**INTERVIEW QUESTIONS DEFINITIONS**

**T-SQL and BI DBA**

**What is the difference between CAST and CONVERT?**

**CAST** converts some data types and takes only one argument.

SELECT CAST(‘123’, AS int)

**CONVERT** allows you to format STRING values

SELECT ‘US DATE: ‘ + CONVERT(VARCHAR(50), GETDATE(), 101)

**What does a cross join do?**

The query will return every possible combination of rows.

**What does a CROSS APPLY do?**

There are two forms of APPLY: CROSS APPLY and OUTER APPLY.

The APPLY operator allows you to invoke a table-valued function for each row returned by an outer table expression of a query.

**Problem**SQL Server 2005 introduced the APPLY operator, which is very much like a [join clause](http://www.mssqltips.com/sqlservertip/1667/sql-server-join-example/) and which allows joining between two table expressions i.e. joining a left/outer table expression with a right/inner table expression. The difference between join and APPLY operator becomes evident when you have a table-valued expression on the right side and you want this table-valued expression to be evaluated for each row from the left table expression. In this tip I am going to demonstrate what APPLY operator is, how it differs from [regular JOINs](http://www.mssqltips.com/sqlservertip/1667/sql-server-join-example/) and what are few of its applications.

**Solution**The APPLY operator allows you to join two table expressions; the right table expression is processed every time for each row from the left table expression. As you might have guessed, the left table expression is evaluated first and then right table expression is evaluated against each row of the left table expression for final result-set. The final result-set contains all the selected columns from the left table expression followed by all the columns of right table expression.

The APPLY operator comes in two variants, the CROSS APPLY and the OUTER APPLY. The CROSS APPLY operator returns only those rows from left table expression (in its final output) if it matches with right table expression. In other words, the right table expression returns rows for left table expression match only.  Whereas the OUTER APPLY operator returns all the rows from left table expression irrespective of its match with the right table expression.  For those rows for which there are no corresponding matches in right table expression, it contains NULL values in columns of right table expression. So you might now conclude, the CROSS APPLY is semantically equivalent to INNER JOIN (or to be more precise its like a CROSS JOIN with a correlated sub-query) with a implicit join condition of 1=1 whereas OUTER APPLY is semantically equivalent to LEFT OUTER JOIN.

You might be wondering if the same can be achieved with regular JOIN clause then why and when to use APPLY operator? Though the same can be achieved with [normal JOIN](http://www.mssqltips.com/sqlservertip/1667/sql-server-join-example/), the need of APPLY arises if you have table-valued expression on right part and also in some cases use of APPLY operator boost the [performance](http://www.mssqltips.com/category.asp?catid=9) of your query. Let me explain you with help of some examples.

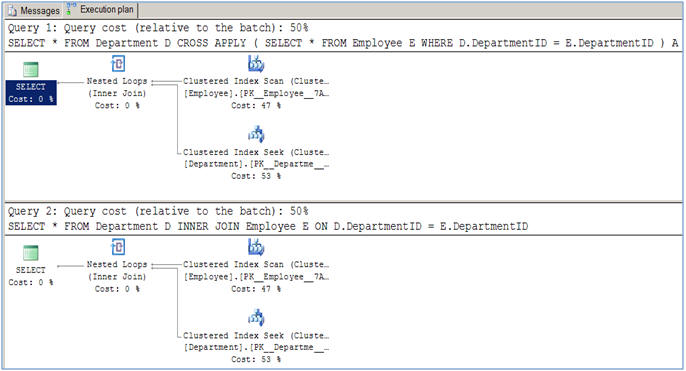
Script #1 creates a Department table to hold information about departments. Then it creates an Employee table which hold information about the employees. Please note, each employee belongs to a department, hence the Employee table has referential integrity with the Department table.

|  |
| --- |
| **Script #1 - Creating some temporary objects to work on...** |
| USE [tempdb]  GO  IF EXISTS (SELECT \* FROM sys.objects WHERE OBJECT\_ID = OBJECT\_ID(N'[Employee]') AND type IN (N'U'))  BEGIN     DROP TABLE [Employee]  END  GO  IF EXISTS (SELECT \* FROM sys.objects WHERE OBJECT\_ID = OBJECT\_ID(N'[Department]') AND type IN (N'U'))  BEGIN     DROP TABLE [Department]  END  CREATE TABLE [Department](     [DepartmentID] [int] NOT NULL PRIMARY KEY,     [Name] VARCHAR(250) NOT NULL,  ) ON [PRIMARY]  INSERT [Department] ([DepartmentID], [Name])   VALUES (1, N'Engineering')  INSERT [Department] ([DepartmentID], [Name])   VALUES (2, N'Administration')  INSERT [Department] ([DepartmentID], [Name])   VALUES (3, N'Sales')  INSERT [Department] ([DepartmentID], [Name])   VALUES (4, N'Marketing')  INSERT [Department] ([DepartmentID], [Name])   VALUES (5, N'Finance')  GO  CREATE TABLE [Employee](     [EmployeeID] [int] NOT NULL PRIMARY KEY,     [FirstName] VARCHAR(250) NOT NULL,     [LastName] VARCHAR(250) NOT NULL,     [DepartmentID] [int] NOT NULL REFERENCES [Department](DepartmentID),  ) ON [PRIMARY]  GO  INSERT [Employee] ([EmployeeID], [FirstName], [LastName], [DepartmentID])  VALUES (1, N'Orlando', N'Gee', 1 )  INSERT [Employee] ([EmployeeID], [FirstName], [LastName], [DepartmentID])  VALUES (2, N'Keith', N'Harris', 2 )  INSERT [Employee] ([EmployeeID], [FirstName], [LastName], [DepartmentID])  VALUES (3, N'Donna', N'Carreras', 3 )  INSERT [Employee] ([EmployeeID], [FirstName], [LastName], [DepartmentID])  VALUES (4, N'Janet', N'Gates', 3 ) |

First query in Script #2 selects data from Department table and uses CROSS APPLY to evaluate the Employee table for each record of the Department table. Second query simply joins the Department table with the Employee table and all the matching records are produced.

|  |
| --- |
| **Script #2 - CROSS APPLY and INNER JOIN** |
| SELECT \* FROM Department D  CROSS APPLY     (     SELECT \* FROM Employee E     WHERE E.DepartmentID = D.DepartmentID     ) A  GO  SELECT \* FROM Department D  INNER JOIN Employee E ON D.DepartmentID = E.DepartmentID  GO  http://www.mssqltips.com/tipImages2/1958_CROSS1.jpg |

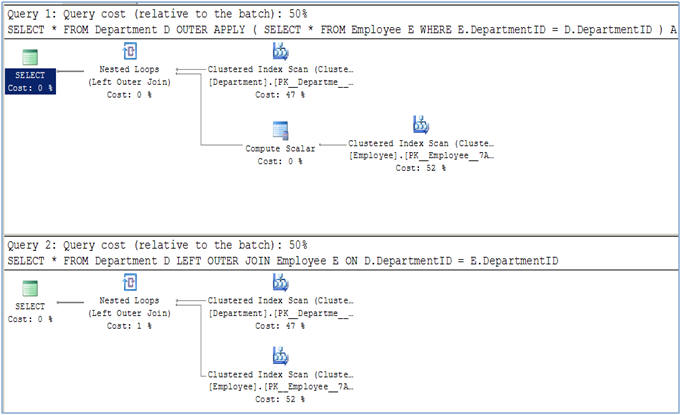
If you look at the results they produced, it is the exact same result-set; not only that even the execution plan for these queries are similar to each other and has equal query cost, as you can see in the image below. So what is the use of APPLY operator?  How does it differ from a JOIN and how does it help in writing more efficient queries. I will discuss this later, but first let me show you an example of OUTER APPLY also.



The first query in Script #3 selects data from Department table and uses OUTER APPLY to evaluate the Employee table for each record of the Department table.  For those rows for which there is not a match in Employee table, those rows contains NULL values as you can see in case of row 5 and 6. The second query simply uses a LEFT OUTER JOIN between the Department table and the Employee table.  As expected the query returns all rows from Department table; even for those rows for which there is no match in the Employee table.

|  |
| --- |
| **Script #3 - OUTER APPLY and LEFT OUTER JOIN** |
| SELECT \* FROM Department D  OUTER APPLY     (     SELECT \* FROM Employee E     WHERE E.DepartmentID = D.DepartmentID     ) A  GO  SELECT \* FROM Department D  LEFT OUTER JOIN Employee E ON D.DepartmentID = E.DepartmentID  GO  http://www.mssqltips.com/tipImages2/1958_CROSS3.jpg |

Even though the above two queries return the same information, the execution plan is a bit different. Although cost wise there is not much difference, the query with the OUTER APPLY uses a Compute Scalar operator (which has an estimated operator cost of 0.0000103 or almost 0% of total query cost) before Nested Loops operator to evaluate and produce the columns of Employee table.



