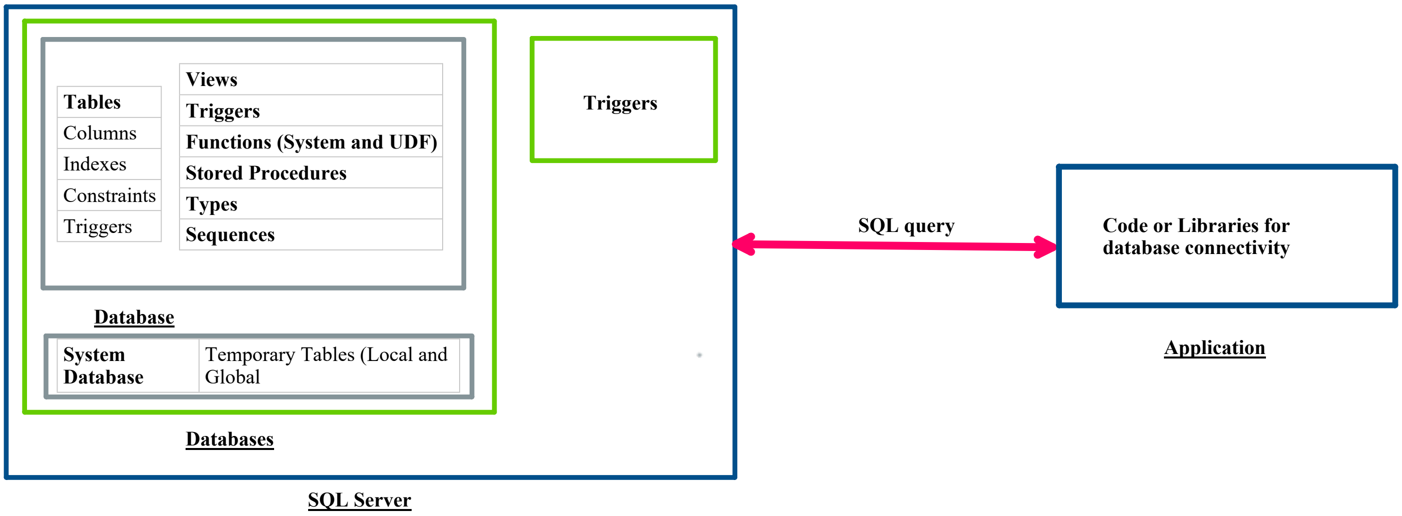
SQL Server Tree

Thursday, February 18, 2021 9:37 AM



Tables
Columns
Indexes
Constraints
Triggers

Views
Triggers
Functions (System and UDF)
Stored Procedures
Types
Sequences

Database

System Database
Temporary Tables (Local and Global

Databases

Triggers

SQL Server

SQL query

Code or Libraries for database connectivity

Application

1. **First Main Storage Things are**
   - □ **Temporary Tables**
   - □ **Tables (Permanent)**

A. Tables or Permanent Tables
- □ Can Store data as long as we wish
- □ Can Add Constraints on column Like
  - □ Default, Unique, Check
  - □ Primary Key, Foreign Key
  - □ Index
- □ Can Create DML Triggers and also DDL Triggers are also
- □ Followed By Tables.

A. Temporary tables

a. Local:
- □ Which Individual for Every User
- □ As Connection By User is Close table got destroy
- □ As Individual For User so possible Same Tables by Different Users so some Value is There along with table name, Which help to store duplicate table uniquely and stop same user to create duplicate table

b. Global
- □ Which Global among Every Connection
- □ No Duplication Allowed as All User can Access once Created By Any User
- □ Get Terminate when last user who using that terminate the connection, No matter who created that.
- □

- □ Note : DDL Trigger Not Applicable, View and Function is Not Applicable or temporary function, if temporary table is in store proc or function then as that function complete they got destroy.
- □ Other then that we can have all options which we can implement using table like constraint, Index, Keys and Many More.
- □ While Creating Table Just have to Add Prefix in table name # for Local and ## for Global

2. **Triggers**
   - Mainly Used Triggers are
     - ▪ **DML :** On Particular Table To Mange Manipulation of data Events
       - □ There are Two Types of DML triggers are there FOR and Instead of
       - □ FOR: Trigger after Operation P{performed
       - □ INSTEAD: Instead of that event, Mainly Useful for Manipulation of data from View
       - □ **Signature:**

```
Create trigger <trigger name>
 on tblname
For / instead Insert/Update/Delete
As
Begin
      // Code
End
```

       - □ In This Inserted and Deleted Table is Useful.

