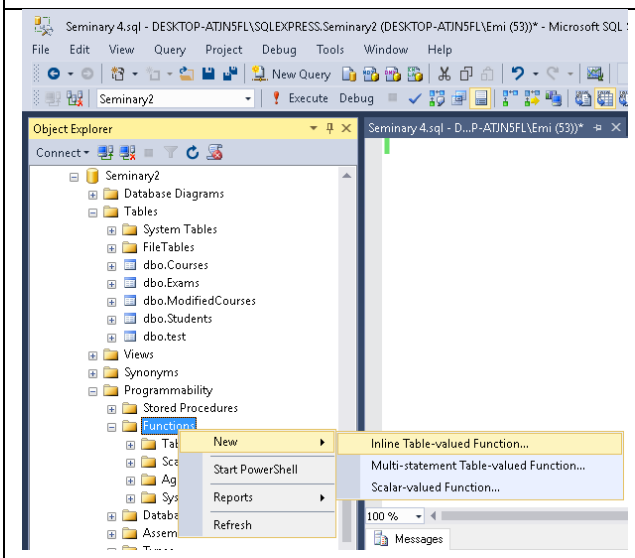
Databases
Seminary 4

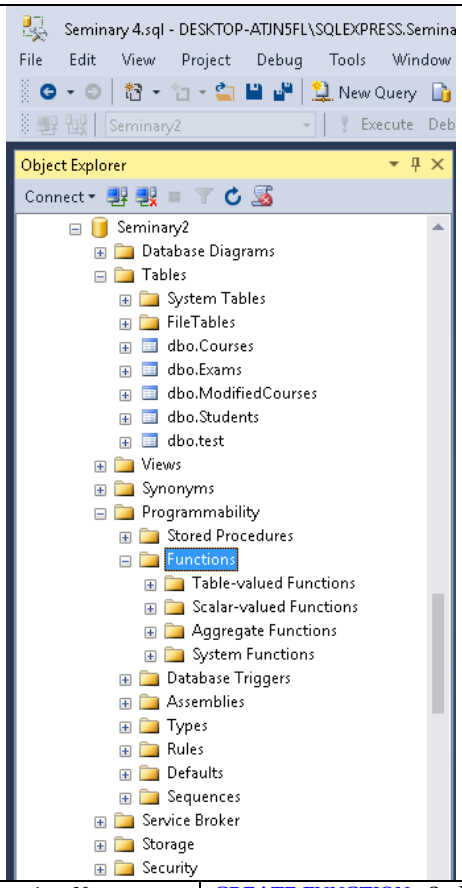
# ***Functions, views, system tables, triggers, MERGE instruction – in SQL Server***

## ***Functions defined by users***

- the programmer can define his/her own functions; these ones can be use later in SQL queries;
- 3 types of functions defined by users in SQL Server

a. scalar
b. inline table-valued
c. multi-statement table valued




```
CREATE FUNCTION <Inline_Function_Name, sysname, FunctionName>
(
	-- Add the parameters for the function here
	<@param1, sysname, @p1> <Data_Type_For_Param1, , int>,
	<@param2, sysname, @p2> <Data_Type_For_Param2, , char>
)
RETURNS TABLE
AS
RETURN
(
	-- Add the SELECT statement with parameter references here
	SELECT 0
)
GO
```

```
CREATE FUNCTION <Table_Function_Name, sysname, FunctionName>
(
	-- Add the parameters for the function here
	<@param1, sysname, @p1> <data_type_for_param1, , int>,
	<@param2, sysname, @p2> <data_type_for_param2, , char>
)
RETURNS <@Table_Variable_Name, sysname, @Table_Var> TABLE
(
	-- Add the column definitions for the TABLE variable here
	<Column_1, sysname, c1> <Data_Type_For_Column1, , int>,
	<Column_2, sysname, c2> <Data_Type_For_Column2, , int>
)
AS
BEGIN
```

```
CREATE FUNCTION <Scalar_Function_Name, sysname, FunctionName>
(
	-- Add the parameters for the function here
	<@Param1, sysname, @p1> <Data_Type_For_Param1, , int>
)
RETURNS <Function_Data_Type, ,int>
AS
BEGIN
	-- Declare the return variable here
	DECLARE <@ResultVar, sysname, @Result> <Function_Data_Type, ,int>

	-- Add the T-SQL statements to compute the return value here
	SELECT <@ResultVar, sysname, @Result> = <@Param1, sysname, @p1>

	-- Return the result of the function
	RETURN <@ResultVar, sysname, @Result>
```

