Отчет по лабораторным работам студентки 2 курса 11 группы Дрожжа Анастасии

*Лабораторная работа №1*

Знакомство с инструментами СУБД Microsoft SQL Server 2012

|  |
| --- |
| SELECT Номер\_заказа, Дата\_поставки, Заказчик  FROM Заказы  WHERE (DAY(Дата\_поставки) > 15)  ------2------  SELECT Наименование, Цена  FROM Товары  WHERE (Цена < 19) AND (Цена > 7)  ------3------  SELECT Заказчик, Дата\_поставки, Наименование\_товара  FROM Заказы  WHERE (Заказчик = N'Лола и Стич')  ORDER BY Дата\_поставки  ------4------  SELECT Наименование\_товара, Заказчик  FROM Заказы  WHERE (Наименование\_товара = N'Безе')  ------5------  SELECT Наименование, Цена  FROM Товары  WHERE (Цена < 19) AND (Цена > 7) |
|  |

*Лабораторная работа №2*

Проектирование баз данных. Нормализация

При проектировании реляционной базы данных необходимо исследовать предметную область с целью определения объектов, нормализовать данные и установить связи между ними. **Нормализация** данных − это процесс, в результате выполнения которого таблицы базы данных проверяются на наличие зависимостей между столбцами и, если необходимо, то таблица разделяется на несколько таблиц.

|  |
| --- |
|  |

*Лабораторная работа №3*

T-SQL − язык реляционной базы данных

Язык реляционной базы данных в **SQL Server** называется **Transact-SQL** (**T-SQL**). Операторы языка делятся на несколько групп, основными из которых является язык определения данных (**Data Definition Language**, **DDL**) и язык манипулирования данными (**Data Manipulation Language**, **DML**).

Язык **DDL** содержит три обобщенных оператора: **create** *объект* (создание объекта базы данных), **alter** *объект* (изменение характеристик объекта) и **drop** *объект* (удаление существующего объекта). Эти опера­торы создают, изменяют и удаляют объекты базы данных, такие как сама база данных, таблицы, столбцы и индексы.

Язык **DML** содержит операторы, которые манипулируют данными, осуществляя выборку информации (**SELECT**), добавление (**INSERT**), удаление (**DELETE**) и изменение (**UPDATE**). При записи операторов можно использовать на клавиатуре любой регистр.

|  |
| --- |
| Use master;  create database drozhzha\_univer;  use drozhzha\_univer;  create table Student(  [Номер записи] int primary key,  Фамилия nvarchar(50) not null,  [№ зачетки] nvarchar(6) not null,  [№ группы] int not null,  )  insert into Student([Номер записи],Фамилия,[№ зачетки],[№ группы])  Values('1', 'Грибоедов', '123456', '11'),  ('2', 'Пушкин', '654123', '12'),  ('3', 'Леромонтов','654321','5'),  ('4', 'Толстой','654371','4'),  ('5', 'Ахматова','789371','11'),  ('6', 'Гоголь','765971','10')  SELECT [Номер записи],Фамилия,[№ зачетки],[№ группы] From Student order by Фамилия asc  insert into Student([Номер записи],Фамилия,[№ зачетки],[№ группы])  Values('7', 'Берюков', '173456', '12'),  ('8', 'Чацкий', '454123', '1'),  ('9', 'Фамусов','674321','3'),  ('10', 'Сколозуб','612371','4'),  ('11', 'Базаров','788971','6'),  ('12', 'Хлестоков','455971','7')  SELECT \* From Student;  SELECT [Номер записи],Фамилия,[№ зачетки] From Student;  SELECT count(\*) From Student;  SELECT [Номер записи], Фамилия [Студенты 11 группы] From Student  where [№ группы] = 11;  SELECT DISTINCT [№ группы] From Student;  SELECT TOP 5 [Номер записи],Фамилия,[№ зачетки] From Student;  UPDATE Student set [№ группы] = 5;  delete from Student Where [№ зачетки] > '6000000';  SELECT \* From Student;  SELECT count(\*) From Student;  select \* from Student where пол like 'м%';  select Фамилия,[№ зачетки] from Student where Фамилия like 'Грибоедов%' and [№ группы] like '5%' ;  select Фамилия,[№ зачетки] from Student where пол like 'ж%' or [№ зачетки] like '45%' ;  select Фамилия,[№ зачетки] from Student where not Фамилия like 'г%' and [№ зачетки] between 400000 and 700000 ;  select \* from Student where Фамилия in ('Грибоедов','Ахматова', 'Пушкин', 'Чацкий');  Drop table Student;  create table Results(  ID int primary key identity(1, 1),  [имя студента] nvarchar(20) not null,  оценка\_ооп int not null,  оценка\_игиг int not null,  оценка\_бд int not null,  [среднее значение] as (оценка\_ооп+оценка\_игиг+оценка\_бд)/3  )  insert into Results(  [имя студента] ,  оценка\_ооп,  оценка\_игиг,  оценка\_бд  )  values ('Петров','5','6','7'),  ('Белокрылый','10','9','8'),  ('Кемеров','8','5','8'),  ('Делеров','5','9','10'),  ('Краснов','7','8','8'),  ('Бусел','10','9','8'),  ('Трак','8','6','8')  Select \* from Results;  drop table Results; |
|  |

*Лабораторная работа №4*

Настройка базы данных

|  |
| --- |
| USE D\_UNIVER;  drop table PROGRESS  drop table STUDENT  drop table GROUPS  drop table SUBJECT  drop table TEACHER  drop table PULPIT  drop table PROFESSION  drop table FACULTY  drop table AUDITORIUM  drop table AUDITORIUM\_TYPE  use master;  DROP database D\_UNIVER;  use master;  CREATE database D\_UNIVER  ON PRIMARY  (  name=N'D\_UNIVER\_mdf', filename=N'C:\UNIVER\D\_UNIVER.mdf',  size=5MB, maxsize=10MB,filegrowth=1MB  ),  (  name=N'D\_UNIVER\_ndf', filename=N'C:\UNIVER\D\_UNIVER.ndf',  size=5MB, maxsize=10MB,filegrowth=1%  ),  filegroup G1  (  name=N'D\_UNIVER11\_ndf', filename=N'C:\UNIVER\D\_UNIVER11.ndf',  size=10MB, maxsize=15MB,filegrowth=1MB  ),  (  name=N'D\_UNIVER12\_ndf', filename=N'C:\UNIVER\D\_UNIVER12.ndf',  size=2MB, maxsize=5MB,filegrowth=1MB  ),  filegroup G2  (  name=N'D\_UNIVER21\_ndf', filename=N'C:\UNIVER\D\_UNIVER21.ndf',  size=5MB, maxsize=10MB,filegrowth=1MB  ),  (  name=N'D\_UNIVER22\_ndf', filename=N'C:\UNIVER\D\_UNIVER22.ndf',  size=2MB, maxsize=5MB,filegrowth=1MB  )  LOG ON  (  name=N'D\_UNIVER\_log', filename=N'C:\UNIVER\d\_UNIVER.ldf',  size=5MB, maxsize=UNLIMITED,filegrowth=1MB  );  ------------Создание и заполнение таблицы AUDITORIUM\_TYPE  USE D\_UNIVER;  create table AUDITORIUM\_TYPE  ( AUDITORIUM\_TYPE char(10) constraint AUDITORIUM\_TYPE\_PK primary key,  AUDITORIUM\_TYPENAME varchar(30)  )  insert into AUDITORIUM\_TYPE (AUDITORIUM\_TYPE, AUDITORIUM\_TYPENAME ) values ('ЛК', 'Лекционная');  insert into AUDITORIUM\_TYPE (AUDITORIUM\_TYPE, AUDITORIUM\_TYPENAME ) values ('ЛБ-К', 'Компьютерный класс');  insert into AUDITORIUM\_TYPE (AUDITORIUM\_TYPE, AUDITORIUM\_TYPENAME ) values ('ЛК-К', 'Лекционная с уст. проектором');  insert into AUDITORIUM\_TYPE (AUDITORIUM\_TYPE, AUDITORIUM\_TYPENAME ) values ('ЛБ-X', 'Химическая лаборатория');  insert into AUDITORIUM\_TYPE (AUDITORIUM\_TYPE, AUDITORIUM\_TYPENAME ) values ('ЛБ-СК', 'Спец. компьютерный класс');  ------Создание и заполнение таблицы TEACHER  create table TEACHER  ( TEACHER char(10) constraint TEACHER\_PK primary key,  TEACHER\_NAME varchar(100),  GENDER char(1) CHECK (GENDER in ('м', 'ж')),  PULPIT char(20) constraint TEACHER\_PULPIT\_FK foreign key  references PULPIT(PULPIT)  ) ON G1;  ------Создание и заполнение таблицы SUBJECT  create table SUBJECT  ( SUBJECT char(10) constraint SUBJECT\_PK primary key,  SUBJECT\_NAME varchar(100) unique,  PULPIT char(20) constraint SUBJECT\_PULPIT\_FK foreign key  references PULPIT(PULPIT)  ) ON G1;    ------Создание и заполнение таблицы GROUPS  create table GROUPS  ( IDGROUP integer identity(1,1) constraint GROUP\_PK primary key,  FACULTY char(10) constraint GROUPS\_FACULTY\_FK foreign key  references FACULTY(FACULTY),  PROFESSION char(20) constraint GROUPS\_PROFESSION\_FK foreign key  references PROFESSION(PROFESSION),  YEAR\_FIRST smallint check (YEAR\_FIRST<=YEAR(GETDATE())),  ) ON G1; |
|  |
|  |

