Contents

[Part 69 - Merge in SQL Server 1](#_Toc443855527)

[70. sql server concurrent transactions 4](#_Toc443855528)

[71. sql server dirty read example 6](#_Toc443855529)

[72. sql server lost update problem 8](#_Toc443855530)

[73. Non repeatable read example in sql server 11](#_Toc443855531)

[74. Phantom reads example in sql server 13](#_Toc443855532)

[75. Snapshot isolation level in sql server 16](#_Toc443855533)

[76. Read committed snapshot isolation level in sql server 18](#_Toc443855534)

[77. Difference between snapshot isolation and read committed snapshot 20](#_Toc443855535)

[78. SQL Server deadlock example 24](#_Toc443855536)

[79. SQL Server deadlock victim selection 27](#_Toc443855537)

[80. Logging deadlocks in sql server 31](#_Toc443855538)

[81. SQL Server deadlock analysis and prevention 33](#_Toc443855539)

[82. Capturing deadlocks in sql profiler 34](#_Toc443855540)

[83. SQL Server deadlock error handling 37](#_Toc443855541)

[84. Handling deadlocks in ado.net 38](#_Toc443855542)

[85. Retry logic for deadlock exceptions 43](#_Toc443855543)

[86. How to find blocking queries in sql server 47](#_Toc443855544)

[87. SQL Server except operator 49](#_Toc443855545)

[88. Difference between except and not in sql server 52](#_Toc443855546)

[89. Intersect operator in sql server 54](#_Toc443855547)

[90. Difference between union intersect and except in sql server 57](#_Toc443855548)

[91. Cross apply and outer apply in sql server 61](#_Toc443855549)

[92. DDL Triggers in sql server 64](#_Toc443855550)

[93. Server-scoped ddl triggers 66](#_Toc443855551)

[94. sql server trigger execution order 68](#_Toc443855552)

[95. Audit table changes in sql server 70](#_Toc443855553)

[96. Logon triggers in sql server 72](#_Toc443855554)

[97. Select into in sql server 74](#_Toc443855555)

[98. Difference between where and having in sql server 76](#_Toc443855556)

[99. Table valued parameters in SQL Server 79](#_Toc443855557)

[100. Send datatable as parameter to stored procedure 81](#_Toc443855558)

[101. Grouping Sets in SQL Server 87](#_Toc443855559)

[102. Rollup in SQL Server 93](#_Toc443855560)

[103. Cube in SQL Server 97](#_Toc443855561)

[104. Difference between cube and rollup in SQL Server 99](#_Toc443855562)

[105. Grouping function in SQL Server 104](#_Toc443855563)

[106. GROUPING\_ID function in SQL Server 108](#_Toc443855564)

[107. Debugging sql server stored procedures 111](#_Toc443855565)

[108. Over clause in SQL Server 118](#_Toc443855566)

[109. Row\_Number function in SQL Server 121](#_Toc443855567)

[110. Rank and Dense\_Rank in SQL Server 123](#_Toc443855568)

[111. Difference between rank dense\_rank and row\_number in SQL 126](#_Toc443855569)

[112. Calculate running total in SQL Server 2012 129](#_Toc443855570)

[113. NTILE function in SQL Server 131](#_Toc443855571)

[114. Lead and Lag functions in SQL Server 2012 134](#_Toc443855572)

[115. FIRST\_VALUE function in SQL Server 137](#_Toc443855573)

[116. Window functions in SQL Server 138](#_Toc443855574)

[117. Difference between rows and range 141](#_Toc443855575)

[118. LAST\_VALUE function in SQL Server 145](#_Toc443855576)

[119. UNPIVOT in SQL Server 147](#_Toc443855577)

[120. Reverse PIVOT table in SQL Server 148](#_Toc443855578)

[121. Choose function in SQL Server 152](#_Toc443855579)

[122. IIF function in SQL Server 154](#_Toc443855580)

[123. TRY\_PARSE function in SQL Server 2012 156](#_Toc443855581)

[124. TRY\_CONVERT function in SQL Server 2012 158](#_Toc443855582)

[125. EOMONTH function in SQL Server 2012 161](#_Toc443855583)

[126. DATEFROMPARTS function in SQL Server 164](#_Toc443855584)

[127. Difference between DateTime and SmallDateTime in SQL Server 165](#_Toc443855585)

[128. DateTime2FromParts function in SQL Server 2012 166](#_Toc443855586)

[129. Difference between DateTime and DateTime2 in SQL Server 167](#_Toc443855587)

[130. Offset fetch next in SQL Server 2012 170](#_Toc443855588)

[131. Identifying object dependencies in SQL Server 173](#_Toc443855589)

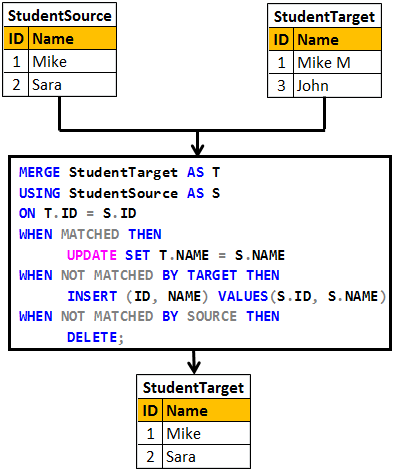
[132. sys.dm\_sql\_referencing\_entities in SQL Server 176](#_Toc443855590)

[133. sp\_depends in SQL Server 177](#_Toc443855591)

[134. Sequence object in SQL Server 2012 178](#_Toc443855592)

[135. Difference between sequence and identity in SQL Server 182](#_Toc443855593)

# Part 69 - Merge in SQL Server

**Suggested Videos**  
[Part 66 - Writing re-runnable sql server scripts](http://csharp-video-tutorials.blogspot.com/2013/06/writing-re-runnable-sql-server-scripts.html)  
[Part 67 - Alter database table columns without dropping table](http://csharp-video-tutorials.blogspot.com/2013/08/part-67-alter-database-table-columns.html)   
[Part 68 - Optional parameters in sql server stored procedures](http://csharp-video-tutorials.blogspot.com/2013/09/part-68-optional-parameters-in-sql.html)   
  
  
  
**What is the use of MERGE statement in SQL Server**  
Merge statement introduced in SQL Server 2008 allows us to perform Inserts, Updates and Deletes in one statement. This means we no longer have to use multiple statements for performing Insert, Update and Delete.   
  
**With merge statement we require 2 tables**  
1. Source Table - Contains the changes that needs to be applied to the target table  
2. Target Table - The table that require changes (Inserts, Updates and Deletes)  
  
  
  
The merge statement joins the target table to the source table by using a common column in both the tables. Based on how the rows match up as a result of the join, we can then perform insert, update, and delete on the target table.   
  
**Merge statement syntax**  
**MERGE [TARGET] AS T**  
**USING [SOURCE] AS S**  
**ON [JOIN\_CONDITIONS]**  
**WHEN MATCHED THEN**  
**[UPDATE STATEMENT]**  
**WHEN NOT MATCHED BY TARGET THEN**  
**[INSERT STATEMENT]**  
**WHEN NOT MATCHED BY SOURCE THEN**  
**[DELETE STATEMENT]**  
  
**Example 1 :** In the example below, INSERT, UPDATE and DELETE are all performed in one statement  
**1.**When matching rows are found, StudentTarget table is UPDATED (i.e WHEN MATCHED)  
  
**2.** When the rows are present in StudentSource table but not in StudentTarget table those rows are INSERTED into StudentTarget table (i.e WHEN NOT MATCHED BY TARGET)  
  
**3.** When the rows are present in StudentTarget table but not in StudentSource table those rows are DELETED from StudentTarget table (i.e WHEN NOT MATCHED BY SOURCE)   
  


Create table StudentSource

(

     ID int primary key,

     Name nvarchar(20)

)

GO

Insert into StudentSource values (1, 'Mike')

Insert into StudentSource values (2, 'Sara')

GO

Create table StudentTarget

(

     ID int primary key,

     Name nvarchar(20)

)

GO

Insert into StudentTarget values (1, 'Mike M')

Insert into StudentTarget values (3, 'John')

GO

MERGE StudentTarget AS T

USING StudentSource AS S

ON T.ID = S.ID

WHEN MATCHED THEN

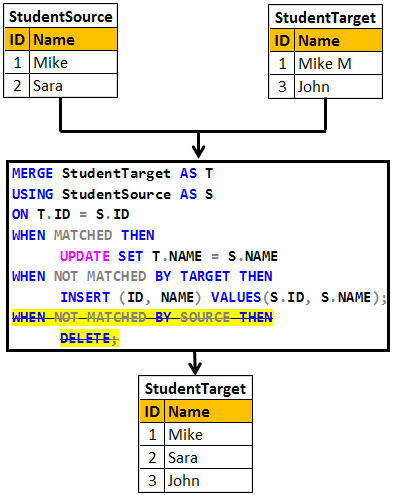
     UPDATE SET T.NAME = S.NAME

WHEN NOT MATCHED BY TARGET THEN

     INSERT (ID, NAME) VALUES(S.ID, S.NAME)

WHEN NOT MATCHED BY SOURCE THEN

     DELETE;

**Please Note :** Merge statement should end with a semicolon, otherwise you would get an error stating - A MERGE statement must be terminated by a semi-colon (;)  
  
**In real time we mostly perform INSERTS and UPDATES.**The rows that are present in target table but not in source table are usually not deleted from the target table.  
  
**Example 2 :** In the example below, only INSERT and UPDATE is performed. We are not deleting the rows that are present in the target table but not in the source table.   
  


Truncate table StudentSource

Truncate table StudentTarget

GO

Insert into StudentSource values (1, 'Mike')

Insert into StudentSource values (2, 'Sara')

GO

Insert into StudentTarget values (1, 'Mike M')

Insert into StudentTarget values (3, 'John')

GO

MERGE StudentTarget AS T

USING StudentSource AS S

ON T.ID = S.ID

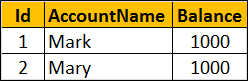
WHEN MATCHED THEN

     UPDATE SET T.NAME = S.NAME

WHEN NOT MATCHED BY TARGET THEN

     INSERT (ID, NAME) VALUES(S.ID, S.NAME);

# 70. sql server concurrent transactions

**Suggested Videos**  
[Part 67 - Alter database table columns without dropping table](http://csharp-video-tutorials.blogspot.com/2013/08/part-67-alter-database-table-columns.html)  
[Part 68 - Optional parameters in sql server stored procedures](http://csharp-video-tutorials.blogspot.com/2013/09/part-68-optional-parameters-in-sql.html)   
[Part 69 - Merge in SQL Server](http://csharp-video-tutorials.blogspot.com/2014/09/part-69-merge-in-sql-server.html)   
  
  
  
**In this video we will discuss**  
1. What a transaction is  
2. The problems that might arise when tarnsactions are run concurrently  
3. The different transaction isolation levels provided by SQL Server to address concurrency side effects   
  
  
  
**What is a transaction**  
A transaction is a group of commands that change the data stored in a database. A transaction, is treated as a single unit of work. A transaction ensures that, either all of the commands succeed, or none of them. If one of the commands in the transaction fails, all of the commands fail, and any data that was modified in the database is rolled back. In this way, transactions maintain the integrity of data in a database.   
  
   
  
**Example :** The following transaction ensures that both the UPDATE statements succeed or both of them fail if there is a problem with one UPDATE statement.

-- Transfer $100 from Mark to Mary Account

BEGIN TRY

    BEGIN TRANSACTION

         UPDATE Accounts SET Balance = Balance - 100 WHERE Id = 1

         UPDATE Accounts SET Balance = Balance + 100 WHERE Id = 2

    COMMIT TRANSACTION

    PRINT 'Transaction Committed'

END TRY

BEGIN CATCH

    ROLLBACK TRANSACTION

    PRINT 'Transaction Rolled back'

END CATCH

Databases are powerful systems and are potentially used by many users or applications at the same time. Allowing concurrent transactions is essential for performance but may introduce concurrency issues when two or more transactions are working with the same data at the same time.  
  
**Some of the common concurrency problems**

* Dirty Reads
* Lost Updates
* Nonrepeatable Reads
* Phantom Reads

We will discuss what these problems are in detail with examples in our upcomning videos  
  
One way to solve all these concurrency problems is by allowing only one user to execute, only one transaction at any point in time. Imagine what could happen if you have a large database with several users who want to execute several transactions. All the transactions get queued and they may have to wait a long time before they could get a chance to execute their transactions. So you are getting poor performance and the whole purpose of having a powerful database system is defeated if you serialize access this way.   
  
At this point you might be thinking, for best performance let us allow all transactions to execute concurrently. The problem with this approach is that it may cause all sorts of concurrency problems (i.e Dirty Reads, Lost Updates, Nonrepeatable Reads, Phantom Reads) if two or more transactions work with the same data at the same time.  
  
SQL Server provides different **transaction isolation levels**, to balance concurrency problems and performance depending on our application needs.

* Read Uncommitted
* Read Committed
* Repeatable Read
* Snapshot
* Serializable

**The isolation level that you choose for your transaction**, defines the degree to which one transaction must be isolated from resource or data modifications made by other transactions. Depending on the isolation level you have chosen you get varying degrees of performance and concurrency problems. The table here has the list of isoltaion levels along with concurrency side effects.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Isolation Level** | **Dirty Reads** | **Lost Update** | **Nonrepeatable Reads** | **Phantom Reads** |
| **Read Uncommitted** | Yes | Yes | Yes | Yes |
| **Read Committed** | No | Yes | Yes | Yes |
| **Repeatable Read** | No | No | No | Yes |
| **Snapshot** | No | No | No | No |
| **Serializable** | No | No | No | No |

If you choose the lowest isolation level (i.e Read Uncommitted), it increases the number of concurrent transactions that can be executed at the same time, but the down side is you have all sorts of concurrency issues. On the other hand if you choose the highest isolation level (i.e Serializable), you will have no concurrency side effects, but the downside is that, this will reduce the number of concurrent transactions that can be executed at the same time if those transactions work with same data.  
  
In our upcoming videos we will discuss the concurrency problems in detail with examples

# 71. sql server dirty read example

**Suggested Videos**  
[Part 68 - Optional parameters in sql server stored procedures](http://csharp-video-tutorials.blogspot.com/2013/09/part-68-optional-parameters-in-sql.html)  
[Part 69 - Merge in SQL Server](http://csharp-video-tutorials.blogspot.com/2014/09/part-69-merge-in-sql-server.html)   
[Part 70 - sql server concurrent transactions](http://csharp-video-tutorials.blogspot.com/2015/08/sql-server-concurrent-transactions.html)   
  
  
  
In this video we will discuss, **dirty read concurrency problem** with an example. This is continuation to [Part 70](http://csharp-video-tutorials.blogspot.com/2015/08/sql-server-concurrent-transactions.html). Please watch [Part 70](http://csharp-video-tutorials.blogspot.com/2015/08/sql-server-concurrent-transactions.html) from [SQL Server tutorial for beginners](https://www.youtube.com/playlist?list=PL08903FB7ACA1C2FB).   
  
  
  
A dirty read happens when one transaction is permitted to read data that has been modified by another transaction that has not yet been committed. In most cases this would not cause a problem. However, if the first transaction is rolled back after the second reads the data, the second transaction has dirty data that does not exist anymore.   
  
SQL script to create table tblInventory

Create table tblInventory

(

    Id int identity primary key,

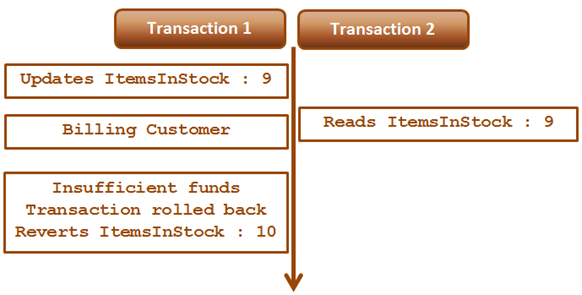
    Product nvarchar(100),

    ItemsInStock int

)

Go

Insert into tblInventory values ('iPhone', 10)

**Table tblInventory**   
dirty read problem in concurrency control   
  
**Dirty Read Example :** In the example below, Transaction 1, updates the value of ItemsInStock to 9. Then it starts to bill the customer. While Transaction 1 is still in progress, Transaction 2 starts and reads ItemsInStock value which is 9 at the moment. At this point, Transaction 1 fails because of insufficient funds and is rolled back. The ItemsInStock is reverted to the original value of 10, but Transaction 2 is working with a different value (i.e 10).   
  
  
  
**Transaction 1 :**  
Begin Tran

Update tblInventory set ItemsInStock = 9 where Id=1

-- Billing the customer

Waitfor Delay '00:00:15'

-- Insufficient Funds. Rollback transaction

Rollback Transaction

**Transaction 2 :**

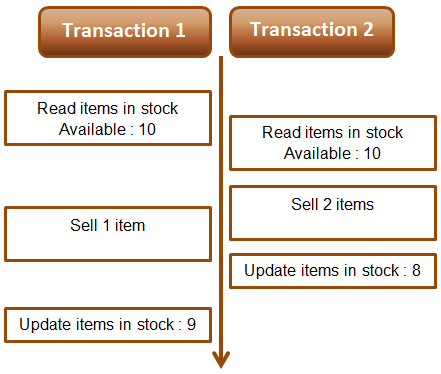
Set Transaction Isolation Level Read Uncommitted

Select \* from tblInventory where Id=1

Read Uncommitted transaction isolation level is the only isolation level that has dirty read side effect. This is the least restrictive of all the isolation levels. When this transaction isolation level is set, it is possible to read uncommitted or dirty data. Another option to read dirty data is by using NOLOCK table hint. The query below is equivalent to the query in Transaction 2.

Select \* from tblInventory (NOLOCK) where Id=1

# 72. sql server lost update problem

**Suggested Videos**  
[Part 69 - Merge in SQL Server](http://csharp-video-tutorials.blogspot.com/2014/09/part-69-merge-in-sql-server.html)  
[Part 70 - sql server concurrent transactions](http://csharp-video-tutorials.blogspot.com/2015/08/sql-server-concurrent-transactions.html)   
[Part 71 - sql server dirty read example](http://csharp-video-tutorials.blogspot.com/2015/08/sql-server-dirty-read-example.html)   
  
  
  
In this video we will discuss, **lost update problem in sql server** with an example.   
  
  
  
**Lost update problem happens when 2 transactions read and update the same data**. Let's understand this with an example. We will use the following table **tblInventory**for this example.   
  
lost update concurrency problem example   
  
As you can see in the diagram below there are 2 transactions - Transaction 1 and Transaction 2. Transaction 1 starts first, and it is processing an order for 1 iPhone. It sees ItemsInStock as 10.   
  
At this time Transaction 2 is processing another order for 2 iPhones. It also sees ItemsInStock as 10. Transaction 2 makes the sale first and updates ItemsInStock with a value of 8.   
  
At this point Transaction 1 completes the sale and silently overwrites the update of Transaction 2. As Transaction 1 sold 1 iPhone it has updated ItemsInStock to 9, while it actually should have updated it to 7.   
  
   
  
**Example :** The lost update problem example. Open 2 instances of SQL Server Management studio. From the first window execute Transaction 1 code and from the second window, execute Transaction 2 code. Transaction 1 is processing an order for 1 iPhone, while Transaction 2 is processing an order for 2 iPhones. At the end of both the transactions ItemsInStock must be 7, but we have a value of 9. This is because Transaction 1 silently overwrites the update of Transaction 2. This is called the **lost update problem**.

-- Transaction 1

Begin Tran

Declare @ItemsInStock int

Select @ItemsInStock = ItemsInStock

from tblInventory where Id=1

-- Transaction takes 10 seconds

Waitfor Delay '00:00:10'

Set @ItemsInStock = @ItemsInStock - 1

Update tblInventory

Set ItemsInStock = @ItemsInStock where Id=1

Print @ItemsInStock

Commit Transaction

-- Transaction 2

Begin Tran

Declare @ItemsInStock int

Select @ItemsInStock = ItemsInStock

from tblInventory where Id=1

-- Transaction takes 1 second

Waitfor Delay '00:00:1'

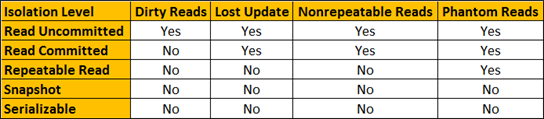
Set @ItemsInStock = @ItemsInStock - 2

Update tblInventory

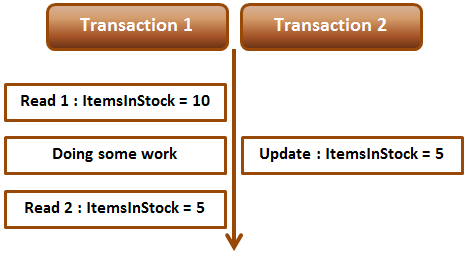
Set ItemsInStock = @ItemsInStock where Id=1

Print @ItemsInStock

Commit Transaction

Both Read Uncommitted and Read Committed transaction isolation levels have the lost update side effect. Repeatable Read, Snapshot, and Serializable isolation levels does not have this side effect. If you run the above Transactions using any of the higher isolation levels (Repeatable Read, Snapshot, or Serializable) you will not have lost update problem. The repeatable read isolation level uses additional locking on rows that are read by the current transaction, and prevents them from being updated or deleted elsewhere. This solves the lost update problem.   
  
   
  
For both the above transactions, set Repeatable Read Isolation Level. Run Transaction 1 first and then a few seconds later run Transaction 2. Transaction 1 completes successfully, but Transaction 2 competes with the following error.  
Transaction was deadlocked on lock resources with another process and has been chosen as the deadlock victim. Rerun the transaction.   
  
Once you rerun Transaction 2, ItemsInStock will be updated correctly as expected.

# 73. Non repeatable read example in sql server

**Suggested Videos**  
[Part 70 - sql server concurrent transactions](http://csharp-video-tutorials.blogspot.com/2015/08/sql-server-concurrent-transactions.html)  
[Part 71 - sql server dirty read example](http://csharp-video-tutorials.blogspot.com/2015/08/sql-server-dirty-read-example.html)   
[Part 72 - sql server lost update problem](http://csharp-video-tutorials.blogspot.com/2015/08/sql-server-lost-update-problem.html)   
  
  
  
In this video we will discuss **non repeatable read concurrency problem with an example**.   
  
  
  
Non repeatable read problem happens when one transaction reads the same data twice and another transaction updates that data in between the first and second read of transaction one.    
  
We will use the following table **tblInventory** in this demo   
sql server non-repeatable read   
  
**The following diagram explains the problem :** Transaction 1 starts first. Reads ItemsInStock. Gets a value of 10 for first read. Transaction 1 is doing some work and at this point Transaction 2 starts and UpdatesItemsInStock to 5. Transaction 1 then makes a second read. At this point Transaction 1 gets a value of 5, reulting in non-repeatable read problem.   
  
   
  
**Non-repeatable read example :** Open 2 instances of SQL Server Management studio. From the first window execute Transaction 1 code and from the second window, execute Transaction 2 code. Notice that when Transaction 1 completes, it gets different values for read 1 and read 2, resulting in non-repeatable read.

-- Transaction 1

Begin Transaction

Select ItemsInStock from tblInventory where Id = 1

-- Do Some work

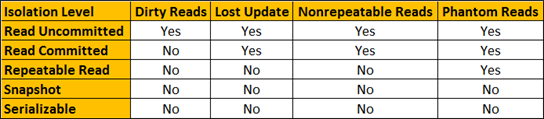
waitfor delay '00:00:10'

Select ItemsInStock from tblInventory where Id = 1

Commit Transaction

-- Transaction 2

Update tblInventory set ItemsInStock = 5 where Id = 1

Repeatable read or any other higher isolation level should solve the non-repeatable read problem.  
   
  
**Fixing non repeatable read concurrency problem :** To fix the non-repeatable read problem, set transaction isolation level of Transaction 1 to repeatable read. This will ensure that the data that Transaction 1 has read, will be prevented from being updated or deleted elsewhere. This solves the non-repeatable read problem.    
  
When you execute Transaction 1 and 2 from 2 different instances of SQL Server management studio, Transaction 2 is blocked until Transaction 1 completes and at the end of Transaction 1, both the reads get the same value for ItemsInStock.

-- Transaction 1

Set transaction isolation level repeatable read

Begin Transaction

Select ItemsInStock from tblInventory where Id = 1

-- Do Some work

waitfor delay '00:00:10'

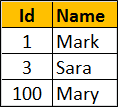
Select ItemsInStock from tblInventory where Id = 1

Commit Transaction

-- Transaction 2

Update tblInventory set ItemsInStock = 5 where Id = 1

# 74. Phantom reads example in sql server

**Suggested Videos**  
[Part 71 - sql server dirty read example](http://csharp-video-tutorials.blogspot.com/2015/08/sql-server-dirty-read-example.html)  
[Part 72 - sql server lost update problem](http://csharp-video-tutorials.blogspot.com/2015/08/sql-server-lost-update-problem.html)   
[Part 73 - Non repeatable read example in sql server](http://csharp-video-tutorials.blogspot.com/2015/08/non-repeatable-read-example-in-sql.html)   
  
  
  
In this video we will discuss **phantom read concurrency problem** with examples.   
  
  
  
Phantom read happens when one transaction executes a query twice and it gets a different number of rows in the result set each time. This happens when a second transaction inserts a new row that matches the WHERE clause of the query executed by the first transaction.    
  
We will use the following table tblEmployees in this demo  
   
  
**Scrip to create the table tblEmployees**

Create table tblEmployees

(

    Id int primary key,

    Name nvarchar(50)

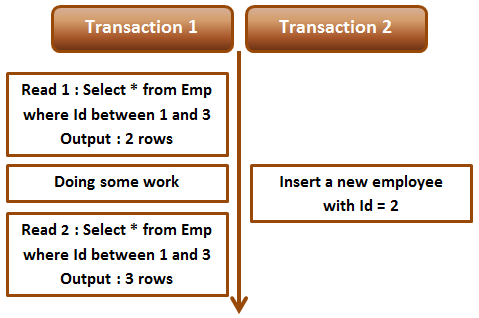
)

Go

Insert into tblEmployees values(1,'Mark')

Insert into tblEmployees values(3, 'Sara')

Insert into tblEmployees values(100, 'Mary')

**The following diagram explains the problem :** Transaction 1 starts first. Reads from Emp table where Id between 1 and 3. 2 rows retrieved for first read. Transaction 1 is doing some work and at this point Transaction 2 starts and inserts a new employee with Id = 2. Transaction 1 then makes a second read. 3 rows retrieved for second read, reulting in phantom read problem.   
  
   
  
**Phantom read example :** Open 2 instances of SQL Server Management studio. From the first window execute Transaction 1 code and from the second window, execute Transaction 2 code. Notice that when Transaction 1 completes, it gets different number of rows for read 1 and read 2, resulting in phantom read.

-- Transaction 1

Begin Transaction

Select \* from tblEmployees where Id between 1 and 3

-- Do Some work

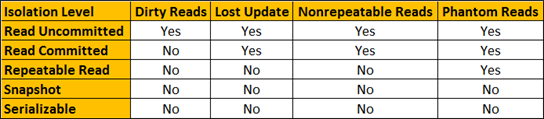
waitfor delay '00:00:10'

Select \* from tblEmployees where Id between 1 and 3

Commit Transaction

-- Transaction 2

Insert into tblEmployees values(2, 'Marcus')

Serializable or any other higher isolation level should solve the phantom read problem.  
   
  
**Fixing phantom read concurrency problem :** To fix the phantom read problem, set transaction isolation level of Transaction 1 to serializable. This will place a range lock on the rows between 1 and 3, which prevents any other transaction from inserting new rows with in that range. This solves the phantom read problem.    
  
When you execute Transaction 1 and 2 from 2 different instances of SQL Server management studio, Transaction 2 is blocked until Transaction 1 completes and at the end of Transaction 1, both the reads get the same number of rows.

-- Transaction 1

Set transaction isolation level serializable

Begin Transaction

Select \* from tblEmployees where Id between 1 and 3

-- Do Some work

waitfor delay '00:00:10'

Select \* from tblEmployees where Id between 1 and 3

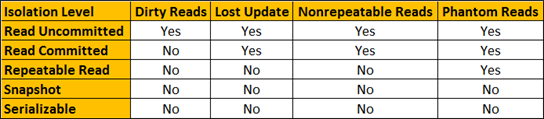
Commit Transaction

-- Transaction 2

Insert into tblEmployees values(2, 'Marcus')

**Difference between repeatable read and serializable**  
**Repeatable read prevents only non-repeatable read.** Repeatable read isolation level ensures that the data that one transaction has read, will be prevented from being updated or deleted by any other transaction, but it doe not prevent new rows from being inserted by other transactions resulting in phantom read concurrency problem.   
  
**Serializable prevents both non-repeatable read and phantom read problems.**Serializable isolation level ensures that the data that one transaction has read, will be prevented from being updated or deleted by any other transaction. It also prevents new rows from being inserted by other transactions, so this isolation level prevents both non-repeatable read and phantom read problems.

# 75. Snapshot isolation level in sql server

**Suggested Videos**  
[Part 72 - sql server lost update problem](http://csharp-video-tutorials.blogspot.com/2015/08/sql-server-lost-update-problem.html)  
[Part 73 - Non repeatable read example in sql server](http://csharp-video-tutorials.blogspot.com/2015/08/non-repeatable-read-example-in-sql.html)   
[Part 74 - Phantom reads example in sql server](http://csharp-video-tutorials.blogspot.com/2015/08/phantom-reads-example-in-sql-server.html)   
  
  
  
In this video we will discuss, **snapshot isolation level in sql server** with examples.   
  
  
  
As you can see from the table below, just like serializable isolation level, snapshot isolation level does not have any concurrency side effects.   
   
  
**What is the difference between serializable and snapshot isolation levels**  
Serializable isolation is implemented by acquiring locks which means the resources are locked for the duration of the current transaction. This isolation level does not have any concurrency side effects but at the cost of significant reduction in concurrency.   
  
Snapshot isolation doesn't acquire locks, it maintains versioning in Tempdb. Since, snapshot isolation does not lock resources, it can significantly increase the number of concurrent transactions while providing the same level of data consistency as serializable isolation does.  
  
Let us understand Snapshot isolation with an example. We will be using the following table tblInventory for this example.   
snapshot isolation example   
  
Open 2 instances of SQL Server Management studio. From the first window execute Transaction 1 code and from the second window execute Transaction 2 code. Notice that Transaction 2 is blocked until Transaction 1 is completed.

--Transaction 1

Set transaction isolation level serializable

Begin Transaction

Update tblInventory set ItemsInStock = 5 where Id = 1

waitfor delay '00:00:10'

Commit Transaction

-- Transaction 2

Set transaction isolation level serializable

Select ItemsInStock from tblInventory where Id = 1

Now change the isolation level of Transaction 2 to snapshot. To set snapshot isolation level, it must first be enabled at the database level, and then set the transaction isolation level to snapshot.

-- Transaction 2

-- Enable snapshot isloation for the database

Alter database SampleDB SET ALLOW\_SNAPSHOT\_ISOLATION ON

-- Set the transaction isolation level to snapshot

Set transaction isolation level snapshot

Select ItemsInStock from tblInventory where Id = 1

From the first window execute Transaction 1 code and from the second window, execute Transaction 2 code. Notice that Transaction 2 is not blocked and returns the data from the database as it was before Transaction 1 has started.   
  
**Modifying data with snapshot isolation level :** Now let's look at an example of what happens when a transaction that is using snapshot isolation tries to update the same data that another transaction is updating at the same time.   
  
In the following example, Transaction 1 starts first and it is updating ItemsInStock to 5. At the same time, Transaction 2 that is using snapshot isolation level is also updating the same data. Notice that Transaction 2 is blocked until Transaction 1 completes. When Transaction 1 completes, Transaction 2 fails with the following error to prevent concurrency side effect - Lost update. If Transaction 2 was allowed to continue, it would have changed the ItemsInStock value to 8 and when Transaction 1 completes it overwrites ItemsInStock to 5, which means we have lost an update. To complete the work that Transaction 2 is doing we will have to rerun the transaction.   
  
Snapshot isolation transaction aborted due to update conflict. You cannot use snapshot isolation to access table 'dbo.tblInventory' directly or indirectly in database 'SampleDB' to update, delete, or insert the row that has been modified or deleted by another transaction. Retry the transaction or change the isolation level for the update/delete statement.

--Transaction 1

Set transaction isolation level serializable

Begin Transaction

Update tblInventory set ItemsInStock = 5 where Id = 1

waitfor delay '00:00:10'

Commit Transaction

-- Transaction 2

-- Enable snapshot isloation for the database

Alter database SampleDB SET ALLOW\_SNAPSHOT\_ISOLATION ON

-- Set the transaction isolation level to snapshot

Set transaction isolation level snapshot

Update tblInventory set ItemsInStock = 8 where Id = 1

# 76. Read committed snapshot isolation level in sql server

**Suggested Videos**  
[Part 73 - Non repeatable read example in sql server](http://csharp-video-tutorials.blogspot.com/2015/08/non-repeatable-read-example-in-sql.html)  
[Part 74 - Phantom reads example in sql server](http://csharp-video-tutorials.blogspot.com/2015/08/phantom-reads-example-in-sql-server.html)   
[Part 75 - Snapshot isolation level in sql server](http://csharp-video-tutorials.blogspot.com/2015/08/snapshot-isolation-level-in-sql-server.html)   
  
  
  
In this video we will discuss **Read committed snapshot isolation level** in sql server. This is continuation [Part 75](http://csharp-video-tutorials.blogspot.com/2015/08/snapshot-isolation-level-in-sql-server.html). Please watch [Part 75](http://csharp-video-tutorials.blogspot.com/2015/08/snapshot-isolation-level-in-sql-server.html) from [SQL Server tutorial](https://www.youtube.com/playlist?list=PL08903FB7ACA1C2FB) before proceeding.   
  
  
  
**We will use the following table tblInventory in this demo**   
Read committed snapshot isolation level example   
  
Read committed snapshot isolation level is not a different isolation level. It is a different way of implementing Read committed isolation level. One problem we have with Read Committed isloation level is that, it blocks the transaction if it is trying to read the data, that another transaction is updating at the same time.   
  
