National University of Computer and Emerging Sciences

Lab Manual 6

"Stored Procedures and Views"

Database Systems Lab

Fall 2020

Department of Computer Science

FAST-NU, Lahore, Pakistan

Table of Contents

1. Objective

The purpose of this lab manual is to introduce views and stored procedures and how to create them and use them.

2. Prerequisites

- SQL Server 2014 Database Development.
- Chapter 5 Elmasri

3. Task Distribution

Total Time	120 Minutes
Views	20 Minutes
Stored Procedures	20 Minutes
Exercise	60 Minutes
Evaluation	20 Minutes

4. Views

In previous lab manuals, you have learned how to write select query to retrieve data. While some select queries you write might be used only for one time activity, some select queries are used again and again within your application/environment. Some of these queries that you reuse within your environment contain complex logic, and you would not want to rewrite them every time you use them. SQL server allows you to store a SELECT statement within a database using an object called a view. In this section, you will learn how to CREATE a view, modify data through a view, how to ALTER a view, and how to use a view.

We will use the Student schema for all the examples (given in previous labs)

Students

StudentID	StudentName	StudentBatch	CGPA
1	Ali	2013	2.625
2	Aysha	2013	4
3	Ahmed	2013	2.2
4	Bilal	2012	2.5
5	Zafar	2012	3.5

Instructors

InstructorID	InstructorsName
1	Zafar
2	Sadia
3	Saima

Courses

CourseID	CourseName	CourseCreditHours	InstructorID
1	Computer Programming	3	1
2	Computer Organization	3	2
3	Computer Programmi...	1	NULL
4	Database	3	2
5	Database Lab	1	1

Registerations

StudentID	CourseID	GPA
1	1	3
1	3	3
1	4	2
1	5	3
2	1	2.5
2	2	0
2	4	3

Create a View

View is simply a select statement that has been given a name and stored in dataset. View is also called a virtual table, because there is no data in the view itself, it's just a select query that get data from base tables.

```
create View <ViewName>
AS
<Select Query>
```

When you excute a create view statement you should get command successful notification, just like when you created a table.

TRY IT

```
--CREATE A VIEW THAT GIVE NAMES OF ALL THE STUDENTS WITH GPA=3 IN ANY SUBJECT
Create View [3GPAStudents]
AS
Select S.StudentName
from Students S inner join Registration R on S.StudentID=R.StudentID
where R.GPA=3
```

Here the base tables are Student and Registration

TRY THIS

```
--Create a view to given Student Name, Roll Number and His CGPA (calcualte CGPA using Aggregation)
Create View StudentCGPA
as
Select S.StudentName,S.StudentID , SUM(C.CourseCreditHours* R.GPA)/ SUM(C.CourseCreditHours)  AS [CGPA]
From Students S inner join Registration R on R.StudentID=S.StudentID
inner join Courses C on C.CourseID=R.CourseID
Group by S.StudentName,S.StudentID
go
```

```
Messages
ommand(s) completed successfully.
```

Here the base tables are Students, Registration and Courses.

**NOTE: EVERY COLUMN RETURNED BY SELECT QUERY OF VIEW SHOULD HAVE UNIQUE NAME, DERIVED COLUMNS SHOULD BE GIVEN ALIAS. COLUMNS WITH SAME NAMES SHOULD ALSO BE GIVEN DISTINCT ALIAS

Use a View

As already told view are virtual tables. You can use them as regular tables in SELECT statement.

TRY IT

```
select * from  StudentCGPA
```

Results | Messages

StudentName	StudentID	CGPA
Ali	1	2.625
Aysha	2	1.83333333333333

**NOTE: this data was not present in StudentCGPA view, rather when you select a view, the Select query in body of view is executed and result is returned.

Similarly you can join views with tables of views, you can take aggregates of view.

TRY IT

```
--Give total number of students with CGPA >2
select COUNT(*) from  StudentCGPA
where CGPA>2
```

Results | Messages

(No column name)
1

Alter a View

You can change the select query of your view by using following syntax

Alter View <ViewName>
AS
<Select Query>

TRY IT

```
--Change you [3GPAStudents] view, now it should given student name and subject name in which student got 3 GPA
ALTER View [3GPAStudents]
AS
Select S.StudentName, C.CourseName
from Students S inner join Registration R on S.StudentID=R.StudentID
inner join Courses C on C.CourseID=R.CourseID
where R.GPA=3
```

Messages

```
ommand(s) completed successfully.
```

Now retrived the data from view

```
select * from  [3GPAStudents]
```

Results | Messages

StudentName	CourseName
Ali	Computer Programming
Ali	Computer Programming Lab
Ali	Database Lab
Aysha	Database

Insert Update Delete Data Through View

As view is a virtual table and has no data of its own, if you run delete insert or update query on view, the data in base table will change (if the change is feasible and is not violating any constraint). If the select query in View has joins and aggregates then delete insert or update would not work.
Read Elmasri Chapter 5 for more details.

