Лабораторная работа № 21

«Создание таблиц базы данных в среде MS SQL Server 2005»

Цель: «Сформировать навыки проектирования таблиц в среде MS SQL Server 2005, построить схему данных»

Ход Работы

1. Общий вид оператора CREATE TABLE

Перед созданием таблиц БД необходимо продумать определение всех столбцов таблицы и характеристик каждого столбца (таких, как тип, длина, обязательность для ввода, ограничения, накладываемые на значения и пр.), индексов, ограничений целостности по отношению к другим таблицам.
БД в которую будет добавлена создаваемая таблица, должна быть открыта, т.е. с ней должно быть установлено активное соединение. Создание таблицы БД осуществляется оператором

CREATE TABLE [databasename].Имя Таблицы (<опр_столбца> [, <опр_столбца>| <ограничение>)
<опр_столбца> = _ опр_столбца{тип_данных }
[DEFAULT { литерал NULL |USER}]
[NOT NULL] [<огранич_столбца>]
[COLLATE collation]

DEFAULT определяет значение, которое по умолчанию заносится в столбец при создании записи таблицы; это значение будет присутствовать в соответствующем столбце данной записи до тех пор, пока пользователь не изменит его каким-либо образом; значения по умолчанию.
<огранич_столбца>- ограничения, накладываемые на значения столбца.

COLLATE collation - определяет порядок сортировки символов

При проектировании базы данных создаётся концептуальная модель, которая преобразуется в реляционную модель. Объектами реляционной модели являются таблицы. Рассмотрим создание базы данных учебного процесса.
Таблицы создаются в среде SQL Server Management Studio. Запустите виртуальную машину. Для запуска программы выберите **Пуск | Все программы | Microsoft SQL Server 2005 | SQL Server Management Studio**. Появится окно **Connect to Server**. (Подключение к серверу). Нажмите кнопку Connect. После этого появится окно:

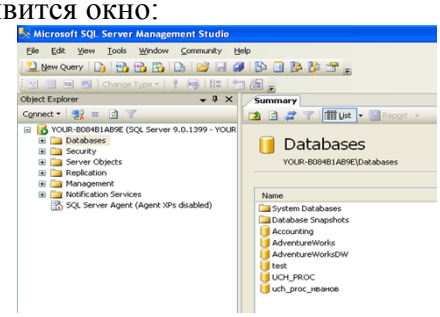


Рис 1. Окно SQL Server Management Studio

Слева находится окно обозревателя объектов Obejct Explorer. В нём отображается дерево объектов **SQL** сервера. Чтобы создать новую базу данных, выделите мышью Databases и из контекстного меню выберите New Database, появится окно

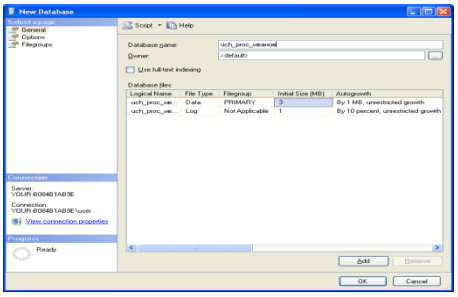


Рис 2. Окно для создания новой базы.

В строке Database name укажите имя вашей базы, нажмите на OK. В обозревателе объектов появится новая база данных.

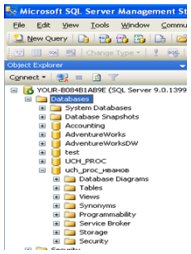


Рис. 3. Отображение новой базы данных

Для того, чтобы ввести код по созданию таблиц нужно на панели инструментов нажать кнопку New Query. В редакторе запросов введите операторы по созданию таблиц. После каждой таблице нажмите кнопку 

. Если ошибок нет, то среди списка таблиц Tables появится вновь созданная таблица. Если имеются ошибки, то они будут перечислены в нижней части окна.
Схема данных приведена в приложении А. Первичный ключ задаётся: *primary key (ng).*
Внешний ключ для связи двух таблиц по ключевому полю ng :
constraint FK_NG foreign key (ng) references gruppa (ng)

Задание 1. Создайте таблицы с использованием операторов Transact SQL
Структуры таблиц приведённые ниже

Use uch_proz Create table **gruppa** (ng varchar(3) not null, kol int , pball float , *primary key (ng)*)	Use uch_proz Create table **student** (ng varchar(3) not null, ns varchar(3) not null, fio varchar(20) not null, pball float , *primary key (ng, ns)* *constraint FK_NG foreign key (ng)* *references gruppa (ng))*
Use uch_proz Create table **kafedra** (kkaf varchar(3) not null, namekaf varchar(20) not null, *primary key (*kkaf*)*)	Use uch_proz Create table **predmet** (kp varchar(3) not null, np varchar(30) not null, chas int, lek int, pr int, ch int, *primary key (*kp*)*)
Use uch_proz Create table **prepodavatel** (tabn varchar(3) not null, fio varchar(30) not null, kkaf varchar(3) not null,	Use uch_proz Create table **isuchenie** (ng varchar(3) not null kp varchar(3) not null tabn varchar(3) not null,

*primary key (*tabn*)* , *constraint FK_prepk foreign key (kkaf)* *references kafedra (kkaf)*)	vidz varchar(3) not null, chas int, *primary key (ng,* tabn, kp, vidz))
Use uch_proz Create table **uspevaemost** (ng varchar(3) not null, ns varchar(3) not null, kp varchar(3) not null tabn varchar(3) not null, vidz varchar(3) not null, ozenka int, *primary key (ng,ns,* tabn, kp, vidz))	

Задание 2.Построение диаграммы базы данных.

Для установления отношений между таблицами необходимо построить диаграмму.

a) Щёлкните правой кнопкой мыши по Database Diagramm.
b) В появившемся контекстном меню выберите New Database Diagramm. Появится перечень таблиц, которые можно включить в схему. Выберите таблицы и нажмите на Close.
c) Установите отношения между таблицами, переместив ключевые поля с таблиц мощности 1 на поля «многие»

Контрольные задания.

1. Заполните данными построенные таблицы базы «Учебный процесс»
2. Постройте концептуальную модель по выбранной вами предметной области.
3. Преобразуйте концептуальную модель в реляционную.

Лабораторная работа №22

«Создание таблиц визуальными средствами SQL Server»

Цель: «сформировать навыки построения таблиц и диаграммы данных визуальными средствами Sql Serever»

Ход Работы

1. Создание таблицы.

Для создания таблицы в Sql Server Management Studio необходимо выполнить следующие действия:
В окне обозревателя Object Explorer откройте объект Databases, а затем узел базы данных.
Щёлкните правой кнопкой мыши по объекту Tables, в контекстном меню выберите команду New Tables.
Откроется окно конструктора таблиц

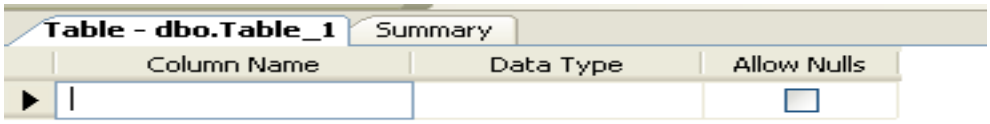


Рис 4 Конструктор таблиц

Окно предназначено для ввода сведений о полях таблицы. В конструкторе три колонки: имя поля Column Name, тип данных Data Type и разрешение не вводить значение в поле при добавлении новой записи в таблицу Allow Nulls.

Задание 1.

Спроектируйте концептуальную модель вашей базы данных. Преобразуйте её в реляционную модель. Создайте в среде **SQL Server Management Studio** вашу базу данных.

Задание 2.

Создайте структуры ваших таблиц в новой базе данных.

- Щёлкните правой кнопкой мыши по объекту Tables, в контекстном меню выберите команду New Tables.
- В колонке Column Name введите имена полей вашей таблицы.
- В колонке Data Type укажите тип каждого поля таблицы.
- Ключевые поля выделите мышью и на панели инструментов нажмите изображение ‘ключ’
- Выберите существующее поле таблицы и в окне Column Properties появятся его свойства.
- Для сохранения таблицы нажмите на дискету. Введите имя таблицы. Закройте структуру созданной таблицы.

Добавить поле в таблицу можно только в окне Table, а корректировать его название и тип данных можно как в этом окне, так и в окне Column Properties. При выборе в качестве типа данных Decimal в окне Column Properties появляются дополнительные строки: точность Precesion и степень Scale, которые содержат максимальное количество десятичных знаков и максимальное количество знаков после десятичной точки.

2. Просмотр информации о таблице

Sql Server Management Studio даёт возможность пользователю получить информацию о таблице. Для этого

1) В окне обозревателя выберите нужную базу данных.
2) Откройте список её объектов, щёлкнув по значку плюс, расположенному слева от названия базы.
3) Откройте узел таблиц Tables
4) Щёлкните правой кнопкой мыши по названию таблицы, информацию о которой нужно получить.
5) В появившемся контекстном меню выберите свойства Properties.
6) Откроется окно «Таблица свойств»

В появившемся окне три вкладки: общие ‘general’, разрешения ‘Permissions’ и расширенные ‘extended Properties’. На этих вкладках содержится:

- Размер таблицы на жёстком диске (Data Spase)
- Размер области, которую занимают индексы (Index Spase)
- Количество строк в таблице (Row Count)
- Дата создания таблицы (Data Created)
- Дата последней модификации (Last Update)

Задание 3

Выберите любую таблицу вашей базы данных и просмотрите её свойства. Опишите свойства таблице в отчёте.

3. Копирование, переименование и удаление таблиц

Для создания копии таблицы лучше всего использовать Transact SQL. На панели нажмите кнопку New Query.

Задание 4.

Выполним копирование таблицы kafedra. Копию таблицы назовём kafedra_copy

```
use test
select * into kafedra_copy from kafedra
```

Для удаления или переименования таблицы необходимо выполнить действия:

- Выберите нужную таблицу базы данных из списка Tables. Из контекстного меню выберите команду View Dependencies (Просмотреть зависимости). При удалении таблицы или её переименовании указанные в этом окне зависимости между объектами будут разорваны. Внимательно изучите их.
- Для удаления таблицы щёлкните правой кнопкой мыши по её имени и выберите в контекстном меню команду Delete.
- Для переименования таблицы из контекстного меню выберите Rename.

Задание 5.

Переименуйте таблицу kafedra_copy в таблицу copy. Удалите эту таблицу.

4. Построение диаграммы базы данных.

- Выберите в окне обозревателя Object Explorer базу данных, для которой нужно построить диаграмму данных.
- В базе данных выберите паку Database Diagrams. Из контекстного меню выберите пункт New Database Diagrams
- Появится первое окно мастера. Оно предназначено для выбора таблиц, которые требуется включить в диаграмму.
- После выбора таблиц щелкните на кнопке Close. Запустится процесс построения диаграмм.
- Для установления отношений между таблицами переместите ключевые поля с таблицы мощности «один» на таблицу со стороны «многие». Типы и размер полей связи должны совпадать.
- После установления связей между таблицами диаграмму нужно сохранить.

5. Ввод данных в таблицу, просмотр данных в таблице.

Для того, чтобы ввести данные в таблицу, необходимо выбрать таблицу базы данных из списка Tables, из контекстного меню этой таблицы выбрать Open Table. Откроется окно таблицы, в нижней части которого размещены кнопки для перехода по записям.

Контрольные задания.

1. Создайте структуры таблиц вашей базы данных.
2. Постройте диаграмму данных.
3. Заполните данными построенные таблицы .
4. Просмотрите свойства нескольких таблиц. Опишите свойства таблиц в отчёте.
5. Создайте копию одной из таблиц.
6. Переименуйте скопированную таблицу, удалите её.

Лабораторная работа №23

«Структурированный Язык Запросов SQL. Команда SELECT»

Цель: «Изучить структуру оператора sql, выработать навыки построения запросов на выборку»

Ход Работы

SQL символизирует собой Структурированный Язык Запросов. Запросы - вероятно наиболее часто используемый аспект SQL. Запрос - команда которую вы даете вашей программе базы данных, и которая сообщает ей чтобы она вывела определенную информацию из таблиц в память. Эта информация обычно посылается непосредственно на экран компьютера.

1. Формат команды SELECT

Общий вид команды:

```
SELECT * | { [ DISTINCT | ALL] < value expression >.,..}
  FROM { < table name > [ < alias > ] }.,..
  [ WHERE ]
  [ GROUP BY { | }.,..]
  [ HAVING ]
  [ ORDERBY { | }.,..]
```

В самой простой форме, команда SELECT просто инструктирует базу данных чтобы извлечь информацию из таблицы. Например, вы могли бы вывести таблицу ‘Студент’ напечатав следующее:

SELECT ns,fio FROM student
Эта команда просто выводит все данные из таблицы.

SELECT Ключевое слово которое сообщает базе данных что эта команда - запрос. Все запросы начинаются этим словом

ns, fio Это - список столбцов из таблицы которые выбираются запросом. Любые столбцы не перечисленные здесь не будут включены в вывод команды. Это, конечно, не значит что они будут удалены или их информация будет стерта из таблиц, потому что запрос не воздействует на информацию в таблицах; он только показывает данные/

FROM - ключевое слово, подобно SELECT, которое должно быть представлено в каждом запросе. Оно сопровождается пробелом и затем именем таблицы используемой в качестве источника информации.
В данном случае - это таблица student

Если вы хотите видеть каждый столбец таблицы, имеется необязательное сокращение которое вы можете использовать. Звездочка (*) может применяться для вывода полного списка столбцов следующим образом:

SELECT * FROM student;
Это приведет к тому же результату что и наша предыдущая команда.

DISTINCT (ОТЛИЧИЕ) - аргумент который обеспечивает Вас способом устранять двойные значения из вашего предложения SELECT. Предположим что вы хотите знать какие студенты получали оценки.

```
SELECT DISTINCT ns
FROM uspevaemost
```

Другими словами, DISTINCT следит за тем, какие значения были ранее, так что бы они не были продублированы в списке. Это - полезный способ избежать избыточности данных. DISTINCT опускает строки где все выбранные поля идентичны. Строки в которых некоторые значения одинаковы а некоторые различны - будут сохранены.

WHERE - предложение команды SELECT, которое позволяет вам устанавливать предикаты, условие которых может быть или верным или неверным для любой строки таблицы. Команда извлекает только те строки из таблицы для которой такое утверждение верно. Например, предположим вы хотите видеть номера групп с количеством студентов более 20 человек.

```
SELECT ng
FROM gruppa
WHERE kol > 20
```

Когда предложение WHERE представлено, программа базы данных просматривает всю таблицу по одной строке и исследует каждую строку чтобы определить верно ли утверждение.

2. Реляционные операторы

Реляционный оператор - математический символ который указывает на определенный тип сравнения между двумя значениями. Реляционные операторы которыми располагает SQL :

- = Равный к
- > Больше чем
- < Меньше чем
- >= Больше чем или равно
- <= Меньше чем или равно
- <> Не равно

Основные Булевы операторы также распознаются в SQL. . Стандартными операторами Буля распознаваемыми в SQL являются:

AND, OR, и **NOT**

Задание 1.
Вывести наименования предметов, с количеством часов равным 40 или 50.

```
SELECT np
FROM predmet
WHERE chas = 40 or chas = 50
```

Задание 2.
Вывести наименования предметов, у которых количество часов не равно сумме лекционных и практических.

```
SELECT np
FROM predmet
WHERE chas<> lek + pr
```

3 Использование специальных операторов в условиях.

SQL использует специальные операторы IN, BETWEEN, LIKE, и IS NULL. Мы рассмотрим как их использовать и как реляционные операторы позволяют создавать более сложные и мощные предикаты. Обсуждение оператора IS NULL будет включать отсутствие данных и значение

NULL, которое указывает на то: что данные отсутствуют. Вы также узнаете о разновидностях использования оператора NOT применяющегося с этими операторами.

Оператор IN

Оператор IN определяет набор значений в которое данное значение может или не может быть включено.

*Задание 3.*Вывести номера студентов и номера предметов, по которым получали оценки 3 или 4

```
SELECT ns, np
FROM uspevaemost
WHERE ozenka in ( 3, 4)
```

Оператор BETWEEN

Оператор BETWEEN похож на оператор IN. В отличии от определения по номерам из набора, как это делает IN, BETWEEN определяет диапазон, значения которого должны уменьшаться что делает предикат верным. Вы должны ввести ключевое слово BETWEEN с начальным значением, ключевое AND и конечное значение. В отличие от IN, BETWEEN чувствителен к по-
рядку, и первое значение в предложении должно быть первым по алфавитному или числовому порядку.

