НАО «КАРАГАНДИНСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ИМЕНИ АБЫЛКАСА САГИНОВА»

Кафедра информационных технологий и безопасности

**Лабораторная работа №5**

**Дисциплина:** "Введение в базы данных"

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

**Принял:**

Преподаватель: Жакина М.М

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(подпись)      (дата)*

**Выполнил:**

ст.гр.СИБ-23-7  Аксеной А.И

Караганда 2025

**Лабораторная работа №** **5**

**Цель:** Использование хранимых процедур и функций для организации сложных выборок данных

**Задачи:**

а) Создать хранимые процедуры в соответствии с требованиями индивидуального варианта задания на разработку курсового проекта.

б) Создать пользовательские функции (*назначение выбирается по своему усмотрению*):

* возвращающие скалярные значения,
* возвращающие таблицу.

**Мое ИД:**

1. На каких дисках находится *i-ый фильм*
2. Все фильмы, в названии которых есть слово «…»
3. Список клиентов, задержавших диск на текущую дату
4. Общее количество фильмов у каждого режиссёра
5. Какие жанры интересовали пользователей в текущем месяце

**Ход работы:**

**a)**

USE Laba5;

GO

-- На каких дисках находится i-ый фильм

CREATE OR ALTER PROCEDURE sp\_GetDisksByFilm

@FilmID INT

AS

BEGIN

SELECT

f.FilmName,

d.DiskID,

d.DepositPrice,

d.RentPerDay,

d.AvailableCopies

FROM DiskContents dc

JOIN Films f ON dc.FilmID = f.FilmID

JOIN Disks d ON dc.DiskID = d.DiskID

WHERE f.FilmID = @FilmID;

END;

GO

-- Все фильмы, в названии которых есть заданное слово

CREATE OR ALTER PROCEDURE sp\_FindFilmsByWord

@SearchWord NVARCHAR(100)

AS

BEGIN

SELECT

FilmID,

FilmName,

ReleaseDate,

Description

FROM Films

WHERE FilmName LIKE N'%' + @SearchWord + N'%';

END;

GO

-- Список клиентов, задержавших диск на текущую дату

CREATE OR ALTER PROCEDURE sp\_GetOverdueClients

AS

BEGIN

SELECT

c.FullName AS ClientName,

m.DiskID,

m.IssueDate,

DATEADD(DAY, m.RentDays, m.IssueDate) AS ExpectedReturn,

ISNULL(m.ReturnDate, '—') AS ReturnDate

FROM Movements m

JOIN Clients c ON m.ClientID = c.ClientID

WHERE m.ReturnDate IS NULL

AND DATEADD(DAY, m.RentDays, m.IssueDate) < GETDATE();

END;

GO

-- Общее количество фильмов у каждого режиссёра

CREATE OR ALTER PROCEDURE sp\_CountFilmsByDirector

AS

BEGIN

SELECT

d.FullName AS DirectorName,

COUNT(f.FilmID) AS TotalFilms

FROM Directors d

LEFT JOIN Films f ON d.DirectorID = f.DirectorID

GROUP BY d.FullName;

END;

GO

-- Жанры фильмов, которые интересовали пользователей в текущем месяце

CREATE OR ALTER PROCEDURE sp\_GenresRentedThisMonth

AS

BEGIN

SELECT

g.GenreName,

COUNT(DISTINCT m.MovementID) AS RentalsCount

FROM Movements m

JOIN DiskContents dc ON m.DiskID = dc.DiskID

JOIN Films f ON dc.FilmID = f.FilmID

JOIN Genres g ON f.GenreID = g.GenreID

WHERE MONTH(m.IssueDate) = MONTH(GETDATE())

AND YEAR(m.IssueDate) = YEAR(GETDATE())

GROUP BY g.GenreName

ORDER BY RentalsCount DESC;

END;

GO

**Вызовы:**

-- На каких дисках находится i-ый фильм

EXEC sp\_GetDisksByFilm @FilmID = 1;

-- Все фильмы, в названии которых есть заданное слово

EXEC sp\_FindFilmsByWord @SearchWord = N'Билл';

-- Список клиентов, задержавших диск на текущую дату

EXEC sp\_GetOverdueClients;

