

Laboratorium 4 – MS SQL Server 2008

Temat: Język T-SQL

Opracowanie: A.Dydejczyk, A.Lemański

Ćwiczenie F. Procedury składowane

Procedury przechowywane i wykonywane po stronie serwera, mogą zmieniać dane w bazie, są wydajne, znajdują zastosowanie w np. przy sprawdzaniu uprawnień czy walidacji danych. Umożliwiają kapsułkowanie (jeżeli nie zmienia się interfejs SP to body możemy zmieniać dowolnie, a użytkownik nie zarejestruje zmian). W MS SQL 2005 procedury przechowywane mogą być systemowe, extended i zdefiniowane przez użytkownika. Do tworzenia procedur składowanych używamy polecenia języka SQL - CREATE PROCEDURE (lub CREATE PROC).

```
CREATE { PROC | PROCEDURE } [nazwa_schematu.] nazwa_procedury
    [ { @parametr typ_danych }
    [ WITH <opcje_procedury> [ ,...n ] ]
    [ FOR REPLICATION ]
    AS { <wyrażenie_sql> [;][ ...n ] }
    [;]
    <opcja_procedury> ::=
        [ ENCRYPTION ]
        [ RECOMPILE ]
        [ EXECUTE_AS_Clause ]
    <wyrażenie_sql > ::=
    { [ BEGIN ] wyrażenie [ END ] }
```

W definicji procedury składowanej określamy:

- nazwę procedury;
- nazwy,
- typy danych,
- kierunek działania parametrów procedury;
- ciało procedury – czyli kod wykonywany przez procedurę;
- opcjonalnie deklarujemy, czy procedura ma być przy każdym wykonaniu rekompilowana.

Dodatkowo w ramach zajęć wykorzystamy kursory i wyzwalacze.

Kursory

Kursor to obiekt, który umożliwia poruszanie się po wynikach zapytania rekord po rekordzie. Umożliwia on przetwarzanie rekordów jeden po drugim co daje możliwość zaawansowanego formatowania wyników wyszukiwania danych (ale w ściśle określonej kolejności, determinowanej przez wynik zapytania użytego w definicji kursora).

Implementacja kursora w języku T-SQL wygląda następująco:

```
DECLARE nazwa_kursora CURSOR [ LOCAL | GLOBAL ]  
  
    [ FORWARD_ONLY | SCROLL ]  
    [ STATIC | KEYSET | DYNAMIC | FAST_FORWARD ]  
    [ READ_ONLY | SCROLL_LOCKS | OPTIMISTIC ]  
    [ TYPE_WARNING ]  
    FOR wyrażenie_select [ FOR UPDATE [ OF nazwa_kolumny [ ,...n ] ] ]  
[:]
```

Wyzwalacze DML i DDL

Wyzwalacz (ang. trigger) jest specjalnym rodzajem procedury składowanej. W przeciwieństwie do zwykłej procedury składowanej wyzwalacz nie może zostać jawnie wywołany. Wyzwalacz jest wywoływany w reakcji na określone akcje. Akcje te to wykonanie przez użytkownika określonego polecenia SQL (INSERT, UPDATE, DELETE) na danej tabeli, dla której został określony wyzwalacz.

Tworzenie wyzwalaczy

Wyzwalacze tworzymy używając polecenia CREATE TRIGGER. W definicji wyzwalacza określamy:

- nazwę wyzwalacza;
- dla jakiej tabeli tworzymy wyzwalacz;
- na jakie akcje wyzwalacz będzie reagował;
- jakiego typu wyzwalacz tworzymy;
- ciało wyzwalacza (odpowiednik ciała procedury składowanej) - czyli kod wykonywany przez wyzwalacz.

W SQL Server 2008 istnieją trzy rodzaje wyzwalaczy:

- A. Wyzwalacze obsługujące operacje DML (ang. Data Manipulation Language), czyli INSERT, UPDATE oraz DELETE wykonywane na tabeli lub widoku.

```
CREATE TRIGGER [ nazwa_schematu . ]nazwa_wyzwalacza  
ON { table | view }  
[ WITH <dml_opcje_wyzwalacza> [ ,...n ] ]  
{ FOR | AFTER | INSTEAD OF }  
{ [ INSERT ] [ , ] [ UPDATE ] [ , ] [ DELETE ] }  
[ WITH APPEND ]  
[ NOT FOR REPLICATION ]  
AS { wyrażenie_sql [ ; ] [ ,...n ] }  
<dml_opcje_wyzwalacza> ::=  
    [ ENCRYPTION ]  
    [ EXECUTE AS Clause ]
```