Задание 4. Выведите из таблицы predmet все наименования предметов с количеством часов, принадлежащих интервалу 100 - 140 :

```
SELECT *
FROM predmet
WHERE chas BETWEEN 100 AND 140;
```

Контрольные задания.

1. Из таблицы Успеваемость выведите код предмета, табельный номер преподавателя, номер студента, который получал двойку.
2. Выведите на экран количество студентов в ‘101’ группе
3. Создайте запрос, который выводит на экран код кафедры ‘математика’
4. Из таблицы Преподаватель выведите на экран фамилии и табельные номера преподавателей с кафедры ‘02’
5. Из таблицы Изучение выведите на экран номер группы и табельный номер преподавателя, который ведёт предмет с количеством часов (chas) более 50 .
6. По разработанной базе данных сформулируйте и выполните запросы с использованием операторов: AND, OR, IN, BETWEEN

Лабораторная работа №24

«Управляющие конструкции языка запросов SQL »

Цель: «Сформировать навыки и умения реализации запроса с использованием управляющих конструкций»

Ход Работы

1.Операторные скобки

Как им все алгоритмические языки Transact-SQL содержит в своём составе операторные скобки. Синтаксис конструкции имеет следующий вид

```
BEGIN
  <команда 1>
  <команда 2>

END
```

Задание 1. Вывести фамилии преподавателей с кафедры ‘02’ и список студентов из группы ‘101’

```
begin
use uch_proc
select fio as фио_преподавателей from prepodavatel where kkaf = '02'

select fio as фио _студентов from student where ng = '101'
end
```

2 Операторы ветвления

Для организации ветвления используется конструкция IF…ELSE. Синтаксис конструкции:

```
IF  <условие>
   <команда 1>
ELSE
     <команда 2>
```

Если в какой-либо ветке алгоритма необходимо выполнить более чем по одной команде, то следует воспользоваться конструкцией BEGIN.. END

Задание 2. Определить содержится ли фамилия ‘Иванов’ в таблице Преподаватель.

Код запроса имеет вид:

```
use uch_proc
IF 'Глухов' in (select fio  from prepodavatel )
  print 'фамилия есть в списке преподавателей'
else
  print 'фамилия нет в списке преподавателей'
```

Если разветвлений в алгоритме много и они одиночные, то для их замены можно воспользоваться конструкцией Case …..End

```
CASE  <входное выражение>
  WHEN < выражение 1>   THEN < выражение результат>
  WHEN < выражение 2>   THEN < выражение результат>

  ELSE < выражение результат>
End
```

Конструкция фактически представляет собой функцию. У функции имеется один параметр (входное выражение). Он указывается после Case, но не в скобках. Функция возвращает результат, поэтому эта конструкция должна входить в состав какого либо выражения. Если в примере, рассмотренном далее убрать PRINT, то система выдаст сообщение об ошибке, т.к. это будет неверный вызов функции.

Работает конструкция следующим образом. Если значение входного выражения и одного из перечисленных, стоящих после WHEN, совпадают, то возвращается выражение – результат, указанное после соответствующего THEN. Если не одного совпадения не отмечено, то возвращается результат, стоящий после ELSE. Если значение входного выражения совпадает более чем с одним значением выражения, стоящего после WHEN, то выполняется результат, соответствующий первому совпадению.

Задание 3 Вывести на экран значение суммы прописью, используя конструкцию Case …..End

```
declare @rub char(25)
set @rub = ‘6 руб’
```

```
PRINT
CASE  @rub
  WHEN ‘ 1 руб’   THEN ‘ один рубль’
  WHEN ‘ 2 руб’   THEN ‘ два рубля’
  WHEN ‘ 3 руб’   THEN ‘ три рубля’
  ELSE ‘ таких денег в кассе нет’
End
```

Результат выполнения команды : таких денег в кассе нет

В следующем примере рассмотрим применение конструкции в запросе.

Задание 4 В запросе формируется список номеров студентов, оценок и добавлен столбец , в котором указано одно из значений (двоечник, троечник, хорошист, отличник)

```
declare @o int
use uch_proc
select ozenka, ns, kp ,tit=
case ozenka
  when 2 then 'двоечник'
  when 3 then 'троечник'
  when 4 then 'хорошист'
  when 5 then 'отличник'
end
from uspevaemost
```

3 Операторы цикла

В Transact-SQL имеется конструкция для организации многократных повторений команд в программе : WHELE……CONTINUE. Одна обеспечивает выполнение цикла одного типа. Это цикл с предусловием. Синтаксис конструкции:

```
WHELE  < условие>
      { команда | блок  }
[BREAK]
      { команда | блок  }

[ CONTINUE ]
```

Цикл можно завершить принудительно, для этого в нужном месте цикла нужно поместить служебное слово BREAK .

Задание 5 Вычислите и выведите на экран квадраты и кубы чисел от 1 до 5.

```
declare @r int
PRINT '      Число       квадрат    куб'

set @r = 1
WHILE @r <= 5
BEGIN
   PRINT STR(@r )+ STR(@r*@r )+ STR(@r*@r*@r )
   SET @r = @r+1
END
```

4 Оператор LIKE

LIKE применим только к полям типа CHAR или VARCHAR, с которыми он используется чтобы находить подстроки. Т.е. он ищет поле символа чтобы видеть, совпадает ли с условием часть его строки.

Задание 5 Вывести фамилии преподавателей, начинающихся с буквы ‘А’

```
USE uch_proc
SELECT * FROM prepodavatel
where fio              like 'A%'
```

5 Операторы для обработки исключений

В Transact-SQL существует возможность обработки исключений:
BEGIN TRY
----- ЗАПРОС, ВЫЗЫВАЮЩИЙ ОПАСЕНИЯ
END TRY

BEGIN CATCH
----- ОБРАБОТКА ОШИБКИ, КОТОРАЯ МОЖЕТ ВОЗНИКНУТЬ
END CATCH

Контрольные задания.

1. По таблице ‘успеваемость’ определить получал студент с номером ‘01’ оценки. Если получал, то вывести на печать сообщение ‘Студент опрашивался’ в противном случае напечатать ‘Студент не получал оценок’ .
2. Если вид занятия ‘лек’, то в сформированном столбце записать «лекционное занятие», если вид занятия ‘пр’, то в сформированном столбце записать «практическое занятие»
3. В индивидуальной базе данных сформулировать и привести пример с использованием оператора LIKE
4. В индивидуальной базе данных сформулировать и привести пример с использованием оператора Case …..End

Лабораторная работа №25

«Групповые операции. Агрегатные функции »

Цель: «Сформировать навыки и умения реализации запроса с использованием групповых операций и агрегатных функций»

Ход Работы

1. Агрегатные функции

Запросы могут производить обобщенное групповое значение полей точно также как и значение одного поля. Это делает с помощью агрегатых функций. Агрегатные функции производят одиночное значение для всей группы таблицы. Имеется список этих функций:

- COUNT производит номера строк или не-NULL значения полей которые выбрал запрос.
- SUM производит арифметическую сумму всех выбранных значений данного поля.
- AVG производит усреднение всех выбранных значений данного поля.
- MAX производит наибольшее из всех выбранных значений данного поля.
- MIN производит наименьшее из всех выбранных значений данного поля.

Только числовые поля могут использоваться с SUM и AVG. С COUNT, MAX, и MIN, могут использоваться и числовые или символьные поля.

*Задание 1.*Вывести среднюю оценку по таблице Успеваемость для студента с номером ‘01’

```
USE uch_proc
SELECT AVG(ozenka) FROM uspevaemost
where ns='01'
```

Функция COUNT несколько отличается от всех. Она считает число значений в данном столбце, или число строк в таблице. Когда она считает значения столбца, она используется с DISTINCT чтобы производить счет чисел различных значений в данном поле.

*Задание 2.*Определить в таблице Студент количество студентов в таблице

```
USE uch_proc
SELECT count(ns) FROM student
```

Чтобы подсчитать общее число строк в таблице, используйте функцию COUNT со звездочкой вместо имени поля, как в следующем примере:

```
USE uch_proc
SELECT count(*) FROM student
```

2. Групповые операции

Предложение GROUP BY позволяет вам определять подмножество значений в особом поле в терминах другого поля, и применять функцию агрегата к подмножеству. Это дает вам возможность объединять поля и агрегатные функции в едином предложении SELECT.

*Задание 3.*Определить для каждого студента по таблице Успеваемость минимальную оценку.

```
USE uch_proc
SELECT ns, min(ozenka) as 'наихудшая'
FROM uspevaemost
group by ns
```

GROUP BY применяет агрегатные функции независимо от серий групп которые определяются с помощью значения поля в целом. В этом случае, каждая группа состоит из всех строк с тем же самым значением поля ns, и MIN функция применяется отдельно для каждой такой группы. Это значение поля, к которому применяется GROUP BY, имеет, по определению, только одно значение на группу вывода, также как это делает агрегатная функция. Результатом является совместимость которая позволяет агрегатам и полям объединяться таким образом.

Вы можете также использовать GROUP BY с многочисленными полями.

Задание 4. Совершенствуя вышеупомянутый пример далее, предположим что вы хотите увидеть наименьшую оценку полученную каждым студентом по каждому предмету. Чтобы сделать это, вы должны сгруппировать таблицу uspevaemost по студентам и кодам предметов, и применить функцию MIN к каждой такой группе, подобно этому:

```
USE uch_proc
SELECT ns, min(ozenka) as 'наихудшая' , kp
FROM uspevaemost
group by ns, kp
```

3.Предложение HAVING

Предположим, что в предыдущем примере, вы хотели бы увидеть MIN оценку только студента ‘01’ Вы не сможете использовать агрегатную функцию в предложении WHERE

Вы не сможете сделать что-нибудь подобно следующему:

```
USE uch_proc
SELECT ns, min(ozenka) as 'наихудшая' , kp
FROM uspevaemost
group by ns, kp
WHERE ns = '01'
```

Предложение HAVING определяет критерии используемые чтобы удалять определенные группы из вывода, точно также как предложение WHERE делает это для индивидуальных строк.

Задание 5. Вывести наименьшую оценку полученную студентом '01' по каждому предмету.

```
USE uch_proc
SELECT ns, min(ozenka) as 'наихудшая' , kp
FROM uspevaemost
group by ns, kp
HAVING ns = '01'
```

4. Упорядочение вывода полей. Команда ORDER BY

Таблицы - это неупорядоченные наборы данных, и данные которые выходят из их, не обязательно появляются в какой-то определенной последовательности. SQL использует команду ORDER BY чтобы позволять вам упорядочивать ваш вывод. Эта команда упорядочивает вывод запроса согласно значениям в том или ином количестве выбранных столбцов. Многочисленные столбцы упорядочиваются один внутри другого, также как с GROUP BY, и вы можете определять возрастание (ASC) или убывание (DESC) для каждого столбца. По умолчанию установлено - возрастание.
Обратите внимание что, во всех случаях, столбцы которые упорядочиваются должны быть указаны в выборе SELECT.

Задание 6. Вывести по алфавиту список фамилий преподавателей

```
USE uch_proc
SELECT fio
FROM prepodavatel
order by fio
```

ORDER BY может кроме того, использоваться с GROUP BY для упорядочения групп. Если это так, то ORDER BY всегда приходит последним. Вот пример из Задания 4 с добавлением предложения ORDER BY. Перед группированием вывода, порядок студентов был произвольным.

Задание 7. Вывести по порядку номера студентов в задании 4

```
USE uch_proc
SELECT ns, min(ozenka) as 'наихудшая' , kp
FROM uspevaemost
group by ns, kp
order by ns
```

Контрольные задания.

1. По таблице Успеваемость определить среднюю оценку в каждой группе.
2. По таблице Успеваемость определить количество оценок в каждой группе.
3. Из таблицы Студент вывести список студентов по алфавиту.
4. По таблице Изучение определить суммарное количество часов по предметам в каждой группе.
5. По таблице Изучение определить суммарное количество часов по предметам с vidz ='пр' в каждой группе

6. Определить количество групп, изучающих предмет с кодом ‘01’
7. Определите количество преподавателей на каждой кафедре.
8. Сформулируйте и выполните запросы с использованием агрегатных функций в индивидуальной базе данных.

Лабораторная работа №26

«Создание многотабличного запроса»

Цель: «Сформировать навыки и умения реализации запроса, построенного на основе данных из нескольких таблиц»

Ход Работы

Одна из наиболее важных особенностей запросов SQL - это их способность определять связи между многочисленными таблицами и выводить информацию из них в терминах этих связей.
Эта особенность часто используется просто для эксплуатации связей встроенных в базу данных.
Полное имя столбца таблицы фактически состоит из имени таблицы, сопровождаемого точкой и затем именем столбца. Имеются несколько примеров имен :
Student. Ns
Student.fio
Predmet.np
До этого, вы могли опускать имена таблиц, потому что вы запрашивали только одну таблицу одновременно.

Когда данные выбираются из таблиц, связанных отношением один ко многим, то кроме условия запроса после ключевого слова WHERE записываются связи по ключевым полям. Например, если данные выбираются из таблиц student, gruppa , то необходимо указать равенство по связующим полям:
student.ng= gruppa.ng

Задание 1. Вывести на экран фамилии студентов, номер группы, в которой учится более 30 человек.

```
USE uch_proc
select student.fio, gruppa.ng, gruppa.kol
from student, gruppa
where student.ng= gruppa.ng and gruppa.kol>30
```

Если данные выбираются из таблиц, напрямую не связанных между собой, то указываются все таблицы, находящиеся между ними.

Задание 2. Вывести на экран фамилии преподавателей математики.
Таблицы ‘преподаватель’ и ‘предмет’ не связаны между собой. Но они имеют общие связующие поля с таблицей ‘изучение’

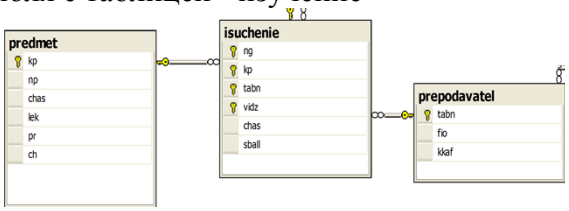


Рис 5 схема данных запроса

```
USE uch_proc
select prepodavatel.fio, predmet.np
from prepodavatel, predmet, isuchenie
where prepodavatel.tabn = isuchenie.tabn and
```

isuchenie.kp= predmet.kp and predmet.np='математика'

Задание 3 Вывести на экран фамилии преподавателей , количество часов, код предмета, номера групп, в которых ведут преподаватели с кафедры ‘информатика’

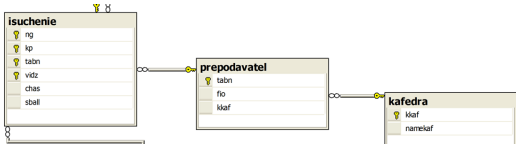


Рис 6 схема данных запроса

USE uch_proc
select prepodavatel.fio, isuchenie.ng, isuchenie.chas, isuchenie.kp
from prepodavatel, isuchenie, kafedra
where prepodavatel.tabn = isuchenie.tabn and
prepodavatel.kkaf =kafedra.kkaf and kafedra.namekaf='информатика'

Задание 4 Определите номер и фамилию студентов, которые получали оценку 4

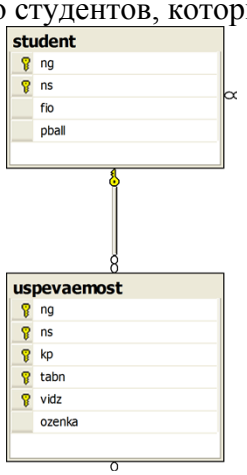


Рис 7 схема данных запроса

USE uch_proc
select student.fio, student.ns , uspevaemost.ozenka
from student, uspevaemost
where student.ng= uspevaemost.ng and student.ns= uspevaemost.ns and uspevaemost.ozenka=4
Задание 5 Определите наименования предметов, по которым студенты получали 2

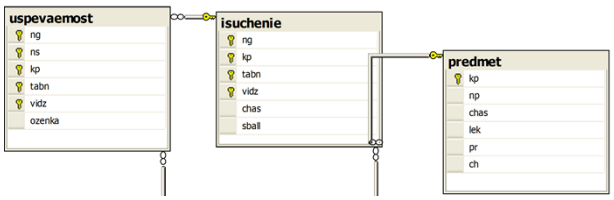


Рис 8 схема данных запроса

USE uch_proc
select uspevaemost.ozenka , predmet.np
from uspevaemost, isuchenie, predmet
where uspevaemost.ng = isuchenie.ng and uspevaemost.kp = isuchenie.kp and
uspevaemost.tabn = isuchenie.tabn and uspevaemost.vidz = isuchenie.vidz and
isuchenie.kp= predmet. kp and uspevaemost.ozenka=2

Контрольные задания.