-- Общее количество фильмов у каждого режиссёра

EXEC sp\_CountFilmsByDirector;

-- Жанры фильмов, которые интересовали пользователей в текущем месяце

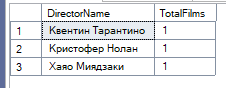
EXEC sp\_GenresRentedThisMonth;

**Результаты:**











**б)**CREATE OR ALTER FUNCTION fn\_GetTotalDeposit(@DiskID INT)

RETURNS DECIMAL(10,2)

AS

BEGIN

DECLARE @Total DECIMAL(10,2);

SELECT @Total = DepositPrice \* AvailableCopies

FROM Disks

WHERE DiskID = @DiskID;

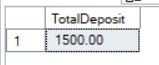
RETURN ISNULL(@Total, 0);

END;

GO

**Проверка:**

SELECT dbo.fn\_GetTotalDeposit(1) AS TotalDeposit;



CREATE OR ALTER FUNCTION fn\_GetFilmsByGenre(@GenreID INT)

RETURNS TABLE

AS

RETURN

(

SELECT

f.FilmID,

f.FilmName,

f.ReleaseDate,

f.Duration,

g.GenreName,

d.FullName AS DirectorFullName,

c.CountryName

FROM Films f

JOIN Genres g ON f.GenreID = g.GenreID

JOIN Directors d ON f.DirectorID = d.DirectorID

JOIN Countries c ON f.CountryID = c.CountryID

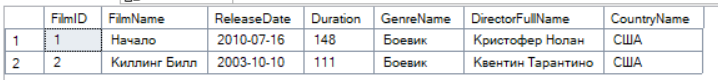
WHERE f.GenreID = @GenreID

);

GO

**Проверка:**

SELECT \* FROM dbo.fn\_GetFilmsByGenre(1);



**Контрольные вопросы**:

**1. Назначение хранимых процедур?**

Это заранее сохраненные SQL-команды, которые выполняют определенные действия с БД. Имеет цели: Автоматизации типовых операций, безопасности, оптимизации и структурности.  
Как пример: EXEC sp\_GetDisksByFilm @FilmID = 2;

**2. Классификация хранимых процедур?**

Системные(sp\_detach\_db), пользовательские(CREATE PROCEDURE sp\_GetDisksByFilm @FilmID INT AS), локальные(sp\_GetDirectorFilmCount), глобальные(доступ из любой базы на сервере, редкое использование, в основном администрирование), вложенные(одна процедура вызывает другую процедуру

CREATE PROCEDURE sp\_Main

AS

BEGIN

EXEC sp\_GetOverdueClients;

END;).

**3. Как можно осуществлять управление ходом процедуры?**

Выполнять условия, повторять действия, обрабатывать ошибки, завершать выполнение в нужный момент.

Условные операторы(IF-ELSE):

IF EXISTS (SELECT \* FROM Clients WHERE Phone IS NULL)

PRINT 'Есть клиенты без телефона';

ELSE

PRINT 'Все клиенты указали телефон';

Циклы(WHILE):

DECLARE @i INT = 1;

WHILE @i <= 3

BEGIN

INSERT INTO Genres (GenreName) VALUES (N'Жанр ' + CAST(@i AS NVARCHAR(10)));

SET @i += 1;

END;

Управление потоком(RETURN):

IF @FilmID IS NULL

BEGIN

PRINT 'Не указан код фильма';

RETURN;

END;

Обработка ошибок(TRY, CATCH):

BEGIN TRY

INSERT INTO Disks (DepositPrice, RentPricePerDay) VALUES (NULL, 100);

END TRY

BEGIN CATCH

PRINT 'Ошибка: ' + ERROR\_MESSAGE();

END CATCH;

Передача параметров:

CREATE PROCEDURE sp\_GetFilmsByGenre @GenreID INT AS

SELECT \* FROM Films WHERE GenreID = @GenreID;

**4. Способы создания хранимых процедур?**

Команда создания новой(CREATE PROCEDURE):

CREATE PROCEDURE sp\_GetDisksByFilm

@FilmID INT

AS

BEGIN

SELECT d.DiskID, d.DepositPrice, d.RentPricePerDay

FROM Disks d

JOIN DiskContent dc ON d.DiskID = dc.DiskID

WHERE dc.FilmID = @FilmID;

END;

GO

Команда изменения уже существующей(ALTER PROCEDURE):

ALTER PROCEDURE sp\_GetDisksByFilm

@FilmID INT

AS

BEGIN

SELECT d.DiskID, d.DepositPrice, d.TotalCopies, d.AvailableCopies

FROM Disks d

JOIN DiskContent dc ON d.DiskID = dc.DiskID

WHERE dc.FilmID = @FilmID;

END;

GO

Команда(CREATE OR ALTER PROCEDURE): Автоматически создает процедуру, если ее нет и изменяет, если она существует.