The following example demonstrates the above point. Open 2 instances of SQL Server Management studio. From the first window execute Transaction 1 code and from the second window execute Transaction 2 code. Notice that Transaction 2 is blocked until Transaction 1 is completed.

--Transaction 1

Set transaction isolation level Read Committed

Begin Transaction

Update tblInventory set ItemsInStock = 5 where Id = 1

waitfor delay '00:00:10'

Commit Transaction

-- Transaction 2

Set transaction isolation level read committed

Begin Transaction

Select ItemsInStock from tblInventory where Id = 1

Commit Transaction

We can make Transaction 2 to use row versioning technique instead of locks by enabling Read committed snapshot isolation at the database level. Use the following command to enable READ\_COMMITTED\_SNAPSHOT isolation

Alter database SampleDB SET READ\_COMMITTED\_SNAPSHOT ON

**Please note :** For the above statement to execute successfully all the other database connections should be closed.   
  
After enabling READ\_COMMITTED\_SNAPSHOT, execute Transaction 1 first and then Transaction 2 simultaneously. Notice that the Transaction 2 is not blocked. It immediately returns the committed data that is in the database before Transaction 1 started. This is because Transaction 2 is now using Read committed snapshot isolation level.   
  
Let's see if we can achieve the same thing using snapshot isolation level instead of read committed snapshot isolation level.    
  
**Step 1 :** Turn off READ\_COMMITTED\_SNAPSHOT

Alter database SampleDB SET READ\_COMMITTED\_SNAPSHOT OFF

**Step 2 :** Enable snapshot isolation level at the database level

Alter database SampleDB SET ALLOW\_SNAPSHOT\_ISOLATION ON

**Step 3 :**Execute Transaction 1 first and then Transaction 2 simultaneously. Just like in the previous example, notice that the Transaction 2 is not blocked. It immediately returns the committed data that is in the database before Transaction 1 started.

--Transaction 1

Set transaction isolation level Read Committed

Begin Transaction

Update tblInventory set ItemsInStock = 5 where Id = 1

waitfor delay '00:00:10'

Commit Transaction

-- Transaction 2

Set transaction isolation level snapshot

Begin Transaction

Select ItemsInStock from tblInventory where Id = 1

Commit Transaction

**So what is the point in using read committed snapshot isolation level over snapshot isolation level?**  
There are some differences between read committed snapshot isolation level and snapshot isolation level. We will discuss these in our next video.

# 77. Difference between snapshot isolation and read committed snapshot

**Suggested Videos**  
[Part 74 - Phantom reads example in sql server](http://csharp-video-tutorials.blogspot.com/2015/08/phantom-reads-example-in-sql-server.html)  
[Part 75 - Snapshot isolation level in sql server](http://csharp-video-tutorials.blogspot.com/2015/08/snapshot-isolation-level-in-sql-server.html)   
[Part 76 - Read committed snapshot isolation level in sql server](http://csharp-video-tutorials.blogspot.com/2015/08/read-committed-snapshot-isolation-level.html)   
  
  
  
In this video we will discuss the differences between snapshot isolation and read committed snapshot isolation in sql server. This is continuation to [Parts 75](http://csharp-video-tutorials.blogspot.com/2015/08/snapshot-isolation-level-in-sql-server.html) and [76](http://csharp-video-tutorials.blogspot.com/2015/08/read-committed-snapshot-isolation-level.html). Please watch [Part 75](http://csharp-video-tutorials.blogspot.com/2015/08/snapshot-isolation-level-in-sql-server.html) and [76](http://csharp-video-tutorials.blogspot.com/2015/08/read-committed-snapshot-isolation-level.html) from [SQL Server tutorial](https://www.youtube.com/playlist?list=PL08903FB7ACA1C2FB) before proceeding. 

|  |  |
| --- | --- |
| **Read Committed Snapshot Isolation** | **Snapshot Isolation** |
| No update conflicts | Vulnerable to update conflicts |
| Works with existing applications without requiring any change to the application | Application change may be required to use with an existing application |
| Can be used with distributed transactions | Cannot be used with distributed transactions |
| Provides statement-level read consistency | Provides transaction-level read consistency |

**Update conflicts :** Snapshot isolation is vulnerable to update conflicts where as Read Committed Snapshot Isolation is not. When a transaction running under snapshot isolation triess to update data that an another transaction is already updating at the sametime, an update conflict occurs and the transaction terminates and rolls back with an error.   
  
**We will use the following table tblInventory in this demo**   
Read committed snapshot isolation level example   
  
Enable Snapshot Isolation for the SampleDB database using the following command

Alter database SampleDB SET ALLOW\_SNAPSHOT\_ISOLATION ON

Open 2 instances of SQL Server Management studio. From the first window execute Transaction 1 code and from the second window execute Transaction 2 code. Notice that Transaction 2 is blocked until Transaction 1 is completed. When Transaction 1 completes, Transaction 2 raises an update conflict and the transaction terminates and rolls back with an error.

--Transaction 1

Set transaction isolation level snapshot

Begin Transaction

Update tblInventory set ItemsInStock = 8 where Id = 1

waitfor delay '00:00:10'

Commit Transaction

-- Transaction 2

Set transaction isolation level snapshot

Begin Transaction

Update tblInventory set ItemsInStock = 5 where Id = 1

Commit Transaction

Now let's try the same thing using **Read Committed Sanpshot Isolation**   
  
**Step 1 :** Disable Snapshot Isolation for the SampleDB database using the following command

Alter database SampleDB SET ALLOW\_SNAPSHOT\_ISOLATION OFF

**Step 2 :**Enable Read Committed Sanpshot Isolation at the database level using the following command

Alter database SampleDB SET READ\_COMMITTED\_SNAPSHOT ON

**Step 3 :**Open 2 instances of SQL Server Management studio. From the first window execute Transaction 1 code and from the second window execute Transaction 2 code. Notice that Transaction 2 is blocked until Transaction 1 is completed. When Transaction 1 completes, Transaction 2 also completes successfully without any update conflict.

--Transaction 1

Set transaction isolation level read committed

Begin Transaction

Update tblInventory set ItemsInStock = 8 where Id = 1

waitfor delay '00:00:10'

Commit Transaction

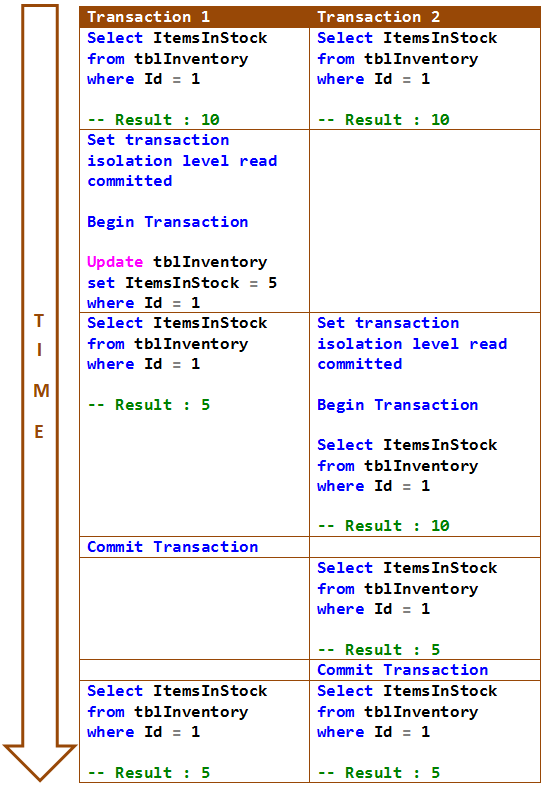
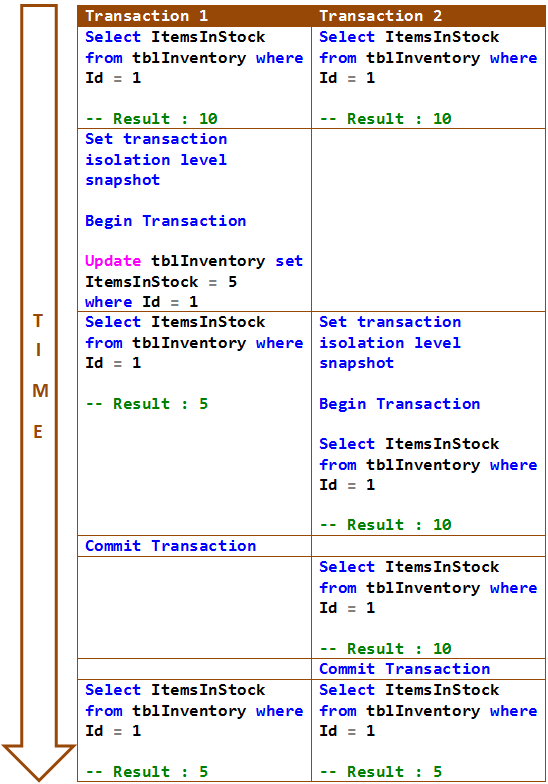
-- Transaction 2

Set transaction isolation level read committed

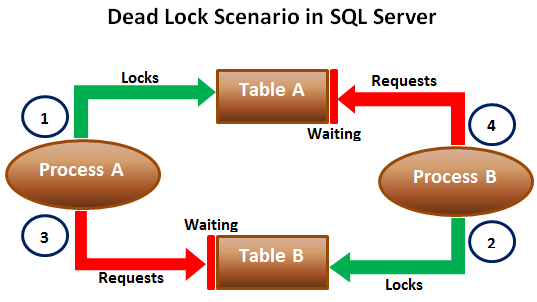
Begin Transaction

Update tblInventory set ItemsInStock = 5 where Id = 1

Commit Transaction

**Existing application :** If your application is using the default Read Committed isolation level, you can very easily make the application to use Read Committed Snapshot Isolation without requiring any change to the application at all. All you need to do is turn on READ\_COMMITTED\_SNAPSHOT option in the database, which will change read committed isolation to use row versioning when reading the committed data.  
  
**Distributed transactions :** Read Committed Snapshot Isolation works with distributed transactions, whereas snapshot isolation does not.  
  
**Read consistency :**Read Committed Snapshot Isolation provides statement-level read consistency where as Snapshot Isolation provides transaction-level read consistency. The following diagrams explain this.  
  
Transaction 2 has 2 select statements. Notice that both of these select statements return different data. This is because Read Committed Snapshot Isolation returns the last committed data before the select statement began and not the last committed data before the transaction began.   
   
  
In the following example, both the select statements of Transaction 2 return same data. This is because Snapshot Isolation returns the last committed data before the transaction began and not the last committed data before the select statement began.   


# 78. SQL Server deadlock example

**Suggested Videos**  
[Part 75 - Snapshot isolation level in sql server](http://csharp-video-tutorials.blogspot.com/2015/08/snapshot-isolation-level-in-sql-server.html)  
[Part 76 - Read committed snapshot isolation level in sql server](http://csharp-video-tutorials.blogspot.com/2015/08/read-committed-snapshot-isolation-level.html)   
[Part 77 - Difference between snapshot isolation and read committed snapshot](http://csharp-video-tutorials.blogspot.com/2015/08/difference-between-snapshot-isolation.html)   
  
  
  
In this video we will discuss a scenario when a deadlock can occur in SQL Server.   
  
  
  
**When can a deadlock occur**  
In a database, a deadlock occurs when two or more processes have a resource locked, and each process requests a lock on the resource that another process has already locked. Neither of the transactions here can move forward, as each one is waiting for the other to release the lock. The following diagram explains this.   
   
  
When deadlocks occur, SQL Server will choose one of processes as the deadlock victim and rollback that process, so the other process can move forward. The transaction that is chosen as the deadlock victim will produce the following error.  
Msg 1205, Level 13, State 51, Line 1  
Transaction (Process ID 57) was deadlocked on lock resources with another process and has been chosen as the deadlock victim. Rerun the transaction.   
  
Let us look at this in action. We will use the following 2 tables for this example.

|  |  |
| --- | --- |
| http://3.bp.blogspot.com/-eIhjPb9oNyg/Vdy8R2YUqKI/AAAAAAAAdp0/StcdK_dsUUI/s1600/497.png | http://2.bp.blogspot.com/-i2tMMheW3fo/Vdy8a9yPJzI/AAAAAAAAdp8/T1xsVOUgG44/s1600/498.png |

SQL script to create the tables and populate them with test data

Create table TableA

(

    Id int identity primary key,

    Name nvarchar(50)

)

Go

Insert into TableA values ('Mark')

Go

Create table TableB

(

    Id int identity primary key,

    Name nvarchar(50)

)

Go

Insert into TableB values ('Mary')

Go

The following 2 transactions will result in a dead lock. Open 2 instances of SQL Server Management studio. From the first window execute Transaction 1 code and from the second window execute Transaction 2 code.

-- Transaction 1

Begin Tran

Update TableA Set Name = 'Mark Transaction 1' where Id = 1

-- From Transaction 2 window execute the first update statement

Update TableB Set Name = 'Mary Transaction 1' where Id = 1

-- From Transaction 2 window execute the second update statement

Commit Transaction

-- Transaction 2

Begin Tran

Update TableB Set Name = 'Mark Transaction 2' where Id = 1

-- From Transaction 1 window execute the second update statement

Update TableA Set Name = 'Mary Transaction 2' where Id = 1

-- After a few seconds notice that one of the transactions complete

-- successfully while the other transaction is made the deadlock victim

Commit Transaction

**Next Video :** We will discuss the criteria SQL Server uses to choose a deadlock victim

# 79. SQL Server deadlock victim selection

**Suggested Videos**  
[Part 76 - Read committed snapshot isolation level in sql server](http://csharp-video-tutorials.blogspot.com/2015/08/read-committed-snapshot-isolation-level.html)  
[Part 77 - Difference between snapshot isolation and read committed snapshot](http://csharp-video-tutorials.blogspot.com/2015/08/difference-between-snapshot-isolation.html)   
[Part 78 - SQL Server deadlock example](http://csharp-video-tutorials.blogspot.com/2015/08/sql-server-deadlock-example.html)   
  
  
  
**In this video we will discuss**   
1. How SQL Server detects deadlocks  
2. What happens when a deadlock is detected  
3. What is DEADLOCK\_PRIORITY  
4. What is the criteria that SQL Server uses to choose a deadlock victim when there is a deadlock   
  
  
  
This is continuation to [Part 78](http://csharp-video-tutorials.blogspot.com/2015/08/sql-server-deadlock-example.html), please watch [Part 78](http://csharp-video-tutorials.blogspot.com/2015/08/sql-server-deadlock-example.html) before proceeding.   
  
**How SQL Server detects deadlocks**  
Lock monitor thread in SQL Server, runs every 5 seconds by default to detect if there are any deadlocks. If the lock monitor thread finds deadlocks, the deadlock detection interval will drop from 5 seconds to as low as 100 milliseconds depending on the frequency of deadlocks. If the lock monitor thread stops finding deadlocks, the Database Engine increases the intervals between searches to 5 seconds.   
  
**What happens when a deadlock is detected**  
When a deadlock is detected, the Database Engine ends the deadlock by choosing one of the threads as the deadlock victim. The deadlock victim's transaction is then rolled back and returns a 1205 error to the application. Rolling back the transaction of the deadlock victim releases all locks held by that transaction. This allows the other transactions to become unblocked and move forward.   
  
**What is DEADLOCK\_PRIORITY**  
By default, SQL Server chooses a transaction as the deadlock victim that is least expensive to roll back. However, a user can specify the priority of sessions in a deadlock situation using the SET DEADLOCK\_PRIORITY statement. The session with the lowest deadlock priority is chosen as the deadlock victim.   
  
Example : SET DEADLOCK\_PRIORITY NORMAL   
  
**DEADLOCK\_PRIORITY**  
1. The default is Normal  
2. Can be set to LOW, NORMAL, or HIGH  
3. Can also be set to a integer value in the range of -10 to 10.  
 LOW : -5  
 NORMAL : 0  
 HIGH : 5   
  
**What is the deadlock victim selection criteria**  
1. If the DEADLOCK\_PRIORITY is different, the session with the lowest priority is selected as the victim  
2. If both the sessions have the same priority, the transaction that is least expensive to rollback is selected as the victim  
3. If both the sessions have the same deadlock priority and the same cost, a victim is chosen randomly   
  
**SQL Script to setup the tables for the examples**

Create table TableA

(

    Id int identity primary key,

    Name nvarchar(50)

)

Go

Insert into TableA values ('Mark')

Insert into TableA values ('Ben')

Insert into TableA values ('Todd')

Insert into TableA values ('Pam')

Insert into TableA values ('Sara')

Go

Create table TableB

(

    Id int identity primary key,

    Name nvarchar(50)

)

Go

Insert into TableB values ('Mary')

Go

Open 2 instances of SQL Server Management studio. From the first window execute Transaction 1 code and from the second window execute Transaction 2 code. We have not explicitly set DEADLOCK\_PRIORITY, so both the sessions have the default DEADLOCK\_PRIORITY which is NORMAL. So in this case SQL Server is going to choose Transaction 2 as the deadlock victim as it is the least expensive one to rollback.

-- Transaction 1

Begin Tran

Update TableA Set Name = Name + ' Transaction 1' where Id IN (1, 2, 3, 4, 5)

-- From Transaction 2 window execute the first update statement

Update TableB Set Name = Name + ' Transaction 1' where Id = 1

-- From Transaction 2 window execute the second update statement

Commit Transaction

-- Transaction 2

Begin Tran

Update TableB Set Name = Name + ' Transaction 2' where Id = 1

-- From Transaction 1 window execute the second update statement

Update TableA Set Name = Name + ' Transaction 2' where Id IN (1, 2, 3, 4, 5)

-- After a few seconds notice that this transaction will be chosen as the deadlock

-- victim as it is less expensive to rollback this transaction than Transaction 1

Commit Transaction

In the following example we have set DEADLOCK\_PRIORITY of Transaction 2 to HIGH. Transaction 1 will be chosen as the deadlock victim, because it's DEADLOCK\_PRIORITY (Normal) is lower than the DEADLOCK\_PRIORITY of Transaction 2.

-- Transaction 1

Begin Tran

Update TableA Set Name = Name + ' Transaction 1' where Id IN (1, 2, 3, 4, 5)

-- From Transaction 2 window execute the first update statement

Update TableB Set Name = Name + ' Transaction 1' where Id = 1

-- From Transaction 2 window execute the second update statement

Commit Transaction

-- Transaction 2

SET DEADLOCK\_PRIORITY HIGH

GO

Begin Tran

Update TableB Set Name = Name + ' Transaction 2' where Id = 1

-- From Transaction 1 window execute the second update statement

Update TableA Set Name = Name + ' Transaction 2' where Id IN (1, 2, 3, 4, 5)

-- After a few seconds notice that Transaction 2 will be chosen as the

-- deadlock victim as it's DEADLOCK\_PRIORITY (Normal) is lower than the

-- DEADLOCK\_PRIORITY this transaction (HIGH)

Commit Transaction

# 80. Logging deadlocks in sql server

**Suggested Videos**  
[Part 77 - Difference between snapshot isolation and read committed snapshot](http://csharp-video-tutorials.blogspot.com/2015/08/difference-between-snapshot-isolation.html)  
[Part 78 - SQL Server deadlock example](http://csharp-video-tutorials.blogspot.com/2015/08/sql-server-deadlock-example.html)   
[Part 79 - SQL Server deadlock victim selection](http://csharp-video-tutorials.blogspot.com/2015/08/sql-server-deadlock-victim-selection.html)   
  
  
  
In this video we will discuss **how to write the deadlock information to the SQL Server error log**  
  
  
  
**When deadlocks occur**, SQL Server chooses one of the transactions as the deadlock victim and rolls it back. There are several ways in SQL Server to track down the queries that are causing deadlocks. One of the options is to use SQL Server trace flag 1222 to write the deadlock information to the SQL Server error log.   
  
**Enable Trace flag :** To enable trace flags use DBCC command. -1 parameter indicates that the trace flag must be set at the global level. If you omit -1 parameter the trace flag will be set only at the session level.

DBCC Traceon(1222, -1)   
  
To check the status of the trace flag  
DBCC TraceStatus(1222, -1)   
  
To turn off the trace flag  
DBCC Traceoff(1222, -1)  
  
The following SQL code generates a dead lock. This is the same code we discussed in[Part 78](http://csharp-video-tutorials.blogspot.com/2015/08/sql-server-deadlock-example.html) of [SQL Server Tutorial](https://www.youtube.com/playlist?list=PL08903FB7ACA1C2FB).

**--SQL script to create the tables and populate them with test data**

Create table TableA

(

    Id int identity primary key,

    Name nvarchar(50)

)

Go

Insert into TableA values ('Mark')

Go

Create table TableB

(

    Id int identity primary key,

    Name nvarchar(50)

)

Go

Insert into TableB values ('Mary')

Go

**--SQL Script to create stored procedures**

Create procedure spTransaction1

as

Begin

    Begin Tran

    Update TableA Set Name = 'Mark Transaction 1' where Id = 1

    Waitfor delay '00:00:05'

    Update TableB Set Name = 'Mary Transaction 1' where Id = 1

    Commit Transaction

End

Create procedure spTransaction2

as

Begin

    Begin Tran

    Update TableB Set Name = 'Mark Transaction 2' where Id = 1

    Waitfor delay '00:00:05'

    Update TableA Set Name = 'Mary Transaction 2' where Id = 1

    Commit Transaction

End

Open 2 instances of SQL Server Management studio. From the first window execute**spTransaction1** and from the second window execute **spTransaction2**.    
  
After a few seconds notice that one of the transactions complete successfully while the other transaction is made the deadlock victim and rollback.   
  
The information about this deadlock should now have been logged in sql server error log.   
  
**To read the error log**  
execute sp\_readerrorlog   
  
**Next video :**How to read and understand the deadlock information that is logged in the sql server error log

# 81. SQL Server deadlock analysis and prevention

**Suggested Videos**  
[Part 78 - SQL Server deadlock example](http://csharp-video-tutorials.blogspot.com/2015/08/sql-server-deadlock-example.html)  
[Part 79 - SQL Server deadlock victim selection](http://csharp-video-tutorials.blogspot.com/2015/08/sql-server-deadlock-victim-selection.html)   
[Part 80 - Logging deadlocks in sql server](http://csharp-video-tutorials.blogspot.com/2015/08/logging-deadlocks-in-sql-server.html)   
  
  
  
In this video we will discuss **how to read and analyze sql server deadlock information captured in the error log**, so we can understand what's causing the deadlocks and take appropriate actions to prevent or minimize the occurrence of deadlocks. This is continuation to [Part 80](http://csharp-video-tutorials.blogspot.com/2015/08/logging-deadlocks-in-sql-server.html). Please watch [Part 80](http://csharp-video-tutorials.blogspot.com/2015/08/logging-deadlocks-in-sql-server.html) from [SQL Server tutorial](https://www.youtube.com/playlist?list=PL08903FB7ACA1C2FB) before proceeding.   
  
  
  
**The deadlock information in the error log has three sections**

|  |  |
| --- | --- |
| **Section** | **Description** |
| Deadlock Victim | Contains the ID of the process that was selected as the deadlock victim and killed by SQL Server. |
| Process List | Contains the list of the processes that participated in the deadlock. |
| Resource List | Contains the list of the resources (database objects) owned by the processes involved in the deadlock |

**Process List :** The process list has lot of items. Here are some of them that are particularly useful in understanding what caused the deadlock.

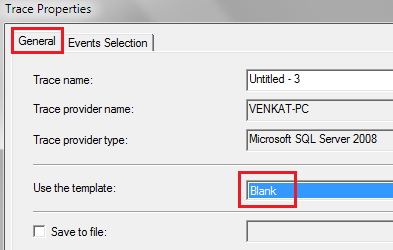
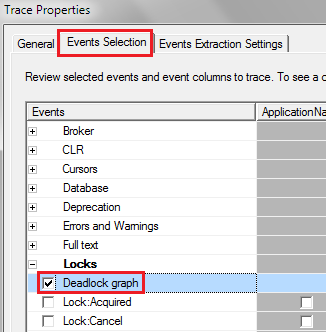
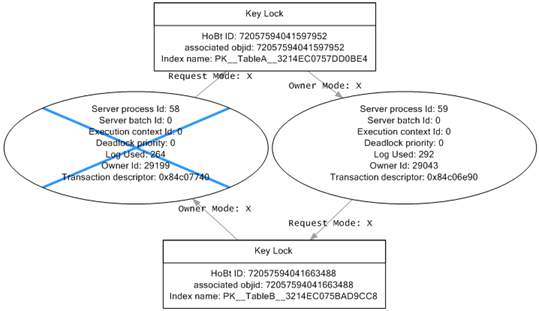
|  |  |
| --- | --- |
| **Node** | **Description** |
| loginname | The loginname associated with the process |
| isolationlevel | What isolation level is used |
| procname | The stored procedure name |
| Inputbuf | The code the process is executing when the deadlock occured |

**Resource List :** Some of the items in the resource list that are particularly useful in understanding what caused the deadlock.

|  |  |
| --- | --- |
| **Node** | **Description** |
| objectname | Fully qualified name of the resource involved in the deadlock |
| owner-list | Contains (owner id) the id of the owning process and the lock mode it has acquired on the resource. lock mode determines how the resource can be accessed by concurrent transactions. S for Shared lock, U for Update lock, X for Exclusive lock etc |
| waiter-list | Contains (waiter id) the id of the process that wants to acquire a lock on the resource and the lock mode it is requesting |

To prevent the deadlock that we have in our case, we need to ensure that database objects (Table A & Table B) are accessed in the same order every time.

# 82. Capturing deadlocks in sql profiler

**Suggested Videos**  
[Part 79 - SQL Server deadlock victim selection](http://csharp-video-tutorials.blogspot.com/2015/08/sql-server-deadlock-victim-selection.html)  
[Part 80 - Logging deadlocks in sql server](http://csharp-video-tutorials.blogspot.com/2015/08/logging-deadlocks-in-sql-server.html)   
[Part 81 - SQL Server deadlock analysis and prevention](http://csharp-video-tutorials.blogspot.com/2015/08/sql-server-deadlock-analysis-and.html)   
  
  
  
In this video we will discuss **how to capture deadlock graph using SQL profiler.**  
  
  
  
To capture deadlock graph, all you need to do is add Deadlock graph event to the trace in SQL profiler.   
  
**Here are the steps :**   
**1.** Open SQL Profiler  
**2.** Click **File - New Trace**. Provide the credentials and connect to the server  
**3.** On the general tab, select **"Blank"** template from **"Use the template"** dropdownlist   
   
  
**4.** On the **"Events Selection"** tab, expand **"Locks"**section and select **"Deadlock graph"**event   
   
  
**5.** Finally click the **Run**button to start the trace  
**6.** At this point execute the code that causes deadlock  
**7.** The deadlock graph should be captured in the profiler as shown below.   
   
  
**The deadlock graph data is captured in XML format.** If you want to extract this XML data to a physical file for later analysis, you can do so by following the steps below.  
**1.** In SQL profiler, click on **"File - Export - Extract SQL Server Events - Extract Deadlock Events"**  
**2.** Provide a name for the file  
**3.** The extension for the deadlock xml file is **.xdl**  
**4.** Finally choose if you want to export all events in a single file or each event in a separate file  
  
The deadlock information in the XML file is similar to what we have captured using the trace flag 1222.  
  
**Analyzing the deadlock graph**  
**1.** The oval on the graph, with the blue cross, represents the transaction that was chosen as the deadlock victim by SQL Server.  
**2.** The oval on the graph represents the transaction that completed successfully.  
**3.** When you move the mouse pointer over the oval, you can see the SQL code that was running that caused the deadlock.  
**4.** The oval symbols represent the process nodes

* **Server Process Id :** If you are using SQL Server Management Studio you can see the server process id on information bar at the bottom.
* **Deadlock Priority :**If you have not set DEADLOCK PRIORITY explicitly using SET DEADLOCK PRIORITY statement, then both the processes should have the same default deadlock priority NORMAL (0).
* **Log Used :**The transaction log space used. If a transaction has used a lot of log space then the cost to roll it back is also more. So the transaction that has used the least log space is killed and rolled back.

**5.** The rectangles represent the resource nodes.

* **HoBt ID :** Heap Or Binary Tree ID. Using this ID query **sys.partitions** view to find the database objects involved in the deadlock.

SELECT object\_name([object\_id])

FROM sys.partitions

WHERE hobt\_id = 72057594041663488

**6.** The arrows represent types of locks each process has on each resource node.

# 83. SQL Server deadlock error handling

**Suggested Videos**  
[Part 80 - Logging deadlocks in sql server](http://csharp-video-tutorials.blogspot.com/2015/08/logging-deadlocks-in-sql-server.html)  
[Part 81 - SQL Server deadlock analysis and prevention](http://csharp-video-tutorials.blogspot.com/2015/08/sql-server-deadlock-analysis-and.html)   
[Part 82 - Capturing deadlocks in SQL Profiler](http://csharp-video-tutorials.blogspot.com/2015/08/capturing-deadlocks-in-sql-profiler.html)   
  
  
  
In this video we will discuss **how to catch deadlock error using try/catch in SQL Server**.   
  
  
  
Modify the stored procedure as shown below to catch the deadlock error. The code is commented and is self-explanatory.

Alter procedure spTransaction1

as

Begin

    Begin Tran

    Begin Try

         Update TableA Set Name = 'Mark Transaction 1' where Id = 1

         Waitfor delay '00:00:05'

         Update TableB Set Name = 'Mary Transaction 1' where Id = 1

         -- If both the update statements succeeded.

         -- No Deadlock occurred. So commit the transaction.

         Commit Transaction

         Select 'Transaction Successful'

    End Try

    Begin Catch

         -- Check if the error is deadlock error

         If(ERROR\_NUMBER() = 1205)

         Begin

             Select 'Deadlock. Transaction failed. Please retry'

         End

         -- Rollback the transaction

         Rollback

    End Catch

End

Alter procedure spTransaction2

as

Begin

    Begin Tran

    Begin Try

         Update TableB Set Name = 'Mary Transaction 2' where Id = 1

         Waitfor delay '00:00:05'

         Update TableA Set Name = 'Mark Transaction 2' where Id = 1

         Commit Transaction

         Select 'Transaction Successful'

    End Try

    Begin Catch

         If(ERROR\_NUMBER() = 1205)

         Begin

             Select 'Deadlock. Transaction failed. Please retry'

        End

         Rollback

    End Catch

End

After modifying the stored procedures, execute both the procedures from 2 different windows simultaneously. Notice that the deadlock error is handled by the catch block.   
  
In our next video, we will discuss **how applications using ADO.NET can handle deadlock errors**.

# 84. Handling deadlocks in ado.net

**Suggested Videos**  
[Part 81 - SQL Server deadlock analysis and prevention](http://csharp-video-tutorials.blogspot.com/2015/08/sql-server-deadlock-analysis-and.html)  
[Part 82 - Capturing deadlocks in SQL profiler](http://csharp-video-tutorials.blogspot.com/2015/08/capturing-deadlocks-in-sql-profiler.html)  
[Part 83 - SQL Server deadlock error handling](http://csharp-video-tutorials.blogspot.com/2015/08/sql-server-deadlock-error-handling.html)   
  
  
  
In this video we will discuss **how to handle deadlock errors in an ADO.NET application**.   
  
  
  
**To handle deadlock errors in ADO.NET**  
**1.** Catch the SqlException object  
**2.** Check if the error is deadlock error using the Number property of the SqlException object   
  
**Stored Procedure 1 Code**

Alter procedure spTransaction1

as

Begin

    Begin Tran

    Update TableA Set Name = 'Mark Transaction 1' where Id = 1

    Waitfor delay '00:00:05'

    Update TableB Set Name = 'Mary Transaction 1' where Id = 1

    Commit Transaction

End

**Stored Procedure 2 Code**

Alter procedure spTransaction2

as

Begin

    Begin Tran

    Update TableB Set Name = 'Mark Transaction 2' where Id = 1

    Waitfor delay '00:00:05'

    Update TableA Set Name = 'Mary Transaction 2' where Id = 1

    Commit Transaction

End

**WebForm1.aspx HTML**

<table>

    <tr>

        <td>

            <asp:Button ID="Button1" runat="server"

                Text="Update Table A and then Table B"

                OnClick="Button1\_Click" />

        </td>

    </tr>

    <tr>

        <td>

            <asp:Label ID="Label1" runat="server"></asp:Label>

        </td>

    </tr>

</table>

**WebForm1.aspx.cs code**

using System;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

namespace Demo

{

    public partial class WebForm1 : System.Web.UI.Page

    {

        protected void Page\_Load(object sender, EventArgs e)

        { }

        protected void Button1\_Click(object sender, EventArgs e)

        {

            try

            {

                string cs = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;

                using (SqlConnection con = new SqlConnection(cs))

                {

                    SqlCommand cmd = new SqlCommand("spTransaction1", con);

                    cmd.CommandType = CommandType.StoredProcedure;

                    con.Open();

                    cmd.ExecuteNonQuery();

                    Label1.Text = "Transaction successful";

                    Label1.ForeColor = System.Drawing.Color.Green;

                }

            }

            catch (SqlException ex)

            {

                if (ex.Number == 1205)

                {

                    Label1.Text = "Deadlock. Please retry";

                }

                else

                {

                    Label1.Text = ex.Message;

                }

                Label1.ForeColor = System.Drawing.Color.Red;

            }

        }

    }

}

**WebForm2.aspx HTML**

<table>

    <tr>

        <td>

            <asp:Button ID="Button1" runat="server"

                Text="Update Table B and then Table A"

                OnClick="Button1\_Click" />

        </td>

    </tr>

    <tr>

        <td>

            <asp:Label ID="Label1" runat="server"></asp:Label>

        </td>

    </tr>

</table>

**WebForm2.aspx.cs code**

using System;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

namespace Demo

{

    public partial class WebForm1 : System.Web.UI.Page

    {

        protected void Page\_Load(object sender, EventArgs e)

        { }

        protected void Button1\_Click(object sender, EventArgs e)

        {

            try

            {

                string cs = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;

                using (SqlConnection con = new SqlConnection(cs))

                {

                    SqlCommand cmd = new SqlCommand("spTransaction1", con);

                    cmd.CommandType = CommandType.StoredProcedure;

                    con.Open();

                    cmd.ExecuteNonQuery();

                    Label1.Text = "Transaction successful";

                    Label1.ForeColor = System.Drawing.Color.Green;

                }

            }

            catch (SqlException ex)

            {

                if (ex.Number == 1205)

                {

                    Label1.Text = "Deadlock. Please retry";

                }

                else

                {

                    Label1.Text = ex.Message;

                }

                Label1.ForeColor = System.Drawing.Color.Red;

            }

        }

    }

}

# 85. Retry logic for deadlock exceptions

**Suggested Videos**  
[Part 82 - Capturing deadlocks in SQL profiler](http://csharp-video-tutorials.blogspot.com/2015/08/capturing-deadlocks-in-sql-profiler.html)  
[Part 83 - SQL Server deadlock error handling](http://csharp-video-tutorials.blogspot.com/2015/08/sql-server-deadlock-error-handling.html)   
[Part 84 - Handling deadlocks in ado.net](http://csharp-video-tutorials.blogspot.com/2015/08/handling-deadlocks-in-adonet.html)   
  
  
  
In this video we will discuss implementing **retry logic for deadlock exceptions**.    
  