Now comes the time to see where the APPLY operator is really required. In Script #4, I am creating a table-valued function which accepts DepartmentID as its parameter and returns all the employees who belong to this department. The next query selects data from Department table and uses CROSS APPLY to join with the function we created.  It passes the DepartmentID for each row from the outer table expression (in our case Department table) and evaluates the function for each row similar to a [correlated subquery](http://www.mssqltips.com/sqlservertip/1918/different-strategies-for-removing-duplicate-records-in-sql-server/). The next query uses the OUTER APPLY in place of CROSS APPLY and hence unlike CROSS APPLY which returned only correlated data, the OUTER APPLY returns non-correlated data as well, placing NULLs into the missing columns.

|  |
| --- |
| **Script #4 - APPLY with table-valued function** |
| IF EXISTS (SELECT \* FROM sys.objects WHERE OBJECT\_ID = OBJECT\_ID(N'[fn\_GetAllEmployeeOfADepartment]') AND type IN (N'IF'))  BEGIN     DROP FUNCTION dbo.fn\_GetAllEmployeeOfADepartment  END  GO  CREATE FUNCTION dbo.fn\_GetAllEmployeeOfADepartment(@DeptID AS INT)   RETURNS TABLE  AS  RETURN     (     SELECT \* FROM Employee E     WHERE E.DepartmentID = @DeptID     )  GO  SELECT \* FROM Department D  CROSS APPLY dbo.fn\_GetAllEmployeeOfADepartment(D.DepartmentID)  GO  SELECT \* FROM Department D  OUTER APPLY dbo.fn\_GetAllEmployeeOfADepartment(D.DepartmentID)  GO  http://www.mssqltips.com/tipImages2/1958_CROSS5.jpg |

So now if you are wondering, can we use a [simple join](http://www.mssqltips.com/sqlservertip/1667/sql-server-join-example/) in place of the above queries? Then the answer is NO, if you replace CROSS/OUTER APPLY in the above queries with INNER JOIN/LEFT OUTER JOIN, specify ON clause (something as 1=1) and run the query, you will get "The multi-part identifier "D.DepartmentID" could not be bound."  error. This is because with JOINs the execution context of outer query is different from the execution context of the function (or a [derived table](http://www.mssqltips.com/sqlservertip/1042/using-derived-tables-to-simplify-the-sql-server-query-process/)), and you can not bind a value/variable from the outer query to the function as a parameter.  Hence the APPLY operator is required for such queries.

So in summary the APPLY operator is required when you have to use table-valued function in the query, but it can also be used with an inline SELECT statements.

Now let me show you another query with a [Dynamic Management Function (DMF)](http://www.mssqltips.com/category.asp?catid=31). Script #5 returns all the currently executing user queries except for the queries being executed by the current session. As you can see the script below, the [sys.dm\_exec\_requests](http://www.mssqltips.com/sqlservertip/1811/how-to-isolate-the-current-running-commands-in-sql-server/) dynamic management view is being CROSS APPLY'ed with the [sys.dm\_exec\_sql\_text](http://www.mssqltips.com/sqlservertip/1363/identifying-the-input-buffer-in-sql-server-2000-vs-sql-server-2005/) dynamic management function which accepts a "plan handle" for the query and the same "plan handle" is being passed from the left/outer expression to the function to work and to return the data.

|  |
| --- |
| **Script #5 - APPLY with Dynamic Management Function (DMF)** |
| USE master  GO  SELECT DB\_NAME(database\_id) AS [Database], [text] AS [Query]   FROM sys.dm\_exec\_requests r  CROSS APPLY sys.dm\_exec\_sql\_text(r.plan\_handle) st  WHERE session\_Id > 50           -- Consider spids for users only, no system spids.  AND session\_Id NOT IN (@@SPID)  -- Don't include request from current spid. |

Please note the [text] column in the above query returns the all queries submitted in a batch; if you want to see only active (currently executing) query you can use statement\_start\_offset and statement\_end\_offset columns to trim the active part of the query. Tim Ford has provided a very good explanation of usage of these columns in his [How to isolate the current running commands in SQL Server](http://www.mssqltips.com/sqlservertip/1811/how-to-isolate-the-current-running-commands-in-sql-server/) tip.

As I told you before there are certain scenarios where a query with APPLY operator performs better than a query with regular joins but I am not going to delve into much details rather here are some articles which discuss this topic in greater details.

* [INNER JOINS vs CROSS APPLY](http://explainextended.com/2009/07/16/inner-join-vs-cross-apply/)
* [Using CROSS APPLY to optimize joins on BETWEEN conditions](http://sqlblog.com/blogs/alexander_kuznetsov/archive/2009/07/07/using-cross-apply-to-optimize-joins-on-between-conditions.aspx)

Please note, APPLY operator is not an ANSI operator but rather an extension of SQL Server T-SQL (available in SQL Server 2005 and above), so if you plan to port your database to some other DBMS take this into your considerations.

**What does SUM OVER do?**

The SUM OVER command

One of new features in SQL 2005 that I haven't seen much talk about is that you can now add aggregate functions to any SELECT (even without a GROUP BY clause) by specifying an OVER() partition for each function. Unfortunately, it isn't especially powerful, and you can't do running totals with it, but it does help you make your code a little shorter and in many cases it might be just what you need.

select customerID, productID, orderDate, orderAmount

from Orders

customerID productID orderDate orderAmount

----------- ----------- ----------------------- ---------------------

1 1 2007-01-01 00:00:00.000 20.00

1 2 2007-01-02 00:00:00.000 30.00

1 2 2007-01-05 00:00:00.000 23.00

1 3 2007-01-04 00:00:00.000 18.00

2 1 2007-01-03 00:00:00.000 74.00

2 1 2007-01-06 00:00:00.000 34.00

2 2 2007-01-08 00:00:00.000 10.00

(7 row(s) affected)

You can now easily return the total orderAmount per customer as an additional column in this SELECT, simply by adding an aggregate SUM() function with an OVER() clause:

select customerID, productID, orderDate, orderAmount,

sum(orderAmount) OVER (Partition by CustomerID) as Total

from Orders

customerID productID orderDate orderAmount Total

----------- ----------- ----------------------- ------------- ---------

1 1 2007-01-01 00:00:00.000 20.00 91.00

1 2 2007-01-02 00:00:00.000 30.00 91.00

1 2 2007-01-05 00:00:00.000 23.00 91.00

1 3 2007-01-04 00:00:00.000 18.00 91.00

2 1 2007-01-03 00:00:00.000 74.00 118.00

2 1 2007-01-06 00:00:00.000 34.00 118.00

2 2 2007-01-08 00:00:00.000 10.00 118.00

(7 row(s) affected)

The previous SQL is essentially shorthand for:

select

o.customerID, o.productID, o.orderDate, o.orderAmount, t.Total

from

Orders o

inner join

(

select customerID, sum(orderAmount) as Total from Orders group by customerID

)

t on t.customerID = o.customerID

since the two return the same results.

Note that the total returned using SUM(..) OVER (..) is not the total for the entire table, just for the scope of the SELECT where it is used. For example, if you add a filter to the SELECT to return only rows for ProductID 2, the totals will reflect that criteria as well:

select customerID, productID, orderDate, orderAmount,

sum(orderAmount) OVER (Partition by CustomerID) as Total

from Orders

where productID = 2

customerID productID orderDate orderAmount Total

----------- ----------- ----------------------- ------------ ------------

1 2 2007-01-02 00:00:00.000 30.00 53.00

1 2 2007-01-05 00:00:00.000 23.00 53.00

2 2 2007-01-08 00:00:00.000 10.00 10.00

(3 row(s) affected)

That is a nice advantage over the old way of linking to a derived table, since in that case you'd need to repeat the criteria for both the primary (outer) SELECT and also the derived table.

**Typically, SUM(..) OVER(..) is most useful for calculating a percentage of a total for each row.** For example, for each Order we can calculate the percentage of that order's orderAmount compared to the customer's total orderAmount:

select customerID, productID, orderDate, orderAmount,

orderAmount / sum(orderAmount) OVER (Partition by CustomerID) as Pct

from Orders

customerID productID orderDate orderAmount Pct

----------- ----------- ----------------------- ------------ -------

1 1 2007-01-01 00:00:00.000 20.00 0.2197

1 2 2007-01-02 00:00:00.000 30.00 0.3296

1 2 2007-01-05 00:00:00.000 23.00 0.2527

1 3 2007-01-04 00:00:00.000 18.00 0.1978

2 1 2007-01-03 00:00:00.000 74.00 0.6271

2 1 2007-01-06 00:00:00.000 34.00 0.2881

2 2 2007-01-08 00:00:00.000 10.00 0.0847

(7 row(s) affected)

Of course, be sure that you don't encounter any divide by zero errors by using a CASE if necessary.

While I've made many references to using the SUM() function, of course this technique works with any of the other aggregate functions as well, such as MIN() or AVG(). For example, you could return only Orders where the orderAmount is below the average for the product that was ordered by writing:

select x.\*

from

(

select customerId, productID, orderDate, orderAmount,

avg(orderAmount) over (partition by productID) as ProductAvg

from orders

) x

where x.orderAmount < x.productAvg

customerId productID orderDate orderAmount ProductAvg

----------- ----------- ----------------------- ------------- -----------

1 1 2007-01-01 00:00:00.000 20.00 42.6666

2 1 2007-01-06 00:00:00.000 34.00 42.6666

2 2 2007-01-08 00:00:00.000 10.00 21.00

(3 row(s) affected)

It is my understanding that some SQL implementations allow you to use SUM(..) OVER (..) to calculate running totals for a SELECT, but unfortunately that does not appear to be possible using SQL Server 2005. However, there are [other ways to accomplish this in T-SQL](http://www.sqlteam.com/item.asp?ItemID=3856) if you really need to; my general recommendation is to do this at your presentation layer if those totals are not needed for further processing at the database.