1. Выведите на экран фамилии студентов, которые получали оценки 3 или 4.

2. Определите количество студентов, получивших каждую оценку. (2, 3, 4, 5)
3. Выведите на экран оценки, вид занятия, наименование предмета, по которому студент ‘Аристов’ получал оценки
4. Выведите номер группы, количество человек в группе, в которой обучается студентка ‘Борисова’
5. Укажите фамилии преподавателей и номера групп, в которых изучаются предметы с количеством часов больше 100
6. Определите наименования предметов, которые читаются преподавателями с кафедры ‘информатика’
7. Выведите на экран фамилии преподавателей, которые ведут предметы в группе ‘102’
8. Сформулируйте и создайте несколько многотабличных запросов в индивидуальной базе данных

Лабораторная работа №27

«Использование подзапросов»

Цель: «Сформировать навыки и умения реализации вложенных подзапросов, построенных на основе данных из нескольких таблиц»

Ход Работы

1. Построение подзапросов.

Часто невозможно решить поставленную задачу путем использования одного запроса. Это особенно актуально в тех случаях, когда при использовании *условия поиска* в предложении WHERE

<сравниваемое значение> <оператор> <значение, с которым сравнивать>

значение, с *которым надо сравнивать,* заранее не определено и должно быть вычислено в момент выполнения оператора SELECT.

Другой причиной, которая должна побудить к использованию вложенных подзапросов, является то, что во многих случаях *значение,* с *которым надо сравнивать,* должно представлять собой не одно, а несколько значений.

Внутренний подзапрос представляет собой также оператор SELECT и кодирование его предложений подчиняется тем же правилам, что и для основного оператора SELECT.
В общем случае оператор SELECT с подзапросом имеет вид

SELECT ...
FROM ...
WHERE <сравниваемое значение> <оператор> SELECT ...
FROM ... WHERE ...
Задание 1. С помощью подзапроса вывести номера студентов, обучающихся в группе с количеством человек >32

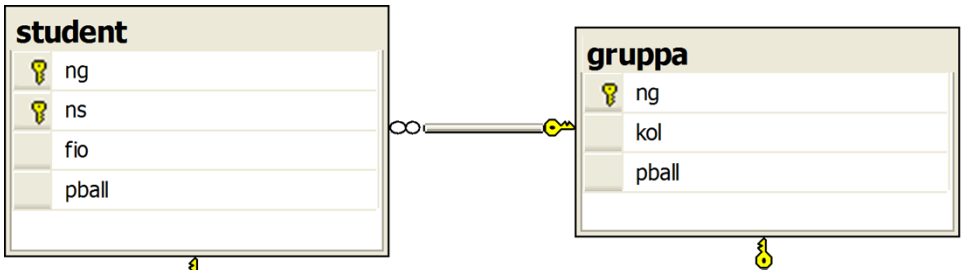


Рис 9 схема данных запроса

```
use uch_proc
select fio
from student
where ng in ( select ng
```

Внешний запрос

Внутренний запрос

```
        from gruppa
        where kol > 32 )
```

В начале выполняется подзапрос, в котором выбираются все группы с количеством человек более 32. Внешний запрос выбирает запись из таблицы ‘студент’ и определяет входит ли номер группы в список номеров групп, выбранных подзапросом.

Задание 2. С помощью подзапроса вывести ФИО преподавателей, которые ведут предметы, с количеством часов chas > 150

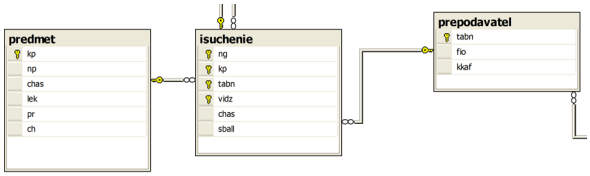


Рис 10 схема данных запроса

```
use uch_proc
select fio
from prepodavatel
where tabn in ( select tabn
        from isuchenie
        where kp in (  select kp
                from predmet
                where chas > 150 and ng ='102'
                  ) )
```

В начале выполняется подзапрос, в котором выбираются коды предметов с количеством часов более 150 и для группы ‘101’ . Второй подзапрос выбирает табельные номера преподавателей, которые ведут предметы, указанные в 1-м подзапросе. Внешний запрос выбирает запись из таблицы ‘преподаватель’ и определяет входит ли его табельный номер в список значений второго подзапроса.

Задание 3.

С помощью подзапроса вывести табельный номер преподавателей, которые ведут предмет ‘история’ в группе ‘101’.

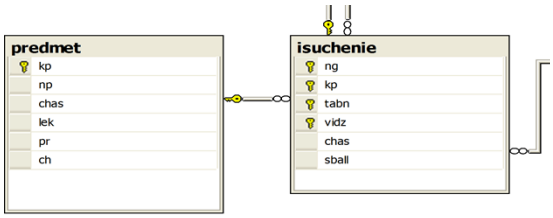


Рис 11 схема данных запроса

```
use uch_proc
 select tabn
        from isuchenie
        where ng = '101' and kp in (  select kp
                from predmet
                where np ='история'
                  )
```

Подзапрос выбирает код предмета с названием 'история'. Внешний запрос выводит табельные номера преподавателей, которые ведут предмет с таким номером в группе ‘101’

2. Использование оператора EXISTS

Бывают случаи, когда в условии поиска нужно указать, что из таблицы требуется отобрать только те записи, для которых *подзапрос* возвращает один или более значений. В этом случае в условии поиска указывается предложение

EXISTS (<подзапрос>)

Это предложение содержит истину, если запрос возвращает хотя бы одну запись.

Задание 4.
Вывести ФИО студентов, которые хотя бы раз получали ‘4’.

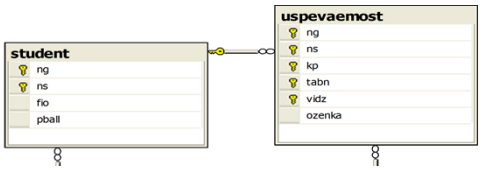


Рис 12 схема данных запроса

```
use uch_proc
 select fio
        from student
        where ns in (  select ns
                from uspevaemost
                where ozenka = 4 and uspevaemost.ns= student.ns and uspevaemost.ng= student.ng
                  )
```

Если студент хотя бы раз получал 4, то его номер будет выбран в подзапросе.

Контрольные задания.

1. С помощью подзапроса вывести фамилии преподавателей с кафедры ‘иностранный язык’.
2. С помощью подзапроса вывести фамилии преподавателей, которые ведут предметы с vidz=’пр’.
3. С помощью подзапроса вывести наименование предметов, у которых в таблице ‘изучение’ количество часов более 70.
4. С помощью подзапроса вывести номер группы и количество человек в ней, если в этой группе ведет преподаватель с табельным номером ‘103’
5. Вывести ФИО преподавателей, которые ведут хотя бы один предмет.
6. Вывести ФИО студентов, которые хотя бы раз получали ‘2’. или ‘3’.
7. Определите с помощью подзапроса номера студентов, группу и их оценки, кто изучает предмет с кодом ‘02’ и видом занятия ‘пр’.
8. Сформулируйте и создайте несколько подзапросов в индивидуальной базе данных

Лабораторная работа №28

«Реализация запросов для нахождения минимального и максимального значений»
Цель: «Сформировать навыки и умения реализации вложенных подзапросов, построенных для нахождения минимального и максимального значений»

Ход Работы

Для того, чтобы рассмотреть запросы на нахождение минимального и максимального значений, создадим базу данных ‘склад’. Она состоит из таблиц:

Tovar (kod_tov, zena), post (kod_post, name), postavka (n_post, data, kol, kod_tov, n_sklad)
Rashod_tov (n_rash, data, kol, kod_tov, n_sklad)

Задание1. Создайте структуру приведённых таблиц. Заполните таблицы данными.

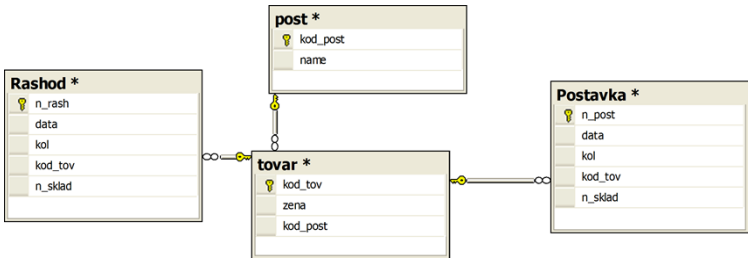


Рис 13 схема данных запроса

Tovar

kod_tov	*kod_post*	*zena*
001	12	120
002	11	300
003	14	340
004	11	100

post

kod_post	*name*
11	«Весёлый молочник»
12	«Сластёна»
13	«Колокольчик»
14	«Южная ночь»

Postavka

n_post	data	kol	kod_tov	n_sklad
01	11 сентября	100	003	3
02	11 сентября	200	001	2
03	13 сентября	50	003	1
04	16 сентября	130	002	1
05	16 сентября	230	004	2
06	18 сентября	70	003	2
07	20 сентября	200	001	2

Rashod

n_rash	data	kol	kod_tov	n_sklad
111	1 октября	30	003	1
112	1 октября	10	004	2
113	2 октября	50	001	2
114	3 октября	20	004	2
115	4 октября	15	002	1

Задание2. Выведите код товара с минимальной ценой

```
use sklad
Select  zena , kod_tov
From tovar
where zena in (
  Select min ( zena)
  From tovar
)
```

При выполнении подзапроса определяется величина минимальной цены (в примере это 100). Во внешнем запросе цена каждого товара сравнивается с найденной минимальной ценой, если совпадает, то запись выводится на экран. Вместо равенства используется оператор ***in*** .

Задание3. По таблице 'Postavka' определите дату и склад с максимальной величиной поставки.

```
use sklad
Select  data , n_sklad,kol
From Postavka
where kol in (
  Select max ( kol)
  From Postavka
)
```

При выполнении подзапроса определяется величина максимального количества (в примере это 230). Во внешнем запросе количество каждой поставки сравнивается с найденным максимальным количеством, если совпадает, то запись выводится на экран.

В приведённых примерах значение максимума и минимума определяется по числовым полям таблицы. Но иногда возникает необходимость вычислить значения по вычисляемым полям.

Использование ALL, SOME

Если в условиях поиска необходимо указать, что *сравниваемое значение* (значение столбца, результат вычисления выражения) должно находиться в определенных отношениях со всеми значениями из множества значений, возвращаемых подзапросом, применяют предложение типа

<сравниваемое значение> {[NOT] <оператор>
{ALL | SOME | ANY} (<подзапрос>)

где *подзапрос* может возвращать более одного значения. Оператор определяет операцию сравнения (>, >=, < и т.д.). Отношение *сравниваемого значения* и значений, возвращаемых подзапросом, устанавливается словами ALL и SOME (ANY).

- ALL определяет, что условие поиска истинно, когда *сравниваемое значение* находится в отношении, определяемом *оператором,* со всеми значениями, возвращаемыми *подзапросом.*

Например: $\simeq$.
WHERE STOLBEZ ALL (SELECT POLE FROM TABLIZA)

определяет, что текущее значение столбца STOLBEZ должно быть больше всех значений в столбце POLE из таблицы TABLIZA

- SOME (вместо него можно указать ANY) что условие поиска истинно, когда *сравниваемое значение* находится в отношении, определяемом *оператором,* хотя бы с одним значением, возвращаемым *подзапросом.*
Например:
WHERE STOLBEZ > SOME (SELECT POLE FROM T ABLIZA)
определяет, что текущее значение столбца STOLBEZ должно быть больше хотя бы одного значения в столбце POLE из таблицы TABLIZA

Задание4. По таблице 'Postavka' определите склад, на который поставлено максимальное количество товара.

```
use sklad
select n_sklad, sum( kol) as sum_kol
from Postavka
group by n_sklad
having sum( kol) >= All (
      select  sum( kol)
      from Postavka
      group by n_sklad)
```

При выполнении подзапроса для каждого склада определяется суммарное количество поставленного товара. Внешний подзапрос также определяет для каждого склада суммарное количество товара и выводит на печать запись, у которой количество товара больше или равно величины каждой записи суммы в подзапросе.
В этом запросе не было готового поля, по которому определяется наибольшее количество. Для реализации запроса использовался оператор ALL.

Задание5. Определить имя поставщика, который поставляет минимальный перечень товара. При построении этого запроса используется две таблицы : Tovar, post. Для того, чтобы узнать сколько наименований товара поставляет каждый поставщик используется конструкция

```
1)select count(kod_tov)
from tovar
group by kod_post
```

Это самый вложенный подзапрос. Чтобы узнать код поставщика, который поставляет минимальный перечень товара, нужно записать код:

```
2)select kod_post
from tovar
group by kod_post
having count(kod_tov) >= All(
select count(kod_tov)
from tovar
group by kod_post )
```

Имя поставщика находится в таблице post. Нам необходимо выбрать тех поставщиков, у которых, код совпадает со значениями, полученными в списке второго подзапроса. Итоговый вид запроса:

```
3)use sklad
select name
from post
where kod_post in (
select  kod_post
from tovar
group by kod_post
having count(kod_tov) >= All(
select count(kod_tov)
```

```
from tovar
group by kod_post )
)
```

Контрольные задания.

1. Определить количество и дату, когда было отпущено минимальное количество товара.
2. Определить имя поставщика, который поставляет товар, с максимальной ценой.
3. Для каждой даты определите сумму отпущенного товара. На основе этого запроса определите дату с максимальной суммой расхода.
4. Определите номер склада, с которого отпускалось минимальное количество товара.
5. Определите код товара, который поставлялся на максимальное количество складов.
6. Сформулируйте и выполните запросы на нахождение минимального и максимального значений в своей базе данных.

Лабораторная работа №29

«*Добавление, изменение, удаление записей в таблицах*»
Цель: «Сформировать навыки и умения построения запросов действий»

Ход Работы

Все запросы, рассмотренные ранее являются запросами на выборку. Добавление, изменение и удаление записей выполняется запросами действиями. Эти запросы не выводят данные на экран, а производят их изменение. Если необходимо просмотреть изменённые данные, то после запроса – действия , выполните запрос выборку.

Язык SQL ориентирован на выполнение операций над группами записей, хотя в некоторых случаях операция может проводиться и над отдельной записью. Поэтому неудивительно, что операторы добавления, изменения и удаления записей в общем случае вызывают соответствующие операции над группами записей.

1.Оператор INSERT

Оператор INSERT применяется для добавления записей в объект. В качестве объекта может выступать таблица БД или просмотр (VIEW), созданный оператором CREATE VIEW. В последнем случае записи могут добавляться в несколько таблиц.
Формат оператора INSERT:

INSERT INTO <объект> [(столбец1[(, столбец2 ...])]
{VALUES «значение1> (, <значение2> ...)) | <оператор SELECT> }

Список столбцов указывает *столбцы,* которым будут присвоены *значения* в добавляемых записях. Список столбцов может быть опущен. В этом случае подразумеваются все столбцы объекта, причем в том порядке, в котором они определены в данном объекте.
Поставить в соответствие столбцам списки значений можно двумя способами. Первый состоит в явном указании значений после слова VALUES, второй - в формирован значений при помощи оператора SELECT.

Явное указание списка значений:
В этом случае оператор INSER T при меняется для добавления одной записи и имеет формат

INSERT INTO <объект> (столбец1 (, столбец2 ...)) VALUES «значение1> (, <значение2> ...)

Значения назначаются *столбцам* по порядку следования тех и других в операторе: первому по порядку столбцу назначается первое значение, второму столбцу - второе значение и Т.д.

Задание 1. Добавить в таблицу tovar новую запись:

INSERT INTO tovar (kod_tov, zena, kod_post) VALUES (‘005’, 350, ‘12’)

Поскольку столбцы таблицы RASHOD указаны в полном составе и именно в том порядке, в котором они перечислены при создании таблицы tovar оператором CREAT TEABLE, оператор можно упростить:

INSERT INTO tovar VALUES (‘005’, 350, ‘12’)

Указание значений при помощи оператора SELECT :

INSERT INTO <объект> (столбец1 (, столбец2 ...)) <оператор SELECT>

При этом *значениями,* которые присваиваются *столбцам,* являются значения, возвращаемые оператором SELECT. Порядок их назначения столбцам аналогичен предыдущей форме оператора INSERT: значение первого по порядку столбца результирующего набора данных оператора SELECT присваивается первому столбцу оператора INSERT, второй - второму и Т.Д. Следует обратить внимание на важную особенность: поскольку оператор SELECT в общем случае возвращает множество записей, то и оператор INSERT в данной форме приведет к добавлению в *объект* аналогичного количества новых записей.