*Лабораторная работа №5*

Многотабличные SELECT-запросы

|  |
| --- |
| use D\_UNIVER;  ------------1 ЗАДАНИЕ  SELECT AUDITORIUM.AUDITORIUM, AUDITORIUM\_TYPE.AUDITORIUM\_TYPENAME  FROM AUDITORIUM INNER JOIN AUDITORIUM\_TYPE  ON AUDITORIUM.AUDITORIUM\_TYPE=AUDITORIUM\_TYPE.AUDITORIUM\_TYPE;  ------------2 ЗАДАНИЕ  SELECT AUDITORIUM.AUDITORIUM, AUDITORIUM\_TYPE.AUDITORIUM\_TYPENAME  FROM AUDITORIUM INNER JOIN AUDITORIUM\_TYPE  ON AUDITORIUM.AUDITORIUM\_TYPE=AUDITORIUM\_TYPE.AUDITORIUM\_TYPE AND AUDITORIUM\_TYPE.AUDITORIUM\_TYPENAME LIKE 'КОМПЬЮТЕР%';  ------------3 ЗАДАНИЕ ТЕЖЕ ЗАПРОСЫ, НО БЕЗ ИСПОЛЬЗОВАНИЯ ОБЪЕДИНЕНИЯ  SELECT AUDITORIUM.AUDITORIUM, AUDITORIUM\_TYPE.AUDITORIUM\_TYPENAME  FROM AUDITORIUM, AUDITORIUM\_TYPE  WHERE AUDITORIUM.AUDITORIUM\_TYPE=AUDITORIUM\_TYPE.AUDITORIUM\_TYPE;  SELECT TABLE1.AUDITORIUM, TABLE2.AUDITORIUM\_TYPENAME  FROM AUDITORIUM AS TABLE1, AUDITORIUM\_TYPE AS TABLE2  WHERE TABLE1.AUDITORIUM\_TYPE=TABLE2.AUDITORIUM\_TYPE AND TABLE2.AUDITORIUM\_TYPENAME LIKE 'КОМПЬЮТЕР%';  ------------4 ЗАДАНИЕ  SELECT PULPIT.FACULTY[ФАКУЛЬТЕТ],PULPIT.PULPIT\_NAME[КАФЕДРА],GROUPS.PROFESSION[СПЕЦИАЛЬНОСТЬ], SUBJECT.SUBJECT[ДИСЦИПЛИНА],STUDENT.NAME[ИМЯ СТУДЕНТА],  CASE  WHEN (PROGRESS.NOTE=6) THEN 'ШЕСТЬ'  WHEN (PROGRESS.NOTE=7) THEN 'СЕМЬ'  WHEN (PROGRESS.NOTE=8) THEN 'ВОСЕМЬ'  ELSE 'НЕ подходит в диапозон'  END [ОЦЕНКИ]  FROM FACULTY INNER JOIN PULPIT  ON FACULTY.FACULTY=PULPIT.FACULTY  INNER JOIN SUBJECT  ON PULPIT.PULPIT=SUBJECT.PULPIT  INNER JOIN PROGRESS  ON SUBJECT.SUBJECT=PROGRESS.SUBJECT  INNER JOIN STUDENT  ON PROGRESS.IDSTUDENT=STUDENT.IDSTUDENT AND PROGRESS.NOTE BETWEEN 6 AND 8  INNER JOIN GROUPS  ON STUDENT.IDGROUP=GROUPS.IDGROUP  ORDER BY FACULTY.FACULTY, PULPIT.PULPIT, GROUPS.PROFESSION, STUDENT.NAME, PROGRESS.NOTE DESC ;  ------------5 ЗАДАНИЕ  SELECT PULPIT.FACULTY[ФАКУЛЬТЕТ],PULPIT.PULPIT\_NAME[КАФЕДРА],GROUPS.PROFESSION[СПЕЦИАЛЬНОСТЬ], SUBJECT.SUBJECT[ДИСЦИПЛИНА],STUDENT.NAME[ИМЯ СТУДЕНТА],  CASE  WHEN (PROGRESS.NOTE=6) THEN 'ШЕСТЬ'  WHEN (PROGRESS.NOTE=7) THEN 'СЕМЬ'  WHEN (PROGRESS.NOTE=8) THEN 'ВОСЕМЬ'  ELSE 'НЕ подходит в диапозон'  END [ОЦЕНКИ]  FROM FACULTY INNER JOIN PULPIT  ON FACULTY.FACULTY=PULPIT.FACULTY  INNER JOIN SUBJECT  ON PULPIT.PULPIT=SUBJECT.PULPIT  INNER JOIN PROGRESS  ON SUBJECT.SUBJECT=PROGRESS.SUBJECT  INNER JOIN STUDENT  ON PROGRESS.IDSTUDENT=STUDENT.IDSTUDENT AND PROGRESS.NOTE BETWEEN 6 AND 8  INNER JOIN GROUPS  ON STUDENT.IDGROUP=GROUPS.IDGROUP  ORDER BY  (  CASE  WHEN (PROGRESS.NOTE=6) THEN 3  WHEN (PROGRESS.NOTE=7) THEN 1  WHEN (PROGRESS.NOTE=8) THEN 2  END  );  ------------6 ЗАДАНИЕ  SELECT PULPIT.PULPIT\_NAME, ISNULL (TEACHER.TEACHER\_NAME, '\*\*\*')[ПРЕПОДАВАТЕЛИ]  FROM PULPIT LEFT OUTER JOIN TEACHER  ON PULPIT.PULPIT=TEACHER.PULPIT;  ------------7 ЗАДАНИЕ  SELECT PULPIT.PULPIT\_NAME, ISNULL (TEACHER.TEACHER\_NAME, '\*\*\*')[ПРЕПОДАВАТЕЛИ]  FROM TEACHER LEFT OUTER JOIN PULPIT  ON PULPIT.PULPIT=TEACHER.PULPIT;  SELECT PULPIT.PULPIT\_NAME, ISNULL (TEACHER.TEACHER\_NAME, '\*\*\*')[ПРЕПОДАВАТЕЛИ]  FROM TEACHER RIGHT OUTER JOIN PULPIT  ON PULPIT.PULPIT=TEACHER.PULPIT;  ------------8 ЗАДАНИЕ  CREATE DATABASE CAKE;  USE CAKE;  CREATE TABLE CAKESS(  NAMECAKE CHAR(10) NOT NULL,  PRICE INT,  WEIGHT FLOAT,  CONSTRAINT PK\_CAKES PRIMARY KEY (NAMECAKE)  );  CREATE TABLE BUYERS(  NAMEBU CHAR(100) NOT NULL,  DATE DATETIME NOT NULL,  NAMECAKE CHAR(10) NOT NULL,  CONSTRAINT PK\_BUYER PRIMARY KEY (NAMEBU),  CONSTRAINT FK\_BUYER FOREIGN KEY (NAMECAKE) REFERENCES CAKESS(NAMECAKE)  );  INSERT CAKESS VALUES  ('ТОРТИК1','12','1'),  ('ТОРТИК2','14','1'),  ('ТОРТИК3','15','1.2'),  ('ТОРТИК4','16','1.5'),  ('ТОРТИК5','17','2'),  ('ТОРТИК6','18','2.1'),  ('ТОРТИК7','19','2.2')  ;  INSERT BUYERS VALUES  ('ЗАКАЗЧИК1','12 APR 2021','ТОРТИК1'),  ('ЗАКАЗЧИК2','14 APR 2021','ТОРТИК4'),  ('ЗАКАЗЧИК3','14 APR 2021','ТОРТИК3'),  ('ЗАКАЗЧИК4','21 APR 2021','ТОРТИК7')  ;  SELECT BUYERS.NAMEBU[ИМЯ ЗАКАЗЧИКА],BUYERS.DATE[ДАТА ЗАКАЗА], CAKESS.NAMECAKE[НАЗВАНИЕ ТОРТА],CAKESS.PRICE[ЦЕНА]  FROM BUYERS INNER JOIN CAKESS  ON BUYERS.NAMECAKE=CAKESS.NAMECAKE;  ------------ПРОВЕРКА НА КОММУТАТИВНОСТЬ  SELECT BUYERS.NAMEBU[ИМЯ ЗАКАЗЧИКА],BUYERS.DATE[ДАТА ЗАКАЗА], CAKESS.NAMECAKE[НАЗВАНИЕ ТОРТА],CAKESS.PRICE[ЦЕНА]  FROM BUYERS FULL OUTER JOIN CAKESS  ON BUYERS.NAMECAKE=CAKESS.NAMECAKE;  SELECT BUYERS.NAMEBU[ИМЯ ЗАКАЗЧИКА],BUYERS.DATE[ДАТА ЗАКАЗА], CAKESS.NAMECAKE[НАЗВАНИЕ ТОРТА],CAKESS.PRICE[ЦЕНА]  FROM CAKESS FULL OUTER JOIN BUYERS  ON BUYERS.NAMECAKE=CAKESS.NAMECAKE;  ------------FULL=LEFT+RIGHT  SELECT BUYERS.NAMEBU[ИМЯ ЗАКАЗЧИКА],BUYERS.DATE[ДАТА ЗАКАЗА], CAKESS.NAMECAKE[НАЗВАНИЕ ТОРТА],CAKESS.PRICE[ЦЕНА]  FROM CAKESS LEFT OUTER JOIN BUYERS  ON BUYERS.NAMECAKE=CAKESS.NAMECAKE;  SELECT BUYERS.NAMEBU[ИМЯ ЗАКАЗЧИКА],BUYERS.DATE[ДАТА ЗАКАЗА], CAKESS.NAMECAKE[НАЗВАНИЕ ТОРТА],CAKESS.PRICE[ЦЕНА]  FROM CAKESS RIGHT OUTER JOIN BUYERS  ON BUYERS.NAMECAKE=CAKESS.NAMECAKE;  ------------LEFT NOT RIGHT  SELECT BUYERS.NAMEBU[ИМЯ ЗАКАЗЧИКА],BUYERS.DATE[ДАТА ЗАКАЗА], CAKESS.NAMECAKE[НАЗВАНИЕ ТОРТА],CAKESS.PRICE[ЦЕНА]  FROM CAKESS FULL OUTER JOIN BUYERS  ON BUYERS.NAMECAKE=CAKESS.NAMECAKE WHERE BUYERS.NAMECAKE IS NOT NULL;  ------------9 ЗАДАНИЕ cross join  USE D\_UNIVER;  SELECT AUDITORIUM.AUDITORIUM, AUDITORIUM\_TYPE.AUDITORIUM\_TYPENAME  FROM AUDITORIUM CROSS JOIN AUDITORIUM\_TYPE  WHERE AUDITORIUM.AUDITORIUM\_TYPE=AUDITORIUM\_TYPE.AUDITORIUM\_TYPE;  ------------10 ЗАДАНИЕ  use D\_UNIVER;  CREATE TABLE TIMETABLE(  LESSON\_ID int primary key identity(10234,13),  LESSON\_TIME varchar(50),  DAYOFTHEWEEK char(20),  AUDITORIUM char(20) foreign key references AUDITORIUM(AUDITORIUM),  GROUP\_ID int foreign key references GROUPS(IDGROUP),  SUBJECT char(10) foreign key references SUBJECT(SUBJECT),  TEACHER char(10) foreign key references TEACHER(TEACHER),  );  INSERT INTO TIMETABLE(LESSON\_TIME,DAYOFTHEWEEK,AUDITORIUM,GROUP\_ID,SUBJECT,TEACHER)  values('08:00', 'пн', '313-1', 13,'ВТЛ', 'НСКВЦ'),  ('08:00', 'вт', '408-2', 11,'ИНФ', 'ДТК'),  ('09:50', 'вт', '236-1', 1,'КМС', 'ЖЛК'),  ('17:30', 'сб', '324-1', 28,'ЭП', 'НВРВ'),  ('15:25', 'ср', '206-1', 5,'ЭТ', 'БРНВСК'),  ('08:00', 'чт', '423-1', 7,'МСОИ', 'БРКВЧ'),  ('11:40', 'пт', '413-1', 23,'ЛВ', 'ЗВГЦВ'),  ('13:50', 'ср', '313-1', 4,'СУБД', 'АКНВЧ');  SELECT \* from TIMETABLE;  SELECT t.LESSON\_TIME[Время], t.DAYOFTHEWEEK[День недели],  a.AUDITORIUM[Аудитория],[Статус]='Свободна'  from TIMETABLE as t cross join AUDITORIUM as a  where t.AUDITORIUM!=a.AUDITORIUM;  SELECT t.LESSON\_TIME[Время], t.DAYOFTHEWEEK[День недели],  tech.TEACHER\_NAME[Преподаватель],[Статус]='Свободен'  from TIMETABLE as t cross join TEACHER as tech  where t.TEACHER != tech.TEACHER;  SELECT t.LESSON\_TIME[Время], t.DAYOFTHEWEEK[День недели],  g.[IDGROUP],[Статус]='Свободен'  from TIMETABLE as t cross join GROUPS as g  where t.GROUP\_ID != g.IDGROUP; |

*Лабораторная работа № 6*

[Использование подзапросов](#Лаб7)

**Подзапрос** – это SELECT-запрос, который выполняется в рамках другого запроса. Подзапросы могут применяться в секции WHERE. Подзапросы бывают двух видов: коррелируемые и независимые.

*Коррелируемый* подзапрос зависит от внешнего запроса и выполняется для каждой строки результирующего набора.

*Независимый* подзапрос не зависит от внешнего запроса и выполняется только один раз, но результат его выполнения подставляется в каждую строку результирующего набора.

|  |
| --- |
| use D\_UNIVER;  ------Задание 1 (Наименование кафедр со словом технологии/я)  SELECT PULPIT.PULPIT\_NAME [Кафедра], PROFESSION.PROFESSION\_NAME [Специальности]  FROM PULPIT, PROFESSION  WHERE PULPIT.FACULTY=PROFESSION.FACULTY AND  PROFESSION\_NAME IN (SELECT PROFESSION\_NAME FROM PROFESSION  WHERE (PROFESSION\_NAME LIKE '%Технологи%'));  ------Задание 2 (Наименование кафедр со словом технологии/я INNER JOIN)  SELECT PULPIT.PULPIT\_NAME [Кафедра], FACULTY.FACULTY\_NAME [Факультет], PROFESSION.PROFESSION\_NAME [Специальности]  FROM PULPIT INNER JOIN FACULTY  ON PULPIT.FACULTY=FACULTY.FACULTY  INNER JOIN PROFESSION  ON FACULTY.FACULTY=PROFESSION.FACULTY  WHERE PROFESSION\_NAME IN (SELECT PROFESSION\_NAME FROM PROFESSION  WHERE (PROFESSION\_NAME LIKE '%Технологи%'));  ------Задание 3 (Наименование кафедр со словом технологии/я INNER JOIN)  SELECT PULPIT.PULPIT\_NAME [Кафедра], FACULTY.FACULTY\_NAME [Факультет], PROFESSION.PROFESSION\_NAME [Специальности]  FROM PULPIT INNER JOIN FACULTY  ON PULPIT.FACULTY=FACULTY.FACULTY  INNER JOIN PROFESSION  ON FACULTY.FACULTY=PROFESSION.FACULTY  WHERE (PROFESSION\_NAME LIKE '%Технологи%');  ------Задание 4 (Наименование самых больших аудиторий)  SELECT DISTINCT AUDITORIUM\_TYPE[ТИП АУДИТОРИИ], AUDITORIUM\_CAPACITY[ВМЕСТИМОСТЬ]  FROM AUDITORIUM A  WHERE AUDITORIUM\_CAPACITY=(SELECT TOP(1) AUDITORIUM\_CAPACITY FROM AUDITORIUM AA  WHERE AA.AUDITORIUM\_TYPE=A.AUDITORIUM\_TYPE  ORDER BY AUDITORIUM\_CAPACITY DESC);  ------Задание 5 (Наименование факультетов на котором нет кафедр)  SELECT FACULTY.FACULTY\_NAME FROM FACULTY  WHERE NOT EXISTS(SELECT \* FROM PULPIT  WHERE FACULTY.FACULTY= PULPIT.FACULTY)  ------Задание 6 (Оценки по оаип бд и субд)  SELECT TOP 1  (SELECT AVG(NOTE) FROM PROGRESS  WHERE SUBJECT LIKE 'ОАиП')[ОАИП],  (SELECT AVG(NOTE) FROM PROGRESS  WHERE SUBJECT LIKE 'БД')[БД],  (SELECT AVG(NOTE) FROM PROGRESS  WHERE SUBJECT LIKE 'СУБД')[СУБД],  (SELECT AVG(NOTE) FROM PROGRESS  WHERE SUBJECT LIKE 'КГ')[КГ]  FROM PROGRESS  ------Задание 7 (ALL)  SELECT NOTE[ОЦЕНКА],SUBJECT[ПРЕДМЕТ] FROM PROGRESS  WHERE NOTE = ALL(SELECT NOTE FROM PROGRESS  WHERE NOTE=4)  ------Задание 8 (ANY)  SELECT FACULTY[ФАКУЛЬТЕТ],PROFESSION\_NAME[НАЗВАНИЕ СПЕЦИАЛЬНОСТИ],PROFESSION[КОД] FROM PROFESSION  WHERE FACULTY = ANY(SELECT FACULTY FROM PROFESSION  WHERE FACULTY LIKE '%Т%')  ------Задание 9\* (ДР в один день)  SELECT DISTINCT A.IDSTUDENT, A.NAME, A.BDAY, AA.IDSTUDENT, AA.NAME, AA.BDAY  FROM STUDENT A INNER JOIN STUDENT AA  ON A.BDAY=AA.BDAY  WHERE AA.NAME!= A.NAME; |
|  |

*Лабораторная работа №7*

Группировка данных

Основное назначение **группировки** с помощью секции GROUP BY – разбиение множества строк, сформированных секциями FROM и WHERE, на группы в соответствии со значениями в заданных столбцах, а также выполнение вычислений над группами строк с помощью наиболее часто используемых функций: **AVG**(вычисление среднего значения), **COUNT** (вычисление количества строк), **MAX**(вычисление максимального значения), **MIN** (вычисление минимального значения), **SUM** (вычисление суммы значений). При использовании секции **GROUP BY** в SELECT-списке допускается указывать **только** те столбцы, по которым осуществляется группировка.

