
Лабораторная работа №5

Тема: Использование хранимых процедур и функций для организации сложных выборок данных

Цель работы

Использование хранимых процедур и пользовательских функций в СУБД MS SQL Server для организации сложных выборок данных и инкапсуляции логики обработки данных на основе базы данных `Ugol`.

---

1. Индивидуальное задание

На основе разработанной базы данных `Ugol` необходимо:

1. Создать хранимые процедуры для реализации сложных выборок и операций в соответствии с логикой предметной области (добыча и вывоз угля).
2. Создать пользовательские функции:
    - возвращающую скалярное значение;
    - возвращающую таблицу.
3. Оформить скрипты созданных процедур и функций.

---

2. Скрипты хранимых процедур

2.1. Хранимая процедура `usp_GetDobychaByDate`

Назначение: выводит детализированную информацию о добыче угля за указанную дату: смена, марка угля, её характеристики и объём.

```
USE Ugol;
GO

IF OBJECT_ID('dbo.usp_GetDobychaByDate', 'P') IS NOT NULL
   DROP PROCEDURE dbo.usp_GetDobychaByDate;
GO

CREATE PROCEDURE dbo.usp_GetDobychaByDate
   @Date date
AS
BEGIN
   SET NOCOUNT ON;

   SELECT
      dv.Data,
      dv.Smena,
      dv.MarkaUglia,
      u.Zolnost,
      u.Vlazhnost,
      u.TeplotaSgoraniya,
      dv.Obem
   FROM DobychaVyvoz dv
   JOIN Ugol u ON dv.MarkaUglia = u.MarkaUglia
   WHERE dv.Data = @Date
   ORDER BY dv.Smena, dv.MarkaUglia;
END;
GO

-- Пример вызова:
EXEC dbo.usp_GetDobychaByDate @Date = '2025-01-02';
```

2.2. Хранимая процедура `usp_GetFullReisInfo`

Назначение: возвращает полную информацию по указанному рейсу:
дату, смену, объём, марку угля, данные об экскаваторе, самосвале, экскаваторщике и водителе.

```
USE Ugol;
GO

IF OBJECT_ID('dbo.usp_GetFullReisInfo', 'P') IS NOT NULL
   DROP PROCEDURE dbo.usp_GetFullReisInfo;
```

```
GO

CREATE PROCEDURE dbo.usp_GetFullReisInfo
    @NomerReisa int
AS
BEGIN
    SET NOCOUNT ON;

    SELECT
        dv.NomerReisa,
        dv.Data,
        dv.Smena,
        dv.Obem,
        dv.MarkaUglia,
        u.Zolnost,
        u.Vlazhnost,
        u.TeplotaSgoraniya,
        e.KodEkskavatora,
        e.Nazvanie      AS NazvanieEkskavatora,
        e.ObemKovsha,
        de.TabNomer     AS TabNomerEkskavatorshik,
        r1.FIO          AS FIOEkskavatorshik,
        s.KodSamosvala,
        s.Nazvanie      AS NazvanieSamosvala,
        s.Tonnazh,
        vs.TabNomer     AS TabNomerVoditel,
        r2.FIO          AS FIOVoditel
    FROM DobychaVyvoz dv
    LEFT JOIN Ugol u ON dv.MarkaUglia = u.MarkaUglia
    LEFT JOIN DobychaEkskavator de ON dv.NomerReisa = de.NomerReisa
    LEFT JOIN Ekskavatory e ON de.KodEkskavatora = e.KodEkskavatora
    LEFT JOIN Rabotniki r1 ON de.TabNomer = r1.TabNomer
    LEFT JOIN VyvozSamosval vs ON dv.NomerReisa = vs.NomerReisa
    LEFT JOIN Samosvaly s ON vs.KodSamosvala = s.KodSamosvala
    LEFT JOIN Rabotniki r2 ON vs.TabNomer = r2.TabNomer
    WHERE dv.NomerReisa = @NomerReisa;
END;
GO

-- Пример вызова:
EXEC dbo.usp_GetFullReisInfo @NomerReisa = 10;
```

