Trigger

Triggers are special kind of stored procedure. They can be executed after or before data modification happens on a table. There are two types of triggers "Instead of triggers" and "After triggers".

"Instead of triggers" executes prior to data modification while "after trigger" executes after data modification.

In what scenarios will you use instead of trigger and after trigger?

You will use "INSTEAD trigger" to take alternative actions before the update happens.

Some of the uses of instead of trigger's are:-

1. Reject updates which are not valid.
2. Take some alternative action if any error occurs.
3. To implement cascading deletes. For instance you want to delete a customer record. But in order to delete the customer record you also have to delete address records. So you can create a instead of trigger which will first delete the address table before executing delete on customer table.

While "AFTER trigger" is useful when you want to execute trigger logic after the data has been updated.

Some uses of after triggers are:-

1. For recording Audit trail where you want new and old values to be inserted in to audit table.
2. Updating values after the update has happened. For instance in a sales table if number of products and per product is inserted, you can create an after trigger which will calculate the total sales amount using these two values.

Magic Tables

Magic tables are internal tables called ‘inserted’ and ‘deleted’ which are created and managed by SQL server.

‘**Inserted**‘table holds the values that have been recently inserted.
‘**Inserted**‘table also holds the updated values from an update operation on a table.

‘**Deleted**‘table holds the values that have been recently deleted.
‘**Deleted**‘table also holds the previous values from an update operation on a table.

These virtual tables are usually used with triggers to retrieve the inserted, deleted or updated rows.

Classes in C#

1. Abstract Class (sometimes called a Pure Virtual Class)
2.Partial Class
3. Sealed Class
4.Static Class

Abstract Class:

An Abstract Class means that, no object of this class can be instantiated, but can make derivation of this. It can serve the purpose of base class only as no object of this class can be created.
Abstract Class is denoted by the keyword *abstract*.

Example:

```
abstract class myClass
{
        public myClass()
{
                // code to initialize the class…
}
        abstract public void anyMethod_01();
        abstract public void anyMethod_02(int anyVariable);
        abstract public int anyMethod_03 (int anyvariable);
}
```

It is important here to note that abstract classes can have non-abstract method(s), even can have only non-abstract method(s).

Partial Class:

This special type of class called "Partial Class" is introduced with .Net Framework 2.0. Partial Class allows its members – method, properties, and events – to be divided into multiple source files (.cs). At compile time these files get combined into a single class.
Partial Class is denoted by the keyword partial.
Some do's and don'ts about partial class:-
• All the parts of a partial class must be prefixed with the keyword partial.
• Accessibility, signature etc. must be same in all parts of the partial class.
• You cannot sealed one part of the partial class. In that case entire class in sealed.
• If you define any part of the partial class abstract, entire class will become abstract.
• Inheritance cannot be applied to a part of partial class. If you do so, it applies to entire class.

Example:

```
public partial class myPartialClass
{
        public void firstMethod()
        {
                // code…
        }
}
```

```
public partial class myPartialClass
{
	public void secondMethod()
	{
		// code…
	}
}
```

Sealed Class:

A sealed class is a class which cannot be inherited. A sealed class cannot be a base class. The modifier abstract cannot be applied to a sealed class. By default, struct (structure) is sealed. It is the last class in hierarchy. To access the members of a sealed class, you must create objects of that class.
Sealed Class is denoted by the keyword *sealed*.

Example:

```
sealed class mySealedClass
{
	int a;
	int b;
}
Class mainClass
{
	public static void Main()
	{
		mySealedClass obj = new mySealedClass();
		obj.a = 5;
		obj.b = 7;
		Console.WriteLine("a = {0}, b = {1}", obj.a, obj.b);
	}
}
```

Static Class:

A Static Class is one which cannot be instantiated. The keyword new cannot be used with static classes as members of such class can be called directly by using the class name itself.
Following are the main characteristics of a static class:-

• A Static Class can only have static members.
• A Static Class cannot be instantiated.
• A Static Class is sealed, so cannot be inherited.
• A Static Class cannot have a constructor (except static constructor).

Static Class is denoted by the keyword *static*.