| | | |
|---|---|---|
| | -- Fill the table variable with the rows for your result set<br><br>RETURN<br>END<br>GO | END<br>GO |

a. Scalar functions

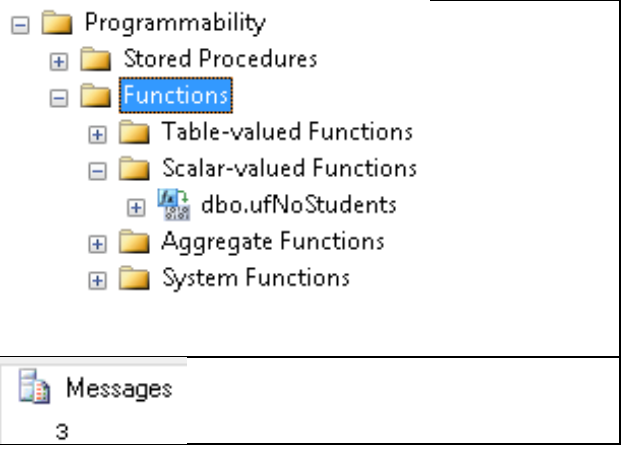
- return a value
- disadvantage – in the case of a scalar function that operates on more rows, SQL Server executes the function once for each row from the result, fact that can have a significant impact through the performance
- example

```
select * from Students
```

Results | Messages

| | sid | sname | email | age | sgroup |
|---|---|---|---|---|---|
| 1 | 1 | Maria | m@yahoo.com | 21 | 926 |
| 2 | 1234 | Ada | a@cs.ro | 20 | 921 |
| 3 | 1235 | Razvan | r@cs.ro | 21 | 921 |
| 4 | 1236 | Monica | m@cs.ro | 20 | 922 |
| 5 | 1237 | Paula | p@cs.ro | 20 | 924 |

```
-- drop function ufNoStudents
CREATE FUNCTION ufNoStudents(@age INT)
RETURNS INT AS
BEGIN
DECLARE @no INT
SET @no = 0
SELECT @no= COUNT(*)
FROM Students
WHERE age = @age
RETURN @no
END
GO
```




```
PRINT dbo.ufNoStudents(20)
```

Messages

3

b. Inline table-valued functions

- return a table
- can be used anywhere where a table can be inside of a T-SQL query (ussualy in the FROM clause)
- example

```
CREATE FUNCTION ufStudentsName(@age INT)
RETURNS TABLE
AS
RETURN
SELECT sname
FROM Students
WHERE age= @age
GO
```

```
SELECT *
FROM ufStudentsName(20)
```

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

Results | Messages

| | sname |
|---|---|
| 1 | Ada |
| 2 | Monica |
| 3 | Paula |

c. Multi-statement table-valued functions

- returns a table
- the difference with the inline table-valued functions is that these ones can have more than one instruction

```
select * from Courses
```

Results | Messages

| | cid | cname | credits |
|---|---|---|---|
| 1 | Alg1 | Algorithms 1 | 7 |
| 2 | DB1 | Databases 1 | 6 |
| 3 | DB2 | Databases Modified | 6 |
| 4 | MAP1 | MAP 1 | 5 |
| 5 | MAP2 | MAP 2 | 4 |

```
CREATE FUNCTION ufCoursesWithCredits(@credits INT)
RETURNS @CoursesWithCredits TABLE (cid
varchar(10), cname VARCHAR(50))
AS
BEGIN
INSERT INTO @CoursesWithCredits
SELECT cid, cname
FROM Courses
WHERE credits = @credits
IF @@ROWCOUNT = 0
INSERT INTO @CoursesWithCredits
VALUES (0,'No course was found for that number of
credits.')
RETURN
END
GO
```

```
SELECT * FROM ufCoursesWithCredits(5)

SELECT * FROM ufCoursesWithCredits(6)

SELECT * FROM ufCoursesWithCredits(8)
```