**What is the difference between a CTE and a Temporary table?**

A common table expression (CTE) can be thought of as a temporary result set that is defined within the execution scope of a single SELECT, INSERT, UPDATE, DELETE, or CREATE VIEW statement. A CTE is similar to a derived table in that it is not stored as an object and lasts only for the duration of the query. **Unlike a derived table, a CTE can be self-referencing and can be referenced multiple times in the same query.**

A CTE can be used to:

* Create a recursive query. For more information, see [Recursive Queries Using Common Table Expressions](http://msdn.microsoft.com/en-us/library/ms186243(v=sql.105).aspx).
* Substitute for a view when the general use of a view is not required; that is, you do not have to store the definition in metadata.
* Enable grouping by a column that is derived from a scalar subselect, or a function that is either not deterministic or has external access.
* Reference the resulting table multiple times in the same statement.

Using a CTE offers the advantages of improved readability and ease in maintenance of complex queries. The query can be divided into separate, simple, logical building blocks. These simple blocks can then be used to build more complex, interim CTEs until the final result set is generated.

CTEs can be defined in user-defined routines, such as functions, stored procedures, triggers, or views.

**What is the difference between a primary key and a unique constraint:**

A table can have only one PRIMARY KEY whereas there can be any number of UNIQUE keys. The columns that compose PK are automatically defined NOT NULL, whereas a column that composes a UNIQUE key is not automatically defined to be NOT NULL must also specify the column IS NOT NULL

**Primay Key                     Unique key**

There is only one         there may be more than 1  
Primary key for           Unique Key in table   
one table

It cannot contain           It Can contain one Null Value  
Null value

Primary key                  UNIQUE KEY creates a non clustered index  
creates a clustered   
index

**Primary key**:  
1)Primary key uniquely identifies each row in Table.  
2)Primary key does not allow duplicate values and Null values.  
3)Primary key is default Clustered indexes.  
4)One table can have only one Primary key.

**Unique Key: (Alternate key)**  
1)Unique Key uniquely identifies each row in Table.  
2)Unique key does not allow duplicate values but allows only one Null value.  
3)Unique key is default Non- Clustered indexes.

4)One table can have many unique keys.

A primary key, as well as a unique constraint will uniquely identify a row in a given table. Each table, can have a single primary key, and multiple unique constraints. As Raj mentioned, a Primary Key cannot have a NULL value and a Unique Constraint can have a single NULL Value (Unless using a filtered index).

Both a Primary Key and a Unique Constraint, are implemented as indexes on the table. And, while SQL Server will by default create the PK as the clustered index, it isn't a requirement that the clustered index be the PK on the table, it is possible to create another index as the clustered index.

Additionally, you can create a UNIQUE CLUSTERED INDEX on the table, that while not technically a unique constraint can be the clustered index and not the PK.

Whereas, a UNIQUE Constraint is implemented as an Index, a Unique Index is not implemented as a constraint.

Often, unique constraints are referred to as ALTERNATE KEYS.

One other item of interest, is that either a PK or a Unique Constraint can be used as a Foreign Key....

I think Raj has a complete answer, but I'll add one thing. A Primary Key is a unique constraint, but a unique constraint is not a primary key. A unique constraint is a superset of primary keys, or PKs a subset of the unique constraints.

One difference that hasn't been mentioned yet is that a PRIMARY KEY constraint will be the default target for a FOREIGN KEY constraint referencing that table if you create the FOREIGN KEY without specifying a column list. Otherwise PRIMARY KEY achieves the same thing as a UNIQUE constraint on NOT NULL columns.

It is a very widely observed convention of SQL database design that every table should have a PRIMARY KEY constraint. This is of little practical consequence however and it would be quite possible (and very possibly desirable) to do without PRIMARY KEY constraints altogether.

Once you establish a UNIQUE constraint (also known as alternate key), every value in the named columns must be unique. A unique constraint requires a unique value throughout the named column or combination of columns.

**What Kind of User-Defined Functions can I Create.**

If we have written a scalar function then it returns a single value. If we have written a table valued function then it returns multiple rows. We cannot write the function without returning any value to the calling program. The function can be called using the select query. It can be called from the select/where/having clause. The temporary table cannot be created in the function.

There are three types of User-Defined functions in SQL Server 2000 and they are **Scalar, Inline Table-Valued and Multi-statement Table-valued.**

#### Three Types of User-Defined Functions

Now that you have seen how easy it is to create and implement a simple function, let’s cover the three different types of user-defined functions and some of the nuances of how they are implemented.

#### Scalar Functions

A scalar function returns a single value of the data type referenced in the RETURNS clause of the CREATE FUNCTION statement. The returned data can be of any type except text, ntext, image, cursor, or timestamp.

The example we covered in the previous section is a scalar function. Although the previous example only contained one statement in the BEGIN…END block, a scalar function can contain an unlimited number of statements as long as only one value is returned. The following example uses a WHILE construct to demonstrate this.

**CREATE FUNCTION fx\_SumTwoValues2**

**( @Val1 int, @Val2 int )**

**RETURNS int**

**AS**

**BEGIN**

**WHILE @Val1 <100**

**BEGIN**

**SET @Val1 =@Val1 +1**

**END**

**RETURN (@Val1+@Val2)**

**END**

**go**

**SELECT dbo.fx\_SumTwoValues2(1,7) AS SumOfTwoValues**

**SumOfTwoValues**

**--------------**

**107**

The @Val1 input parameter is set to 1 when the function is called, but the WHILE increments the parameter to 100 before the RETURN statement is executed. Note that the two-part name (owner.object\_name) is used to call the function. Scalar functions require that their two-part names be used when they are called. As you will see in the next two sections, this is not the case with the other two types of functions.

#### Inline Table-Valued Functions

An inline table-valued function returns a variable of data type table whose value is derived from a single **SELECT statement**. Since the return value is derived from the SELECT statement, there is no BEGIN/END block needed in the CREATE FUNCTION statement. There is also no need to specify the table variable name (or column definitions for the table variable) because the structure of the returned value is generated from the columns that compose the SELECT statement. Because the results are a function of the columns referenced in the SELECT, no duplicate column names are allowed and all derived columns must have an associated alias.

The following uses the Customer table in the Northwind database to show how an inline table-valued function is implemented.

**USE Northwind**

**go**

**CREATE FUNCTION fx\_Customers\_ByCity**

**( @City nvarchar(15) )**

**RETURNS table**

**AS**

**RETURN (**

**SELECT CompanyName**

**FROM Customers**

**WHERE City =@City**

**)**

**go**

**SELECT \* FROM fx\_Customers\_ByCity('London')**

**CompanyName**

**----------------------------------------**

**Around the Horn**

**. . .**

**Seven Seas Imports**

#### Multi-Statement Table-Valued Functions

The multi-statement table-valued function is slightly more complicated than the other two types of functions because **it uses multiple statements to build the table that is returned to the calling statement**. Unlike the inline table-valued function, **a table variable must be explicitly declared and defined**. The following example shows how to implement a multi-statement table-valued function that populates and returns a table variable.

**USE Northwind**

**go**

**CREATE FUNCTION fx\_OrdersByDateRangeAndCount**

**( @OrderDateStart smalldatetime,**

**@OrderDateEnd smalldatetime,**

**@OrderCount smallint )**

**RETURNS @OrdersByDateRange TABLE**

**( CustomerID nchar(5),**

**CompanyName nvarchar(40),**

**OrderCount smallint,**

**Ranking char(1) )**

**AS**

**BEGIN**

**--Statement 1**

**INSERT @OrdersByDateRange**

**SELECT a.CustomerID,**

**a.CompanyName,**

**COUNT(a.CustomerID)AS OrderCount,**

**'B'**

**FROM Customers a**

**JOIN Orders b ON a.CustomerID =b.CustomerID**

**WHERE OrderDate BETWEEN @OrderDateStart AND @OrderDateEnd**

**GROUP BY a.CustomerID,a.CompanyName**

**HAVING COUNT(a.CustomerID)>@OrderCount**

**--Statement 2**

**UPDATE @OrdersByDateRange**

**SET Ranking ='A'**

**WHERE CustomerID IN (SELECT TOP 5 WITH TIES CustomerID**

**FROM (SELECT a.CustomerID,**

**COUNT(a.CustomerID)AS OrderTotal**

**FROM Customers a**

**JOIN Orders b ON a.CustomerID =b.CustomerID**

**GROUP BY a.CustomerID) AS DerivedTable**

**ORDER BY OrderTotal DESC)**

**RETURN**

**END**

The main difference between this example and the one in the previous section is that we were required to specify the structure of the @OrdersByDateRange table variable used to hold the resultset and list @OrdersByDateRange in the RETURNS clause. As you can see from the input parameter list, the function accepts a start date, an end date and an order count value to filter the resultset.

The first statement (--Statement 1) uses the input parameters to populate the table variable with customers who meet the specified criteria. The second statement (-Statement 2) updates the rows in table variable to identify the top five overall order placers. The IN portion of the UPDATE may seem a little confusing at first glance, but all its doing is using a derived table to select the CustomerID values of the top five order producers. Derived tables are discussed in Chapter 4. You can use the following to find the companies who have submitted more than two orders between 1/1/96 and 1/1/97.

**SELECT \***

**FROM fx\_OrdersByDateRangeAndCount ('1/1/96','1/1/97',2)**

**ORDER By Ranking**

**CustomerID CompanyName OrderCount Ranking**

**---------- ------------------------------ ---------- -------**

**ERNSH Ernst Handel 6 A**

**FOLKO Folk och fä HB 3 A**

**HUNGO Hungry Owl All-Night Grocers 5 A**

**QUICK QUICK-Stop 6 A**

**SAVEA Save-a-lot Markets 3 A**

**SEVES Seven Seas Imports 3 B**

**SPLIR Split Rail Beer &Ale 5 B**

**...**

The rows ranking values of ‘A’ indicate the top five order placers of all companies. The function allows you to perform two operations with one object. Retrieve the companies who have placed more than two orders between 1/1/96 and 1/1/97 and let me know if any of these companies are my top five order producers.