UI интерфейс:

Клик на БД, процедуры, пкм, новая процедура и нажимаем Execute.

**5. Способы передачи и возврата параметров в хранимую процедуру?**

**Передача входных параметров:**CREATE OR ALTER PROCEDURE sp\_GetFilmsByGenre

@GenreID INT

AS

BEGIN

SELECT FilmName, ReleaseDate

FROM Films

WHERE GenreID = @GenreID;

END;

GO

EXEC sp\_GetFilmsByGenre @GenreID = 2;

**Возврат через** SELECT**:**

CREATE OR ALTER PROCEDURE sp\_GetOverdueClients

AS

BEGIN

SELECT c.FullName, m.DiskID, m.IssueDate

FROM Movements m

JOIN Clients c ON m.ClientID = c.ClientID

WHERE m.ReturnDate IS NULL;

END;

GO

EXEC sp\_GetOverdueClients;

**Возврат скалярного значения через** RETURN**:**

CREATE OR ALTER PROCEDURE sp\_CheckFilmExistence

@FilmID INT

AS

BEGIN

IF EXISTS (SELECT 1 FROM Films WHERE FilmID = @FilmID)

RETURN 0; -- найден

ELSE

RETURN 1; -- нет такого фильма

END;

GO

DECLARE @Result INT;

EXEC @Result = sp\_CheckFilmExistence @FilmID = 3;

PRINT 'Результат: ' + CAST(@Result AS NVARCHAR);

**Возврат значений через выходные параметры OUTPUT:**

CREATE OR ALTER PROCEDURE sp\_GetFilmCountByGenre

@GenreID INT,

@FilmCount INT OUTPUT

AS

BEGIN

SELECT @FilmCount = COUNT(\*) FROM Films WHERE GenreID = @GenreID;

END;

GO

DECLARE @Count INT;

EXEC sp\_GetFilmCountByGenre @GenreID = 1, @FilmCount = @Count OUTPUT;

PRINT 'Количество фильмов: ' + CAST(@Count AS NVARCHAR);

**6. В чем заключается управление хранимыми процедурами?**

Это комплекс действий по их созданию, изменению, выполнению, удалению и контролю выполнения. По сути, это все то, что позволяет руководить их жизненным циклом. Все что в 4 вопросе(создания), удаление (DROP PROCEDURE sp\_GetFilmsByGenre;), выполнение(EXEC sp\_GetFilmsByGenre @GenreID = 2;), просмотр и анализ(SELECT \* FROM sys.procedures;), управление выполнением(IF-ELSE).

**7. Назначение механизма пользовательских функций?**

UDF (User-Defined Functions) это объекты БД, которые позволяют выполнять повторяющиеся вычисления, проверки и выборки данных без дублирования кода в запросах и процедурах. Нужно по сути для повторного использования кода, улучшению читаемости, поддержка целостности(проверка корректности данных при вставке или обновлении), формирование отчетов.

CREATE FUNCTION fn\_GetTotalDeposit(@DiskID INT)

RETURNS DECIMAL(10, 2)

AS

BEGIN

DECLARE @Total DECIMAL(10, 2);

SELECT @Total = DepositPrice \* AvailableCount

FROM Disks

WHERE DiskID = @DiskID;

RETURN @Total;

END;

**8. Способы создания пользовательских функций**

Создаются командой(CREATE FUNCTION), и по сути делятся только в возвращении значений. Скалярные функции(одно значение), табличные(вернут таблицу или набор строк), однооператорные(без begin end, делаются проще и быстрее), функция изменения существующей функции(ALTER FUNCTION), функция удаления функции(DROP FUNCTION).