     - ▪ **DDL : It is used on Database or Server to trigger on defining or altering or dropping particular definition.**
       - □ **Signature :**

```
Create trigger <trigger name>
 on database / ALL server
For Event
As
Begin
      // Code
End
```

        - **LOGON:** If Event in DDL is Logon then it is logon trigger, and code is related that and generally it is on server
            - □ To Get Login Details sys.dm_exec_sessions table is used
            - □ To Get Name of Current User Original_Login()

    - Can Set Order of Triggers using sp_settriggerorder including paras like @triggername, @order : first/last, @namespace : database or server, @stmttype: Can Say Event

3. **<u>Data Types:</u>**
    - There are many types of data types are available like int, float, bigint, decimal, varchar(size), varbinary(size) and many more
    - Date and Time Related Types are also there like
        - Time : 3-5 Bytes
        - Date : 3 Byte
        - SmallDateTime : 4 Byte
        - DateTime : 8 Byte
        - DateTime2 : 6 - 8 Byte
        - DateTimeOffset 8 - 10 Byte
        - Note: Just Differnec Between all is Precession like 1day, 1 min to 100ns, can store time zone too.
    - There are many inbuild functions are there which are available in to perfrom operations or Get data Like For Date GETDATE(),ISDATE() and Many More
    - DATALENGTH() is useful to get size of data type.
    - Sys.types table have information of all data types
    - UNIQUEIDENTIFIER or Guid is very helpful when we need to store data which helps to identify uniquely across server, No doubt as it is largest size of data storage have to use only when require, GETID() to get new GUID().
    - CAST and Convert Function Helps To Cast one data type to other if possible, cast is more preferable where as Convert is just helpful when we want to apply style on date specifically.
    - May Possible Value is Null and Want To Preplace then for That ISNULL and Coalescc is helpful function.
    - If We Want to pass table as argument then can create our table type to an then need to just create variable of that type and then just want to pas
        - Create type name as table
          ```
          (
                structure
          )
          ```

4. **<u>Constraint and Index:</u>**
    - Basically Index is Useful to get data in very best way in terms of speed.
    - Types of Index are
        - Clustered, Non Clustered, Unique, Non Unique
    - Many Constraint like Primary key, Unique is using index indirectly.
    - Can Add Constraint Inline, At Time Of Declaration of table but after declaring all column or by altering table, Can also Drop the Constraint.
    - Few Constraint on Table where few are on column.
    - Can Add Identity to our column, and to check that Scope_Identity(), @@identity, Ident_current(tblname), To reset DBCC CHECKIDENT(tablname,RESEED,value).
    - <u>Signature :</u>
        - Index:
            - □ Create type/es index index name
              On tablename/viewname(column)
        - Constraint:
            - □ Constraint constraintname type(column name) if need then extra (On refrences tblname(column name) in fk)

5. **<u>Basics of Tables:</u>**
    - Can Create, Alter or Drop Table to deal with structure or definition of table   : For Data Definition
    - Can Insert, Update, Delete or Truncate data of table : For data Manipulation
    - Can Select or Retrieve Data : For Data Retrieval

6. **<u>Retrieval of Data:</u>**
    - Can Retrieve data using Select column list query.
        - Simple Select Query With Column list.

    - Can Replace Null Values by Null Replacement or casting or function value

    - Can Join Table using Join
        - Different Types of Join To Join table on relation and then retrieval of data
        - Types of Join : Inner, left, right, full, cross

    - Can Operate Set Operation Like Union, Intersect, Minus(Except).
        - Similar like Join We Can operate Set Operation on two different set of data of same structure
        - Set Operations like Union, Union All, Intersect, Except

    - Can Filter/short Data, Group or Aggregate Data
        - Can Filter Data Based On Conditions where clause using Operators like IN, NOT, Between, LIKE, other Logical and Conditional Ops
        - We Can Group Data By Group by and Over clause

        - □ Over clause is used with aggregate functions only and it can do partition and order by , Can Say Helpful for small grouping or instead of multiple set operations used like
            - ◆ Select SUM(colname) over(partition with colname) from tblname
        - □ Group by is helpful to aggregate data with having to filter
            - ◆ Have GROUPING(colname) to check column is aggregated or not, Grouping_ID(col list) can say to check level
            - ◆ GROUP BY(col list), CUBE(col list), ROLLUP(collist),GROUPING set((col1,col2),(col1),(col2),())
        - □ Order by is Useful to short data and when grouping is there then Grouping function is use here too for batter shorting, Distinct keyword to remove duplicates

- Can Insert Data retrieved data.
    - Select * into bactablename from table
    - To Copy Data or Structure of table.

- More :
    - Case Statement : Like If else
    - Choose : Same like Switch
    - IIF : Like Ternary Operator.
    - Alias name: name as alias
    - Cross Apply and Outer Apply are like join but useful when join with table value function
    - We Can Store Particular Value it declare variable too instead of printing.