One of the advantages of using this type of function over a view is that the body of the function can contain multiple SQL statements to populate the table variable, whereas a view is composed of only one statement. The advantage of using multi-statement table-valued function versus a stored procedure is that the function can be referenced in the FROM clause of a SELECT statement while a stored procedure cannot. Had a stored procedure been used to return the same data, the resultset could only be accessed with the EXECUTE command.

#### A Real-World Example

Now that you have an idea of the different types of functions available in SQL Server 2000, let's wrap up this article with an example you might be able to use on one of your projects. The following statements create a function that determines the last day of the month (LDOM) for a given date parameter.

**CREATE FUNCTION fx\_LDOM**

**( @Date varchar(20) )**

**RETURNS datetime**

**AS**

**BEGIN**

**--ensure valid date**

**IF ISDATE(@Date) = 1**

**BEGIN**

**--determine first day of month**

**SET @Date = DATEADD(day,-DAY(@Date)+1,@Date)**

**--determine last day of month**

**SET @Date = DATEADD(day,-1,DATEADD(month,1,@Date))**

**END**

**ELSE**

**SET @Date = '1/1/80'**

**RETURN @Date**

**END**

The function's parameter (@Date) is defined as varchar(20), so error-checking code can be implemented. The ISDATE function makes sure the supplied value is a valid date. When an invalid date value is supplied, the function returns '1/1/80' to the calling statement. If you do not use this type of error-checking, the call to the function will fail when an invalid date is supplied.

When a valid date value is supplied, the DATEADD function is used to:

1. Determine the first day of the month, and
2. Determine the last day of the month.

If you have never used DATEADD before this may seem a little confusing, but a quick explanation should eliminate any that might exist. You use DATEADD to add or substract a date/time unit from a given date. The first parameter (in this case 'day') indicates the portion of the date that should be incremented. You can also specify year, quarter, month...millisecond. The second parameter is the number of units to add or substract. When subtracting, you simply make the value negative as shown in the example. The third parameter is the date value on which the calculation is performed.

The first day of the month is determined by calculating the number of elapsed days in the supplied parameter with the DAY function, adding 1 and then substracting it from the parameter. For an @Date value of 1/15/01, it simply subtracts 14 (15-1) days to get 1/1/01.

The last day of the month is determined by adding 1 month to the current month value and subtracting one day. So, 1/1/01 plus 1 month is equal to 2/1/01 and when you substract one day you get: 1/31/01.

The following shows how fx\_LDOM is used in a SELECT statement to calculate the number of days remaining in a month.

**CREATE TABLE fx\_Testing (DateValue datetime)**

**go**

**INSERT fx\_Testing VALUES ('1/1/01')**

**INSERT fx\_Testing VALUES ('2/15/01')**

**INSERT fx\_Testing VALUES ('2/15/02')**

**INSERT fx\_Testing VALUES ('2/15/03')**

**INSERT fx\_Testing VALUES ('2/15/04')**

**SELECT DateValue,**

**dbo.fx\_LDOM(DateValue) AS LDOM,**

**DATEDIFF(day,DateValue,dbo.fx\_LDOM(DateValue)) AS DaysLeftInMonth**

**FROM fx\_Testing**

**DateValue LDOM DaysLeftInMonth**

**------------------------ ----------------------- ---------------**

**2001-01-01 00:00:00.000 2001-01-31 00:00:00.000 30**

**2001-02-15 00:00:00.000 2001-02-28 00:00:00.000 13**

**2002-02-15 00:00:00.000 2002-02-28 00:00:00.000 13**

**2003-02-15 00:00:00.000 2003-02-28 00:00:00.000 13**

**2004-02-15 00:00:00.000 2004-02-29 00:00:00.000 14**

The DATEDIFF function is used to determine the difference between two dates. In this case, the number of days between the value in the DateValue column and the last day of the month calculated by fx\_LDOM.

**How do I create and use a Scalar User-Defined Function?**

A Scalar user-defined function returns one of the scalar data types. **Text, ntext, image and timestamp data types are not supported**. These are the type of user-defined functions that most developers are used to in other programming languages. You pass in 0 to many parameters and you get a return value. Below is an example that is based in the data found in the NorthWind Customers Table.

CREATE FUNCTION whichContinent

(@Country nvarchar(15))

RETURNS varchar(30)

AS

BEGIN

declare @Return varchar(30)

select @return = case @Country

when 'Argentina' then 'South America'

when 'Belgium' then 'Europe'

when 'Brazil' then 'South America'

when 'Canada' then 'North America'

when 'Denmark' then 'Europe'

when 'Finland' then 'Europe'

when 'France' then 'Europe'

else 'Unknown'

end

return @return

end

Because this function returns a scalar value of a varchar(30) this function could be used anywhere a varchar(30) expression is allowed such as a computed column in a table, view, a T-SQL select list item.

Stored procedures have long given us the ability to pass parameters and get a value back, but the ability to use a function in such a variety of different places where you cannot use a stored procedure make this a very powerful database object. Also notice the logic of my function is not exactly brain surgery. But it does encapsulate the business rules for the different continents in one location in my application. If you were to build this logic into T-SQL statements scattered throughout your application and you suddenly noticed that you forgot a country (like I missed Austria!) you would have to make the change in every T-SQL statement where you had used that logic. Now, with the SQL Server User-Defined Function, you can quickly maintain this logic in just one place.

# Scalar user-defined functions

Reuse of code is one of the fundamental principles we learn when programming in any language, and the SQL language is no exception. It provides many means by which to logically group code and reuse it.

One such means in SQL Server is the scalar user-defined function. It seems so convenient to hide away all those complex calculations in a function, and then simply invoke it in our queries. However, the hidden "sting in the tail" is that it can bring a heavy toll **in terms of performance. When used in a query, scalar functions are evaluated for each row and, with large tables, this can result in very slow running queries. This is especially true when the scalar function needs to access another table to retrieve data.**

Here is one example. Given tables with products and sales for products, the request is to retrieve total sales per product. Since the total sales value can be reused in another place, you decide to use a scalar function to calculate the total sales for a product:

CREATE FUNCTION dbo.GetTotalSales(@sku INT)

RETURNS DECIMAL(15, 2)

AS

BEGIN

  RETURN(SELECT SUM(sale\_amount)

         FROM Sales

         WHERE sku = @sku);

END

Then the query to retrieve the total sales for each product will look like this;

SELECT sku, product\_description, dbo.GetTotalSales(sku) AS total\_sales

FROM Products;

Isn't this a very neat and good looking query? But just wait until you run it over a large data set. The total sales calculation will be repeated for each and every row, and the overhead will be proportional to the number of rows. **The correct way to handle this is, if possible, is to rewrite the function as a table-valued function, or simply perform the calculation in the main query.** In our example, performing the calculation in the query will look like this:

SELECT P.sku, P.product\_description, SUM(S.sale\_amount) As total\_sales

FROM Products AS P

JOIN Sales AS S

  ON P.sku = S.sku

GROUP BY P.sku, P.product\_description;

And here is a table-valued function that can be used to calculate total sales:

CREATE FUNCTION dbo.GetTotalSales(@sku INT)

**RETURNS TABLE**

AS

RETURN(SELECT SUM(sale\_amount) AS total\_sales

       FROM Sales

       WHERE sku = @sku);

Scalar function –

SQL scalar functions return a single value, based on the input value.

Useful scalar functions:

* UCASE() - Converts a field to upper case
* LCASE() - Converts a field to lower case
* MID() - Extract characters from a text field
* LEN() - Returns the length of a text field
* ROUND() - Rounds a numeric field to the number of decimals specified
* NOW() - Returns the current system date and time
* FORMAT() - Formats how a field is to be displayed

Aggregate Functions -

SQL aggregate functions return a single value, calculated from values in a column.

Useful aggregate functions:

* AVG() - Returns the average value
* COUNT() - Returns the number of rows
* FIRST() - Returns the first value
* LAST() - Returns the last value
* MAX() - Returns the largest value
* MIN() - Returns the smallest value
* SUM() - Returns the sum

TABLE Functions –

User-defined functions that return a table data type can be powerful alternatives to views. These functions are referred to as table-valued functions. A table-valued user-defined function can be used where table or view expressions are allowed in Transact-SQL queries. While views are limited to a single SELECT statement, user-defined functions can contain additional statements that allow more powerful logic than is possible in views.

A table-valued user-defined function can also replace stored procedures that return a single result set. The table returned by a user-defined function can be referenced in the FROM clause of a Transact-SQL statement, but stored procedures that return result sets cannot.

[Components of a Table-Valued User-defined Function](javascript:void(0))

In a table-valued user-defined function:

* The RETURNS clause defines a local return variable name for the table returned by the function. The RETURNS clause also defines the format of the table. The scope of the local return variable name is local within the function.
* The Transact-SQL statements in the function body build and insert rows into the return variable defined by the RETURNS clause.
* When a RETURN statement is executed, the rows inserted into the variable are returned as the tabular output of the function. The RETURN statement cannot have an argument.

No Transact-SQL statements in a table-valued function can return a result set directly to a user. The only information the function can return to the user is the table returned by the function.

**How do I create and use an Inline Table-Value User-Defined Function?**

An Inline Table-Value user-defined function returns a table data type and is an exceptional alternative to a view as the user-defined function can pass parameters into a T-SQL select command and in essence provide us with a parameterized, non-updateable view of the underlying tables.