2.3. Хранимая процедура `usp_InsertFullReis`

Назначение: выполняет вставку нового рейса в таблицы `DobychaVyvoz`, `DobychaEkskavator` и `VyvozSamosval` в одной транзакции.
Возвращает код результата через выходной параметр `@ReturnCode` (1 — успех, иначе код ошибки).

```
USE Ugol;
GO

IF OBJECT_ID('dbo.usp_InsertFullReis', 'P') IS NOT NULL
   DROP PROCEDURE dbo.usp_InsertFullReis;
GO

CREATE PROCEDURE dbo.usp_InsertFullReis
   @NomerReisa        int,
   @Data              date,
   @Smena             int,
   @Obem              decimal(10,2),
   @MarkaUglia        varchar(50),
   @KodEkskavatora    int,
   @TabNomerEks       int,
   @KodSamosvala      int,
   @TabNomerVoditel   int,
   @ReturnCode        int OUTPUT
AS
BEGIN
   SET NOCOUNT ON;
   SET @ReturnCode = 0;

   BEGIN TRY
      BEGIN TRAN;

      INSERT INTO DobychaVyvoz (NomerReisa, Data, Smena, Obem, MarkaUglia)
      VALUES (@NomerReisa, @Data, @Smena, @Obem, @MarkaUglia);

      INSERT INTO DobychaEkskavator (NomerReisa, KodEkskavatora, TabNomer)
      VALUES (@NomerReisa, @KodEkskavatora, @TabNomerEks);

      INSERT INTO VyvozSamosval (NomerReisa, KodSamosvala, TabNomer)
      VALUES (@NomerReisa, @KodSamosvala, @TabNomerVoditel);

      COMMIT TRAN;
      SET @ReturnCode = 1; -- успешное выполнение
```

```
    END TRY
    BEGIN CATCH
        IF @@TRANCOUNT > 0
            ROLLBACK TRAN;
        SET @ReturnCode = ERROR_NUMBER();
    END CATCH;
END;
GO

-- Пример вызова:
DECLARE @rc int;
EXEC dbo.usp_InsertFullReis
    @NomerReisa = 21,
    @Data = '2025-01-07',
    @Smena = 1,
    @Obem = 150,
    @MarkaUglia = 'A',
    @KodEkskavatora = 301,
    @TabNomerEks = 1,
    @KodSamosvala = 201,
    @TabNomerVoditel = 2,
    @ReturnCode = @rc OUTPUT;
SELECT @rc AS ReturnCode;
```

---

3. Скрипты пользовательских функций

3.1. Скалярная функция `fn_GetTotalObemByMarka`

Назначение: возвращает суммарный объём добычи по заданной марке угля.

```
USE Ugol;
GO

IF OBJECT_ID('dbo.fn_GetTotalObemByMarka', 'FN') IS NOT NULL
    DROP FUNCTION dbo.fn_GetTotalObemByMarka;
GO

CREATE FUNCTION dbo.fn_GetTotalObemByMarka
(
    @MarkaUglia varchar(50)
)
```

```
RETURNS decimal(18,2)
AS
BEGIN
    DECLARE @Result decimal(18,2);

    SELECT @Result = SUM(Obem)
    FROM DobychaVyvoz
    WHERE MarkaUglia = @MarkaUglia;

    RETURN ISNULL(@Result, 0);
END;
GO

-- Примеры использования:
SELECT dbo.fn_GetTotalObemByMarka('A') AS TotalObemA;
SELECT dbo.fn_GetTotalObemByMarka('B') AS TotalObemB;
```

3.2. Табличная функция `fn_GetReisyByDateAndSmena`

Назначение: возвращает таблицу рейсов за указанную дату и смену.
Если `@Smena` передана как `NULL`, возвращаются все смены за указанную дату.

```
USE Ugol;
GO

IF OBJECT_ID('dbo.fn_GetReisyByDateAndSmena', 'IF') IS NOT NULL
    DROP FUNCTION dbo.fn_GetReisyByDateAndSmena;
GO

CREATE FUNCTION dbo.fn_GetReisyByDateAndSmena
(
    @Date  date,
    @Smena int
)
RETURNS TABLE
AS
RETURN
(
    SELECT
        dv.NomerReisa,
        dv.Data,
        dv.Smena,
```

```
    dv.MarkaUglia,
    dv.Obem
  FROM DobychaVyvoz dv
  WHERE dv.Data = @Date
    AND (@Smena IS NULL OR dv.Smena = @Smena)
);
GO

-- Примеры использования:
SELECT * FROM dbo.fn_GetReisyByDateAndSmena('2025-01-02', 1);
SELECT * FROM dbo.fn_GetReisyByDateAndSmena('2025-01-02', NULL);
```