Results | Messages

| | cid | cname |
|---|---|---|
| 1 | MAP1 | MAP 1 |

| | cid | cname |
|---|---|---|
| 1 | DB1 | Databases 1 |
| 2 | DB2 | Databases Modified |

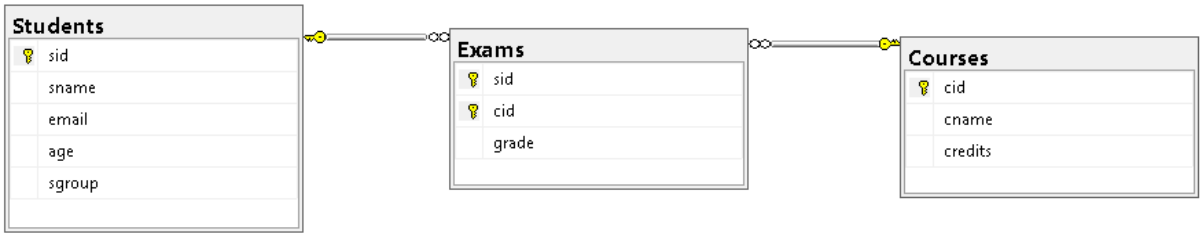
| | cid | cname |
|---|---|---|
| 1 | 0 | No course was found for that number of credits. |

***Views***

- a view creates a virtual table that represent the dates from one or more tables in an alternative way;
- the content of the virtual table (columns and rows) is defined by a query;
- it can have a maximum of 1024 columns;
- the syntax

```
CREATE VIEW view_name
AS SELECT_instruction
```

- example




```
CREATE VIEW vStudentsExam
AS
       SELECT s.sid, sname, sgroup, grade
       FROM Students s INNER JOIN Exams e ON s.sid=e.sid
GO

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

```
select * from vStudentsExam
```

Results | Messages

| | sid | sname | sgroup | grade |
|---|---|---|---|---|
| 1 | 1234 | Ada | 921 | 9 |
| 2 | 1234 | Ada | 921 | 9 |
| 3 | 1235 | Razvan | 921 | 10 |
| 4 | 1237 | Paula | 924 | 9 |

```
-- with encryption
CREATE VIEW vStudentsAge
WITH ENCRYPTION
AS
SELECT sname, age
FROM Students
WHERE age>=19;
GO

select * from vStudentsAge
```

Results | Messages

| | sname | age |
|---|---|---|
| 1 | Maria | 21 |
| 2 | Ada | 20 |
| 3 | Razvan | 21 |
| 4 | Monica | 20 |
| 5 | Paula | 20 |

```
-- with check option
CREATE VIEW vSEC
AS
SELECT s.sname, c.cname, e.grade
FROM Students s INNER JOIN Exams e ON
s.sid=e.sid
INNER JOIN Courses c ON c.cid=e.cid
WHERE e.grade>=5
WITH CHECK OPTION ;
GO

select * from vSEC
```

Results | Messages

| | sname | cname | grade |
|---|---|---|---|
| 1 | Ada | Algorithms 1 | 9 |
| 2 | Ada | MAP 2 | 9 |
| 3 | Razvan | Algorithms 1 | 10 |
| 4 | Paula | Databases Modified | 9 |

```
ALTER TABLE test
ADD tDate DATETIME

-- using built-in functions
CREATE VIEW vTest
AS
SELECT TOP (10) tid, SUM(code) AS CodeSum
FROM test
WHERE tDate > CONVERT(DATETIME,'20001231',101)
GROUP BY tid;
GO

select * from vTest
```

dbo.test
- Columns
  - tid (PK, int, not null)
  - tname (varchar(50), null)
  - description (varchar(50), null)
  - code (int, null)
  - tDate (datetime, null)

Results | Messages

| | tid | CodeSum |
|---|---|---|

## ***System Tables***

- special tables with information about all the objects created in a database (tables, columns, indexes, stored procedures, functions defined by the user, views, and so on)
- are managed by the server (are not modified directly by the user)
- examples:
- *sys.objects* – contains a row for each object (constraint, stored procedure, table and so on) created in a database;
- *sys.columns* – a row for each column of an object that has columns, e.g. tables or views;
- *sys.sql_modules* – a row for each object that it is a defined module in the SQL language in SQL Server (e.g., objects like procedures, functions, with a SQL module asociated).

