts a sequential number or a unique identifier that will never be used more than once. Tables can also contain two or more columns that together represent the unique row of the table. A table can also host foreign keys which describe its table reference, or relation to another table. For example, a “customer table” has a unique “customer id”-column, the same “customer id” is then represented in an “order table” to tie the actual order to its customer. This allows for one customer to have none to many orders, and requires the customer to be created ahead of the order.

Besides tables, a database can host other objects such as stored procedures, views, functions, defaults, rules and much more. All objects are organized into schemas as files are organized into folders. A schema can also leverage permissions which ease the process of securing the database.

## Normalization



- Database modeling - deciding which tables and columns, (together with other objects) are needed to support the application that consumes the data.
- Database normalization is the process that seeks to eliminate the need for multiple repetitions of the same data
  - Division of large tables into smaller (and less redundant) tables and defining relationships between them
- In June of 1970, Edgar F Codd introduced the concept of normalization known as First Normal Form (1NF)
  - Codd went on to also define the Second Normal Form (2NF) and Third Normal Form (3NF)

Modeling, or designing, a database is deciding which tables and columns, together with other objects are needed to support the application that consumes the data. As applications work with objects and attributes, a common mistake creating a "physical" table structure as a copy of the object model. Even if the database should support the object model, the main responsibility of the database is to handle data. When different objects share the same types of data, the database designer needs to look further and model the table based on the types of data.

Database normalization is the process that seeks to eliminate the need for multiple repetitions of the same data. It implies specific form of fields and tables organization that will minimize data redundancy and dependency. Therefore, the process of normalization often requires division of large tables into smaller (and less redundant) tables and defining relationships between them. The objective is to isolate data so that additions, deletions, and modifications of a field can be made in just one table and then propagated through the rest of the database via the defined relationships.

In June of 1970, Edgar F Codd published an article "A Relational Model of Data for Large Shared Data Banks" introduced the concept of normalization and what we now know as the First Normal Form (1NF) that laid the groundwork for the theory of relational databases and the ruling for how they should be constructed. "Codd's 12 rules" are a set of thirteen rules numbered 0 to 12 that describe the requirements for what should be considered as a relational database.

Codd went on to also define the Second Normal Form (2NF) and Third Normal Form (3NF) in 1971 and the three the basis upon which subsequent NFs were formed.

**1NF** – Every table must uniquely represent each row, not carry any duplicate rows, and not have any repeating groups. For example a Contacts table, with the attributes:

- Name
- City
- EmailAddress
- PhoneNumber1
- PhoneNumber2

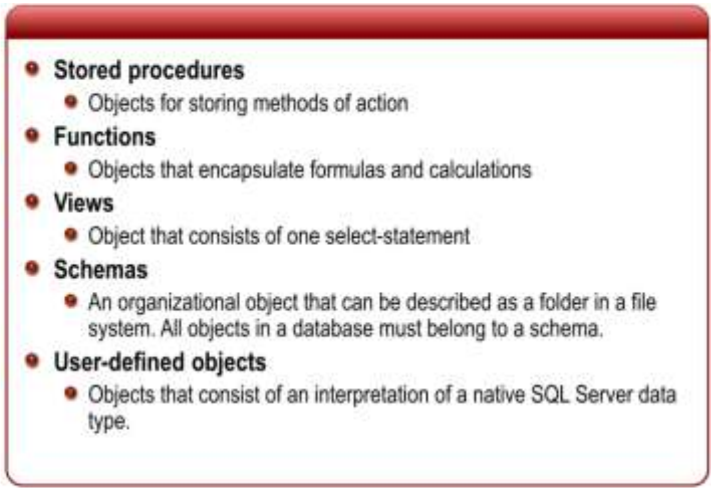
- PhoneNumber3

The PhoneNumberX-attributes are a repeating group and the table should consist of four columns;

- Name
- City
- EmailAddress
- PhoneNumber.

- 2NF – The table must meet the criteria of 1NF and any of the non-key attributes doesn't rely on a subset of the key. As a contact might have many phone numbers, this model might generate many rows for one person. If Name uniquely represents a contact, the model should be: Contacts(Name, City, EmailAddress) and PhoneNumbers(Name, PhoneNumber)
- 3NF – The table must meet the criteria of 2NF and every non-key attribute is directly dependent on its key, a superkey. For example, an employee-table: Employees(EmployeeKey, Name, Department, DepartmentLocation). DepartmentLocation is dependent of the Department, not the Employee. To adhere to the 3NF a better solution would be: Employees(EmployeeKey, Name, DepartmentKey), Departments(DepartmentKey, Name, LocationKey), and Locations(LocationKey, Name).