TRY IT

```
Create View Students2013Batch
AS
Select  *
From Students
where StudentBatch=2013
go

insert into Students2013Batch
Values (12,'xyz',2014, 3)
go

Select  * from students
select * from Students2013Batch
go
```

Results | Messages

StudentID	StudentName	StudentBatch	CGPA
1	Ali	2013	2.625
2	Aysha	2013	4
3	Ahmed	2013	2.2
4	Bilal	2012	2.5
5	Zafar	2012	3.5
12	xyz	2014	3

StudentID	StudentName	StudentBatch	CGPA
1	Ali	2013	2.625
2	Aysha	2013	4
3	Ahmed	2013	2.2

With Check Option

With Check option ensures that the only data manipulation that can occur through view also must be retrievable though that view.
In previous example, the XYZ student we added though the view, was not retrievable thought view. If we add with check option that insertion would not have been possible though view.

TRY IT

```
Create View Students2014Batch
AS
Select  *
From Students
where StudentBatch=2014
with check option
go

insert into Students2014Batch
Values (13,'ABC',2014, 3)
go

Select  * from students
select * from Students2014Batch
go
```

Results | Messages

StudentID	StudentName	StudentBatch	CGPA
1	Ali	2013	2.625
2	Aysha	2013	4
3	Ahmed	2013	2.2
4	Bilal	2012	2.5
5	Zafar	2012	3.5
12	xyz	2014	3
13	ABC	2014	3

StudentID	StudentName	StudentBatch	CGPA
12	xyz	2014	3
13	ABC	2014	3

Now try adding a row that thought Student2014Batch that will not be retrievable though it

TRY IT

```
insert into Students2014Batch
Values (15,'ABC',2013, 3)
go
```

Messages

```
Msg 550, Level 16, State 1, Line 1
The attempted insert or update failed because the target view either specifies WITH CHECK OPTION or spans a view that specifies WITH CHECK OPTION an
The statement has been terminated.
```

5. Stored Procedures

Stored Procedure in SQL server can be defined as the set of logical group of SQL statements which are grouped to perform a specific task. A stored procedure is a prepared SQL code that you save so that you can reuse the code over and over again.

Benefits of Stored Procedures

Benefit	Explanation
Modular Programming	•You can write a stored procedure once, then call it from multiple places in your application hence reducing development time •It can accept input parameters, return output values as parameters, or return success or failure status messages
Performance	•Stored procedures provide faster code execution
	•Reduced network traffic
Security	•Users can execute a stored procedure without needing to execute any of the statements directly •Users can specifically be granted permission to execute only Stored procedures instead of allowing them to execute queries on tables directly.

Every time you execute simple SQL statements, syntax checking and compilation are done before execution and data return. However, syntax check and compilation is done while creating a procedure, and not on every execution which makes it faster than simple SQL statements.

Variables.

Before we start with stored procedures, we should get to know the variables. Like in any other programing language SQL also provides scalar variables, which are very useful when creating stored procedures.

- Variable in SQL start with @ symbol
- Variable is declared using DECLARE keyword as follow
  - *DECLARE @variableName datatype;*
    Or to declare multiple variables in one statement.
  - *DECLARE @variable1Name Datatype,@variable2Name datatype;*
- Variable can be assigned a constant scalar value as follow
  - *SET @ variableName = value;*
    Or To assign values to multiple variables in one statement
  - *select @ variable1Name = value, @variable2Name =value;*
- Variable can be assigned a scalar value thought SQL statement as well
  - *SELECT @vairableName = columnName FROM Table WHERE <condition>*
    If SQL query returns more than one row, 1st value will be assigned to variable
- You can retrieve the value of variable as follow
  - *Select @variableName*
- You can perform operations on variables like addition, concatenation, substring etc.

TRY IT

```
--• Variable is declared using DECLARE keyword as follow
DECLARE @Name varchar(10);

--Or to declare multiple variables in one statement.
DECLARE @FirstName varchar(10),@LastName  varchar(10);

--• Variable can be assigned a constant scalar value as follow
SET  @Name   = 'Ali Ahmed';

--Or To assign values to multiple variables in one statement
select @FirstName='Ali',@LastName='Ahmed';

--• Variable can be assigned a scalar value thought SQL statement as well
SELECT @Name = StudentName FROM Students WHERE  StudentBatch=2014
--This SQL query returns more than one row, so first name is assigned to the variable

--• You can retrieve the value of variable as follow
Select @Name, @FirstName, @LastName

--You can perform operations on variables like addition, concatenation, substring etc
Select @LastName+', '+@FirstName as FullName
```

Results Messages

(No column name)	(No column name)	(No column name)
ABC	Ali	Ahmed

FullName
Ahmed, Ali

NOTE: USE AND DECLARE VARIABLE IN SAME BATCH OF STATEMENTS, IF DECLARE STATEMENT IS NOT IN SAME BATCH, YOU WILL GET ERROR WHILE USING A VARIABLE.