Задание 2. Создайте таблицу itog, в которую добавляются записи, выбранные запросом.
Создание таблицы:

```
use sklad
create table itog(
data varchar(15),
kol int )
```

В эту таблицу добавляется записи за каждую дату с суммарным количеством поступившего товара.

```
use sklad
INSERT INTO itog
SELECT data,sum (kol)
FROM postavka
group by  data
```

2.Оператор UPDATE

Оператор UPDATE применяется для изменения значения в группе записей или - в частном случае - в одной записи объекта. В качестве объекта могут выступать ТБД или просмотр, созданный оператором CREATE VIEW. В последнем случае могут изменяться значения записей из нескольких таблиц.
Формат оператора UPDATE:

UPDATE <объект>
SET столбец1 = <значение1> (,столбец2 = <значение2>...)
(WHERE <условие поиска >)

При корректировке каждому из перечисленных *столбцов* присваивается соответствующее *значение.* Корректировка выполняется для всех записей, удовлетворяющих *условию поиска.* Условие поиска задается так же, как в операторе SELECT

Задание 3. Увеличьте цену товара на 10% для поставщика с кодом ‘11’

```
use sklad
select *
from tovar
where kod_post='11'
```

```
update tovar
set zena=zena+50
where kod_post='11'

select *
from tovar
where kod_post='11'
```

Перед выполнением запроса – действия запускается запрос на выборку, чтобы посмотреть состав данных перед обновлением. После выполнения обновления снова выведем данные для просмотра.

3.Оператор DELETE

Оператор DELETE предназначен для удаления группы записей из объекта. В качестве объекта могут выступать ТБД или просмотр, созданный оператор CREATE VIEW..
Формат оператора DELETE:

DELETE FROM <объект> [WHERE <условие поиска>];

Удаляются все записи из *объекта,* удовлетворяющие *условию* Условие поиска задается так же, как в операторе SELECT.

*Задание 4.*Удалите из таблицы расход все записи за 2 октября.

```
use sklad
select *
from rashod

delete from rashod
where data = '2 октября'

select *
from rashod
```

Перед выполнением запроса – действия запускается запрос на выборку, чтобы посмотреть состав данных перед обновлением. После выполнения обновления снова выведем данные для просмотра.

Контрольные задания.

1. В таблицу ‘поставщик’ добавьте новую запись о поставщике с кодом «17», «Лесная ягода»
2. Создайте таблицу ‘лучший товар’ с полями kod_tov, kol. Выполните последовательность команд: **удалите** все имеющиеся записи из таблицы ‘лучший товар’; **добавьте** в таблицу ‘лучший товар’ код товара и его количество, который больше всего был отпущен со складов (по таблице rashod, вложенный запрос на нахождение максимума); **выведите** на экран содержимое таблицы ‘лучший товар’.
3. Увеличьте количество поставленного товара на 5 единиц для номера поставки ‘04’. Выведите содержимое таблице до обновления и после обновления.
4. В таблицу Rashod добавьте новую запись о продаже товара с кодом ‘001’. Выведите содержимое таблице до добавления и после добавления записи.
5. Из таблицы Rashod удалите все записи с количеством товара меньше 15. Выведите содержимое таблице до удаления и после удаления.
6. В индивидуальной базе данных выполните примеры на добаление, удаление и обновление записей.

Лабораторная работа №30

«*Деловая игра по теме ‘Создание запросов на языке SQL’* »

Цель:

Создать условия для**:**

- обобщения и систематизации знаний, умений и навыков работы учащихся по теме «Создание запросов на языке SQL»;
- совершенствования учебного процесса путем введения заданий от репродуктивных к конструктивным и творческим, в соответствии с требованиями компьютерной подготовки учащихся;
- выработки основных приемов создания запросов;
- использования знаний, полученных на уроках дисциплин: «Базы данных» , «Технология разработки программных продуктов» для создания проекта базы данных и составления документации проекта

Ход Работы

Группа студентов поделена на «информационные отделы» и «руководителей предприятия». Для каждого отдела выдаются задания. Руководители предприятия принимают выполненные задания и следят за ходом их реализации. В конце занятия выявляются проблемы, с которыми столкнулись студенты при конструировании запросов и определяется «информационный отдел», который выполнил наибольшее количество запросов.

1 Постановка главной задачи группам, уточнение их роли в игре.

Цель этапа: довести до учащихся цель предстоящей работы, плана действий и методики работы.

В ходе практической работы вы должны научиться создавать запросы к базе данных, документировать ошибки, которые вы допустили при отладке запросов. «Информационным отделам» будут выданы задания по разработке запросов, которые различаются по уровню сложности. За более сложный запрос даётся большее количество баллов. Каждый информационный отдел состоит из: проектировщика базы данных, администратора б.д., кодировщиков. *Проектировщики* представляют заранее разработанный проект базы данных, который состоит из концептуальной и реляционной модели. *Администратор* координирует работу отдела, консультирует сотрудников при возникновении ошибок. *Кодировщики* создают и отлаживают код запросов.

2. Создание игровой ситуации.

Цель этапа: Постановка технических заданий для каждой группы разработчиков.

Группа студентов разделена на три «информационных отдела» и «руководство предприятия». Каждому отделу заранее было дано задание спроектировать базы данных по определённой тематике. Руководители выдают администраторам Б.Д. технические задания в форме запросов. Объявляются заказчики заданий.

3. ***Актуализация опорных знаний и умений для реализации запросов в форме теста***

Цель этапа: повторение основных терминов и понятий, применяемых при создании запроса.

- Перед началом реализации заданий проводится «планёрка» по обсуждению базовых понятий, необходимых для успешной работы.

- Один из руководителей рассматривает *пример* создания запроса (на доске):

Отобразить фамилии преподавателей с кафедры ‘информатика’

```
Select kafedra.namekaf, prepodavatel.fio
From kafedra, prepodavatel
Where kafedra.kkaf = prepodavatel. Kkaf and kafedra.namekaf= ‘информатика’
```

- Оговариваются наиболее вероятные ошибки: при выборе полей из разных таблиц не указана таблица; условия в выражении where разделены запятыми
- Каждому отделу выдаются тестовые задания, состоящие из 10 вопросов.

4. Представление концептуальной и реляционной модели проектов. Утверждение проектов.
Цель этапа: *проверка домашнего задания, закрепление пройденного материала*
В качестве домашнего задания (повторение темы : создание таблиц) каждый отдел разработал концептуальную и реляционную модель предложенной информационной системы. Для моделей составляется тех. документация. С помощью плаката каждая команда представляет схему базы данных. Если были выявлены ошибки, то они фиксируются. За представление концептуальной модели проектировщик получает 1 балл, за реляционную модель (1 - 2 баллов), за документацию от (2 – 3 баллов). В тех. документации строится концептуальная модель (на плакате), реляционная модель, описывается назначение каждой таблицы и её структура. После представления моделей проектировщик принимает участие в разработке запросов.

<u>База данных для команды №1</u>
Спроектировать информационную систему, которая позволяет хранить данные в структурированной форме о: поставщиках осуществляющих доставку медикаментов для заказчиков; медикаменты разделены по классификациям.

Схема данных

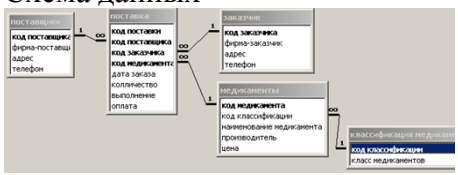


Рис 14 схема данных запроса

<u>База данных для команды 2</u>
Спроектировать информационную систему, которая позволяет хранить данные в структурированной форме о: приёме врачей в поликлинике, во время которого он ставит диагноз больному; приём осуществляют врачи различной специализации.

Схема данных

Рис 15 схема данных запроса

<u>База данных для команды 3</u>
Спроектировать информационную систему, которая позволяет хранить данные в структурированной форме о: перевозках груза для клиентов в различные города.

Схема данных

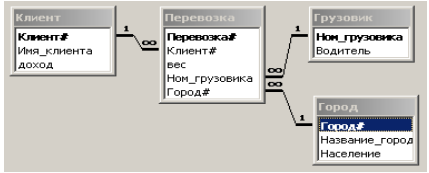


Рис 16 схема данных запроса

5 Работа по реализации запросов проекта.

Цель этапа: получение практических навыков по созданию запросов, развитие творческой активности, групповой деятельности.

51 Реализация запросов на языке SQL.

Каждый отдел получает пакет с тех. заданиями по запросам.

Запросы разделены по типам:

- Простой запрос на основе одной таблицы (1 балл).
- Запрос на выборку данных из нескольких таблиц (2 балла).
- Вложенные запросы (3 балла).
- Запросы на существование хотя бы одной записи (2 балла).

Правильно созданный запрос приносит команде разработчиков соответствующее количество баллов. Если команда затрудняется в реализации запроса, то можно обратиться за помощью к другой команде. Одна правильная подсказка приносит соперникам 1 балл.

52 Документирование запросов и проблем, возникших при их реализации.

Каждый реализованный запрос должен быть документирован (**Приложение В**). Необходимо указать тип запроса: простой – на основе одной таблицы; запрос с выбором данных из нескольких таблиц; сложный запрос с использованием подзапроса. Далее записывается код SQL запроса. В таблице перечисляются ошибки, зафиксированные при отладке запроса.

По мере выполнения запросы представляются «руководителям предприятия», которые делают отметку о реализации запроса в техническом задании.

5.3Решение проблемной ситуации. (администратор базы данных)

Администраторам информационных отделов выдаются задания , в которых необходимо указать ошибки в запросах и сформулировать сам запрос. Правильно выполненное задание приносит команде 2б.

6.Защита и представление проектов

Цель этапа: публичное представление выполненных заданий

Каждый отдел отчитывается о выполненных запросах и публично представляет несколько из них. На этом этапе определяются положительные стороны в работе каждого отдела и группа, получившая максимальное количество балов.

Студентам выставляются оценки на основе полученных балов.

Критерии оценок:

- «неудовлетворительно» если студент набрал менее 2 б;
- «удовлетворительно» если студент набрал 2 б;
- «хорошо» если студент набрал 3 б;
- «отлично» если студент набрал 4 и более балов;

Текст заданий выдаётся как раздаточный материал.

Лабораторная работа №31

«Создание триггеров в ИС «Учебный процесс» для поддержания целостности данных»

Цель: «Сформировать навыки и умения реализации триггеров для поддержания целостности данных»

Ход Работы

1. *Определение триггера*

Триггер – это хранимая процедура особого типа, вызываемая на выполнение в ответ на определённые события. Триггеры подразделяются на два основных типа: триггеры языка определения данных (DDL) и триггеры языка манипулирования данными (DML)

Триггеры DDL активизируются в ответ на внесение каких-либо изменений в структуру б.д пользователями (создание, удаление, изменение структуры таблиц,).
Триггеры DML представляют собой фрагменты кода, которые закрепляются за конкретной таблицей.
В *отличие от хранимых процедур*, при использовании которых необходимо явно вызывать на выполнение определённый код, триггеры вызываются на выполнение автоматически при обнаружении события, связанного с ним. Триггеры не получают параметров и не возвращают значений.

Конструкция FOR (или AFTER) позволяет указать какое действие приводит к запуску триггера.
Триггер |**Insert** этот триггер вызывается на выполнение каждый раз, когда вставляется новая строка в таблицу, за которой закреплён триггер. Для каждой новой строки создаётся её копия и вставляется в таблицу INSERTED. Эта таблица сохраняется до тех пор пока действует триггер (с момента запуска до его завершения).

Триггер |**Delete** вызывается на выполнение каждый раз, при удалении записи. Копия удаляемой записи помещается во временную таблицу Deleted.

Триггер |**Update** вызывается на выполнение при обновлении записи таблицы, для которой он создан. операция модификации строки трактуется как удаление старой версии строки (помещение её в Deleted) и добавление новой строки (помещение её в INSERTED). При этом триггеры на удаление и добавление не запускаются.

2. Условный оператор для триггера в языке SQL-Transact

Один оператор	Несколько операторов

IF условие
Оператор1
Else
Оператор2
	IF условие
	begin
	Оператор1
	Оператор2
	end
	Else
	begin
	Оператор3
	Оператор4
	end

Создадим триггеры на удаление для таблиц со стороны мощности ‘многие’ в схеме данных «Учебный процесс»

Чтобы ввести код триггера для события «удаление» данных в таблице kafedra, щёлкните по изображению + . В появившемся списке выберите папку **Triggers** . Выделите её правой кнопкой мыши и из контекстного меню выберите New Trigger. В представленном шаблоне введите код

```
create trigger del_kafedra
on kafedra
for delete
as
declare @kk varchar(3)
begin
  select @kk=kkaf from deleted
```

```
  delete from prepodavatel where prepodavatel.kkaf=@kk
end
```
Запустите его на выполнение. Для того чтобы просмотреть код откомпеллированного триггера нужно для таблицы *kafedra* выбрать подраздел **Triggers**.
При работе с таблицей *kafedra* во время удаления текущей записи, все записи об этой кафедре в таблице prepodavatel будут удалены.

Задание 1. Триггер для события «удаление» данных в таблице prepodavatel
```
create trigger del_prepodavatel
on prepodavatel
for delete
as
declare @tn varchar(3)
begin
  select @tn=tabn from deleted
  delete from isuchenie where isuchenie.tabn=@tn
end
```
При работе с таблицей prepodavatel во время удаления текущей записи, все записи об этом преподавателе в таблице isuchenie будут удалены.

Задание 2. Триггер для события «удаление» данных в таблице predmet
```
create trigger del_predmet
on predmet
for delete
as
declare @kp varchar(3)
begin
  select @kp=kp from deleted
  delete from isuchenie where isuchenie.kp=@kp
end
```
При работе с таблицей predmet во время удаления текущей записи, все записи об этом предмете в таблице isuchenie будут удалены.

Задание 3. Триггер для события «удаление» данных в таблице isuchenie
```
create trigger del_isuchenie
on isuchenie
for delete
as
declare @kp varchar(3)
declare @tn varchar(3)
declare @ng varchar(3)
declare @vidz varchar(3)
begin
  select @kp=kp, @tn=tabn, @ng=ng, @vidz=vidz from deleted

  delete from uspevaemost where kp=@kp and tabn=@tn and vidz=@vidz and ng=@ng
end
```

Если в созданный триггер необходимо ***внести изменения***, то нужно выбрать таблицу, для которой он создавался, перейти к подразделу **Triggers**, выделить его и из контекстного меню выбрать Modify. Триггер отобразится в окне редактирования. После внесения изменений на панели выбрать кнопку **Execute**.

Контрольные задания

1. создать триггер на удаление для таблицы gruppa.
2. создать триггер на удаление для таблицы student.
3. создать триггер на удаление для таблицы uspevaemost. В теле триггера выполняется следующий код: при удалении студента из таблицы uspevaemost в таблице gruppa изменяется pball для группы студента.
4. создать триггер на обновление для таблицы uspevaemost. В теле триггера выполняется следующий код: при изменении оценки студента в таблице uspevaemost его pball в таблице student обновляется

Лабораторная работа №32

«Создание триггеров в ИС «Учебный процесс» для поддержания целостности данных»

Цель: «Сформировать навыки и умения реализации триггеров, реализующих бизнес - правила»

Ход Работы

Создадим таблицу “svodnaj”, которая содержит количество всех оценок из таблицы «Успеваемость».