Example:

```
// static class definition…
public static class myclass
{
	public static int addNumbers(int a, int b)
	{
		return (a + b);
	}
}

// to use it, we call directly on the class…
Console.WriteLine("The addition of 5 and 7 is: " + myClass.addNumbers(5, 7));
```

Index in SQL

Why do we use indexes:

To get the high performence index are used in sql server database.Indexes can use to get the information quickly,Like we used indexes in books to get information quickly.Indexes are created on columns in tables or view.The index provides a fast way to look up data in table and database.Index creation in table helps us to query optimization and faster access of data.

Types of Indexes:

There are different types of indexes you can create in sql server database.The indexes are given below:

1:Clustered
2:Nonclustered
3:Unique
4:Index with included columns
5:Indexed views

1:Clustered Index:

A clustered index sorts and stores the data rows of the table or view in order based on the clustered index key.We can have only 1 clustered index in a table.Clustered index is applied on primary key.In case of clustered index , indexed values are sorted in either ascending or descending order.

2: Nonclustered Index:

A nonclustered index can be defined on a table or view with a clustered index or on a heap. Each index row in the nonclustered index contains the nonclustered key value and a row locator.We can have max 249 Nonclustered index in a table.

3:Unique Index:

A unique index ensures that the index key contains no duplicate values and therefore every row in the table or view is in some way unique.
Both clustered and nonclustered indexes can be unique.

4:Index with included columns:

A nonclustered index that is extended to include nonkey columns in addition to the key columns.

5:Indexed views:

A index in view is called Indexed view.We can add index on view.The view and result set is permanently stored in a unique clustered index in the same way a table with a clustered index is stored. Nonclustered indexes on the view can be added after the clustered index is created.

Page Life Cycle

PreInt: This is the entry point of the ASP.NET page life cycle - it is the pre-initialization, so you have access to the page before it is initialized. Controls can be created within this event. Also, master pages and themes can be accessed. You can check the IsPostBack property here to determine if it is the first time a page has been loaded.

Init: This event fires when all controls on the page have been initialized and skin settings have been applied. You can use this event to work with control properties. The Init event of the page is not fired until all control Init events have triggered - this occurs from the bottom up.

InitComplete: This event fires once all page and control initializations complete. This is the last event fired where ViewState is not set, so ViewState can be manipulated in this event.

PreLoad: This event is triggered when all ViewState and Postback data have been loaded for the page and all of its controls - ViewState loads first, followed by Postback data.

Load: This is the first event in the page life cycle where everything is loaded and has been set to its previous state (in the case of a postback). The page Load event occurs first followed by the Load event for all controls (recursively). This is where most coding is done, so you want to check the IsPostBack property to avoid unnecessary work.

LoadComplete: This event is fired when the page is completely loaded. Place code here that requires everything on the page to be loaded.

PreRender: This is the final stop in the page load cycle where you can make changes to page contents or controls. It is fired after all PostBack events and before ViewState has been saved. Also, this is where control databinding occurs.

PreRenderComplete: This event is fired when PreRender is complete. Each control raises this event after databinding (when a control has its DataSourceID set).

SaveStateComplete: This is triggered when view and control state have been saved for the page and all controls within it. At this point, you can make changes in the rendering of the page, but those changes will not be reflected on the next page postback since view state is already saved.

Unload: This event fires for each control and then the page itself. It is fired when the HTML for the page is fully rendered. This is where you can take care of cleanup tasks, such as properly closing and disposing database connections.

Difference b/w Function and Stroed Procedure

1. Function must return a value but in Stored Procedure it is optional(Procedure can return zero or n values).
2. Functions can have only input parameters for it whereas Procedures can have input/output parameters.
3. Function takes one input parameter it is mandatory but Stored Procedure may take o to n input parameters.
4. Functions can be called from Procedure whereas Procedures cannot be called from Function.
5. Procedure allows SELECT as well as DML(INSERT/UPDATE/DELETE) statement in it whereas Function allows only SELECT statement in it.
6. Procedures cannot be utilized in a SELECT statement whereas Function can be embedded in a SELECT statement.
7. Exception can be handled by try-catch block in a Procedure whereas try-catch block cannot be used in a Function.
8. Stored Procedures cannot be used in the SQL statements anywhere in the WHERE/HAVING/SELECT section whereas Function can be.