- B. Wyzwalacze obsługujące operacje DDL (ang. Data Definition Language) czyli CREATE, ALTER, DROP oraz pewne procedury składowane, które wykonują operacje DDL.

```
CREATE TRIGGER nazwa_wyzwalacza
ON { ALL SERVER | DATABASE }
[ WITH <ddl_opcje_wyzwalacza> [ ,...n ] ]
{ FOR | AFTER }
{ event_type | event_group } [ ,...n ]
AS { wyrażenie_sql [ ; ] [ ,...n ] }
<ddl_opcje_wyzwalacza> ::=
    [ ENCRYPTION ]
    [ EXECUTE AS Clause ]
```

- C. Wyzwalacze obsługujące zdarzenie logowania (LOGON), który jest wywoływany, kiedy ustalana jest sesja logującego się użytkownika.

```
CREATE TRIGGER nazwa_wyzwalacza
ON ALL SERVER
[ WITH <opcje_wyzwalacza_logon > [ ,...n ] ]
{ FOR | AFTER }
LOGON
AS { wyrażenie_sql [ ; ] [ ,...n ] }
< opcje_wyzwalacza_logon > ::=
    [ ENCRYPTION ]
    [ EXECUTE AS Clause ]
```

1. Procedura składowana i parametry wejściowe.

Procedura zwraca wyniki posortowane po danym parametrze (dodatkowa funkcjonalność – uprawnienia dla użytkownika, który może wykonać procedurę a nie może przeglądać tabeli).

```
--- Skrypt Lab04.26
USE AdventureWorks;
GO
IF OBJECT_ID('dbo.usp_GetSortedPersons') IS NOT NULL
    DROP PROC dbo.usp_GetSortedPersons;
GO
-- Stored procedure usp_GetSortedPersons
-- zwraca posortowaną tabelę wg nazwy kolumny podanej w parametrze
CREATE PROC dbo.usp_GetSortedPersons
    @colname AS sysname = NULL
AS
    DECLARE @msg AS NVARCHAR(500);
    -- Input validation
    IF @colname IS NULL
    BEGIN
        SET @msg = N'A value must be supplied for parameter @colname.';
```

```
RAISERROR(@msg, 16, 1);
RETURN;
END
IF @colname
NOT IN(N'ContactID', N'LastName', N'Phone')
BEGIN
SET @msg = N'Valid values for @colname are: '
+ N'N"ContactID", N"LastName", N"Phone":.';
RAISERROR(@msg, 16, 1);
RETURN;
END
-- Return person sorted by requested sort column
IF @colname = N'ContactID'
SELECT ContactID, LastName, Phone
FROM Person.Contact
ORDER BY ContactID;
ELSE IF @colname = N'LastName'
SELECT ContactID, LastName, Phone
FROM Person.Contact
ORDER BY LastName;
ELSE IF @colname = N'Phone'
SELECT ContactID, LastName, Phone
FROM Person.Contact
ORDER BY Phone;
GO
```

Po poprawnym wprowadzeniu procedury, wykonujemy test procedury.

```
-- test procedury
USE AdventureWorks;
EXEC dbo.usp_GetSortedPersons @colname = N'LastName';
```

W ramach ćwiczenia, można sprawdzić działanie procedur przy różnych uprawnieniach użytkownika. Użytkownik nie ma prawa do przeglądania tabel źródłowych, jednak może wykonać procedurę składowaną przetwarzającą tablicę do której użytkownik nie ma dostępu.

```
-- zadanie z uprawnieniami
DENY SELECT ON Person.Contact TO user1;
GRANT EXECUTE ON dbo.usp_GetSortedPersons TO user1
```

```
-- test z uprawnieniami
SELECT PersonID, LastName, Phone
FROM Person.Contact;
```

```
-- usuwanie obiektów z bazy danych
USE AdventureWorks;
GO
IF OBJECT_ID('dbo.usp_GetSortedPersons') IS NOT NULL
DROP PROC dbo.usp_GetSortedPersons;
```