CREATE TABLE svodnaj (
Ozenka int not null,
Kol int not null,
Primary key (Ozenka))

.1 добавлении новой записи

При добавлении новой записи в «Успеваемость» (Триггер на добавление для «Успеваемость») просматривается таблица “svodnaj”, **если** строка с такой оценкой уже есть, то изменяется её количество

До вставки

ozenka	kol
3	4

После вставки

ozenka	kol
3	5

В противном случае в таблицу “svodnaj” добавляется новая строка с этой оценкой и количеством: 1

svodnaj

ozenka	kol
3	1

Пример триггера

```
CREATE trigger InsUspevaemost
On uspevaemost
for INSERT
as
declare @k int
begin
select @k=ozenka  from inserted
if @k>1 AND @k<=5
begin
  IF exists (select s.ozenka from svodnaj s where s.ozenka=@k )// УЖЕ ЕСТЬ
     UPDATE  svodnaj set kol=kol+1 where svodnaj.ozenka=@k
```

```
  else
     insert into  svodnaj ( ozenka,kol) values (@k, 1)

end
else
 BEGIN
  RAISERROR ('ENTER OZENKA ' ,16,1)
  ROLLBACK TRAN
 END
End
```

2 Удаление записи

При удалении записи в «Успеваемости» (Триггер на удаление для «Успеваемость»), в таблице “svodnaj” для удаляемой оценки уменьшается количество

До удаления

ozenka	kol
3	4

Удаление из «Успеваемости»

ng	ns	kp	ozenka
101	01	02	3

После удаления

ozenka	kol
3	3

Пример триггера

```
create trigger DelUspevaemost
on uspevaemost
for delete
as
declare @oz  int
declare  @k int
begin
select @oz=ozenka  from deleted
select @k=kol  from svodnaj where ozenka=@oz

UPDATE  svodnaj set kol=kol-1 where ozenka=@oz
if @k=1
  delete from svodnaj where ozenka=@oz

end
```

3 Обновление записи

При обновлении записи в «Успеваемости» (Триггер на обновление для «Успеваемость»), в таблице “svodnaj” для удаляемой оценки изменяется количество

До обновления

ozenka	kol
3	4

После обновления

ozenka	kol
3	**3**
5	**1**

Старое значение «Успеваемости»

ng	ns	kp	ozenka
101	01	02	3

Новое значение «Успеваемости»

ng	ns	kp	ozenka
101	01	02	5

Пример триггера

```
create trigger [UpdUspevaemost]
on [uspevaemost]
for update
as
declare @prev_oz  int
declare  @new_oz int
declare  @k int
begin
select @prev_oz=ozenka from deleted
select @new_oz=ozenka from inserted
if @prev_oz<>@new_oz
begin
  // изменение старой оценки: как при удалении
select @k=kol  from svodnaj where ozenka=@prev_oz
UPDATE  svodnaj set kol=kol-1 where ozenka=@prev_oz
if @k=1
  delete from svodnaj where ozenka=@prev_oz

// изменение новой оценки: как при добавлении
if @new_oz>1 AND @new_oz<=5
begin
  IF exists (select s.ozenka from svodnaj s where s.ozenka=@new_oz )
     UPDATE  svodnaj set kol=kol+1 where svodnaj.ozenka=@new_oz
  else
     insert into  svodnaj ( ozenka,kol) values (@new_oz, 1)

end
else
 BEGIN
  RAISERROR ('ENTER OZENKA ' ,16,1)
  ROLLBACK TRAN
 END

end

end
```

Контрольные задания

1. Создать триггер на удаление в индивидуальной базе данных для всех таблиц со стороны «одного» в мощности отношений.
2. Создать триггер на обновление в индивидуальной базе данных для всех таблиц со стороны «одного» в мощности отношений.
3. Создать триггер на добавление записи в индивидуальной базе данных.

Лабораторная работа№33

«Хранимые процедуры в SQL Server»

Цель: «познакомить студентов с понятием хранимая процедура, рассмотреть основные методы их создания»

Ход Работы

Хранимая процедура (stored procedure) – это именованный набор команд Transact-SQL, хранящийся непосредственно на сервере и представляющий собой самостоятельный объект базу данных. Она существует независимо от таблиц или каких-либо других объектов баз данных. Хранимая процедура может быть вызвана клиентской программой, другой хранимой процедурой или триггером. Возможно управлять правами доступа пользователей к хранимы процедурам. Прежде чем выполнить хранимую процедуру, сервер генерирует для неё так называемый план исполнения (execution plan), выполняет её оптимизацию и компиляцию. Выполняется кэширование плана исполнения процедуры, а также оптимизированного компилирования кода. Использование хранимых процедур реализует принцип модульного проектирования.

1. Типы хранимых процедур

Системные хранимые процедуры (system stored procedures) – это хранимые процедуры, поставляемые в составе SQL Server 2000. Предназначены для выполнения различных административных действий. Такие процедуры имеют префикс sp_ . Они хранятся в база данных master.

Пользовательские хранимые процедуры (user-defined stored procedures) – это процедуры созданные пользователями, реализующие те или иные действия (полноценный объект баз данных). Следствием этого является то, что каждая хранимая процедура хранится в конкретной базе данных.

Временные хранимые процедуры (temporary stored procedures) – эти процедуры существуют лишь некоторое время, после чего автоматически уничтожаются сервером. Бывают локальные и глобальные.

Локальные временные хранимые процедуры(local temporary stored procedure) – могут быть вызваны только из того соединения, в котором они были созданы. При создании такой процедуры необходимо дать ей имя, начинающееся символом #. Они хранятся в база данных tempdb и автоматически удаляются при отключении пользователя.

2. Синтаксис процедуры

Для создания хранимой процедуры на языке Transact-SQL используется SQL-оператор

CREATE PROCEDURE
Синтаксис данного оператора для MS SQL Server 2000

```
CREATE PROC [ EDURE ] procedure_name [ ; number ]
[ { @parameter data_type }
[ VARYING ] [ = default ] [ OUTPUT ]
] [ ,...n ]
[ WITH
{ RECOMPILE | ENCRYPTION | RECOMPILE , ENCRYPTION } ]
[ FOR REPLICATION ]
AS
```

Аргументы:

procedure_name – имя создаваемой процедуры. Используются префиксы sp_, #, ## (значение указано выше). Как видно из синтаксиса, не допускается указывать имя владельца, которому будет принадлежать создаваемая процедура, а также имя базы данных, в которой должна быть размещена процедура. Чтобы разместить хранимую процедуру в конкретной базе данных нужно создать её в контексте этой базы данных.

;number – идентификационный номер хранимой процедуры, однозначно определяющий её в группе процедур. На пример: orderproc;1, orderproc;2. Процедура orderproc объединяет в себе две

процедуры. При вызове DROP PROCEDURE orderproc, будут удалены все процедуры этой группы.
@parameter – имя параметра, который будет использоваться создаваемой хранимой процедурой для передачи входных или выходных данных. Имена параметров должны начинаться с символа @. В одной хранимой процедуре может использоваться до 1024 параметров.
data_type – тип данных, который будет иметь соответствующий параметр хранимой процедуры. Можно использовать все типы данных, включая text, ntext и image и пользовательские типа данных. Однако, тип данных cursor может использоваться только как выходной параметр (с указанием ключевого слова OUTPUT).
VARYING – ключевое слово, которое используется совместно с параметром OUTPUT, имеющим тип данных cursor. В качестве выходного параметра будет представлено результирующее множество.
default – значение, которое будет принимать соответствующий параметр по умолчанию. При вызове процедуры, явно можно будет не указывать значение соответствующего параметра. Будет использовано значение, созданное с помощью этого параметра.
OUTPUT – его наличие указывает, что параметр предназначается для возвращения данных из хранимой процедуры. Но этот параметр также может использоваться и для передачи значений в процедуру. Значение соответствующего параметра при вызове процедуры может быть задано только с помощью локальной переменной. Нельзя использовать выражения и константы, допустимые для обычных параметров.
n – количество определённых параметров.
{RECOMPILE | ENCRYPTION | RECOMPILE, ENCRYPTION}
RECOMPILE - Указывает, что SQL Server должен создавать план выполнения хранимой процедуры при каждом её вызове.
ENCRYPTION - Указывает, что SQL Server должен выполнить кодирование хранимой процедуры. Обратите внимание, сохранённые процедуры, созданные с выбором ENCRYPTION не могут рассматриваться с sp_helptext.
FOR REPLICATION – используется при репликации данных и включение создаваемой хранимой процедуры в качестве статьи в публикацию. Этот параметр не может быть использован совместно с параметром RECOMPILE.
AS – ключевое слово, свидетельствующее о начале тела процедуры

3. Вызов хранимой процедуры
Для вызова хранимой процедуры используется оператор EXECUTE.
Синтаксис оператора:

EXEC procedure_name
[[@parameter =] { value | @variable [OUTPUT] | [DEFAULT] }]

Указывая ключевое слово OUTPUT, вы предписываете присвоить соответствующей локальной переменной (внешней по отношению к процедуре) при завершении хранимой процедуры значение соответствующего параметра. При этом значения параметров могут изменяться в ходе процедуры.

DROP PROCEDURE – удаляет процедуру с сервера.

Задание 1.
Создать процедуру, которая выводит среднюю оценку по каждому предмету. Функция не принимает и не возвращает параметров
Выберите базу uch_proc , откройте папку Programmbility, выберите Stored Procedures.
Из контекстного меню для Stored Procedures выберите New Stored Procedure.

```
CREATE PROCEDURE sred_ozenka
AS
BEGIN
```

```
        SELECT kp, avg(ozenka)
   from uspevaemost
   group by kp
END
```

Для компиляции нажмите !Execute. Процедура появится в списке.
Для выполнения процедуры в редакторе запросов введите команду Exec sred_ozenka.

Задание 2.
Создать процедуру, которая выводит среднюю оценку по предмету, переданному через параметр

```
CREATE PROCEDURE sred_ozenka_pred
 @pred varchar(20)
AS
BEGIN

        SELECT  avg(uspevaemost.ozenka)
   from uspevaemost, isuchenie, predmet
   where uspevaemost.ng=isuchenie.ng and uspevaemost.kp=isuchenie.kp and
uspevaemost.tabn=isuchenie.tabn
        and uspevaemost.vidz=isuchenie.vidz and isuchenie.kp= predmet.kp and predmet.np=@pred
END
```

Для выполнения процедуры в редакторе запросов введите команду
```
 Exec sred_ozenka_pred 'информатика'
```

Задание 3.
Создать процедуру, которая выводит через выходной параметр фамилию студента, который получал наибольшее количество двоек в группе, переданной через входной параметр. Итак, процедура будет иметь один входной и один выходной параметр. Около выходного параметра указывается ключевое слово output

```
CREATE PROCEDURE neud
@ngr varchar(3) ,   входной параметр – номер группы
@fio_s varchar(20) output  выходной параметр
AS
BEGIN

        SELECT  @fio_s=fio
   from student
   where ng=@ngr and ns in(
   SELECT  ns
   from uspevaemost
   group by ns, ozenka
   having ozenka=2 and count(kp)>= all (
     SELECT count(kp)
      from uspevaemost
      group by ns, ozenka, ng
      having ozenka=2 and ng=@ngr) )

END
```

Для выполнения процедуры в редакторе запросов введите команду
```
use uch_proc
declare @f varchar (20)
exec neud '101', @fio_s=@f output
print @f
```

Задание 4.
Создать процедуру, которая выводит коды предметов и вид занятий, по которым получал двойки самый неуспевающий студент указанной группы. Самый неуспевающий студент определяется процедурой neud . Т. е. создаваемая процедура будет вызывать neud

```
CREATE PROCEDURE neud_pred
@ngg varchar(3)

AS
declare @f varchar (20)
BEGIN
        exec neud  @ngg, @fio_s=@f output    // вызов процедуры neud
select @f                          // вывод фамилии неуспевающего

// вывод кода предмета и вида занятия,по которым он получил 2

select uspevaemost.kp, uspevaemost.vidz
from uspevaemost, student
where uspevaemost.ng= student.ng and uspevaemost.ns= student.ns and student.fio=@f

END
```

Для выполнения процедуры в редакторе запросов введите команду

```
use uch_proc
exec neud_pred '101'
```

Контрольные задания

1. Создать процедуру, которая в базе данных ‘sklad’определяет количество товара в таблице ‘tovar’.
2. Создать процедуру, которая по входному параметру наименование поставщика выводит наименование, код, цену товара.
3. Создать процедуру, у которой два параметра: входной параметр - наименование товара, выходной – цена товара. По входному параметру определить цену товара.
4. Создать процедуру_2, которая выводит товар и его цену большую средней цены. Среднюю цену возвращает процедура_1 (как в задании 4).
5. База данных Учебный процесс. Создать процедуру, которая по входному параметру номер группы определяет количество положительных оценок (выходной параметр)
6. База данных Учебный процесс. Создать процедуру, которая через выходной параметр возвращает наименование предмета с максимальным количеством часов. Входного параметра нет.
7. База данных Учебный процесс. Создать процедуру, которая по входному параметру фамилия студента, возвращает номер группы.
8. Сформулируйте и выполните процедуры в индивидуальной базе данных (без параметров, с входным и выходным параметром)

Лабораторная работа №34

«Создание запросов с использованием внешнего соединения»

Цель: «выявить отличия внешнего и внутреннего соединений, создать запросы для внешних соединений»

Ход Работы

При внутреннем соединении, рассмотренном ранее, таблицы связывались в условном операторе предложением:

<имя столбца таблицы 1> = < имя столбца таблицы 2>

Например: *отобразить фамилии преподавателей с кафедры ‘информатика’*

Select prepodavatel.fio
From kafedra, prepodavatel
Where kafedra.kkaf = prepodavatel.kkaf and kafedra.nkaf=*'информатика'*

В этом случае осуществляется декартово произведение таблиц kafedra, prepodavatel и из полученного набора данных отбираются записи, удовлетворяющие условию поиска (kafedra.kkaf = prepodavatel.kkaf)

Существует и другой вид соединения таблиц – **внешнее соединение**. Оно определяется предложением From согласно спецификации
Select * from <таблица 1> <вид соединения> **JOIN** < таблица 2>
ON <условие поиска>

Внешнее соединение похоже на внутреннее, но в результирующий набор включаются записи *ведущей* таблицы соединения, которые объединяются с пустым множеством записей другой таблицы. Какая из таблиц будет ведущей определяет *вид соединения*:

- Left - (левое внешнее соединение), когда ведущей является таблица1 (расположенная слева от вида соединения);
- Right – (правое внешнее соединение), когда ведущей является таблица2 (расположенная справа от вида соединения);

Задание 1: создайте две таблицы

```
Create table tab1(
  P1 varchar(1),
  P2 varchar(1),
  P3 int
)
```

```
Create table tab2(
  P2 varchar(1),
  P4 varchar(1),
  )
```

Заполните данными эти таблицы.

tab1

P1	**P2**	**P3**
a	x	400
b	x	200
c	y	500
d		

tab2

P2	**P4**
x	1
y	2
z	2

Тогда выполнение оператора Select, реализующего внешнее левое соединение
Select tab1.P1, tab1.P2, tab2.P4 **from** tab1 **Left JOIN** tab2
ON tab1.P2= tab2.P2
Приведёт к выдаче результирующего набора

tab1.**P1**	*tab1*.**P2**	*tab2*.**P2**
a	x	1
b	x	1
c	y	2
d		

Пунктиром показаны столбцы ведущей таблицы *tab1*. Как видно для записи таблицы tab1, где столбец tab1.P1 имеет значение «d», нет парных записей в таблице tab2, для которых бы удовлетворялось условие поиска tab1.P2= tab2.P2. Поэтому данная запись таблицы tab1 показана в соединении с пустой записью.
В то же время, выполнение оператора Select, реализующего внешнее правое соединение
Select tab1.P1, tab1.P2, tab2.P4 **from** tab1 **Right JOIN** tab2
ON tab1.P2= tab2.P2
Приведёт к выдаче результирующего набора

tab1.**P1**	tab1.**P2**	Tab2.**P2**
a	x	1
b	x	1
c	y	2
		2

Пунктиром показаны столбцы ведущей таблицы *tab2*. Как видно, для записи таблицы tab2, где столбец tab2.P1 имеет значение 'z' и столбец tab2.P2 имеет значение «2», нет парных записей в таблице tab1, для которых бы удовлетворялось условие поиска tab1.P2= tab2.P2. Поэтому данная запись таблицы tab2 показана в соединении с пустой записью.