## Database Objects



- **Stored procedures**
  - Objects for storing methods of action
- **Functions**
  - Objects that encapsulate formulas and calculations
- **Views**
  - Object that consists of one select-statement
- **Schemas**
  - An organizational object that can be described as a folder in a file system. All objects in a database must belong to a schema.
- **User-defined objects**
  - Objects that consist of an interpretation of a native SQL Server data type.

Notwithstanding the fact that the data is stored in tables, stored procedures, functions, views and the schemas are objects that share equal importance in daily work with databases.

**Stored procedures** are objects in the database for storing methods of action. A procedure could be described as a program that processes actions in the database. For example, inserts, updates, and/or delete rows in a table.

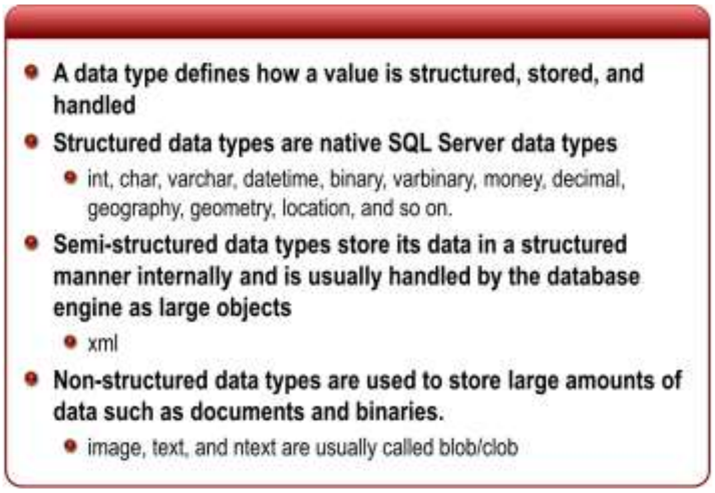
**Functions** are objects in the database that encapsulate formulas and calculations and return either scalar values, or sets of data.

**Views** are an object that consists of one select-statement, and are referenced as a table. Normalized database views bring normalized data together and masks complex table structures.

**Schemas** are an organizational object and can be described as a folder in a file system. All objects in a database must belong to a schema.

**User-defined objects** are objects that consist of an interpretation of a native SQL Server data type and offer a standardized method of describing the value of columns, variables, and parameters.

## SQL Server Data Types



- A data type defines how a value is structured, stored, and handled
- Structured data types are native SQL Server data types
  - int, char, varchar, datetime, binary, varbinary, money, decimal, geography, geometry, location, and so on.
- Semi-structured data types store its data in a structured manner internally and is usually handled by the database engine as large objects
  - xml
- Non-structured data types are used to store large amounts of data such as documents and binaries.
  - image, text, and ntext are usually called blob/clob

Microsoft SQL Server 2012 Database Engine utilizes a wide selection of data types. A data type is a definition of how a value is structured, stored, and handled. There are data types for any kind of structured, semi-structured, and non-structured type of data.

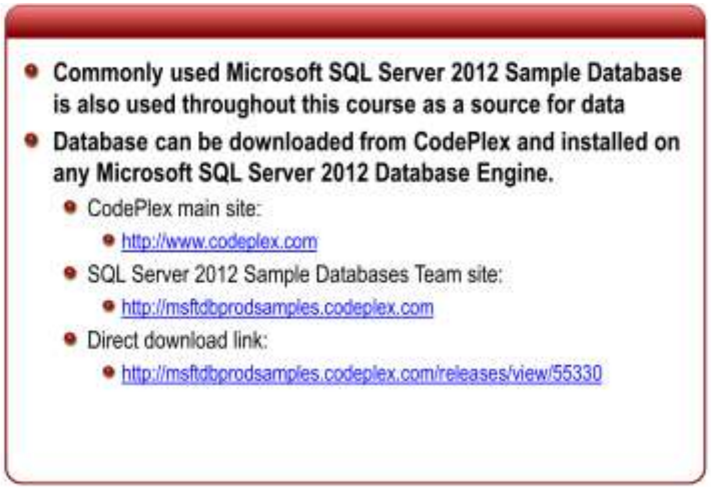
Structured data types are native SQL Server data types such as int, char, varchar, datetime, binary, varbinary, money, decimal, geography, geometry, location, and so on. Character-based data types support both non-unicode, char/varchar, and Unicode, nchar/nvarchar.

Semi-structured data types such as xml, store its data in a structured manner internally and is usually handled by the database engine as large objects, but at the same time offers flexibility to add custom functions and indexes to efficiently display its content.