CREATE FUNCTION CustomersByContinent

(@Continent varchar(30))

**RETURNS TABLE**

AS

RETURN

SELECT dbo.WhichContinent(Customers.Country) as continent,

customers.\*

FROM customers

WHERE dbo.WhichContinent(Customers.Country) = @Continent

GO

SELECT \* from CustomersbyContinent('North America')

SELECT \* from CustomersByContinent('South America')

SELECT \* from customersbyContinent('Unknown')

Note that the example uses another function (WhichContinent) to select out the customers specified by the parameter of this function. After creating the user-defined function, I can use it in the FROM clause of a T-SQL command unlike the behavior found when using a stored procedure which can also return record sets. Also note that I do not have to reference the dbo in my reference to this function. However, when using SQL Server built-in functions that return a table, you must now add the prefix :: to the name of the function.

**How do I create and use a Multi-statement Table-Value User-Defined Function?**

A Multi-Statement Table-Value user-defined function returns a table and is also an exceptional alternative to a view as the function can support multiple T-SQL statements to build the final result where the view is limited to a single SELECT statement. Also, the ability to pass parameters into a T-SQL select command or a group of them gives us the capability to in essence create a parameterized, non-updateable view of the data in the underlying tables. Within the create function command you must define the table structure that is being returned. After creating this type of user-defined function, I can use it in the FROM clause of a T-SQL command unlike the behavior found when using a stored procedure which can also return record sets.

CREATE FUNCTION dbo.customersbycountry ( @Country varchar(15) )

RETURNS

@CustomersbyCountryTab table (

[CustomerID] [nchar] (5), [CompanyName] [nvarchar] (40),

[ContactName] [nvarchar] (30), [ContactTitle] [nvarchar] (30),

[Address] [nvarchar] (60), [City] [nvarchar] (15),

[PostalCode] [nvarchar] (10), [Country] [nvarchar] (15),

[Phone] [nvarchar] (24), [Fax] [nvarchar] (24)

)

AS

BEGIN

INSERT INTO @CustomersByCountryTab

SELECT [CustomerID],

[CompanyName],

[ContactName],

[ContactTitle],

[Address],

[City],

[PostalCode],

[Country],

[Phone],

[Fax]

FROM [Northwind].[dbo].[Customers]

WHERE country = @Country

DECLARE @cnt INT

SELECT @cnt = COUNT(\*) FROM @customersbyCountryTab

IF @cnt = 0

INSERT INTO @CustomersByCountryTab (

[CustomerID],

[CompanyName],

[ContactName],

[ContactTitle],

[Address],

[City],

[PostalCode],

[Country],

[Phone],

[Fax] )

VALUES ('','No Companies Found','','','','','','','','')

RETURN

END

GO

SELECT \* FROM dbo.customersbycountry('USA')

SELECT \* FROM dbo.customersbycountry('CANADA')

SELECT \* FROM dbo.customersbycountry('ADF')

**What are the benefits of User-Defined Functions?**

The benefits to SQL Server User-Defined functions are numerous. First, we can use these functions in so many different places when compared to the SQL Server stored procedure. The ability for a function to act like a table (for Inline table and Multi-statement table functions) gives developers the ability to break out complex logic into shorter and shorter code blocks. This will generally give the additional benefit of making the code less complex and easier to write and maintain. In the case of a Scalar User-Defined Function, the ability to use this function anywhere you can use a scalar of the same data type is also a very powerful thing. Combining these advantages with the ability to pass parameters into these database objects makes the SQL Server User-Defined function a very powerful tool.

**Summary**

So, if you have ever wanted to use the results of a stored procedure as part of a T-SQL command, use parameterized non-updateable views, or encapsulate complex logic into a single database object, the SQL Server 2000 User-Defined function is a new database object that you should examine to see if its right for your particular environment.

**VARIABLES:**

The difference between a variable and a parameter is:

Parameter - Represents a constant value that cannot be changed throughout the session run.

Variable - Represents a value that can be changed during session run. There are some functions available to change the variable value like setvariable(), setmaxvariable(),..

**USER Defined Variables**

When a variable is first declared, its value is set to NULL. To assign a value to a variable, use the SET statement. This is the preferred method of assigning a value to a variable. A variable can also have a value assigned by being referenced in the select list of a SELECT statement.

To assign a variable a value by using the SET statement, include the variable name and the value to assign to the variable. This is the preferred method of assigning a value to a variable. The following batch, for example, declares two variables, assigns values to them, and then uses them in the WHERE clause of a SELECT statement:

**What do Synonyms do?**

In what version of SQL Server were synonyms released, what do synonyms do and when could you make the case for using them?

* + Synonyms were released with SQL Server 2005.
  + Synonyms enable the reference of another object (View, Table, Stored Procedure or Function) potentially on a different server, database or schema in your environment. In short, this means that the original object that is referenced in all of your code is really using a completely different underlying object, but no coding changes are necessary. Think of this as an alias as a means to simplify migrations and application testing without the need to make any dependent coding changes.
  + Synonyms can offer a great deal of value when converting underlying database objects without breaking front end or middle tier code.  This could be useful during a re-architecture or upgrade project.
  + Additional information - [How and why should I use SQL Server 2005 synonyms?](http://www.mssqltips.com/sqlservertip/1076/how-and-why-should-i-use-sql-server-2005-synonyms/)
* What are bitwise operators and what is the value from a database design perspective?
  + The bitwise operators in SQL Server are:
    - & (Bitwise AND)
    - ~ (Bitwise NOT)
    - | (Bitwise OR)
    - ^ (Bitwise Exclusive OR)
  + From a database design perspective, bitwise operators can be used to store a complex set of criteria as a single value as opposed to having numerous lookup tables or numerous columns used as a 'flag' or condition indicator.
  + Additional information - [Using SQL Server Bitwise operators to store multiple values in one column](http://www.mssqltips.com/sqlservertip/1218/sql-server-bitwise-operators-store-multiple-values-in-one-column/)

***Question Difficulty = Advanced***

* **Question 1:** What two commands were released in SQL Server 2005 related to comparing data sets from two or more separate SELECT statements?
  + INTERSECT - Final result set where values in both of the tables match
  + EXCEPT - Final result set where data exists in the first dataset and not in the second dataset
  + Additional information - [Comparing Multiple SQL Server Datasets with the INTERSECT and EXCEPT operators](http://www.mssqltips.com/sqlservertip/1327/compare-sql-server-datasets-with-intersect-and-except/)

* **Question 2:** How can you capture the length of a column when it is a Text, NText and/or Image data type?
  + Use the DATALENGTH command to capture the length.
  + The LEN command is invalid for Text, NText and Image data types.
  + Additional information - [How to get length of data in Text, NText and Image columns in SQL Server](http://www.mssqltips.com/sqlservertip/1188/how-to-get-length-of-text-ntext-and-image-columns-in-sql-server/)

* **Question 3:** Is it possible to import data directly from T-SQL commands without using SQL Server Integration Services?  If so, what are the commands?
  + Yes - Six commands are available to import data directly in the T-SQL language.  These commands include:
    - BCP
    - Bulk Insert
    - OpenRowSet
    - OPENDATASOURCE
    - OPENQUERY
    - Linked Servers
  + Additional information - [Different Options for Importing Data into SQL Server](http://www.mssqltips.com/sqlservertip/1207/different-options-for-importing-data-into-sql-server/)

* **Question 4:** What is the native system stored procedure to issue a command against all databases?
  + The sp\_MSforeachdb system stored procedure accepts the @Command parameter which can be issued against all databases.  The '?' is used as a placeholder for the database name to issue the same command.
  + The alternative is to use a [cursor](http://www.mssqltips.com/sqlservertip/1599/sql-server-cursor-example/) to process specific commands against each database.
  + Additional information - [Run The Same SQL Command Against All SQL Server Databases](http://www.mssqltips.com/sqlservertip/1414/run-same-command-on-all-sql-server-databases-without-cursors/)