Задание 2:Построить внешнее соединение по таблице uspevaemost с таблицей student, т.е. показать студента, соответствующего каждой оценке.

```
use uch_proc
Select uspevaemost.ozenka,uspevaemost.kp, student.ns, student.ng from uspevaemost Left JOIN
student
ON uspevaemost.ns=student.ns and uspevaemost.ng=student.ng
```
Или
```
use uch_proc
Select uspevaemost.ozenka,uspevaemost.kp, student.ns, student.ng from student Right_JOIN
uspevaemost
ON uspevaemost.ns=student.ns and uspevaemost.ng=student.ng
```

Результат запроса:

Ozenka	**kp**	**ns**	**ng**
3	01	01	101
4	01	01	101
3	03	01	101

Задание3:Построить внешнее соединение по таблице student с таблицей uspevaemost, т.е. показать все оценки по каждому студенту.

```
use uch_proc
Select uspevaemost.ozenka,uspevaemost.kp, student.ns, student.ng from student Left_JOIN
uspevaemost
ON uspevaemost.ns=student.ns and uspevaemost.ng=student.ng
```
Или
```
use uch_proc
Select uspevaemost.ozenka,uspevaemost.kp, student.ns, student.ng from uspevaemost Right_JOIN
student
ON uspevaemost.ns=student.ns and uspevaemost.ng=student.ng
```

Результат запроса:

Ozenka	**kp**	**ns**	**ng**
3	01	01	101
4	01	01	101
3	03	01	101
NULL	NULL	03	102
NULL	NULL	01	103
NULL	NULL	02	103
NULL	NULL	01	104

Задание 4. Используя внутреннее соединение между таблицами student ,isuchenie, predmet и условия существования хотябы одной записи (exists) в наборе данных, вывести фамилии студентов, которые не получали оценок по изучаемым предметам.

```
use uch_proc
Select  student.fio, student.ng , predmet.np,isuchenie.vidz
from student ,isuchenie, predmet
where  isuchenie.ng=student.ng and predmet.kp=isuchenie.kp and not exists(
select uspevaemost.ns
from uspevaemost
where uspevaemost.ns=student.ns and uspevaemost.kp=isuchenie.kp and
isuchenie.vidz=uspevaemost.vidz

)
```

Аристов	101	математика	лек
Аристов	101	математика	пр
Аристов	101	история	лек
Аристов	101	ин яз	пр
Аристов	101	философия	лек

.

Контрольные задания

1. Построить внешнее соединение по таблице kafedra с таблицей prepodsavatel, т.е. показать состав каждой кафедры. Построить левое внешнее соединение и правое соединение.
2. Построить внешнее соединение по таблице predmet с таблицей isuchenie, которое отобразит в каких группах и с каим количеством часов преподаются предметы. Построить левое внешнее соединение и правое соединение
3. Построить внешнее соединение по таблице gruppa с таблицей student, которое покажет состав групп. Построить левое внешнее соединение и правое соединение.
4. Сформулировать и выполнить запросы в индивидуальной базе данных с использованием левого внешнего соединения.
5. Сформулировать и выполнить запросы в индивидуальной базе данных с использованием правого внешнего соединения.

Лабораторная работа №35

«Транзакции. Уровни изоляций транзакций. Резервирование данных»

Цель: «познакомить студентов с основными принципами администрирования баз данных в SQL Server, изучить основные приемы при резервировании баз данных и журналов транзакций, а также их восстановления»

Ход Работы

1.Уровни изоляции транзакций

Разработчику предоставляется возможность выбора уровня изоляции транзакций:

- *READ COMMITTED(*значение, применяемое по умолчанию*)*
- *READ UNCOMMITTED*
- *SERIALIZABLE*

Для выбора конкретного уровня изоляции транзакций применяется оператор:

SET TRANSACTION ISOLATION LEVEL < *READ COMMITTED* | *READ UNCOMMITTED* | *SERIALIZABLE* >

Опция *READ COMMITTED*
При использовании этой опции все созданные разделяемые блокировки автоматически освобождаются после завершения выполнения оператора, в котором они были созданы.

Задание 1 "уровень изоляции транзакции «READ COMMITTED» "

Установите соединение с базой данных «Учебный процесс».
В окне редактора нового запроса (New Query) введите команду начала транзакции и обновления записи для группы 101

```
begin tran
use uch_proc
update gruppa set kol=35 where ng='101'
```

выполните запрос (! Execute). Запись обновляется, но транзакция не завершена. Транзакция блокирует запись.
В новом окне введите код запроса для отображения записи из таблицы «gruppa» с номером '101'.

```
use uch_proc
select * from gruppa  where ng='101'
```

Внизу на панели будет отображаться процесс «зависания» выполнения запроса: Executing query. При обновлении на запись была установлена «исключительная блокировка», которая не даёт возможность выполнится «разделяемой блокировке» чтения данных.
Если в новом окне редактора запросов ввести код, отображающий информацию о группах кроме 101, то он будет выполнен.

```
use uch_proc
select * from gruppa  where ng<>'101'
```

Результат

ng	kol	pbal
102	32	4,5
103	29	4,8
104	35	4,4
105	45	4,8

.
.
Все остальные записи таблицы оказались не заблокированы.

В окне, где запускалась транзакция завершите её подтверждением : **commit tran**. После этого код запроса для отображения записи из таблицы «gruppa» с номером '101' будет выполнен.
Результат

ng	kol	pbal
101	35	4,5

Задание 2 "уровень изоляции транзакции «READ UNCOMMITTED» "

В окне редактора нового запроса (New Query) введите команду начала транзакции и обновления записи для группы 101
Транзакция № 1

```
begin tran
use uch_proc
update gruppa set kol=32 where ng='101'
```

выполните запрос (! Execute). Запись обновляется, но транзакция не завершена.
В новом окне введите код запроса для отображения записи из таблицы «gruppa» с номером '101'.
Для выборки данных запустим транзакцию с уровнем изоляции *READ UNCOMMITTED*
Транзакция №2

```
set transaction isolation level read uncommitted
begin tran
use uch_proc
select * from gruppa  where ng='101'
```

ng	kol	pbal
101	32	4,5
102	32	4,5
103	29	4,8

.
.
Транзакции с этим уровнем изоляции удалось прочитать «незафиксированные» данные.
В окне начала транзакции на обновление введём команду «отката» изменений: rollback tran.
Просмотрим данные из таблицы «группа» после отката транзакции

ng	kol	pbal
101	35	4,5
102	32	4,5
103	29	4,8

.
2. *Резервирование данных и журналов транзакций*

Одни из лучших способов управления резервной копией информации предоставляется SQL Server Enterprise Manager. Для начала работы с системой резервирования выберите базу данных, которую требуется сохранить, правой кнопкой мыши и выполните команду Tasks | Back Up. Появится диалоговое окно SQL Server Backup, где расположены вкладки *General и Options*.

Основными видами резервирования в SQL Server являются:

- **Full (полная копия базы данных).** В этом случае сохраняются любые действия над базой данных, производимые в процессе копирования, все незафиксированные транзакции и, естественно, все данные. Для создания первой копии необходимо выбрать именно этот вариант.
- **Differential (дифференциальная копия).** В этом случае SQL Server сохраняет только те части базы данных, которые изменились со времени создания полной копии и имеющиеся в

журнале незафиксированные транзакции. В этом случае резервное копирование производится быстрее, чем в предыдущем, но полная копия должна существовать.

- Transaction log (копия журнала транзакций). В этом случае сохраняются все изменения, произошедшие в базе данных.

Для эффективной работы с базой данных, спроектированной в SQL Server, обязательно должны резервироваться следующие объекты:

1) главная база данных;
2) все базы данных, информацию из которых нельзя потерять;
3) все журналы транзакций для каждой базы данных, которые работают в системе или подвергаются тестированию.

Если база данных была повреждена в силу некорректных действий пользователя или вследствие сбоев в работе аппаратной и программной части системы, то необходимо провести восстановление системы. Основными действиями для её восстановления являются:

1) восстановление главной базы данных;
2) восстановление последней полной копии базы данных, если она была сделана;
восстановление журналов транзакций, которые были зарезервированы пользователем, начиная с момента получения последней резервной/

Задание 3. Создать резервную копию базы данных ‘sklad’.

- В выберите базу данных ‘sklad’, для которой нужно выполнить резервное копирование.
- Выполните последовательность действий: **Tasks → Back Up**.
- В результате чего на экран будет выведено диалоговое окно **SQL Server Backup**, содержащее вкладки **General** и **Options**.
- В поле **Database** укажите имя базы данных, подлежащей резервированию; в поле **Name** – имя ее резервной копии; **Description** – задается описание, использование которого облегчит работу с резервируемой базой данных.
- С помощью клавиши Tab перейдите в область **Backup type** и укажите тип создаваемой резервной копии (Full, Differential, Transaction log).
- Затем активизируйте область окна **Destination**, в которой указывается область диска или диск для размещения резервной копии базы данных.
- Нажмите кнопку **Add** в группе **Destination**.
- В диалоговом окне **Select Backup Destination** определите месторасположения резервной копии базы данных: в отдельном файле или на внешнем устройстве. Для сохранения резервной копии в отдельном файле достаточно указать полный путь доступа к нему; во втором случае – указывается и тип носителя, на котором будет сохраняться копия, и новое имя файла. Для возврата в диалоговое окно **SQL Server Backup** достаточно нажать кнопку ОК.

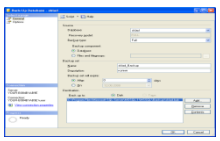


Рис 17 в диалоговое окно **SQL Server Backup**

Задание 4 Выполнить восстановления базы данных ‘sklad’.

- В начале операции восстановления базы данных выполните следующую последовательность команд: **Tasks → Restore ->Database.**
- После чего на экран будет выведено диалоговое окно **Restore Database**, которое также содержит вкладки **General** и **Options**.
- Затем из предлагаемого списка выберите базу данных для восстановления.
- После заполнения диалогового окна **Restore Database** нажмите на кнопку ОК. Если все параметры для восстановления базы данных указаны верно, то на экран будет выведено окно сообщений, в котором отображается процесс восстановления базы данных.

Контрольные задания

1. Выполните резервное копирование индивидуальной базы данных.
2. Выполните восстановление индивидуальной базы данных.
3. В индивидуальной базе данных запустите транзакцию с уровнем изоляции *«READ COMMITTED»*
4. В индивидуальной базе данных запустите транзакцию с уровнем изоляции *READ UNCOMMITTED*

Лабораторная работа №36

«Обеспечение безопасности пользователей в SQL Server»

Цель: «изучить основные приемы администрирования баз данных в SQL Server, рассмотреть возможность добавления новых пользователей в систему и разграничения их полномочий»

Ход Работы

Решение задач обеспечения безопасности данных остается актуальным и при использовании SQL Server.

В системе безопасности SQL Server выделяется два уровня: сервера и базы данных. В общем случае принять рассматривать следующие три типа безопасности со стороны сервера в SQL Server.

Стандартная безопасность, при которой система отвечает за администрирование серверной части приложения. Для обеспечения безопасности данных в SQL Server со стороны сервера используются следующие средства обеспечения безопасности:

- идентификация (identification) выполняется присвоением субъекту какого-либо формального идентификатора, например имени учётной записи (login) и пароля (password);
- аутентификация (authentication) подлинности пользователя с помощью пароля;
- создание учетных записей (login);
- использование встроенных ролей сервера (fixed server roles).

Интегрированная безопасность данных в SQL Server осуществляется средствами сетевых версий операционной системы Windows. В этом случае для обеспечения безопасности используются списки контроля доступа ACL (Access Control List). Основным достоинством этого подхода при администрировании приложений, функционирующих в SQL Server, является то, что пользователь получает доступ ко всем ресурсам домена операционной системы Windows при задании пароля и использовании методов шифрования при передаче данных по сети. При таком методе администрирования серверной части приложения, функционирующего в SQL Server, при регистрации на сервере операционной системы одновременно осуществляется и автоматическое подключение к SQL Server, которое называется доверительной *регистрацией.*

При доверительной регистрации пользователя в SQL Server он может подключиться к базе данных одним из следующих способов:

1) зарегистрированный пользователь при идентификации (опознании) имени;
2) стандартный пользователь с именем Guest;
3) системный администратор Windows – SA (System Administrator).

В *стандартном режиме* обеспечения безопасности данных контроль и управление учетными записями, используемыми для доступа к серверу, осуществляет SQL Server. Кроме того, он самостоятельно выполняет аутентификацию пользователей, хранит все данные о правах доступа, именах и паролях.

При использовании стандартного режима в SQL Server применяются два уровня доступа пользователей: первый уровень – учетные записи, второй – записи пользователей.

Учетные записи используются для подключения к серверу самого SQL Server, а область их действия распространяется на весь сервер. Учетная запись в SQL Server ассоциируется с паролем, позволяющем получить доступ к любой базе данных сервера.

Записи пользователя служат для контроля за правами доступа к определенным ресурсам сервера, например, таблицы, хранимые процедуры, триггеры и т. д. Записи пользователя могут быть созданы в одной или нескольких базах данных одновременно, но независимо друг от друга.

Права доступа (permission) представляют собой разрешение на получение доступа к определенному объекту базы данных, в частности, таблице, представлению и т.д. Они разрешают выполнять пользователям те или иные операции с объектами базы данных. Для каждого из объектов базы данных имеется несколько видов прав доступа. Права доступа ко всем объектам базы данных автоматически предоставляются владельцу или разработчику базы данных. Наибольшими правами доступа обладают следующие категории пользователей:

- системный администратор (SA) имеет все права доступа ко всем объектам во всех базах данных сервера;
- владелец базы данных (Database Owner) имеет все права доступа ко всем объектам его базы данных.

Права доступа к объекту (object permission) представляют собой разрешения на выполнение конкретных действий над объектами базы данных, например, таблицами, представлениями, запросами, хранимыми процедурами.

Поддерживаются два вида учётных записей (имён входа) подключения к серверу – учётные записи сервера, создаваемые на основании учётных записей операционной системы, и учётные записи сервера, создаваемые для прямого подключения к серверу.
Учётную запись можно создать либо с помощью **SQL Server Management Studio,** либо командой create login.

1. Учётная запись входа SQL Server

Сами по себе имена входа не имеют доступа к базам данных SQL Server, они лишь позволяют подключиться к SQL Server. Эти имена входа дают разрешение на выполнение некоторых действий. Эти действия встроены в такие серверные роли как sysadmin, diskadmin, dbcreaster, securityadmin.
Sysadmin – разрешено выполнять любые действия в SQL Server. Все администраторы Windows являются членами фиксированной серверной роли sysadmin.
Dbcreaster – разрешено создание, удаление, изменение и восстановление любой базы данных.
Diskadmin – предназначена для управления дисковыми файлами.
Securityadmin – разрешено управлять именами входа и их свойствами.
Создание учётной записи сервера на основании учётной записи операционной системы:

```
CREATE LOGIN имя
From windows
[ With default database ]
```

После With default database указывается имя базы данных, с которой пользователь работает по умолчанию.

Задание 1.

Создайте учётную запись для своей фамилии на основании учётной записи операционной системы, например

```
Use master;
CREATE LOGIN Ivanov
From windows
With default_database = sklad
```

Задание 2.

Создайте учётную запись сервера для прямой аутентификации на SQL Server.

```
use master;
create login [teddy]
with
password = 'pasteddy',
 default_database = sklad
```

Задание 3

Учётную запись сервера можно создать средствами **SQL Server Management Studio** .
Для этого откройте папку Security->Logins вашего SQL Server.

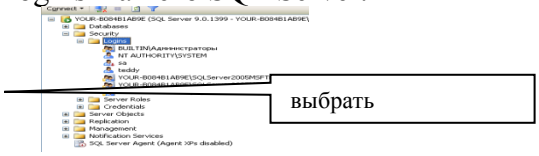


Рис 18 Создание учётной записи средствами **SQL Server Management Studio**

Для Logins из контекстного меню выбрать New Login, откроется окно редактирования учётной записи.
Появится окно « Login New ». здесь на вкладке **General** можно создать учётную запись входа в SQL Server через учётную запись windows

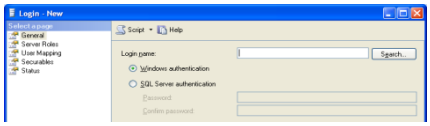


Рис 19 Создание учётных записей на основании операционной системы

Если учётная запись создаётся для прямого подключения к серверу, то переключатель устанавливается в положение

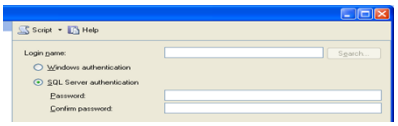


Рис 20 Создание учётных записей для прямого подключения к серверу

Здесь заполняются поля ввода пароля и имя учётной записи.
На вкладке Server Roles необходимо указать серверную роль созданной учётной записи, т.е. права на сервере.

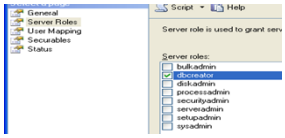


Рис 21 Назначение серверной роли для учётной записи сервера.

После всех установок нажмите на ‘Ok’. Ваша учётная запись появится в списке Logins.

