Процедуры

Некоторые операции, рассматриваемые как неделимые, трудно выразить с помощью одного запроса к БД.

Примеры:

- занести данные о названиях дней недели в таблицу – 7 операций вставки;
- выполнить банковскую проводку с занесением двух операций по дебету и кредиту;

Процедуры

Синтаксис для создания процедур может различаться в различных БД. Вот типичный пример запроса на создание процедуры в MS SQL Server:

```
CREATE PROCEDURE CreateWeek
    @first int = 0
AS
BEGIN
  DROP TABLE Weekdays;
  CREATE TABLE Weekdays(DayNumber INTEGER, DayName VARCHAR(20),
         PRIMARY KEY(DayNumber));
  INSERT INTO WeekDays(DayNumber, DayName) VALUES(@first, 'Monday');
  INSERT INTO WeekDays(DayNumber, DayName) VALUES(@first+1, 'Tuesday');
  INSERT INTO WeekDays(DayNumber, DayName) VALUES(@first+2,
'Wednesday');
  INSERT INTO WeekDays(DayNumber, DayName) VALUES(@first+3, 'Thursday');
  INSERT INTO WeekDays(DayNumber, DayName) VALUES(@first+4, 'Friday');
  INSERT INTO WeekDays(DayNumber, DayName) VALUES(@first+5, 'Saturday');
  INSERT INTO WeekDays(DayNumber, DayName) VALUES(@first+6, 'Sunday');
END
```

JDBC – Создание и вызов процедур

```
// Подключаемся к драйверу базы данных
Connection conn = null;
Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
conn = DriverManager.getConnection(
   "jdbc:sqlserver://127.0.0.1\\SQLAKOUB;databaseName=academy;" +
   "user=sa;password=*****");

// Подготавливаем вызов хранимой процедуры
CallableStatement cst = conn.prepareCall("Exec CreateWeek 1");

// Осуществляем вызов
cst.execute();
```

Процедуры с выходными параметрами

Изменим наш пример таким образом, чтобы процедура формировала некоторое значение в качестве результата:

```
ALTER PROCEDURE CreateWeek
    @first int = 0,
    @maxnum int OUTPUT
AS
BEGIN
  DROP TABLE Weekdays;
  CREATE TABLE Weekdays(DayNumber INTEGER, DayName VARCHAR(20),
         PRIMARY KEY(DayNumber));
  INSERT INTO WeekDays(DayNumber, DayName) VALUES(@first, 'Monday');
  INSERT INTO WeekDays(DayNumber, DayName) VALUES(@first+1, 'Tuesday');
  INSERT INTO WeekDays(DayNumber, DayName) VALUES(@first+2,
'Wednesday');
  INSERT INTO WeekDays(DayNumber, DayName) VALUES(@first+3, 'Thursday');
  INSERT INTO WeekDays(DayNumber, DayName) VALUES(@first+4, 'Friday');
  INSERT INTO WeekDays(DayNumber, DayName) VALUES(@first+5, 'Saturday');
  INSERT INTO WeekDays(DayNumber, DayName) VALUES(@first+6, 'Sunday');
  SET @maxnum = (SELECT Max(DayNumber) FROM Weekdays);
END
```

JDBC – Работа с выходными параметрами

```
// Подготавливаем вызов хранимой процедуры
CallableStatement cst = conn.prepareCall("Exec CreateWeek ?, ?");

// Подготавливаем параметры (входные и выходные)
cst.setInt(1, 0);
cst.registerOutParameter(2, Types.INTEGER);

// Осуществляем вызов
cst.execute();

// Доступ к значению выходного параметра:
System.out.println(cst.getInt(2));
```

JDBC – Работа с именованными параметрами

```
// Подготавливаем вызов хранимой процедуры
CallableStatement cst = conn.prepareCall(
    "Exec CreateWeek @first=?, @maxnum=?");

// Подготавливаем параметры (входные и выходные)
cst.setInt("first", 0);
cst.registerOutParameter("maxnum", Types.INTEGER);

// Осуществляем вызов
cst.execute();

// Доступ к значению выходного параметра:
System.out.println(cst.getInt("maxnum"));
```

Синтаксис вызова может отличаться в разных БД и даже с разными драйверами. Например, в других БД вызов мог бы выглядеть

```
Call CreateWeek(?,?)
{CreateWeek(?,?)}
```

и т.д.