What are the different levels at which events can occur in an ASP.NET web application?
Web application events occur at the application, page, and server control levels

Give some examples for application level events?
1. Application_Start
Occurs when the first user visits a page within your Web application.

2. Application_End
Occurs when there are no more users of the application.

3. Application_BeginRequest
Occurs when at the beginning of each request to the server. A request happens every time a browser navigates to any of the pages in the application.

4. Application_EndRequest
Occurs when at the end of each request to the server.

5. Session_Start
Occurs when a new user visits a page within your application.

6. Session_End
Occurs when a user stops requesting pages from the Web application and their session times out. Sessions time out after a period specified in the Web.config file.

7. Application_Error
Occurs when when there is an unhandled exception in an application.

Common Language Runtime (CLR)

.Net Framework provides runtime environment called Common Language Runtime (CLR).It provides an environment to run all the .Net Programs. The code which runs under the CLR is called as Managed Code. Programmers need not to worry on managing the memory if the programs are running under the CLR as it provides memory management and thread management.
Programmatically, when our program needs memory, CLR allocates the memory for scope and de-allocates the memory if the scope is completed.
Language Compilers (e.g. C#, VB.Net, J#) will convert the Code/Program to Microsoft Intermediate Language (MSIL) intern this will be converted to Native Code by CLR.

.Net Framework Class Library (FCL)

This is also called as Base Class Library and it is common for all types of applications i.e. the way you access the Library Classes and Methods in VB.NET will be the same in C#, and it is common for all other languages in .NET.

The following are different types of applications that can make use of .net class library.

1. Windows Application.
2. Console Application
3. Web Application.
4. XML Web Services.
5. Windows Services.

In short, developers just need to import the BCL in their language code and use its predefined methods and properties to implement common and complex functions like reading and writing to file, graphic rendering, database interaction, and XML document manipulation.

Common Type System (CTS)

It describes set of data types that can be used in different .Net languages in common. (i.e), CTS ensures that objects written in different .Net languages can interact with each other.
For Communicating between programs written in any .NET complaint language, the types have to be compatible on the basic level.
The common type system supports two general categories of types:

Value types:

Value types directly contain their data, and instances of value types are either allocated on the stack or allocated inline in a structure. Value types can be built-in (implemented by the runtime), user-defined, or enumerations.

Reference types:

Reference types store a reference to the value's memory address, and are allocated on the heap. Reference types can be self-describing types, pointer types, or interface types. The type of a reference type can be determined from values of self-describing types. Self-describing types are further split into arrays and class types. The class types are user-defined classes, boxed value types, and delegates.

Common Language Specification (CLS)

It is a sub set of CTS and it specifies a set of rules that needs to be adhered or satisfied by all language compilers targeting CLR. It helps in cross language inheritance and cross language debugging.

Common language specification Rules:

It describes the minimal and complete set of features to produce code that can be hosted by CLR. It ensures that products of compilers will work properly in .NET environment.
Sample Rules:

1. Representation of text strings
2. Internal representation of enumerations
3. Definition of static members and this is a subset of the CTS which all .NET languages are expected to support.
4. Microsoft has defined CLS which are nothing but guidelines that language to follow so that it can communicate with other .NET languages in a seamless manner.

.Net Assembly?

The .NET assembly is the standard for components developed with the Microsoft.NET. Dot NET assemblies may or may not be executable, i.e., they might exist as the executable (.exe) file or dynamic link library (DLL) file. All the .NET assemblies contain the definition of types, versioning information for the type, meta-data, and manifest. The designers of .NET have worked a lot on the component (assembly) resolution.
There are two kind of assemblies in .NET;

Private assemblies are simple and copied with each calling assemblies in the calling assemblies folder.