|  |
| --- |
| USE D\_UNIVER;  ------1 Задание(макс, мин средня вместимость аудиторий)  SELECT MIN(AUDITORIUM\_CAPACITY)[МИНИМАЛЬНАЯ ВМЕСТИМОСТЬ],  MAX(AUDITORIUM\_CAPACITY)[МАКСИМАЛЬНАЯ ВМЕСТИМОСТЬ],  AVG(AUDITORIUM\_CAPACITY)[СРЕДНЯЯ ВМЕСТИМОСТЬ],  SUM(AUDITORIUM\_CAPACITY)[СУММАРНАЯ ВМЕСТИМОСТЬ]  FROM AUDITORIUM  ------2 Задание(GROUP BY)  SELECT AUDITORIUM\_TYPENAME[ТИП АУДИТОРИИ],  MIN(AUDITORIUM\_CAPACITY)[МИНИМАЛЬНАЯ ВМЕСТИМОСТЬ],  MAX(AUDITORIUM\_CAPACITY)[МАКСИМАЛЬНАЯ ВМЕСТИМОСТЬ],  AVG(AUDITORIUM\_CAPACITY)[СРЕДНЯЯ ВМЕСТИМОСТЬ],  SUM(AUDITORIUM\_CAPACITY)[СУММАРНАЯ ВМЕСТИМОСТЬ]  FROM AUDITORIUM\_TYPE INNER JOIN AUDITORIUM  ON AUDITORIUM\_TYPE.AUDITORIUM\_TYPE=AUDITORIUM.AUDITORIUM\_TYPE  AND AUDITORIUM.AUDITORIUM\_CAPACITY>16 GROUP BY AUDITORIUM\_TYPENAME  ------3 Задание(ДИАПАЗОН)  SELECT \*  FROM(SELECT  CASE  WHEN NOTE BETWEEN 1 AND 3 THEN '1-3'  WHEN NOTE BETWEEN 4 AND 5 THEN '4-5'  WHEN NOTE BETWEEN 6 AND 7 THEN '6-7'  WHEN NOTE BETWEEN 8 AND 9 THEN '8-9'  ELSE '10'  END [ПРЕДЕЛЫ ОЦЕНОК],COUNT(\*)[КОЛИЧЕСТВО]  FROM PROGRESS GROUP BY CASE  WHEN NOTE BETWEEN 1 AND 3 THEN '1-3'  WHEN NOTE BETWEEN 4 AND 5 THEN '4-5'  WHEN NOTE BETWEEN 6 AND 7 THEN '6-7'  WHEN NOTE BETWEEN 8 AND 9 THEN '8-9'  ELSE '10'  END) AS T  ORDER BY CASE[ПРЕДЕЛЫ ОЦЕНОК]  WHEN '1-3' THEN 4  WHEN '4-5' THEN 3  WHEN '6-7' THEN 2  WHEN '8-9' THEN 1  ELSE 0  END  ------4 Задание(СРЕДНЯЯ ЭКЗАМЕНАЦИОННАЯ ОЦЕНКА)  SELECT FACULTY.FACULTY[ФАКУЛЬТЕТ] ,GROUPS.PROFESSION[СПЕЦИАЛЬНОСТЬ],ROUND(AVG(CAST(PROGRESS.NOTE AS FLOAT(4))),2)[СРЕДНЯЯ ОЦЕНКА]  FROM FACULTY JOIN GROUPS  ON FACULTY.FACULTY =GROUPS.FACULTY  JOIN STUDENT  ON STUDENT.IDGROUP=GROUPS.IDGROUP  JOIN PROGRESS  ON PROGRESS.IDSTUDENT=STUDENT.IDSTUDENT  GROUP BY FACULTY.FACULTY, GROUPS.PROFESSION  ORDER BY [СРЕДНЯЯ ОЦЕНКА] DESC  ------ПЕЕИРЕПИСАННЫЙ СЕЛЕКТ ЗАПРОС  SELECT FACULTY.FACULTY[ФАКУЛЬТЕТ] ,GROUPS.PROFESSION[СПЕЦИАЛЬНОСТЬ],ROUND(AVG(CAST(PROGRESS.NOTE AS FLOAT(4))),2)[СРЕДНЯЯ ОЦЕНКА]  FROM FACULTY JOIN GROUPS  ON FACULTY.FACULTY =GROUPS.FACULTY  JOIN STUDENT  ON STUDENT.IDGROUP=GROUPS.IDGROUP  JOIN PROGRESS  ON PROGRESS.IDSTUDENT=STUDENT.IDSTUDENT WHERE PROGRESS.SUBJECT LIKE 'БД' OR PROGRESS.SUBJECT LIKE 'ОАиП'  GROUP BY FACULTY.FACULTY, GROUPS.PROFESSION  ORDER BY [СРЕДНЯЯ ОЦЕНКА] DESC  -----------5 СТУДЕТЫ С ТОВА И ФУНКЦЯ РОЛАП  SELECT GROUPS.PROFESSION[СПЕЦИАЛЬНОСТЬ],PROGRESS.SUBJECT[ПРЕДМЕТ], ROUND(AVG(CAST(PROGRESS.NOTE AS FLOAT(4))),2)[СРЕДНЯЯ ОЦЕНКА]  FROM FACULTY JOIN GROUPS  ON FACULTY.FACULTY =GROUPS.FACULTY  JOIN STUDENT  ON STUDENT.IDGROUP=GROUPS.IDGROUP  JOIN PROGRESS  ON PROGRESS.IDSTUDENT=STUDENT.IDSTUDENT  WHERE FACULTY.FACULTY LIKE 'ТОВ'  GROUP BY GROUPS.PROFESSION, PROGRESS.SUBJECT  SELECT GROUPS.PROFESSION, PROGRESS.SUBJECT,ROUND(AVG(CAST(PROGRESS.NOTE AS FLOAT(4))),2)[СРЕДНЯЯ ОЦЕНКА]  FROM FACULTY  JOIN GROUPS ON FACULTY.FACULTY = GROUPS.FACULTY  JOIN STUDENT ON STUDENT.IDGROUP= GROUPS.IDGROUP  JOIN PROGRESS ON PROGRESS.IDSTUDENT=STUDENT.IDSTUDENT  GROUP BY ROLLUP (GROUPS.PROFESSION, PROGRESS.SUBJECT)  -----------6 ПЕРЕДЕЛАННЫЙ ЗАПРОС С ФУНКЦИЕЙ CUBE  SELECT GROUPS.PROFESSION, PROGRESS.SUBJECT,ROUND(AVG(CAST(PROGRESS.NOTE AS FLOAT(4))),2)[СРЕДНЯЯ ОЦЕНКА]  FROM FACULTY  JOIN GROUPS ON FACULTY.FACULTY = GROUPS.FACULTY  JOIN STUDENT ON STUDENT.IDGROUP= GROUPS.IDGROUP  JOIN PROGRESS ON PROGRESS.IDSTUDENT=STUDENT.IDSTUDENT  GROUP BY CUBE (GROUPS.PROFESSION, PROGRESS.SUBJECT)  -----------7 РЕЗУЛЬТАТЫ ЭКЗОВ ФУНКЦИИ UNION/ALL  SELECT GROUPS.FACULTY[FACULTY], GROUPS.PROFESSION[PROFFESSION], PROGRESS.SUBJECT,ROUND(AVG(CAST(PROGRESS.NOTE AS FLOAT(4))),2)[СРЕДНЯЯ ОЦЕНКА]  FROM GROUPS, STUDENT, PROGRESS  WHERE GROUPS.FACULTY LIKE 'ТОВ'  GROUP BY GROUPS.FACULTY, GROUPS.PROFESSION, PROGRESS.SUBJECT  UNION  SELECT GROUPS.FACULTY[FACULTY], GROUPS.PROFESSION[PROFFESSION], PROGRESS.SUBJECT,ROUND(AVG(CAST(PROGRESS.NOTE AS FLOAT(4))),2)[СРЕДНЯЯ ОЦЕНКА]  FROM GROUPS, STUDENT, PROGRESS  WHERE GROUPS.FACULTY LIKE 'ХТиТ'  GROUP BY GROUPS.FACULTY, GROUPS.PROFESSION, PROGRESS.SUBJECT  SELECT GROUPS.FACULTY[FACULTY], GROUPS.PROFESSION[PROFFESSION], PROGRESS.SUBJECT,ROUND(AVG(CAST(PROGRESS.NOTE AS FLOAT(4))),2)[СРЕДНЯЯ ОЦЕНКА]  FROM GROUPS, STUDENT, PROGRESS  WHERE GROUPS.FACULTY LIKE 'ТОВ'  GROUP BY GROUPS.FACULTY, GROUPS.PROFESSION, PROGRESS.SUBJECT  UNION ALL  SELECT GROUPS.FACULTY[FACULTY], GROUPS.PROFESSION[PROFFESSION], PROGRESS.SUBJECT,ROUND(AVG(CAST(PROGRESS.NOTE AS FLOAT(4))),2)[СРЕДНЯЯ ОЦЕНКА]  FROM GROUPS, STUDENT, PROGRESS  WHERE GROUPS.FACULTY LIKE 'ХТиТ'  GROUP BY GROUPS.FACULTY, GROUPS.PROFESSION, PROGRESS.SUBJECT  ------8 ПЕРЕСЕССЕНИЕ  SELECT GROUPS.FACULTY, GROUPS.PROFESSION, PROGRESS.SUBJECT, ROUND(AVG(CAST(PROGRESS.NOTE AS FLOAT(4))),2)[СРЕДНЯЯ ОЦЕНКА]  FROM GROUPS, STUDENT, PROGRESS  WHERE GROUPS.FACULTY IN('ТОВ','ХТиТ')  GROUP BY GROUPS.FACULTY, GROUPS.PROFESSION, PROGRESS.SUBJECT  INTERSECT  SELECT GROUPS.FACULTY, GROUPS.PROFESSION, PROGRESS.SUBJECT, ROUND(AVG(CAST(PROGRESS.NOTE AS FLOAT(4))),2)[СРЕДНЯЯ ОЦЕНКА]  FROM GROUPS, STUDENT, PROGRESS  WHERE GROUPS.FACULTY LIKE 'ХТиТ'  GROUP BY GROUPS.FACULTY, GROUPS.PROFESSION, PROGRESS.SUBJECT  ------9 РАЗНИЦА МЕЖДУ МНОЖЕСТВАМИ ОПЕРАУИЯ ЕКСЕПТ  SELECT GROUPS.FACULTY, GROUPS.PROFESSION, PROGRESS.SUBJECT, ROUND(AVG(CAST(PROGRESS.NOTE AS FLOAT(4))),2)[СРЕДНЯЯ ОЦЕНКА]  FROM GROUPS, STUDENT, PROGRESS  WHERE GROUPS.FACULTY IN('ТОВ','ХТиТ')  GROUP BY GROUPS.FACULTY, GROUPS.PROFESSION, PROGRESS.SUBJECT  EXCEPT  SELECT GROUPS.FACULTY, GROUPS.PROFESSION, PROGRESS.SUBJECT, ROUND(AVG(CAST(PROGRESS.NOTE AS FLOAT(4))),2)[СРЕДНЯЯ ОЦЕНКА]  FROM GROUPS, STUDENT, PROGRESS  WHERE GROUPS.FACULTY IN('ИТ','ХТиТ')  GROUP BY GROUPS.FACULTY, GROUPS.PROFESSION, PROGRESS.SUBJECT  -----10 КОЛ-ВО СТУДЕНТОВ ПО ДИСЦИПЛИНАМ СДАВШИЕ НА ОТЛИЧНО  SELECT PROGRESS.SUBJECT, COUNT(\*) [Кол-во 8-9]  FROM PROGRESS  GROUP BY PROGRESS.SUBJECT, PROGRESS.NOTE  HAVING PROGRESS.NOTE BETWEEN 8 AND 9 |
|  |

*Лабораторная работа №8*

Использование представлений

**Представление** (View) – это объект базы данных, представляющий собой *поименованный* SELECT-запрос, который хранится в базе данных. Представление создается с помощью оператора CREATE, удаляется с помощью оператора DROP и изменяется с помощью ALTER.

|  |
| --- |
| USE D\_UNIVER;  -------1 СОЗДАТЬ ПРЕДСТАВЛЕНИЕ [ПРЕПОДАВАТЕЛИ\_БГТУ]  SELECT TEACHER [КОД],  TEACHER\_NAME [ИМЯ ПРЕПОДАВАТЕЛЯ],  GENDER[ПОЛ],  PULPIT[КАФЕДРА] FROM TEACHER;  GO  CREATE VIEW [ПРЕПОДАВАТЕЛИ\_БГТУ] AS SELECT TEACHER [КОД],  TEACHER\_NAME [ИМЯ ПРЕПОДАВАТЕЛЯ],  GENDER[ПОЛ],  PULPIT[КАФЕДРА] FROM TEACHER;  -----ТОТ ЖЕ ЗАПРОС МОЖНО ЗАПИСАТЬ КОРОЧЕ  GO  SELECT \* FROM [ПРЕПОДАВАТЕЛИ\_БГТУ]  SELECT \* FROM [ПРЕПОДАВАТЕЛИ\_БГТУ] ORDER BY [ИМЯ ПРЕПОДАВАТЕЛЯ]  DROP VIEW [ПРЕПОДАВАТЕЛИ\_БГТУ]  -------2 СОЗДАТЬ ПРЕДСТАВЛЕНИЕ [КОЛИЧЕСТВО КАФЕДР]  GO  CREATE VIEW [dbo].[КОЛИЧЕСТВО\_КАФЕДР] AS SELECT  FACULTY.FACULTY\_NAME [ФАКУЛЬТЕТ],  (SELECT COUNT(\*) FROM PULPIT WHERE FACULTY.FACULTY=PULPIT.FACULTY) [КОЛИЧЕСТВО ФАКУЛЬТЕТОВ]  FROM FACULTY JOIN PULPIT  ON FACULTY.FACULTY=PULPIT.FACULTY  GROUP BY FACULTY.FACULTY, FACULTY.FACULTY\_NAME;  GO  SELECT \* FROM [КОЛИЧЕСТВО\_КАФЕДР]  GO  DROP VIEW [КОЛИЧЕСТВО\_КАФЕДР]  -------3 СОЗДАТЬ ПРЕДСТАВЛЕНИЕ [АУДИТОРИЯ]  GO  CREATE VIEW АУДИТОРИЯ (КОД, НАИМЕНОВАНИЕ\_АУДИТОРИИ) AS SELECT  AUDITORIUM,  AUDITORIUM\_TYPE FROM AUDITORIUM  WHERE AUDITORIUM\_TYPE LIKE 'ЛК'  GO  INSERT АУДИТОРИЯ VALUES ('350-5','ЛК');  DELETE FROM АУДИТОРИЯ WHERE [КОД]='3464-5';  UPDATE АУДИТОРИЯ SET [НАИМЕНОВАНИЕ\_АУДИТОРИИ]='ЛК-К' WHERE [КОД]='345-5';  SELECT \* FROM [АУДИТОРИЯ]  GO  DROP VIEW [АУДИТОРИЯ]  -------4 СОЗДАТЬ ПРЕДСТАВЛЕНИЕ [Лекционные\_аудитории]  GO  CREATE VIEW ЛЕКЦИОННЫЕ\_АУДИТОРИИ (КОД, НАИМЕНОВАНИЕ\_АУДИТОРИЙ) AS SELECT  AUDITORIUM,  AUDITORIUM\_TYPE FROM AUDITORIUM  WHERE AUDITORIUM\_TYPE LIKE 'ЛК'  GO  ALTER VIEW ЛЕКЦИОННЫЕ\_АУДИТОРИИ (КОД, НАИМЕНОВАНИЕ\_АУДИТОРИЙ) AS SELECT  AUDITORIUM,  AUDITORIUM\_TYPE FROM AUDITORIUM  WHERE AUDITORIUM\_TYPE LIKE 'ЛК' WITH CHECK OPTION;  GO  INSERT ЛЕКЦИОННЫЕ\_АУДИТОРИИ VALUES ('374-5','ЛК');  UPDATE ЛЕКЦИОННЫЕ\_АУДИТОРИИ SET [НАИМЕНОВАНИЕ\_АУДИТОРИЙ]='ЛК' WHERE [КОД] LIKE '236-1';  --INSERT ЛЕКЦИОННЫЕ\_АУДИТОРИИ VALUES ('364-5','ЛК-К');  --UPDATE ЛЕКЦИОННЫЕ\_АУДИТОРИИ SET [НАИМЕНОВАНИЕ\_АУДИТОРИЙ]='ЛК-К' WHERE [КОД] LIKE '236-1';  SELECT \* FROM [ЛЕКЦИОННЫЕ\_АУДИТОРИИ]  GO  DROP VIEW [ЛЕКЦИОННЫЕ\_АУДИТОРИИ]  -------5 СОЗДАТЬ ПРЕДСТАВЛЕНИЕ [ДИСЦПЛЛИНЫ]  GO  CREATE VIEW ДИСЦИПЛИНЫ (КОД, НАИМЕНОВАНИЕ\_ДИСЦИПЛИНЫ, КОД\_КАФЕДРЫ) AS SELECT TOP 30  SUBJECT,  SUBJECT\_NAME,  PULPIT FROM SUBJECT ORDER BY PULPIT;  GO  SELECT \* FROM [ДИСЦИПЛИНЫ]  GO  DROP VIEW [ДИСЦИПЛИНЫ]  ------- 6 ИЗМЕНИТЬ ПРЕДСТАВЛЕНИЕ [КОЛИЧЕСТВО КАФЕДР]  GO  ALTER VIEW [dbo].[КОЛИЧЕСТВО\_КАФЕДР] WITH SCHEMABINDING AS SELECT  dbo.FACULTY .FACULTY\_NAME [ФАКУЛЬТЕТ],  (SELECT COUNT(\*) FROM dbo.PULPIT WHERE dbo.FACULTY .FACULTY=dbo.PULPIT.FACULTY) [КОЛИЧЕСТВО ФАКУЛЬТЕТОВ]  FROM dbo.FACULTY JOIN dbo.PULPIT  ON dbo.FACULTY .FACULTY=dbo.PULPIT.FACULTY  GROUP BY dbo.FACULTY .FACULTY, dbo.FACULTY .FACULTY\_NAME, dbo.PULPIT.FACULTY;  GO  SELECT \* FROM [dbo].[КОЛИЧЕСТВО\_КАФЕДР]  GO  DROP VIEW [dbo].[КОЛИЧЕСТВО\_КАФЕДР]  ------- 7\* СОЗДАТЬ ПРЕДСТАВЛЕНИЕ [РАСПИСАНИЕ] ДЛЯ ТАБЛИЦЫ ИЗ 6 ЛАБОРАТОРНОЙ PIVOT  ------- ПОДСЧИТАТЬ КОЛИЧЕСТВО ДИСЦИПЛИН КОТОРЫЕ ПРОХОДТ В ОДИН ДЕНЬ  GO  -------ЗАПРОС БЕЗ ИСПОЛЬЗОВАНИЯ PIVOT  CREATE VIEW РАСПИСАНИЕ AS SELECT  A.DAYOFTHEWEEK [ДЕНЬ\_НЕДЕЛИ], (SELECT DISTINCT COUNT (\*) FROM TIMETABLE A WHERE A.DAYOFTHEWEEK=AA.DAYOFTHEWEEK) [КОЛИЧЕСТВО]  FROM TIMETABLE A JOIN TIMETABLE AA  ON A.DAYOFTHEWEEK=AA.DAYOFTHEWEEK  GROUP BY A.DAYOFTHEWEEK,AA.DAYOFTHEWEEK;  GO  SELECT \* FROM РАСПИСАНИЕ  DROP VIEW РАСПИСАНИЕ  ------- PIVOT  GO  USE CAKE;  -------БЫЛО  SELECT \* FROM SHOP  -------ПРЕДСТАВЛЕНИЕ С ФУНКЦИЕЙ ПИВОТ(ПО СУТИ ОНА ТРАНСПОНИРУЕТ МАТРИЦУ)  GO  CREATE VIEW ПИВОТ AS SELECT NAMECAKE, [2017],[2018],[2019],[2020]  FROM SHOP  PIVOT (SUM(COSTS)for NAMEYEAR in ([2017],[2018],[2019],[2020])  ) AS test\_pivot  GO  SELECT \* FROM ПИВОТ |
|  |
|  |