```
select * from sys.objects
```

Results | Messages

| | name | object_id | principal_id | schema_id | parent_object_id | type | type_desc | create_date | modify_date | is_ms_shipped | is_published | is_schema_publi |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | sysrscols | 3 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2016-04-30 00:44:21.290 | 2016-04-30 00:44:21.300 | 1 | 0 | 0 |
| 2 | sysrowsets | 5 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2009-04-13 12:59:11.093 | 2016-04-30 00:44:21.993 | 1 | 0 | 0 |
| 3 | sysclones | 6 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2016-04-30 00:44:21.750 | 2016-04-30 00:44:21.760 | 1 | 0 | 0 |
| 4 | sysallocunits | 7 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2009-04-13 12:59:11.077 | 2016-04-30 00:44:21.340 | 1 | 0 | 0 |
| 5 | sysfiles1 | 8 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2003-04-08 09:13:38.093 | 2003-04-08 09:13:38.093 | 1 | 0 | 0 |

Query executed successfully. DESKTOP-ATJN5FL\SQLEXPRESS ... DESKTOP-ATJN5FL\Emi (54) Seminary2 00:00:00 125 rows

```
select * from sys.columns
```

Results | Messages

| | object_id | name | column_id | system_type_id | user_type_id | max_length | precision | scale | collation_name | is_nullable | is_ansi_padded | is_rowguidcol | is_identity | is_computed | is_filestre |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 3 | rsid | 1 | 127 | 127 | 8 | 19 | 0 | NULL | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 3 | rscolid | 2 | 56 | 56 | 4 | 10 | 0 | NULL | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 3 | hbcolid | 3 | 56 | 56 | 4 | 10 | 0 | NULL | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 3 | rcmodified | 4 | 127 | 127 | 8 | 19 | 0 | NULL | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 3 | ti | 5 | 56 | 56 | 4 | 10 | 0 | NULL | 0 | 0 | 0 | 0 | 0 | 0 |

Query executed successfully. DESKTOP-ATJN5FL\SQLEXPRESS ... DESKTOP-ATJN5FL\Emi (54) Seminary2 00:00:00 984 rows

```
select * from sys.sql_modules
```

Results | Messages

| | object_id | definition | uses_ansi_nulls | uses_quoted_identifier | is_schema_bound | uses_database_collation | is_recompiled | null_on_null_input |
|---|---|---|---|---|---|---|---|---|
| 1 | 354100302 | CREATE PROCEDURE uspNameStudents(@age INT) AS ... | 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 386100416 | CREATE PROCEDURE uspNoStudents(@age INT, @No INT ... | 1 | 1 | 0 | 0 | 0 | 0 |
| 3 | 418100530 | CREATE FUNCTION ufNoStudents(@age INT) RETURNS IN... | 1 | 1 | 0 | 0 | 0 | 0 |
| 4 | 434100587 | CREATE FUNCTION ufStudentsName(@age INT) RETURNS... | 1 | 1 | 0 | 0 | 0 | 0 |
| 5 | 450100644 | CREATE FUNCTION ufCoursesWithCredits(@credits INT) RE... | 1 | 1 | 0 | 1 | 0 | 0 |

Query executed successfully. DESKTOP-ATJN5FL\SQLEXPRESS ... DESKTOP-ATJN5FL\Emi (54) Seminary2 00:00:00 18 rows

```
SELECT SCHEMA_NAME(schema_id) AS schema_name, name AS
table_name
FROM sys.tables
WHERE OBJECTPROPERTY(object_id,'TableHasPrimaryKey') = 0
ORDER BY schema_name, table_name;
GO
```

Results | Messages

| schema_name | table_name |
|---|---|

***Triggers***

- a special type of stored procedures;
- are automatically executed when a DML (Data Manipulation Language) instruction is executed (INSERT, UPDATE, DELETE) or a DDL (Data Definition Language) instruction (e.g. CREATE_DATABASE, CREATE_TABLE, DROP_TABLE, DROP_LOGIN, UPDATE_STATISTICS, DROP_TRIGGER, ALTER_TABLE);
- are not executed directly
- trigger syntax for the instruction INSER/UPDATE/DELETE on a table or view:

```
CREATE TRIGGER < trigger_name>
ON { table | view}
[ WITH <trigger_option_DML > [ ,...n ] ]
```

```
{ FOR | AFTER | INSTEAD OF }
{ [INSERT] [,] [UPDATE] [,] [DELETE] }
[ WITH APPEND ]
[ NOT FOR REPLICATION ]
AS
{ sql_instruction [;] [ ,...n ] |
EXTERNAL NAME <method specifier[;] > }
```

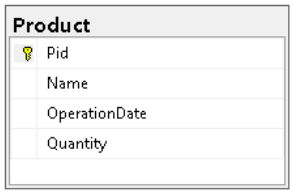
- the execution moment for a trigger is specified through one of the options:
  - FOR, AFTER – the DML trigger is starting only when all the operations specified in the starting instruction have been executed with success (more triggers can be defined);
  - INSTEAD OF - the DML trigger is executed instead of the starting action.
- if more triggers are defined for the same action, these ones are executed random;
- when a trigger is executed, one can access 2 special tables called *inserted* and *deleted*.
- example

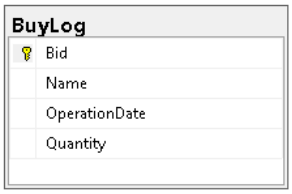
```
Create table Product(
Pid int primary key identity,
Name varchar(50),
OperationDate date,
Quantity int
)

insert into Product Values ('cherries', '2018-11-11', 5),
('oranges', '2018-12-11', 6)

create table BuyLog(
Bid int primary key identity,
Name varchar(50),
OperationDate date,
Quantity int
)

select * from Product
select * from BuyLog
```

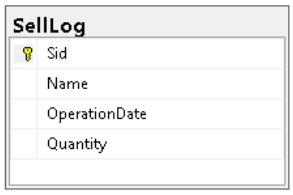
Results | Messages

| | Pid | Name | OperationDate | Quantity |
|---|---|---|---|---|
| 1 | 1 | cherries | 2018-11-11 | 5 |
| 2 | 2 | oranges | 2018-12-11 | 6 |

| Bid | Name | OperationDate | Quantity |
|---|---|---|---|



```sql
CREATE TRIGGER Add_Product
ON Product
FOR INSERT
AS
BEGIN
INSERT INTO BuyLog (Name, OperationDate, Quantity)
SELECT Name, GETDATE(), Quantity
FROM inserted
END
GO
--
select * from Product
select * from BuyLog
```

Results | Messages

| | Pid | Name | OperationDate | Quantity |
|---|---|---|---|---|
| 1 | 1 | cherries | 2018-11-11 | 5 |
| 2 | 2 | oranges | 2018-12-11 | 6 |

| | Bid | Name | OperationDate | Quantity |
|---|---|---|---|---|

```sql
insert into Product Values ('tomatoes', '2018-11-11', 5),
('potatoes', '2018-12-11', 6)
select * from Product
select * from BuyLog
-- after create trigger - one obtain the values inserted in
table Product also in table BuyLog
```

Results | Messages

| | Pid | Name | OperationDate | Quantity |
|---|---|---|---|---|
| 1 | 1 | cherries | 2018-11-11 | 5 |
| 2 | 2 | oranges | 2018-12-11 | 6 |
| 3 | 3 | tomatoes | 2018-11-11 | 5 |
| 4 | 4 | potatoes | 2018-12-11 | 6 |

| | Bid | Name | OperationDate | Quantity |
|---|---|---|---|---|
| 1 | 1 | potatoes | 2018-11-13 | 6 |
| 2 | 2 | tomatoes | 2018-11-13 | 5 |

Query executed successfully.