7. **<u>Sub Query, Derived Table, Pivoting and Unpivoting:</u>**
    a. Sub Query: Generally to execute query into query is called Sub query.
    b. Derived Table : Can Say it just Store Query or intermediate data just like view or temporary table to manage our execution ways lik
        i. Table variable
        ii. Select * from (select query) as tblname
        iii. CTE: with tblname (columnlist) as (select query).
    c. Pivoting: Is Used To Rotate Data on any Dimension
        i. <u>Signature:</u>

```
Select Col1,col2
From tbl
PIVOT
(
    Agrregatefunction()
    For columnname IN (col1,col2)
)
```

Ex:

Gender	Population
Male	20000
Female	10244

Then PIVOTING ON GENDER

	Male	Female
Population	20000	10244

    d. Unpivoting : Revert of Pivoting
        i. <u>Signature:</u>

```
Select Column,Value
From tbl
UNPIVOT
(
    Agrregatefunction()
    For columnname IN (col1,col2)
)
```

8. **<u>Transaction and Error Handling:</u>**
    a. To Maintain Constancy of database we need use transaction so that if any failure will ocuur then either all or no any query reflect in database
        i. For Which Just Have To Write Begin Tran
        ii. Commit tran if No error
        iii. Rollback Tran if any error raise
    b. To Throw an we using raiserror function
    c. For Error Handling Unlike PL using try and catch (begin try end try, begin catch end catch)
    d. To Get Details about error have different function like Error_Numbrer() and many more and for log have store procedure sp_readerrorlog()
    e. Also Have Option of @@Error Function But it reset value before every line scan because of which have to maintain seprate variable for that

9. **<u>Functions:</u>**
    a. Three Types of function is there
        i. Scaler
            1) <u>Signature:</u>

```
Create function functionName(para with type)
Returns returntype
```

```
As
Begin
      //code
      Return return type variable
End
```

2) Can Not Return Timestamp, Cursor or Text value

ii. Inline Table Value Function

1) Signature:

```
Create function functionName(para with type)
Returns table
As
      Return (select query)
End
```

2) Treats as View
3) Changes are affect to base table is change is easy to implement otherwise throws an error

iii. Table Value Function

1) Signature:

```
Create function functionName(para with type)
Returns @tablename table (paras with datatype)
As
Begin
      //code
return
End
```

2) Treats as a stored Procedure
3) As It is return variable value not as table so Changes are not Possible as Value is assigned in table and returned now to variable is exists so.

10. View

a. View is also called as virtual table.

b. It is very similar to inline table value function, Difference is Can Create Trigger on View but not on function, View can Not Accept Value, Can Create index on View not on table and generally creating index in that case when data is not change frequently like OLAP and if View is indexed then it capable to store the data other then that it is just storing queries and also it is best way to share table with users as it is more secure in compare to direct access of table and whatever changes made is apply to base table is easy to implement otherwise it throw an error like in case of data modification affects to multiple table. Can Create only unique clustered index means Primary Key.

c. Signature

```
Create view name
As
Select query
```

d. Can Create, Alter or Drop a view.

11. Stored Procedure

a. It is one of safest way to execute queries.

b. Just a small difference between stored procedure and function is stored procedure can accept input paras and have output para where as functions not have output para as well stored procedure returns only integer not other then that where function can return any type of value.

c. Signature:

```
Create proc name
Para_list with type [=default value null or 0 based on type] in/out
As
Begin
      //code
end
```

d. Main Advantages of the store proc is One it is complied then plan is set and at runtime or every time no need to compile which saves time and traffic a lot, And Plan reuse every time, Database Logic is separate so debugging of big project is easy.

12. Cursor

a. It is not that much helpful as it read record row by row which is very costly so.

b. It is like pointer which read every row one by one.

c. Signature:

```
declare name  cursor FOR <select Query>
Open name
      fetch Next From name in @var1 @car2
While(@@fetch=0)
      fetch Next From name in @var1 @car2
Close name  // Release the result Set
Deallocate name;
```

13. Dynamic SQL

a. Basically It means Bind Query in such a way that it efficiently work

b. QUOTENAME('value',quote) is used to quote name so that safe from Injection Attack.

c. Signature:
Declare @sql varchar(1000);
Declare @params vrachar(1000);
Append Query with Paraname in @sql (set @sql='select * from @tablename')
Append Para Details in @params (set @params='@tablename varchar(20)')
Exec sp_executeSql @sql,@params,@tableName='tablename'

d. Can Use Exeute() to execute but it only accept sql not other parameter so need to need to bind para values too.

14. <u>More</u>

a. To Get Stored Object Details: select * from sysobjects, select * from Information sys.objects, select * from Information_Schema.

b. To Merge Same Structure:
Merge tablename as t1
Using tablename as t2
On t1.id=t2.id
When matched then
//code
When not matched by target then
//code
When not matched by source then

c. To Clear Plans cache DBCC FREEPROCACHE, sys.exec_cached_plans to get data.

d. SEQUENCE object to Create Counter

   i. <u>Signature :</u>
   Create Sequence dbo.Name
   As datatype number
   Start with val number
   Increment by number
   Minvalue number
   MaxValue number
   Cycle
   Cache number

   ii. Next value for dbo.sequence name

   iii. To Get Current Sequence current Value from sys.sequences where name=<name of sequence>

   iv. To Reset Alter sequence name reeset value