2. Procedura składowana i parametry wejściowe i wyjściowe.

Parametry wejściowe i wyjściowe w procedurach składowanych

```
--- Skrypt Lab04.27
-- INTERFACE - input and output parameters
USE AdventureWorks;
GO

IF OBJECT_ID('dbo.usp_GetCustOrders') IS NOT NULL
    DROP PROC dbo.usp_GetCustOrders;
GO

CREATE PROC dbo.usp_GetCustOrders
    @custid AS NCHAR(5),
    @fromdate AS DATETIME = '19000101',
    @todate AS DATETIME = '99991231'
AS
    SET NOCOUNT ON;
    SELECT SalesOrderID, CustomerID, SalesPersonID, OrderDate
    FROM Sales.SalesOrderHeader
    WHERE CustomerID = @custid
        AND OrderDate >= @fromdate
        AND OrderDate < @todate;
GO
```

Po poprawnym zarejestrowaniu procedury wykonujemy testy.

```
-- sposoby uzycia
EXEC dbo.usp_GetCustOrders N'676';
EXEC dbo.usp_GetCustOrders N'676', DEFAULT, '20040212';
EXEC dbo.usp_GetCustOrders N'676', '20021010', '20041010';

EXEC dbo.usp_GetCustOrders
    @custid = N'676',
    @fromdate = '19970101',
    @todate = '19980101';
```

Modyfikujemy procedurę dodając parametr wyjściowy.

```
--- Skrypt Lab04.28
-- output parameters
ALTER PROC dbo.usp_GetCustOrders
    @custid AS NCHAR(5),
    @fromdate AS DATETIME = '19000101',
    @todate AS DATETIME = '99991231',
    @numrows AS INT OUTPUT
AS
    SET NOCOUNT ON;
    DECLARE @err AS INT;
```

```
SELECT SalesOrderID, CustomerID, SalesPersonID, OrderDate
FROM Sales.SalesOrderHeader
WHERE CustomerID = @custid
AND OrderDate >= @fromdate
AND OrderDate < @todate;
SELECT @numrows=@@ROWCOUNT,@err=@@ERROR;
RETURN @err;
GO
```

Po rejestracji procedury wykonujemy testy.

```
DECLARE @myerr AS INT, @mynumrows AS INT;
```

```
EXEC @myerr = dbo.usp_GetCustOrders
    @custid = N'676',
    @fromdate = '20021010',
    @todate = '20041010',
    @numrows = @mynumrows OUTPUT;
```

```
SELECT @myerr AS err, @mynumrows AS rc;
```

Przetwarzanie wyników z procedury składowanej z wykorzystaniem tablicy tymczasowej.

--- Skrypt Lab04.29

-- w celu dalszego przetwarzania wyniku musimy uzyc temporary tabel

```
IF OBJECT_ID('tempdb..#CustOrders') IS NOT NULL
```

```
DROP TABLE #CustOrders;
```

```
GO
```

```
CREATE TABLE #CustOrders
```

```
(
```

```
    SalesOrderID INT NOT NULL PRIMARY KEY,
```

```
    CustomerID NCHAR(5) NOT NULL,
```

```
    SalesPersonID INT NOT NULL,
```

```
    OrderDate DATETIME NOT NULL
```

```
);
```

```
DECLARE @myerr AS INT, @mynumrows AS INT;
```

```
INSERT INTO #CustOrders(SalesOrderID, CustomerID, SalesPersonID, OrderDate)
```

```
EXEC @myerr = dbo.usp_GetCustOrders
```

```
    @custid = N'676',
```

```
    @fromdate = '20021010',
```

```
    @todate = '20040101',
```

```
    @numrows = @mynumrows OUTPUT;
```

```
SELECT SalesOrderID, CustomerID, SalesPersonID, OrderDate
FROM #CustOrders;
```

```
SELECT @myerr AS err, @mynumrows AS rc;
```

```
GO
```

Po zakończeniu zadania usuwamy utworzone obiekty w bazie danych.

3. Procedura składowana, obsługa błędów.

Interpretacja błędów podczas realizacji procedur składowanych.

```
--- Skrypt Lab04.30
-- jak kompilowane sa stored procedure
USE tempdb;
GO
IF OBJECT_ID('dbo.usp_Proc1') IS NOT NULL
    DROP PROC dbo.usp_Proc1;
GO
IF OBJECT_ID('dbo.usp_Proc2') IS NOT NULL
    DROP PROC dbo.usp_Proc2;
GO
IF OBJECT_ID('dbo.T1') IS NOT NULL
    DROP TABLE dbo.T1;

--tworzymy stored procedure usp_Proc1 - tabela T1 nie istnieje
CREATE PROC dbo.usp_Proc1
AS
    SELECT col1 FROM dbo.T1;
GO

CREATE PROC dbo.usp_Proc1
AS
    SELECT col1 FROM dbo.T1;
GO

-- test
EXEC dbo.usp_Proc1;
-- wynik błąd wykonania
-- Msg 208, Level 16, State 1, Procedure usp_Proc1, Line 3
-- Invalid object name 'dbo.T1'.

-- tworzymy tabelę T1 z col1
CREATE TABLE dbo.T1(col1 INT);
INSERT INTO dbo.T1(col1) VALUES(1);

-- i ponownie test
EXEC dbo.usp_Proc1;
-- teraz SUKCES

--tworzymy stored procedure usp_Proc2 tabela T1 istnieje, nie istnieje col2
CREATE PROC dbo.usp_Proc2
AS
    SELECT col2 FROM dbo.T1;
GO
```

```
-- wynik - procedura nie kompiluje sie
--Msg 207, Level 16, State 1, Procedure usp_Proc2, Line 3
--Invalid column name 'col2'.
```