*Лабораторная работа №9*

Основы программирования на T-SQL

|  |
| --- |
| USE D\_UNIVER;  -----1 ЗАДАНИЕ ОБЪЯВЛЕНИЕ ПЕРЕМЕННЫХ  -----объявить переменные типа char, varchar, datetime, time, int, smallint, tinint, numeric(12, 5);  -----SELECT  DECLARE @VAR CHAR='V',  @VARCHAR VARCHAR(4)='CHAR',  @DATE DATETIME= GETDATE(),  @YC SMALLINT= 45,  @CH TINYINT;  SELECT @VAR CHAR,@VARCHAR VARCHAR,@DATE DATETIME,@YC SMALLINT,@CH TINYINT;  -----PRINT  DECLARE @INT INT,  @NUM NUMERIC(12,5)=13 ;  SET @INT = '10';  PRINT 'INT= '+CAST(@INT AS VARCHAR(10));  PRINT 'NUM= '+CAST(@NUM AS VARCHAR(10));  -----2 ЗАДАНИЕ НАХОЖДЕНИЕ ОБЩЕЙ ВМЕСТИМОСТИ  ----- то вывести количество аудиторий, среднюю вместимость аудиторий, количество аудиторий, вместимость которых меньше средней,  -----и процент таких аудиторий.  DECLARE @ALLCAPACITY NUMERIC(8,3)=(SELECT CAST(SUM(AUDITORIUM\_CAPACITY) AS NUMERIC(8,3)) FROM AUDITORIUM),@QUANTCAPACITY REAL,  @AVGCAPACITY REAL, @CAPACITOFTASK INT, @PROCENT NUMERIC(8,3)  IF @ALLCAPACITY>200  BEGIN  SELECT @QUANTCAPACITY=(SELECT CAST(COUNT(\*) AS NUMERIC(8,3)) FROM AUDITORIUM),  @AVGCAPACITY=(SELECT CAST(AVG(AUDITORIUM\_CAPACITY) AS NUMERIC(8,3)) FROM AUDITORIUM)  SET @CAPACITOFTASK=(SELECT CAST(COUNT (\*) AS NUMERIC(8,3)) FROM AUDITORIUM  WHERE AUDITORIUM\_CAPACITY< @AVGCAPACITY)  SELECT @PROCENT= (100.\*@CAPACITOFTASK)/@QUANTCAPACITY FROM AUDITORIUM  SELECT @ALLCAPACITY 'ОБЩАЯ ВМЕСТИМОСТЬ', @QUANTCAPACITY 'КОЛИЧЕСТВО АУДИТОРИЙ',  @AVGCAPACITY 'СРЕДНЯЯ ВМЕСТИМОСТЬ АУДИТОРИЙ',  @CAPACITOFTASK 'КОЛИЧЕСТВО АУДИТОРИЙ, ВМЕСТИМОСТЬ КОТОРЫХ МЕНЬШЕ СРЕДНЕЙ', @PROCENT 'ПРОЦЕНТ АУДИТОРИЙ С ВМЕСТИМОСТЬЮ МЕНЬШЕ СРЕДНЕЙ'  END  ELSE IF @ALLCAPACITY<200 PRINT 'ВМЕСТИМОСТЬ МЕНЬШЕ 200'  -----3 ВЫВОД ГЛОБАЛЬНЫХ ПЕРЕМЕННЫХ  SELECT @@ROWCOUNT [ROWCOUNT],--- КОЛИЧЕСТВО СТРОК ОБРАБ ПРЕДЫДУЩЕЙ КОМАНДОЙ  @@VERSION [VERSION], ---- ВЕРСИЯ SQL  @@SPID [SPID],----СИСТЕМНЫЙ ИДЕНТИФИКАТОР ПРОЦЕССОРА, НАЗНАЧЕННЫЙ СЕРВЕРУ  @@ERROR [ERROR],----ПОСЛЕДНИЙ КОД ОШИБКИ  @@SERVERNAME [SERVERNAME],----ИМЯ СЕРВЕРА  @@TRANCOUNT [TRANCOUNT],---УРОВЕНЬ ВЛОЖЕННОСТИ ТРАНЗАКЦИЙ  @@FETCH\_STATUS [FETCH\_STATUS],----ПРОВЕРКА РЕЗУЛЬТАТА СЧИТЫВАНИЯ ТРОК РЕЗУЛЬТИРУЮЩЕГО НАБОРА  @@NESTLEVEL [NESTLEVEL];----УРОВЕНЬ ВЛОЖЕННОСТИ ТЕКУЩЕЙ ПРОЦЕДУРЫ  -----4 ЗАДАНИЕ  -----ВЫЧИСЛИТЬ Z  DECLARE @X INT=25, @T INT=10, @Z REAL;  IF @T>@X SET @Z=POWER(SIN(@T),2);  IF @T<@X SET @Z=4.\*(@T+@X);  IF @T=@X SET @Z=1.-EXP(@X-2.);  PRINT 'Z= '+ CAST(@Z AS VARCHAR(12));  -----ПРЕОБРАЗОВАНИЕ ФИО  DECLARE @NAME VARCHAR(10)='АНАСТАСИЯ',@FATHERNAME VARCHAR(11)='ДМИТРИЕВНА', @SECONDNAME VARCHAR(7)='ДРОЖЖА';  PRINT 'ПОЛНОЕ ИМЯ '+CAST(@SECONDNAME AS VARCHAR(7))+' '+CAST(@NAME AS VARCHAR(10))+' '+CAST(@FATHERNAME AS VARCHAR(11));  PRINT 'СОКРАЩЕННОЕ ИМЯ '+CAST(@SECONDNAME AS VARCHAR(7))+' '+CAST(SUBSTRING(@NAME,1,1) AS VARCHAR(1))  +'. '+CAST(SUBSTRING(@FATHERNAME,1,1) AS VARCHAR(1))+'. ';  -----СТУДЕНТЫ ДР  DECLARE @DATEMONTH DATETIME=GETDATE();  SELECT STUDENT.BDAY,STUDENT.NAME,  (select floor(datediff(day, STUDENT.BDAY, @DATEMONTH) / 365.25))AS DОЗРАСТ  FROM STUDENT WHERE MONTH(STUDENT.BDAY)= MONTH(@DATEMONTH)+1;  ----- ДНИ НЕДЕЛИ, КОГДА СТ СДАВАЛИ СУБД  DECLARE @PATTERN VARCHAR(5)='СУБД';  SELECT DISTINCT SUBJECT, DATENAME(WEEKDAY,PDATE) AS [День недели] FROM PROGRESS WHERE PROGRESS.SUBJECT=@PATTERN;  DECLARE @PATTERN2 VARCHAR(5)='КГ';  SELECT DISTINCT SUBJECT, DATENAME(WEEKDAY,PDATE) AS [День недели] FROM PROGRESS WHERE PROGRESS.SUBJECT=@PATTERN2;  ----5 IF....ELSE  DECLARE @QTEACHERS INT=(SELECT COUNT(\*) FROM TEACHER);  IF @QTEACHERS>10  BEGIN  PRINT 'ПРЕПОДАВАТЕЛЕЙ В УНИВЕРЕ ДОСТАТОЧНО';  PRINT 'КОЛИЧЕТВО ПРЕПОДАВАТЕЛЕЙ '+CAST(@QTEACHERS AS VARCHAR(5));  END;  ELSE  PRINT 'ПРЕПОДАВАТЕЛЕЙ В УНИВЕРЕ МАЛО'+' '+'КОЛИЧЕТВО ПРЕПОДАВАТЕЛЕЙ '+CAST(@QTEACHERS AS VARCHAR(5));  ----6 CASE ОЦЕНКИ СТУДЕНТОВ ЗА ЭКЗАМЕН  SELECT CASE  WHEN NOTE BETWEEN 1 AND 3 THEN ' Пересдача'  WHEN NOTE BETWEEN 4 AND 5 THEN ' Минимальные оценки'  WHEN NOTE BETWEEN 6 AND 7 THEN ' Неплохие оценки '  WHEN NOTE BETWEEN 8 AND 9 THEN ' Хорошие оценки '  WHEN NOTE=10 THEN ' Отличные оценки '  END ОЦЕНКИ, COUNT(\*) [КОЛИЧЕСТВО]  FROM dbo.PROGRESS  GROUP BY CASE  WHEN NOTE BETWEEN 1 AND 3 THEN ' Пересдача'  WHEN NOTE BETWEEN 4 AND 5 THEN ' Минимальные оценки'  WHEN NOTE BETWEEN 6 AND 7 THEN ' Неплохие оценки '  WHEN NOTE BETWEEN 8 AND 9 THEN ' Хорошие оценки '  WHEN NOTE=10 THEN ' Отличные оценки '  END;  ----7 CASE ВРЕМЕННАЯ ЛОКАЛЬНАЯ ТАБЛИЦА  CREATE TABLE #MYTABLE(  ID INT IDENTITY(1,1),  ROUND INT,  READY VARCHAR(3)  );  SET NOCOUNT ON;  DECLARE @TEMP INT=0;  WHILE @TEMP<11  BEGIN  INSERT #MYTABLE(ROUND,READY)  VALUES(FLOOR(30000\*RAND()), REPLICATE('ДА',1));  IF (@TEMP< 11)  PRINT @TEMP;  SET @TEMP=@TEMP+1;  END;  SELECT \* FROM #MYTABLE;  DROP TABLE #MYTABLE;  ----8 ОПЕРАТОР RETURN  DECLARE @P INT=25, @TK INT=10, @ZZ REAL;  PRINT 'ZZ=REAL '+ ' ,'+'P='+CAST(@P AS VARCHAR(12))+' ,'+'TK='+CAST(@TK AS VARCHAR(12));  RETURN  IF @TK>@P SET @ZZ=POWER(SIN(@TK),2);  IF @TK<@P SET @ZZ=4.\*(@TK+@P);  IF @TK=@P SET @ZZ=1.-EXP(@P-2.);  PRINT 'ZZ= '+ CAST(@ZZ AS VARCHAR(12));  PRINT @ZZ+25;  ----9 ОПЕРАТОР TRY.. CATCH  DECLARE @M INT = 25, @N INT=12, @K REAL;  BEGIN TRY  IF @M>@N SET @K=(@M+@N)/0;  END TRY  BEGIN CATCH  SELECT ERROR\_MESSAGE()[НАЗВАНИЕ ОШИБКИ], ERROR\_NUMBER()[НОМЕР ОШИБКИ],ERROR\_LINE() [КОД ОШИБКИ], ERROR\_PROCEDURE()[ИМЯ ПРОЦЕДУРЫ],  ERROR\_SEVERITY()[УРОВЕНЬ УГРОЗЫ], ERROR\_STATE()[МЕТКА ОШИБКИ];  END CATCH |
|  |

*Лабораторная работа №10*

Создание и применение индексов

**Индекс** – это объект базы данных, позволяющий **ускорить поиск** в определенной таблице, так как при этом данные организуются в виде сбалансированного бинарного дерева поиска.