Non-structured data types such as image, text, and ntext are usually referred to as large objects called blob/clob and are used to store large amounts of data such as documents and binaries in the database. Also, varbinary(max), varchar(max), and nvarchar (max) are seen as non-structured objects, but function a bit differently than the previous three.

Every data type offers specific features for a specific use. When designing a database, it’s important to choose the right data type for every column of a table.

## SQL Server Sample Databases



- Commonly used Microsoft SQL Server 2012 Sample Database is also used throughout this course as a source for data
- Database can be downloaded from CodePlex and installed on any Microsoft SQL Server 2012 Database Engine.
  - CodePlex main site:
    - http://www.codeplex.com
  - SQL Server 2012 Sample Databases Team site:
    - http://msftdbprodsamples.codeplex.com
  - Direct download link:
    - http://msftdbprodsamples.codeplex.com/releases/view/55330

The Microsoft SQL Server 2012 Sample Database  used throughout this course as a source for data. The reason we use the database is that it is commonly used when describing syntax and scenarios in the SQL Server 2012 Books-Online, on websites and blogs all over the internet, and that it supports every aspect of data in the real world. It's also prepared to support all functionality that is available in SQL Server.

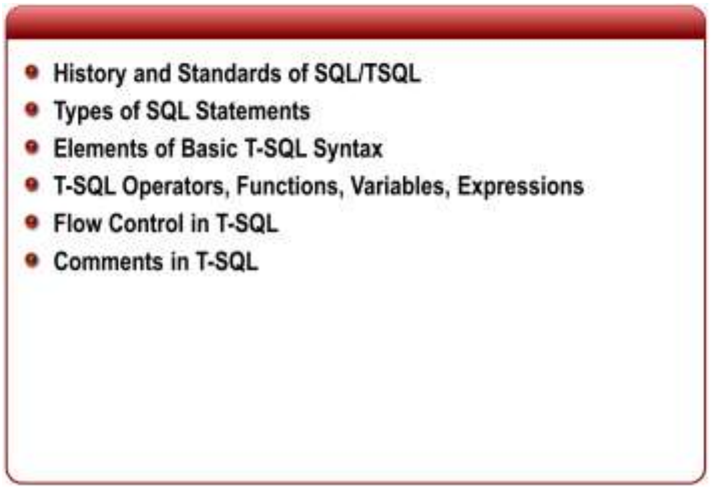
The SQL Server 2012 Sample Database can be downloaded from CodePlex, an Open Source Project hosting community, to be installed on any Microsoft SQL Server 2012 Database Engine.

CodePlex main site: http://www.codeplex.com

SQL Server 2012 Sample Databases Team site: http://msftdbprodsamples.codeplex.com

Direct download link: http://msftdbprodsamples.codeplex.com/releases/view/55330

## Lesson 3: T-SQL Syntax Elements



- History and Standards of SQL/TSQL
- Types of SQL Statements
- Elements of Basic T-SQL Syntax
- T-SQL Operators, Functions, Variables, Expressions
- Flow Control in T-SQL
- Comments in T-SQL

In order to successfully write queries against Microsoft SQL Server 2012 databases you need to understand the query language T-SQL, Transact - Structured Query Language. T-SQL, and both the ANSI and ISO standard, offers words like SELECT, INSERT, UPDATE, and DELETE as well as FROM, JOIN, WHERE, GROUP BY, and ORDER BY. To understand the syntax and what these words add to build better results when querying the database.

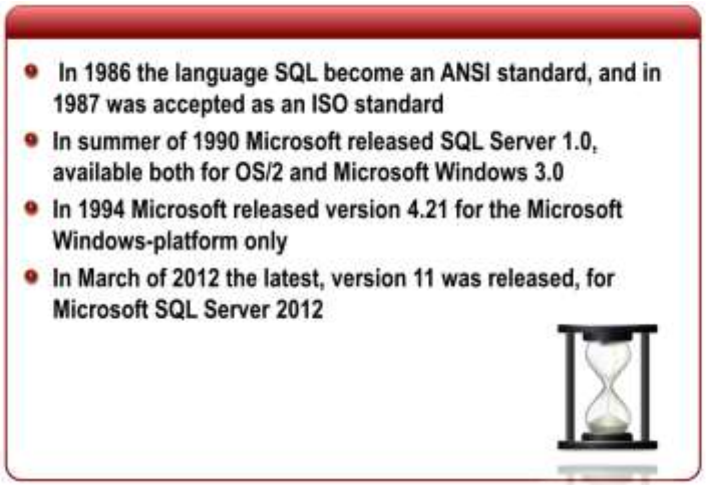
Even if the standards have minimal support for flow control, T-SQL offers additional scripting and batch processing flow control support such as IF..ELSE, WHILE.

