Seminar 4

12th November 2021

Group 927

# Functions, Views, System Catalog, Triggers, MERGE – in SQL Server

## User-defined functions

* Defined by developers
* Used in sql queries
* 3 types:
  + Scalar
  + Inline table-valued
  + Multi-statement table-valued

### Scalar functions

* Return a scalar/value
* Disadvantage – the function is applied for each row => impact on performance

Example:

CREATE FUNCTION ufNoPersons(@age INT)

RETURNS INT AS

BEGIN

DECLARE @no INT

SET @no = 0

SELECT @no = COUNT(\*)

FROM Persons P

WHERE age = @age

RETURN @no

END

GO

PRINT dbo. ufNoPersons(20)

Dbo – database owner

Example with a function returning a varchar value

CREATE/ALTER FUNCTION ufNamePerson(@id INT)

RETURNS VARCHAR(100) AS

BEGIN

DECLARE @name VARCHAR(100)

SET @name = ‘’

SELECT @name = name

FROM Persons P

WHERE id = @id

END

GO

### Inline table-valued functions

* Return table
* Can be used in FROM clauses

Example

CREATE FUNCTION ufPersonsNames(@age INT)

RETURNS TABLE

AS

RETURN

SELECT name

FROM Persons P

WHERE age = @age

GO

SELECT \*

FROM dbo. ufPersonsNames(20)

SELECT \*

FROM ufPersonsNames(20)

Schema dbo is optional in SELECT queries, SQL with automatically search for the schema in some contexts

### Multi-statement table-valued functions

* Return a table
* Can contain more than just one statement

Example:

CREATE/ ALTER FUNCTION ufPersonsFilteredByAge(@age INT)

RETURNS @PersonsAge TABLE (pid INT, pname VARCHAR(100))

AS

BEGIN

INSERT INTO @PersonsAge

SELECT pid, name

FROM Persons P

WHERE age = @age

IF @@ROWCOUNT = 0

INSERT INTO @PersonsAge VALUES (0, ‘No persons found with the specified age’)

RETURN

END

GO

SELECT \*

FROM dbo. ufPersonsFilteredByAge(20)

SELECT \*

FROM ufPersonsFilteredByAge(20)

Schema dbo is optional in SELECT queries, SQL with automatically search for the schema in some contexts

## Views

* Virtual tables, represent or aggregate data from one or more tables
* The view’s columns and row can be defined in a query
* Maximum no. columns: 1024

CREATE VIEW view\_name

AS SELECT\_STATEMENT

Example

CREATE OR ALTER VIEW view\_exams

AS

SELECT S.sid, S.name, C.cid, C.name, E.grade

FROM Students S INNER JOIN Exams E ON S.sid = E.sid INNER JOIN Courses C ON E.cid = C.cid

GO

SELECT \*

FROM view\_exams

## System Catalog

* Build-in tables / system tables
* Managed by the server
* Provide info about the database objects: tables, trigger, stored procedures, user-defined functions etc.
* Examples:
  + sys.objects – each row in this table assigned to a database object
    - \_\_type = ‘P’ | ‘PK’ | ‘FK’ | ‘U’ | ‘IF’ | ‘TF’ | ‘FN’ | ‘TR’ | ‘D’ | ‘C’

SELECT \*

From sys.objects

Where \_\_type = ‘PK’

* + sys.columns – every row represents a column for the db objects that have columns such as tables, views
  + sys.indexes
  + sys.parameters
  + sys.triggers
  + sys.sql\_modules

## Triggers

* special type of stored procedures
* automatically executed
* can NOT be executed directly by the user
* are executed when a DML (INSERT, UPDATE, DELETE) or DDL (CREATE, ALTER, DROP) event is triggered

-- SQL Server Syntax (<https://docs.microsoft.com/en-us/sql/t-sql/statements/create-trigger-transact-sql?view=sql-server-ver15>)

-- Trigger on an INSERT, UPDATE, or DELETE statement to a table or view (DML Trigger)

CREATE [ OR ALTER ] TRIGGER [ schema\_name . ]trigger\_name

ON { table | view }

[ WITH <dml\_trigger\_option> [ ,...n ] ]