Shared assemblies (also called strong named assemblies) are copied to a single location (usually the Global assembly cache). For all calling assemblies within the same application, the same copy of the shared assembly is used from its original location. Hence, shared assemblies are not copied in the private folders of each calling assembly. Each shared assembly has a four part name including its face name, version, public key token and culture information. The public key token and version information makes it almost impossible for two different assemblies with the same name or for two similar assemblies with different version to mix with each other.

An assembly can be a single file or it may consist of the multiple files. In case of multi-file, there is one master module containing the manifest while other assemblies exist as non-manifest modules. A module in .NET is a sub part of a multi-file .NET assembly. Assembly is one of the most interesting and extremely useful areas of .NET architecture along with reflections and attributes, but unfortunately very few people take interest in learning such theoretical looking topics.

Session State

ASP.NET allows you to save values using session state, a storage mechanism that is accessible from all pages requested by a single Web browser session.

ASP.NET session state supports several different storage options for session data:

A. **InProc** Stores session state in memory on the Web server. This is the default, and it offers much better performance than using the ASP.NET state service or storing state information in a database server. InProc is fine for simple applications, but robust applications that use multiple Web servers or must persist session data between application restarts should use State Server or SQLServer.

B. **StateServer** Stores session state in a service called the ASP.NET State Service. This ensures that session state is preserved if the Web application is restarted and also makes session state available to multiple Web servers in a Web farm. ASP.NET State Service is included with any computer set up to run ASP.NET Web applications; however, the service is set up to start manually by default. Therefore, when configuring the ASP.NET State Service, you must set the startup type to Automatic.

C. **SQLServer** Stores session state in a SQL Server database. This ensures that session state is preserved if the Web application is restarted and also makes session state available to multiple Web servers in a Web farm. On the same hardware, the ASP.NET State Service outperforms SQLServer. However, a SQL Server database offers more robust data integrity and reporting capabilities.

Differentiate between a Local and a Global temporary table?

- A local temporary table exists only for the duration of a connection or, if defined inside a compound statement, for the duration of the compound statement.

- Global temporary tables (created with a double "##") are visible to all sessions.
- Global temporary tables are dropped when the session that created it ends, and all other sessions have stopped referencing it.

What is a View?

A simple view can be thought of as a subset of a table. It can be used for retrieving data as well as updating or deleting rows. Rows updated or deleted in the view are updated or deleted in the table the view was created with. It should also be noted that as data in the original table changes, so does the data in the view as views are the way to look at parts of the original table. The results of using a view are not permanently stored in the database. The data accessed through a view is actually constructed using standard T-SQL select command and can come from one to many different base tables or even other views.

What is an Index?

An index is a physical structure containing pointers to the data. Indices are created in an existing table to locate rows more quickly and efficiently. It is possible to create an index on one or more columns of a table, and each index is given a name. The users cannot see the indexes; they are just used to speed up queries. Effective indexes are one of the best ways to improve performance in a database application. A table scan happens when there is no index available to help a query. In a table scan, the SQL Server examines every row in the table to satisfy the query results. Table scans are sometimes unavoidable, but on large tables, scans have a terrific impact on performance.

What is Collation?

Collation refers to a set of rules that determine how data is sorted and compared. Character data is sorted using rules that define the correct character sequence with options for specifying case sensitivity, accent marks, Kana character types, and character width. Just a day before one of our SQL SERVER 2005 needed Case-Sensitive Binary Collation. When we install SQL SERVER 2005 it gives options to select one of the many collation. I says in words like 'Dictionary order, case-insensitive, uppercase preference'. I was confused for little while as I am used to read collation like 'SQL_Latin1_General_Pref_Cp1_CI_AS_KI_WI'. I did some research and find following link which explains many of the SQL SERVER 2005 collation.

What is the Difference between a Function and a Stored Procedure?

UDF can be used in the SQL statements anywhere in the WHERE/HAVING/SELECT section, whereas Stored procedures cannot be. UDFs that return tables can be treated as another rowset. This can be used in JOINs with other tables. Inline UDF's can be thought of as views that take parameters and can be used in JOINs and other Rowset operations.

What are Different Types of Join?