### Objectives

After completing this lesson, you will be able to:

- Understand the history and standards of SQL/T-SQL
- Understand the types of SQL statements
- Use the basics of basic T-SQL syntax
- Understand T-SQL operators, functions, variables and expressions
- Handle flow-control in T-SQL
- Work with comments in T-SQL

## History and Standards of SQL / TSQL

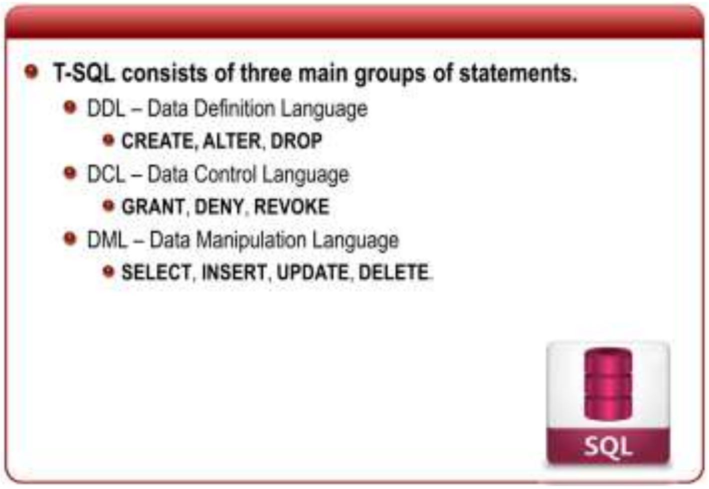


- In 1986 the language SQL become an ANSI standard, and in 1987 was accepted as an ISO standard
- In summer of 1990 Microsoft released SQL Server 1.0, available both for OS/2 and Microsoft Windows 3.0
- In 1994 Microsoft released version 4.21 for the Microsoft Windows-platform only
- In March of 2012 the latest, version 11 was released, for Microsoft SQL Server 2012

During the seventies the IBM research center in San Jose California created a research team named “System R” based on Edgar F. Codd’s article “A Relational Model of Data for Large Shared Data Banks”. This later evolved into IBM System/38 in August of 1978. In 1986 the language SQL become an ANSI standard, and in 1987 was accepted as an ISO standard. The latest revision of this standard is SQL:2003 which introduced XML-support, windowing functions, sequences, and columns with auto-generated values.

On March 27th 1987 Microsoft and Sybase agreed that Microsoft will be the sole developer and marketer of Sybase DataServer for OS/2 and other Microsoft-platforms. This leads us to the summer of 1990 and the Microsoft SQL Server 1.0, available both for OS/2 and Microsoft Windows 3.0 version 4.2, released in January of 1992, was the last version developed for both OS/2 and Microsoft Windows. In 1994 version 4.21 was released for the Microsoft Windows-platform only and in March of 2012 the latest, version 11 was released, for Microsoft SQL Server 2012.

## Types of SQL Statements

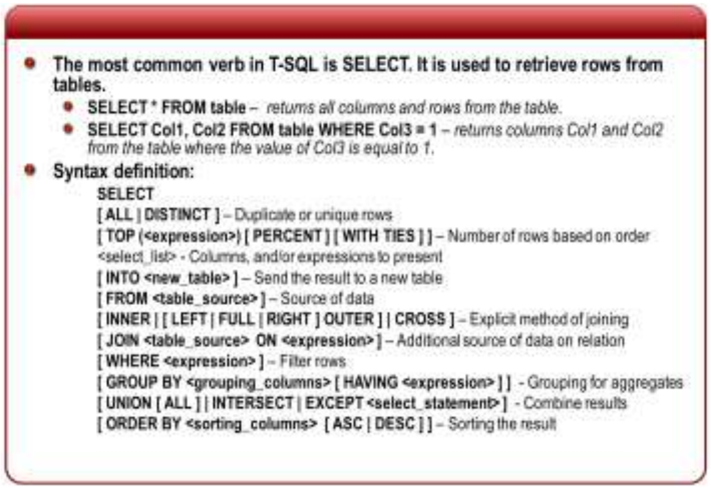


- **T-SQL consists of three main groups of statements.**
  - DDL – Data Definition Language
    - **CREATE, ALTER, DROP**
  - DCL – Data Control Language
    - **GRANT, DENY, REVOKE**
  - DML – Data Manipulation Language
    - **SELECT, INSERT, UPDATE, DELETE**

T-SQL consists of three main groups of statements.