Как и любой другой объект базы данных, индекс может быть создан с помощью оператора CREATE, модифицирован с помощью ALTER и удален с помощью оператора DROP. Для одной таблицы возможно построение нескольких индексов.

Индексы бывают кластеризованные, некластеризованные, уникальные, неуникальные и др.

|  |
| --- |
| -----1 ЗАДАНИЕ  use D\_UNIVER;  EXEC sp\_helpindex AUDITORIUM;  EXEC sp\_helpindex AUDITORIUM\_TYPE;  EXEC sp\_helpindex FACULTY;  EXEC sp\_helpindex GROUPS;  EXEC sp\_helpindex PROFESSION;  EXEC sp\_helpindex PULPIT;  EXEC sp\_helpindex STUDENT;  EXEC sp\_helpindex SUBJECT; -----ЗДЕСЬ ЕСТЬ НЕКЛАСТИРИХОВАННЫЙ ИНДЕКС  EXEC sp\_helpindex TEACHER;  EXEC sp\_helpindex TIMETABLE;  DROP TABLE #TEMPTABLE;  CREATE TABLE #TEMPTABLE(  ID INT IDENTITY(1,1),  ROUND INT,  READY VARCHAR(3)  );  SET NOCOUNT ON;  DECLARE @TEMP INT=0;  WHILE @TEMP<1000  BEGIN  INSERT #TEMPTABLE(ROUND,READY)  VALUES(FLOOR(30000\*RAND()), REPLICATE('ДА',1));  IF (@TEMP< 1000)  PRINT @TEMP;  SET @TEMP=@TEMP+1;  END;  SELECT \* FROM #TEMPTABLE WHERE ROUND BETWEEN 1 AND 20000;  ----СТОИМОСТЬ ДАННОГО ПОДДЕРЕВА 0.0066  CHECKPOINT; -----ФИКСАЦИЯ БД  DBCC DROPCLEANBUFFERS;----ОЧИСТКА БУФЕРНОГО КЭША  CREATE CLUSTERED INDEX TEMPTABLE\_CL ON #TEMPTABLE(ROUND ASC)  ----СТОИМОСТЬ ДАННОГО ПОДДЕРЕВА СТАЛА 0.0054  DROP INDEX TEMPTABLE\_CL ON #TEMPTABLE;  -----2 ЗАДАНИЕ СОСТАВНОЙ ИНДЕКС  DROP TABLE #TABLE2;  CREATE TABLE #TABLE2(  ID INT IDENTITY(1,1),  ROUND INT,  READY VARCHAR(3)  );  SET NOCOUNT ON;  DECLARE @TEM INT=0;  WHILE @TEM<10000  BEGIN  INSERT #TABLE2(ROUND,READY)  VALUES(FLOOR(30000\*RAND()), REPLICATE('ДА',1));  IF (@TEM< 10000)  PRINT @TEM;  SET @TEM=@TEM+1;  END;  SELECT \* FROM #TABLE2 ;  ----СТОИМОСТЬ ДАННОГО ПОДДЕРЕВА 0.035  SELECT COUNT(\*) [КОЛИЧЕСТВО СТРОК] FROM #TABLE2 ;  ----СТОИМОСТЬ ДАННОГО ПОДДЕРЕВА 0.035  CREATE INDEX TABLE2\_NONCL ON #TABLE2(ROUND,ID)  SELECT \* FROM #TABLE2 WHERE ROUND BETWEEN 1 AND 20000 AND ID>400 ;  SELECT \* FROM #TABLE2 WHERE ROUND =25060 AND ID>100 ;  DROP INDEX TABLE2\_NONCL ON #TABLE2;  -----3 ЗАДАНИЕ ИНДЕКС ПОКРЫТИЯ  DROP TABLE #TABLE3;  CREATE TABLE #TABLE3(  ID INT IDENTITY(1,1),  ROUND INT,  READY VARCHAR(3)  );  SET NOCOUNT ON;  DECLARE @T INT=0;  WHILE @T<10000  BEGIN  INSERT #TABLE3(ROUND,READY)  VALUES(FLOOR(30000\*RAND()), REPLICATE('БD',1));  IF (@T< 10000)  PRINT @T;  SET @T=@T+1;  END;  SELECT \* FROM #TABLE3 ;  CREATE INDEX TABLE3\_INDEX ON #TABLE3(ROUND)INCLUDE (ID)  SELECT ID FROM #TABLE3 WHERE ROUND>1500;  DROP INDEX TABLE3\_INDEX ON #TABLE3;  -----4 ЗАДАНИЕ ФИЛЬТРУЮЩИЙ ИНДЕКС  --------------------------------  DROP TABLE #EXPLORE;  USE D\_UNIVER;  CREATE TABLE #EXPLORE  (  TKEY INT,  CC INT IDENTITY(1,1),  TFIELD VARCHAR(100)  );  GO  SET NOCOUNT ON;  DECLARE @I INT =0;  WHILE @I<10000  BEGIN  INSERT #EXPLORE(TKEY,TFIELD)  VALUES(FLOOR(30000\*RAND()), REPLICATE('СТРОКА',10));  IF (@I%100=0)  PRINT @I;  SET @I=@I+1;  END;  GO    USE tempdb;  GO  SELECT COUNT(\*) FROM #EXPLORE;  CREATE INDEX #EXPLORE\_TKEY ON #EXPLORE(TKEY);  GO  EXEC sp\_helpindex '#EXPLORE';  SELECT TKEY FROM #EXPLORE WHERE TKEY BETWEEN 15001 AND 19999;----0.0051  SELECT TKEY FROM #EXPLORE WHERE TKEY> 10000 AND TKEY<30000;----0.02  SELECT TKEY FROM #EXPLORE WHERE TKEY=2561;----0.0032  USE tempdb;  GO  CREATE INDEX #EXPLORE\_WHERE ON #EXPLORE(TKEY) WHERE (TKEY >15000 AND TKEY<20000);  GO  EXEC SP\_HELPINDEX '#EXPLORE';  GO  SELECT TKEY FROM #EXPLORE WHERE TKEY> 15001 AND TKEY<19999;----0.0073  GO  DROP INDEX #EXPLORE\_WHERE ON #EXPLORE;  GO  DROP INDEX #EXPLORE\_TKEY ON #EXPLORE;  -----5 ЗАДАНИЕ ФРАГМЕНТАЦИЯ  --------------------------------  USE tempdb;  GO  EXEC sp\_helpindex '#EXPLORE';  -----ДАННЫЕ О ФРАГМЕНТАЦИИ---------  use tempdb;  SELECT NAME [ИНДЕКС], avg\_fragmentation\_in\_percent [Фрагментация (%)] FROM  sys.dm\_db\_index\_physical\_stats(DB\_ID(N'TEMPDB'),  OBJECT\_ID(N'#EX'), NULL, NULL, NULL) ss  JOIN sys.indexes ii on ss.object\_id = ii.object\_id and ss.index\_id = ii.index\_id  WHERE name is not null;  GO  -------БОЛЕЕ 0%  USE tempdb;  GO  ----------------------------------------  CREATE INDEX #EXPLORE\_TKEY ON #EXPLORE(TKEY);  EXEC sp\_helpindex '#EXPLORE';  SELECT NAME [ИНДЕКС], avg\_fragmentation\_in\_percent [Фрагментация (%)] FROM  sys.dm\_db\_index\_physical\_stats(DB\_ID(N'TEMPDB'),  OBJECT\_ID(N'#EX'), NULL, NULL, NULL) ss  JOIN sys.indexes ii on ss.object\_id = ii.object\_id and ss.index\_id = ii.index\_id  WHERE name is not null;  GO  -----ДАННЫЕ О ФРАГМЕНТАЦИИ---------  INSERT TOP(40000) #EXPLORE(TKEY,TFIELD) SELECT TKEY,TFIELD FROM #EXPLORE;  go  SELECT NAME [ИНДЕКС], avg\_fragmentation\_in\_percent [Фрагментация (%)] FROM  sys.dm\_db\_index\_physical\_stats(DB\_ID(N'TEMPDB'),  OBJECT\_ID(N'#EX'), NULL, NULL, NULL) ss  JOIN sys.indexes ii on ss.object\_id = ii.object\_id and ss.index\_id = ii.index\_id  WHERE name is not null;  GO  ALTER INDEX #EXPLORE\_TKEY ON #EXPLORE REORGANIZE;  ALTER INDEX #EXPLORE\_TKEY ON #EXPLORE REBUILD WITH (ONLINE=OFF);  GO  -----ДАННЫЕ О ФРАГМЕНТАЦИИ---------  SELECT NAME [ИНДЕКС], avg\_fragmentation\_in\_percent [Фрагментация (%)] FROM  sys.dm\_db\_index\_physical\_stats(DB\_ID(N'TEMPDB'),  OBJECT\_ID(N'#EX'), NULL, NULL, NULL) ss  JOIN sys.indexes ii on ss.object\_id = ii.object\_id and ss.index\_id = ii.index\_id  WHERE name is not null;  GO  -----6 ЗАДАНИЕ FILLFACTOR---------  DROP index #EXPLORE\_TKEY on #EXPLORE;  CREATE index #EXPLORE\_TKEY on #EXPLORE(TKEY)  with (fillfactor = 65);  --После добавления строк в таблицу можно оценить уровень фрагментации:  INSERT top(50)percent INTO #EXPLORE(TKEY, TFIELD)  SELECT TKEY, TFIELD FROM #EXPLORE;  SELECT name [Индекс], avg\_fragmentation\_in\_percent [Фрагментация (%)]  FROM sys.dm\_db\_index\_physical\_stats(DB\_ID(N'TEMPDB'),  OBJECT\_ID(N'#EXPLORE'), NULL, NULL, NULL) ss JOIN sys.indexes ii  ON ss.object\_id = ii.object\_id and ss.index\_id = ii.index\_id  WHERE name is not null;  GO |

*Лабораторная работа № 11*

Обработка результатов запросов с помощью курсоров

**Курсор** является программной конструкцией, которая дает возможность пользователю обрабатывать строки результирующего набора запись за записью. Курсоры бывают *локальные* и *глобальные* (по умолчанию), *статические* и *динамические* (по умолчанию).