  
  
This is continuation to [Part 84](http://csharp-video-tutorials.blogspot.com/2015/08/handling-deadlocks-in-adonet.html). Please watch [Part 84](http://csharp-video-tutorials.blogspot.com/2015/08/handling-deadlocks-in-adonet.html), before proceeding.   
  
When a transaction fails due to deadlock, we can write some logic so the system can resubmit the transaction. The deadlocks usually last for a very short duration. So upon resubmitting the transaction it may complete successfully. This is much better from user experience standpoint.  
  
To achieve this we will be using the following technologies  
C#  
ASP.NET  
SQL Server  
jQuery AJAX  
  
**Result.cs**

public class Result

{

    public int AttemptsLeft { get; set; }

    public string Message { get; set; }

    public bool Success { get; set; }

}

**WebForm1.aspx HTML and jQuery code**

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

    <title></title>

    <script src="jquery-1.11.2.js"></script>

    <script type="text/javascript">

        $(document).ready(function () {

            var lblMessage = $('#Label1');

            var attemptsLeft;

            function updateData() {

                $.ajax({

                    url: 'WebForm1.aspx/CallStoredProcedure',

                    method: 'post',

                    contentType: 'application/json',

                    data: '{attemptsLeft:' + attemptsLeft + '}',

                    dataType: 'json',

                    success: function (data) {

                        lblMessage.text(data.d.Message);

                        attemptsLeft = data.d.AttemptsLeft;

                        if (data.d.Success) {

                            $('#btn').prop('disabled', false);

                            lblMessage.css('color','green');

                        }

                        else if(attemptsLeft > 0){

                            lblMessage.css('color', 'red');

                            updateData();

                        }

                        else {

                            lblMessage.css('color', 'red');

                            lblMessage.text('Deadlock Occurred. ZERO attempts left. Please try later');

                        }

                    },

                    error: function (err) {

                        lblMessage.css('color', 'red');

                        lblMessage.text(err.responseText);

                    }

                });

            }

            $('#btn').click(function () {

                $(this).prop('disabled', true);

                lblMessage.text('Updating....');

                attemptsLeft = 5;

                updateData();

            });

        });

    </script>

</head>

<body style="font-family: Arial">

    <form id="form1" runat="server">

        <input id="btn" type="button"

            value="Update Table A and then Table B" />

        <br />

        <asp:Label ID="Label1" runat="server"></asp:Label>

    </form>

</body>

</html>

**WebForm1.aspx.cs code**

using System;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

namespace Demo

{

    public partial class WebForm1 : System.Web.UI.Page

    {

        protected void Page\_Load(object sender, EventArgs e)

        { }

        [System.Web.Services.WebMethod]

        public static Result CallStoredProcedure(int attemptsLeft)

        {

            Result \_result = new Result();

            if (attemptsLeft > 0)

            {

                try

                {

                    string cs =ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;

                    using (SqlConnection con = new SqlConnection(cs))

                    {

                        SqlCommand cmd = new SqlCommand("spTransaction15", con);

                        cmd.CommandType = CommandType.StoredProcedure;

                        con.Open();

                        cmd.ExecuteNonQuery();

                        \_result.Message = "Transaction successful";

                        \_result.AttemptsLeft = 0;

                        \_result.Success = true;

                    }

                }

                catch (SqlException ex)

                {

                    if (ex.Number == 1205)

                    {

                        \_result.AttemptsLeft = attemptsLeft - 1;

                        \_result.Message = "Deadlock occurred. Retrying. Attempts left : "

                            + \_result.AttemptsLeft.ToString();

                    }

                    else

                    {

                        throw;

                    }

                    \_result.Success = false;

                }

            }

            return \_result;

        }

    }

}

Copy and paste the above code in WebForm2.aspx and make the required changes as described in the video.

# 86. How to find blocking queries in sql server

**Suggested Videos**  
[Part 83 - SQL Server deadlock error handling](http://csharp-video-tutorials.blogspot.com/2015/08/sql-server-deadlock-error-handling.html)  
[Part 84 - Handling deadlocks in ado.net](http://csharp-video-tutorials.blogspot.com/2015/08/handling-deadlocks-in-adonet.html)   
[Part 85 - Retry logic for deadlock exceptions](http://csharp-video-tutorials.blogspot.com/2015/08/retry-logic-for-deadlock-exceptions.html)   
  
  
  
In this video we will discuss, **how to find blocking queries in sql server**.   
  
  
  
Blocking occurs if there are open transactions. Let us understand this with an example.   
  
**Execute the following 2 sql statements**  
Begin Tran  
Update TableA set Name='Mark Transaction 1' where Id = 1   
  
Now from a different window, execute any of the following commands. Notice that all the queries are blocked.

Select Count(\*) from TableA

Delete from TableA where Id = 1

Truncate table TableA

Drop table TableA

This is because there is an open transaction. Once the open transaction completes, you will be able to execute the above queries.   
  
So the obvious next question is - **How to identify all the active transactions**.    
  
One way to do this is by using DBCC OpenTran. DBCC OpenTran will display only the oldest active transaction. It is not going to show you all the open transactions.  
DBCC OpenTran   
  
The following link has the SQL script that you can use to identify all the active transactions.   
<http://www.sqlskills.com/blogs/paul/script-open-transactions-with-text-and-plans>   
  
The beauty about this script is that it has a lot more useful information about the open transactions  
Session Id  
Login Name  
Database Name  
Transaction Begin Time  
The actual query that is executed   
  
You can now use this information and ask the respective developer to either commit or rollback the transactions that they have left open unintentionally.   
  
For some reason if the person who initiated the transaction is not available, you also have the option to KILL the associated process. However, this may have unintended consequences, so use it with extreme caution.   
  
There are 2 ways to kill the process are described below  
  
**Killing the process using SQL Server Activity Monitor :**   
1. Right Click on the Server Name in Object explorer and select **"Activity Monitor"**  
2. In the **"Activity Monitor"** window expand Processes section  
3. Right click on the associated **"Session ID"** and select **"Kill Process"** from the context menu   
  
**Killing the process using SQL command :**  
KILL Process\_ID   
  
**What happens when you kill a session**  
All the work that the transaction has done will be rolled back. The database must be put back in the state it was in, before the transaction started.

# 87. SQL Server except operator

**Suggested Videos**  
[Part 84 - Handling deadlocks in ado.net](http://csharp-video-tutorials.blogspot.com/2015/08/handling-deadlocks-in-adonet.html)  
[Part 85 - Retry logic for deadlock exceptions](http://csharp-video-tutorials.blogspot.com/2015/08/retry-logic-for-deadlock-exceptions.html)   
[Part 86 - How to find blocking queries in sql server](http://csharp-video-tutorials.blogspot.com/2015/09/how-to-find-blocking-queries-in-sql.html)   
  
  
  
In this video we will discuss **SQL Server except operator with examples.**  
  
  
  
**EXCEPT operator** returns unique rows from the left query that aren’t in the right query’s results.

* Introduced in SQL Server 2005
* The number and the order of the columns must be the same in all queries
* The data types must be same or compatible
* This is similar to minus operator in oracle

Let us understand this with an example. We will use the following 2 tables for this example.

|  |  |
| --- | --- |
| http://2.bp.blogspot.com/-H0sALFpqm1c/VenyNidzfTI/AAAAAAAAd3s/p5VK_jhqaMg/s1600/Table%2BA.png | http://3.bp.blogspot.com/-e4qIE5tJBGs/Venyb0MPQJI/AAAAAAAAd30/pN_5ahniMhI/s1600/Table%2BB.png |

**SQL Script to create the tables**

Create Table TableA

(

    Id int primary key,

    Name nvarchar(50),

    Gender nvarchar(10)

)

Go

Insert into TableA values (1, 'Mark', 'Male')

Insert into TableA values (2, 'Mary', 'Female')

Insert into TableA values (3, 'Steve', 'Male')

Insert into TableA values (4, 'John', 'Male')

Insert into TableA values (5, 'Sara', 'Female')

Go

Create Table TableB

(

    Id int primary key,

    Name nvarchar(50),

    Gender nvarchar(10)

)

Go

Insert into TableB values (4, 'John', 'Male')

Insert into TableB values (5, 'Sara', 'Female')

Insert into TableB values (6, 'Pam', 'Female')

Insert into TableB values (7, 'Rebeka', 'Female')

Insert into TableB values (8, 'Jordan', 'Male')

Go

Notice that the following query returns the unique rows from the left query that aren’t in the right query’s results.

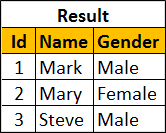
Select Id, Name, Gender

From TableA

Except

Select Id, Name, Gender

From TableB

**Result :**   
   
  
To retrieve all of the rows from Table B that does not exist in Table A, reverse the two queries as shown below.

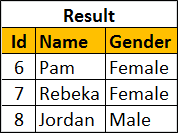
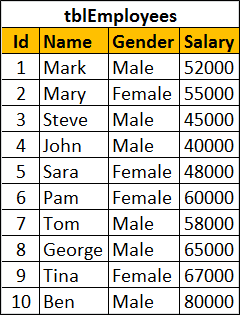
Select Id, Name, Gender

From TableB

Except

Select Id, Name, Gender

From TableA

**Result :**   
   
  
You can also use Except operator on a single table. Let's use the following tblEmployees table for this example.   
   
  
SQL script to create tblEmployees table

Create table tblEmployees

(

    Id int identity primary key,

    Name nvarchar(100),

    Gender nvarchar(10),

    Salary int

)

Go

Insert into tblEmployees values ('Mark', 'Male', 52000)

Insert into tblEmployees values ('Mary', 'Female', 55000)

Insert into tblEmployees values ('Steve', 'Male', 45000)

Insert into tblEmployees values ('John', 'Male', 40000)

Insert into tblEmployees values ('Sara', 'Female', 48000)

Insert into tblEmployees values ('Pam', 'Female', 60000)

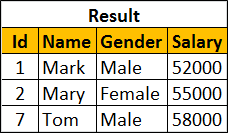
Insert into tblEmployees values ('Tom', 'Male', 58000)

Insert into tblEmployees values ('George', 'Male', 65000)

Insert into tblEmployees values ('Tina', 'Female', 67000)

Insert into tblEmployees values ('Ben', 'Male', 80000)

Go

**Result :**   
   
  
**Order By clause should be used only once after the right query**

Select Id, Name, Gender, Salary

From tblEmployees

Where Salary >= 50000

Except

Select Id, Name, Gender, Salary

From tblEmployees

Where Salary >= 60000

order By Name

# 88. Difference between except and not in sql server

**Suggested Videos**  
[Part 85 - Retry logic for deadlock exceptions](http://csharp-video-tutorials.blogspot.com/2015/08/retry-logic-for-deadlock-exceptions.html)  
[Part 86 - How to find blocking queries in sql server](http://csharp-video-tutorials.blogspot.com/2015/09/how-to-find-blocking-queries-in-sql.html)   
[Part 87 - SQL Server except operator](http://csharp-video-tutorials.blogspot.com/2015/09/sql-server-except-operator.html)   
  
  
  
In this video we will discuss the **difference between EXCEPT and NOT IN operators in SQL Server**.   
  
  
  
We will use the following 2 tables for this example.

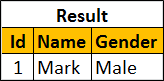
|  |  |
| --- | --- |
| http://2.bp.blogspot.com/-sa3kuHvlhJE/Ves5ZsDDQxI/AAAAAAAAd5o/ZiJORt2qQfg/s1600/Table%2BA.png | http://2.bp.blogspot.com/-sx2Bc67CPrs/Ves5hwogNbI/AAAAAAAAd5w/bOLsicEVeIU/s1600/Table%2BB.png |

The following query returns the rows from the left query that aren’t in the right query’s results.

Select Id, Name, Gender From TableA

Except

Select Id, Name, Gender From TableB

**Result :**   
   
  
**The same result can also be achieved using NOT IN operator.**

Select Id, Name, Gender From TableA

Where Id NOT IN (Select Id from TableB)

**So, what is the difference between EXCEPT and NOT IN operators**  
1. Except filters duplicates and returns only DISTINCT rows from the left query that aren’t in the right query’s results, where as NOT IN does not filter the duplicates.  
  
Insert the following row into TableA

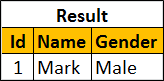
Insert into TableA values (1, 'Mark', 'Male')

Now execute the following EXCEPT query. Notice that we get only the DISTINCT rows

Select Id, Name, Gender From TableA

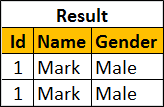
Except

Select Id, Name, Gender From TableB

Result:   
   
  
Now execute the following query. Notice that the duplicate rows are not filtered.

Select Id, Name, Gender From TableA

Where Id NOT IN (Select Id from TableB)

Result:   
   
  
2. EXCEPT operator expects the same number of columns in both the queries, where as NOT IN, compares a single column from the outer query with a single column from the subquery.  
  
In the following example, the number of columns are different.

Select Id, Name, Gender From TableA

Except

Select Id, Name From TableB

The above query would produce the following error.  
Msg 205, Level 16, State 1, Line 1  
All queries combined using a UNION, INTERSECT or EXCEPT operator must have an equal number of expressions in their target lists.  
  
NOT IN, compares a single column from the outer query with a single column from subquery.   
  
In the following example, the subquery returns multiple columns

Select Id, Name, Gender From TableA

Where Id NOT IN (Select Id, Name from TableB)

Msg 116, Level 16, State 1, Line 2  
Only one expression can be specified in the select list when the subquery is not introduced with EXISTS.

# 89. Intersect operator in sql server

**Suggested Videos**  
[Part 86 - How to find blocking queries in sql server](http://csharp-video-tutorials.blogspot.com/2015/09/how-to-find-blocking-queries-in-sql.html)  
[Part 87 - SQL Server except operator](http://csharp-video-tutorials.blogspot.com/2015/09/sql-server-except-operator.html)   
[Part 88 - Difference between except and not in sql server](http://csharp-video-tutorials.blogspot.com/2015/09/difference-between-except-and-not-in.html)   
  
  
  
**In this video we will discuss**  
1. Intersect operator in sql server  
2. Difference between intersect and inner join   
  
  
  
**Intersect operator retrieves the common records from both the left and the right query of the Intersect operator.**

* Introduced in SQL Server 2005
* The number and the order of the columns must be same in both the queries
* The data types must be same or at least compatible

Let us understand INTERSECT operator with an example.   
  
We will use the following 2 tables for this example.

|  |  |
| --- | --- |
| http://4.bp.blogspot.com/-Mc7Xw7RolFE/VexzNsl9n-I/AAAAAAAAd7A/IvZRCYKICDs/s1600/Table%2BA.png | http://4.bp.blogspot.com/-rF2mc6VNMRM/VexzXRo989I/AAAAAAAAd7I/rFs99jziFg0/s1600/Table%2BB.png |

SQL Script to create the tables and populate with test data

Create Table TableA

(

    Id int,

    Name nvarchar(50),

    Gender nvarchar(10)

)

Go

Insert into TableA values (1, 'Mark', 'Male')

Insert into TableA values (2, 'Mary', 'Female')

Insert into TableA values (3, 'Steve', 'Male')

Go

Create Table TableB

(

    Id int,

    Name nvarchar(50),

    Gender nvarchar(10)

)

Go

Insert into TableB values (2, 'Mary', 'Female')

Insert into TableB values (3, 'Steve', 'Male')

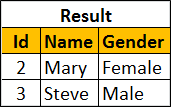
Go

The following query retrieves the common records from both the left and the right query of the Intersect operator.

Select Id, Name, Gender from TableA

Intersect

Select Id, Name, Gender from TableB

**Result :**   
   
  
We can also achieve the same thinkg using INNER join. The following INNER join query would produce the exact same result.

Select TableA.Id, TableA.Name, TableA.Gender

From TableA Inner Join TableB

On TableA.Id = TableB.Id

**What is the difference between INTERSECT and INNER JOIN**  
1. INTERSECT filters duplicates and returns only DISTINCT rows that are common between the LEFT and Right Query, where as INNER JOIN does not filter the duplicates.   
  
To understand this difference, insert the following row into TableA

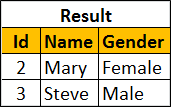
Insert into TableA values (2, 'Mary', 'Female')

Now execute the following INTERSECT query. Notice that we get only the DISTINCT rows

Select Id, Name, Gender from TableA

Intersect

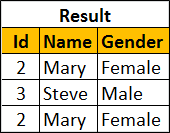
Select Id, Name, Gender from TableB

**Result :**  
   
  
Now execute the following INNER JOIN query. Notice that the duplicate rows are not filtered.

Select TableA.Id, TableA.Name, TableA.Gender

From TableA Inner Join TableB

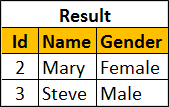
On TableA.Id = TableB.Id

**Result :**    
   
  
You can make the INNER JOIN behave like INTERSECT operator by using the DISTINCT operator

Select DISTINCT TableA.Id, TableA.Name, TableA.Gender

From TableA Inner Join TableB

On TableA.Id = TableB.Id

**Result :**   
   
  
**2. INNER JOIN treats two NULLS as two different values**. So if you are joining two tables based on a nullable column and if both tables have NULLs in that joining column then, INNER JOIN will not include those rows in the result-set, where as INTERSECT treats two NULLs as a same value and it returns all matching rows.  
  
To understand this difference, execute the following 2 insert statements

Insert into TableA values(NULL, 'Pam', 'Female')

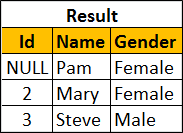
Insert into TableB values(NULL, 'Pam', 'Female')

**INTERSECT query**

Select Id, Name, Gender from TableA

Intersect

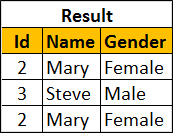
Select Id, Name, Gender from TableB

**Result :**   
   
  
**INNER JOIN query**

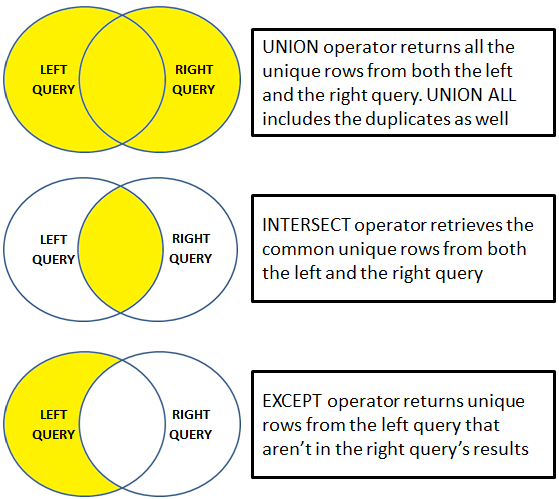
Select TableA.Id, TableA.Name, TableA.Gender

From TableA Inner Join TableB

On TableA.Id = TableB.Id

**Result :**   


# 90. Difference between union intersect and except in sql server

**Suggested Videos**  
[Part 87 - SQL Server except operator](http://csharp-video-tutorials.blogspot.com/2015/09/sql-server-except-operator.html)  
[Part 88 - Difference between except and not in sql server](http://csharp-video-tutorials.blogspot.com/2015/09/difference-between-except-and-not-in.html)   
[Part 89 - Intersect operator in sql server](http://csharp-video-tutorials.blogspot.com/2015/09/intersect-operator-in-sql-server.html)   
  
  
  
In this video we will discuss the **difference between union intersect and except in sql server with examples**.   
  
  
  
The following diagram explains the difference graphically   
   
  
UNION operator returns all the unique rows from both the left and the right query. UNION ALL included the duplicates as well.   
  
INTERSECT operator retrieves the common unique rows from both the left and the right query.   
  
EXCEPT operator returns unique rows from the left query that aren’t in the right query’s results.    
  
Let us understand these differences with examples. We will use the following 2 tables for the examples.

|  |  |
| --- | --- |
| http://2.bp.blogspot.com/-1YvhZANjCPM/Ve3pvnCaVAI/AAAAAAAAd9E/aE39Q_HNB08/s1600/Table%2BA.png | http://1.bp.blogspot.com/-Z3fMMdyHheg/Ve3p2IYPm_I/AAAAAAAAd9M/VMlBeBSU1Xg/s1600/Table%2BB.png |

**SQL Script to create the tables**

Create Table TableA

(

    Id int,

    Name nvarchar(50),

    Gender nvarchar(10)

)

Go

Insert into TableA values (1, 'Mark', 'Male')

Insert into TableA values (2, 'Mary', 'Female')

Insert into TableA values (3, 'Steve', 'Male')

Insert into TableA values (3, 'Steve', 'Male')

Go

Create Table TableB

(

    Id int primary key,

    Name nvarchar(50),

    Gender nvarchar(10)

)

Go

Insert into TableB values (2, 'Mary', 'Female')

Insert into TableB values (3, 'Steve', 'Male')

Insert into TableB values (4, 'John', 'Male')

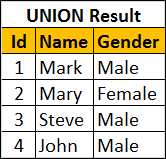
Go

UNION operator returns all the unique rows from both the queries. Notice the duplicates are removed.

Select Id, Name, Gender from TableA

UNION

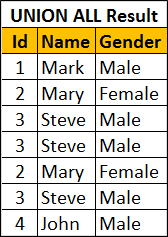
Select Id, Name, Gender from TableB

**Result :**   
   
  
UNION ALL operator returns all the rows from both the queries, including the duplicates.

Select Id, Name, Gender from TableA

UNION ALL

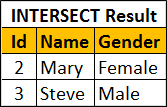
Select Id, Name, Gender from TableB

**Result :**   
   
  
INTERSECT operator retrieves the common unique rows from both the left and the right query. Notice the duplicates are removed.

Select Id, Name, Gender from TableA

INTERSECT

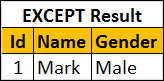
Select Id, Name, Gender from TableB

**Result :**   
   
  
EXCEPT operator returns unique rows from the left query that aren’t in the right query’s results.

Select Id, Name, Gender from TableA

EXCEPT

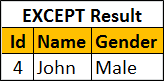
Select Id, Name, Gender from TableB

**Result :**   
   
  
If you wnat the rows that are present in Table B but not in Table A, reverse the queries.

Select Id, Name, Gender from TableB

EXCEPT

Select Id, Name, Gender from TableA

Result :   
   
  
**For all these 3 operators to work the following 2 conditions must be met**

* The number and the order of the columns must be same in both the queries
* The data types must be same or at least compatible

For example, if the number of columns are different, you will get the following error  
Msg 205, Level 16, State 1, Line 1  
All queries combined using a UNION, INTERSECT or EXCEPT operator must have an equal number of expressions in their target lists.

# 91. Cross apply and outer apply in sql server

**Suggested Videos**  
[Part 88 - Difference between except and not in sql server](http://csharp-video-tutorials.blogspot.com/2015/09/difference-between-except-and-not-in.html)  
[Part 89 - Intersect operator in sql server](http://csharp-video-tutorials.blogspot.com/2015/09/intersect-operator-in-sql-server.html)   
[Part 90 - Difference between union intersect and except in sql server](http://csharp-video-tutorials.blogspot.com/2015/09/difference-between-union-intersect-and.html)   
  
  
  
In this video we will discuss **cross apply and outer apply in sql server** with examples.   
  
  
  
We will use the following 2 tables for examples in this demo

|  |  |
| --- | --- |
| department table | employee table |

SQL Script to create the tables and populate with test data

Create table Department

(

    Id int primary key,

    DepartmentName nvarchar(50)

)

Go

Insert into Department values (1, 'IT')

Insert into Department values (2, 'HR')

Insert into Department values (3, 'Payroll')

Insert into Department values (4, 'Administration')

Insert into Department values (5, 'Sales')

Go

Create table Employee

(

    Id int primary key,

    Name nvarchar(50),

    Gender nvarchar(10),

    Salary int,

    DepartmentId int foreign key references Department(Id)

)

Go

Insert into Employee values (1, 'Mark', 'Male', 50000, 1)

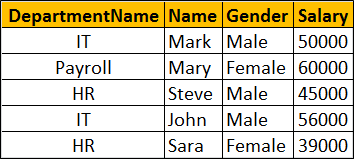
Insert into Employee values (2, 'Mary', 'Female', 60000, 3)

Insert into Employee values (3, 'Steve', 'Male', 45000, 2)

Insert into Employee values (4, 'John', 'Male', 56000, 1)

Insert into Employee values (5, 'Sara', 'Female', 39000, 2)

Go

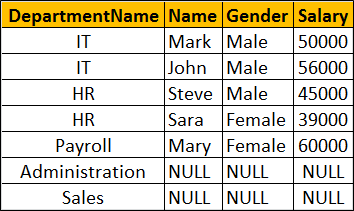
We want to retrieve all the matching rows between **Department**and **Employee**tables.   
   
  
This can be very easily achieved using an Inner Join as shown below.

Select D.DepartmentName, E.Name, E.Gender, E.Salary

from Department D

Inner Join Employee E

On D.Id = E.DepartmentId

Now if we want to retrieve all the matching rows between **Department**and **Employee**tables + the non-matching rows from the LEFT table (**Department**)   
   
  
This can be very easily achieved using a Left Join as shown below.

Select D.DepartmentName, E.Name, E.Gender, E.Salary

from Department D

Left Join Employee E

On D.Id = E.DepartmentId

Now let's assume we do not have access to the Employee table. Instead we have access to the following Table Valued function, that returns all employees belonging to a department by Department Id.

Create function fn\_GetEmployeesByDepartmentId(@DepartmentId int)

Returns Table

as

Return

(

    Select Id, Name, Gender, Salary, DepartmentId

    from Employee where DepartmentId = @DepartmentId

)

Go

The following query returns the employees of the department with Id =1.

Select \* from fn\_GetEmployeesByDepartmentId(1)

Now if you try to perform an Inner or Left join between **Department**table and**fn\_GetEmployeesByDepartmentId**() function you will get an error.

Select D.DepartmentName, E.Name, E.Gender, E.Salary

from Department D

Inner Join fn\_GetEmployeesByDepartmentId(D.Id) E

On D.Id = E.DepartmentId

If you execute the above query you will get the following error  
Msg 4104, Level 16, State 1, Line 3  
The multi-part identifier "D.Id" could not be bound.   
  
This is where we use **Cross Apply** and **Outer Apply** operators. **Cross Apply** is semantically equivalent to **Inner Join**and **Outer Apply**is semantically equivalent to **Left Outer Join**.   
  
Just like Inner Join, Cross Apply retrieves only the matching rows from the Department table and fn\_GetEmployeesByDepartmentId() table valued function.

Select D.DepartmentName, E.Name, E.Gender, E.Salary

from Department D

Cross Apply fn\_GetEmployeesByDepartmentId(D.Id) E

Just like Left Outer Join, Outer Apply retrieves all matching rows from the Department table and fn\_GetEmployeesByDepartmentId() table valued function + non-matching rows from the left table (Department)

Select D.DepartmentName, E.Name, E.Gender, E.Salary

from Department D

Outer Apply fn\_GetEmployeesByDepartmentId(D.Id) E

**How does Cross Apply and Outer Apply work**

* The APPLY operator introduced in SQL Server 2005, is used to join a table to a table-valued function.
* The Table Valued Function on the right hand side of the APPLY operator gets called for each row from the left (also called outer table) table.
* Cross Apply returns only matching rows (semantically equivalent to Inner Join)
* Outer Apply returns matching + non-matching rows (semantically equivalent to Left Outer Join). The unmatched columns of the table valued function will be set to NULL.

# 92. DDL Triggers in sql server

**Suggested Videos**  
[Part 89 - Intersect operator in sql server](http://csharp-video-tutorials.blogspot.com/2015/09/intersect-operator-in-sql-server.html)  
[Part 90 - Difference between union intersect and except in sql server](http://csharp-video-tutorials.blogspot.com/2015/09/difference-between-union-intersect-and.html)   
[Part 91 - Cross apply and outer apply in sql server](http://csharp-video-tutorials.blogspot.com/2015/09/cross-apply-and-outer-apply-in-sql.html)   
  
  
  
In this video we will discuss **DDL Triggers in sql server**.   
  
  
  
**In SQL Server there are 4 types of triggers**   
**1.** DML Triggers - Data Manipulation Language. Discussed in Parts 43 to 47 of [SQL Server](https://www.youtube.com/playlist?list=PL08903FB7ACA1C2FB) Tutorial.  
**2.** DDL Triggers - Data Definition Language  
**3.** CLR triggers - Common Language Runtime  
**4.** Logon triggers  
  
**What are DDL triggers**  
**DDL triggers fire in response to DDL events** - CREATE, ALTER, and DROP (Table, Function, Index, Stored Procedure etc...). For the list of all DDL events please visit https://msdn.microsoft.com/en-us/library/bb522542.aspx   
  
**Certain system stored procedures** that perform DDL-like operations can also fire DDL triggers. Example - sp\_rename system stored procedure   
  
**What is the use of DDL triggers**

* If you want to execute some code in response to a specific DDL event
* To prevent certain changes to your database schema
* Audit the changes that the users are making to the database structure

**Syntax for creating DDL trigger**

CREATE TRIGGER [Trigger\_Name]

ON [Scope (Server|Database)]

FOR [EventType1, EventType2, EventType3, ...],

AS

BEGIN

   -- Trigger Body

END

**DDL triggers scope :** DDL triggers can be created in a specific database or at the server level.    
  
**The following trigger will fire in response to CREATE\_TABLE DDL event.**

CREATE TRIGGER trMyFirstTrigger

ON Database

FOR CREATE\_TABLE

AS

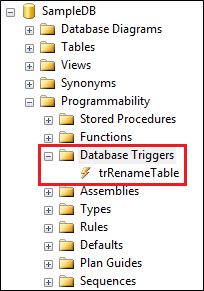
BEGIN

   Print 'New table created'

END

**To check if the trigger has been created**

1. In the Object Explorer window, expand the **SampleDB**database by clicking on the plus symbol.
2. Expand **Programmability**folder
3. Expand **Database Triggers** folder

   
  
**Please note :** If you can't find the trigger that you just created, make sure to refresh the Database Triggers folder.  
  
When you execute the following code to create the table, the trigger will automatically fire and will print the message - New table created  
Create Table Test (Id int)  
  
The above trigger will be fired only for one DDL event CREATE\_TABLE. If you want this trigger to be fired for multiple events, for example when you alter or drop a table, then separate the events using a comma as shown below.

ALTER TRIGGER trMyFirstTrigger

ON Database

FOR CREATE\_TABLE, ALTER\_TABLE, DROP\_TABLE

AS

BEGIN

   Print 'A table has just been created, modified or deleted'

END

Now if you create, alter or drop a table, the trigger will fire automatically and you will get the message - A table has just been created, modified or deleted.    
  
The 2 DDL triggers above execute some code in response to DDL events  
  
Now let us look at an example of how to prevent users from creating, altering or dropping tables. To do this modify the trigger as shown below.

ALTER TRIGGER trMyFirstTrigger

ON Database

FOR CREATE\_TABLE, ALTER\_TABLE, DROP\_TABLE

AS

BEGIN

   Rollback

   Print 'You cannot create, alter or drop a table'

END

To be able to create, alter or drop a table, you either have to disable or delete the trigger.   
  
**To disable trigger**  
**1.** Right click on the trigger in object explorer and select **"Disable"**from the context menu   
**2.** You can also disable the trigger using the following T-SQL command

DISABLE TRIGGER trMyFirstTrigger ON DATABASE

**To enable trigger**  
**1.** Right click on the trigger in object explorer and select "Enable" from the context menu   
**2.** You can also enable the trigger using the following T-SQL command  
ENABLE TRIGGER trMyFirstTrigger ON DATABASE  
  
**To drop trigger**  
**1.** Right click on the trigger in object explorer and select "Delete" from the context menu   
**2.** You can also drop the trigger using the following T-SQL command  
DROP TRIGGER trMyFirstTrigger ON DATABASE  
  
Certain system stored procedures that perform DDL-like operations can also fire DDL triggers. The following trigger will be fired when ever you rename a database object usingsp\_rename system stored procedure.

CREATE TRIGGER trRenameTable

ON DATABASE

FOR RENAME

AS

BEGIN

    Print 'You just renamed something'

END

The following code changes the name of the TestTable to NewTestTable. When this code is executed, it will fire the trigger trRenameTable

sp\_rename 'TestTable', 'NewTestTable'

The following code changes the name of the Id column in NewTestTable to NewId. When this code is executed, it will fire the trigger trRenameTable

sp\_rename 'NewTestTable.Id' , 'NewId', 'column'

# 93. Server-scoped ddl triggers

**Suggested Videos**  
[Part 90 - Difference between union intersect and except in sql server](http://csharp-video-tutorials.blogspot.com/2015/09/difference-between-union-intersect-and.html)  
[Part 91 - Cross apply and outer apply in sql server](http://csharp-video-tutorials.blogspot.com/2015/09/cross-apply-and-outer-apply-in-sql.html)   
[Part 92 - DDL Triggers in sql server](http://csharp-video-tutorials.blogspot.com/2015/09/ddl-triggers-in-sql-server.html)   
  
  
  
In this video we will discuss **server-scoped ddl triggers**  
  
  
  
The following trigger is a database scoped trigger. This will prevent users from creating, altering or dropping tables only from the database in which it is created.

CREATE TRIGGER tr\_DatabaseScopeTrigger

ON DATABASE

FOR CREATE\_TABLE, ALTER\_TABLE, DROP\_TABLE

AS

BEGIN

    ROLLBACK

    Print 'You cannot create, alter or drop a table in the current database'

END

If you have another database on the server, they will be able to create, alter or drop tables in that database. If you want to prevent users from doing this you may create the trigger again in this database.   
  