- **DDL** – Data Definition Language
  - **CREATE**, to create database objects.
  - **ALTER**, to modify database objects.
  - **DROP**, to remove database objects.
- **DCL** – Data Control Language
  - **GRANT**, to grant users permission to objects.
  - **DENY**, to deny users permissions to objects.
  - **REVOKE**, to remove either grant or deny.
- **DML** – Data Manipulation Language
  - **SELECT**, to return rows of data from tables.
  - **INSERT**, to add rows of data to tables.
  - **UPDATE**, to modify rows of data in tables.
  - **DELETE**, to remove rows of data from tables.

## Elements of Basic T-SQL Syntax



- The most common verb in T-SQL is SELECT. It is used to retrieve rows from tables.
  - SELECT * FROM table – *returns all columns and rows from the table.*
  - SELECT Col1, Col2 FROM table WHERE Col3 = 1 – *returns columns Col1 and Col2 from the table where the value of Col3 is equal to 1.*
- Syntax definition:

```
SELECT
[ ALL | DISTINCT ] – Duplicate or unique rows
[ TOP (<expression>) [ PERCENT ] [ WITH TIES ] ] – Number of rows based on order
<select_list> - Columns, and/or expressions to present
[ INTO <new_table> ] – Send the result to a new table
[ FROM <table_source> ] – Source of data
[ INNER | [ LEFT | FULL | RIGHT ] OUTER ] | CROSS ] – Explicit method of joining
[ JOIN <table_source> ON <expression> ] – Additional source of data on relation
[ WHERE <expression> ] – Filter rows
[ GROUP BY <grouping_columns> [ HAVING <expression> ] ] - Grouping for aggregates
[ UNION [ ALL ] | INTERSECT | EXCEPT <select_statement> ] - Combine results
[ ORDER BY <sorting_columns> [ ASC | DESC ] ] – Sorting the result
```

The most common verb in T-SQL is SELECT. It is used to retrieve rows from tables.

```
      SELECT * FROM table;
```

returns all columns and rows from the table.

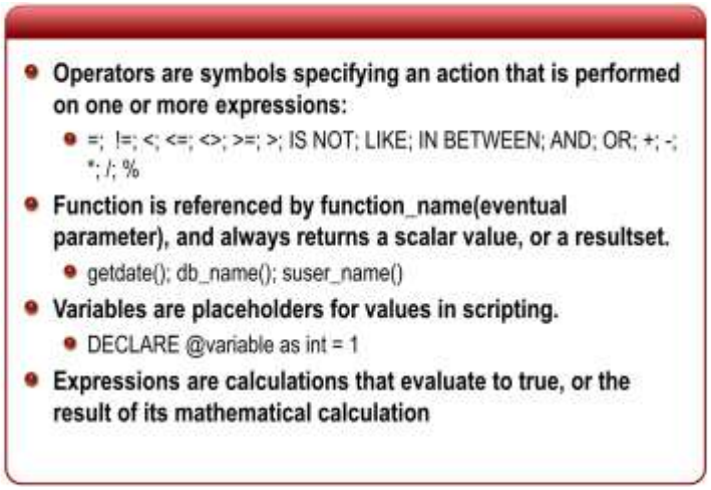
```
      SELECT Col1, Col2 FROM table WHERE Col3 = 1;
```

returns columns Col1 and Col2 from the table where the value of Col3 is equal to 1.

Syntax definition:

```
SELECT
[ ALL | DISTINCT ] – Duplicate or unique rows
[ TOP (<expression>) [ PERCENT ] [ WITH TIES ] ] – Number of rows based on order
<select_list> - Columns, and/or expressions to present
[ INTO <new_table> ] – Send the result to a new table
[ FROM <table_source> ] – Source of data
[ INNER | [ LEFT | FULL | RIGHT ] OUTER ] | CROSS ] – Explicit method of joining
[ JOIN <table_source>  ON <expression> ] – Additional source of data on relation
[ WHERE <expression> ] – Filter rows
[ GROUP BY <grouping_columns> [ HAVING <expression> ] ]  - Grouping for aggregates
[ UNION [ ALL ] | INTERSECT | EXCEPT <select_statement> ]  - Combine results
[ ORDER BY <sorting_columns>  [ ASC | DESC ] ] – Sorting the result
```

## T-SQL Operators, Functions, Variables, Expressions



- Operators are symbols specifying an action that is performed on one or more expressions:
  - =; !=; <; <=; <>; >=; >; IS NOT; LIKE; IN BETWEEN; AND; OR; +; -; *; /; %
- Function is referenced by function_name(eventual parameter), and always returns a scalar value, or a resultset.
  - getdate(); db_name(); suser_name()
- Variables are placeholders for values in scripting.
  - DECLARE @variable as int = 1
- Expressions are calculations that evaluate to true, or the result of its mathematical calculation

T-SQL Operators are symbols specifying an action that is performed on one or more expressions: =; !=; <; <=; <>; >=; >; IS NOT; LIKE; IN BETWEEN; AND; OR; +; -; *; /; %

There are numerous functions built into T-SQL. A function is always referenced by function_name(eventual parameter), and always returns a scalar value, or a resultset. The language supports a broad selection of functions, for example getdate(), db_name(), suser_name().