* **Question 5:** From a T-SQL perspective, how would you prevent T-SQL code from running on a production SQL Server?
  + Use IF logic with the @@SERVERNAME function compared against a string with a RETURN command before any other logic.
  + Additional information - [Preventing T-SQL code from running on a Production SQL Server](http://www.mssqltips.com/sqlservertip/1241/preventing-tsql-code-from-running-on-a-production-sql-server/)
* **1. Difference between SQL Server 2000 & SQL Server 2005 features, or new features of 2005 vs 2000.**  
  [- Ranking functions (ROW\_NUMBER, RANK, DENSE\_RANK, NTILE)](http://sqlwithmanoj.wordpress.com/2010/08/09/ranking-functions-with-sql-server-2005/)  
  [- Exception handling (TRY-CATCH block)](http://sqlwithmanoj.wordpress.com/2010/06/16/try-catch-exception-handling/)  
  [- CTE (Common Table Expressions)](http://sqlwithmanoj.wordpress.com/2011/05/23/cte-recursion-sequence-dates-factorial-fibonacci-series/)  
  [- PIVOT, UNPOVIT](http://sqlwithmanoj.wordpress.com/2009/04/12/ms-sql-server-2005-new-feature-pivot-and-unpivot/)  
  [- CUBE, ROLUP & GROUPING SET](http://sqlwithmanoj.wordpress.com/2010/11/12/cube-rollup-compute-compute-by-grouping-sets/)  
  [- SYNOMYMS](http://sqlwithmanoj.wordpress.com/2009/04/05/ms-sql-server-2005-new-feature-create-synonyms/)
* **2. What tools do you use for performance tuning?**  
  Query Analyzer, Profiler, Execution Plan, Index Wizard, Performance Monitor  
  Link: <http://www.sqlteam.com/article/sql-server-2000-performance-tuning-tools>
* **3. What is SQL Profiler? What it does? What template do you use?**  
  More Info: <http://www.extremeexperts.com/SQL/Articles/TraceTemplate.aspx>  
  <http://msdn.microsoft.com/en-us/library/ms190176.aspx>
* **4. How can you execute an SQL query from command prompt?**  
  OSQL & SQLCMD  
  More Info: <http://msdn.microsoft.com/en-us/library/ms162773.aspx>  
  <http://blog.sqlauthority.com/2009/01/05/sql-server-sqlcmd-vs-osql-basic-comparison/>  
  [http://www.databasejournal.com/features/mssql/article.php/3654176/SQL-Server-2005-Command-Line-Tool-SQLCMD–Part-I.htm](http://www.databasejournal.com/features/mssql/article.php/3654176/SQL-Server-2005-Command-Line-Tool-SQLCMD--Part-I.htm)
* **5. Difference between DELETE & TRUNCATE statement? Which statement can be Rollbacked?**  
  - With DELETE we can provide conditional WHERE clause to remove/delete specific rows, which is not possible with TRUNCATE.  
  - TRUNCATE is faster than DELETE as Delete keeps log of each row it deletes in transaction logs, but truncate keeps log of only de-allocated pages in transaction logs.  
  - Both statements can be rolled backed if provided in a transaction (BEGIN TRANS). If not then none of them can be rollbacked.  
  - DELETE is DML just like INSERT, UPDATE, but TRANCATE is DDL, just like CREATE, ALTER, DROP  
  **Link:** [***Complete differences on in Delete & Truncate***](http://sqlwithmanoj.wordpress.com/2009/02/22/difference-between-truncate-delete-and-drop-commands/)**.**
* **6. What are extended stored procedures? Can you create your own extended stored-proc?**  
  More Info: <http://msdn.microsoft.com/en-us/library/ms175200.aspx>  
  <http://msdn.microsoft.com/en-us/library/ms164627.aspx>
* **7. How can you execute a DOS command from SQL or through SQL query by using xp\_cmdshell?**
* [view source](http://sqlwithmanoj.wordpress.com/2010/12/09/tsql-interview-questions/#viewSource)
* 
* [print](http://sqlwithmanoj.wordpress.com/2010/12/09/tsql-interview-questions/#printSource)[?](http://sqlwithmanoj.wordpress.com/2010/12/09/tsql-interview-questions/#about)

|  |  |
| --- | --- |
| 1 | exec xp\_cmdshell 'dir c:\\*.exe' |

* More Info: <http://msdn.microsoft.com/en-us/library/aa260689%28SQL.80%29.aspx>  
  <http://msdn.microsoft.com/en-us/library/ms175046.aspx>
* **8. How will you insert result set of the above proc in a table?**
* [view source](http://sqlwithmanoj.wordpress.com/2010/12/09/tsql-interview-questions/#viewSource)
* 
* [print](http://sqlwithmanoj.wordpress.com/2010/12/09/tsql-interview-questions/#printSource)[?](http://sqlwithmanoj.wordpress.com/2010/12/09/tsql-interview-questions/#about)

|  |  |
| --- | --- |
| 1 | insert into |
| 2 | exec xp\_cmdshell 'dir c:\\*.exe' | |

* **9. What are Cursors and their types? What type do you use most and which one is fast?**  
  FORWARD-ONLY, FAST-FORWARD or READ-ONLY cursors.  
  Fastest to slowest: Dynamic, Static, and Keyset.  
  More Info: <http://msdn.microsoft.com/en-us/library/ms180169.aspx>  
  <http://www.sql-server-performance.com/tips/cursors_p1.aspx>
* **10. Why you should not use a cursor? What are its alternatives?**  
  Alternatives: while loops with temp tables, derived tables, correlated sub-queries, CASE stmt  
  More Info: <http://www.sql-server-performance.com/tips/cursors_p1.aspx>  
  <http://sqlserverpedia.com/wiki/Cursor_Performance_Issues>  
  <http://searchsqlserver.techtarget.com/feature/Part-3-Cursor-disadvantages>
* **11. Difference between LEFT JOIN with WHERE clause & LEFT JOIN with no WHERE clause.**  
  OUTER LEFT/RIGHT JOIN with WHERE clause can act like an INNER JOIN if not used wisely or logically.
* **12. How will you migrate an SSIS package from Development to Production environment?**  
  Do not include db connections and file paths in your workflow, instead create configuration files. This will help in deploying the pkg created in DEV server to Testing and finally to the PROD environment.  
  More Info: <http://msdn.microsoft.com/en-us/library/cc966389.aspx>  
  <http://www.wpconfig.com/2010/03/26/ssis-package-configurations/>
* **13. Multiple ways to execute a dynamic query.**  
  EXEC sp\_executesql, EXECUTE()  
  More Info: <http://sqlwithmanoj.wordpress.com/2011/03/22/execute-or-exec-vs-sp_executesql/>
* **14. Difference between COALESCE() & ISNULL()**  
  More Info: <http://sqlwithmanoj.wordpress.com/2010/12/23/isnull-vs-coalesce/>
* **15. Which of the following has higher performance:**  
  a. OR, AND  
  b. =, <>, >  
  c. IN, NOT IN, EXISTS  
  d. UNION, UNION ALL
* **16. What should be the ideal combination with IN & UNION (ALL) in terms of performance?**  
  a. SELECT \*FROM  
  WHERE
* IN (SELECT… UNION SELECT…)  
  OR  
  b. SELECT \* FROM
* WHERE
* IN (SELECT… UNION ALL SELECT…)
* **17. What is an IDENTITY column and its usage in INSERT statements?**  
  IDENTITY column can be used with a tables column to make it auto incremental, or a surrogate key.  
  MS BOL link: <http://msdn.microsoft.com/en-us/library/ms174639(v=SQL.90).aspx>
* **18. Can you create a Primary key without clustered index?**  
  Creation of PK automatically creates a clustered index upon the column(s).  
  More Info: <http://vadivel.blogspot.com/2006/03/primary-keys-without-clustered-index.html>
* **19. There are two tables one Master and another Feed table, both with 2 columns: ID & Price. Feed table gets truncated and re-populated every day.**
* Master Table Feed Table
* ID, Price ID, Price
* 1 100 1 200
* 3 200 2 250
* 5 300 4 500
* 6 400 6 750
* 7 500 7 800
* Create a job with an optimal script that will update the Master table by the Feed table.
* **20. What are CUBE & ROLLUP sets?**  
  CUBE & ROLLUP are the grouping sets used with GROUP BY clause and are very helpful in creating reports.  
  More Info: <http://sqlwithmanoj.wordpress.com/2010/11/12/cube-rollup-compute-compute-by-grouping-sets/>  
  <http://msdn.microsoft.com/en-us/library/ms177673.aspx>
* **21. What new indexes are introduced in SQL Server 2005 in comparison to 2000?**  
  - Spatial  
  - XML  
  More Info: <http://msdn.microsoft.com/en-us/library/ms175049.aspx>
* **22. What are XML indexes, what is their use?**  
  More Info: <http://msdn.microsoft.com/en-us/library/ms345121%28SQL.90%29.aspx>
* **23. How many types of functions (UDF) are there in SQL Server? What are inline functions?**  
  - Scalar functions  
  - Inline Table-valued functions  
  - Multi-statement Table-valued functions  
  More Info: <http://sqlwithmanoj.wordpress.com/2010/12/11/udf-user-defined-functions/>  
  <http://msdn.microsoft.com/en-us/library/ms189593.aspx>
* **24. How will you handle exceptions in SQL Server programming?**  
  By using TRY-CATCH constructs, putting our SQL statements/scripts inside the TRY block and error handling in the CATCH block, [**link**](http://sqlwithmanoj.wordpress.com/2010/06/16/try-catch-exception-handling/).  
  More Info: <http://msdn.microsoft.com/en-us/library/ms179296%28v=SQL.90%29.aspx>
* **25. How would you send an e-mail form SQL Server?**  
  [Configure Database mail here](http://sqlwithmanoj.wordpress.com/2010/09/29/database-mail-setup-sql-server-2005/).  
  More Info: <http://msdn.microsoft.com/en-us/library/ms175887%28v=SQL.90%29.aspx>
* What type of joins have you used?
  + Answer: Joins knowledge is MUST HAVE. This interview question is quite nice becausemost people used inner join and (left/right) outer join which is rather mandatory knowledgebut those more experienced will also mention cross join and self-join. In SQL Server you canalso get full outer join.
* How can you combine two tables/views together? For instance one tablecontains 100 rows and the other one contains 200 rows, have exactly the same fields andyou want to show a query with all data (300 rows). This sql interview question can getcomplicated.
  + Answer: You use UNION operator. You can drill down this question and ask what is thedifferent between UNION and UNION ALL (the first one removes duplicates (not alwaysdesirable)… in other words shows only DISTINCT rows….Union ALL just combines so it isalso faster). More tricky question are how to sort the view (you use order by at the last query),how to name fields so they appear in query results/view schema (first query field names areused). How to filter groups when you use union using SQL (you would create separate queryor use common table expression (CTE) or use unions in from with ().
* Question: What is the difference between where and having clause?
  + Answer: in SQL Where filters data on lowest row level. Having filters data after group by hasbeen performed so it filters on "groups"
* Question: How would apply date range filter?
  + Answer: This is tricky question. You can use simple condition >= and <= or similar or use between/and but the trick is to know your exact data type. Sometimes date fields contain time and that is where the query can go wrong so it is recommended to use some date related functions to remove the time issue. In SQL Server common function to do that is datediff function. You also have to be aware of different time zones and server time zone.
* Question: What type of wildcards have you used? This is usually one of mandatory sql interview question.
  + Answer: First question is what is a wildcard? Wildcards are special characters that allow matching string without having exact match. In simple word they work like contains or begins with. Wildcard characters are software specific and in SQL Server we have % which represent any groups of characters, \_ that represent one character (any) and you also get [] where we can [ab] which means characters with letter a or b in a specific place.
* Question: How do you find orphans?
  + Answer: This is more comprehensive SQL and database interview question. First of all we test if the candidate knows what an orphan is. **An Orphan is a foreign key value in "child table" which doesn’t exist in primary key column in parent table.** To get it you can use left outer join (important: child table on left side) with join condition on primary/foreign key columns and with where clause where primary key is null. Adding distinct or count to select is common practise. In SQL Server you can also except which will show all unique values from first query that dont exist in second query.
* Question: How would you solve the following sql queries using todays date? First day of previous month First day of current month Last day of previous month Last day of current month
  + Answer: These tasks require good grasp of SQL functions but also logical thinking which is one of the primary skills involved in solving sql questions. In this case I provided links toactual answers with code samples. Experienced people should give correct answer almost immediately. People with less experience might need more time or would require some help(Google).
  + SELECT DATEADD(month, DATEDIFF(month, 0, GETDATE())-1, 0),