**But, what if you have 100 different databases on your SQL Server**, and you want to prevent users from creating, altering or dropping tables from all these 100 databases. Creating the same trigger for all the 100 different databases is not a good approach for 2 reasons.  
1. It is tedious and error prone  
2. Maintainability is a night mare. If for some reason you have to change the trigger, you will have to do it in 100 different databases, which again is tedious and error prone.  
  
This is where server-scoped DDL triggers come in handy. When you create a server scoped DDL trigger, it will fire in response to the DDL events happening in all of the databases on that server.    
  
**Creating a Server-scoped DDL trigger :** Similar to creating a database scoped trigger, except that you will have to change the scope to ALL Server as shown below.

CREATE TRIGGER tr\_ServerScopeTrigger

ON ALL SERVER

FOR CREATE\_TABLE, ALTER\_TABLE, DROP\_TABLE

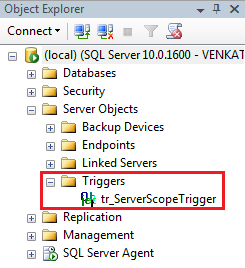
AS

BEGIN

    ROLLBACK

    Print 'You cannot create, alter or drop a table in any database on the server'

END

Now if you try to create, alter or drop a table in any of the databases on the server, the trigger will be fired.  
  
**Where can I find the Server-scoped DDL triggers**  
**1.** In the Object Explorer window, expand "Server Objects" folder  
**2.** Expand Triggers folder   
   
  
**To disable Server-scoped ddl trigger**  
1. Right click on the trigger in object explorer and select "Disable" from the context menu   
2. You can also disable the trigger using the following T-SQL command

DISABLE TRIGGER tr\_ServerScopeTrigger ON ALL SERVER

**To enable Server-scoped ddl trigger**  
1. Right click on the trigger in object explorer and select "Enable" from the context menu   
2. You can also enable the trigger using the following T-SQL command  
ENABLE TRIGGER tr\_ServerScopeTrigger ON ALL SERVER   
  
**To drop Server-scoped ddl trigger**  
1. Right click on the trigger in object explorer and select "Delete" from the context menu   
2. You can also drop the trigger using the following T-SQL command  
DROP TRIGGER tr\_ServerScopeTrigger ON ALL SERVER

# 94. sql server trigger execution order

**Suggested Videos**  
[Part 91 - Cross apply and outer apply in sql server](http://csharp-video-tutorials.blogspot.com/2015/09/cross-apply-and-outer-apply-in-sql.html)  
[Part 92 - DDL Triggers in sql server](http://csharp-video-tutorials.blogspot.com/2015/09/ddl-triggers-in-sql-server.html)   
[Part 93 - Server-scoped ddl triggers](http://csharp-video-tutorials.blogspot.com/2015/09/server-scoped-ddl-triggers.html)   
  
  
  
In this video we will discuss **how to set the execution order of triggers** using**sp\_settriggerorder**stored procedure.   
  
  
  
**Server scoped triggers will always fire before any of the database scoped triggers**. This execution order cannot be changed.   
  
In the example below, we have a database-scoped and a server-scoped trigger handling the same event (CREATE\_TABLE). When you create a table, notice that server-scoped trigger is always fired before the database-scoped trigger.

CREATE TRIGGER tr\_DatabaseScopeTrigger

ON DATABASE

FOR CREATE\_TABLE

AS

BEGIN

    Print 'Database Scope Trigger'

END

GO

CREATE TRIGGER tr\_ServerScopeTrigger

ON ALL SERVER

FOR CREATE\_TABLE

AS

BEGIN

    Print 'Server Scope Trigger'

END

GO

Using the **sp\_settriggerorder**stored procedure, you can set the execution order of server-scoped or database-scoped triggers.   
  
**sp\_settriggerorder stored procedure has 4 parameters**

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| @triggername | Name of the trigger |
| @order | Value can be First, Last or None. When set to None, trigger is fired in random order |
| @stmttype | SQL statement that fires the trigger. Can be INSERT, UPDATE, DELETE or any DDL event |
| @namespace | Scope of the trigger. Value can be DATABASE, SERVER, or NULL |

EXEC sp\_settriggerorder

@triggername = 'tr\_DatabaseScopeTrigger1',

@order = 'none',

@stmttype = 'CREATE\_TABLE',

@namespace = 'DATABASE'

GO

**If you have a database-scoped and a server-scoped trigger handling the same event**, and if you have set the execution order at both the levels. Here is the execution order of the triggers.  
1. The server-scope trigger marked First  
2. Other server-scope triggers  
3. The server-scope trigger marked Last  
4. The database-scope trigger marked First  
5. Other database-scope triggers  
6. The database-scope trigger marked Last

# 95. Audit table changes in sql server

**Suggested Videos**  
[Part 92 - DDL Triggers in sql server](http://csharp-video-tutorials.blogspot.com/2015/09/ddl-triggers-in-sql-server.html)  
[Part 93 - Server-scoped ddl triggers](http://csharp-video-tutorials.blogspot.com/2015/09/server-scoped-ddl-triggers.html)   
[Part 94 - SQL Server trigger execution order](http://csharp-video-tutorials.blogspot.com/2015/09/sql-server-trigger-execution-order.html)   
  
  
  
In this video we will discuss, **how to audit table changes in SQL Server using a DDL trigger**.   
  
  
  
**Table to store the audit data**

Create table TableChanges

(

    DatabaseName nvarchar(250),

    TableName nvarchar(250),

    EventType nvarchar(250),

    LoginName nvarchar(250),

    SQLCommand nvarchar(2500),

    AuditDateTime datetime

)

Go

**The following trigger audits all table changes in all databases on a SQL Server**

CREATE TRIGGER tr\_AuditTableChanges

ON ALL SERVER

FOR CREATE\_TABLE, ALTER\_TABLE, DROP\_TABLE

AS

BEGIN

    DECLARE @EventData XML

    SELECT @EventData = EVENTDATA()

    INSERT INTO SampleDB.dbo.TableChanges

    (DatabaseName, TableName, EventType, LoginName,

     SQLCommand, AuditDateTime)

    VALUES

    (

         @EventData.value('(/EVENT\_INSTANCE/DatabaseName)[1]', 'varchar(250)'),

         @EventData.value('(/EVENT\_INSTANCE/ObjectName)[1]', 'varchar(250)'),

         @EventData.value('(/EVENT\_INSTANCE/EventType)[1]', 'nvarchar(250)'),

         @EventData.value('(/EVENT\_INSTANCE/LoginName)[1]', 'varchar(250)'),

         @EventData.value('(/EVENT\_INSTANCE/TSQLCommand)[1]', 'nvarchar(2500)'),

         GetDate()

    )

END   
  
In the above example we are using **EventData**() function which returns event data in XML format. The following XML is returned by the **EventData**() function when I created a table with name = **MyTable** in **SampleDB**database.

<EVENT\_INSTANCE>

  <EventType>CREATE\_TABLE</EventType>

  <PostTime>2015-09-11T16:12:49.417</PostTime>

  <SPID>58</SPID>

  <ServerName>VENKAT-PC</ServerName>

  <LoginName>VENKAT-PC\Tan</LoginName>

  <UserName>dbo</UserName>

  <DatabaseName>SampleDB</DatabaseName>

  <SchemaName>dbo</SchemaName>

  <ObjectName>MyTable</ObjectName>

  <ObjectType>TABLE</ObjectType>

  <TSQLCommand>

    <SetOptions ANSI\_NULLS="ON" ANSI\_NULL\_DEFAULT="ON"

                ANSI\_PADDING="ON" QUOTED\_IDENTIFIER="ON"

                ENCRYPTED="FALSE" />

    <CommandText>

      Create Table MyTable

      (

         Id int,

         Name nvarchar(50),

         Gender nvarchar(50)

      )

    </CommandText>

  </TSQLCommand>

</EVENT\_INSTANCE>

# 96. Logon triggers in sql server

**Suggested Videos**  
[Part 93 - Server-scoped ddl triggers](http://csharp-video-tutorials.blogspot.com/2015/09/server-scoped-ddl-triggers.html)  
[Part 94 - SQL Server trigger execution order](http://csharp-video-tutorials.blogspot.com/2015/09/sql-server-trigger-execution-order.html)   
[Part 95 - Audit table changes in sql server](http://csharp-video-tutorials.blogspot.com/2015/09/audit-table-changes-in-sql-server.html)   
  
  
  
In this video we will discuss **Logon triggers in SQL Server**.   
  
  
  
As the name implies **Logon triggers fire in response to a LOGON event**. Logon triggers fire after the authentication phase of logging in finishes, but before the user session is actually established.    
  
**Logon triggers can be used for**  
1. Tracking login activity  
2. Restricting logins to SQL Server  
3. Limiting the number of sessions for a specific login   
  
**Logon trigger example :** The following trigger limits the maximum number of open connections for a user to 3.

CREATE TRIGGER tr\_LogonAuditTriggers

ON ALL SERVER

FOR LOGON

AS

BEGIN

    DECLARE @LoginName NVARCHAR(100)

    Set @LoginName = ORIGINAL\_LOGIN()

    IF (SELECT COUNT(\*) FROM sys.dm\_exec\_sessions

         WHERE is\_user\_process = 1

         AND original\_login\_name = @LoginName) > 3

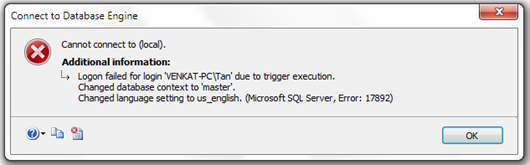
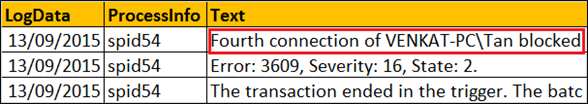
    BEGIN

         Print 'Fourth connection of ' + @LoginName + ' blocked'

         ROLLBACK

    END

END

**An attempt to make a fourth connection, will be blocked.**   
   
  
The trigger error message will be written to the error log. Execute the following command to read the error log.  
Execute sp\_readerrorlog   
  


# 97. Select into in sql server

**Suggested Videos**  
[Part 94 - SQL Server trigger execution order](http://csharp-video-tutorials.blogspot.com/2015/09/sql-server-trigger-execution-order.html)  
[Part 95 - Audit table changes in sql server](http://csharp-video-tutorials.blogspot.com/2015/09/audit-table-changes-in-sql-server.html)   
[Part 96 - Logon triggers in sql server](http://csharp-video-tutorials.blogspot.com/2015/09/logon-triggers-in-sql-server.html)   
  
  
  
In this video we will discuss the power and use of **SELECT INTO statement in SQL Server**.    
  
  
  
We will be using the following 2 tables for the examples.

|  |  |
| --- | --- |
| departments table | employees table |

**SQL Script to create Departments and Employees tables**

Create table Departments

(

    DepartmentId int primary key,

    DepartmentName nvarchar(50)

)

Go

Insert into Departments values (1, 'IT')

Insert into Departments values (2, 'HR')

Insert into Departments values (3, 'Payroll')

Go

Create table Employees

(

    Id int primary key,

    Name nvarchar(100),

    Gender nvarchar(10),

    Salary int,

    DeptId int foreign key references Departments(DepartmentId)

)

Go

Insert into Employees values (1, 'Mark', 'Male', 50000, 1)

Insert into Employees values (2, 'Sara', 'Female', 65000, 2)

Insert into Employees values (3, 'Mike', 'Male', 48000, 3)

Insert into Employees values (4, 'Pam', 'Female', 70000, 1)

Insert into Employees values (5, 'John', 'Male', 55000, 2)

Go

The **SELECT INTO statement in SQL Server**, selects data from one table and inserts it into a new table.   
  
**SELECT INTO statement in SQL Server can do the following**  
1. Copy all rows and columns from an existing table into a new table. This is extremely useful when you want to make a backup copy of the existing table.

SELECT \* INTO EmployeesBackup FROM Employees

2. Copy all rows and columns from an existing table into a new table in an external database.

SELECT \* INTO HRDB.dbo.EmployeesBackup FROM Employees

3. Copy only selected columns into a new table

SELECT Id, Name, Gender INTO EmployeesBackup FROM Employees

4. Copy only selected rows into a new table

SELECT \* INTO EmployeesBackup FROM Employees WHERE DeptId = 1

5. Copy columns from 2 or more table into a new table

SELECT \* INTO EmployeesBackup

FROM Employees

INNER JOIN Departments

ON Employees.DeptId = Departments.DepartmentId

6. Create a new table whose columns and datatypes match with an existing table.

SELECT \* INTO EmployeesBackup FROM Employees WHERE 1 <> 1

7. Copy all rows and columns from an existing table into a new table on a different SQL Server instance. For this, create a linked server and use the 4 part naming convention

SELECT \* INTO TargetTable

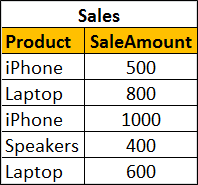
FROM [SourceServer].[SourceDB].[dbo].[SourceTable]

**Please note :** You cannot use SELECT INTO statement to select data into an existing table. For this you will have to use INSERT INTO statement.

INSERT INTO ExistingTable (ColumnList)

SELECT ColumnList FROM SourceTable

# 98. Difference between where and having in sql server

**Suggested Videos**  
[Part 95 - Audit table changes in sql server](http://csharp-video-tutorials.blogspot.com/2015/09/audit-table-changes-in-sql-server.html)  
[Part 96 - Logon triggers in sql server](http://csharp-video-tutorials.blogspot.com/2015/09/logon-triggers-in-sql-server.html)   
[Part 97 - Select into in sql server](http://csharp-video-tutorials.blogspot.com/2015/09/select-into-in-sql-server.html)   
  
  
  
In this video we will discuss the **difference between where and having** clauses in SQL Server.   
  
  
  
Let us understand the difference with an example. For the examples in this video we will use the following Sales table.   
   
  
**SQL Script to create and populate Sales table with test data**

Create table Sales

(

    Product nvarchar(50),

    SaleAmount int

)

Go

Insert into Sales values ('iPhone', 500)

Insert into Sales values ('Laptop', 800)

Insert into Sales values ('iPhone', 1000)

Insert into Sales values ('Speakers', 400)

Insert into Sales values ('Laptop', 600)

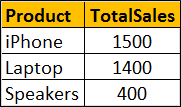
Go

To calculate total sales by product, we would write a GROUP BY query as shown below

SELECT Product, SUM(SaleAmount) AS TotalSales

FROM Sales

GROUP BY Product

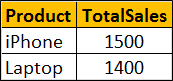
**The above query produces the following result**   
   
  
Now if we want to find only those **products where the total sales amount is greater than $1000**, we will use HAVING clause to filter products

SELECT Product, SUM(SaleAmount) AS TotalSales

FROM Sales

GROUP BY Product

HAVING SUM(SaleAmount) > 1000

**Result :**   
   
  
If we use WHERE clause instead of HAVING clause, we will get a syntax error. This is because the WHERE clause doesn’t work with aggregate functions like sum, min, max, avg, etc.

SELECT Product, SUM(SaleAmount) AS TotalSales

FROM Sales

GROUP BY Product

WHERE SUM(SaleAmount) > 1000

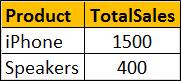
So in short, the difference is **WHERE clause cannot be used with aggregates where as HAVING can.**   
  
However, there are other differences as well that we need to keep in mind when using WHERE and HAVING clauses. WHERE clause filters rows before aggregate calculations are performed where as HAVING clause filters rows after aggregate calculations are performed. Let us understand this with an example.   
  
Total sales of iPhone and Speakers can be calculated by using either WHERE or HAVING clause   
  
**Calculate Total sales of iPhone and Speakers using WHERE clause :** In this example the WHERE clause retrieves only iPhone and Speaker products and then performs the sum.

SELECT Product, SUM(SaleAmount) AS TotalSales

FROM Sales

WHERE Product in ('iPhone', 'Speakers')

GROUP BY Product

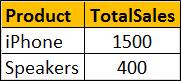
**Result :**   
   
  
**Calculate Total sales of iPhone and Speakers using HAVING clause :** This example retrieves all rows from Sales table, performs the sum and then removes all products except iPhone and Speakers.

SELECT Product, SUM(SaleAmount) AS TotalSales

FROM Sales

GROUP BY Product

HAVING Product in ('iPhone', 'Speakers')

**Result :**   
   
  
So from a performance standpoint, HAVING is slower than WHERE and should be avoided when possible.   
  
Another difference is WHERE comes before GROUP BY and HAVING comes after GROUP BY.   
  
**Difference between WHERE and Having**  
1. WHERE clause cannot be used with aggregates where as HAVING can. This means WHERE clause is used for filtering individual rows where as HAVING clause is used to filter groups.  
  
2. WHERE comes before GROUP BY. This means WHERE clause filters rows before aggregate calculations are performed. HAVING comes after GROUP BY. This means HAVING clause filters rows after aggregate calculations are performed. So from a performance standpoint, HAVING is slower than WHERE and should be avoided when possible.  
  
3. WHERE and HAVING can be used together in a SELECT query. In this case WHERE clause is applied first to filter individual rows. The rows are then grouped and aggregate calculations are performed, and then the HAVING clause filters the groups.

# 99. Table valued parameters in SQL Server

**Suggested Videos**  
[Part 96 - Logon triggers in sql server](http://csharp-video-tutorials.blogspot.com/2015/09/logon-triggers-in-sql-server.html)  
[Part 97 - Select into in sql server](http://csharp-video-tutorials.blogspot.com/2015/09/select-into-in-sql-server.html)   
[Part 98 - Difference between where and having in sql server](http://csharp-video-tutorials.blogspot.com/2015/09/difference-between-where-and-having-in.html)   
  
  
  
In this video we will discuss **table valued parameters in SQL Server**.   
  
  
  
**Table Valued Parameter** is a new feature introduced in SQL SERVER 2008. Table Valued Parameter allows a table (i.e multiple rows of data) to be passed as a parameter to a stored procedure from T-SQL code or from an application. Prior to SQL SERVER 2008, it is not possible to pass a table variable as a parameter to a stored procedure.   
  
Let us understand how to pass multiple rows to a stored procedure using Table Valued Parameter with an example. We want to insert multiple rows into the following Employees table. At the moment this table does not have any rows.  
employees table   
  
**SQL Script to create the Employees table**

Create Table Employees

(

    Id int primary key,

    Name nvarchar(50),

    Gender nvarchar(10)

)

Go

**Step 1 :** Create User-defined Table Type

CREATE TYPE EmpTableType AS TABLE

(

    Id INT PRIMARY KEY,

    Name NVARCHAR(50),

    Gender NVARCHAR(10)

)

Go

**Step 2 :** Use the User-defined Table Type as a parameter in the stored procedure. Table valued parameters must be passed as read-only to stored procedures, functions etc. This means you cannot perform DML operations like INSERT, UPDATE or DELETE on a table-valued parameter in the body of a function, stored procedure etc.

CREATE PROCEDURE spInsertEmployees

@EmpTableType EmpTableType READONLY

AS

BEGIN

    INSERT INTO Employees

    SELECT \* FROM @EmpTableType

END

**Step 3 :** Declare a table variable, insert the data and then pass the table variable as a parameter to the stored procedure.

DECLARE @EmployeeTableType EmpTableType

INSERT INTO @EmployeeTableType VALUES (1, 'Mark', 'Male')

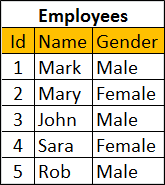
INSERT INTO @EmployeeTableType VALUES (2, 'Mary', 'Female')

INSERT INTO @EmployeeTableType VALUES (3, 'John', 'Male')

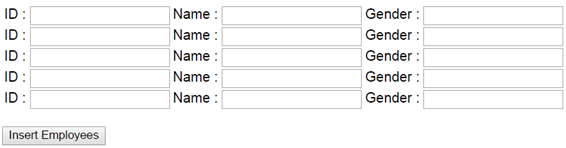
INSERT INTO @EmployeeTableType VALUES (4, 'Sara', 'Female')

INSERT INTO @EmployeeTableType VALUES (5, 'Rob', 'Male')

EXECUTE spInsertEmployees @EmployeeTableType

That's it. Now select the data from Employees table and notice that all the rows of the table variable are inserted into the Employees table.   
   
  
In our next video, we will discuss **how to pass table as a parameter to the stored procedure from an ADO.NET application**

# 100. Send datatable as parameter to stored procedure

**Suggested Videos**  
[Part 97 - Select into in sql server](http://csharp-video-tutorials.blogspot.com/2015/09/select-into-in-sql-server.html)  
[Part 98 - Difference between where and having in sql server](http://csharp-video-tutorials.blogspot.com/2015/09/difference-between-where-and-having-in.html)   
[Part 99 - Table valued parameters in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/09/table-valued-parameters-in-sql-server.html)   
  
  
  
In this video we will discuss **how to send datatable as parameter to stored procedure**. This is continuation to [Part 99](http://csharp-video-tutorials.blogspot.com/2015/09/table-valued-parameters-in-sql-server.html). Please watch [Part 99](http://csharp-video-tutorials.blogspot.com/2015/09/table-valued-parameters-in-sql-server.html) from [SQL Server tutorial](https://www.youtube.com/playlist?list=PL08903FB7ACA1C2FB) before proceeding.   
  
  
  
In [Part 99](http://csharp-video-tutorials.blogspot.com/2015/09/table-valued-parameters-in-sql-server.html), we discussed creating a stored procedure that accepts a table as a parameter. In this video we will discuss **how to pass a datatable from a web application to the SQL Server stored procedure**.  
  
**Here is what we want to do.**  
**1.** Design a webform that looks as shown below. This form allows us to insert 5 employees at a time into the database table.   
   
  
**2. When "Insert Employees"** button is clicked, retrieve the from data into a datatabe and then pass the datatable as a parameter to the stored procedure.   
  
**3.** The stored procedure will then insert all the rows into the Employees table in the database.   
  
**Here are the steps to achieve this.**  
**Step 1 :**Create new asp.net web application project. Name it Demo.   
  
**Step 2 :**Include a connection string in the web.config file to your database.

<add name="DBCS"

      connectionString="server=.;database=SampleDB;integrated security=SSPI"/>

**Step 3 :**Copy and paste the following HTML in WebForm1.aspx

<asp:Button ID="btnFillDummyData" runat="server" Text="Fill Dummy Data"

    OnClick="btnFillDummyData\_Click" />

<br /><br />

<table>

    <tr>

        <td>

            ID : <asp:TextBox ID="txtId1" runat="server"></asp:TextBox>

        </td>

        <td>

            Name : <asp:TextBox ID="txtName1" runat="server"></asp:TextBox>

        </td>

        <td>

            Gender : <asp:TextBox ID="txtGender1" runat="server"></asp:TextBox>

        </td>

    </tr>

    <tr>

        <td>

            ID : <asp:TextBox ID="txtId2" runat="server"></asp:TextBox>

        </td>

        <td>

            Name : <asp:TextBox ID="txtName2" runat="server"></asp:TextBox>

        </td>

        <td>

            Gender : <asp:TextBox ID="txtGender2" runat="server"></asp:TextBox>

        </td>

    </tr>

    <tr>

        <td>

            ID : <asp:TextBox ID="txtId3" runat="server"></asp:TextBox>

        </td>

        <td>

            Name : <asp:TextBox ID="txtName3" runat="server"></asp:TextBox>

        </td>

        <td>

            Gender : <asp:TextBox ID="txtGender3" runat="server"></asp:TextBox>

        </td>

    </tr>

    <tr>

        <td>

            ID : <asp:TextBox ID="txtId4" runat="server"></asp:TextBox>

        </td>

        <td>

            Name : <asp:TextBox ID="txtName4" runat="server"></asp:TextBox>

        </td>

        <td>

            Gender : <asp:TextBox ID="txtGender4" runat="server"></asp:TextBox>

        </td>

    </tr>

    <tr>

        <td>

            ID : <asp:TextBox ID="txtId5" runat="server"></asp:TextBox>

        </td>

        <td>

            Name : <asp:TextBox ID="txtName5" runat="server"></asp:TextBox>

        </td>

        <td>

            Gender : <asp:TextBox ID="txtGender5" runat="server"></asp:TextBox>

        </td>

    </tr>

</table>

<br />

<asp:Button ID="btnInsert" runat="server" Text="Insert Employees"

    OnClick="btnInsert\_Click" />

**Step 4 :** Copy and paste the following code in the code-behind file

using System;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

namespace Demo

{

    public partial class WebForm1 : System.Web.UI.Page

    {

        protected void Page\_Load(object sender, EventArgs e)

        { }

        private DataTable GetEmployeeData()

        {

            DataTable dt = new DataTable();

            dt.Columns.Add("Id");

            dt.Columns.Add("Name");

            dt.Columns.Add("Gender");

            dt.Rows.Add(txtId1.Text, txtName1.Text, txtGender1.Text);

            dt.Rows.Add(txtId2.Text, txtName2.Text, txtGender2.Text);

            dt.Rows.Add(txtId3.Text, txtName3.Text, txtGender3.Text);

            dt.Rows.Add(txtId4.Text, txtName4.Text, txtGender4.Text);

            dt.Rows.Add(txtId5.Text, txtName5.Text, txtGender5.Text);

            return dt;

        }

        protected void btnInsert\_Click(object sender, EventArgs e)

        {

            string cs = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;

            using (SqlConnection con = new SqlConnection(cs))

            {

                SqlCommand cmd = new SqlCommand("spInsertEmployees", con);

                cmd.CommandType = CommandType.StoredProcedure;

                SqlParameter paramTVP = new SqlParameter()

                {

                    ParameterName = "@EmpTableType",

                    Value = GetEmployeeData()

                };

                cmd.Parameters.Add(paramTVP);

                con.Open();

                cmd.ExecuteNonQuery();

                con.Close();

            }

        }

        protected void btnFillDummyData\_Click(object sender, EventArgs e)

        {

            txtId1.Text = "1";

            txtId2.Text = "2";

            txtId3.Text = "3";

            txtId4.Text = "4";

            txtId5.Text = "5";

            txtName1.Text = "John";

            txtName2.Text = "Mike";

            txtName3.Text = "Sara";

            txtName4.Text = "Pam";

            txtName5.Text = "Todd";

            txtGender1.Text = "Male";

            txtGender2.Text = "Male";

            txtGender3.Text = "Female";

            txtGender4.Text = "Female";

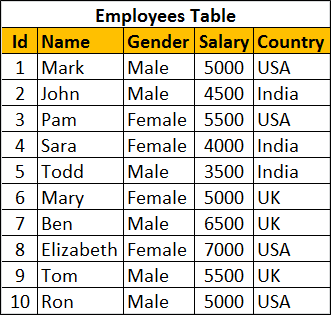
            txtGender5.Text = "Male";

        }

    }

}

# 101. Grouping Sets in SQL Server

**Suggested Videos**  
[Part 98 - Difference between where and having in sql server](http://csharp-video-tutorials.blogspot.com/2015/09/difference-between-where-and-having-in.html)  
[Part 99 - Table valued parameters in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/09/table-valued-parameters-in-sql-server.html)   
[Part 100 - Send datatable as parameter to stored procedure](http://csharp-video-tutorials.blogspot.com/2015/09/send-datatable-as-parameter-to-stored.html)   
  
  
  
**Grouping sets is a new feature introduced in SQL Server 2008**. Let us understand Grouping sets with an example.    
  
  
  
We will be using the following **Employees table**for the examples in this video.   
   
  
**SQL Script to create and populate Employees table**

Create Table Employees

(

    Id int primary key,

    Name nvarchar(50),

    Gender nvarchar(10),

    Salary int,

    Country nvarchar(10)

)

Go

Insert Into Employees Values (1, 'Mark', 'Male', 5000, 'USA')

Insert Into Employees Values (2, 'John', 'Male', 4500, 'India')

Insert Into Employees Values (3, 'Pam', 'Female', 5500, 'USA')

Insert Into Employees Values (4, 'Sara', 'Female', 4000, 'India')

Insert Into Employees Values (5, 'Todd', 'Male', 3500, 'India')

Insert Into Employees Values (6, 'Mary', 'Female', 5000, 'UK')

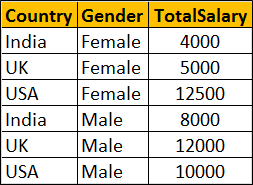
Insert Into Employees Values (7, 'Ben', 'Male', 6500, 'UK')

Insert Into Employees Values (8, 'Elizabeth', 'Female', 7000, 'USA')

Insert Into Employees Values (9, 'Tom', 'Male', 5500, 'UK')

Insert Into Employees Values (10, 'Ron', 'Male', 5000, 'USA')

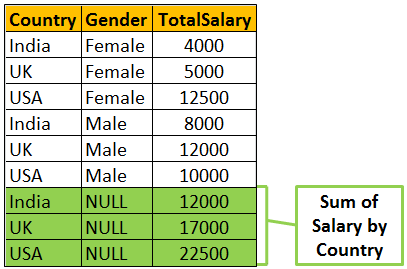
Go

We want to calculate **Sum of Salary by Country and Gender**. The result should be as shown below.   
   
  
We can very easily achieve this using a Group By query as shown below

Select Country, Gender, Sum(Salary) as TotalSalary

From Employees

Group By Country, Gender

Within the same result set we also want Sum of Salary just by Country. The Result should be as shown below. Notice that Gender column within the resultset is NULL as we are grouping only by Country column   
   
  
To achieve the above result we could combine 2 Group By queries using UNION ALL as shown below.

Select Country, Gender, Sum(Salary) as TotalSalary

From Employees

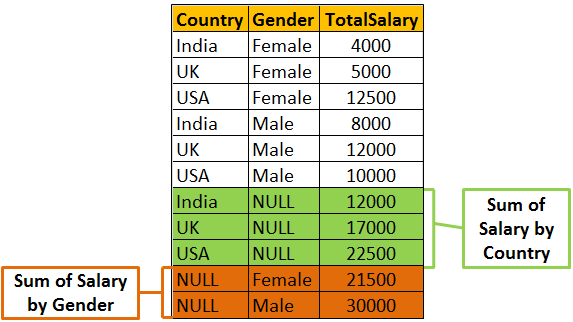
Group By Country, Gender

UNION ALL

Select Country, NULL, Sum(Salary) as TotalSalary

From Employees

Group By Country

Within the same result set we also want Sum of Salary just by Gender. The Result should be as shown below. Notice that the Country column within the resultset is NULL as we are grouping only by Gender column.    
   
  
We can achieve this by combining 3 Group By queries using UNION ALL as shown below

Select Country, Gender, Sum(Salary) as TotalSalary

From Employees

Group By Country, Gender

UNION ALL

Select Country, NULL, Sum(Salary) as TotalSalary

From Employees

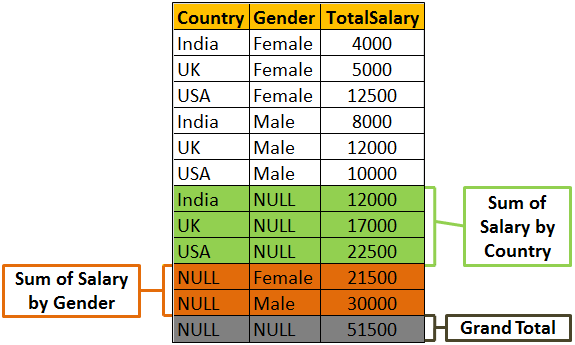
Group By Country

UNION ALL

Select NULL, Gender, Sum(Salary) as TotalSalary

From Employees

Group By Gender

Finally we also want the grand total of Salary. In this case we are not grouping on any particular column. So both Country and Gender columns will be NULL in the resultset.   
  
  
To achieve this we will have to combine the fourth query using UNION ALL as shown below.

Select Country, Gender, Sum(Salary) as TotalSalary

From Employees

Group By Country, Gender

UNION ALL

Select Country, NULL, Sum(Salary) as TotalSalary

From Employees

Group By Country

UNION ALL

Select NULL, Gender, Sum(Salary) as TotalSalary

From Employees

Group By Gender

UNION ALL

Select NULL, NULL, Sum(Salary) as TotalSalary

From Employees    
  
**There are 2 problems with the above approach.**  
1. The query is huge as we have combined different Group By queries using UNION ALL operator. This can grow even more if we start to add more groups  
2. The Employees table has to be accessed 4 times, once for every query.  
  
If we use **Grouping Sets** feature introduced in SQL Server 2008, the amount of T-SQL code that you have to write will be greatly reduced. The following Grouping Sets query produce the same result as the above UNION ALL query.

Select Country, Gender, Sum(Salary) TotalSalary

From Employees

Group BY

      GROUPING SETS

      (

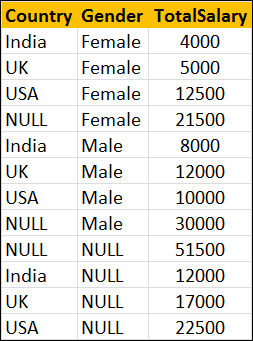
            (Country, Gender), -- Sum of Salary by Country and Gender

            (Country),               -- Sum of Salary by Country

            (Gender) ,               -- Sum of Salary by Gender

            ()                             -- Grand Total

      )

Output of the above query   
   
  
The order of the rows in the result set is not the same as in the case of UNION ALL query. To control the order use order by as shown below.

Select Country, Gender, Sum(Salary) TotalSalary

From Employees

Group BY

      GROUPING SETS

      (

            (Country, Gender), -- Sum of Salary by Country and Gender

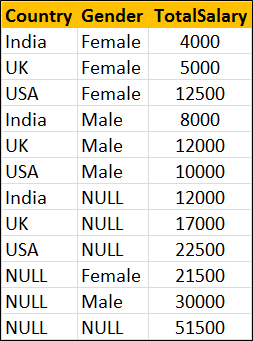
            (Country),               -- Sum of Salary by Country

            (Gender) ,               -- Sum of Salary by Gender

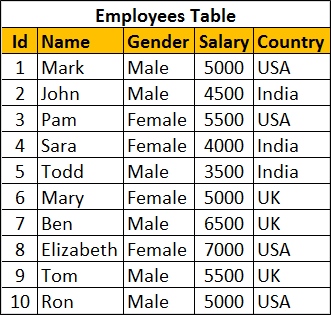
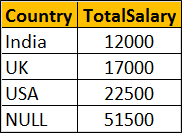
            ()                             -- Grand Total

      )

Order By Grouping(Country), Grouping(Gender), Gender

Output of the above query   


# 102. Rollup in SQL Server

**Suggested Videos**  
[Part 99 - Table valued parameters in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/09/table-valued-parameters-in-sql-server.html)  
[Part 100 - Send datatable as parameter to stored procedure](http://csharp-video-tutorials.blogspot.com/2015/09/send-datatable-as-parameter-to-stored.html)   
[Part 101 - Grouping sets in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/09/grouping-sets-in-sql-server.html)   
  
  
  
**ROLLUP in SQL Server** is used to do aggregate operation on multiple levels in hierarchy.    
  
  
  
Let us understand Rollup in SQL Server with examples. We will use the following**Employees table** for the examples in this video.   
   
