**T/SQL Programming**

Transaction structure query language.

Same as PL/SQL in Oracle.

Difference between SQL and T/SQL

|  |  |
| --- | --- |
| SQL | T/SQL |
| Declarative programming paradigm | Proceedural programming paradigm |
| Can execute a single line statement at a time | Execute block of statements at a time |
| Don’t have conditional statements and looing statements | Can implement conditional statements and looing statements |
| Doesnot provide reusability | Provide reusability by defining objects such as procedures and functions |

T/SQL commands can be embedded inside the programs where program is a block of code

T/SQL program blocks are can be divided into two types:

1. Anonymous blocks

Unnamed block of code

Executed any time but doesn’t store in database.

Can be written on query window and execute.

1. Sub-Programs

Named block of code.

Executed at any point of time and stored in database.

Provide reusability.

Declaring variables in T/SQL :

Syntax : declare @VariableName Datatype [=Value(optional)]

Example : declare @EmployeeId int

declare @EmployeeName varchar(20)=’Seetha’

Assigning values to variables

Syntax : set @variablename = <value>

Example : set @employeeId=1

Printing values

Syntax : print @variableName

Example : print @employeeId

Structure

Declare @ <var1> [data type][size]………………

Set @ <var>=<values>

<Statements>;

Print @<var>…….

Conditional Statements

declare @a int;

set @a=10;

if(@a>0)

print 'a is positive'

else if(@a=0)

print 'a is neutral'

else

print 'a is negative'

Syntax :

If (condition)

{ Statements }

Else

{ Statements }

Syntax :

declare @a int;

set @a=0;

while(@a<10)

begin

set @a=@a+1

print @a

end

While(condition)

{ statements }

**SUB PROGRAMS**

Named block of code

Directly saved on server and can be reused when required.

There are two types of sub programs in SQL:

1. Stored procedures
2. Stored functions.

**Stored procedures**

A database object which contains precompiled queries.

It is a block of code, designed to perform a task whenever called.

Why we need stored procedure?

* Whenever we want to run a query, first it will be compiled, and if we need the same query in some other location then the query has to be compiled and used again.
* But, if we use a stored procedure, it would be easier to use in multiple locations and performance will also be increased as it is a pre compiled. So, generally we write the query in a stored procedure and is used multiple times.

Advantages

* Unneccasary burden on database is reduced.
* Facilitates reusability.
* Performance is improved.
* User will get quick response

Proceedure creation without parameters

Syntax :

Create procedure proceedureName

As

Begin

<Statements>

End

Once the procedure is called, it is physically saved as a database object and used whenever possible needed.

We can call the procedure from anywhere, even in the Java or .net applications.

Execution of procedure: Exec <ProceedureName>

Drop a stored procedure : Drop procedure <procedure Name>

Example :

Create procedure Welcome

As

Begin

Print ‘Welcome to Stored proceedure’

End

Proceedure creation with parameters

Syntax

Create Procedures proceedurename(Passing parameters)

As

Begin

<statements>

End

Example :

create procedure test2(@a int,@b int)

as

begin

declare @c int

set @c=@a+@b;

print 'Addition of two variables are:-'+cast(@c as varchar);

end

Using the procedures for SQL Queries

1. To display employee details

Create procedure employeeDetails

As

Begin

Select \* from employee;

End

Execution : exec employeeDetails

1. Proceedure to accept employeeId and delete the record

Create procedure deleteWithEmpID

@employeeId int

As

Begin

Delete from employee where id=@employeeId

End

Execution : exec deleteWithEmpID 14

1. Procedure to update employee details

Create procedure updateEmployee

@eid int,@ename varchar(30),@eage int

As

Begin

Update employee

Set

eid=@eid,ename=@ename,eage=@eage

where eid=@eid;

end

Execution: exec updateEmployee 1,’Seeta’,25

1. To add records

create procedure insertEmployee

@eid int,@ename varchar(50),@eage int

as

begin

insert into Employee values(@eid,@ename,@eage)

end

Execution: exec insertEmployee 1,’Seeta’,25

**Stored functions**

Block of code, similar to stored procedure which also performs action and returns a result as value.

Stored functions can be divided into two types:

1. Scalar valued function

Returns an attribute datatype from the function.

Syntax:

Create Function <Function Name> (@parameter <Data Type> [size])

Returns <return attribute data type>

As

Begin

<Function Body>

Return <return attribute name>

End

Example : a function that takes an employee id and returns the salary of that

Employee

Create function getSalary(@empID)

Returns decimal

As

Begin

Declare @sal decimal

Select @sal=salary from employee where empid=@empID

Return @sal

End

Execution : select dbo.getSalary(1)

1. Table Valued function

Can return table as output

Syntax :

Create Function <Function Name> (@parameter <Data Type> [size])

Returns <Table>

As

Return <return select statement>

Calling a function

Select \* from functionName(parameters)

Example : accept the Address and returns the list of employee working in given address from the table. create function getAddress(@add text)

Returns table

As

Return (select \* from employee where empAddress=@add)

Calling function: select \* from getAddress(‘hyderabad’)

To drop function:

Drop function function\_name

**Difference between Stored procedure and Stored function**

|  |  |
| --- | --- |
| **Stored proceedure** | **Stored function** |
| Never returns a value | Returns a value |
| Can run with or without parameters | Always need parameters |
| Able to do select, insert, update and delete | Only select |
| Permit transaction management | Not permitted in function |
| Called by exec command | Called by select command |