DATEADD(month, DATEDIFF(month, 0, GETDATE()), 0)

Following script demonstrates the script to find last day of previous, current and next month.  
----Last Day of Previous Month  
SELECT DATEADD(s,-1,DATEADD(mm, DATEDIFF(m,0,GETDATE()),0))  
LastDay\_PreviousMonth

----Last Day of Current Month  
SELECT DATEADD(s,-1,DATEADD(mm, DATEDIFF(m,0,GETDATE())+1,0))  
LastDay\_CurrentMonth

----Last Day of Next Month  
SELECT DATEADD(s,-1,DATEADD(mm, DATEDIFF(m,0,GETDATE())+2,0))  
LastDay\_NextMonth  
**ResultSet:** *LastDay\_PreviousMonth  
———————–  
2007-07-31 23:59:59.000*

*LastDay\_CurrentMonth  
———————–  
2007-08-31 23:59:59.000*

*LastDay\_NextMonth  
———————–  
2007-09-30 23:59:59.000*

If you want to find last day of month of any day specified use following script.  
--Last Day of Any Month and Year  
DECLARE @dtDate DATETIME  
SET @dtDate = '8/18/2007'  
SELECT DATEADD(s,-1,DATEADD(mm, DATEDIFF(m,0,@dtDate)+1,0))  
LastDay\_AnyMonth  
**ResultSet:**  
*LastDay\_AnyMonth  
———————–  
2007-08-31 23:59:59.000*

* Intermediate SQL Interview questions and answersQuestion: You have a table that records website traffic. The table contains website name(multiple websites), page name, IP address and UTC date time. What would be thequery to show all websites visited in the last 30 days with total number or visits, total number if unique page view and total number of unique visitors (using IP Address)?
  + Answer: This test is mainly about good understanding of aggregate functions and date time. In this we need to group by Website, Filter data using datediff but the trick in here is to use correct time zone. If I want to do that using UTC time than I could use Get UTCDate() in sql server and the final answer related to calculated fields using aggregate functions that I will liston separate lines below: TotalNumberOfClicks = Count(\*) nothing special hereTotalUniqueVisitors = Count(distinct Ipaddress) we count ipaddress fields but only unique ipaddresses. The next field should be in here but as it is more complicated I put it as third field.TotalNumberOfUniquePageViews = Count(distinct PageName+IPAddress) This one is trickyto get unique pageview we need to count all visits but per page but only for unique IP address.So I combined pagename with ipaddress to counted unique values. Just to explain one pagecould receive 3 vists from 2 unique visits and another page could receive one visit from ip thatvisited previous page so Unique IP is 2, PageView is 3 (1 visitor 2 pages and 1 visitor 1 page)and visits is 4
* Question: How to display top 5 employees with the higest number of sales (total) and display position as a field. Note that if both of employees have the same total sales values they should receive the same position, in other words Top 5 employees might return more than 5 employees.
  + Answer: Microsoft introduced in SQL Server 2005 ranking function and it is ideal to solve this query. RANK() function can be used to do that, DENSE\_Rank() can also be used. Actually the question is ambiguous because if your two top employees have the same total sales which position should the third employee get 2 (Dense\_Rank() function) or 3 (Rank()Function)? In order to filter the query Common Table Expression (CTE) can be used or querycan be put inside FROM using brackets ().Now that we covered basic and intermediate questions lets continue with more complicate ones. These questions and answers are suitable for experienced candidates: Advanced SQL Interview Questions and answers
* Question: How to get accurate age of an employee using SQL?
  + Answer: The word accurate is crucial here. The short answer is you have to play with several functions. For more comprehensive answer see the following link SQL Age Function. Calculate accurate age using SQL Server
* Question: This is SQL Server interview question. You have three fields ID, Date and Total. Your table contains multiple rows for the same day which is valid data however for reporting purpose you need to show only one row per day. The row with the highest ID per day should be returned the rest should be hidden from users (not returned).To better picture the question below is sample data and sample output:ID, Date, Total1, 2011-12-22, 502, 2011-12-22, 150The correct result is:2, 2012-12-22, 150The correct output is single row for 2011-12-22 date and this row was chosen because it hasthe highest ID (2>1)
  + Answer: Usually Group By and aggregate function are used (MAX/MIN) but in this case thatwill not work. Removing duplications with this kind of rules is not so easy however SQLServer provides ranking functions and the candidate can use dense\_rank function partition byDate and order by id (desc) and then use cte/from query and filter it using rank = 1. There areseveral other ways to solve that but I found this way to be most efficient and simple.
* Question: How to return truly random data from a table? Let say top 100 random rows?I must admit I didnt answer correctly this sql interview question a few years back.
  + Answer: Again this is more SQL Server answer and you can do that using new\_id() function in order by clause and using top 100 in select. There is also table sample function but it is not truly random as it operates on pages not rows and it might not also return the number of rows you wanted.
* Question: How to create recursive query in SQL Server?
  + Answer: The first question is actually what is a recursive query? The most common example is parent child hierarchy for instance employee hierarchy where employee can have only one manager and manager can have none or many employees reporting to it. Recursive query can be create in sql using stored procedure but you can also use CTE (Common table expression) for more information visit SQL Interview question - recursive query (microsoft). It might be also worth asking about performance as CTE is not always very fast but in this case I don’t know which one is would perform betters. I will try to find time to add more questions soon. Feel free to suggest new questions (add comments).See also:SQL Server Interview questions and answers collection of interview question includingMicrosft BI stack.Hope that helps!Emil

**BI Questions**

**Explain the concepts and capabilities of Business Intelligence:**

Business Intelligence helps to manage data by applying different skills, technologies, security and quality risks. This also helps in achieving a better understanding of data. Business intelligence can be considered as the collective information. It helps in making predictions of business operations using gathered data in a warehouse. Business intelligence application helps to tackle sales, financial, production etc business data. It helps in a better decision making and can be also considered as a decision support system.

Business Intelligence is all about processes, skills, technologies, practices and applications used for supporting decision making.

Business Intelligence applications could perform

- Centrally initiated by the business needs  
- It includes decision support system, query reporting, OLAP, data mining, forecasting

#### Explain the Dashboard in the business intelligence.

A dashboard in business intellgence allows huge data and reports to be read in a single graphical interface. They help in making faster decisions by replying on measurable data seen at a glance. They can also be used to get into details of this data to analyze the root cause of any business performance. It represents the business data and business state at a high level. Dashboards can also be used for cost control. Example of need of a dashboard: Banks run thousands of ATM’s. They need to know how much cash is deposited, how much is left etc.

Dashboard in business intelligence is used for rapid prototyping, cloning and deployment for all databases, operational applications or spread sheets through an organization.

A dashboard in BI allows an enterprise’s status/position, heading to, by using graphs, maps and chars. The drill-down and roll-over capabilities allows organizing things without revealing important information. It is fully customizable, including free-form design options. Dashboard consolidates vital statistics of business into an easy-to-read page.

### What is Data warehousing?

**Answer**  
A data warehouse can be considered as a storage area where interest specific or relevant data is stored irrespective of the source. What actually is required to create a data warehouse can be considered as Data Warehousing. Data warehousing merges data from multiple sources into an easy and complete form.

##### Data warehousing - What is Data warehousing? - May 11, 2009 at 13:40 pm by Vidya Sagar

#### What is Data warehousing?

Data warehousing is a process of repository of electronic data of an organization. For the purpose of reporting and analysis, data warehousing is used. The essence concept of data warehousing is to provide data flow of architectural model from operational system to decision support environments

### What are fact tables and dimension tables?

**Answer**  
As mentioned, data in a warehouse comes from the transactions. Fact table in a data warehouse consists of facts and/or measures. The nature of data in a fact table is usually numerical.

On the other hand, dimension table in a data warehouse contains fields used to describe the data in fact tables. A dimension table can provide additional and descriptive information (dimension) of the field of a fact table.

**e.g. If I want to know the number of resources used for a task, my fact table will store the actual measure (of resources) while my Dimension table will store the task and resource details.**

Hence, the relation between a fact and dimension table is one to many.

##### Data warehousing - What are fact tables and dimension tables? - May 11, 2009 at 14:40 pm by Vidya Sagar

#### What are fact tables and dimension tables?

Business facts or measures and foreign keys are persisted in fact tables which are referred as candidate keys in dimension tables. Additive values are usually provided by the fact tables which acts as independent variables by which dimensional attributes are analyzed.