CREATE Stored Procedure

Following is the syntax to create stored procedure: Input and output parameter a uses as required.

CREATE PROCEDURE [procedureName]
@input_param1 datatype,
@input_param2 datatype,
@output_param1 datatype OUTPUT,
@output _param2 datatype OUTPUT
AS
BEGIN

(SQL Queries)

```
END
go
```

How to execute Stored Procedure

```
declare @my_output_param1 int,
@my_output_param2 varchar(10) --these are the variables in which output variables of procedure will return values

Exec dbo.procedure_name
@input_param1=value,
@input_param2 =value,
@output_param1=@my_output_param1  OUTPUT  ,
@output_param2 =@my_output_param2 OUTPUT

select  @my_output_param1  ,@my_output_param2  – you will then have to use select statements to retrieve data from parameters
```

Stored Procedures without I/O parameters

TRY IT:

Create this procedure to obtain all the students of batch 2013

```
CREATE PROCEDURE StudentBatch2013
AS
BEGIN
    SELECT * FROM Students WHERE StudentBatch=2013
END
GO
```

```
Messages
Command(s) completed successfully.
```

Now execute this procedure

```
EXECUTE StudentBatch2013
```

Results | Messages

StudentID	StudentName	StudentBatch	CGPA
1	Ali	2013	2.625
2	Aysha	2013	4
3	Ahmed	2013	2.2

Stored procedure with input parameters

TRY IT
Create a SP which takes batchNo as input and returns all students of that batch.

```
Create Procedure StudentofBatch
@Batch int
AS
BEGIN
    select * from Students where StudentBatch=@Batch
END
go
```

```
Messages
mmand(s) completed successfully.
```

Now execute it

```
Declare @BatchNo int =2014
Execute StudentofBatch
@Batch=@BatchNo
```

Results | Messages

StudentID	StudentName	StudentBatch	CGPA
12	xyz	2014	3
13	ABC	2014	3

Store Procedures with output parameters

TRY IT:
Create a stored procedure that will return max CGPA in an output parameter

```
Create Procedure GetHighestCGPA
@highestCGPA float OUTPUT
AS
BEGIN

    Select top 1 @highestCGPA=   CGPA  from Students order by CGPA desc

END
go
```

```
Messages
mmand(s) completed successfully.
```

Execute it

```
Declare @CGPA float

Execute GetHighestCGPA
@highestCGPA= @CGPA output

select @CGPA
```

Results | Messages

(No column name)
4

QUESTION: WRITE A SP TO GET AVERAGE CGPA.

IF-ELSE conditions

Like in any programing language IF—ELSE in SQL provide ability to conditionally execute a code.
TRY THIS

```
--CREATE A STORED PROCEDURE THAT TAKES A TEACHER ID AS INPUT
--AND RETURN A INT Flag= 1  as OUTPUT IF ANY TEACHER EXISTS WITH THAT NAME
--AND RETRIVES ALL THOSE TEAHERS ARE WELL, IF NO TEACHER EXISTS OF THAT NAME Flag 0

Create Procedure GetTeacherByName
@Name int,
@Flag int OUTPUT
AS
BEGIN
 if exists (Select * from Instructors where InstructorsName=@Name)
 Begin
    set @Flag=1
    Select * from Instructors where InstructorsName=@Name
 end
 else
 Begin
    set @Flag=0
 end
END
go
```

Messages

```
mmand(s) completed successfully.
```

Execute it

```
Declare @outflag float

Execute GetTeacherByName
 @Name='ALI' ,
@Flag= @outflag output

select @outflag
go
```

TRY ANOTHER

```
--CREATE a STORE PROCEDURE THAT TAKES A CHARACTER FROM A-Z AS INPUT
--AND RETRIEVES ALL STTUDENTS WITH NAME STARTING WITH THAT LETTER
--IF AN INVALID LETTER IS GIVEN AS INPUT THE PROCEDUTE SHOULD PRINT 'INVALID LETTER, ONLNY a-Z ALLOWED'
--letter should not be case sensitive
create Procedure GetStudents
@letter varchar(30)
AS
BEGIN
 if LOWER(@letter) like '[a-z]'
 Begin

    Select * from Students where StudentName like @letter+'%'
 end
 else
 Begin
     print 'INVALID LETTER, ONLNY a-Z ALLOWED'
 end
END
go
```

TRY EXECUTING THESE

```
execute GetStudents @letter= 'B'

execute GetStudents @letter= '1'
```

Self-exploration

- What are default values? How can you set default values of parameters of Stored Procedures?
- How can you alter your procedure? (Hint same as View)