Cross Join

A cross join that does not have a WHERE clause produces the Cartesian product of the tables involved in the join. The size of a Cartesian product result set is the number of rows in the first table multiplied by the number of rows in the second table. The common example is when company wants to combine each product with a pricing table to analyze each product at each price.

Inner Join

A join that displays only the rows that have a match in both joined tables is known as inner Join. This is the default type of join in the Query and View Designer.

Outer Join

A join that includes rows even if they do not have related rows in the joined table is an Outer Join. You can create three different outer join to specify the unmatched rows to be included:

Left Outer Join: In Left Outer Join, all the rows in the first-named table, i.e. "left" table, which appears leftmost in the JOIN clause, are included. Unmatched rows in the right table do not appear.

Right Outer Join: In Right Outer Join, all the rows in the second-named table, i.e. “right” table, which appears rightmost in the JOIN clause are included. Unmatched rows in the left table are not included.

Full Outer Join: In Full Outer Join, all the rows in all joined tables are included, whether they are matched or not.

Self Join

This is a particular case when one table joins to itself with one or two aliases to avoid confusion. A self join can be of any type, as long as the joined tables are the same. A self join is rather unique in that it involves a relationship with only one table. The common example is when company has a hierarchal reporting structure whereby one member of staff reports to another. Self Join can be Outer Join or Inner Join.

What are Primary Keys and Foreign Keys?

Primary keys are the unique identifiers for each row. They must contain unique values and cannot be null. Due to their importance in relational databases, Primary keys are the most fundamental aspect of all keys and constraints. A table can have only one primary key.
Foreign keys are a method of ensuring data integrity and manifestation of the relationship between tables.

What are the Difference between Clustered and a Non-clustered Index?

A clustered index is a special type of index that reorders the way records in the table are physically stored. Therefore, the table can have only one clustered index. The leaf nodes of a clustered index contain the data pages.
A non-clustered index is a special type of index in which the logical order of the index does not match the physical stored order of the rows on disk. The leaf node of a non-clustered index does not consist of the data pages. Instead, the leaf nodes contain index rows.

What’s the Difference between a Primary Key and a Unique Key?

Both primary key and unique key enforce uniqueness of the column on which they are defined. But by default, the primary key creates a clustered index on the column, whereas unique key creates a non-clustered index by default. Another major difference is that primary key doesn’t allow NULLs, but unique key allows one NULL only.

What is Difference between DELETE and TRUNCATE Commands?

Delete command removes the rows from a table on the basis of the condition that we provide with a WHERE clause. Truncate will actually remove all the rows from a table, and there will be no data in the table after we run the truncate command.

TRUNCATE

1. TRUNCATE is faster and uses fewer system and transaction log resources than DELETE. (Read all the points below)

2. TRUNCATE removes the data by deallocating the data pages used to store the table's data, and only the page deallocations are recorded in the transaction log.
3. TRUNCATE removes all the rows from a table, but the table structure, its columns, constraints, indexes and so on remains. The counter used by an identity for new rows is reset to the seed for the column.
4. You cannot use TRUNCATE TABLE on a table referenced by a FOREIGN KEY constraint.
5. Using T-SQL – TRUNCATE cannot be rolled back unless it is used in TRANSACTION. OR TRUNCATE can be rolled back when used with BEGIN … END TRANSACTION using T-SQL.
6. TRUNCATE is a DDL Command.
7. TRUNCATE resets the identity of the table.

DELETE

1. DELETE removes rows one at a time and records an entry in the transaction log for each deleted row.
2. DELETE does not reset Identity property of the table.
3. DELETE can be used with or without a WHERE clause
4. DELETE activates Triggers if defined on table.
5. DELETE can be rolled back.
6. DELETE is DML Command.
7. DELETE does not reset the identity of the table.

What is the Difference between a HAVING clause and a WHERE clause?

They specify a search condition for a group or an aggregate. But the difference is that HAVING can be used only with the SELECT statement. HAVING is typically used in a GROUP BY clause. When GROUP BY is not used, HAVING behaves like a WHERE clause. Having Clause is basically used only with the GROUP BY function in a query, whereas WHERE Clause is applied to each row before they are part of the GROUP BY function in a query.