2.Управление учётными записями пользователей базы данных (users)

Учётные записи пользователей базы данных являются тем субъектом системы, которому назначаются разрешение на доступ к объектам базы данных.
На основании одной учётной записи сервера может быть создана в одной базе данных только одна учётная запись пользователя.
Пользователя базы данных можно создать средствами **SQL Server Management Studio,** либо командой create user.
После создания пользователя базы данных его можно включить в одну из *ролей*:
Db_accessadmin – разрешено предоставлять и лишать права доступа учётным записям SQL Server.
Db_datareader – разрешено чтение всех пользовательских таблиц.
Db_datawriter разрешено добавление, изменение, удаление всех пользовательских таблиц.
Db_denydatawriter – запрещено добавление, изменение, удаление всех пользовательских таблиц.
Db_denydatareader – запрещено чтение всех пользовательских таблиц.
Db_securityadmin – разрешено изменять членство в *ролях* базы данных и управлять разрешениями.

Существует также особая роль, в которую системные администраторы не могут явно включать пользователей – роль public. Все создаваемые пользователи включаются в эту роль. Данная роль

предоставляет пользователям конкретной базы данных разрешения по умолчанию. Для защиты данных необходимо минимизировать разрешения роли public.

Задание 4

Создадим пользователя базы данных sklad для учётной записи teddy с помощью оператора create user

use sklad;

create user [user_teddy]

for login teddy

Имя пользователя user_teddy .

3. Понятие схемы SQL Server 2005.

Схема – это средство группировки объектов (не пользователей), позволяющее обращаться с набором объектов как с единым целым.

В *SQL Server 2005* каждая база данных состоит из схем “Shemas”. Схема содержит таблицы, триггеры, виды и т.д. Все объекты нашей базы данных sklad принадлежат одной схеме. Владельцем этих объектов является только она. Её имя – dbo, что означает «владелец базы данных». В SQL Server кроме схем баз данных существуют пользователи базы данных. Пользователи владеют схемами. Несколько пользователей могут владеть одной и той же схемой данных. Схемы используются для ограничения видимости объектов. Объекты можно перемещать из одной схемы в другую в пределах одной и той же базы данных.

Для *создания новой схемы*:

- В окне обозревателя объектов Object Explorer выберите базу данных.
- В выбранной базе раскройте узел Securite (безопасность). Сделайте щелчок правой кнопкой мыши по узлу Schemas (Схемы). Появится меню.
- Выберите в меню New Schemas. Появится окно, содержащее три страницы.
- На первой странице General (общие) в поле Schema name (имя схемы) введите её имя. В поле Schema Owner (владелец схемы) укажите имя её владельца, выбранного среди пользователей базы .
- Для пользователя владельца предназначена кнопка Search. После выбора пользователя нажмите Ок.

Задание 4

Создадим схему для пользователей базы данных sklad, которые должны иметь доступ только к таблицам: Tovar, post. Схему назовём ‘tovars’, которая предназначена для, сотрудников, работающих с ассортиментом товара

- В выбранной базе раскройте узел Securite (безопасность). Сделайте щелчок правой кнопкой мыши по узлу Schemas (Схемы). Появится меню.
- Выберите в меню New Schemas. Появится окно, содержащее три страницы

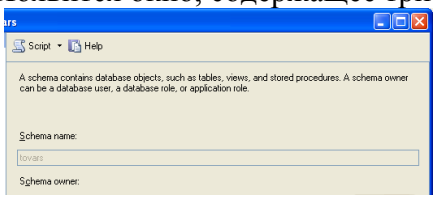


Рис 22 Окно Schema New вкладка General

Поле Schema Owner оставьте пустым.

По умолчанию все таблицы при создании, если не указана схема, помещаются в схему dbo. Поэтому перед именем всех таблиц, созданных ранее, было указано dbo.имя_таблицы (dbo.tovar, dbo.post, dbo.postavka, dbo.rashod). для того, чтобы переместить таблицу из одной схемы в другую нужно записать

use имя_базы

go

alter schema *имя_схемы_куда* **transfe**r *имя_схемы_откуда.имяТаблицы*

Переместим таблицы tovar, post из схемы **dbo** в схему '**tovars**'
use sklad
go
alter schema tovars transfer dbo.post

use sklad
go
alter schema tovars transfer dbo.tovar

В обозревателе объектов перед именем таблицы появится имя схемы

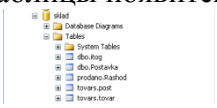


Рис 23 Список таблиц в обозревателе объектов

Задание 5

Создадим схему, в которую поместим таблицу Rashod. Схему создадим аналогично заданию 4. Имя схемы: 'prodano'. Переместим таблицу из схемы **dbo** в схему 'prodano'
use sklad
go
alter schema prodano transfer dbo.rashod

Задание 6

Создадим пользователя с именем user_teddy базы данных sklad для учётной записи teddy с помощью **SQL Server Management Studio** . Этот пользователь может редактировать, добавлять, изменять, удалять записи в таблицах (post, tovar) из схемы tovars. В схеме prodano пользователь может только читать данные из таблицы rashod .

6.1 В базе данных sklad раскройте папку Security, перейдите к Users. Из контекстного меню Users выберите пункт «New User». Откроется окно создания нового пользователя базы данных sklad.

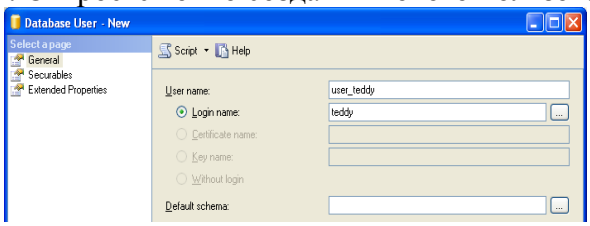


Рис 24 Окно создания нового пользователя базы данных sklad страница General

Пользователь user_teddy появится в списке Users.

Определим возможности работы пользователя user_teddy в базе данных sklad.

A) В схеме prodano. Выберите sklad ->Securite-> Schemas. Из списка схем выберите prodano . Из контекстного меню выберите Properties. В окне свойств выберите вкладку Permissions.

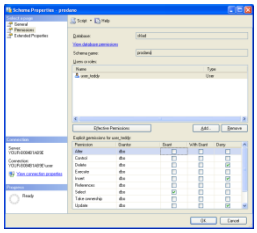


Рис 25 Свойства схемы prodano.

Нажмите на кнопку Add. Появится окно Select Users or Roles. В этом окне нажмите кнопку Browse. Далее выведется окно для выбора пользователя. Выберите user_teddy .

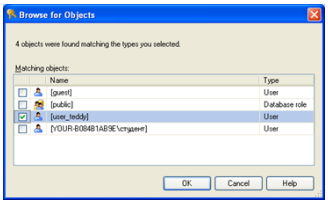


Рис 26 Выбор пользователя для схемы.

Нажмите Ок. В окне свойства схемы в разделе Users or Roles появится пользователь user_teddy . Далее необходимо установить его полномочия в указанной схеме. Для этого перейдём в раздел *Explict permissions for user_teddy*. Управление разрешениями пользователя в схеме данных выполняется через ключевые понятия
GRANT (разрешить), **DENY** (запретить).
Пользователь в этой схеме может только выбирать данные (поэтому ставим галочку около Select в столбце **GRANT** (разрешить)). Обновление (update), удаление (delete), добавление (insert) запрещено, поэтому в столбце **DENY** (запретить) ставим галочку около этих операций. Нажмите Ок и полномочия пользователя в схеме prodano будут заданы.

Б) В схеме tovars. Выберите sklad ->Securite-> Schemas. Из списка схем выберите tovars . Из контекстного меню выберите Properties. В окне свойств выберите вкладку Permissions. В этой схеме пользователь может выбирать, обновлять, удалять, добавлять записи в таблицы (post, tovar). Добавьте пользователя user_teddy по аналогии с предыдущей схемой и установите его полномочия в схеме:

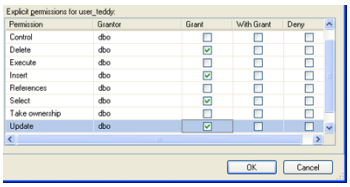


Рис 27 полномочия пользователя user_teddy в схеме tovars

Нажмите Ок и полномочия пользователя в схеме tovars будут заданы.

4. Выполнение запросов к таблицам, находящимся в разных схемах.

Если таблицы принадлежат к разным схемам, то при указании имени таблицы нужно указать имя схемы.
Задание 7
Создадим запрос на выборку данных из таблицы Tovar (схема tovars) и rashod (схема prodano)

```
use sklad
select postavka.data, postavka.kol, tovars.tovar.zena
from postavka, tovars.tovar
where postavka.kod_tov= tovars.tovar.kod_tov
```

Если необходимо просмотреть к каким схемам относятся таблицы базы данных, то необходимо выполнить запрос

```
use sklad
go
use sklad
select * from information_schema.tables
```

Результат запроса:

sklad	*prodano*	Rashod	BASE TABLE
sklad	dbo	sysdiagrams	BASE TABLE
sklad	**tovars**	post	BASE TABLE
sklad	dbo	itog	BASE TABLE
sklad	**tovars**	tovar	BASE TABLE

sklad	dbo	Postavka	BASE TABLE

Контрольные задания.

1. Создать учётные записи входа в SQL Server student, prepodavatel, otdel_kadrov.
2. В базе данных **Учебный процесс** создать пользователей базы: *студент, преподаватель, отдел кадров*. Связать учётные записи с пользователями базы Создать схемы для пользователей. Сопоставьте каждой схеме таблицу.(как в заданиях 4 и 5).
3. В **индивидуальной базе** данных создать список предполагаемых пользователей. Для каждого пользователя задать действия, которые он должен выполнять в базе данных.
4. Для нескольких пользователей из предыдущего задания выполнить: учётные записи входа в SQL Server; записи пользователей базы; схемы данных; сопоставить записи пользователей и схемы данных.

Лабораторная работа №37

«Создание простого клиента при помощи MS Access»

Цель: «изучить принципы подключения к SQL – совместимым базам данных и организации обмена данными между приложениями»

Ход Работы

Одним из способов, с помощью которых различные приложения могут подключиться базам данных SQL - сервера, является интерфейс Open Database Connectivity (открытый интерфейс подключения к базам данных). ODBC обеспечивает набор функций программного интерфейса приложений (API), которые упрощают подключение к базам данных самых различных форматов.

Доступ к базам данных в этом случае осуществляется с помощью драйверов ODBC, библиотек DLL, в которых содержатся функции для обеспечения таких возможностей. Драйверы ODBC устанавливаются в системе одновременно с установкой в ней утилит SQL - сервера.

Утилита Источники данных ODBC (Open Database Connectivity) позволяет настроить соединение ODBC для получения доступа объектам баз данных. Для запуска утилиты выберите соответствующий значок в Панели управления Windows. Диалоговое окно утилиты предлагает доступ к настройке трех типов источников данных (DSN - Data Source Name, имя источника данных).

В MS Access, начиная с версии XP, появилась специализированная возможность настроить подключение к MS SQL Server по **OLE DB** - ***Access Project***. Как ей воспользоваться:

1) закрыть текущую базу данных (если она открыта) и в меню File выбрать New. Затем в списке New File (справа) выбрать Project (Проект с имеющимися данными) и выбрать место для сохранения файла с расширением ADB.

Рис 28 Поля в окне «Файл новой базы данных»

2. в стандартном окне настроить параметры подключения по OLE DB.

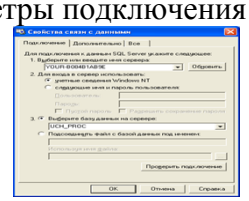


Рис 29 Параметры подключения

Все таблицы в базе данных SQL Server будут помещены в контейнер ‘Таблицы’ все представления и хранимые процедуры - в контейнер ‘Запросы’.

При работе через проект возможностей гораздо больше - можно создавать таблицы, представления, хранимые процедуры, функции на сервере, изменять их и т.п.

Контрольные задания

1. Создать формы для ввода данных во все таблицы базы данных «Учебный процесс».
2. Выполните запуск процедур, расположенных на сервере.
3. Создайте соединение с базой данных sklad.
4. Создайте формы ввода данных для таблиц.
5. Создайте отчёты для базы данных sklad.
6. Создайте соединение с индивидуальной базой данных.
7. Создайте формы ввода данных для таблиц индивидуальной базы данных.
8. Спроектируйте главную кнопочную форму для индивидуальной базы данных.

Лабораторная работа №38

Тема: **Операторы языка SQL**

Цели:

- научиться проектировать базу данных
- ознакомиться с понятиями языка запросов
- научиться создавать таблицу в MS SQL Server
- ознакомиться с операторами языка SQL

В результате выполнения практического занятия студент должен:

Знать:

- основные понятия БД
- принципы проектирования базы данных
- принципы построения таблиц в MS SQL Server
- основные операторы языка SQL

Уметь:

- проектировать базу данных
- производить корректировку структур таблиц БД
- строить основные операторы языка запросов

Пояснения к работе:

Все упражнения относятся к БД «Договор». В БД хранятся спецификации к договорам и номенклатурный справочник (NOMENCLATURE). Спецификация договора состоит из паспортной части (TITLE_CONTRACT) и номенклатурного списка (SPC_CONTRACT). Логическая схема базы данных «Договор» приведена на рисунке 25.

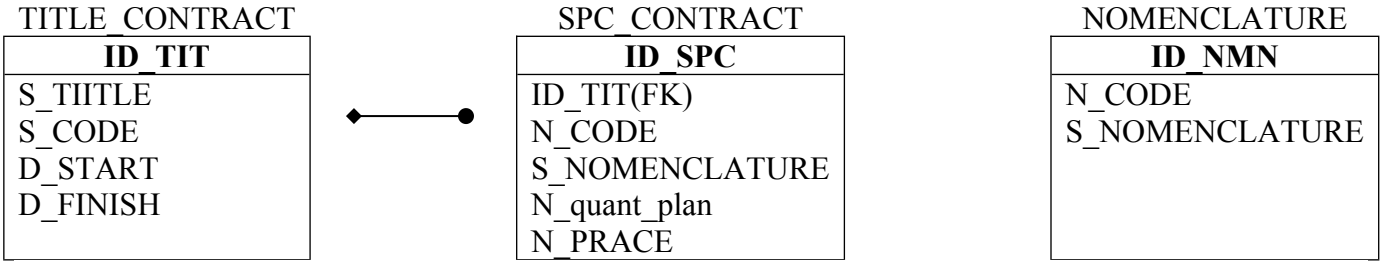


Рис. 25. Логическая схема БД

1. Написать оператор Select формирующий отчет обо всех договорах и их спецификациях. Шапка отчета должна включать: *название договора, название номенклатуры* и *количество.* Для каждого конкретного договора если в его номенклатурном списке встречается несколько строк с одной и той же номенклатурой, то для этой номенклатуры в отчет включается одна строка содержащая суммарное количество.
2. Написать оператор select формирующий отчет состоящий из договоров, не имеющих спецификации.
3. Написать оператор Select формирующий отчет состоящий из договоров, имеющих спецификации.
4. Написать оператор select формирующий отчет состоящий из договоров, заключенных до 10.10.2004.
5. Написать оператор Select формирующий отчет с суммарной стоимостью договоров.
6. Написать оператор select формирующий отчет, содержащий договора с максимальной стоимостью.
7. Написать оператор select формирующий отчет, содержащий договора с минимальной стоимостью.
8. Написать оператор select формирующий отчет о средней стоимости договоров.
9. Написать оператор Select формирующий отчет, содержащий договора с максимальной и минимальной стоимостью.

10. Написать оператор select формирующий отчет с номенклатурой, входящей в таблицу SPC_CONTRACT и не входящей в таблицу NOMENCLATURE .

11. Написать оператор Select формирующий отчет с номенклатурой, входящей в таблицу NOMENCLATURE и не входящей в таблицу

SPC_CONTRACT.

12. Написать оператор select формирующий отчет со всеми договорами, даты начала которых совпадают; количество таких договоров строго более 1.

13. Написать оператор Select, случайно выбирающий строку из таблицы NOMENCLATURE.

14. Написать оператор Select, случайно выдающий пронумерованный список названий номенклатуры из таблицы NOMENCLATURE.

Оператор UPDATE

1. В таблице Table одним оператором UPDATE поменяйте содержимое перечисленных ниже столбцов

Table

A	-A и B
B	-A и B; C и D
C	-A и B; B и C
D	-A и B; B и C; C и D, D и C

В следующих задачах используется БД, схема которой представлена на рис. 25.

2. Поменять все значения поля s_nomenciature, входящие в таблицу SPCJNOMENCLATURE с одноименными полями, имеющими тот же код, из таблицы NOMENCLATURE.