  
Retrieve Salary by country along with grand total   
   
  
There are several ways to achieve this. The easiest way is by using Rollup with Group By.

SELECT Country, SUM(Salary) AS TotalSalary

FROM Employees

GROUP BY ROLLUP(Country)

The above query can also be rewritten as shown below

SELECT Country, SUM(Salary) AS TotalSalary

FROM Employees

GROUP BY Country WITH ROLLUP

We can also use UNION ALL operator along with GROUP BY

SELECT Country, SUM(Salary) AS TotalSalary

FROM Employees

GROUP BY Country

UNION ALL

SELECT NULL, SUM(Salary) AS TotalSalary

FROM Employees

We can also use Grouping Sets to achieve the same result

SELECT Country, SUM(Salary) AS TotalSalary

FROM Employees

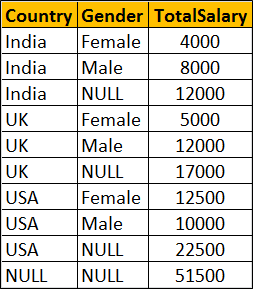
GROUP BY GROUPING SETS

(

    (Country),

    ()

)

Let's look at another example.    
  
Group Salary by Country and Gender. Also compute the Subtotal for Country level and Grand Total as shown below.   
   
  
**Using ROLLUP with GROUP BY**

SELECT Country, Gender, SUM(Salary) AS TotalSalary

FROM Employees

GROUP BY ROLLUP(Country, Gender)

--OR

SELECT Country, Gender, SUM(Salary) AS TotalSalary

FROM Employees

GROUP BY Country, Gender WITH ROLLUP

**Using UNION ALL with GROUP BY**

SELECT Country, Gender, SUM(Salary) AS TotalSalary

FROM Employees

GROUP BY Country, Gender

UNION ALL

SELECT Country, NULL, SUM(Salary) AS TotalSalary

FROM Employees

GROUP BY Country

UNION ALL

SELECT NULL, NULL, SUM(Salary) AS TotalSalary

FROM Employees

**Using GROUPING SETS**

SELECT Country, Gender, SUM(Salary) AS TotalSalary

FROM Employees

GROUP BY GROUPING SETS

(

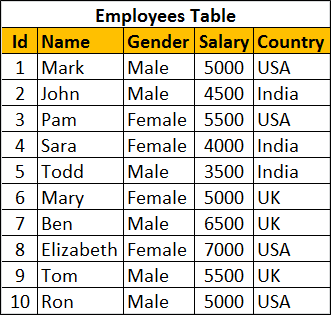
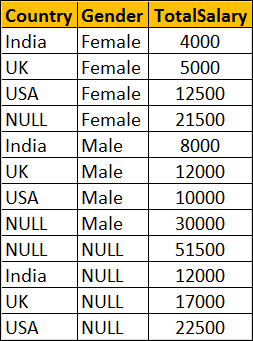
    (Country, Gender),

    (Country),

    ()

)

# 103. Cube in SQL Server

**Suggested Videos**  
[Part 100 - Send datatable as parameter to stored procedure](http://csharp-video-tutorials.blogspot.com/2015/09/send-datatable-as-parameter-to-stored.html)  
[Part 101 - Grouping sets in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/09/grouping-sets-in-sql-server.html)   
[Part 102 - Rollup in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/09/rollup-in-sql-server.html)   
  
  
  
Cube() in SQL Server produces the result set by generating all combinations of columns specified in GROUP BY CUBE().    
  
  
  
Let us understand Cube() in SQL Server with examples. We will use the following**Employees table** for the examples in this video.   
   
  
Write a query to retrieve Sum of Salary grouped by all combinations of the following 2 columns as well as Grand Total.  
Country,  
Gender    
  
**The output of the query should be as shown below**   
   
  
**Using Cube with Group By**

SELECT Country, Gender, SUM(Salary) AS TotalSalary

FROM Employees

GROUP BY Cube(Country, Gender)

--OR

SELECT Country, Gender, SUM(Salary) AS TotalSalary

FROM Employees

GROUP BY Country, Gender with Cube

**The above query is equivalent to the following Grouping Sets query**

SELECT Country, Gender, SUM(Salary) AS TotalSalary

FROM Employees

GROUP BY

    GROUPING SETS

    (

         (Country, Gender),

         (Country),

         (Gender),

         ()

    )

**The above query is equivalent to the following UNION ALL query.** While the data in the result set is the same, the ordering is not. Use ORDER BY to control the ordering of rows in the result set.

SELECT Country, Gender, SUM(Salary) AS TotalSalary

FROM Employees

GROUP BY Country, Gender

UNION ALL

SELECT Country, NULL, SUM(Salary) AS TotalSalary

FROM Employees

GROUP BY Country

UNION ALL

SELECT NULL, Gender, SUM(Salary) AS TotalSalary

FROM Employees

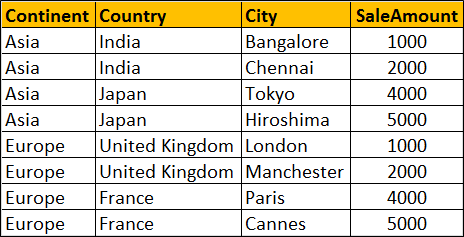
GROUP BY Gender

UNION ALL

SELECT NULL, NULL, SUM(Salary) AS TotalSalary

FROM Employees

# 104. Difference between cube and rollup in SQL Server

**Suggested Videos**  
[Part 101 - Grouping sets in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/09/grouping-sets-in-sql-server.html)  
[Part 102 - Rollup in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/09/rollup-in-sql-server.html)   
[Part 103 - Cube in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/09/cube-in-sql-server.html)   
  
  
  
In this video we will discuss the **difference between cube and rollup in SQL Server**.   
  
  
  
**CUBE generates a result set** that shows aggregates for all combinations of values in the selected columns, where as ROLLUP generates a result set that shows aggregates for a hierarchy of values in the selected columns.  
  
Let us understand this difference with an example. Consider the following **Sales**table.   
   
  
**SQL Script to create and populate Sales table**

Create table Sales

(

    Id int primary key identity,

    Continent nvarchar(50),

    Country nvarchar(50),

    City nvarchar(50),

    SaleAmount int

)

Go

Insert into Sales values('Asia','India','Bangalore',1000)

Insert into Sales values('Asia','India','Chennai',2000)

Insert into Sales values('Asia','Japan','Tokyo',4000)

Insert into Sales values('Asia','Japan','Hiroshima',5000)

Insert into Sales values('Europe','United Kingdom','London',1000)

Insert into Sales values('Europe','United Kingdom','Manchester',2000)

Insert into Sales values('Europe','France','Paris',4000)

Insert into Sales values('Europe','France','Cannes',5000)

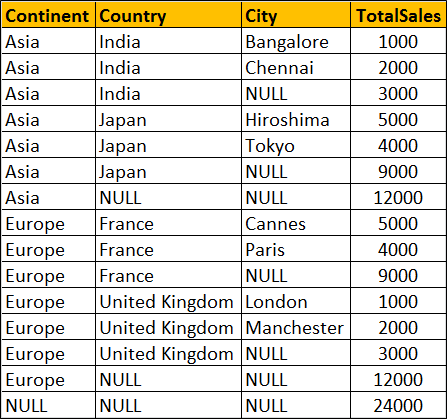
Go

**ROLLUP(Continent, Country, City)** produces Sum of Salary for the following hierarchy  
Continent, Country, City  
Continent, Country,   
Continent  
()   
  
**CUBE(Continent, Country, City)** produces Sum of Salary for all the following column combinations  
Continent, Country, City  
Continent, Country,   
Continent, City  
Continent  
Country, City  
Country,  
City  
()

SELECT Continent, Country, City, SUM(SaleAmount) AS TotalSales

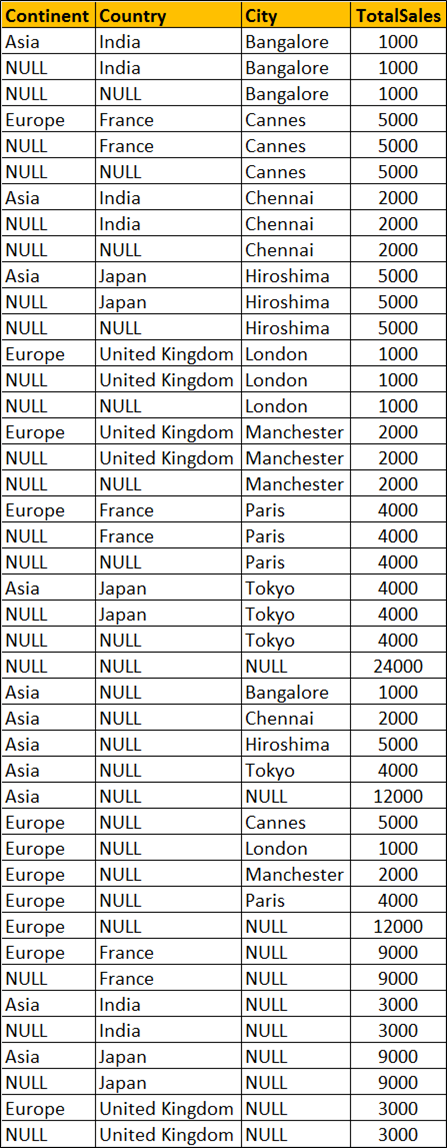
FROM Sales

GROUP BY ROLLUP(Continent, Country, City)



SELECT Continent, Country, City, SUM(SaleAmount) AS TotalSales

FROM Sales

GROUP BY CUBE(Continent, Country, City)  
   
  
You won't see any difference when you use ROLLUP and CUBE on a single column. Both the following queries produces the same output.

SELECT Continent, Sum(SaleAmount) AS TotalSales

FROM Sales

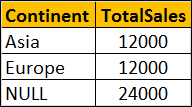
GROUP BY ROLLUP(Continent)

-- OR

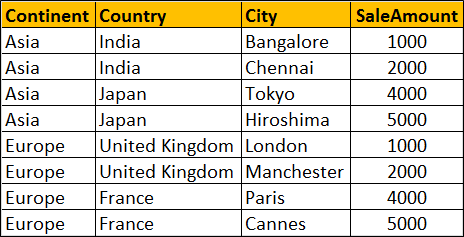
SELECT Continent, SUM(SaleAmount) AS TotalSales

FROM Sales

GROUP BY CUBE(Continent)



# 105. Grouping function in SQL Server

**Suggested Videos**  
[Part 102 - Rollup in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/09/rollup-in-sql-server.html)  
[Part 103 - Cube in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/09/cube-in-sql-server.html)   
[Part 104 - Difference between cube and rollup in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/09/difference-between-cube-and-rollup-in.html)  
  
  
  
In this video we will discuss the use of **Grouping function in SQL Server**.    
  
  
  
This is continuation to [Part 104](http://csharp-video-tutorials.blogspot.com/2015/09/difference-between-cube-and-rollup-in.html). Please watch [Part 104](http://csharp-video-tutorials.blogspot.com/2015/09/difference-between-cube-and-rollup-in.html) from [SQL Server tutorial](https://www.youtube.com/playlist?list=PL08903FB7ACA1C2FB) before proceeding. We will use the following Sales table for this example.  
   
  
**What is Grouping function**  
Grouping(Column) indicates whether the column in a GROUP BY list is aggregated or not. Grouping returns 1 for aggregated or 0 for not aggregated in the result set.   
  
The following query returns 1 for aggregated or 0 for not aggregated in the result set

SELECT   Continent, Country, City, SUM(SaleAmount) AS TotalSales,

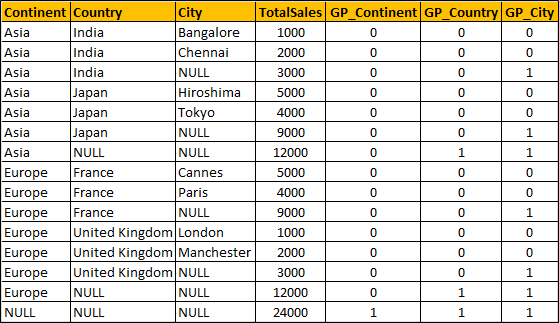
         GROUPING(Continent) AS GP\_Continent,

         GROUPING(Country) AS GP\_Country,

         GROUPING(City) AS GP\_City

FROM Sales

GROUP BY ROLLUP(Continent, Country, City)

**Result :**  
   
  
**What is the use of Grouping function in real world**  
When a column is aggregated in the result set, the column will have a NULL value. If you want to replace NULL with All then this GROUPING function is very handy.

SELECT

    CASE WHEN

         GROUPING(Continent) = 1 THEN 'All' ELSE ISNULL(Continent, 'Unknown')

    END AS Continent,

    CASE WHEN

         GROUPING(Country) = 1 THEN 'All' ELSE ISNULL(Country, 'Unknown')

    END AS Country,

    CASE

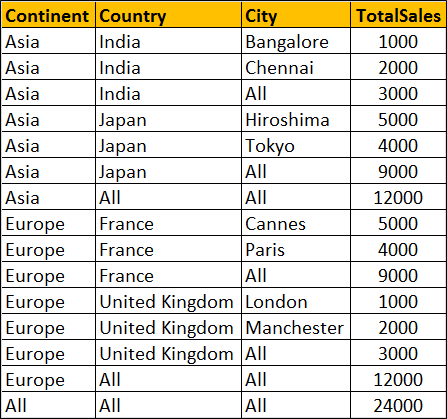
         WHEN GROUPING(City) = 1 THEN 'All' ELSE ISNULL(City, 'Unknown')

    END AS City,

    SUM(SaleAmount) AS TotalSales

FROM Sales

GROUP BY ROLLUP(Continent, Country, City)

Result :   
   
  
**Can't I use ISNULL function instead as shown below**

SELECT   ISNULL(Continent, 'All') AS Continent,

         ISNULL(Country, 'All') AS Country,

         ISNULL(City, 'All') AS City,

         SUM(SaleAmount) AS TotalSales

FROM Sales

GROUP BY ROLLUP(Continent, Country, City)

Well, you can, but only if your data does not contain NULL values. Let me explain what I mean.  
  
At the moment the raw data in our Sales has no NULL values. Let's introduce a NULL value in the City column of the row where Id = 1

Update Sales Set City = NULL where Id = 1

Now execute the following query with ISNULL function

SELECT   ISNULL(Continent, 'All') AS Continent,

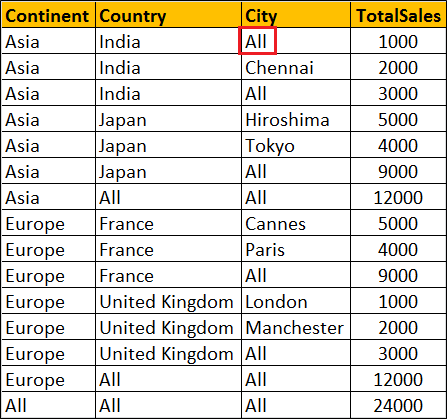
         ISNULL(Country, 'All') AS Country,

         ISNULL(City, 'All') AS City,

         SUM(SaleAmount) AS TotalSales

FROM Sales

GROUP BY ROLLUP(Continent, Country, City)

**Result :** Notice that the actuall NULL value in the raw data is also replaced with the word 'All', which is incorrect. Hence the need for Grouping function.  
   
  
**Please note :** Grouping function can be used with Rollup, Cube and Grouping Sets

# 106. GROUPING\_ID function in SQL Server

**Suggested Videos**  
[Part 103 - Cube in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/09/cube-in-sql-server.html)  
[Part 104 - Difference between cube and rollup in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/09/difference-between-cube-and-rollup-in.html)  
[Part 105 - Grouping function in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/09/grouping-function-in-sql-server.html)  
  
  
  
**In this video we will discuss**  
1. GROUPING\_ID function in SQL Server  
2. Difference between GROUPING and GROUPING\_ID functions  
3. Use of GROUPING\_ID function  
  
  
  
GROUPING\_ID function computes the level of grouping.  
  
**Difference between GROUPING and GROUPING\_ID**  
  
**Syntax :** GROUPING function is used on single column, where as the column list for GROUPING\_ID function must match with GROUP BY column list.

GROUPING(Col1)

GROUPING\_ID(Col1, Col2, Col3,...)

GROUPING indicates whether the column in a GROUP BY list is aggregated or not. Grouping returns 1 for aggregated or 0 for not aggregated in the result set.   
  
GROUPING\_ID() function concatenates all the GOUPING() functions, perform the binary to decimal conversion, and returns the equivalent integer. In short  
GROUPING\_ID(A, B, C) =  GROUPING(A) + GROUPING(B) + GROUPING(C)  
  
**Let us understand this with an example.**

SELECT   Continent, Country, City, SUM(SaleAmount) AS TotalSales,

         CAST(GROUPING(Continent) AS NVARCHAR(1)) +

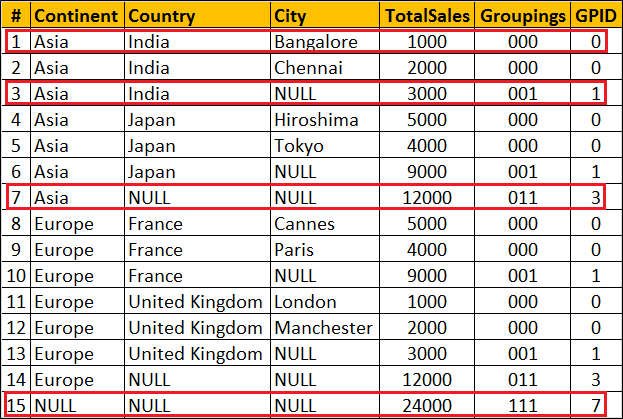
         CAST(GROUPING(Country) AS NVARCHAR(1)) +

         CAST(GROUPING(City) AS NVARCHAR(1)) AS Groupings,

         GROUPING\_ID(Continent, Country, City) AS GPID

FROM Sales

GROUP BY ROLLUP(Continent, Country, City)

**Query result :**  
  
  
**Row Number 1 :** Since the data is not aggregated by any column GROUPING(Continent), GROUPING(Country) and GROUPING(City) return 0 and as result we get a binar string with all ZEROS (000). When this converted to decimal we get 0 which is displayed in GPID column.  
  
**Row Number 7 :** The data is aggregated for Country and City columns, so GROUPING(Country) and GROUPING(City) return 1 where as  GROUPING(Continent) return 0. As result we get a binar string (011). When this converted to decimal we get 10 which is displayed in GPID column.  
  
**Row Number 15 :** This is the Grand total row. Notice in this row the data is aggregated by all the 3 columns. Hence all the 3 GROUPING functions return 1. So we get a binary string with all ONES (111). When this converted to decimal we get 7 which is displayed in GPID column.  
  
**Use of GROUPING\_ID function :** GROUPING\_ID function is very handy if you want to sort and filter by level of grouping.  
  
**Sorting by level of grouping :**

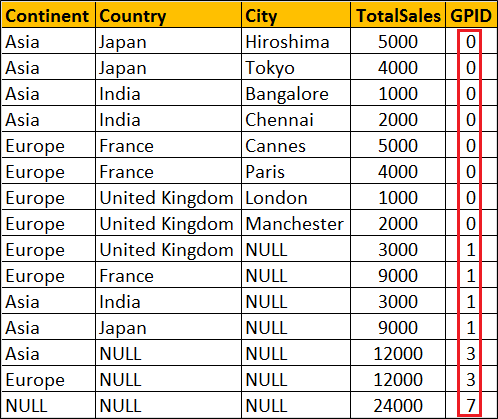
SELECT   Continent, Country, City, SUM(SaleAmount) AS TotalSales,

         GROUPING\_ID(Continent, Country, City) AS GPID

FROM Sales

GROUP BY ROLLUP(Continent, Country, City)

ORDER BY GPID

**Result :**   
   
  
**Filter by level of grouping :** The following query retrieves only continent level aggregated data

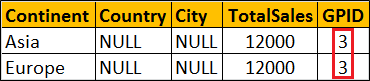
SELECT   Continent, Country, City, SUM(SaleAmount) AS TotalSales,

         GROUPING\_ID(Continent, Country, City) AS GPID

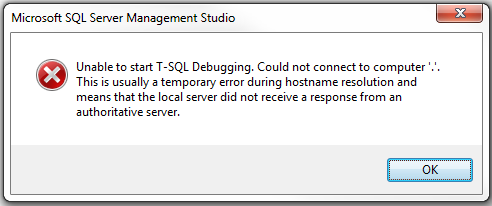
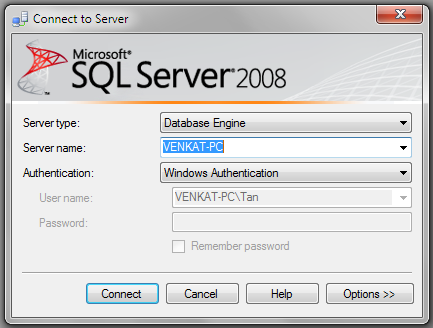
FROM Sales

GROUP BY ROLLUP(Continent, Country, City)

HAVING GROUPING\_ID(Continent, Country, City) = 3

**Result :**   


# 107. Debugging sql server stored procedures

**Suggested Videos**  
[Part 104 - Difference between cube and rollup in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/09/difference-between-cube-and-rollup-in.html)  
[Part 105 - Grouping function in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/09/grouping-function-in-sql-server.html)   
[Part 106 - Grouping\_Id function in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/09/groupingid-function-in-sql-server.html)   
  
  
  
In this video we will discuss **how to debug stored procedures in SQL Server**.   
  
  
  
**Setting up the Debugger in SSMS :** If you have connected to SQL Server using (local) or . (period), and when you start the debugger you will get the following error  
Unable to start T-SQL Debugging. Could not connect to computer.   
   
  
To fix this error, use the computer name to connect to the SQL Server instead of using (local) or .   
   
  
For the examples in this video we will be using the following stored procedure.

Create procedure spPrintEvenNumbers

@Target int

as

Begin

     Declare @StartNumber int

     Set @StartNumber = 1

     while(@StartNumber < @Target)

     Begin

          If(@StartNumber%2 = 0)

          Begin

              Print @StartNumber

          End

          Set @StartNumber = @StartNumber + 1

     End

     Print 'Finished printing even numbers till ' + RTRIM(@Target)

End

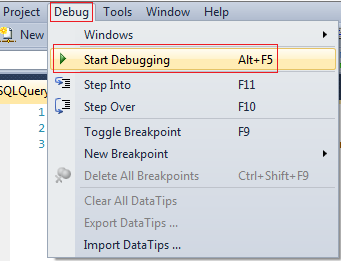
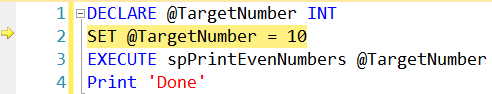
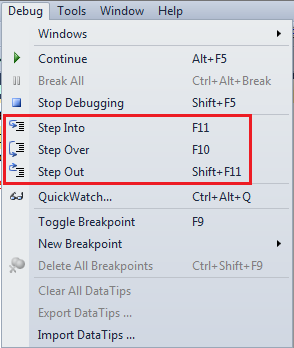
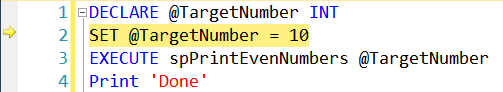
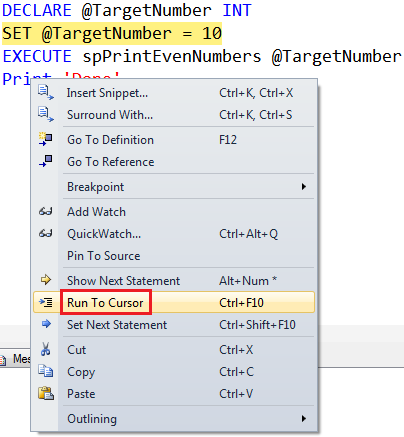
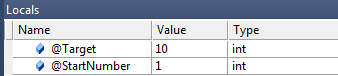
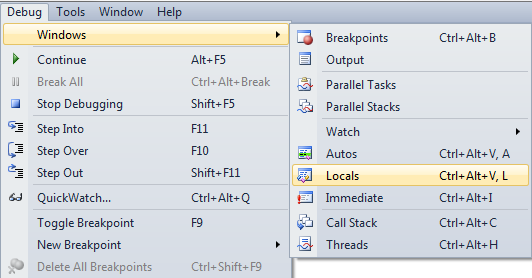
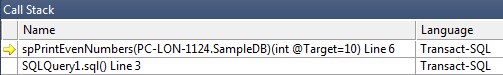
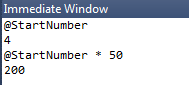
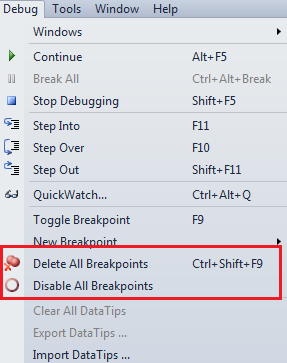
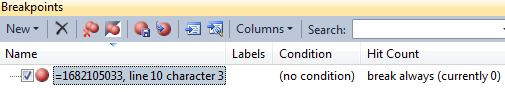
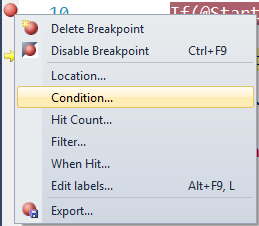
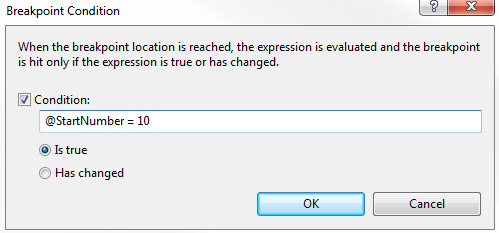
Connect to SQL Server using your computer name, and then execute the above code to create the stored procedure. At this point, open a New Query window. Copy and paste the following T-SQL code to execute the stored procedure.

DECLARE @TargetNumber INT

SET @TargetNumber = 10

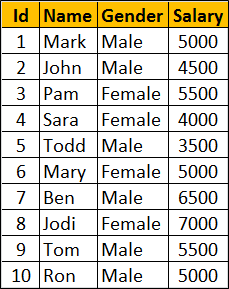
EXECUTE spPrintEvenNumbers @TargetNumber

Print 'Done'

**Starting the Debugger in SSMS :** There are 2 ways to start the debugger  
1. In SSMS, click on the **Debug**Menu and select **Start Debugging**   
   
  
2. Use the keyboard shortcut **ALT + F5**  
  
At this point you should have the debugger running. The line that is about to be executed is marked with an yellow arrow   
   
  
Step Over, Step into and Step Out in SSMS : You can find the keyboard shortcuts in the Debug menu in SSMS.   
   
  
Let us understand what Step Over, Step into and Step Out does when debugging the following piece of code   
   
  
1. There is no difference when you STEP INTO (F11) or STEP OVER (F10) the code on LINE 2  
  
2. On LINE 3, we are calling a Stored Procedure. On this statement if we press F10 (STEP OVER), it won't give us the opportunity to debug the stored procedure code. To be able to debug the stored procedure code you will have to STEP INTO it by pressing F11.  
  
3. If the debugger is in the stored procedure, and you don't want to debug line by line with in that stored procedure, you can STEP OUT of it by pressing SHIFT + F11. When you do this, the debugger completes the execution of the stored procedure and waits on the next line in the main query, i.e on LINE 4 in this example.  
  
**To stop debugging :** There are 2 ways to stop debugging  
1. In SSMS, click on the Debug Menu and select Stop Debugging  
2. Use the keyboard shortcut SHIFT + F5  
  
**Show Next Statement**shows the next statement that the debugger is about to execute.  
Run to Cursor command executes all the statements in a batch up to the current cursor position   
   
  
**Locals Window in SSMS :** Displays the current values of variables and parameters    
   
  
If you cannot see the locals window or if you have closed it and if you want to open it, you can do so using the following menu option. Locals window is only available if you are in DEBUG mode.   
   
  
**Watch Window in SSMS :** Just like Locals window, Watch window is used to watch the values of variables. You can add and remove variables from the watch window. To add a variable to the Watch Window, right click on the variable and select "Add Watch" option from the context menu.   
Watch Window in SSMS   
  
**Call Stack Window in SSMS :** Allows you to navigate up and down the call stack to see what values your application is storing at different levels. It's an invaluable tool for determining why your code is doing what it's doing.   
   
  
**Immediate Window in SSMS :** Very helpful during debugging to evaluate expressions, and print variable values. To clear immediate window type **>cls** and press enter.   
   
  
**Breakpoints in SSMS :** There are 2 ways to set a breakpoint in SSMS.  
1. By clicking on the grey margin on the left hand side in SSMS (to remove click again)  
2. By pressing F9 (to remove press F9 again)  
  
**Enable, Disable or Delete all breakpoints :** There are 2 ways to Enable, Disable or Delete all breakpoints  
  
1. From the Debug menu   
   
  
2. From the Breakpoints window. To view Breakpoints window select Debug => Windows => Breakpoints or use the keyboard shortcut ALT + CTRL + B   
   
  
**Conditional Breakpoint :** Conditional Breakpoints are hit only when the specified condition is met. These are extremely useful when you have some kind of a loop and you want to break, only when the loop variable has a specific value (For example loop varible = 100).  
  
**How to set a conditional break point in SSMS :**  
1. Right click on the Breakpoint and select **Condition** from the context menu   
   
  
2. In the Breakpoint window **specify the condition**   


# 108. Over clause in SQL Server

**Suggested Videos**  
[Part 105 - Grouping function in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/09/grouping-function-in-sql-server.html)  
[Part 106 - Grouping\_Id function in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/09/groupingid-function-in-sql-server.html)   
[Part 107 - Debugging sql server stored procedures](http://csharp-video-tutorials.blogspot.com/2015/09/debugging-sql-server-stored-procedures.html)   
  
  
  
In this video we will discuss the power and use of Over clause in SQL Server.   
  
  
  
The **OVER** clause combined with **PARTITION BY** is used to break up data into partitions.   
**Syntax :** function (...) OVER (PARTITION BY col1, Col2, ...)

The specified function operates for each partition.   
  
**For example :**  
COUNT(Gender) OVER (PARTITION BY Gender) will partition the data by **GENDER**i.e there will 2 partitions (Male and Female) and then the COUNT() function is applied over each partition.   
  
Any of the following functions can be used. Please note this is not the complete list.  
COUNT(), AVG(), SUM(), MIN(), MAX(), ROW\_NUMBER(), RANK(), DENSE\_RANK() etc.   
  
**Example :**We will use the following **Employees table**for the examples in this video.   
   
  
**SQl Script to create Employees table**

Create Table Employees

(

     Id int primary key,

     Name nvarchar(50),

     Gender nvarchar(10),

     Salary int

)

Go

Insert Into Employees Values (1, 'Mark', 'Male', 5000)

Insert Into Employees Values (2, 'John', 'Male', 4500)

Insert Into Employees Values (3, 'Pam', 'Female', 5500)

Insert Into Employees Values (4, 'Sara', 'Female', 4000)

Insert Into Employees Values (5, 'Todd', 'Male', 3500)

Insert Into Employees Values (6, 'Mary', 'Female', 5000)

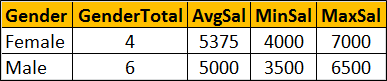
Insert Into Employees Values (7, 'Ben', 'Male', 6500)

Insert Into Employees Values (8, 'Jodi', 'Female', 7000)

Insert Into Employees Values (9, 'Tom', 'Male', 5500)

Insert Into Employees Values (10, 'Ron', 'Male', 5000)

Go

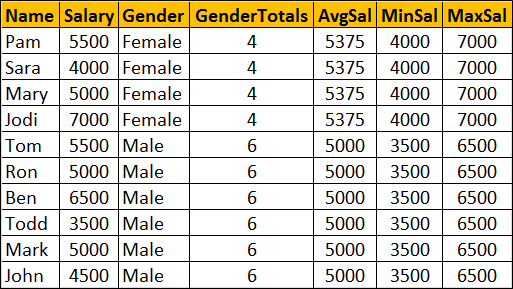
Write a query to retrieve total count of employees by Gender. Also in the result we want Average, Minimum and Maximum salary by Gender. The result of the query should be as shown below.   
   
  
This can be very easily achieved using a simple **GROUP BY** query as show below.

SELECT Gender, COUNT(\*) AS GenderTotal, AVG(Salary) AS AvgSal,

        MIN(Salary) AS MinSal, MAX(Salary) AS MaxSal

FROM Employees

GROUP BY Gender

What if we want **non-aggregated values** (like employee Name and Salary) in result set along with aggregated values   
   
  
You cannot include **non-aggregated** columns in the **GROUP BY** query.

SELECT Name, Salary, Gender, COUNT(\*) AS GenderTotal, AVG(Salary) AS AvgSal,

        MIN(Salary) AS MinSal, MAX(Salary) AS MaxSal

FROM Employees

GROUP BY Gender

The above query will result in the following error  
Column 'Employees.Name' is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause   
  
One way to achieve this is by including the aggregations in a subquery and then **JOINING**it with the main query as shown in the example below. Look at the amount of T-SQL code we have to write.