Более сложные процедуры

В разных БД есть синтаксис для исполнения программ, подобный синтаксису обычных языков программирования. Например, в Transact-SQL:

```
CREATE FUNCTION dbo.WeekDay(@day INT) RETURNS VARCHAR(20) AS
BEGIN
  RETURN
    CASE @day
      WHEN 0 THEN 'Monday'
      WHEN 1 THEN 'Tuesday'
      WHEN 2 THEN 'Wednesday'
      WHEN 3 THEN 'Thursday'
      WHEN 4 THEN 'Friday'
      WHEN 5 THEN 'Saturday'
      WHEN 6 THEN 'Sunday'
    END
END
```

Более сложные процедуры

Теперь наша процедура по созданию таблицы в Transact-SQL могла бы выглядеть так:

```
CREATE PROCEDURE CreateWeek
    @first INT = 0
AS
BEGIN
  DECLARE @ndx INT, @ord INT;
  SET @ndx = 0;
  SET @ord = (7 - @first) % 7;
  DROP TABLE Weekdays;
  CREATE TABLE Weekdays(DayOrder INT, DayNumber INT,
        DayName VARCHAR(20), PRIMARY KEY(DayNumber));
  WHILE (@ndx < 7) BEGIN
    INSERT INTO WeekDays(DayOrder, DayNumber, DayName)
                VALUES(@ndx, @ord, dbo.WeekDay(@ndx));
    SET @ndx = @ndx + 1;
    SET @ord = (@ord + 1) % 7;
  END
END
```

Другие объекты в БД

Представления (VIEW): виртуальные таблицы

Новое представление создается с помощью предложения `CREATE VIEW`, например:

```
CREATE VIEW Weekend AS
SELECT * FROM Weekdays WHERE DayOrder >= 5
```

Теперь это представление можно использовать как обычную таблицу, например:

```
SELECT DayName FROM Weekend
```

И даже:

```
INSERT INTO Weekend (DayOrder, DayNumber, DayName)
       VALUES (7, 8, 'Googleday')
```

Обычно представления используют для создания запросов, используемых позже в качестве «подзапросов»:

```
CREATE VIEW MaxPercent AS
SELECT Uid, Max([Percent]) AS MaxPercent FROM pu GROUP BY
Uid
SELECT Name, MaxPercent FROM Users INNER JOIN MaxPercent
       ON Users.Id = MaxPercent.Uid
```

Другие объекты в БД

Функции, возвращающие таблицу

Представления не могут быть параметризованы. Однако, во многих БД (MS SQL Server) существует мощный механизм, позволяющий создавать параметризованные функции, возвращающие таблицу.

Сначала создадим «общую виртуальную таблицу процентов»

```
CREATE VIEW UsersPercent AS
SELECT Uid, Sum([Percent]) AS SumPercent FROM pu GROUP BY Uid

CREATE FUNCTION FreeUsers(@p INT)
RETURNS TABLE
AS
  RETURN
    SELECT Users.*, SumPercent
    FROM Users INNER JOIN UsersPerecent
    ON Users.Id = UsersPercent.Uid
    WHERE SumPercent < @p

SELECT fu.* FROM FreeUsers(80) AS fu
```

Другие объекты в БД

Триггеры: реакция на события внутри БД

```
CREATE TRIGGER trigger_name ON table_view_name
FOR [AFTER | INSTEAD OF] [INSERT][,][UPDATE][,][DELETE]
AS sql_statement
```

- Триггеров на одну таблицу может быть несколько; порядок исполнения можно определить дополнительно;
- Триггеры исполняются после того, как исполнены каскадные действия (`AFTER`) или до исполнения каскадных действий (`INSTEAD OF`);
- Если проверка выявила необходимость отмены действия, то такую отмену можно выполнить непосредственно из тела триггера (`ROLLBACK TRANSACTION`);
- В теле триггера можно использовать псевдо-таблицы `inserted` и `deleted`, содержащие информацию об измененных записях;
- Из тела триггера можно вызывать другие процедуры и функции.

```
CREATE TRIGGER CheckNewDay ON Weekdays
FOR INSERT, UPDATE
AS DECLARE @min INT, @max INT
   SELECT @min = Min(DayOrder), @max = Max(DayOrder) FROM inserted
   IF (@min < 0 OR @max > 6) ROLLBACK TRANSACTION
```