---

4. Ответы на контрольные вопросы

1. Назначение хранимых процедур

Хранимые процедуры нужны для инкапсуляции логики работы с данными на стороне сервера. Они позволяют:

- выполнять сложные последовательности операторов T-SQL как единый модуль;
- уменьшать сетевой трафик (вместо множества запросов вызывается одна процедура);
- повторно использовать код;
- повысить безопасность (работать через процедуры вместо прямого доступа к таблицам);
- реализовывать проверки, транзакции, обработку ошибок и управление ходом выполнения.

---

2. Классификация хранимых процедур

Хранимые процедуры классифицируют:

- по источнику:

    - системные (предоставляются SQL Server, обычно с префиксом `sp_`);
    - пользовательские (создаются разработчиком под конкретные задачи).
- по месту выполнения:
    - обычные T-SQL-процедуры;
    - расширенные процедуры (`xp_`), реализованные во внешних DLL;
- по назначению:
    - административные и сервисные;
    - прикладные (отчёты, бизнес-логика, интеграция и т.п.).

---

3. Как можно осуществлять управление ходом процедуры?

Управление ходом выполнения в хранимых процедурах выполняется с помощью:

- условных операторов `IF...ELSE`, `CASE`;
- циклов `WHILE` (с операторами `BREAK`, `CONTINUE`);
- оператора `GOTO` и меток;
- оператора `WAITFOR` (задержка или ожидание времени);
- блоков `TRY...CATCH` для обработки ошибок;
- оператора `RETURN` для немедленного выхода с кодом возврата.

---

4. Способы создания хранимых процедур

Хранимые процедуры создаются:

- оператором `CREATE PROCEDURE` / `CREATE PROC` в T-SQL;

- через SQL Server Management Studio (SSMS) в Object Explorer:
  `Programmability → Stored Procedures → New Stored Procedure`;
- с помощью мастеров (wizard) в SSMS.

Изменение — `ALTER PROCEDURE`, удаление — `DROP PROCEDURE`.

---

5. Способы передачи и возврата параметров в хранимую процедуру

В процедурах используются:

- входные параметры — задаются в объявлении и передаются при вызове;
- выходные параметры (`OUTPUT`) — возвращают значения в вызывающий код;
- код возврата (`RETURN int`) — целое значение (например, код статуса);
- выборки данных через `SELECT` — возвращают набор строк.

Параметрам можно задавать значения по умолчанию, а при вызове — передавать по позиции или по имени.

---

6. В чем заключается управление хранимыми процедурами?

Управление хранимыми процедурами включает:

- создание (`CREATE PROCEDURE`);
- изменение (`ALTER PROCEDURE`) без потери прав;
- удаление (`DROP PROCEDURE`);
- переименование (`sp_rename`);
- выдачу/отзыв прав (`GRANT`, `REVOKE`, `DENY`);

- просмотр текста и свойств (SSMS, системные представления `sys.objects`, `sys.sql_modules`).

Выполнение — через `EXEC` / `EXECUTE`.

---

7. Назначение механизма пользовательских функций

Пользовательские функции позволяют упаковать логику вычислений и переиспользовать её в запросах:

- возвращают одно значение (скалярные) или таблицу (табличные функции);
- могут использоваться в `SELECT`, `WHERE`, `JOIN`, ограничениях и вычисляемых столбцах;
- делают код запросов компактнее и чище;
- стандартизируют повторяющиеся вычисления и бизнес-правила.

---

8. Способы создания пользовательских функций

Пользовательские функции создаются:

- через оператор `CREATE FUNCTION` в T-SQL:
    - скалярные: `RETURNS тип`;
    - табличные: `RETURNS TABLE` (inline) или `RETURNS @tbl TABLE (...)` (многооператорные);
- через SSMS: `Programmability → Functions → New → Scalar/Table-valued Function`.

Изменение — `ALTER FUNCTION`, удаление — `DROP FUNCTION`.