SELECT Name, Salary, Employees.Gender, Genders.GenderTotals,

        Genders.AvgSal, Genders.MinSal, Genders.MaxSal

FROM Employees

INNER JOIN

(SELECT Gender, COUNT(\*) AS GenderTotals,

          AVG(Salary) AS AvgSal,

         MIN(Salary) AS MinSal, MAX(Salary) AS MaxSal

FROM Employees

GROUP BY Gender) AS Genders

ON Genders.Gender = Employees.Gender

Better way of doing this is by using the **OVER**clause combined with **PARTITION BY**

SELECT Name, Salary, Gender,

        COUNT(Gender) OVER(PARTITION BY Gender) AS GenderTotals,

        AVG(Salary) OVER(PARTITION BY Gender) AS AvgSal,

        MIN(Salary) OVER(PARTITION BY Gender) AS MinSal,

        MAX(Salary) OVER(PARTITION BY Gender) AS MaxSal

FROM Employees

# 109. Row\_Number function in SQL Server

**Suggested Videos**  
[Part 106 - Grouping\_Id function in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/09/groupingid-function-in-sql-server.html)  
[Part 107 - Debugging sql server stored procedures](http://csharp-video-tutorials.blogspot.com/2015/09/debugging-sql-server-stored-procedures.html)  
[Part 108 - Over clause in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/09/over-clause-in-sql-server.html)  
  
  
  
In this video we will discuss Row\_Number function in SQL Server. This is continuation to[Part 108](http://csharp-video-tutorials.blogspot.com/2015/09/over-clause-in-sql-server.html). Please watch [Part 108](http://csharp-video-tutorials.blogspot.com/2015/09/over-clause-in-sql-server.html) from [SQL Server tutorial](https://www.youtube.com/playlist?list=PL08903FB7ACA1C2FB) before proceeding.   
  
  
  
**Row\_Number function**

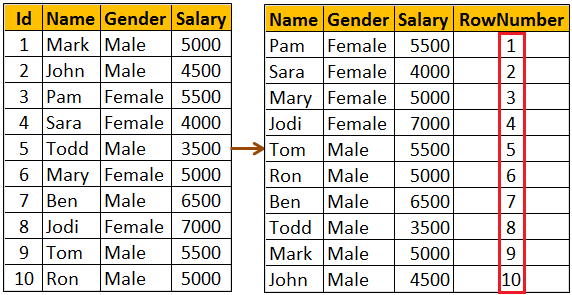
* Introduced in SQL Server 2005
* Returns the sequential number of a row starting at 1
* ORDER BY clause is required
* PARTITION BY clause is optional
* When the data is partitioned, row number is reset to 1 when the partition changes

**Syntax :** ROW\_NUMBER() OVER (ORDER BY Col1, Col2)

**Row\_Number function without PARTITION BY :** In this example, data is not partitioned, so ROW\_NUMBER will provide a consecutive numbering for all the rows in the table based on the order of rows imposed by the ORDER BY clause.

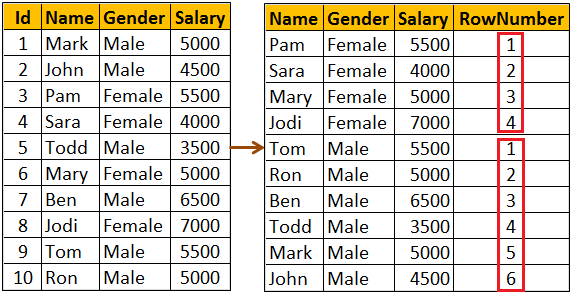
SELECT Name, Gender, Salary,

        ROW\_NUMBER() OVER (ORDER BY Gender) AS RowNumber

FROM Employees  
  
   
  
**Please note :** If ORDER BY clause is not specified you will get the following error  
The function 'ROW\_NUMBER' must have an OVER clause with ORDER BY  
  
**Row\_Number function with PARTITION BY :** In this example, data is partitioned by Gender, so ROW\_NUMBER will provide a consecutive numbering only for the rows with in a parttion. When the partition changes the row number is reset to 1.

SELECT Name, Gender, Salary,

        ROW\_NUMBER() OVER (PARTITION BY Gender ORDER BY Gender) ASRowNumber

FROM Employees  
  
   
  
**Use case for Row\_Number function :** Deleting all duplicate rows except one from a sql server table.   
  
Discussed in detail in[Part 4](http://csharp-video-tutorials.blogspot.com/2014/05/part-4-delete-duplicate-rows-in-sql.html) of [SQL Server Interview Questions and Answers video series](https://www.youtube.com/playlist?list=PL6n9fhu94yhXcztdLO7i6mdyaegC8CJwR).

# 110. Rank and Dense\_Rank in SQL Server

**Suggested Videos**  
[Part 107 - Debugging sql server stored procedures](http://csharp-video-tutorials.blogspot.com/2015/09/debugging-sql-server-stored-procedures.html)  
[Part 108 - Over clause in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/09/over-clause-in-sql-server.html)   
[Part 109 - Row\_Number function in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/09/rownumber-function-in-sql-server.html)   
  
  
  
In this video we will discuss **Rank and Dense\_Rank functions in SQL Server**  
  
  
  
**Rank and Dense\_Rank functions**

* Introduced in SQL Server 2005
* Returns a rank starting at 1 based on the ordering of rows imposed by the ORDER BY clause
* ORDER BY clause is required
* PARTITION BY clause is optional
* When the data is partitioned, rank is reset to 1 when the partition changes

**Difference between Rank and Dense\_Rank functions**  
Rank function skips ranking(s) if there is a tie where as Dense\_Rank will not.   
  
**For example :**If you have 2 rows at rank 1 and you have 5 rows in total.

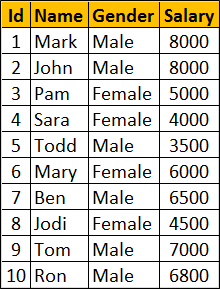
RANK() returns - 1, 1, 3, 4, 5

DENSE\_RANK returns - 1, 1, 2, 3, 4

**Syntax :**

RANK() OVER (ORDER BY Col1, Col2, ...)

DENSE\_RANK() OVER (ORDER BY Col1, Col2, ...)

**Example :** We will use the following **Employees**table for the examples in this video   
   
  
**SQl Script to create Employees table**

Create Table Employees

(

    Id int primary key,

    Name nvarchar(50),

    Gender nvarchar(10),

    Salary int

)

Go

Insert Into Employees Values (1, 'Mark', 'Male', 8000)

Insert Into Employees Values (2, 'John', 'Male', 8000)

Insert Into Employees Values (3, 'Pam', 'Female', 5000)

Insert Into Employees Values (4, 'Sara', 'Female', 4000)

Insert Into Employees Values (5, 'Todd', 'Male', 3500)

Insert Into Employees Values (6, 'Mary', 'Female', 6000)

Insert Into Employees Values (7, 'Ben', 'Male', 6500)

Insert Into Employees Values (8, 'Jodi', 'Female', 4500)

Insert Into Employees Values (9, 'Tom', 'Male', 7000)

Insert Into Employees Values (10, 'Ron', 'Male', 6800)

Go

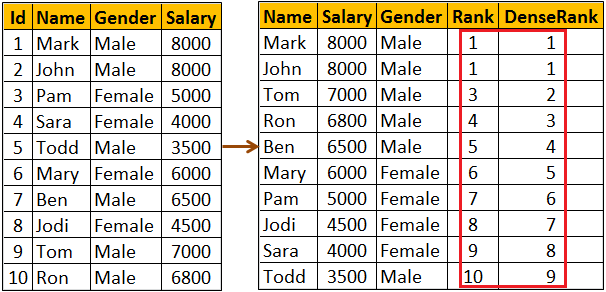
**RANK() and DENSE\_RANK() functions without PARTITION BY clause :** In this example, data is not partitioned, so RANK() function provides a consecutive numbering except when there is a tie. Rank 2 is skipped as there are 2 rows at rank 1. The third row gets rank 3.   
  
DENSE\_RANK() on the other hand will not skip ranks if there is a tie. The first 2 rows get rank 1. Third row gets rank 2.

SELECT Name, Salary, Gender,

RANK() OVER (ORDER BY Salary DESC) AS [Rank],

DENSE\_RANK() OVER (ORDER BY Salary DESC) AS DenseRank

FROM Employees

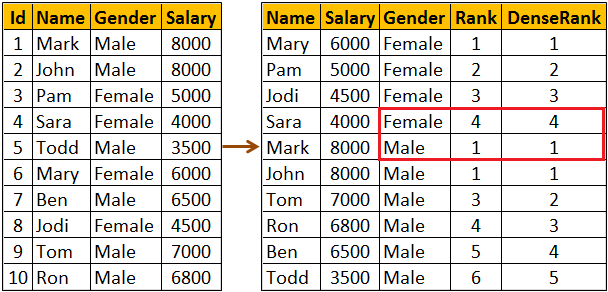
  
  
**RANK() and DENSE\_RANK() functions with PARTITION BY clause :** Notice when the partition changes from Female to Male Rank is reset to 1

SELECT Name, Salary, Gender,

RANK() OVER (PARTITION BY Gender ORDER BY Salary DESC) AS [Rank],

DENSE\_RANK() OVER (PARTITION BY Gender ORDER BY Salary DESC)   
AS DenseRank

FROM Employees

  
  
**Use case for RANK and DENSE\_RANK functions :** Both these functions can be used to find Nth highest salary. However, which function to use depends on what you want to do when there is a tie. Let me explain with an example.   
  
**If there are 2 employees with the FIRST highest salary, there are 2 different business cases**

* If your business case is, not to produce any result for the SECOND highest salary, then use RANK function
* If your business case is to return the next Salary after the tied rows as the SECOND highest Salary, then use DENSE\_RANK function

Since we have 2 Employees with the FIRST highest salary. Rank() function will not return any rows for the SECOND highest Salary.

WITH Result AS

(

    SELECT Salary, RANK() OVER (ORDER BY Salary DESC) AS Salary\_Rank

    FROM Employees

)

SELECT TOP 1 Salary FROM Result WHERE Salary\_Rank = 2

Though we have 2 Employees with the FIRST highest salary. Dense\_Rank() function returns, the next Salary after the tied rows as the SECOND highest Salary

WITH Result AS

(

    SELECT Salary, DENSE\_RANK() OVER (ORDER BY Salary DESC) ASSalary\_Rank

    FROM Employees

)

SELECT TOP 1 Salary FROM Result WHERE Salary\_Rank = 2

You can also use RANK and DENSE\_RANK functions to find the Nth highest Salary among Male or Female employee groups. The following query finds the 3rd highest salary amount paid among the Female employees group

WITH Result AS

(

    SELECT Salary, Gender,

           DENSE\_RANK() OVER (PARTITION BY Gender ORDER BY Salary DESC)

           AS Salary\_Rank

    FROM Employees

)

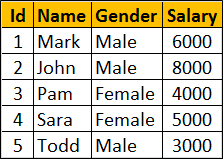
SELECT TOP 1 Salary FROM Result WHERE Salary\_Rank = 3

AND Gender = 'Female'

# 111. Difference between rank dense\_rank and row\_number in SQL

**Suggested Videos**  
[Part 108 - Over clause in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/09/over-clause-in-sql-server.html)   
[Part 109 - Row\_Number function in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/09/rownumber-function-in-sql-server.html)   
[Part 110 - Rank and Dense\_Rank in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/rank-and-denserank-in-sql-server.html)   
  
  
  
In this video we will discuss the similarities and **difference between RANK, DENSE\_RANK and ROW\_NUMBER** functions in SQL Server.   
  
  
  
**Similarities between RANK, DENSE\_RANK and ROW\_NUMBER functions**

* Returns an increasing integer value starting at 1 based on the ordering of rows imposed by the ORDER BY clause (if there are no ties)
* ORDER BY clause is required
* PARTITION BY clause is optional
* When the data is partitioned, the integer value is reset to 1 when the partition changes

We will use the following **Employees**table for the examples in this video  
   
  
**SQL Script to create the Employees table**

Create Table Employees

(

     Id int primary key,

     Name nvarchar(50),

     Gender nvarchar(10),

     Salary int

)

Go

Insert Into Employees Values (1, 'Mark', 'Male', 6000)

Insert Into Employees Values (2, 'John', 'Male', 8000)

Insert Into Employees Values (3, 'Pam', 'Female', 4000)

Insert Into Employees Values (4, 'Sara', 'Female', 5000)

Insert Into Employees Values (5, 'Todd', 'Male', 3000)

Notice that no two employees in the table have the same salary. So all the 3 functions RANK, DENSE\_RANK and ROW\_NUMBER produce the same increasing integer value when ordered by Salary column.

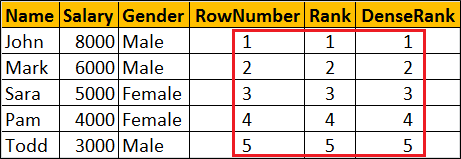
SELECT Name, Salary, Gender,

ROW\_NUMBER() OVER (ORDER BY Salary DESC) AS RowNumber,

RANK() OVER (ORDER BY Salary DESC) AS [Rank],

DENSE\_RANK() OVER (ORDER BY Salary DESC) AS DenseRank

FROM Employees

   
  
You will only see the difference when there ties (duplicate values in the column used in the ORDER BY clause).   
  
Now let's include duplicate values for Salary column.    
  
To do this   
**First delete existing data from the Employees table**  
DELETE FROM Employees   
  
**Insert new rows with duplicate valuse for Salary column**

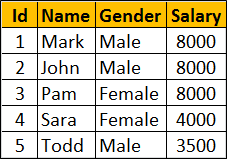
Insert Into Employees Values (1, 'Mark', 'Male', 8000)

Insert Into Employees Values (2, 'John', 'Male', 8000)

Insert Into Employees Values (3, 'Pam', 'Female', 8000)

Insert Into Employees Values (4, 'Sara', 'Female', 4000)

Insert Into Employees Values (5, 'Todd', 'Male', 3500)

At this point data in the Employees table should be as shown below   
   
  
Notice 3 employees have the same salary 8000. When you execute the following query you can clearly see the difference between RANK, DENSE\_RANK and ROW\_NUMBER functions.

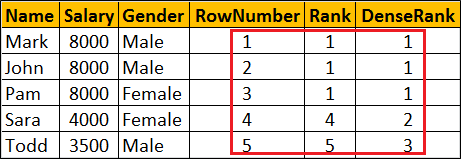
SELECT Name, Salary, Gender,

ROW\_NUMBER() OVER (ORDER BY Salary DESC) AS RowNumber,

RANK() OVER (ORDER BY Salary DESC) AS [Rank],

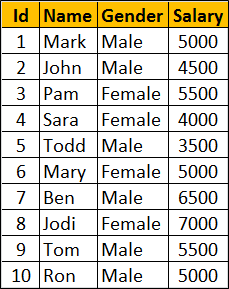
DENSE\_RANK() OVER (ORDER BY Salary DESC) AS DenseRank

FROM Employees

   
  
**Difference between RANK, DENSE\_RANK and ROW\_NUMBER functions**

* **ROW\_NUMBER :** Returns an increasing unique number for each row starting at 1, even if there are duplicates.
* **RANK :**Returns an increasing unique number for each row starting at 1. When there are duplicates, same rank is assigned to all the duplicate rows, but the next row after the duplicate rows will have the rank it would have been assigned if there had been no duplicates. So RANK function skips rankings if there are duplicates.
* **DENSE\_RANK :**Returns an increasing unique number for each row starting at 1. When there are duplicates, same rank is assigned to all the duplicate rows but the DENSE\_RANK function will not skip any ranks. This means the next row after the duplicate rows will have the next rank in the sequence.

# 112. Calculate running total in SQL Server 2012

**Suggested Videos**  
[Part 109 - Row\_Number function in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/09/rownumber-function-in-sql-server.html)  
[Part 110 - Rank and Dense\_Rank in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/rank-and-denserank-in-sql-server.html)   
[Part 111 - Difference between rank dense\_rank and row\_number in SQL](http://csharp-video-tutorials.blogspot.com/2015/10/difference-between-rank-denserank-and.html)  
  
  
  
In this video we will discuss how to calculate running total in SQL Server 2012 and later versions.   
  
  
  
We will use the following **Employees table** for the examples in this video.  
  
  
**SQL Script to create Employees table**  
Create Table Employees

(

     Id int primary key,

     Name nvarchar(50),

     Gender nvarchar(10),

     Salary int

)

Go

Insert Into Employees Values (1, 'Mark', 'Male', 5000)

Insert Into Employees Values (2, 'John', 'Male', 4500)

Insert Into Employees Values (3, 'Pam', 'Female', 5500)

Insert Into Employees Values (4, 'Sara', 'Female', 4000)

Insert Into Employees Values (5, 'Todd', 'Male', 3500)

Insert Into Employees Values (6, 'Mary', 'Female', 5000)

Insert Into Employees Values (7, 'Ben', 'Male', 6500)

Insert Into Employees Values (8, 'Jodi', 'Female', 7000)

Insert Into Employees Values (9, 'Tom', 'Male', 5500)

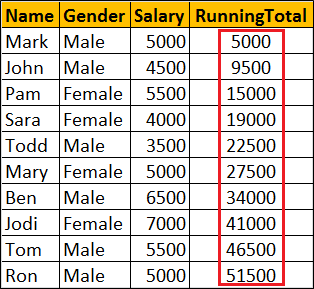
Insert Into Employees Values (10, 'Ron', 'Male', 5000)

Go

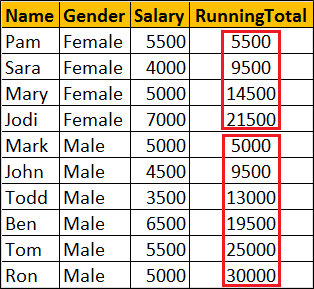
**SQL Query to compute running total without partitions**

SELECT Name, Gender, Salary,

        SUM(Salary) OVER (ORDER BY ID) AS RunningTotal

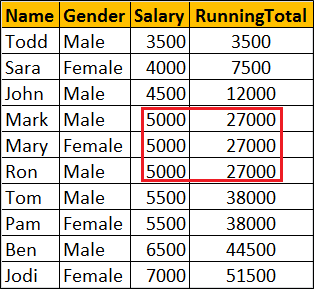
FROM Employees  
  
  
  
**SQL Query to compute running total with partitions**  
SELECT Name, Gender, Salary,

        SUM(Salary) OVER (PARTITION BY Gender ORDER BY ID) AS RunningTotal

FROM Employees  
  
  
  
**What happens if I use order by on Salary column**  
If you have duplicate values in the Salary column, all the duplicate values will be added to the running total at once. In the example below notice that we have 5000 repeated 3 times. So 15000 (i.e 5000 + 5000 + 5000) is added to the running total at once.

SELECT Name, Gender, Salary,

        SUM(Salary) OVER (ORDER BY Salary) AS RunningTotal

FROM Employees  
  
  
  
So when computing running total, it is better to use a column that has unique data in the ORDER BY clause.

# 113. NTILE function in SQL Server

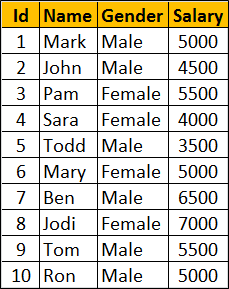
**Suggested Videos**  
[Part 110 - Rank and Dense\_Rank in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/rank-and-denserank-in-sql-server.html)  
[Part 111 - Difference between rank dense\_rank and row\_number in SQL](http://csharp-video-tutorials.blogspot.com/2015/10/difference-between-rank-denserank-and.html)  
[Part 112 - Calculate running total in SQL Server 2012](http://csharp-video-tutorials.blogspot.com/2015/10/calculate-running-total-in-sql-server.html)   
  
  
  
In this video we will discuss **NTILE function in SQL Server**   
  
  
  
**NTILE function**

* Introduced in SQL Server 2005
* ORDER BY Clause is required
* PARTITION BY clause is optional
* Distributes the rows into a specified number of groups
* If the number of rows is not divisible by number of groups, you may have groups of two different sizes.
* Larger groups come before smaller groups

**For example**

* NTILE(2) of 10 rows divides the rows in 2 Groups (5 in each group)
* NTILE(3) of 10 rows divides the rows in 3 Groups (4 in first group, 3 in 2nd & 3rd group)

**Syntax :** NTILE (Number\_of\_Groups) OVER (ORDER BY Col1, Col2, ...)

We will use the following **Employees table** for the examples in this video.  


**SQL Script to create Employees table**

Create Table Employees

(

    Id int primary key,

    Name nvarchar(50),

    Gender nvarchar(10),

    Salary int

)

Go

Insert Into Employees Values (1, 'Mark', 'Male', 5000)

Insert Into Employees Values (2, 'John', 'Male', 4500)

Insert Into Employees Values (3, 'Pam', 'Female', 5500)

Insert Into Employees Values (4, 'Sara', 'Female', 4000)

Insert Into Employees Values (5, 'Todd', 'Male', 3500)

Insert Into Employees Values (6, 'Mary', 'Female', 5000)

Insert Into Employees Values (7, 'Ben', 'Male', 6500)

Insert Into Employees Values (8, 'Jodi', 'Female', 7000)

Insert Into Employees Values (9, 'Tom', 'Male', 5500)

Insert Into Employees Values (10, 'Ron', 'Male', 5000)

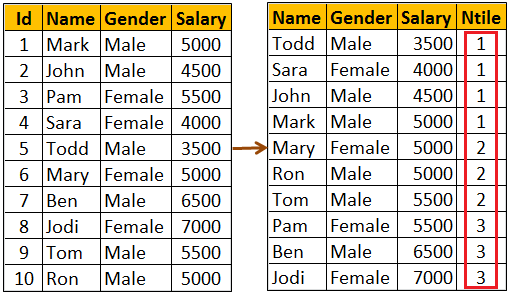
Go

**NTILE function without PARTITION BY clause :** Divides the 10 rows into 3 groups. 4 rows in first group, 3 rows in the 2nd & 3rd group.

SELECT Name, Gender, Salary,

NTILE(3) OVER (ORDER BY Salary) AS [Ntile]

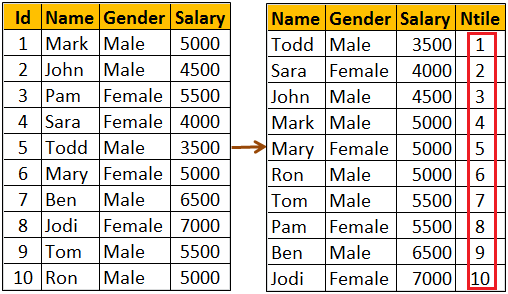
FROM Employees

   
  
**What if the specified number of groups is GREATER THAN the number of rows**  
NTILE function will try to create as many groups as possible with one row in each group.   
  
With 10 rows in the table, NTILE(11) will create 10 groups with 1 row in each group.

SELECT Name, Gender, Salary,

NTILE(11) OVER (ORDER BY Salary) AS [Ntile]

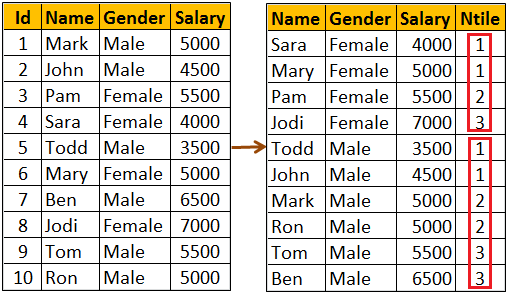
FROM Employees

   
  
**NTILE function with PARTITION BY clause :** When the data is partitioned, NTILE function creates the specified number of groups with in each partition.  
  
The following query partitions the data into 2 partitions (Male & Female). NTILE(3) creates 3 groups in each of the partitions.

SELECT Name, Gender, Salary,

NTILE(3) OVER (PARTITION BY GENDER ORDER BY Salary) AS [Ntile]

FROM Employees



# 114. Lead and Lag functions in SQL Server 2012

**Suggested Videos**  
[Part 111 - Difference between rank dense\_rank and row\_number in SQL](http://csharp-video-tutorials.blogspot.com/2015/10/difference-between-rank-denserank-and.html)  
[Part 112 - Calculate running total in SQL Server 2012](http://csharp-video-tutorials.blogspot.com/2015/10/calculate-running-total-in-sql-server.html)   
[Part 113 - NTILE function in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/ntile-function-in-sql-server.html)  
  
  
  
In this video we will discuss about Lead and Lag functions.   
  
  
  
**Lead and Lag functions**

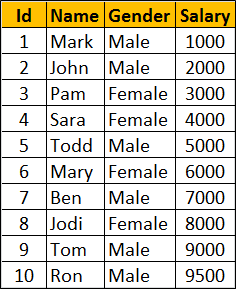
* Introduced in SQL Server 2012
* Lead function is used to access subsequent row data along with current row data
* Lag function is used to access previous row data along with current row data
* ORDER BY clause is required
* PARTITION BY clause is optional

**Syntax**

LEAD(Column\_Name, Offset, Default\_Value) OVER (ORDER BY Col1, Col2, ...)

LAG(Column\_Name, Offset, Default\_Value) OVER (ORDER BY Col1, Col2, ...)

* **Offset -**Number of rows to lead or lag.
* **Default\_Value -** The default value to return if the number of rows to lead or lag goes beyond first row or last row in a table or partition. If default value is not specified NULL is returned.

We will use the following **Employees table** for the examples in this video  
  
  
**SQL Script to create the Employees table**

Create Table Employees

(

     Id int primary key,

     Name nvarchar(50),

     Gender nvarchar(10),

     Salary int

)

Go

Insert Into Employees Values (1, 'Mark', 'Male', 1000)

Insert Into Employees Values (2, 'John', 'Male', 2000)

Insert Into Employees Values (3, 'Pam', 'Female', 3000)

Insert Into Employees Values (4, 'Sara', 'Female', 4000)

Insert Into Employees Values (5, 'Todd', 'Male', 5000)

Insert Into Employees Values (6, 'Mary', 'Female', 6000)

Insert Into Employees Values (7, 'Ben', 'Male', 7000)

Insert Into Employees Values (8, 'Jodi', 'Female', 8000)

Insert Into Employees Values (9, 'Tom', 'Male', 9000)

Insert Into Employees Values (10, 'Ron', 'Male', 9500)

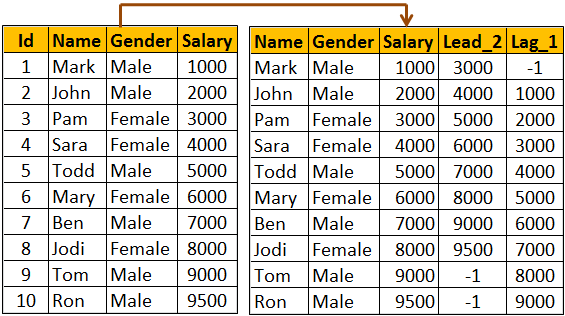
Go  
  
**Lead and Lag functions example WITHOUT partitions :** This example Leads 2 rows and Lags 1 row from the current row.

* When you are on the first row, LEAD(Salary, 2, -1) allows you to move forward 2 rows and retrieve the salary from the 3rd row.
* When you are on the first row, LAG(Salary, 1, -1) allows us to move backward 1 row. Since there no rows beyond row 1, Lag function in this case returns the default value -1.
* When you are on the last row, LEAD(Salary, 2, -1) allows you to move forward 2 rows. Since there no rows beyond the last row 1, Lead function in this case returns the default value -1.
* When you are on the last row, LAG(Salary, 1, -1) allows us to move backward 1 row and retrieve the salary from the previous row.

SELECT Name, Gender, Salary,

        LEAD(Salary, 2, -1) OVER (ORDER BY Salary) AS Lead\_2,

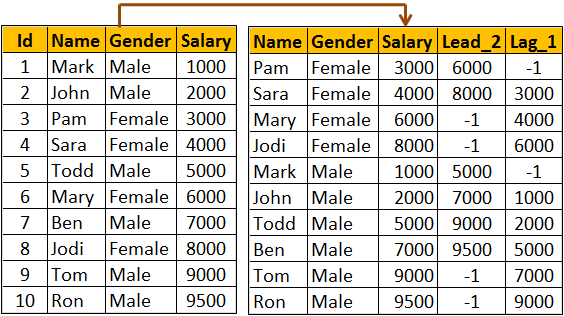
        LAG(Salary, 1, -1) OVER (ORDER BY Salary) AS Lag\_1

FROM Employees  
  
  
  
**Lead and Lag functions example WITH partitions :** Notice that in this example, Lead and Lag functions return default value if the number of rows to lead or lag goes beyond first row or last row in the partition.

SELECT Name, Gender, Salary,

        LEAD(Salary, 2, -1) OVER (PARTITION By Gender ORDER BY Salary) ASLead\_2,

        LAG(Salary, 1, -1) OVER (PARTITION By Gender ORDER BY Salary) AS Lag\_1

FROM Employees  
  


# 115. FIRST\_VALUE function in SQL Server

**Suggested Videos**  
[Part 112 - Calculate running total in SQL Server 2012](http://csharp-video-tutorials.blogspot.com/2015/10/calculate-running-total-in-sql-server.html)  
[Part 113 - NTILE function in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/ntile-function-in-sql-server.html)  
[Part 114 - Lead and Lag functions in SQL Server 2012](http://csharp-video-tutorials.blogspot.com/2015/10/lead-and-lag-functions-in-sql-server.html)  
  
  
  
In this video we will discuss FIRST\_VALUE function in SQL Server   
  
  
  
**FIRST\_VALUE function**

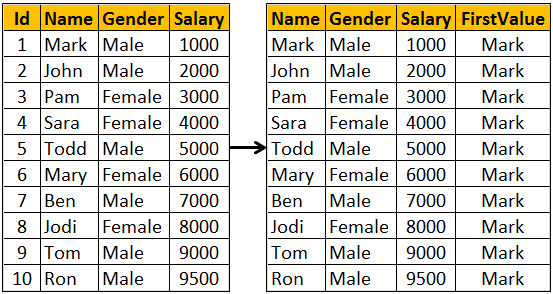
* Introduced in SQL Server 2012
* Retrieves the first value from the specified column
* ORDER BY clause is required
* PARTITION BY clause is optional

**Syntax :** FIRST\_VALUE(Column\_Name) OVER (ORDER BY Col1,Col2, ...)

**FIRST\_VALUE function example WITHOUT partitions :** In the following example, FIRST\_VALUE function returns the name of the lowest paid employee from the entire table.

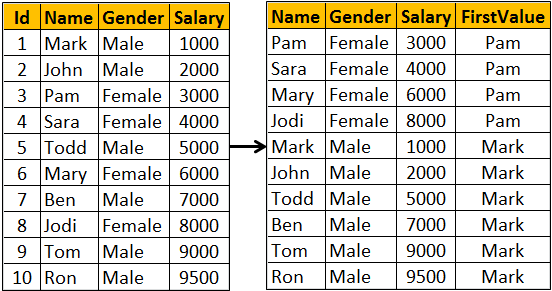
SELECT Name, Gender, Salary,

FIRST\_VALUE(Name) OVER (ORDER BY Salary) AS FirstValue

FROM Employees  
  
   
  
**FIRST\_VALUE function example WITH partitions :** In the following example, FIRST\_VALUE function returns the name of the lowest paid employee from the respective partition.

SELECT Name, Gender, Salary,

FIRST\_VALUE(Name) OVER (PARTITION BY Gender ORDER BY Salary) ASFirstValue

FROM Employees  
  


# 116. Window functions in SQL Server

**Suggested Videos**  
[Part 113 - NTILE function in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/ntile-function-in-sql-server.html)  
[Part 114 - Lead and Lag functions in SQL Server 2012](http://csharp-video-tutorials.blogspot.com/2015/10/lead-and-lag-functions-in-sql-server.html)  
[Part 115 - FIRST\_VALUE function in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/firstvalue-function-in-sql-server.html)   
  
  
  
In this video we will discuss **window functions in SQL Server**   
  
  
  
In SQL Server we have different categories of window functions

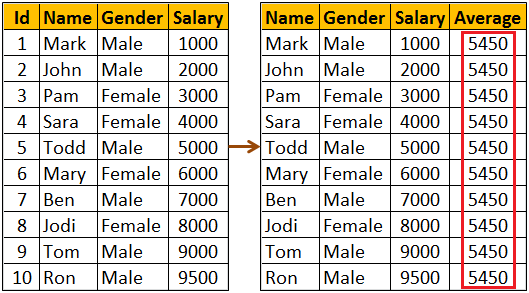
* **Aggregate functions -** AVG, SUM, COUNT, MIN, MAX etc..
* **Ranking functions -** RANK, DENSE\_RANK, ROW\_NUMBER etc..
* **Analytic functions -** LEAD, LAG, FIRST\_VALUE, LAST\_VALUE etc...

**OVER** Clause defines the partitioning and ordering of a rows (i.e a window) for the above functions to operate on. Hence these functions are called window functions. The OVER clause accepts the following three arguments to define a window for these functions to operate on.

* **ORDER BY :** Defines the logical order of the rows
* **PARTITION BY :** Divides the query result set into partitions. The window function is applied to each partition separately.
* **ROWSor RANGE clause :** Further limits the rows within the partition by specifying start and end points within the partition.

The default for **ROWS**or **RANGE**clause is

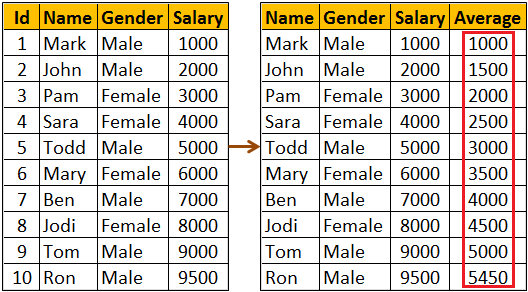
RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW

Let us understand the use of **ROWS**or **RANGE**clause with an example.    
  
Compute average salary and display it against every employee row as shown below.  
   
  
We might think the following query would do the job.

SELECT Name, Gender, Salary,

        AVG(Salary) OVER(ORDER BY Salary) AS Average

FROM Employees

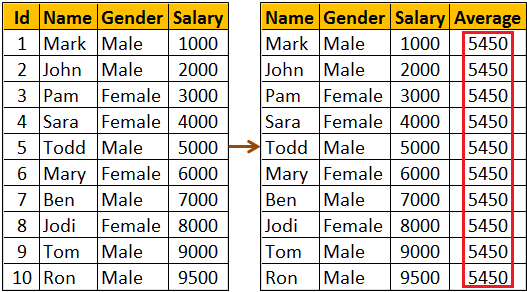
As you can see from the result below, the above query does not produce the overall salary average. It produces the average of the current row and the rows preceeding the current row. This is because, the default value of ROWS or RANGE clause (RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) is applied.   
   