|  |
| --- |
| USE D\_UNIVER;  ---1---РАЗРАБОТАТЬ СЦЕНАРИЙ, ФОРМИРУЮЩИЙ СПИСОК ДИСЦИПЛИН НА КАФЕДРЕ ИСИТ. В ОТЧЕТ ДОЛЖНЫ БЫТЬ ВЫВЕДЕНЫ КРАТКИЕ НАЗВАНИЯ  -------(ПОЛЕ SUBJECT) ИЗ ТАБЛИЦЫ SUBJECT В ОДНУ СТРОКУ ЧЕРЕЗ ЗАПЯТУЮ.  -------ИСПОЛЬЗОВАТЬ ВСТРОЕННУЮ ФУНКЦИЮ RTRIM.  DECLARE @SUBJ CHAR(30), @S CHAR(300) = '';  DECLARE SUBJISIT CURSOR FOR SELECT SUBJECT FROM SUBJECT WHERE PULPIT = 'ИСИТ';  OPEN SUBJISIT;  FETCH SUBJISIT INTO @SUBJ;  PRINT '';  WHILE @@FETCH\_STATUS = 0  BEGIN  SET @S = RTRIM(@SUBJ) + ',' + @S;  FETCH SUBJISIT INTO @SUBJ;  END;  PRINT @S;  CLOSE SUBJISIT;  USE D\_UNIVER;  GO  ---2---РАЗРАБОТАТЬ СЦЕНАРИЙ, ДЕМОНСТРИРУЮЩИЙ ОТЛИЧИЕ ГЛОБАЛЬНОГО КУРСОРА ОТ ЛОКАЛЬНОГО НА ПРИМЕРЕ БАЗЫ ДАННЫХ X\_UNIVER.  DECLARE LOCCUR CURSOR LOCAL FOR SELECT SUBJECT, PULPIT FROM SUBJECT;  DECLARE @SS CHAR(20), @P CHAR(10);  OPEN LOCCUR;  FETCH LOCCUR INTO @SS, @P;  PRINT '1 ' + @SS + @P;  go  DECLARE @SS2 CHAR(20), @P2 CHAR(10);  FETCH LOCCUR INTO @SS2, @P2;  PRINT '2 ' + @SS2 + @P2;  CLOSE LOCUR;  GO  DECLARE GLOB\_CUR CURSOR GLOBAL FOR SELECT SUBJECT, PULPIT FROM SUBJECT;  DECLARE @SS CHAR(20), @P CHAR(10);  OPEN GLOB\_CUR;  FETCH GLOB\_CUR INTO @SS, @P;  PRINT '1 ' + @SS + @P;  GO  DECLARE @SS CHAR(20), @P CHAR(10);  FETCH GLOB\_CUR INTO @SS, @P;  PRINT '2 ' + @SS + @P;  CLOSE GLOB\_CUR;  DEALLOCATE GLOB\_CUR;  GO  --3--ОТЛИЧИЕ СТАТИЧЕСКИХ ОТ ДИНАМИЧЕСКИХ  DECLARE @SUBJJ CHAR(10), @SUBJ\_NAME CHAR(50), @PP CHAR(10);  DECLARE SUBJ CURSOR LOCAL STATIC FOR SELECT SUBJECT, SUBJECT\_NAME, PULPIT FROM SUBJECT;  OPEN SUBJ;  PRINT 'ROWS: ' + CAST(@@CURSOR\_ROWS AS VARCHAR(10));  INSERT SUBJECT (SUBJECT, SUBJECT\_NAME, PULPIT)  VALUES ('(((((jkl', 'CRYINhjkG', 'ИСИТ');  FETCH SUBJ INTO @SUBJJ, @SUBJ\_NAME, @PP;  WHILE @@FETCH\_STATUS = 0  BEGIN  PRINT @SUBJJ + ' - ' + @SUBJ\_NAME + ' - ' + @PP;  FETCH SUBJ INTO @SUBJJ, @SUBJ\_NAME, @PP;  END;  CLOSE SUBJ  DELETE SUBJECT WHERE SUBJECT = 'MEME'  DECLARE @SUBJJ CHAR(10), @SUBJ\_NAME CHAR(50), @PP CHAR(10);  DECLARE SUBJ CURSOR LOCAL DYNAMIC FOR SELECT SUBJECT, SUBJECT\_NAME, PULPIT FROM SUBJECT;  OPEN SUBJ;  PRINT 'ROWS: ' + CAST(@@CURSOR\_ROWS AS VARCHAR(10));  INSERT SUBJECT (SUBJECT, SUBJECT\_NAME, PULPIT)  VALUES ('!\_!', 'KEEP SMILING', 'ИСИТ');  FETCH SUBJ INTO @SUBJJ, @SUBJ\_NAME, @PP;  WHILE @@FETCH\_STATUS = 0  BEGIN  PRINT @SUBJJ + ' - ' + @SUBJ\_NAME + ' - ' + @PP;  FETCH SUBJ INTO @SUBJJ, @SUBJ\_NAME, @PP;  END;  CLOSE SUBJ  ---4---НАВИГАЦИЯ(SCROLL)  DECLARE @SUBJJ CHAR(10), @SUBJ\_NAME CHAR(50), @PP CHAR(10), @RN INT;  DECLARE SUBJ CURSOR LOCAL DYNAMIC SCROLL  FOR SELECT ROW\_NUMBER() OVER (ORDER BY SUBJECT) NO, SUBJECT, SUBJECT\_NAME, PULPIT FROM SUBJECT;  OPEN SUBJ;  FETCH SUBJ INTO @RN, @SUBJJ, @SUBJ\_NAME, @PP;  PRINT 'BEGINNING: ' + CAST(@RN AS VARCHAR(3)) + @SUBJJ + ' - ' + @SUBJ\_NAME + ' - ' + @PP;  FETCH NEXT FROM SUBJ INTO @RN, @SUBJJ, @SUBJ\_NAME, @PP;  PRINT 'SECOND: ' + CAST(@RN AS VARCHAR(3)) + @SUBJJ + ' - ' + @SUBJ\_NAME + ' - ' + @PP;  FETCH FIRST FROM SUBJ INTO @RN, @SUBJJ, @SUBJ\_NAME, @PP;  PRINT 'FIRST: ' + CAST(@RN AS VARCHAR(3)) + @SUBJJ + ' - ' + @SUBJ\_NAME + ' - ' + @PP;  FETCH RELATIVE 5 FROM SUBJ INTO @RN, @SUBJJ, @SUBJ\_NAME, @PP;  PRINT 'SIXTH: ' + CAST(@RN AS VARCHAR(3)) + @SUBJJ + ' - ' + @SUBJ\_NAME + ' - ' + @PP;  FETCH LAST FROM SUBJ INTO @RN, @SUBJJ, @SUBJ\_NAME, @PP;  PRINT 'LAST: ' + CAST(@RN AS VARCHAR(3)) + @SUBJJ + ' - ' + @SUBJ\_NAME + ' - ' + @PP;  FETCH ABSOLUTE -3 FROM SUBJ INTO @RN, @SUBJJ, @SUBJ\_NAME, @PP;  PRINT 'THIRD FROM THE END: ' + CAST(@RN AS VARCHAR(3)) + @SUBJJ + ' - ' + @SUBJ\_NAME + ' - ' + @PP;  FETCH PRIOR FROM SUBJ INTO @RN, @SUBJJ, @SUBJ\_NAME, @PP;  PRINT 'FOURTH FROM THE END: ' + CAST(@RN AS VARCHAR(3)) + @SUBJJ + ' - ' + @SUBJ\_NAME + ' - ' + @PP;  CLOSE SUBJ  ---5--- SELECT-ЗАПРОС, С ПОМОЩЬЮ КОТОРОГО ИЗ ТАБЛИЦЫ PROGRESS УДАЛЯЮТСЯ СТРОКИ,  ---СОДЕРЖАЩИЕ ИНФОРМАЦИЮ О СТУДЕНТАХ, ПОЛУЧИВШИХ ОЦЕНКИ НИЖЕ 4 (ИСПОЛЬЗОВАТЬ ОБЪЕДИНЕНИЕ ТАБЛИЦ PROGRESS, STUDENT, GROUPS).  ---РАЗРАБОТАТЬ SELECT-ЗАПРОС, С ПОМОЩЬЮ КОТОРОГО В ТАБЛИЦЕ PROGRESS ДЛЯ СТУДЕНТА С КОНКРЕТНЫМ НОМЕРОМ  ---IDSTUDENT КОРРЕКТИРУЕТСЯ ОЦЕНКА (УВЕЛИЧИВАЕТСЯ НА ЕДИНИЦУ).  go  INSERT INTO PROGRESS(SUBJECT, IDSTUDENT,PDATE, NOTE)  VALUES ('СУБД', '1024', '01.12.2013', 2);  SELECT \* FROM PROGRESS  DECLARE @STUDENT\_ID INT, @NOTE INT;  DECLARE NOTES CURSOR LOCAL DYNAMIC  FOR SELECT PROGRESS.IDSTUDENT, NOTE FROM PROGRESS  JOIN STUDENT ON STUDENT.IDSTUDENT = PROGRESS.IDSTUDENT  JOIN GROUPS ON STUDENT.IDGROUP = GROUPS.IDGROUP  WHERE NOTE < 4 FOR UPDATE;  OPEN NOTES;  FETCH NOTES INTO @STUDENT\_ID, @NOTE;  DELETE PROGRESS WHERE CURRENT OF NOTES;  CLOSE NOTES;  go  SELECT \* FROM PROGRESS  DECLARE @S INT, @N INT;  DECLARE CUR CURSOR LOCAL DYNAMIC  FOR SELECT PROGRESS.IDSTUDENT, NOTE FROM PROGRESS WHERE IDSTUDENT=1054;  OPEN CUR;  FETCH CUR INTO @S, @N;  UPDATE PROGRESS SET NOTE = NOTE + 1 WHERE CURRENT OF CUR;  CLOSE CUR;  SELECT \* FROM PROGRESS |
|  |

*Лабораторная работа №12*

Особенности использования транзакций

**Транзакция** − это механизм базы данных, позволяющий таким образом объединять несколько операторов, изменяющих базу данных, чтобы при выполнении этой совокупности операторов они или все выполнились или все не выполнились.

Основные свойства транзакции: *атомарность* (операторы изменения БД, включенные в транзакцию, либо выполнятся все, либо не выполнится ни один); *согласованность* (транзакция должна фиксировать новое согласованное состояние БД); *изолированность* (отсутствие взаимного влияния параллельных транзакций на результаты их выполнения); *долговечность* (изменения в БД, выполненные и зафиксированные транзакцией, могут быть отменены только с помощью новой транзакции).

|  |
| --- |
| USE CAKE;  --1-- РЕЖИМ НЕЯВНОЙ ТРАНЗАКЦИИ  -- транзакция начинается, если выполняется один из следующих операторов:  -- CREATE, DROP; ALTER TABLE; INSERT, DELETE, UPDATE, SELECT, TRUNCATE TABLE; OPEN, FETCH; GRANT; REVOKE  -- неявная транзакция продолжается до тех пор, пока не будет выполнен COMMIT или ROLLBACK  set nocount on  if exists (select \* from SYS.OBJECTS  where OBJECT\_ID=object\_id(N'DBO.SHOPS'))  drop table SHOPS;  declare @c int, @flag char = 'c';  SET IMPLICIT\_TRANSACTIONS ON -- вкл режим неявной транзакции  create table SHOPS(K int );  insert SHOPS values (1),(2),(3),(4),(5);  set @c = (select count(\*) from SHOPS);  print 'кол-во строк в SHOPS: ' + cast(@c as varchar(2));  if @flag = 'c' commit -- фиксация  else rollback; -- откат  SET IMPLICIT\_TRANSACTIONS OFF -- действует режим автофиксации  if exists (select \* from SYS.OBJECTS  where OBJECT\_ID= object\_id(N'DBO.SHOPS')) print 'таблица SHOPs есть';  else print 'таблицы SHOPS нет'  --2-- СВОЙСТВО АТОМАРНОСТИ ЯВНОЙ ТРАНЗАКЦИИ  --выполнение либо всех операторов, либо ни одного  USE D\_UNIVER;  select \* from PULPIT;  begin try  begin tran -- начало явной транзакции  DELETE PULPIT WHERE PULPIT='ЛКиП';  insert PULPIT values ('ТБ', 'Телевизионные бои', 'ИТ');  insert PULPIT values ('ТБ2', 'Телевизионные бои2', 'ИТ');  commit tran; -- фиксация транзакции  end try  begin catch  print 'ошибка: '+ case  when error\_number() = 2627 and patindex('%FACULTY\_PK%', error\_message()) > 0 then 'дублирование ' --позиция 1-го вхождения  else 'неизвестная ошибка: '+ cast(error\_number() as varchar(5))+ error\_message()  end;  if @@trancount > 0 rollback tran; -- ур.вложенности тр.>0, транз не завершена  end catch;  select \* from PULPIT;  DELETE PULPIT WHERE PULPIT = 'ТБ';  DELETE PULPIT WHERE PULPIT = 'ТБ2';  select \* from PULPIT;  --3-- ОПЕРАТОР SAVETRAN  -- если транзакция сост из неск независ блоков операторов T-SQL, то исп.  -- SAVE TRANSACTION, формир контр.точку транзакции  declare @point varchar(32);  begin try  begin tran  delete progress where note = 10;  set @point = 'p1' save tran @point;  insert progress values ('дыыа', 1021, '10.12.20', 4);  set @point = 'p2' save tran @point;  insert progress values ('СУБД', 1012, '10.12.20', 10);  commit tran;  end try  begin catch  print 'error: ' + error\_message();  if @@trancount > 0  begin  print 'control point: ' + @point;  rollback tran @point;  commit tran;  end;  end catch  select \* from progress  delete progress where pdate = '10.12.20'  select \* from progress where pdate = '10.12.20'  --4.  --Сценарий A - явную транзакцию с уровнем изолированности READ UNCOMMITED,  --кот. допуск неподтвержд, неповтор. и фантомное чтение  --сценарий B – явную транзакцию с уровнем изолированности READ COMMITED (по умолч)  ---A---  set transaction isolation level READ UNCOMMITTED  begin transaction  ---t1---  select @@SPID, 'insert progress' 'результат', \* from progress where subject = 'СУБД';  select @@SPID,'update student' 'результат', name, bday  from student where bday = '3023-07-12';  commit;  ---t2---  ---B---  set transaction isolation level READ COMMITTED  begin transaction  select @@SPID  insert progress values ('СУБД', 1010, '10.12.20', 8);  update student set bday = '20.12.1995' where bday = '1994-07-12'  ---t1---  ---t2---  rollback;  --5. Разработать два сцена-рия: A и B на примере базы данных X\_BSTU.  --Сценарий A представляет собой явную транзакцию с уровнем изолированности READ COMMITED.  --Сценарий B – явную транзакцию с уровнем изолированности READ COMMITED.  --Сценарий A должен демонстрировать, что уровень READ COMMITED не допускает неподтвержденного чтения,  --но при этом возможно неповторяющееся и фантомное чтение.  -- A ---  set transaction isolation level READ COMMITTED  begin transaction  select count(\*) from progress where subject = 'СУБД';  -------------------------- t1 ------------------  -------------------------- t2 -----------------  select 'update progress' 'результат', count(\*) from progress where subject = 'СУБД';  commit;  --- B ---  begin transaction  -------------------------- t1 --------------------  update progress set subject = 'СУБД' where subject = 'КГ'  -------------------------- t2 --------------------  rollback;  ----  update progress set subject = 'КГ' where PDATE = '01.12.2013'  select \* from progress  --6. Сценарий A представляет собой явную транзакцию с уровнем  --изолированности REPEATABLE READ. Сценарий B – явную транзакцию с уровнем изолированности READ COMMITED.  ---A---  set transaction isolation level REPEATABLE READ  begin tran  select note from progress where subject = 'КГ';  -------------------------- t1 ------------------  -------------------------- t2 -----------------  select case  when note = 4 then 'insert progress' else ' '  end 'результат', note from progress where subject = 'КГ';  commit;  --- B ---  begin tran  -------------------------- t1 --------------------  insert progress values ('КГ', 1033, '01.12.2014', 4);  commit;  -------------------------- t2 --------------------  --7.  ---A---  set transaction isolation level SERIALIZABLE  begin transaction  delete progress where note = 5;  insert progress values ('КГ', 1033, '01.12.2014', 5);  update progress set note = 5 where subject = 'КГ';  select note from progress where subject = 'КГ';  -------------------------- t1 -----------------  select note from progress where subject = 'КГ';  -------------------------- t2 ------------------  rollback  --- B ---  begin transaction  delete progress where note = 5;  insert progress values ('КГ', 1033, '01.12.2014', 5);  update progress set note = 5 where subject = 'КГ';  select note from progress where subject = 'КГ';  -------------------------- t1 --------------------  commit;  select note from progress where subject = 'КГ';  -------------------------- t2 --------------------  -----8----------  begin tran  insert subject values ('DOOO', 'DOOO', 'ИСиТ');  begin tran  update progress set SUBJECT = 'DOOO' where idstudent = 1021;  commit  if @@trancount < 0  rollback;  select \* from subject  select \* from progress  delete from progress where subject = 'DOOO'  delete from subject where subject = 'DOOO' |

*Лабораторная работа №13*

Разработка хранимых процедур

**Хранимая процедура** – это поименованный код на языке Transact-SQL. Хранимая процедура может быть создана с помощью CREATE, изменена с помощью ALTER и удалена с помощью оператора DROP. Процедура может принимать входные и формировать выходные параметры. Результатом ее выполнения может быть целочисленное значение, которое возвращается к точке вызова оператором RETURN, либо один или более результирующих наборов, сформированных операторами SELECT, либо содержимое стандартного выходного потока, полученного при выполнении операторов PRINT.

Вызов процедуры осуществляется оператором EXECUTE (EXEC).