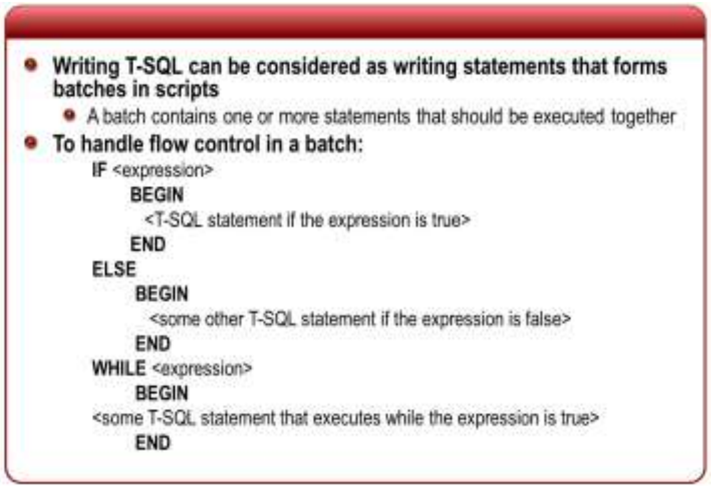
Variables are placeholders for values in scripting. They are given names to describe their use, and are declared with a data type. For example,

```
      DECLARE @variable as int = 1
```

They can later be changed with either a SET or a SELECT statement.

Expressions are calculations that evaluate to true, or the result of its mathematical calculation. Expressions are used in the select list of a query, or a filter in a WHERE-clause or a JOIN. They can also be used to populate a variable with its value in a SET statement.

## Flow Control in T-SQL



- Writing T-SQL can be considered as writing statements that forms batches in scripts
  - A batch contains one or more statements that should be executed together
- To handle flow control in a batch:

```
IF <expression>
     BEGIN
      <T-SQL statement if the expression is true>
     END
ELSE
      BEGIN
       <some other T-SQL statement if the expression is false>
      END
WHILE <expression>
      BEGIN
<some T-SQL statement that executes while the expression is true>
      END
```

Writing T-SQL is writing statements that forms batches in scripts. A statement can end with a semi-colon (;) but it is not mandatory. A batch contains one or more statements that should be executed together. The batch is then ended by the GO operator. One or more batches are the body of a T-SQL script.

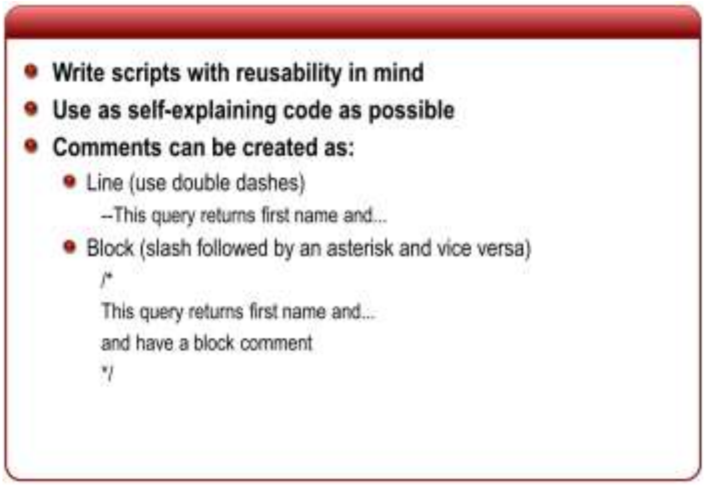
To handle flow control in a batch:

```
IF <expression>
        BEGIN
           <T-SQL statement if the expression is true>
        END
ELSE
         BEGIN
       <some other T-SQL statement if the expression is false>
         END
WHILE <expression>
         BEGIN
       <some T-SQL statement that executes while the
        expression is true>
         END
```

To handle in-row decision:

```
CASE
         WHEN <expression> THEN <value if true>
         ELSE <value if false>
END
```

## Comments in T-SQL


- Write scripts with reusability in mind
- Use as self-explaining code as possible
- Comments can be created as:
  - Line (use double dashes)
    --This query returns first name and...
  - Block (slash followed by an asterisk and vice versa)
    /*
    This query returns first name and...
    and have a block comment
    */