  
To fix this, provide an explicit value for ROWS or RANGE clause as shown below. ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING tells the window function to operate on the set of rows starting from the first row in the partition to the last row in the partition.

SELECT Name, Gender, Salary,

        AVG(Salary) OVER(ORDER BY Salary ROWS BETWEEN

        UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS Average

FROM Employees

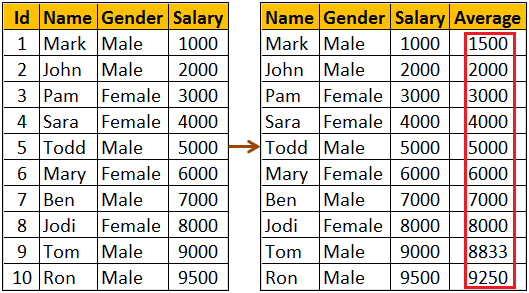
   
  
The same result can also be achieved by using RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING  
  
**What is the difference between ROWS and RANGE**  
We will discuss this in a later video  
  
The following query can be used if you want to compute the average salary of   
1. The current row  
2. One row PRECEDING the current row and   
3. One row FOLLOWING the current row

SELECT Name, Gender, Salary,

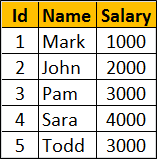
        AVG(Salary) OVER(ORDER BY Salary

        ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING) AS Average

FROM Employees



# 117. Difference between rows and range

**Suggested Videos**  
[Part 114 - Lead and Lag functions in SQL Server 2012](http://csharp-video-tutorials.blogspot.com/2015/10/lead-and-lag-functions-in-sql-server.html)  
[Part 115 - FIRST\_VALUE function in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/firstvalue-function-in-sql-server.html)   
[Part 116 - Window functions in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/window-functions-in-sql-server.html)   
  
  
  
In this video we will discuss the **difference between rows and range in SQL Server**. This is continuation to [Part 116](http://csharp-video-tutorials.blogspot.com/2015/10/window-functions-in-sql-server.html). Please watch [Part 116](http://csharp-video-tutorials.blogspot.com/2015/10/window-functions-in-sql-server.html) from [SQL Server tutorial](https://www.youtube.com/playlist?list=PL08903FB7ACA1C2FB) before proceeding.   
  
  
  
Let us understand the difference with an example. We will use the following **Employees**table in this demo.  
  
  
**SQL Script to create the Employees table**

Create Table Employees

(

     Id int primary key,

     Name nvarchar(50),

     Salary int

)

Go

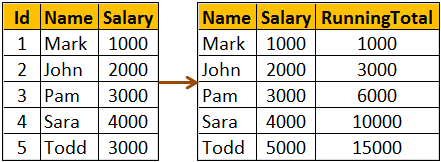
Insert Into Employees Values (1, 'Mark', 1000)

Insert Into Employees Values (2, 'John', 2000)

Insert Into Employees Values (3, 'Pam', 3000)

Insert Into Employees Values (4, 'Sara', 4000)

Insert Into Employees Values (5, 'Todd', 5000)

Go  
  
**Calculate the running total of Salary and display it against every employee row**  
  
  
The following query calculates the running total. We have not specified an explicit value for ROWS or RANGE clause.

SELECT Name, Salary,

        SUM(Salary) OVER(ORDER BY Salary) AS RunningTotal

FROM Employees

So the above query is using the default value which is

RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW

This means the above query can be re-written using an explicit value for ROWS or RANGE clause as shown below.

SELECT Name, Salary,

        SUM(Salary) OVER(ORDER BY Salary

        RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) ASRunningTotal

FROM Employees  
  
We can also achieve the same result, by replacing RANGE with ROWS

SELECT Name, Salary,

        SUM(Salary) OVER(ORDER BY Salary

        ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) ASRunningTotal

FROM Employees  
  
**What is the difference between ROWS and RANGE**  
To understand the difference we need some duplicate values for the Salary column in the Employees table.  
  
Execute the following UPDATE script to introduce duplicate values in the Salary column

Update Employees set Salary = 1000 where Id = 2

Update Employees set Salary = 3000 where Id = 4

Go

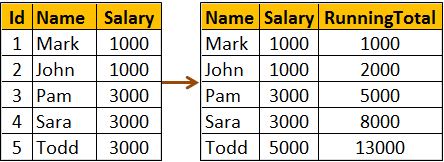
Now execute the following query. Notice that we get the running total as expected.

SELECT Name, Salary,

        SUM(Salary) OVER(ORDER BY Salary

        ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) ASRunningTotal

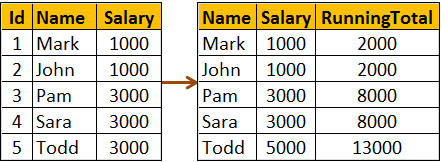
FROM Employees

  
  
**The following query uses RANGE instead of ROWS**

SELECT Name, Salary,

        SUM(Salary) OVER(ORDER BY Salary

        RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) ASRunningTotal

FROM Employees  
  
You get the following result when you execute the above query. Notice we don't get the running total as expected.  
  
  
So, the main difference between ROWS and RANGE is in the way duplicate rows are treated. ROWS treat duplicates as distinct values, where as RANGE treats them as a single entity.  
  
All together side by side. The following query shows how running total changes  
1. When no value is specified for ROWS or RANGE clause  
2. When RANGE clause is used explicitly with it's default value  
3. When ROWS clause is used instead of RANGE clause

SELECT Name, Salary,

        SUM(Salary) OVER(ORDER BY Salary) AS [Default],

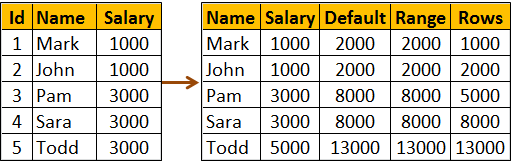
        SUM(Salary) OVER(ORDER BY Salary

        RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS[Range],

        SUM(Salary) OVER(ORDER BY Salary

        ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS[Rows]

FROM Employees



# 118. LAST\_VALUE function in SQL Server

**Suggested Videos**  
[Part 115 - FIRST\_VALUE function in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/firstvalue-function-in-sql-server.html)  
[Part 116 - Window functions in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/window-functions-in-sql-server.html)   
[Part 117 - Difference between rows and range](http://csharp-video-tutorials.blogspot.com/2015/10/difference-between-rows-and-range.html)  
  
  
  
In this video we will discuss LAST\_VALUE function in SQL Server.   
  
  
  
**LAST\_VALUE function**

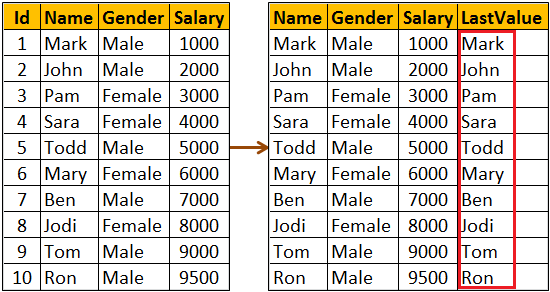
* Introduced in SQL Server 2012
* Retrieves the last value from the specified column
* ORDER BY clause is required
* PARTITION BY clause is optional
* ROWS or RANGE clause is optional, but for it to work correctly you may have to explicitly specify a value

**Syntax :** LAST\_VALUE(Column\_Name) OVER (ORDER BY Col1, Col2, ...)

**LAST\_VALUE function not working as expected :** In the following example, LAST\_VALUE function does not return the name of the highest paid employee. This is because we have not specified an explicit value for ROWS or RANGE clause. As a result it is using it's default value RANGE BETWEEN UNBOUNDED PRECEDING ANDCURRENT ROW.

SELECT Name, Gender, Salary,

    LAST\_VALUE(Name) OVER (ORDER BY Salary) AS LastValue

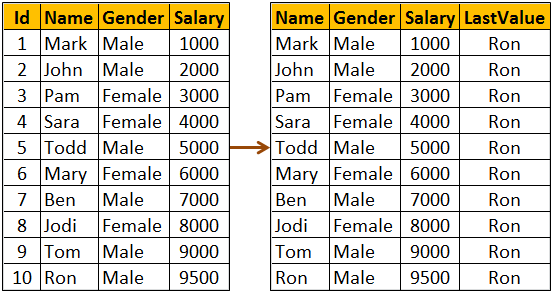
FROM Employees  
  
  
  
**LAST\_VALUE function working as expected :** In the following example, LAST\_VALUE function returns the name of the highest paid employee as expected. Notice we have set an explicit value for ROWS or RANGE clause to ROWS BETWEEN UNBOUNDEDPRECEDING AND UNBOUNDED FOLLOWING

This tells the LAST\_VALUE function that it's window starts at the first row and ends at the last row in the result set.

SELECT Name, Gender, Salary,

    LAST\_VALUE(Name) OVER (ORDER BY Salary ROWS BETWEEN UNBOUNDEDPRECEDING AND UNBOUNDED FOLLOWING) AS LastValue

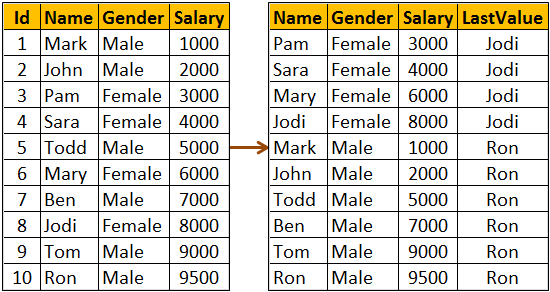
FROM Employees

  
  
**LAST\_VALUE function example with partitions :** In the following example, LAST\_VALUE function returns the name of the highest paid employee from the respective partition.

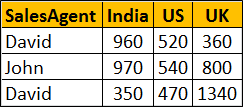
SELECT Name, Gender, Salary,

    LAST\_VALUE(Name) OVER (PARTITION BY Gender ORDER BY Salary

    ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING)AS LastValue

FROM Employees  
  


# 119. UNPIVOT in SQL Server

**Suggested Videos**  
[Part 116 - Window functions in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/window-functions-in-sql-server.html)  
[Part 117 - Difference between rows and range](http://csharp-video-tutorials.blogspot.com/2015/10/difference-between-rows-and-range.html)  
[Part 118 - LAST\_VALUE function in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/lastvalue-function-in-sql-server.html)   
  
  
  
In this video we will discuss UNPIVOT operator in SQL Server.    
  
  
  
PIVOT operator turns ROWS into COLUMNS, where as UNPIVOT turns COLUMNS into ROWS.  
  
We discussed PIVOT operator in [Part 54](http://csharp-video-tutorials.blogspot.com/2012/10/pivot-operator-in-sql-server-part-54.html)of [SQL Server tutorial](https://www.youtube.com/playlist?list=PL08903FB7ACA1C2FB). Please watch [Part 54](http://csharp-video-tutorials.blogspot.com/2012/10/pivot-operator-in-sql-server-part-54.html)before proceeding.  
  
Let us understand UNPIVOT with an example. We will use the following tblProductSales table in this demo.  
  
  
  
**SQL Script to create tblProductSales table**

Create Table tblProductSales

(

 SalesAgent nvarchar(50),

 India int,

 US int,

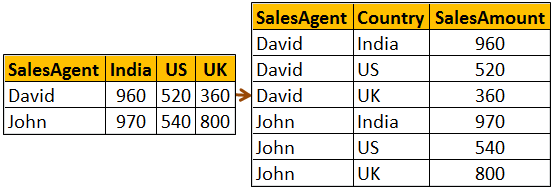
 UK int

)

Go

Insert into tblProductSales values ('David', 960, 520, 360)

Insert into tblProductSales values ('John', 970, 540, 800)

Go  
  
Write a query to turn COLUMNS into ROWS. The result of the query should be as shown below.   
   
  
SELECT SalesAgent, Country, SalesAmount

FROM tblProductSales

UNPIVOT

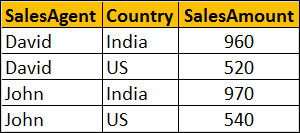
(

       SalesAmount

       FOR Country IN (India, US ,UK)

) AS UnpivotExample

# 120. Reverse PIVOT table in SQL Server

**Suggested Videos**  
[Part 117 - Difference between rows and range](http://csharp-video-tutorials.blogspot.com/2015/10/difference-between-rows-and-range.html)  
[Part 118 - LAST\_VALUE function in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/lastvalue-function-in-sql-server.html)   
[Part 119 - UNPIVOT in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/unpivot-in-sql-server.html)   
  
  
  
In this video we will discuss if it's always possible to reverse what PIVOT operator has done using UNPIVOT operator.   
  
  
  
**Is it always possible to reverse what PIVOT operator has done using UNPIVOT operator.**  
No, not always. If the PIVOT operator has not aggregated the data, you can get your original data back using the UNPIVOT operator but not if the data is aggregated.   
  
Let us understand this with an example. We will use the following table **tblProductSales**for the examples in this video.   
   
  
**SQL Script to create tblProductSales table**

Create Table tblProductSales

(

     SalesAgent nvarchar(10),

     Country nvarchar(10),

     SalesAmount int

)

Go

Insert into tblProductSales values('David','India',960)

Insert into tblProductSales values('David','US',520)

Insert into tblProductSales values('John','India',970)

Insert into tblProductSales values('John','US',540)

Go

**Let's now use the PIVOT operator to turn ROWS into COLUMNS**

SELECT SalesAgent, India, US

FROM tblProductSales

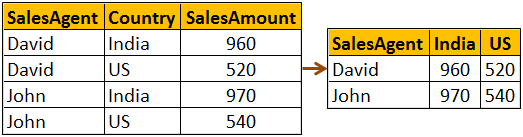
PIVOT

(

     SUM(SalesAmount)

     FOR Country IN (India, US)

) AS PivotTable

The above query produces the following output   
   
  
**Now let's use the UNPIVOT operator to reverse what PIVOT operator has done**

SELECT SalesAgent, Country, SalesAmount

FROM

(SELECT SalesAgent, India, US

FROM tblProductSales

PIVOT

(

     SUM(SalesAmount)

     FOR Country IN (India, US)

) AS PivotTable) P

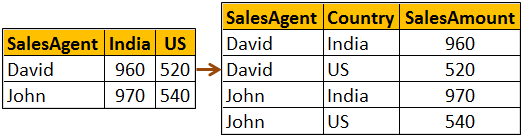
UNPIVOT

(

     SalesAmount

     FOR Country IN (India, US)

) AS UnpivotTable

The above query reverses what PIVOT operator has done, and we get the original data back as shown below. We are able to get the original data back, because the SUM aggregate function that we used with the PIVOT operator did not perform any aggregation.  
   
  
Now execute the following **INSERT** statement to insert a new row into **tblProductSales**table.

Insert into tblProductSales values('David','India',100)

With this new row in the table, if you execute the following **PIVOT** query data will be aggregated

SELECT SalesAgent, India, US

FROM tblProductSales

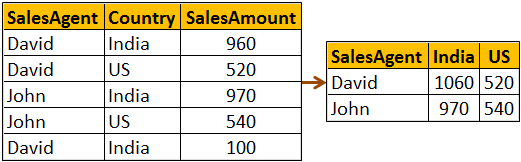
PIVOT

(

     SUM(SalesAmount)

     FOR Country IN (India, US)

) AS PivotTable

**The following is the result of the above query**   
   
  
Now if we use UNPIVOT opertaor with the above query, we wouldn't get our orginial data back as the PIVOT operator has already aggrgated the data, and there is no way for SQL Server to know how to undo the aggregations.

SELECT SalesAgent, Country, SalesAmount

FROM

(SELECT SalesAgent, India, US

FROM tblProductSales

PIVOT

(

     SUM(SalesAmount)

     FOR Country IN (India, US)

) AS PivotTable) P

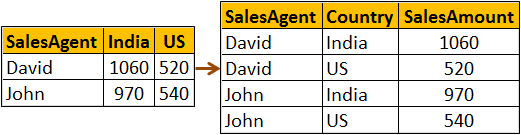
UNPIVOT

(

     SalesAmount

     FOR Country IN (India, US)

) AS UnpivotTable

Notice that for SalesAgent - David and Country - India we get only one row. In the original table we had 2 rows for the same combination.   


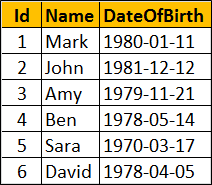
# 121. Choose function in SQL Server

**Suggested Videos**  
[Part 118 - LAST\_VALUE function in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/lastvalue-function-in-sql-server.html)  
[Part 119 - UNPIVOT in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/unpivot-in-sql-server.html)   
[Part 120 - Reverse PIVOT table in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/reverse-pivot-table-in-sql-server.html)   
  
  
  
In this video we will discuss **Choose function in SQL Server**  
  
  
  
**Choose function**

* Introduced in SQL Server 2012
* Returns the item at the specified index from the list of available values
* The index position starts at 1 and NOT 0 (ZERO)

**Syntax :**CHOOSE( index, val\_1, val\_2, ... )   
  
**Example :** Returns the item at index position 2

SELECT CHOOSE(2, 'India','US', 'UK') AS Country

**Output :**   
Choose function in SQL Server   
  
**Example :** Using CHOOSE() function with table data. We will use the following**Employees**table for this example.   
   
  
**SQL Script to create Employees table**

Create table Employees

(

     Id int primary key identity,

     Name nvarchar(10),

     DateOfBirth date

)

Go

Insert into Employees values ('Mark', '01/11/1980')

Insert into Employees values ('John', '12/12/1981')

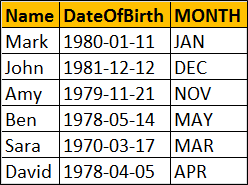
Insert into Employees values ('Amy', '11/21/1979')

Insert into Employees values ('Ben', '05/14/1978')

Insert into Employees values ('Sara', '03/17/1970')

Insert into Employees values ('David', '04/05/1978')

Go

We want to display Month name along with employee Name and Date of Birth.   
   
  
**Using CASE statement in SQL Server**

SELECT Name, DateOfBirth,

        CASE DATEPART(MM, DateOfBirth)

            WHEN 1 THEN 'JAN'

            WHEN 2 THEN 'FEB'

            WHEN 3 THEN 'MAR'

            WHEN 4 THEN 'APR'

            WHEN 5 THEN 'MAY'

            WHEN 6 THEN 'JUN'

            WHEN 7 THEN 'JUL'

            WHEN 8 THEN 'AUG'

            WHEN 9 THEN 'SEP'

            WHEN 10 THEN 'OCT'

            WHEN 11 THEN 'NOV'

            WHEN 12 THEN 'DEC'

        END

       AS [MONTH]

FROM Employees

**Using CHOOSE function in SQL Server :** The amount of code we have to write is lot less than using CASE statement.

SELECT Name, DateOfBirth,CHOOSE(DATEPART(MM, DateOfBirth),

       'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL', 'AUG',

       'SEP', 'OCT', 'NOV', 'DEC') AS [MONTH]

FROM Employees

# 122. IIF function in SQL Server

**Suggested Videos**  
[Part 119 - UNPIVOT in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/unpivot-in-sql-server.html)  
[Part 120 - Reverse PIVOT table in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/reverse-pivot-table-in-sql-server.html)   
[Part 121 - Choose function in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/choose-function-in-sql-server.html)   
  
  
  
In this video we will discuss **IIF function in SQL Server**.   
  
  
  
**IIF function**

* Introduced in SQL Server 2012
* Returns one of two the values, depending on whether the Boolean expression evaluates to true or false
* IIF is a shorthand way for writing a CASE expression

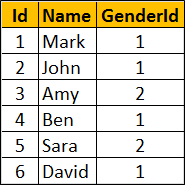
**Syntax :**IIF ( boolean\_expression, true\_value, false\_value )

**Example :** Returns Male as the boolean expression evaluates to TRUE

DECLARE @Gender INT

SET @Gender = 1

SELECT IIF( @Gender = 1, 'Male', 'Femlae') AS Gender

Output :    
iif function in sql server example   
  
**Example :** Using IIF() function with table data. We will use the following **Employees**table for this example.   
   
  
**SQL Script to create Employees table**

Create table Employees

(

     Id int primary key identity,

     Name nvarchar(10),

     GenderId int

)

Go

Insert into Employees values ('Mark', 1)

Insert into Employees values ('John', 1)

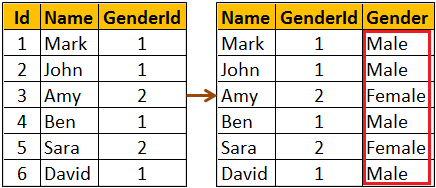
Insert into Employees values ('Amy', 2)

Insert into Employees values ('Ben', 1)

Insert into Employees values ('Sara', 2)

Insert into Employees values ('David', 1)

Go

Write a query to display **Gender**along with **employee Name**and **GenderId**. We can achieve this either by using **CASE**or **IIF**.   
   
  
**Using CASE statement**

SELECT Name, GenderId,

        CASE WHEN GenderId = 1

                      THEN 'Male'

                      ELSE 'Female'

                   END AS Gender

FROM Employees

**Using IIF function**

SELECT Name, GenderId, IIF(GenderId = 1, 'Male', 'Female') AS Gender

FROM Employees

# 123. TRY\_PARSE function in SQL Server 2012

**Suggested Videos**  
[Part 120 - Reverse PIVOT table in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/reverse-pivot-table-in-sql-server.html)  
[Part 121 - Choose function in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/choose-function-in-sql-server.html)   
[Part 122 - IIF function in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/iif-function-in-sql-server.html)   
  
  
  
**In this video we will discuss**

* TRY\_PARSE function
* Difference between PARSE and TRY\_PARSE functions

**TRY\_PARSE function**

* Introduced in SQL Server 2012
* Converts a string to Date/Time or Numeric type
* Returns NULL if the provided string cannot be converted to the specified data type
* Requires .NET Framework Common Language Runtime (CLR)

**Syntax :** TRY\_PARSE ( string\_value AS data\_type )

**Example :** Convert string to INT. As the string can be converted to INT, the result will be 99 as expected.   
  
SELECT TRY\_PARSE('99' AS INT) AS Result   
  
**Output :**   
try_parse function in sql server 2012   
  
**Example :** Convert string to INT. The string cannot be converted to INT, so TRY\_PARSE returns NULL  
  
SELECT TRY\_PARSE('ABC' AS INT) AS Result  
  
**Output :**   
sql server tryparse   
  
Use **CASE**statement or **IIF**function to provide a meaningful error message instead of NULL when the conversion fails.  
  
**Example :**Using CASE statement to provide a meaningful error message when the conversion fails.

SELECT

CASE WHEN TRY\_PARSE('ABC' AS INT) IS NULL

           THEN 'Conversion Failed'

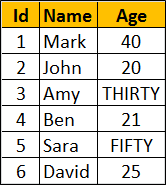
           ELSE 'Conversion Successful'

END AS Result

**Output :** As the conversion fails, you will now get a message 'Conversion Failed' instead of NULL   
sql server try_parse   
  
**Example :**Using IIF function to provide a meaningful error message when the conversion fails.

SELECT IIF(TRY\_PARSE('ABC' AS INT) IS NULL, 'Conversion Failed',

                 'Conversion Successful') AS Result

**What is the difference between PARSE and TRY\_PARSE**  
PARSE will result in an error if the conversion fails, where as TRY\_PARSE will return NULL instead of an error.   
  
Since ABC cannot be converted to INT, PARSE will return an error  
SELECT PARSE('ABC' AS INT) AS Result  
  
Since ABC cannot be converted to INT, TRY\_PARSE will return NULL instead of an error  
SELECT TRY\_PARSE('ABC' AS INT) AS Result  
  
**Example :** Using TRY\_PARSE() function with table data. We will use the following Employees table for this example.   
   
  
**SQL Script to create Employees table**

Create table Employees

(

     Id int primary key identity,

     Name nvarchar(10),

     Age nvarchar(10)

)

Go

Insert into Employees values ('Mark', '40')

Insert into Employees values ('John', '20')

Insert into Employees values ('Amy', 'THIRTY')

Insert into Employees values ('Ben', '21')

Insert into Employees values ('Sara', 'FIFTY')

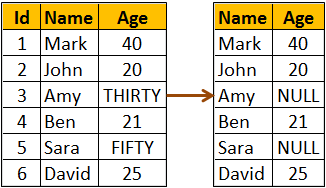
Insert into Employees values ('David', '25')

Go

The data type of Age column is nvarchar. So string values like (THIRTY, FIFTY ) are also stored. Now, we want to write a query to convert the values in Age column to int and return along with the Employee name. Notice TRY\_PARSE function returns NULL for the rows where age cannot be converted to INT.

SELECT Name, TRY\_PARSE(Age AS INT) AS Age

FROM Employees

   
  
If you use PARSE instead of TRY\_PARSE, the query fails with an error.

SELECT Name, PARSE(Age AS INT) AS Age

FROM Employees

The above query returns the following error  
Error converting string value 'THIRTY' into data type int using culture

# 124. TRY\_CONVERT function in SQL Server 2012

**Suggested Videos**  
[Part 121 - Choose function in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/choose-function-in-sql-server.html)  
[Part 122 - IIF function in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/iif-function-in-sql-server.html)   
[Part 123 - TRY\_PARSE function in SQL Server 2012](http://csharp-video-tutorials.blogspot.com/2015/10/tryparse-function-in-sql-server-2012.html)   
  
  
  
**In this video we will discuss**

* TRY\_CONVERT function
* Difference between CONVERT and TRY\_CONVERT functions
* Difference between TRY\_PARSE and TRY\_CONVERT functions

**TRY\_CONVERT function**

* Introduced in SQL Server 2012
* Converts a value to the specified data type
* Returns NULL if the provided value cannot be converted to the specified data type
* If you request a conversion that is explicitly not permitted, then TRY\_CONVERT fails with an error

**Syntax :**TRY\_CONVERT ( data\_type, value, [style] )   
  
**Style parameter is optional**. The range of acceptable values is determined by the target data\_type. For the list of all possible values for style parameter, please visit the following MSDN article  
<https://msdn.microsoft.com/en-us/library/ms187928.aspx>  
  
**Example :** Convert string to INT. As the string can be converted to INT, the result will be 99 as expected.  
  
SELECT TRY\_CONVERT(INT, '99') AS Result  
  
Output :    
try_convert function in sql server 2012   
  
**Example :** Convert string to INT. The string cannot be converted to INT, so TRY\_CONVERT returns NULL  
  
SELECT TRY\_CONVERT(INT, 'ABC') AS Result  
  
Output :    
try convert function in sql   
  
**Example :** Converting an integer to XML is not explicitly permitted. so in this case TRY\_CONVERT fails with an error  
  
SELECT TRY\_CONVERT(XML, 10) AS Result  
  
If you want to provide a meaningful error message instead of NULL when the conversion fails, you can do so using CASE statement or IIF function.  
  
**Example :** Using CASE statement to provide a meaningful error message when the conversion fails.

SELECT

CASE WHEN TRY\_CONVERT(INT, 'ABC') IS NULL

           THEN 'Conversion Failed'

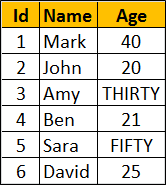
           ELSE 'Conversion Successful'

END AS Result

**Output :** As the conversion fails, you will now get a message 'Conversion Failed' instead of NULL   
sql server try_convert   
  
**Example :**Using IIF function to provide a meaningful error message when the conversion fails.

SELECT IIF(TRY\_CONVERT(INT, 'ABC') IS NULL, 'Conversion Failed',

                 'Conversion Successful') AS Result

**What is the difference between CONVERT and TRY\_CONVERT**  
CONVERT will result in an error if the conversion fails, where as TRY\_CONVERT will return NULL instead of an error.   
  
Since ABC cannot be converted to INT, CONVERT will return an error  
SELECT CONVERT(INT, 'ABC') AS Result  
  
Since ABC cannot be converted to INT, TRY\_CONVERT will return NULL instead of an error  
SELECT TRY\_CONVERT(INT, 'ABC') AS Result  
  
**Example :** Using TRY\_CONVERT() function with table data. We will use the following Employees table for this example.   
   
  
**SQL Script to create Employees table**

Create table Employees

(

     Id int primary key identity,

     Name nvarchar(10),

     Age nvarchar(10)

)

Go

Insert into Employees values ('Mark', '40')

Insert into Employees values ('John', '20')

Insert into Employees values ('Amy', 'THIRTY')

Insert into Employees values ('Ben', '21')

Insert into Employees values ('Sara', 'FIFTY')

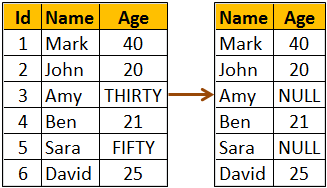
Insert into Employees values ('David', '25')

Go

The data type of Age column is nvarchar. So string values like (THIRTY, FIFTY ) are also stored. Now, we want to write a query to convert the values in Age column to int and return along with the Employee name. Notice TRY\_CONVERT function returns NULL for the rows where age cannot be converted to INT.

SELECT Name, TRY\_CONVERT(INT, Age) AS Age

FROM Employees

   
  
If you use CONVERT instead of TRY\_CONVERT, the query fails with an error.

SELECT NAME, CONVERT(INT, Age) AS Age

FROM Employees

The above query returns the following error  
Conversion failed when converting the nvarchar value 'THIRTY' to data type int.  
  
**Difference between TRY\_PARSE and TRY\_CONVERT functions**  
TRY\_PARSE can only be used for converting from string to date/time or number data types where as TRY\_CONVERT can be used for any general type conversions.  
  
**For example,** you can use TRY\_CONVERT to convert a string to XML data type, where as you can do the same using TRY\_PARSE  
  
Converting a string to XML data type using TRY\_CONVERT  
SELECT TRY\_CONVERT(XML, '<root><child/></root>') AS [XML]  
  
The above query produces the following   
try_parse vs try_convert sql server   
  
Converting a string to XML data type using TRY\_PARSE  
SELECT TRY\_PARSE('<root><child/></root>' AS XML) AS [XML]  
  
The above query will result in the following error  
Invalid data type xml in function TRY\_PARSE  
  
Another difference is TRY\_PARSE relies on the presence of .the .NET Framework Common Language Runtime (CLR) where as TRY\_CONVERT does not.

# 125. EOMONTH function in SQL Server 2012

**Suggested Videos**  
[Part 122 - IIF function in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/iif-function-in-sql-server.html)  
[Part 123 - TRY\_PARSE function in SQL Server 2012](http://csharp-video-tutorials.blogspot.com/2015/10/tryparse-function-in-sql-server-2012.html)   
[Part 124 - TRY\_CONVERT function in SQL Server 2012](http://csharp-video-tutorials.blogspot.com/2015/10/tryconvert-function-in-sql-server-2012.html)   
  
  
  
In this video we will discuss **EOMONTH function in SQL Server 2012**   
  
  
  
**EOMONTH function**

* Introduced in SQL Server 2012
* Returns the last day of the month of the specified date

**Syntax :** EOMONTH ( start\_date [, month\_to\_add ] )

**start\_date :**The date for which to return the last day of the month  
**month\_to\_add :** Optional. Number of months to add to the start\_date. EOMONTH adds the specified number of months to start\_date, and then returns the last day of the month for the resulting date.   
  
**Example :** Returns last day of the month November

SELECT EOMONTH('11/20/2015') AS LastDay

**Output :**   
sql eomonth example   
  
**Example :** Returns last day of the month of February from a NON-LEAP year

SELECT EOMONTH('2/20/2015') AS LastDay

**Output :**  
eomonth function in sql server 2012  
  
**Example :** Returns last day of the month of February from a LEAP year

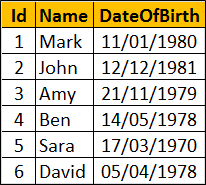
SELECT EOMONTH('2/20/2016') AS LastDay

Output :  
sql server eomonth function   
  
**month\_to\_add** optional parameter can be used to add or subtract a specified number of months from the start\_date, and then return the last day of the month from the resulting date.  
  
The following example adds 2 months to the start\_date and returns the last day of the month from the resulting date

SELECT EOMONTH('3/20/2016', 2) AS LastDay

**Output :**  
ms sql server eomonth   
  
The following example subtracts 1 month from the start\_date and returns the last day of the month from the resulting date

SELECT EOMONTH('3/20/2016', -1) AS LastDay

**Output :**   
sql server 2012 eomonth   
  
Using **EOMONTH** function with table data. We will use the following **Employees**table for this example.  
   
  
**SQL Script to create Employees table**

Create table Employees

(

    Id int primary key identity,

    Name nvarchar(10),

    DateOfBirth date

)

Go

Insert into Employees values ('Mark', '01/11/1980')

Insert into Employees values ('John', '12/12/1981')

Insert into Employees values ('Amy', '11/21/1979')

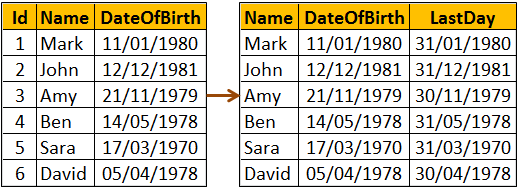
Insert into Employees values ('Ben', '05/14/1978')

Insert into Employees values ('Sara', '03/17/1970')

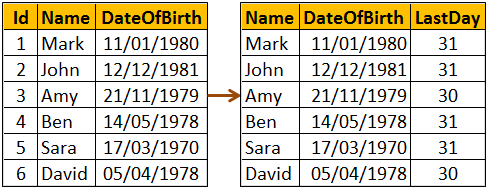
Insert into Employees values ('David', '04/05/1978')

Go   
  
The following example returns the last day of the month from the DateOfBirth of every employee.