```sql
select * from Product
select * from BuyLog

CREATE TRIGGER Delete_Product
ON Product
FOR DELETE
AS
BEGIN
SET NOCOUNT ON;
INSERT INTO BuyLog(Name, OperationDate, Quantity)
SELECT Name, GETDATE(), Quantity
FROM deleted
END
GO

delete from Product
Where Quantity=6 and Name LIKE 'p%'
-- the rows with potatoes should be deleted from
Product and introduced in BuyLog

select * from Product
select * from BuyLog
```

Results | Messages

| | Pid | Name | OperationDate | Quantity |
|---|---|---|---|---|
| 1 | 1 | cherries | 2018-11-11 | 5 |
| 2 | 2 | oranges | 2018-12-11 | 6 |
| 3 | 4 | potatoes | 2018-12-11 | 6 |
| 4 | 5 | potatoes | 2018-12-11 | 6 |

| | Bid | Name | OperationDate | Quantity |
|---|---|---|---|---|
| 1 | 1 | potatoes | 2018-11-13 | 6 |
| 2 | 2 | tomatoes | 2018-11-13 | 5 |
| 3 | 3 | potatoes | 2018-11-13 | 6 |

Command(s) completed successfully.

Results | Messages

| | Pid | Name | OperationDate | Quantity |
|---|---|---|---|---|
| 1 | 1 | cherries | 2018-11-11 | 5 |
| 2 | 2 | oranges | 2018-12-11 | 6 |

| | Bid | Name | OperationDate | Quantity |
|---|---|---|---|---|
| 1 | 1 | potatoes | 2018-11-13 | 6 |
| 2 | 2 | tomatoes | 2018-11-13 | 5 |
| 3 | 3 | potatoes | 2018-11-13 | 6 |
| 4 | 4 | potatoes | 2018-11-13 | 6 |
| 5 | 5 | potatoes | 2018-11-13 | 6 |

```sql
create table SellLog(
Sid int primary key identity,
Name varchar(50),
OperationDate date,
Quantity int)

select * from Product
select * from BuyLog
select * from SellLog
```

Results | Messages

| | Pid | Name | OperationDate | Quantity |
|---|---|---|---|---|
| 1 | 1 | cherries | 2018-11-11 | 5 |
| 2 | 2 | oranges | 2018-12-11 | 6 |

| | Bid | Name | OperationDate | Quantity |
|---|---|---|---|---|
| 1 | 1 | potatoes | 2018-11-13 | 6 |
| 2 | 2 | tomatoes | 2018-11-13 | 5 |
| 3 | 3 | potatoes | 2018-11-13 | 6 |
| 4 | 4 | potatoes | 2018-11-13 | 6 |
| 5 | 5 | potatoes | 2018-11-13 | 6 |

| | Sid | Name | OperationDate | Quantity |
|---|---|---|---|---|

```sql
CREATE TRIGGER Modify_Update_Product
ON Product
FOR UPDATE
AS
BEGIN
SET NOCOUNT ON;
INSERT INTO SellLog(Name, OperationDate, Quantity)
SELECT d.Name, GETDATE(), d.Quantity - i.Quantity
FROM deleted d INNER JOIN inserted i ON d.Pid = i.Pid
WHERE i.Quantity < d.Quantity
INSERT INTO BuyLog(Name, OperationDate, Quantity)
SELECT i.Name, GETDATE(), i.Quantity - d.Quantity
FROM deleted d INNER JOIN inserted i ON d.Pid = i.Pid
WHERE i.Quantity > d.Quantity
END
GO
```

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

```sql
update Product
set Quantity=8
WHERE Quantity=6

select * from Product
select * from BuyLog
select * from SellLog
```

Results | Messages

| | Pid | Name | OperationDate | Quantity |
|---|---|---|---|---|
| 1 | 1 | cherries | 2018-11-11 | 5 |
| 2 | 2 | oranges | 2018-12-11 | 8 |

| | Bid | Name | OperationDate | Quantity |
|---|---|---|---|---|
| 1 | 1 | potatoes | 2018-11-13 | 6 |
| 2 | 2 | tomatoes | 2018-11-13 | 5 |
| 3 | 3 | potatoes | 2018-11-13 | 6 |
| 4 | 4 | potatoes | 2018-11-13 | 6 |
| 5 | 5 | potatoes | 2018-11-13 | 6 |
| 6 | 6 | oranges | 2018-11-13 | 2 |

| | Sid | Name | OperationDate | Quantity |
|---|---|---|---|---|