В хранимых процедурах допускается применение основных DDL, DML и TCL-операторов, конструкций TRY/CATCH, курсоров, временных таблиц.

|  |
| --- |
| USE D\_UNIVER;  go  --1. хранимая процЕДУРА РSUBJECT  --к точке вызова процедура д. возвр кол-во строк, выведенных в рез.набор  --  CREATE procedure PSUBJECT  as begin  DECLARE @n int = (SELECT count(\*) from SUBJECT);  SELECT SUBJECT [КОД], SUBJECT\_NAME [ДИСЦИПЛИНА], PULPIT [КАФЕДРА] from SUBJECT;  return @n;  end;  DECLARE @k int;  EXEC @k = PSUBJECT; -- вызов процедуры  print 'Количество предметов: ' + cast(@k as varchar(3));  go  --DROP procedure PSUBJECT;  ---2 принимала параметры @p (вх. - код кафедры), @c (вых. - кол-во)  ALTER procedure PSUBJECT @p varchar(20), @c nvarchar(2) output  as begin  SELECT \* from SUBJECT where SUBJECT = @p;  set @c = cast(@@rowcount as nvarchar(2));  end;  go  DECLARE @nc int, @no nvarchar(2);  EXEC @nc = PSUBJECT @p = '!\_!', @c = @no output;  print 'Количество предметов: ' + @no;  go  ---3 принимала параметры @p (вх. - код кафедры), врееменная таблица,  ALTER procedure PSUBJECT @p varchar(20)  as begin  SELECT \* from SUBJECT where SUBJECT = @p;  end;  CREATE table #SUBJECT  (  Код\_предмета varchar(20),  Название\_предмета varchar(100),  Кафедра varchar(20)  );  INSERT #SUBJECT EXEC PSUBJECT @p = '!\_!';  INSERT #SUBJECT EXEC PSUBJECT @p = 'КГ';  SELECT \* from #SUBJECT;  go  drop table #SUBJECT;  --4. Процедура 4 параметра (значения столбцов), доб. строку в табл.AUDITORIUM  go  CREATE procedure PAUDITORIUM\_INSERT  @a char(20),  @n varchar(50),  @c int = 0,  @t char(10)  as begin  begin try  INSERT into AUDITORIUM(AUDITORIUM, AUDITORIUM\_TYPE, AUDITORIUM\_CAPACITY, AUDITORIUM\_NAME)  values(@a, @n, @c, @t);  return 1;  end try  begin catch  print 'Номер ошибки: ' + cast(error\_number() as varchar(6));  print 'Сообщение: ' + error\_message();  print 'Уровень: ' + cast(error\_severity() as varchar(6));  print 'Метка: ' + cast(error\_state() as varchar(8));  print 'Номер строки: ' + cast(error\_line() as varchar(8));  if error\_procedure() is not null  print 'Имя процедуры: ' + error\_procedure();  return -1;  end catch;  end;  DECLARE @rc int;  EXEC @rc = PAUDITORIUM\_INSERT @a = '201-3', @n = 'ЛК-К', @c = 200, @t = '20-3';  print 'Код ошибки: ' + cast(@rc as varchar(3));  go  Select \* from AUDITORIUM;  delete AUDITORIUM where AUDITORIUM='20-3';  ---5 форм. список дисциплин на отд.кафедре  ---краткие назв. в строку через зап.(RTRIM)  go  CREATE procedure SUBJECT\_REPORT @p char(10)  as begin  DECLARE @rc int;  begin try  DECLARE @sb char(10), @r varchar(100) = '';  DECLARE sbj CURSOR for  SELECT SUBJECT from SUBJECT where PULPIT = @p;  if not exists(SELECT SUBJECT from SUBJECT where PULPIT = @p)  raiserror('Ошибка', 11, 1);  else  OPEN sbj;  fetch sbj into @sb;  print 'Предметы: ';  while @@fetch\_status = 0  begin  set @r = rtrim(@sb) + ', ' + @r;  set @rc = @rc + 1;  fetch sbj into @sb;  end  print @r;  CLOSE sbj;  return @rc;  end try  begin catch  print 'Ошибка в параметрах'  if error\_procedure() is not null  print 'Имя процедуры: ' + error\_procedure();  return @rc;  end catch;  end;  DECLARE @k2 int;  EXEC @k2 = SUBJECT\_REPORT @p ='ИСиТ';  print 'Количество предметов: ' + cast(@k2 as varchar(3));  go  drop procedure SUBJECT\_REPORT;  --6. Проц. доб. 2 строки: в табл.AUD\_TYPE(@t, @tn) + путем вызова PAUD\_INSERT  --все в рамках явной транзакции с ур.изол.SERIALIZABLE  go  CREATE procedure PAUDITORIUM\_INSERTX  @a char(20),  @n varchar(50),  @c int = 0,  @t char(10),  @tn varchar(50) --доп., для ввода в AUD\_TYPEAUD\_TYPENAME  as begin  DECLARE @rc int = 1;  begin try  set transaction isolation level serializable;  begin tran  INSERT into AUDITORIUM\_TYPE(AUDITORIUM\_TYPE, AUDITORIUM\_TYPENAME)  values(@n, @tn);  EXEC @rc = PAUDITORIUM\_INSERT @a, @n, @c, @t;  commit tran;  return @rc;  end try  begin catch  print 'Номер ошибки: ' + cast(error\_number() as varchar(6));  print 'Сообщение: ' + error\_message();  print 'Уровень: ' + cast(error\_severity() as varchar(6));  print 'Метка: ' + cast(error\_state() as varchar(8));  print 'Номер строки: ' + cast(error\_line() as varchar(8));  if error\_procedure() is not null  print 'Имя процедуры: ' + error\_procedure();  if @@trancount > 0 rollback tran ;  return -1;  end catch;  end;  DECLARE @k3 int;  EXEC @k3 = PAUDITORIUM\_INSERTX '1255-5', @n = 'ЛЮТ', @c = 105, @t = '1255-5', @tn = 'Лаборатория';  print 'Код ошибки: ' + cast(@k3 as varchar(3));  Select \* from AUDITORIUM;  Select \* from AUDITORIUM\_TYPE;  delete AUDITORIUM where AUDITORIUM='1255-5';  delete AUDITORIUM\_TYPE where AUDITORIUM\_TYPE='ЛЮТ';  go  drop procedure PAUDITORIUM\_INSERTX; |
|  |

*Лабораторная работа №14*

Разработка и использование функций

**Функция** – это объект БД, представляющий собой поименованный код T-SQL. Для создания, удаления и изменения функций надо использовать операторы CREATE, DROP и ALTER соответственно. Отличие функций от хранимых процедур в ограничениях, накладываемых на код функции, в форме представления результата работы, а также в способе вызова. В функции не допускается применение DDL-операторов, DML-операторов, изменяющих БД (INSERT, DELETE, UPDATE), конструкций TRY/CATCH, а также использование транзакций. Результатом выполнения функции является возвращаемое к точке вызова значение. Если функция возвращает единственное значение (число, строка, дата, время и пр.), то она называется *скалярной*. Функция, возвращающая таблицу, называется *табличной*. В зависимости от структуры кода, различают *встроенные* функции и *многооператорные* табличные функции.

|  |
| --- |
| USE D\_UNIVER;  ---скалярные функции(выводят в качестве результата скаляр)  ---таблчные функции(возвращает таблицу)  ---встроенные (Встроенные табличные функции возвращают к точке вызова результирующий набор  -------------------SELECT-запроса и по своим возможностям очень напоминают представления.)  ---многооператорные(спользуется более сложная логика)  --1 Скалярная функция подсчета количества студентов по заданному факультету  go  create function COUNT\_STUDENTS(@faculty nvarchar(20)) returns int  as begin  declare @rc int = 0;  set @rc = (  SELECT count(IDSTUDENT) from STUDENT join GROUPS  on STUDENT.IDGROUP = GROUPS.IDGROUP  join FACULTY  on GROUPS.FACULTY = FACULTY.FACULTY  where FACULTY.FACULTY = @faculty);  return @rc;  end;  go  declare @n int = dbo.COUNT\_STUDENTS('ИТ');  print 'Количество студентов: ' + cast(@n as varchar(4));  --drop function COUNT\_STUDENTS;  -- дополнительный параметр @prof специальность  go  alter function COUNT\_STUDENTS(@faculty varchar(20) = null, @prof varchar(20) = null) returns int  as begin  declare @rc int = 0;  set @rc = (  SELECT count(IDSTUDENT) from FACULTY inner join GROUPS  on FACULTY.FACULTY = GROUPS.FACULTY  inner join STUDENT  on GROUPS.IDGROUP = STUDENT.IDGROUP  where FACULTY.FACULTY = @faculty and GROUPS.PROFESSION = @prof);  return @rc;  end;  go  declare @n int = dbo.COUNT\_STUDENTS('ХТиТ', '1-36 01 08');  print 'Количество студентов: ' + cast(@n as varchar(4));  --2. Скалярную функцию FSUBJECTS, парам. @p (код кафедры = SUBJECT.PULPIT)  -- возвр. строку с пеерчнем дисциплин  go  create function FSUBJECTS(@p varchar(20)) returns varchar(300)  as begin  declare @sb varchar(10), @s varchar(100) = '';  declare sbj cursor local static  for select distinct SUBJECT from SUBJECT  where PULPIT like @p;  open sbj;  fetch sbj into @sb;  while @@FETCH\_STATUS = 0  begin  set @s = @s + RTRIM(@sb) + ', ';  fetch sbj into @sb;  end;  return @s  end;  go  select distinct PULPIT, dbo.FSUBJECTS(PULPIT)[Дисциплины] from SUBJECT;  --3. Табличная ф., парам: код фак + код кафедры  -- если оба парам NULL, возвр. список всех кафедр на фак  -- если второй NULL, возвр. все кафедры зад. фак  -- если первый NULL, возвр. строку, соотв-щую зад. кафедре  -- если оба не NULL, возвр. строку, соотв-щую зад. кафедре на зад. фак  -- если нельзя сформир. строки, возвр. пустой рез.набор  go  create function FFACPUL(@f varchar(20), @p varchar(20)) returns table  as return  select FACULTY.FACULTY, PULPIT.PULPIT from FACULTY left outer join PULPIT  on FACULTY.FACULTY = PULPIT.FACULTY  where FACULTY.FACULTY = ISNULL(@f, FACULTY.FACULTY) and ---если первый параметр равен null то возвращается значение, записанное 2 в скобках  PULPIT.PULPIT = ISNULL(@p, PULPIT.PULPIT);  go  select \* from dbo.FFACPUL(null, null);  select \* from dbo.FFACPUL('ИТ', null);  select \* from dbo.FFACPUL(null, 'ЛВ');  select \* from dbo.FFACPUL('ИТ', 'ИСиТ');  --4. Скалярная ф., один парам (код кафедры)  -- возвр. кол-во преподов на зад.кафедре  -- если (NULL), возвр. общее кол-во преподавов  go  create function FCTEACHER(@p varchar(20)) returns int  as begin  declare @rc int = (select count(TEACHER) from TEACHER where PULPIT = ISNULL(@p, PULPIT));  return @rc;  end;  --drop function FCTEACHER  --drop function COUNT\_GROUP  --drop function COUNT\_PULPITs  --drop function COUNT\_PROFESSION  --drop function FACULTY\_REPORTs  go  select PULPIT, dbo.FCTEACHER(PULPIT)[Количество преподавателей] from TEACHER;  select dbo.FCTEACHER(null)[Общее количество преподавателей];  ----------многооператорная табличная функция------------  go  create function COUNT\_GROUP (@faculty varchar(20)) returns int  as begin  declare @rc int=0;  set @rc =(select count(\*) from GROUPS where FACULTY=@faculty);  return @rc;  end;  go  go  create function COUNT\_PULPITs (@faculty varchar(20)) returns int  as begin  declare @rc int=0;  set @rc =(select count(\*) from PULPIT where FACULTY=@faculty);  return @rc;  end;  go  create function COUNT\_PROFESSION(@faculty varchar(20)) returns int  as begin  declare @rc int=0;  set @rc =(select count(\*) from PROFESSION  where FACULTY=@faculty);  return @rc;  end;  go  go  ------создание табличной функции  create function FACULTY\_REPORTs(@c int) returns @fr table  ( [Факультет] varchar(50), [Количество кафедр] int, [Количество групп] int,  [Количество студентов] int, [Количество специальностей] int )  as begin  declare cc CURSOR static for  select FACULTY from FACULTY where dbo.COUNT\_STUDENTS(FACULTY) > @c;  declare @f varchar(30);  open cc;  fetch cc into @f;  while @@fetch\_status = 0  begin  insert @fr values( @f, dbo.COUNT\_PULPITs(@f),  dbo.COUNT\_GROUP( @f), dbo.COUNT\_STUDENTS(@f),  dbo.COUNT\_PROFESSION( @f) );  fetch cc into @f;  end;  return;  end;  go  -----------------------------  select \* from dbo.FACULTY\_REPORTs(20);  go |
|  |

*Лабораторная работа №15*

Применение триггеров

**Триггер** – это особый вид хранимой процедуры, предназначенной для обработки событий в БД. Поддерживается два типа триггеров: *DDL-триггеры* и *DML-триггеры*. Для каждого типа определено свое семейство событий, обработку которых триггер этого типа может выполнять.

**DML-**триггеры бывают двух типов: AFTER-триггеры и INSTEAD OF-триггеры. Триггеры типа AFTER исполняются *после* выполнения оператора, вызвавшего соответствующее событие. При этом создаются автоматически две псевдотаблицы INSERTED и DELETED.

Триггер типа INSTEAD OF выполняется *вместо* оператора, вызвавшего соответствующее событие. Выполнение INSTEAD OF триггера предшествует проверке установленных для таблицы ограничений целостности.