Attributes that are used to constrain and group data for performing data warehousing queries are persisted in the dimension tables.

### Explain the difference between data mining and data warehousing.

**Answer**  
Data warehousing is merely extracting data from different sources, cleaning the data and storing it in the warehouse. Where as data mining aims to examine or explore the data using queries. These queries can be fired on the data warehouse. Explore the data in data mining helps in reporting, planning strategies, finding meaningful patterns etc.

E.g. a data warehouse of a company stores all the relevant information of projects and employees. Using Data mining, one can use this data to generate different reports like profits generated etc.

##### Data warehousing - Difference between data mining and data warehousing - May 11, 2009 at 14:40 pm by Vidya Sagar

#### Explain the difference between data mining and data warehousing.

Data mining is a method for comparing large amounts of data for the purpose of finding patterns. Data mining is normally used **for models and forecasting**. Data mining is the process of correlations, patterns by shifting through large data repositories using pattern recognition techniques.

Data warehousing is the central repository for the data of several business systems in an enterprise. Data from various resources extracted and organized in the data warehouse selectively for analysis and accessibility.

### What is an OLTP system and OLAP system?

**Answer**  
OLTP: Online Transaction and Processing helps and manages applications based on transactions involving high volume of data. Typical example of a transaction is commonly observed in Banks, Air tickets etc. Because OLTP uses client server architecture, it supports transactions to run cross a network.

OLAP: Online analytical processing performs analysis of business data and provides the ability to perform complex calculations on usually low volumes of data. OLAP helps the user gain an insight on the data coming from different sources (multi dimensional).

##### Data warehousing - What is an OLTP system and OLAP system? - May 11, 2009 at 14:40 pm by Vidya Sagar

#### What is an OLTP system and OLAP system?

OLTP stands for OnLine Transaction Processing. Applications that supports and manges transactions which involve high volumes of data are supported by OLTP system. OLTP is based on client-server architecture and supports transactions across networks.

OLAP stands for OnLine Analytical Processing. Business data analysis and complex calculations on low volumes of data are performed by OLAP. An insight of data coming from various resources can be gained by a user with the support of OLAP.

### What is snow flake scheme design in database?

**Answer**  
A snowflake Schema in its simplest form is an arrangement of fact tables and dimension tables. The fact table is usually at the center surrounded by the dimension table. Normally in a snow flake schema the dimension tables are further broken down into more dimension table.

E.g. Dimension tables include employee, projects and status. Status table can be further broken into status\_weekly, status\_monthly.

##### Data warehousing - What is snow flake scheme design in database? - May 11, 2009 at 14:40 pm by Vidya Sagar

#### What is snow flake scheme design in database?

Snow flake schema is one of the designs that are present in database design. Snow flake schema serves the purpose of dimensional modeling in data warehousing. If the dimensional table is split into many tables, where the schema is inclined slightly towards normalization, then the snow flake design is utilized. It contains joins in depth. The reason is that, the tables split further.

### What is surrogate key? Explain it with an example.

**Answer**  
Data warehouses commonly use a surrogate key to uniquely identify an entity. A surrogate is not generated by the user but by the system. A primary difference between a primary key and surrogate key in few databases is that PK uniquely identifies a record while a SK uniquely identifies an entity.

E.g. an employee may be recruited before the year 2000 while another employee with the same name may be recruited after the year 2000. Here, the primary key will uniquely identify the record while the surrogate key will be generated by the system (say a serial number) since the SK is NOT derived from the data.

##### Data warehousing - What is surrogate key? - May 11, 2009 at 14:40 pm by Vidya Sagar

#### What is surrogate key? Explain it with an example.

A surrogate key is a unique identifier in database either for an entity in the modeled word or an object in the database. Application data is not used to derive surrogate key. Surrogate key is an internally generated key by the current system and is invisible to the user. As several objects are available in the database corresponding to surrogate, surrogate key can not be utilized as primary key.

For example, a sequential number can be a surrogate key.

### What is a level of Granularity of a fact table?

**Answer**  
A fact table is usually designed at a low level of Granularity. This means that we need to find the lowest level of information that can store in a fact table.

E.g. Employee performance is a very high level of granularity. Employee\_performance\_daily, employee\_perfomance\_weekly can be considered lower levels of granularity.

##### Data warehousing - What is a level of Granularity of a fact table? - May 11, 2009 at 14:40 pm by Vidya Sagar

#### What is a level of Granularity of a fact table?

The granularity is the lowest level of information stored in the fact table. The depth of data level is known as granularity. In date dimension the level could be year, month, quarter, period, week, day of granularity.

The process consists of the following two steps:

- Determining the dimensions that are to be included  
- Determining the location to place the hierarchy of each dimension of information

The factors of determination will be resent to the requirements.

### Explain the difference between star and snowflake schemas.

**Answer**  
A snow flake schema design is usually more complex than a start schema. In a star schema a fact table is surrounded by multiple fact tables. This is also how the Snow flake schema is designed. However, in a snow flake schema, the dimension tables can be further broken down to sub dimensions. Hence, data in a snow flake schema is more stable and standard as compared to a Start schema.

E.g. Star Schema: Performance report is a fact table. Its dimension tables include performance\_report\_employee, performance\_report\_manager

Snow Flake Schema: the dimension tables can be broken to performance\_report\_employee\_weekly, monthly etc.

##### Data warehousing - Difference between star and snowflake schemas - May 11, 2009 at 14:40 pm by Vidya Sagar

#### Explain the difference between star and snowflake schemas.

Star schema: A highly de-normalized technique. A star schema has one fact table and is associated with numerous dimensions table and depicts a star.

Snow flake schema: The normalized principles applied star schema is known as Snow flake schema. Every dimension table is associated with sub dimension table.

Differences:

* A dimension table will not have parent table in star schema, whereas snow flake schemas have one or more parent tables.
* The dimensional table itself consists of hierarchies of dimensions in star schema, where as hierarchies are split into different tables in snow flake schema. The drilling down data from top most hierarchies to the lowermost hierarchies can be done.

##### Data warehousing - Differences between star and snowflake schema.  - May 19, 2009 at 11:40 am by Rajmeet Ghai

#### Differences between star and snowflake schema.

**A snowflake schema is a more normalized form of a star schema**. In a star schema, one fact table is stored with a number of dimension tables. On the other hand, in a star schema, one dimension table can have multiple sub dimensions. This means that in a star schema, the dimension table is independent without any sub dimensions.

### What are fundamental stages of Data Warehousing?

**Answer**  
Stages of a data warehouse helps to find and understand how the data in the warehouse changes.

At an initial stage of data warehousing data of the transactions is merely copied to another server. Here, even if the copied data is processed for reporting, the source data’s performance won’t be affected.

In the next evolving stage, the data in the warehouse is updated regularly using the source data.

In Real time Data warehouse stage data in the warehouse is updated for every transaction performed on the source data (E.g. booking a ticket)

When the warehouse is at integrated stage, It not only updates data as and when a transaction is performed but also generates transactions which are passed back to the source online data.

##### Data warehousing - What are fundamental stages of Data Warehousing? - May 11, 2009 at 14:40 pm by Vidya Sagar

#### What are fundamental stages of Data Warehousing?

**Offline Operational Databases:** This is the initial stage of data warehousing. In this stage the development of database of an operational system to an off-line server is done by simply copying the databases.

**Offline Data warehouse:** In this stage the data warehouses are updated on a regular time cycle from operational system and the data is persisted in an reporting-oriented data structure.

**Real time Data Warehouse:** Data warehouses are updated based on transaction or event basis in this stage. An operational system performs a transaction every time.

**Integrated Data Warehouse:** The activity or transactions generation which are passed back into the operational system is done in this stage. These transactions or generated transactions are used in the daily activity of the organization.

#### What is Virtual Data Warehousing?

The aggregate view of complete data inventory is provided by Virtual Warehousing. The metadata is utilized for forming logical enterprise data model which is a part of database of record infrastructure , is contained in virtual data warehousing. The infrastructure consists of publishments of legacy database sysems with their metadta extracted. The standards JEE, JMS and EJBs are used in the infrastructure for the purpose of transactional unit requests and extract-tranform-load tools are used for loading real time bulk data.

##### Data warehousing - What is Virtual Data Warehousing? - May 19, 2009 at 10:40 AM by Rajmeet Ghai

#### What is Virtual Data Warehousing?

A virtual data warehouse provides a compact view of the data inventory. It contains Meta data. It uses middleware to build connections to different data sources. They can be fast as they allow users to filter the most important pieces of data from different legacy applications.

#### Difference between ER Modeling and Dimensional Modeling.

Dimensional modelling is very flexible for the user perspective. Dimensional data model is mapped for creating schemas. Where as ER Model is not mapped for creating shemas and does not use in conversion of normalization of data into denormalized form.

ER Model is utilized for OLTP databases that uses any of the 1st or 2nd or 3rd normal forms, where as dimensional data model is used for data warehousing and uses 3rd normal form.

ER model contains normalized data where as Dimensional model contains denormalized data.

##### Data warehousing - ER Modeling and Dimensional Modeling - May 19, 2009 at 10:40 am by Rajmeet Ghai

#### Difference between ER Modeling and Dimensional Modeling.

ER modeling that models an ER diagram represents the entire businesses or applications processes. This diagram can be segregated into multiple Dimensional models. This is to say, an ER model will have both logical and physical model. The Dimensional model will only have physical model.

|  |
| --- |
|  |