МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное  
образовательное учреждение высшего образования  
«Самарский национальный исследовательский университет  
имени академика С.П. Королева»  
(Самарский университет)   
  
  
Факультет информатики  
Кафедра программных систем  
  
Дисциплина  
**Базы данных  
  
  
  
ОТЧЕТ**по лабораторной работе №3  
  
Вариант №12

Студенты: Гижевская В.Д.

Петрова А.А.  
Группа: 6313-020302D  
  
Преподаватель: Попова-Коварцева Д.А.  
  
Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Самара 2021

**Лабораторная работа №3**

Тема работы: создание и использование хранимых процедур и функций.

Цель работы: знакомство с возможностями СУБД по созданию хранимых процедур и функций.

Подготовка к работе:

1. Изучите теоретические основы, команды создания хранимых процедур, функций и пакетов, рассмотренные в лекционном курсе, методических указаниях или других источниках.
2. Ответьте на вопросы к лабораторной работе.
3. Подготовьте отчет к оформлению хода выполнения работы.

Порядок выполнения работы:

1. Создайте хранимые процедуры:
   1. без параметров, с использованием агрегатных функций;
   2. с входным параметром/параметрами;
   3. с входными и выходными параметрами.
2. Создайте хранимые процедуры для добавления и изменения записей в таблицах, проверяющих существование редактируемой записи и определяющие значение суррогатного первичного ключа по умолчанию для новых записей.
3. Внесите такие изменения в хранимые процедуры добавления и изменения записей в таблицах, которые не позволят добавить или изменить записи с дублирующими названиями.
4. Создайте хранимые процедуры удаления записей из таблиц, удаляющих связанные записи из дочерних таблиц.
5. Проверьте работоспособность всех созданных хранимых процедур.
6. Создайте функцию:
   1. возвращающую скалярное значение (тип функции Scalar);
   2. возвращающую набор данных Table (тип функции Inline);
   3. многооператорную функцию, возвращающую таблицу (тип функции Multi-statement).
7. Проверьте работоспособность всех созданных функций.
8. Оформите отчет о выполнении лабораторной работы.
9. **Хранимые процедуры:**
10. Без параметров, с использованием агрегатных функций:

Процедура WithoutParam без параметров выдает информацию о дате происшествия и его типе

USE [Registration of incident]

GO

CREATE PROCEDURE WithoutParam AS

BEGIN

SELECT Name\_type AS Тип\_происшествия, Date AS Дата\_происшествия

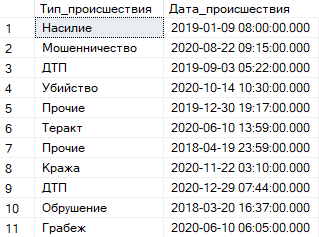
FROM Incident

INNER JOIN Type\_of\_incident ON Type\_of\_incident.ID\_type=Incident.ID\_type

END

GO

EXEC WithoutParam



1. С входным параметром/параметрами:

Процедура WithParam с входным параметром @number\_convictions выдает упорядоченный по количеству судимостей список участников, имеющих меньше двух судимостей

USE [Registration of incident]

GO

CREATE PROCEDURE WithParam(@number\_convictions INT) AS

BEGIN

SELECT Name AS Участник\_происшествия, Number\_of\_convictions AS Количество\_судимостей

FROM Participant\_in\_incident

WHERE @number\_convictions > Number\_of\_convictions

ORDER BY Number\_of\_convictions

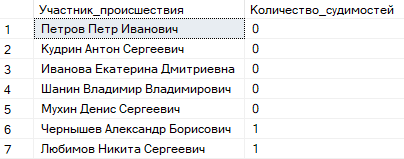
END

GO

DECLARE @number\_convictions INT

SET @number\_convictions = 2

EXEC WithParam @number\_convictions



1. С входными и выходными параметрами:

Процедура InOutParams с входными параметрами @date1 и @date2 и выходным параметром @count\_incident выдает число происшествий, осуществляемых в период между двумя значениями входных параметров

USE [Registration of incident]

GO

CREATE PROCEDURE InOutParams(@count\_incident INT OUTPUT, @date1 DATETIME, @date2 DATETIME) AS

BEGIN

SELECT @count\_incident = COUNT(ID\_incident)

FROM Incident

WHERE (Date BETWEEN @date1 AND @date2)

END

GO

DECLARE @date1 DATETIME, @date2 DATETIME, @count\_incident VARCHAR(30)

SET @date1 = '2019-01-01'

SET @date2 = '2020-31-12'

EXEC InOutParams @count\_incident OUTPUT, @date1, @date2

PRINT 'Количество происшествий за данный период: ' + @count\_incident



1. **Хранимые процедуры для добавления и изменения записей в таблицах, проверяющих существование редактируемой записи и определяющие значение суррогатного первичного ключа по умолчанию для новых записей:**
2. Добавление записи в таблицу

Добавление нового типа происшествия с определением значения суррогатного первичного ключа по умолчанию

USE [Registration of incident]

GO

CREATE PROCEDURE AddData(@type\_name VARCHAR(20)) AS

BEGIN

INSERT INTO Type\_of\_incident

VALUES((SELECT MAX(ID\_type) FROM Type\_of\_incident)+1, @type\_name)

END

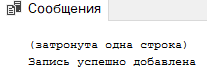
GO

DECLARE @type\_name VARCHAR(20)

SET @type\_name = 'Пожар'

EXEC AddData @type\_name

PRINT 'Запись успешно добавлена'



1. Изменение записи в таблице с проверкой существования данной записи:

Изменение данных о типе происшествия с проверкой на существование редактируемой записи

USE [Registration of incident]