|  |
| --- |
| USE D\_UNIVER;  ----Триггер – это особый вид хранимой процедуры, предназначенной для обработки событий в БД.  --1. созд.табл. для добав строк триггерами  drop table TR\_AUDIT  go  create table TR\_AUDIT  (  ID int identity,  STMT varchar(20)  check (STMT in ('INS', 'DEL', 'UPD')),  TRNAME varchar(50),  CC varchar(300)  )  --drop trigger TR\_TEACHER\_INS  ------Триггеры типа AFTER исполняются после выполнения оператора, вызвавшего соответствующее событие.-----  go  create trigger TR\_TEACHER\_INS  on TEACHER after INSERT  as  declare @a1 char(10), @a2 varchar(100), @a3 char(1), @a4 char(20), @in varchar(300);  print 'Вставка';  set @a1 = (select TEACHER from INSERTED);  set @a2= (select TEACHER\_NAME from INSERTED);  set @a3= (select GENDER from INSERTED);  set @a4 = (select PULPIT from INSERTED);  set @in = @a1+' '+ @a2 +' '+ @a3+ ' ' +@a4;  insert into TR\_AUDIT(STMT, TRNAME, CC)  values('INS', 'TR\_TEACHER\_INS', @in);  return;  go  insert into TEACHER values('ИВНВс', 'Ивановс', 'м', 'ИСиТ');  insert into TEACHER values('ДМНГс', 'Димонговс', 'м', 'ИСиТ');  insert into TEACHER values('ЛМНСс', 'Ломоносовс', 'м', 'ИСиТ');  select \* from TR\_AUDIT  --delete from TEACHER where TEACHER='ИВНВ';  -----2 задание удаление-----------  go  create trigger TR\_TEACHER\_DEL  on TEACHER after DELETE  as  declare @a1 char(10), @a2 varchar(100), @a3 char(1), @a4 char(20), @in varchar(300);  print 'Удаление';  set @a1 = (select TEACHER from DELETED);  set @a2= (select TEACHER\_NAME from DELETED);  set @a3= (select GENDER from DELETED);  set @a4 = (select PULPIT from DELETED);  set @in = @a1+' '+ @a2 +' '+ @a3+ ' ' +@a4;  insert into TR\_AUDIT(STMT, TRNAME, CC)  values('DEL', 'TR\_TEACHER\_DEL', @in);  return;  go  delete TEACHER where TEACHER='ИВНВс'  select \* from TR\_AUDIT  --drop trigger TR\_TEACHER\_DEL  ----3 обновление-------  --drop trigger TR\_TEACHER\_UPD  go  create trigger TR\_TEACHER\_UPD  on TEACHER after UPDATE  as  declare @a1 char(10), @a2 varchar(100), @a3 char(1), @a4 char(20), @in varchar(300);  declare @ins int = (select count(\*) from inserted),  @del int = (select count(\*) from deleted);  print 'Обновление';  set @a1 = (select TEACHER from INSERTED);  set @a2= (select TEACHER\_NAME from INSERTED);  set @a3= (select GENDER from INSERTED);  set @a4 = (select PULPIT from INSERTED);  set @in = @a1+' '+ @a2 +' '+ @a3+ ' ' +@a4;  set @a1 = (select TEACHER from deleted);  set @a2= (select TEACHER\_NAME from DELETED);  set @a3= (select GENDER from DELETED);  set @a4 = (select PULPIT from DELETED);  set @in =@in + '' + @a1+' '+ @a2 +' '+ @a3+ ' ' +@a4;  insert into TR\_AUDIT(STMT, TRNAME, CC)  values('UPD', 'TR\_TEACHER\_UPD', @in);  return;  go  update TEACHER set GENDER = 'ж' where TEACHER='БЗБРДВ';  select \* from TR\_AUDIT;  delete from TR\_AUDIT where STMT = 'UPD'  ----4 всвё вместе і апдейт і деліт і інсерт---------------  go  --drop trigger TR\_TEACHER    create trigger TR\_TEACHER  on TEACHER after INSERT, DELETE, UPDATE  as declare @a1 char(10), @a2 varchar(100), @a3 char(1), @a4 char(20), @in varchar(300);  declare @ins int = (select count(\*) from inserted),  @del int = (select count(\*) from deleted);  if @ins > 0 and @del = 0  begin  print 'Событие: INSERT';  set @a1 = (select TEACHER from INSERTED);  set @a2= (select TEACHER\_NAME from INSERTED);  set @a3= (select GENDER from INSERTED);  set @a4 = (select PULPIT from INSERTED);  set @in = @a1+' '+ @a2 +' '+ @a3+ ' ' +@a4;  insert into TR\_AUDIT(STMT, TRNAME, CC)  values('INS', 'TR\_TEACHER\_INS', @in);  end;  else  if @ins = 0 and @del > 0  begin  print 'Событие: DELETE';  set @a1 = (select TEACHER from DELETED);  set @a2= (select TEACHER\_NAME from DELETED);  set @a3= (select GENDER from DELETED);  set @a4 = (select PULPIT from DELETED);  set @in = @a1+' '+ @a2 +' '+ @a3+ ' ' +@a4;  insert into TR\_AUDIT(STMT, TRNAME, CC)  values('DEL', 'TR\_TEACHER\_DEL', @in);  end;  else  if @ins > 0 and @del > 0  begin  print 'Событие: UPDATE';  set @a1 = (select TEACHER from INSERTED);  set @a2= (select TEACHER\_NAME from INSERTED);  set @a3= (select GENDER from INSERTED);  set @a4 = (select PULPIT from INSERTED);  set @in = @a1+' '+ @a2 +' '+ @a3+ ' ' +@a4;  set @a1 = (select TEACHER from deleted);  set @a2= (select TEACHER\_NAME from DELETED);  set @a3= (select GENDER from DELETED);  set @a4 = (select PULPIT from DELETED);  set @in =@in + '' + @a1+' '+ @a2 +' '+ @a3+ ' ' +@a4;  insert into TR\_AUDIT(STMT, TRNAME, CC)  values('UPD', 'TR\_TEACHER\_UPD', @in);  end;  return;  go  insert into TEACHER values('ИВНВ', 'Иванов', 'м', 'ИСиТ');  delete TEACHER where TEACHER='ИВНВ'  update TEACHER set GENDER = 'ж' where TEACHER='БРКВЧ'  select \* from TR\_AUDIT  ------5  --Разработать сценарий, который демонстрирует на примере базы данных X\_BSTU, что провер-ка  -- ограничения целостности выполняется до срабатывания AFTER-триггера.  update TEACHER set GENDER = 'й' where TEACHER='БЗБРДВ'  select \* from TR\_AUDIT  ----6 создать 3 афтер тригера и изменить их порядок  go  create trigger AUD\_AFTER\_DEL1 on FACULTY after DELETE  as print 'AUD\_AFTER\_DEL1';  return;  go  create trigger AUD\_AFTER\_DEL2 on FACULTY after DELETE  as print 'AUD\_AFTER\_DEL2';  return;  go  create trigger AUD\_AFTER\_DEL3 on FACULTY after DELETE  as print 'AUD\_AFTER\_DEL3';  return;  go  select t.name, e.type\_desc  from sys.triggers t join sys.trigger\_events e on t.object\_id = e.object\_id  where OBJECT\_NAME(t.parent\_id)='FACULTY' and e.type\_desc = 'DELETE' ;  exec SP\_SETTRIGGERORDER @triggername = 'AUD\_AFTER\_DEL3',  @order='First', @stmttype = 'DELETE';  exec SP\_SETTRIGGERORDER @triggername = 'AUD\_AFTER\_DEL2',  @order='Last', @stmttype = 'DELETE';  select t.name, e.type\_desc  from sys.triggers t join sys.trigger\_events e on t.object\_id = e.object\_id  where OBJECT\_NAME(t.parent\_id)='FACULTY' and e.type\_desc = 'DELETE';    --drop trigger AUD\_AFTER\_DEL1  --drop trigger AUD\_AFTER\_DEL2  --drop trigger AUD\_AFTER\_DEL2  -----7  --Разработать сценарий, демонстрирующий на примере базы данных X\_BSTU утверждение:  --AFTER-триггер является частью транзакции, в рамках которого выполняется оператор, активизировавший триггер.(выполнится в любом случае)  go  create trigger PTran  on PULPIT after INSERT, DELETE, UPDATE  as declare @c int = (select count (\*) from PULPIT);  if (@c >26)  begin  raiserror('Общая количество кафедр не может быть >26', 10, 1);  rollback;  end;  return;  insert into PULPIT(PULPIT) values ('ТТПЛ')  delete PULPIT where PULPIT='ТТПЛ'  --drop trigger PTran  -----8  --Создать для таблицы FACULTY INSTEAD OF-триггер, запрещающий удаление строк в таблице.  --Разработать сценарий, который демонстри-рует на примере базы данных X\_BSTU,  --что проверка ограничения целостности выполнена, если есть INSTEADOF-триггер.  --С помощью оператора DROP удалить все DML-триггеры, созданные в этой лабораторной работе.  go  create trigger F\_INSTEAD\_OF  on FACULTY instead of DELETE  as  raiserror(N'Удаление запрещено', 10, 1);  return;  delete FACULTY where FACULTY = 'ИДиП'  drop trigger F\_INSTEAD\_OF  drop trigger PTran  drop trigger TR\_TEACHER  drop trigger TR\_TEACHER\_DEL  go  ----9 Создать DDL-триггер, реагирующий на все DDL-события в БД UNIVER. Триггер должен запрещать создавать новые таблицы и удалять существующие.  ----Свое выполнение триггер должен сопровождать сообщением, которое содержит: тип события, имя и тип объекта, а также пояснительный текст,  ----в случае запрещения выполнения оператора.  ----Разработать сценарий, демонстрирующий работу триггера.  create trigger DDL\_UNIVER on database  for DDL\_DATABASE\_LEVEL\_EVENTS as  declare @t varchar(50) = EVENTDATA().value('(/EVENT\_INSTANCE/EventType)[1]', 'varchar(50)');  declare @t1 varchar(50) = EVENTDATA().value('(/EVENT\_INSTANCE/ObjectName)[1]', 'varchar(50)');  declare @t2 varchar(50) = EVENTDATA().value('(/EVENT\_INSTANCE/ObjectType)[1]', 'varchar(50)');  begin  print 'Тип события: '+@t;  print 'Имя объекта: '+@t1;  print 'Тип объекта: '+@t2;  raiserror( N'операции с таблицами запрещены', 16, 1);  rollback;  end;  return;  go |
|  |

*Лабораторная работа №16*

Использование XML

**XML** (Extensible Markup Language) – расширяемый язык разметки. XML-формат часто используется для обмена данными между компонентами информационных систем. При работе с базами данных важными являются две задачи: преобразование *табличных данных в XML-структуры* и преобразование *XML-структур в строки реляционной таблицы*.

|  |
| --- |
| USE D\_UNIVER;  -------1(созд.XML в режиме PATH из TEACHER для преподов ИСиТ)  -------При использовании режима PATH каждый столбец конфигурируется независимо с помощью псевдонима этого столбца.  go  select PULPIT.FACULTY[Факультет], TEACHER.PULPIT[Название\_кафедры],  TEACHER.TEACHER\_NAME[Имя\_преподавателя]  from TEACHER inner join PULPIT  on TEACHER.PULPIT = PULPIT.PULPIT  where TEACHER.PULPIT = 'ИСиТ' for xml path, root('Список\_преподавателей\_с\_кафедры\_ИСИТ');  -------2 режим AUTO: назв.ауд, тип, вмест + только лекционные ауд.  -------применением вложенных элементов  go  select AUDITORIUM.AUDITORIUM\_NAME[Наименование\_аудитории],AUDITORIUM.AUDITORIUM\_TYPE[Тип\_аудитории],  AUDITORIUM.AUDITORIUM\_CAPACITY[Вместимость\_аудитории] from AUDITORIUM join AUDITORIUM\_TYPE  on AUDITORIUM.AUDITORIUM\_TYPE= AUDITORIUM\_TYPE.AUDITORIUM\_TYPE  where AUDITORIUM.AUDITORIUM\_TYPE='ЛК' for XML auto, root('Лекционные\_аудитории'),elements;    ------3 три новые дисциплины  ------OPENXML, констр. INSERT... SELECT  ------имена его атрибутов совпадают с именами столбцов результирующего набора  go  declare @h int = 0,  @sbj varchar(3000) = '<?xml version="1.0" encoding="windows-1251" ?>  <дисциплины>  <дисциплина код="БЛкК" название="Бессмертная любовь к курочке-гриль" кафедра="ИСиТ" />  <дисциплина код="ПВ" название="Предельное веселье" кафедра="ИСиТ" />  <дисциплина код="МПП" название="Международное поедание пельмешек" кафедра="ИСиТ" />  </дисциплины>';  exec sp\_xml\_preparedocument @h output, @sbj; ----ПОДГОТОВКА ДОКУМЕНТА  insert SUBJECT select[код], [название], [кафедра] from openxml(@h, '/дисциплины/дисциплина',0)  with([код] char(10), [название] varchar(100), [кафедра] char(20));  delete from SUBJECT where SUBJECT.SUBJECT='БЛкК' or SUBJECT.SUBJECT='ПВ' or SUBJECT.SUBJECT='МПП';  ------4  ------Используя таблицу STUDENT разработать XML-структуру, содержащую паспортные данные студента:  ------серию и номер паспорта, личный номер,  ------дата выдачи и адрес прописки. Разработать сценарий, в который включен оператор INSERT, добавляющий строку с XML-столбцом.  insert into STUDENT(IDGROUP, NAME, BDAY, INFO)  values(22, 'Дрожжа Ю.А.', '01.07.2001',  '<студент>  <паспорт серия="АВ" номер="1234567" дата="02.07.2016" />  <телефон>+375292537755</телефон>  <адрес>  <страна>Беларусь</страна>  <город>Минск</город>  <улица>Серова</улица>  <дом>78</дом>  <квартира>106</квартира>  </адрес>  </студент>');  select \* from STUDENT where NAME = 'Дрожжа Ю.А.';  update STUDENT set INFO = '<студент>  <паспорт серия="АВ" номер="1234567" дата="02.07.2016" />  <телефон>+375292537755</телефон>  <адрес>  <страна>Беларусь</страна>  <город>Минск</город>  <улица>Серова</улица>  <дом>78</дом>  <квартира>106</квартира>  </адрес>  </студент>' where NAME='Дрожжа Ю.А.'  select NAME[ФИО], INFO.value('(студент/паспорт/@серия)[1]', 'char(2)')[Серия паспорта],  INFO.value('(студент/паспорт/@номер)[1]', 'varchar(20)')[Номер паспорта],  INFO.query('/студент/адрес')[Адрес]  from STUDENT  where NAME = 'Дрожжа Ю.А.';  delete from STUDENT where NAME='Дрожжа Ю.А.';  ------5  ------Изменить (ALTER TABLE) таблицу STUDENT в базе данных UNIVER таким образом, чтобы значения типизированного столбца с  ------именем INFO контролировались коллекцией XML-схем (XML SCHEMACOLLECTION), представленной в правой части.  go  create xml schema collection Student as  N'<?xml version="1.0" encoding="utf-16" ?>  <xs:schema attributeFormDefault="unqualified"  elementFormDefault="qualified"  xmlns:xs="http://www.w3.org/2001/XMLSchema">  <xs:element name="студент">  <xs:complexType><xs:sequence>  <xs:element name="паспорт" maxOccurs="1" minOccurs="1">  <xs:complexType>  <xs:attribute name="серия" type="xs:string" use="required" />  <xs:attribute name="номер" type="xs:unsignedInt" use="required"/>  <xs:attribute name="дата" use="required">  <xs:simpleType> <xs:restriction base ="xs:string">  <xs:pattern value="[0-9]{2}.[0-9]{2}.[0-9]{4}"/>  </xs:restriction> </xs:simpleType>  </xs:attribute>  </xs:complexType>  </xs:element>  <xs:element maxOccurs="3" name="телефон" type="xs:unsignedInt"/>  <xs:element name="адрес"> <xs:complexType><xs:sequence>  <xs:element name="страна" type="xs:string" />  <xs:element name="город" type="xs:string" />  <xs:element name="улица" type="xs:string" />  <xs:element name="дом" type="xs:string" />  <xs:element name="квартира" type="xs:string" />  </xs:sequence></xs:complexType> </xs:element>  </xs:sequence></xs:complexType>  </xs:element></xs:schema>';  --alter table STUDENT alter column INFO xml(Student);  --drop XML SCHEMA COLLECTION Student;  select Name, INFO from STUDENT where NAME='Дрожжа Ю.А.'  -----6 задание\*\*\*\*\*  go  select PULPIT.FACULTY[факультет/@Название\_факультета], TEACHER.PULPIT[факультет/кафедра/@Название\_кафедры],  TEACHER.TEACHER\_NAME[факультет/кафедра/преподаватель/@Имя\_преподавателя]  from TEACHER inner join PULPIT  on TEACHER.PULPIT = PULPIT.PULPIT  for xml path, root('Список\_преподавателей'); |
|  |