SELECT Name, DateOfBirth, EOMONTH(DateOfBirth) AS LastDay

FROM Employees   
  
   
  
If you want just the last day instead of the full date, you can use **DATEPART** function

SELECT Name, DateOfBirth, DATEPART(DD,EOMONTH(DateOfBirth)) AS LastDay

FROM Employees   
  


# 126. DATEFROMPARTS function in SQL Server

**Suggested Videos**  
[Part 123 - TRY\_PARSE function in SQL Server 2012](http://csharp-video-tutorials.blogspot.com/2015/10/tryparse-function-in-sql-server-2012.html)  
[Part 124 - TRY\_CONVERT function in SQL Server 2012](http://csharp-video-tutorials.blogspot.com/2015/10/tryconvert-function-in-sql-server-2012.html)   
[Part 125 - EOMONTH function in SQL Server 2012](http://csharp-video-tutorials.blogspot.com/2015/10/eomonth-function-in-sql-server-2012.html)   
  
  
  
In this video we will discuss **DATEFROMPARTS function in SQL Server**  
  
  
  
**DATEFROMPARTS function**

* Introduced in SQL Server 2012
* Returns a date value for the specified year, month, and day
* The data type of all the 3 parameters (year, month, and day) is integer
* If invalid argument values are specified, the function returns an error
* If any of the arguments are NULL, the function returns null

**Syntax :** DATEFROMPARTS ( year, month, day )

**Example :**All the function arguments have valid values, so DATEFROMPARTS returns the expected date

SELECT DATEFROMPARTS ( 2015, 10, 25) AS [Date]

**Output :**  
datefromparts function in sql server   
  
**Example :** Invalid value specified for month parameter, so the function returns an error

SELECT DATEFROMPARTS ( 2015, 15, 25) AS [Date]

**Output :** Cannot construct data type date, some of the arguments have values which are not valid.  
  
**Example :** NULL specified for month parameter, so the function returns NULL.

SELECT DATEFROMPARTS ( 2015, NULL, 25) AS [Date]

**Output :**  
datefromparts in sql server  
  
**Other new date and time functions introduced in SQL Server 2012**

* EOMONTH (Discussed in [Part 125](http://csharp-video-tutorials.blogspot.com/2015/10/eomonth-function-in-sql-server-2012.html) of [SQL Server tutorial](https://www.youtube.com/watch?v=ZNObiptSMSI&list=PL08903FB7ACA1C2FB))
* **DATETIMEFROMPARTS :** Returns DateTime
* **Syntax :**DATETIMEFROMPARTS ( year, month, day, hour, minute, seconds,milliseconds )
* **SMALLDATETIMEFROMPARTS :** Returns SmallDateTime
* **Syntax :**SMALLDATETIMEFROMPARTS ( year, month, day, hour, minute )
* **We will discuss the following functions in a later video**
  + TIMEFROMPARTS
  + DATETIME2FROMPARTS
  + DATETIMEOFFSETFROMPARTS

In our next video we will discuss the **difference between DateTime and SmallDateTime**.

# 127. Difference between DateTime and SmallDateTime in SQL Server

**Suggested Videos**  
[Part 124 - TRY\_CONVERT function in SQL Server 2012](http://csharp-video-tutorials.blogspot.com/2015/10/tryconvert-function-in-sql-server-2012.html)  
[Part 125 - EOMONTH function in SQL Server 2012](http://csharp-video-tutorials.blogspot.com/2015/10/eomonth-function-in-sql-server-2012.html)   
[Part 126 - DATEFROMPARTS function in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/datefromparts-function-in-sql-server.html)   
  
  
  
In this video we will discuss the **difference between DateTime and SmallDateTime in SQL Server**  
  
  
  
The following **table summarizes the differences**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **SmallDateTime** | **DateTime** |
| Date Range | January 1, 1900, through June 6, 2079 | January 1, 1753, through December 31, 9999 |
| Time Range | 00:00:00 through 23:59:59 | 00:00:00 through 23:59:59.997 |
| Accuracy | 1 Minute | 3.33 Milli-seconds |
| Size | 4 Bytes | 8 Bytes |
| Default value | 1900-01-01 00:00:00 | 1900-01-01 00:00:00 |

The range for SmallDateTime is **January 1, 1900**, through **June 6, 2079**. A value outside of this range, is not allowed.   
  
**The following 2 queries have values outside of the range of SmallDateTime data type.**

Insert into Employees ([SmallDateTime]) values ('01/01/1899')

Insert into Employees ([SmallDateTime]) values ('07/06/2079')

**When executed, the above queries fail with the following error**  
The conversion of a varchar data type to a smalldatetime data type resulted in an out-of-range value   
  
The range for DateTime is **January 1, 1753**, through **December 31, 9999**. A value outside of this range, is not allowed.  
  
The following query has a value outside of the range of DateTime data type.  
Insert into Employees ([DateTime]) values ('01/01/1752')  
  
**When executed, the above query fails with the following error**  
The conversion of a varchar data type to a datetime data type resulted in an out-of-range value.

# 128. DateTime2FromParts function in SQL Server 2012

**Suggested Videos**  
[Part 125 - EOMONTH function in SQL Server 2012](http://csharp-video-tutorials.blogspot.com/2015/10/eomonth-function-in-sql-server-2012.html)  
[Part 126 - DATEFROMPARTS function in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/datefromparts-function-in-sql-server.html)   
[Part 127 - Difference between DateTime and SmallDateTime in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/difference-between-datetime-and.html)   
  
  
  
In this video we will discuss **DateTime2FromParts function in SQL Server 2012**.   
  
  
  
**DateTime2FromParts function**

* Introduced in SQL Server 2012
* Returns DateTime2
* The data type of all the parameters is integer
* If invalid argument values are specified, the function returns an error
* If any of the required arguments are NULL, the function returns null
* If the precision argument is null, the function returns an error

**Syntax :** DATETIME2FROMPARTS ( year, month, day, hour, minute, seconds, fractions, precision )   
  
**Example :** All the function arguments have valid values, so DATETIME2FROMPARTS returns DATETIME2 value as expected.   
  
SELECT DATETIME2FROMPARTS ( 2015, 11, 15, 20, 55, 55, 0, 0 ) AS [DateTime2]   
  
**Output :**   
datetime2fromparts function in sql server 2012   
  
**Example :** Invalid value specified for month parameter, so the function returns an error   
  
SELECT DATETIME2FROMPARTS ( 2015, 15, 15, 20, 55, 55, 0, 0 ) AS [DateTime2]   
  
**Output :** Cannot construct data type datetime2, some of the arguments have values which are not valid.   
  
**Example :** If any of the required arguments are NULL, the function returns null. NULL specified for month parameter, so the function returns NULL.   
  
SELECT DATETIME2FROMPARTS ( 2015, NULL, 15, 20, 55, 55, 0, 0 ) AS [DateTime2]   
  
Output :    
datetimefromparts sql 2012   
  
**Example :** If the precision argument is null, the function returns an error  
  
SELECT DATETIME2FROMPARTS ( 2015, 15, 15, 20, 55, 55, 0, NULL ) AS[DateTime2]   
  
**Output :** Scale argument is not valid. Valid expressions for data type datetime2 scale argument are integer constants and integer constant expressions.   
  
**TIMEFROMPARTS :** Returns time value  
  
**Syntax :**TIMEFROMPARTS ( hour, minute, seconds, fractions, precision )   
  
**Next video :** We will discuss the **difference between DateTime and DateTime2 in SQL Server**

# 129. Difference between DateTime and DateTime2 in SQL Server

**Suggested Videos**  
[Part 126 - DATEFROMPARTS function in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/datefromparts-function-in-sql-server.html)  
[Part 127 - Difference between DateTime and SmallDateTime in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/difference-between-datetime-and.html)   
[Part 128 - DateTime2FromParts function in SQL Server 2012](http://csharp-video-tutorials.blogspot.com/2015/10/datetime2fromparts-function-in-sql.html)   
  
  
  
In this video we will discuss the **difference between DateTime and DateTime2 in SQL Server**  
  
  
  
**Differences between DateTime and DateTime2**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **DateTime** | **DateTime2** |
| **Date Range** | January 1, 1753, through December 31, 9999 | January 1, 0001, through December 31, 9999 |
| **Time Range** | 00:00:00 through 23:59:59.997 | 00:00:00 through 23:59:59.9999999 |
| **Accuracy** | 3.33 Milli-seconds | 100 nanoseconds |
| **Size** | 8 Bytes | 6 to 8 Bytes (Depends on the precision) |
| **Default Value** | 1900-01-01 00:00:00 | 1900-01-01 00:00:00 |

DATETIME2 has a bigger date range than DATETIME. Also, DATETIME2 is more accurate than DATETIME. So I would recommend using DATETIME2 over DATETIME when possible. I think the only reason for using DATETIME over DATETIME2 is for backward compatibility.   
  
**DateTime2 Syntax :** DATETIME2 [ (fractional seconds precision) ]   
  
**With DateTime2**

* Optional fractional seconds precision can be specified
* The precision scale is from 0 to 7 digits
* The default precision is 7 digits
* For precision 1 and 2, storage size is 6 bytes
* For precision 3 and 4, storage size is 7 bytes
* For precision 5, 6 and 7, storage size is 8 bytes

The following script creates a table variable with 7 DATETIME2 columns with different precision start from 1 through 7

DECLARE @TempTable TABLE

(

    DateTime2Precision1 DATETIME2(1),

    DateTime2Precision2 DATETIME2(2),

    DateTime2Precision3 DATETIME2(3),

    DateTime2Precision4 DATETIME2(4),

    DateTime2Precision5 DATETIME2(5),

    DateTime2Precision6 DATETIME2(6),

    DateTime2Precision7 DATETIME2(7)

)

Insert DateTime value into each column

INSERT INTO @TempTable VALUES

(

    '2015-10-20 15:09:12.1234567',

    '2015-10-20 15:09:12.1234567',

    '2015-10-20 15:09:12.1234567',

    '2015-10-20 15:09:12.1234567',

    '2015-10-20 15:09:12.1234567',

    '2015-10-20 15:09:12.1234567',

    '2015-10-20 15:09:12.1234567'

)

The following query retrieves the prcision, the datetime value, and the storage size.

SELECT 'Precision - 1' AS [Precision],

         DateTime2Precision1 AS DateValue,

         DATALENGTH(DateTime2Precision1) AS StorageSize

FROM @TempTable

UNION ALL

SELECT 'Precision - 2',

         DateTime2Precision2,

         DATALENGTH(DateTime2Precision2) AS StorageSize

FROM @TempTable

UNION ALL

SELECT 'Precision - 3',

         DateTime2Precision3,

         DATALENGTH(DateTime2Precision3)

FROM @TempTable

UNION ALL

SELECT 'Precision - 4',

         DateTime2Precision4,

         DATALENGTH(DateTime2Precision4)

FROM @TempTable

UNION ALL

SELECT 'Precision - 5',

         DateTime2Precision5,

         DATALENGTH(DateTime2Precision5)

FROM @TempTable

UNION ALL

SELECT 'Precision - 6',

         DateTime2Precision6,

         DATALENGTH(DateTime2Precision6)

FROM @TempTable

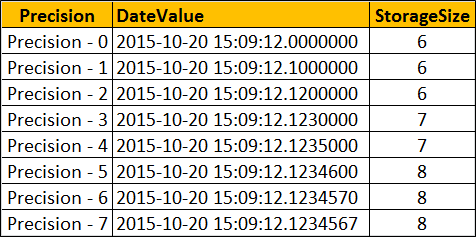
UNION ALL

SELECT 'Precision - 7',

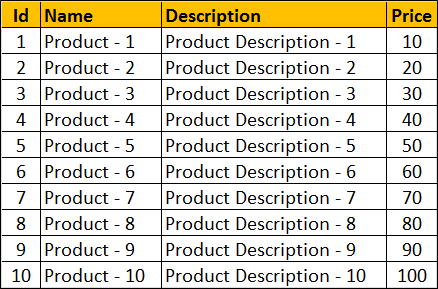
         DateTime2Precision7,

         DATALENGTH(DateTime2Precision7) AS StorageSize

FROM @TempTable

Notice as the precision increases the storage size also increases   


# 130. Offset fetch next in SQL Server 2012

**Suggested Videos**  
[Part 127 - Difference between DateTime and SmallDateTime in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/difference-between-datetime-and.html)  
[Part 128 - DateTime2FromParts function in SQL Server 2012](http://csharp-video-tutorials.blogspot.com/2015/10/datetime2fromparts-function-in-sql.html)   
[Part 129 - Difference between DateTime and DateTime2 in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/difference-between-datetime-and_21.html)   
  
  
  
In this video we will discuss **OFFSET FETCH Clause in SQL Server 2012**  
  
  
  
One of the common tasks for a SQL developer is to come up with a stored procedure that can return a page of results from the result set. With SQL Server 2012 OFFSET FETCH Clause it is very easy to implement paging.    
  
Let's understand this with an example. We will use the following **tblProducts**table for the examples in this video. The table has got 100 rows. In the image I have shown just 10 rows.   
  
   
  
**SQL Script to create tblProducts table**

Create table tblProducts

(

    Id int primary key identity,

    Name nvarchar(25),

    [Description] nvarchar(50),

    Price int

)

Go

**SQL Script to populate tblProducts table with 100 rows**

Declare @Start int

Set @Start = 1

Declare @Name varchar(25)

Declare @Description varchar(50)

While(@Start <= 100)

Begin

    Set @Name = 'Product - ' + LTRIM(@Start)

    Set @Description = 'Product Description - ' + LTRIM(@Start)

    Insert into tblProducts values (@Name, @Description, @Start \* 10)

    Set @Start = @Start + 1

End

**OFFSET FETCH Clause**

* Introduced in SQL Server 2012
* Returns a page of results from the result set
* ORDER BY clause is required

**OFFSET FETCH Syntax :**

SELECT \* FROM Table\_Name

ORDER BY Column\_List

OFFSET Rows\_To\_Skip ROWS

FETCH NEXT Rows\_To\_Fetch ROWS ONLY

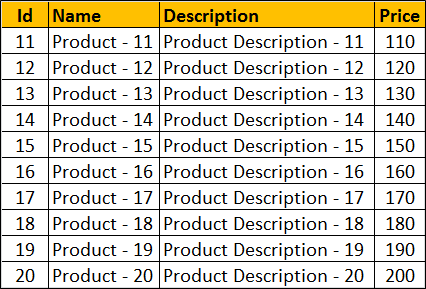
**The following SQL query**  
1. Sorts the table data by Id column  
2. Skips the first 10 rows and  
3. Fetches the next 10 rows

SELECT \* FROM tblProducts

ORDER BY Id

OFFSET 10 ROWS

FETCH NEXT 10 ROWS ONLY

**Result :**   
   
  
From the front-end application, we would typically send the **PAGE NUMBER** and the**PAGE SIZE**to get a page of rows. The following stored procedure accepts PAGE NUMBER and the PAGE SIZE as parameters and returns the correct set of rows.

CREATE PROCEDURE spGetRowsByPageNumberAndSize

@PageNumber INT,

@PageSize INT

AS

BEGIN

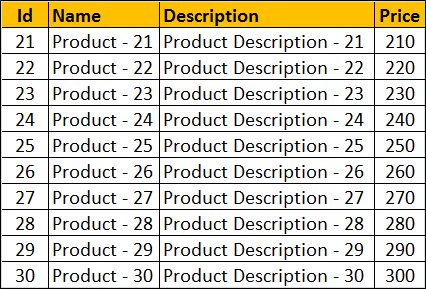
    SELECT \* FROM tblProducts

    ORDER BY Id

    OFFSET (@PageNumber - 1) \* @PageSize ROWS

    FETCH NEXT @PageSize ROWS ONLY

END

With PageNumber = 3 and PageSize = 10, the stored procedure returns the correct set of rows   
  
EXECUTE spGetRowsByPageNumberAndSize 3, 10   
  


# 131. Identifying object dependencies in SQL Server

**Suggested Videos**  
[Part 128 - DateTime2FromParts function in SQL Server 2012](http://csharp-video-tutorials.blogspot.com/2015/10/datetime2fromparts-function-in-sql.html)  
[Part 129 - Difference between DateTime and DateTime2 in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/difference-between-datetime-and_21.html)   
[Part 130 - Offset fetch next in SQL Server 2012](http://csharp-video-tutorials.blogspot.com/2015/10/offset-fetch-next-in-sql-server-2012.html)   
  
  
  
In this video we will discuss **how to identify object dependencies in SQL Server using SQL Server Management Studio**.   
  
  
  
The following SQL Script creates 2 tables, 2 stored procedures and a view

Create table Departments

(

    Id int primary key identity,

    Name nvarchar(50)

)

Go

Create table Employees

(

    Id int primary key identity,

    Name nvarchar(50),

    Gender nvarchar(10),

    DeptId int foreign key references Departments(Id)

)

Go

Create procedure sp\_GetEmployees

as

Begin

    Select \* from Employees

End

Go

Create procedure sp\_GetEmployeesandDepartments

as

Begin

    Select Employees.Name as EmployeeName,

                   Departments.Name as DepartmentName

    from Employees

    join Departments

    on Employees.DeptId = Departments.Id

End

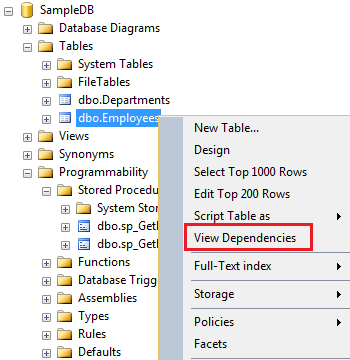
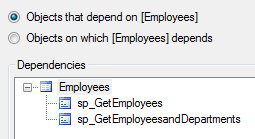
Go

Create view VwDepartments

as

Select \* from Departments

Go

**How to find dependencies using SQL Server Management Studio**  
Use View Dependencies option in SQL Server Management studio to find the object dependencies   
  
**For example :**To find the dependencies on the Employees table, right click on it and select View Dependencies from the context menu   
  
   
  
In the **Object Dependencies** window, depending on the radio button you select, you can find the objects that depend on **Employees**table and the objects on which **Employees**table depends on.   
  
   
  
**Identifying object dependencies** is important especially when you intend to modify or delete an object upon which other objects depend. Otherwise you may risk breaking the functionality.   
  
**For example**, there are 2 stored procedures (sp\_GetEmployees and sp\_GetEmployeesandDepartments) that depend on the Employees table. If we are not aware of these dependencies and if we delete the Employees table, both stored procedures will fail with the following error.  
  
Msg 208, Level 16, State 1, Procedure sp\_GetEmployees, Line 4  
Invalid object name 'Employees'.  
  
There are other ways for finding object dependencies in SQL Server which we will discuss in our upcoming videos.

# 132. sys.dm\_sql\_referencing\_entities in SQL Server

**Suggested Videos**  
[Part 129 - Difference between DateTime and DateTime2 in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/difference-between-datetime-and_21.html)  
[Part 130 - Offset fetch next in SQL Server 2012](http://csharp-video-tutorials.blogspot.com/2015/10/offset-fetch-next-in-sql-server-2012.html)   
[Part 131 - Identifying object dependencies in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/identifying-object-dependencies-in-sql.html)   
  
  
  
In this video we will discuss

* How to find object dependencies using the following dynamic management functions
  + sys.dm\_sql\_referencing\_entities
  + sys.dm\_sql\_referenced\_entities
* Difference between
  + Referencing entity and Referenced entity
  + Schema-bound dependency and Non-schema-bound dependency

This is continuation to [Part 131](http://csharp-video-tutorials.blogspot.com/2015/10/identifying-object-dependencies-in-sql.html), in which we discussed how to find object dependencies using SQL Server Management Studio. Please watch [Part 131](http://csharp-video-tutorials.blogspot.com/2015/10/identifying-object-dependencies-in-sql.html) from [SQL Server tutorial](https://www.youtube.com/playlist?list=PL08903FB7ACA1C2FB)before proceeding.   
  
The following example returns all the objects that depend on Employees table.   
Select \* from sys.dm\_sql\_referencing\_entities('dbo.Employees','Object')  
  
**Difference between referencing entity and referenced entity**  
A dependency is created between two objects when one object appears by name inside a SQL statement stored in another object. The object which is appearing inside the SQL expression is known as referenced entity and the object which has the SQL expression is known as a referencing entity.  
  
To get the REFERENCING ENTITIES use SYS.DM\_SQL\_REFERENCING\_ENTITIES dynamic management function  
  
To get the REFERENCED ENTITIES use SYS.DM\_SQL\_REFERENCED\_ENTITIES dynamic management function  
  
Now, let us say we have a stored procedure and we want to find the all objects that this stored procedure depends on. This can be very achieved using another dynamic management function, sys.dm\_sql\_referenced\_entities.    
  
The following query returns all the referenced entities of the stored procedure sp\_GetEmployeesandDepartments

Select \* from

sys.dm\_sql\_referenced\_entities('dbo.sp\_GetEmployeesandDepartments','Object')

**Please note :** For both these dynamic management functions to work we need to specify the schema name as well. Without the schema name you may not get any results.  
  
**Difference between Schema-bound dependency and Non-schema-bound dependency**  
**Schema-bound dependency :**Schema-bound dependency prevents referenced objects from being dropped or modified as long as the referencing object exists  
  
**Example :**A view created with SCHEMABINDING, or a table created with foreign key constraint.  
  
**Non-schema-bound dependency :**A non-schema-bound dependency doesn't prevent the referenced object from being dropped or modified.

# 133. sp\_depends in SQL Server

**Suggested Videos**  
[Part 130 - Offset fetch next in SQL Server 2012](http://csharp-video-tutorials.blogspot.com/2015/10/offset-fetch-next-in-sql-server-2012.html)  
[Part 131 - Identifying object dependencies in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/identifying-object-dependencies-in-sql.html)   
[Part 132 - sys.dm\_sql\_referencing\_entities in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/sysdmsqlreferencingentities-in-sql.html)  
  
  
  
In this video we will discuss sp\_depends system stored procedure.   
  
  
  
**There are several ways to find object dependencies in SQL Server**

**1.** View Dependencies feature in SQL Server Management Studio - [Part 131](http://csharp-video-tutorials.blogspot.com/2015/10/identifying-object-dependencies-in-sql.html)  
**2.** SQL Server dynamic management functions - [Part 132](http://csharp-video-tutorials.blogspot.com/2015/10/sysdmsqlreferencingentities-in-sql.html)  
     sys.dm\_sql\_referencing\_entities  
     sys.dm\_sql\_referenced\_entities  
**3.** sp\_depends system stored procedure - This video

**sp\_depends**  
A system stored procedure that returns object dependencies  
For example,

* If you specify a table name as the argument, then the views and procedures that depend on the specified table are displayed
* If you specify a view or a procedure name as the argument, then the tables and views on which the specified view or procedure depends are displayed.

**Syntax :**Execute sp\_depends 'ObjectName'

**The following SQL Script creates a table and a stored procedure**

Create table Employees

(

    Id int primary key identity,

    Name nvarchar(50),

    Gender nvarchar(10)

)

Go

Create procedure sp\_GetEmployees

as

Begin

    Select \* from Employees

End

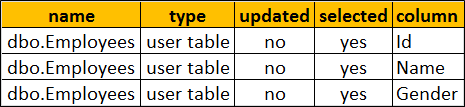
Go

Returns the stored procedure that depends on table Employees

sp\_depends 'Employees'

**Ouptut :**  
sp_depends in sql server  
  
Returns the name of the table and the respective column names on which the stored procedure sp\_GetEmployees depends

sp\_depends 'sp\_GetEmployees'

**Output :**  
  
    
Sometime sp\_depends does not report dependencies correctly. For example, at the moment we have Employees table and a stored procedure sp\_GetEmployees.   
  
**Now drop the table Employees**

Drop table Employees

**and then recreate the table again**

Create table Employees

(

    Id int primary key identity,

    Name nvarchar(50),

    Gender nvarchar(10)

)

Go  
  
Now execute the following, to find the objects that depend on Employees table

sp\_depends 'Employees'

We know that stored procedure **sp\_GetEmployees** still depends on **Employees** table. But sp\_depends does not report this dependency, as the Employees table is dropped and recreated.  
**Object does not reference any object, and no objects reference it.**  
  
sp\_depends is on the deprecation path. This might be removed from the future versions of SQL server.

# 134. Sequence object in SQL Server 2012

**Suggested Videos**  
[Part 131 - Identifying object dependencies in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/identifying-object-dependencies-in-sql.html)  
[Part 132 - sys.dm\_sql\_referencing\_entities in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/sysdmsqlreferencingentities-in-sql.html)   
[Part 133 - sp\_depends in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/spdepends-in-sql-server.html)   
  
  
  
In this video we will discuss **sequence object in SQL Server**.   
  
  
  
**Sequence object**

* Introduced in SQL Server 2012
* Generates sequence of numeric values in an ascending or descending order

**Syntax :**  
CREATE SEQUENCE [schema\_name . ] sequence\_name  
    [ AS [ built\_in\_integer\_type | user-defined\_integer\_type ] ]  
    [ START WITH <constant> ]  
    [ INCREMENT BY <constant> ]  
    [ { MINVALUE [ <constant> ] } | { NO MINVALUE } ]  
    [ { MAXVALUE [ <constant> ] } | { NO MAXVALUE } ]  
    [ CYCLE | { NO CYCLE } ]  
    [ { CACHE [ <constant> ] } | { NO CACHE } ]  
    [ ; ]

|  |  |
| --- | --- |
| **Property** | **Description** |
| DataType | Built-in integer type (tinyint , smallint, int, bigint, decimal etc...) or user-defined integer type. Default bigint. |
| START WITH | The first value returned by the sequence object |
| INCREMENT BY | The value to increment or decrement by. The value will be decremented if a negative value is specified. |
| MINVALUE | Minimum value for the sequence object |
| MAXVALUE | Maximum value for the sequence object |
| CYCLE | Specifies whether the sequence object should restart when the max value (for incrementing sequence object) or min value (for decrementing sequence object) is reached. Default is NO CYCLE, which throws an error when minimum or maximum value is exceeded. |
| CACHE | Cache sequence values for performance. Default value is CACHE. |

**Creating an Incrementing Sequence :** The following code create a Sequence object that starts with 1 and increments by 1   
  
CREATE SEQUENCE [dbo].[SequenceObject]   
AS INT  
START WITH 1  
INCREMENT BY 1  
  
**Generating the Next Sequence Value :** Now we have a sequence object created. To generate the sequence value use NEXT VALUE FOR clause  
  
SELECT NEXT VALUE FOR [dbo].[SequenceObject]  
  
**Output :** 1  
  
Every time you execute the above query the sequence value will be incremented by 1. I executed the above query 5 times, so the current sequence value is 5.  
  
**Retrieving the current sequence value :** If you want to see what the current Sequence value before generating the next, use **sys.sequences**

SELECT \* FROM sys.sequences WHERE name = 'SequenceObject'

**Alter the Sequence object to reset the sequence value :**  
ALTER SEQUENCE [SequenceObject] RESTART WITH 1  
  
**Select the next sequence value to make sure the value starts from 1**

SELECT NEXT VALUE FOR [dbo].[SequenceObject]

**Using sequence value in an INSERT query :**

CREATE TABLE Employees

(

    Id INT PRIMARY KEY,

    Name NVARCHAR(50),

    Gender NVARCHAR(10)

)

-- Generate and insert Sequence values

INSERT INTO Employees VALUES

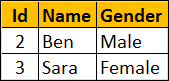
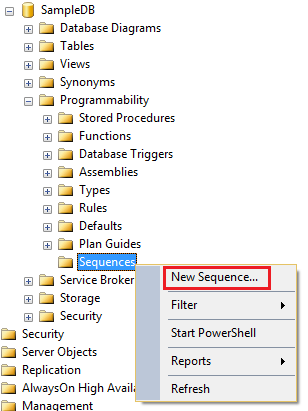
(NEXT VALUE for [dbo].[SequenceObject], 'Ben', 'Male')

INSERT INTO Employees VALUES

(NEXT VALUE for [dbo].[SequenceObject], 'Sara', 'Female')

-- Select the data from the table

SELECT \* FROM Employees

   
  
**Creating the decrementing Sequence :** The following code create a Sequence object that starts with 100 and decrements by 1  
  
CREATE SEQUENCE [dbo].[SequenceObject]   
AS INT  
START WITH 100  
INCREMENT BY -1  
  
**Specifying MIN and MAX values for the sequence :** Use the MINVALUE and MAXVALUE arguments to specify the MIN and MAX values respectively.  
  
**Step 1 :** Create the Sequence object  
CREATE SEQUENCE [dbo].[SequenceObject]  
 START WITH 100  
 INCREMENT BY 10  
 MINVALUE 100  
 MAXVALUE 150  
  
**Step 2 :**Retrieve the next sequence value. The sequence value starts at 100. Every time we call NEXT VALUE, the value will be incremented by 10.   
  
SELECT NEXT VALUE FOR [dbo].[SequenceObject]   
  
If you call NEXT VALUE, when the value reaches 150 (MAXVALUE), you will get the following error  
The sequence object 'SequenceObject' has reached its minimum or maximum value. Restart the sequence object to allow new values to be generated.  
  
**Recycling Sequence values :** When the sequence object has reached it's maximum value, and if you want to restart from the minimum value, set CYCLE option  
  
ALTER SEQUENCE [dbo].[SequenceObject]  
 INCREMENT BY 10  
 MINVALUE 100  
 MAXVALUE 150  
 CYCLE  
  
At this point, whe the sequence object has reached it's maximum value, and if you ask for the NEXT VALUE, sequence object starts from the minimum value again which in this case is 100.  
  
**To improve performance**, the Sequence object values can be cached using the CACHE option. When the values are cached they are read from the memory instead of from the disk, which improves the performance. When the cache option is specified you can also specify the size of th cache , that is the number of values to cache.  
  
The following example, creates the sequence object with 10 values cached. When the 11th value is requested, the next 10 values will be cached again.  
  
CREATE SEQUENCE [dbo].[SequenceObject]  
 START WITH 1  
 INCREMENT BY 1  
 CACHE 10  
  
**Using SQL Server Graphical User Interface (GUI) to create the sequence object :**  
1. Expand the database folder  
2. Expand Programmability folder  
3. Right click on Sequences folder  
4. Select New Sequence   
  
   
  
**Next video :** Difference between SEQUENCE and IDENTITY in SQL Server

# 135. Difference between sequence and identity in SQL Server

**Suggested Videos**  
[Part 132 - sys.dm\_sql\_referencing\_entities in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/sysdmsqlreferencingentities-in-sql.html)  
[Part 133 - sp\_depends in SQL Server](http://csharp-video-tutorials.blogspot.com/2015/10/spdepends-in-sql-server.html)   
[Part 134 - Sequence object in SQL Server 2012](http://csharp-video-tutorials.blogspot.com/2015/10/sequence-object-in-sql-server-2012.html)   
  
  
  
In this video we will discuss the **difference between SEQUENCE and IDENTITY in SQL Server**  
  
  
  
This is continuation to [Part 134](http://csharp-video-tutorials.blogspot.com/2015/10/sequence-object-in-sql-server-2012.html). Please watch [Part 134](http://csharp-video-tutorials.blogspot.com/2015/10/sequence-object-in-sql-server-2012.html) from [SQL Server tutorial](https://www.youtube.com/playlist?list=PL08903FB7ACA1C2FB) before proceeding.  
  
**Sequence object** is similar to the Identity property, in the sense that it generates sequence of numeric values in an ascending order just like the identity property. However there are several differences between the 2 which we will discuss in this video.  
  
Identity property is a table column property meaning it is tied to the table, where as the sequence is a user-defined database object and is not tied to any specific table meaning it's value can be shared by multiple tables.  
  
**Example :** Identity property tied to the Id column of the Employees table.

CREATE TABLE Employees

(

    Id INT PRIMARY KEY IDENTITY(1,1),

    Name NVARCHAR(50),

    Gender NVARCHAR(10)

)

**Example :** Sequence object not tied to any specific table

CREATE SEQUENCE [dbo].[SequenceObject]

AS INT

START WITH 1

INCREMENT BY 1

This means the above sequence object can be used with any table.  
  
**Example :** Sharing sequence object value with multiple tables.  
  
**Step 1 :** Create Customers and Users tables

CREATE TABLE Customers

(

    Id INT PRIMARY KEY,

    Name NVARCHAR(50),

    Gender NVARCHAR(10)

)

GO

CREATE TABLE Users

(

    Id INT PRIMARY KEY,

    Name NVARCHAR(50),

    Gender NVARCHAR(10)

)

GO

**Step 2 :** Insert 2 rows into Customers table and 3 rows into Users table. Notice the same sequence object is generating the ID values for both the tables.

INSERT INTO Customers VALUES

   (NEXT VALUE for [dbo].[SequenceObject], 'Ben', 'Male')

INSERT INTO Customers VALUES

   (NEXT VALUE for [dbo].[SequenceObject], 'Sara', 'Female')

INSERT INTO Users VALUES

   (NEXT VALUE for [dbo].[SequenceObject], 'Tom', 'Male')

INSERT INTO Users VALUES

   (NEXT VALUE for [dbo].[SequenceObject], 'Pam', 'Female')

INSERT INTO Users VALUES

   (NEXT VALUE for [dbo].[SequenceObject], 'David', 'Male')

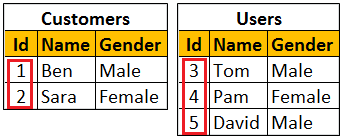
GO

**Step 3 :** Query the tables

SELECT \* FROM Customers

SELECT \* FROM Users

GO

**Output :** Notice the same sequence object has generated the values for ID columns in both the tables   
  
   
  
To generate the next identity value, a row has to be inserted into the table, where as with sequence object there is no need to insert a row into the table to generate the next sequence value. You can use NEXT VALUE FOR clause to generate the next sequence value.  
  
**Example :** Generating Identity value by inserting a row into the table  
  
INSERT INTO Employees VALUES ('Todd', 'Male')  
  
**Example :** Generating the next sequence value using NEXT VALUE FOR clause.  
  
SELECT NEXT VALUE FOR [dbo].[SequenceObject]  
  
Maximum value for the identity property cannot be specified. The maximum value will be the maximum value of the correspoding column data type. With the sequence object you can use the MAXVALUE option to specify the maximum value. If the MAXVALUE option is not specified for the sequence object, then the maximum value will be the maximum value of it's data type.  
  
**Example :** Specifying maximum value for the sequence object using the MAXVALUE option

CREATE SEQUENCE [dbo].[SequenceObject]

START WITH 1

INCREMENT BY 1

MAXVALUE 5

CYCLE option of the Sequence object can be used to specify whether the sequence should restart automatically when the max value (for incrementing sequence object) or min value (for decrementing sequence object) is reached, where as with the Identity property we don't have any such option to automatically restart the identity values.  
  
**Example :** Specifying the CYCLE option of the Sequence object, so the sequence will restart automatically when the max value is exceeded

CREATE SEQUENCE [dbo].[SequenceObject]

START WITH 1

INCREMENT BY 1

MINVALUE 1

MAXVALUE 5

CYCLE