Always write scripts with reusability in mind. A best practice is to use as self-explaining code as possible as well as including comments in the scripts to serve as a refresher to you the author, as well as to help future readers understand the action result of the script. In T-SQL we handle comments either in line or as blocks. For line comments use double dashes that are two consecutive minus-characters

```
--This query returns first name and...
SELECT FirstName, ...
FROM Person.Person
--End of the query
```

A slash followed by an asterisk starts a block comment, also called a slash star comment. Here you can write multiple lines of comments as long as you don't have a single GO in a line, to comment out a GO do as follows.

```
--GO
```

To end the block comment just type an asterisk followed by a slash.

```
/*
This query returns first name and...
and have a block comment
*/
SELECT FirstName, ...
FROM Person.Person
```

## Lesson 4: Using SQL Server Tools

- Tools Overview
- SQL Server Management Studio
- Executing Queries

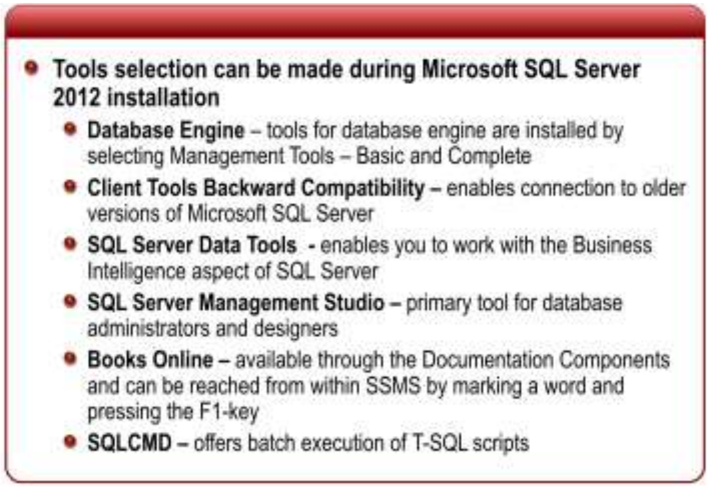
In order to write queries and work with Microsoft SQL Server 2012 and all of its features it is important to know which tools are in place, and their uses. There are both command line and GUI tools.

### Objectives

After completing this lesson, you will be able to:

- Deliver an overview of some of the tools Microsoft SQL Server 2012 offer
- Use SQL Server Management Studio to efficiently write queries
- Execute queries in both command line as well as in SQL Server Management Studio

## Tools Overview



- **Tools selection can be made during Microsoft SQL Server 2012 installation**
  - **Database Engine** – tools for database engine are installed by selecting Management Tools – Basic and Complete
  - **Client Tools Backward Compatibility** – enables connection to older versions of Microsoft SQL Server
  - **SQL Server Data Tools** - enables you to work with the Business Intelligence aspect of SQL Server
  - **SQL Server Management Studio** – primary tool for database administrators and designers
  - **Books Online** – available through the Documentation Components and can be reached from within SSMS by marking a word and pressing the F1-key
  - **SQLCMD** – offers batch execution of T-SQL scripts

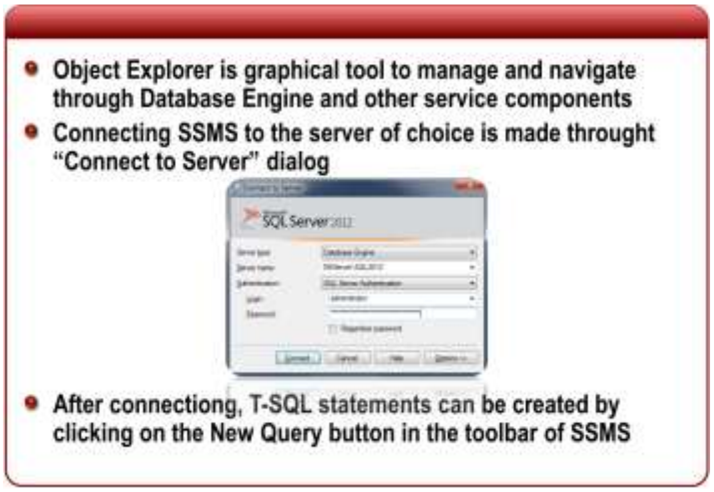
When installing Microsoft SQL Server 2012, the features selection gives you the ability to select which tools you need. Tools for the Database Engine are installed by selecting Management Tools – Basic and Complete, together with Client Tools Connectivity. Selecting Client Tools Backward Compatibility lets you to connect to older versions of SQL Server, and by choosing SQL Server Data Tools you also can work with the Business Intelligence aspect of SQL Server. All of these tools can also be installed on your client computer, and be run remotely against your SQL Server.