GO

CREATE PROCEDURE UpdateData(@type\_name VARCHAR(20), @type\_id INT) AS

IF (@type\_id=(SELECT ID\_type FROM Type\_of\_incident WHERE ID\_type=@type\_id))

BEGIN

UPDATE Type\_of\_incident

SET Name\_type=@type\_name

WHERE ID\_type=@type\_id

PRINT 'Запись успешно изменена'

END

ELSE

PRINT 'Данной записи не существует, вы не можете ее изменить'

GO

DECLARE @type\_name VARCHAR(20), @type\_id INT

SET @type\_name='Нападение'

SET @type\_id=22

EXEC UpdateData @type\_name, @type\_id



1. **Внесите такие изменения в хранимые процедуры добавления и изменения записей в таблицах, которые не позволят добавить или изменить записи с дублирующими названиями:**
2. Изменение записи с проверкой дублирующихся данных:

Изменение данных о типе происшествия

USE [Registration of incident]

GO

CREATE PROCEDURE UpdateDublData(@type\_name VARCHAR(20), @type\_id INT) AS

IF (@type\_id=(SELECT ID\_type FROM Type\_of\_incident WHERE ID\_type=@type\_id))

BEGIN

IF (@type\_name=(SELECT Name\_type FROM Type\_of\_incident WHERE Name\_type=@type\_name))

PRINT 'Запись с таким именем уже существует'

ELSE

BEGIN

UPDATE Type\_of\_incident

SET Name\_type=@type\_name

WHERE ID\_type=@type\_id

PRINT 'Запись успешно изменена'

END

END

ELSE

PRINT 'Данной записи не существует, вы не можете ее изменить'

GO

DECLARE @type\_name VARCHAR(20), @type\_id INT

SET @type\_name='Убийство'

SET @type\_id=12

EXEC UpdateDublData @type\_name, @type\_id

****

1. Добавление записи с проверкой дублирующихся данных:

Добавление нового типа происшествия

USE [Registration of incident]

GO

CREATE PROCEDURE AddDublData(@type\_name VARCHAR(20)) AS

IF (@type\_name=(SELECT Name\_type FROM Type\_of\_incident WHERE Name\_type=@type\_name))

PRINT 'Запись с таким именем уже существует'

ELSE

BEGIN

INSERT INTO Type\_of\_incident

VALUES((SELECT MAX(ID\_type) FROM Type\_of\_incident)+1, @type\_name)

PRINT 'Запись успешно изменена'

END

GO

DECLARE @type\_name VARCHAR(20)

SET @type\_name='Убийство'

EXEC AddDublData @type\_name

****

1. **Создайте хранимые процедуры удаления записей из таблиц, удаляющих связанные записи из дочерних таблиц.**

Удаление записи с type\_id, равным 11, из таблицы Type\_of\_incident и связанной записи из дочерней таблицы Incident

USE [Registration of incident]

GO

CREATE PROCEDURE DeleteData (@type\_id INT) AS

IF (@type\_id=(SELECT ID\_type FROM Type\_of\_incident WHERE ID\_type=@type\_id))

BEGIN

DELETE

FROM Type\_of\_incident

WHERE ID\_type=@type\_id

DELETE

FROM Incident

WHERE ID\_type=@type\_id

PRINT 'Запись и связанные записи из дочерних таблиц успешно удалены'

END

ELSE

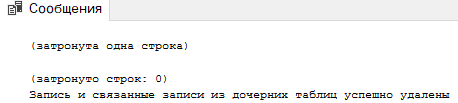
PRINT 'Данной записи не существует, вы не можете ее удалить'

GO

DECLARE @type\_id INT

SET @type\_id = 11

EXEC DeleteData @type\_id

****

1. **Создайте функцию:**
   1. Возвращающую скалярное значение (тип функции Scalar);

Получение количества участников происшествий из города Самара

USE [Registration of incident]

GO

CREATE FUNCTION CountParticipants()

RETURNS INT

AS

BEGIN

DECLARE @count int

SET @count = (SELECT COUNT(Address) FROM Participant\_in\_incident WHERE Address LIKE '%Самара%')

RETURN (@count)

END

GO

PRINT 'Количество участников происшествий из Самары:'

PRINT dbo.CountParticipants()



* 1. Возвращающую набор данных Table (тип функции Inline):

Получение в виде таблицы информации о количестве происшествий по каждому возможному статусу решения

USE [Registration of incident]

GO

CREATE FUNCTION DecisionInfo()

RETURNS TABLE

AS

RETURN (SELECT Status\_decision AS Статус\_решения, COUNT(ID\_incident) AS Количество\_происшествий

FROM Incident

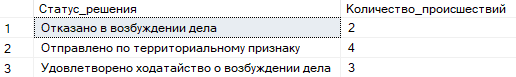
INNER JOIN Incident\_decision

ON Incident\_decision.ID\_decision=Incident.ID\_decision

GROUP BY Status\_decision)

GO

SELECT \* FROM DecisionInfo()



* 1. Многооператорную функцию, возвращающую таблицу (тип функции Multi-statement):

Получение в виде таблицы информации по каждому происшествию

USE [Registration of incident]

GO

CREATE FUNCTION IncidentInfo()

RETURNS @Incidents TABLE (date DATETIME, name\_type VARCHAR(20), count\_participants INT)

AS

BEGIN

INSERT INTO @Incidents(date, name\_type, count\_participants)

(SELECT Date, Name\_type, COUNT(DISTINCT ID\_participant)

FROM Incident

INNER JOIN Participant\_incident

ON Participant\_incident.ID\_incident=Incident.ID\_incident

INNER JOIN Type\_of\_incident

ON Type\_of\_incident.ID\_type=Incident.ID\_type

GROUP BY Date, Name\_type)

RETURN

END

GO

SELECT \* FROM IncidentInfo()