```
update Product
set Quantity=Quantity+1
where Name LIKE '%es'

select * from Product
select * from BuyLog
select * from SellLog
```

Results | Messages

| | Pid | Name | OperationDate | Quantity |
|---|---|---|---|---|
| 1 | 1 | cherries | 2018-11-11 | 6 |
| 2 | 2 | oranges | 2018-12-11 | 9 |

| | Bid | Name | OperationDate | Quantity |
|---|---|---|---|---|
| 1 | 1 | potatoes | 2018-11-13 | 6 |
| 2 | 2 | tomatoes | 2018-11-13 | 5 |
| 3 | 3 | potatoes | 2018-11-13 | 6 |
| 4 | 4 | potatoes | 2018-11-13 | 6 |
| 5 | 5 | potatoes | 2018-11-13 | 6 |
| 6 | 6 | oranges | 2018-11-13 | 2 |
| 7 | 7 | oranges | 2018-11-13 | 1 |
| 8 | 8 | cherries | 2018-11-13 | 1 |

| | Sid | Name | OperationDate | Quantity |
|---|---|---|---|---|

***SET NOCOUNT***

SET NOCOUNT ON/OFF

- the number of the affected rows by a T-SQL instruction or by a stored procedure:
    - no run (ON) – the count is not returned.
    - run (OFF) - the count is returned.
- @@ROWCOUNT always is modifying.

```
select * from test
```

Results | Messages

| | tid | tname | description | code | tDate |
|---|---|---|---|---|---|
| 1 | 1 | test2 | decription 2 | 88 | NULL |
| 2 | 2 | test3 | decription 3 | 888 | NULL |
| 3 | 3 | test4 | decription 4 | 8888 | NULL |

```
SET NOCOUNT ON
insert into test(tname, description, code)
values ('t', 'd', 9)
print @@ROWCOUNT
```

```
1
```

```
SET NOCOUNT OFF
insert into test(tname, description, code)
values ('t', 'd', 10)
print @@ROWCOUNT
```

```
(1 row(s) affected)
1
```

```
select * from test
```

Results | Messages

| | tid | tname | description | code | tDate |
|---|---|---|---|---|---|
| 1 | 1 | test2 | decription 2 | 88 | NULL |
| 2 | 2 | test3 | decription 3 | 888 | NULL |
| 3 | 3 | test4 | decription 4 | 8888 | NULL |
| 4 | 5 | t | d | 9 | NULL |
| 5 | 6 | t | d | 10 | NULL |

***Change Data Capture (CDC)***

- information about the DML changes on the table/database;
- introduced in SQL Server 2008;
- sys.sp_cdc_enable_db – CDC for the database;
- sys.sp_cdc_enable_table – CDC for the monitories tables;

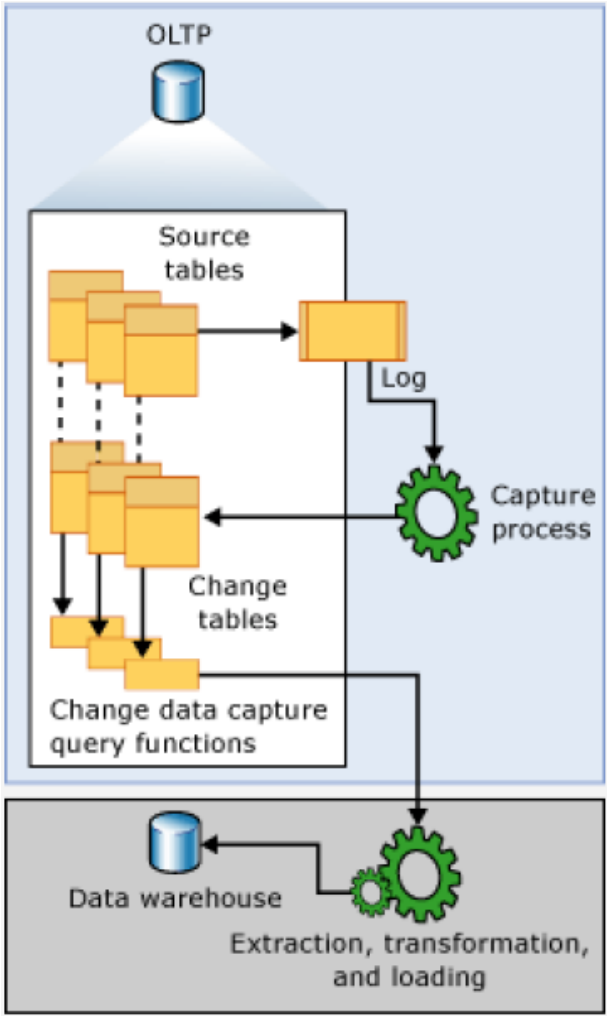
- allows the data archiving and monitoring without a supplementary programming effort (e.g., by writing triggers);
- the changing from the tables created by the user are monitored;
- the result data from the monitoring are stored in tables that can queried with SQL;
- mirror tables are created and contain the columns of the monitoring tables + the metadata that describe the changes.