Usuwanie utworzone obiekty.

```
USE tempdb;
GO
IF OBJECT_ID('dbo.usp_Proc1') IS NOT NULL
    DROP PROC dbo.usp_Proc1;
GO
IF OBJECT_ID('dbo.usp_Proc2') IS NOT NULL
    DROP PROC dbo.usp_Proc2;
GO
IF OBJECT_ID('dbo.T1') IS NOT NULL
    DROP TABLE dbo.T1;
```

4. Procedury składowana, optymalizacja.

- <https://docs.microsoft.com/en-us/sql/t-sql/statements/set-statistics-io-transact-sql?view=sql-server-2017>

Wykorzystanie narzędzi do optymalizacji zapytań

```
--- Skrypt Lab04.31
USE AdventureWorks;
GO
IF OBJECT_ID('dbo.usp_GetOrders') IS NOT NULL
    DROP PROC dbo.usp_GetOrders;
GO

CREATE PROC dbo.usp_GetOrders
    @odate AS DATETIME
AS
SELECT SalesOrderID, CustomerID, SalesPersonID, OrderDate
FROM Sales.SalesOrderHeader
WHERE OrderDate >= @odate;

GO

--Turn on the STATISTICS IO option - informacja o aktywności sesji
SET STATISTICS IO ON;

-- analizujemy execution plan
EXEC dbo.usp_GetOrders '20011010';
```

5. Wyzwalacze DDL w bazie danych MS SQL Server.

- <https://docs.microsoft.com/en-us/sql/relational-databases/triggers/ddl-triggers?view=sql-server-2017>
- <https://docs.microsoft.com/en-us/sql/t-sql/functions/eventdata-transact-sql?view=sql-server-2017>

Przykład ochrony obiektów bazy danych przed modyfikacją ich struktury (ALTER czy DROP).

```
-- skrypt Lab04.32
USE tempdb
GO
CREATE TABLE ddltest( a int not null) ;
go
CREATE TRIGGER safety on database for drop_table
AS
print 'Musisz wyłaczcyc trigger przed usuniecim tabeli'
ROLLBACK
GO
DROP TABLE ddltest
GO
```

Monitorowanie zmian w strukturze bazy danych z wykorzystaniem wyzwalacza DDL oraz obiektu XML tworzonego przez funkcję eventdata().

```
-- skrypt Lab04.33
CREATE TABLE EventLog
( PostTime datetime,
  DBUser nvarchar(100),
  Event nvarchar(100),
  TSQL nvarchar(2000) )
GO
ALTER TRIGGER safety on database for DDL_DATABASE_LEVEL_EVENTS
AS
DECLARE @data xml
SET @data = eventdata()
INSERT EventLog ( PostTime, DBUser, Event, TSQL ) values (GETDATE(),
CONVERT(nvarchar(100),CURRENT_USER),
CONVERT(nvarchar(100),@data.query('data(//EventType)'),
CONVERT(nvarchar(2000), @data.query('data(//TSQLCommand)'))))
GO
CREATE TABLE Foo (a int)
DROP TABLE Foo
SELECT * FROM EventLog
```

Zadania

Zadanie Z1.

Opracować przykład procedury składowanej z wykorzystaniem kursora tworzący wydruk z dowolnej tabeli w bazie danych AdventureWorks w określonym formacie.

Zadanie Z2.

Opracować przykład wyzwalacza typu DML dla dowolnego obiektu (tabeli lub widoku) w bazie danych AdventureWorks.

Zadanie Z3.

Opracować przykład procedury składowanej z wykorzystaniem struktury RAISERROR przesłania informacji o niemożliwości zrealizowania określonego zadania w bazie danych AdventureWorks.