{ FOR | AFTER | INSTEAD OF }

{ [ INSERT ] [ , ] [ UPDATE ] [ , ] [ DELETE ] }

[ WITH APPEND ]

[ NOT FOR REPLICATION ]

AS

{ sql\_statement [ ; ] [ ,...n ] | EXTERNAL NAME <method specifier [ ; ] > }

<dml\_trigger\_option> ::=

[ ENCRYPTION ]

[ EXECUTE AS Clause ]

<method\_specifier> ::=

assembly\_name.class\_name.method\_name

* if multiple triggers defined for the same table and the same event => get executed in **a random** order
* when the trigger gets executed is specified by “ FOR | AFTER | INSTEAD OF “:
  + FOR or AFTER – the DML trigger is executed when all statements specified in the trigger are launched successfully
  + INSTEAD OF – the DML trigger statements will get executed instead of the SQL statements that triggered it
* Triggers have access to two tables: *inserted* and *deleted* (the tables can also be accessed within the OUTPUT clause)

Examples:

CREATE TRIGGER when\_adding\_book

ON Books

FOR INSERT

AS

BEGIN

INSERT INTO BookLogs (booktitle, bookauthors, operation\_date)

SELECT title, authors, GETDATE()

FROM inserted

End

GO

CREATE TRIGGER [dbo].[when\_delete\_book]

ON [dbo].[Books]

FOR DELETE

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO BooksSell(booktitle, bookauthors, operation\_date)

SELECT title, authors, GETDATE()

FROM deleted

End

GO

CREATE TRIGGER [dbo].[when\_update\_book]

ON [dbo].[Books]

FOR UPDATE

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO BookLogs(old\_title, old\_authors, new\_title, new\_authors, operation\_date)

SELECT d.title, d.authors, i.title, i.authors, GETDATE()

FROM deleted d INNER JOIN inserted I On d.bid = i.bid

End

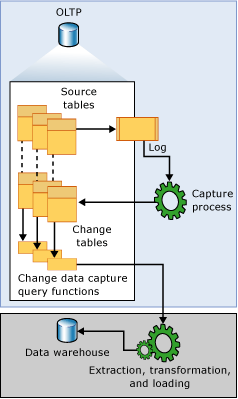
GO

SET NOCOUNT ON; - row count is not returned anymore when executing an INSERT/UPDATE/DELETE

SET NOCOUNT OFF – the no. of rows affected is returned

@@ROWCOUNT – is updated anyway

## Change DATA Capture



* Introduced in 2008
* Data about DML changes in the table/ db
* sys.sp\_cdc\_enable\_db
* sys.sp\_cdc\_enable\_table – CDC for monitored tables
* info and data are archived and monitored automatically (triggers are used for this)
* monitored tables -> Logs -> Logs serve as input to the capturing process -> mirror tables containing the columns, data, and metadata of the monitored tables

## The MERGE Statement

Example:

MERGE Books

USING

(

SELECT MAX(bid), bid, title, MAX(Authors) Authors, MAX(NoPages) Pages

FROM Books

GROUP BY title

) MergeData ON Books.bid = MergeData.bid

WHEN MATCHED THEN

UPDATE SET Books.Title = MergeData.Title

Books.Authors = MergeData.Authors

Books.NoPages = MergeData.NoPages

WHEN NOT MATCHED BY SOURCE THEN DELETE;

|  |  |  |  |
| --- | --- | --- | --- |
| bid | Title | Authors | NoPages |
| 1 | Jane Eyre | NULL | 600 |
| 2 | Jane Eyre | Charlotte Brontë | NULL |
| 3 | Jane Eyre | NULL | NULL |

|  |  |  |  |
| --- | --- | --- | --- |
| bid | Title | Authors | NoPages |
| 3 | Jane Eyre | Charlotte Brontë | 600 |

* a source table is compared with a target table
* execute INSERT/DELETE/ UPDATES based on the result of the comparison
* operations are executed on the target table joined with the source table

MERGE TargetTable AS Target

USING SourceTable AS Source

ON (search terms)

WHEN MATCHED THEN

UPDATE SET

Or

DELETE

WHEN NOT MATCHED [BY TARGET] THEN

INSERT

WHEN NOT MATCHED BY SOURCE THEN

UPDATE SET

Or

DELETE