SQL Server Management Studio (SSMS), is the primary tool for both database administrators as well as database designers. It offers a graphical interface to writing queries, server administration, and create database objects. Almost every action that can be carried out via dialog boxes and windows can also be done using T-SQL scripts. T-SQL also offers an advanced T-SQL editor with support for IntelliSense and code snippets together with full blown templates and linking into the Books Online.

Books Online is available through the Documentation Components and can be reached from within SSMS by marking a word and pressing the F1-key. The documentation can also be downloaded for offline use.

The command line utility, SQLCMD, offers batch execution of T-SQL scripts, and a simple utility for executing T-SQL statements.

## SQL Server Management Studio



- Object Explorer is graphical tool to manage and navigate through Database Engine and other service components
- Connecting SSMS to the server of choice is made throught “Connect to Server” dialog
- After connectiong, T-SQL statements can be created by clicking on the New Query button in the toolbar of SSMS

When starting SSMS, you will be presented with a “Connect to Server” dialog where you can chose Server type, enter a Server name, and select an Authentication type. After selection of previously stated options you are able to connect the Object Explorer to the server of choice.

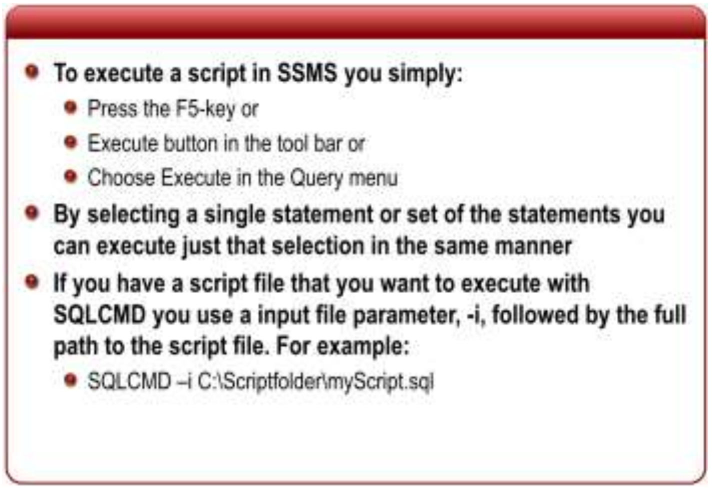
The Object Explorer is your graphical tool to manage and navigate through your Microsoft SQL Server 2012 Database Engine as well as the other service components of the product. From this pane you can reach and configure most of the objects and properties of the server and its databases. It supports context menus that offer scripting functionality, tools, reports, properties and much more.

New T-SQL statements in batches and scripts can be created by clicking on the New Query button in the toolbar, or through File menu, New/Database Engine Query. These scripts are then savable in the file system as script files with a SSMS .sql extension.

There are more views in SSMS such as Registered Servers; which makes it possible to save connections to server objects and to organize them into groups. From this view, you can execute multi server queries as well as view an Error List to help you troubleshoot scripts.

As SSMS is based upon Visual Studio and therefore you also have the ability to work with your scripts in Solutions and Projects, and tie them to a shared source code solution. Through source code sharing platforms such Team Foundation Services, you can share scripts with teams of developers and be able to take advantage of collaboration support such as checking out and in parts of the whole solution.

## Executing Queries



- To execute a script in SSMS you simply:
  - Press the F5-key or
  - Execute button in the tool bar or
  - Choose Execute in the Query menu
- By selecting a single statement or set of the statements you can execute just that selection in the same manner
- If you have a script file that you want to execute with SQLCMD you use a input file parameter, -i, followed by the full path to the script file. For example:
  - SQLCMD –i C:\Scriptfolder\myScript.sql

To execute a script in SSMS you simply press the F5-key or the Execute button in the tool bar, or by choosing Execute in the Query menu. This executes the whole script.

By selecting a single statement, or some of the statements, in a script you can execute just that selection in the same manner.

To execute a batch in SQLCMD you simply type GO after the last statement and then press enter. If you have a script file that you want to execute with SQLCMD you use a input file parameter, -i, followed by the full path to the script file. For example:

```
      SQLCMD –i C:\Scriptfolder\myScript.sql
```

## Summary

**In this module, you learned:**

- How SQL Server 2012 works
- About basic Database Concepts
- How to use T-SQL Syntax Elements
- How to use the SQL Server Tools

In this module you learned how to write queries and work with Microsoft SQL Server 2012 and how the software works. You also learned about database concepts such as relational databases, its objects and data types. You also have an understanding of T-SQL syntax elements, and what tools to use when writing queries.

### Objectives

After completing this module, you learned:

- How SQL Server 2012 works
- About basic Database Concepts
- How to use T-SQL Syntax Elements
- How to use the SQL Server Tools