```
EXECUTE sys.sp_cdc_enable_db
```

```
Msg 22988, Level 16, State 1, Procedure sp_cdc_enable_db, Line 14 [Batch Start Line 256]
This instance of SQL Server is the Express Edition (64-bit). Change data capture is only
available in the Enterprise, Developer, and Enterprise Evaluation editions.
```

```
EXECUTE sys.sp_cdc_enable_table
    @source_schema = N'dbo'
  , @source_name = N'test'
  , @role_name = N'cdc_Admin';
```

```
Msg 22988, Level 16, State 1, Procedure sp_cdc_enable_table, Line 24 [Batch Start Line
257]
This instance of SQL Server is the Express Edition (64-bit). Change data capture is only
available in the Enterprise, Developer, and Enterprise Evaluation editions.
```



***MERGE instruction***

- allows the comparison between the rows from a source table and a destination table;
- based on the result of the comparison, one can execute INSERT, UPDATE or DELETE commands, e.g., one can execute modification operations / delete operations on a destination table based on the result of a join with a source table.

```
MERGE Table_Definition AS Destination
USING (Source_Table) 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
```

- example

```
create table Books(
Bid int primary key,
Title varchar(50),
Author varchar(50),
ISBN varchar(20),
Pages int)

select * from Books
```

Results | Messages

| Bid | Title | Author | ISBN | Pages |
|---|---|---|---|---|

```
insert into Books(Bid, Title, Author) values (1, 'Searching the lost time', 'Marcel Proust')
insert into Books(Bid, Title, Pages) values (2, 'Searching the lost time', 350)
insert into Books(Bid, Title, ISBN) values (3, 'Searching the lost time', '9789731246420')

select * from Books
```

Results | Messages

| | Bid | Title | Author | ISBN | Pages |
|---|---|---|---|---|---|
| 1 | 1 | Searching the lost time | Marcel Proust | NULL | NULL |
| 2 | 2 | Searching the lost time | NULL | NULL | 350 |
| 3 | 3 | Searching the lost time | NULL | 9789731246420 | NULL |

```
MERGE Books
USING
(SELECT MAX(Bid) Bid, Title, MAX(Author) Author, MAX(ISBN) ISBN, MAX(Pages) Pages
FROM Books
GROUP BY Title) MergeData ON Books.Bid = MergeData.Bid
WHEN MATCHED THEN
```

```
UPDATE SET Books.Title = MergeData.Title,
Books.Author = MergeData.Author,
Books.ISBN = MergeData.ISBN,
Books.Pages = MergeData.Pages
WHEN NOT MATCHED BY SOURCE THEN DELETE;
```

```
Warning: Null value is eliminated by an aggregate or other SET operation.

(3 row(s) affected)
```

```
select * from Books
```

Results | Messages

| | Bid | Title | Author | ISBN | Pages |
|---|---|---|---|---|---|
| 1 | 3 | Searching the lost time | Marcel Proust | 9789731246420 | 350 |