3. Поменять все значения поля s_nomenciature , входящие в таблицу SPC NOMENCLATURE с одноименными полями, имеющими тот же код, из таблицы NOMENCLATURE и наоборот.

4. Увеличьте цену каждой строки из спецификации договоров на десять процентов для договоров, имеющих четный код.

5. Уменьшите цену каждой строки из спецификации договоров на десять процентов для договоров, имеющих нечетный код.

6. Замените значения поля S_CODE в таблице спецификации договоров на поля из таблицы NOMENCLATURE — при условии, что одинаковые поля S_NOMENCLATURE в обеих таблицах.

7. Замените значения поля S_CODE в таблице спецификации договоров на поля из таблицы NOMENCLATURE — при условии, что одинаковые поля S_NOMENCLATURE в обеих таблицах.

8. Написать оператор UPDATE, случайно изменяющий поле N_PRICE в случайной строке из таблицы NOMENCLATURE .

***Оператор** DELETE*

1. Удалить договор с заданным номером.
2. Удалить с договором с минимальной и минимальной стоимостью.
3. Удалить с договором со стоимостью большей, чем средняя стоимость.
4. Удалить с договором со стоимостью меньшей, чем средняя стоимость.
5. Удалить строки из спецификации, коды которой не входят в таблицу NOMENCLATURE.
6. Удалить строки из NOMENCLATURE, коды которые не входят в таблицу спецификаций.
7. Удалить с договором без спецификации.
8. Удалить с договором со спецификациями.
9. Удалить с договором с четными кодами.
10. Удалить с договором с датами начала между годами 2002-2003 включительно.

11. Написать оператор DELETE , случайно удаляющий строку из таблицы NOMENCLATURE.

Оператор INSERT

1. Добавьте договор без спецификации.
2. Добавьте договор без спецификации с номером большим на единицу, чем максимальный номер уже существующих договоров.
3. Продублируйте строки в таблице NOMENCLATURE, используя только один оператор INSERT.
4. Добавьте строку к спецификации договора.

Упражнения по теме операторы языка SQL

81

5. Продублируйте строки в таблице NOMENCLATURE, используя только одни оператор INSERT.
6. Добавьте названия номенклатур из спецификаций договоров (SPC_CONTRACT), коды которых не входят в таблицу NOMENCLATURE в каждый договор.
7. Написать оператор INSERT, случайно выбирающий сроку из NOMENCLATURE и записывающий их в SPC_CONTRACT для данного контракта.

ТРИГГЕР

1. Разработать триггер, который выдавал бы на печать свой тип и момент своего вызова.
2. Разработать триггер, который в случае удаления договора удалял бы его спецификацию.
3. Разработать триггер, который в случае удаления строки из номенклатуры удалял бы из спецификаций строки с этой же номенклатурой.
4. Разработать триггер, который в случае изменения названия номенклатуры в таблицы NOMENCLATURE изменял бы соответствующее название в таблице SPC_CONTRACT.

Задачи повышенной сложности

Пусть бинарное отношение задано таблицей, состоящей из двух столбцов. Написать сохраненные процедуры, проверяющие, обладает ли отношение следующими свойствами:

1. Транзитивность бинарного отношения.
2. Симметричность бинарного отношения.
3. Функциональность бинарного отношения.
4. Рефлективность бинарного отношения.
5. Отношение порядка.
6. Отношение полного порядка.
7. Отношение частичного порядка.
8. Сериальность бинарного отношения.
9. Является ли граф связанным.
10. Является ли граф с циклами.
11. Является ли граф деревом.
12. По двух вершинам в графе найти их общий предок(если таковой есть).
13. Проверять для двух графов являются ли они изоморфными.

Контрольные вопросы:

1. Оператор FETCH .
2. Назначение процедуры. Способы задания, удаления и корректировки.
3. Представление (View) назначение. Способы задания, удаления и корректировки.
4. Назначение глобальных переменных; привести примеры.
5. Диалекты SQL — краткая характеристика.
6. Какие коллизии возникают при параллельных вычислениях и способы их преодоления.
7. Назначение блокировки и типы блокировок.
8. Дать определение уровню изоляции 0. Каких коллизий он позволяет избежать?
9. Дать определение уровню изоляции 1. Каких коллизий он позволяет избежать?
10. Дать определение уровню изоляции 2. Каких коллизий он позволяет избежать?

Дать определение уровню изоляции 3. Каких коллизий он позволяет избежать?

Лабораторная работа №39

Тема: **Создание и корректировка структуры таблиц базы данных**

Цели:

- научиться проектировать базу данных
- ознакомиться с понятиями языка запросов
- научиться создавать таблицу в MS SQL Server

В результате выполнения практического занятия студент должен:

Знать:

- основные понятия БД
- принципы проектирования базы данных
- принципы построения таблиц в MS SQL Server

Уметь:

- проектировать базу данных
- производить корректировку структур таблиц БД

Пояснения к работе:

1.Создайте таблицу **ANKETA.** Описание полей приведено в следующей таблице.

Поле	Тип информации	Семантические условия
N	Целое число	Уникальное
Фамилия	Текст длиной 1-40	Непустое
Имя	Текст длиной 1-40	Не пустое
Отчество	Текст длиной 0-40	Текст длиной 0-40
Дата рождения	Дата	Дата более 01.01.1900
Табельный номер	Целое число длиной 8	Уникальное
Фото	Графическая информация	Графическая информация

2.Добавьте поле первичного ключа, который должен быть типа automcrement.

3. Удалите таблицу.

1. Создайте таблицу **GRANT** с паспортными данными гранта. Описание полей приведено в следующей таблице.

Поле	Тип информации	Семантические условия
N	Целое число	Уникальное
Код гранта	Текст длиной 1-40	Непустое
Название гранта	Текст длиной 1-40	Не пустое
Дата начала	Дата	Дата более 01.01.2003
Дата окончания	Дата	Дата более 01.01.2004
Примечание	Текст	E

2. Добавьте поле первичного ключа, который должен быть типа automcrement.

3. Удалите таблицу.

1. Создайте таблицу, предназначенную для хранения информации посетителей сайта. Примите решения о типах данных, обоснуйте принятые решения.

-Имя

-Фамилия

- Год рождения
- Электронный адрес.
- Область интересов

Напишите SQL-операторы создания таблиц базы данных.

2. Добавьте поле первичного ключа ID, который должен быть типа autoincrement.

3. Удалите таблицу.

1. Создайте таблицу «Договор поля». Ограничения, наложенные на эти поля, представлены в следующей таблице.

Поле	Тип информации	Семантические условия
N	Целое число	Уникальное
Код договора	Текст длиной 1-40	Непустое
Название договора	Текст длиной 1-40	Непустое
Дата начала	Дата	Дата более 01.01.2003
Дата окончания	Дата	Дата более 01 .0 1 .2004
Примечание	Текст	E

2. Добавьте поле первичного ключа, который должен быть типа autoincrement.

3. Удалите таблицу.

1. Создайте таблицу для классификатора подразделений предприятия. Каждое подразделение подчиняется вышестоящему подразделению. Количество уровней подчиненности не ограничено.

Поле	Тип информации	Семантические условия
N	Целое число	Уникальное
Код подразделения	Текст длиной 1-40	Не пустое
Название подразделения	Текст длиной 1-40	Непустое
Телефон	9-значное число	
Примечание	Текст	

2. Добавьте поле первичного ключа, который должен быть типа autoin-crement.

3. Удалите таблицу.

1. Создайте таблицу для классификатора подразделений предприятия. Каждое подразделение подчиняется вышестоящему подразделению. Количество уровней подчиненности не ограничено.

Поле	Тип информации	Семантические условия
N	Целое число	Уникальное
Код подразделения	Текст длиной 1-40	Непустое
Название подразделения	Текст длиной 1-40	Непустое
Телефон	9-значное число	
Примечание	Текст	

2. Добавьте поле первичного ключа, который должен быть типа autoincrement.

3. Удалите таблицу.

Контрольные вопросы:

1. Оператор select с простым условием отбора.
2. Подчиненный select.
3. Группировка и агрегатные функции.
4. Назначение триггеров. Способ создания, удаления и корректировки. Типы событий реагирования.
5. Две формы оператора **insert.**
6. Оператор UPDATE .

7. Оператор delete.
8. Определение и назначение транзакции. Способы задания, удаления и корректировки.
9. Объявление курсора.

Лабораторная работа №40

Тема: **Проектирование структуры базы данных, нормализация таблиц**

Цели:

- познакомиться с основными понятиями базы данных
- научиться проектировать структуру базы данных
- научиться производить нормализацию отношений

В результате выполнения практического занятия студент должен:

Знать:

- основные понятия БД
- принципы проектирования структуры базы данных
- этапы проектирования
- принципы нормализации отношений
- виды нормальных форм

Уметь:

- проектировать структуру базы данных
- производить нормализацию отношений

Пояснения к работе:

В соответствии с условиями заданий (по вариантам) требуется:

- сформировать структуру таблиц баз данных
- подобрать подходящие имена таблицам баз данных
- установить взаимосвязи между таблицами
- обеспечить требования нормализации таблиц баз данных

Первым этапом проектирования БД любого типа является анализ предметной области, который заканчивается построением информационной структуры (концептуальной схемы). На данном этапе анализируются запросы пользователей, выбираются информационные объекты и их характеристики и на основе проведенного анализа структурируется предметная область. Анализ предметной области является общезначимым этапом, не зависящим от программной и технической сред, в которых будет реализовываться БД.

Анализ предметной области целесообразно разбить на три фазы: анализ концептуальных требований и информационных потребностей; выявление информационных объектов и связей между ними; построение концептуальной модели предметной области и проектирование концептуальной схемы БД.

На этапе анализа концептуальных требований и информационных потребностей необходимо решить следующие задачи:

- анализ требований пользователей к БД (концептуальных требований);
- выявление имеющихся задач по обработке информации, которая должна быть представлена в БД (анализ приложений); выявление перспективных задач (перспективных приложений);
- документирование результатов анализа.

Требования пользователей к разрабатываемой БД представляют собой список запросов с указанием их интенсивности и объемов данных. Эти сведения разработчики БД получают в диалоге с ее будущими пользователями. Здесь же выясняются требования к вводу, обновлению и корректировке информации. Требования пользователей уточняются и дополняются при анализе имеющихся и перспективных приложений.

Варианты заданий:

Спроектировать структуру базы данных и провести нормализацию отношений для предметной области:

1 вариант: Кинотеатры.

Возможные атрибуты: название, адрес, телефон, категория, вместимость, число залов, кинотеатр, кинофильм, время, дата, режиссер, год выпуска, страна, число серий, тематика, краткое содержание.

2 вариант: Аптека

Возможные атрибуты: номер, дата, врач, поликлиника, лекарство, количество, режим приема, стоимость, особые замечания, шифр, название, группа, краткая рекомендация по применению, срок хранения рецепта, дата поступления, цена, единица измерения, количество, срок годности.

3 вариант: Физкультура

Возможные атрибуты: зачетка, фамилия, группа, дата рождения, преподаватель, специализация, медицинская группа, разряд, вид, особенности, норматив, результат, дата, оценка, семестр, пол.

4 вариант: Домоуправление

Возможные атрибуты: адрес, квартира, фамилия, площадь, число комнат, номер ордера, дата получения, месячная оплата, долг, вид услуги, стоимость, дата введения, дата оплаты.

5 вариант: Почта

Возможные атрибуты: шифр, название, тип учреждения, цена, число экземпляров в год, адрес, фамилия, профессия, возраст, дата начала, длительность, сумма.

6 вариант: Расписание занятий

Возможные атрибуты: номер группы, специальность, факультет, число студентов, староста, номер аудитории, вместимость, тип, шифр дисциплины, название, ФИО преподавателя, звание, должность, кафедра, время, день, неделя.

Контрольные вопросы:

1.Перечислите основные типы моделей баз данных. Дайте им краткие сравнительные характеристики и опишите области их применения.

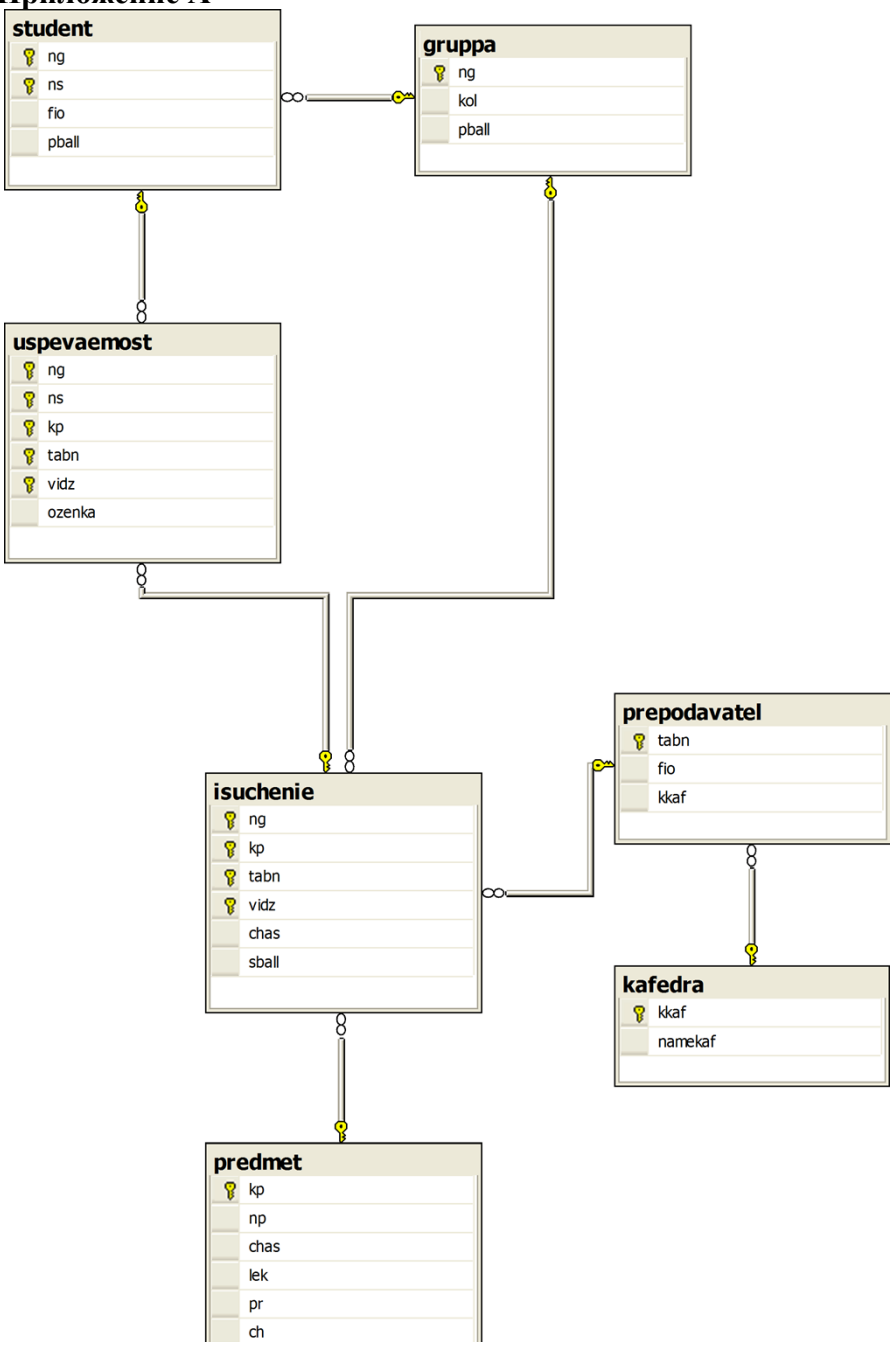
2.Кратко охарактеризуйте СУБД ведущих фирм.

3.Перечислите основные отличия реляционной модели данных от иерархической модели данных.

4.Перечислите и дайте определение основным операторам доступа к данным для иерархической модели данных.

5.Перечислите и охарактеризуйте типы отношений между таблицами в реляционной модели данных.

Приложение А



Приложение В
отдел № _ Запрос № ( ) баллов

Тип запроса_____

Структура запроса на SQL_____
_____
_____
_____

Отметка о выполнении (дата)_____
Проверил _____
Замечания _____

Ошибки, выявленные при отладке

№	Описание	Тип

Литература

1. Голицына О.Л, Максимов Н. В. Базы данных. - М: «Форум», 2010. – 264с.
2. Диго С.М. Базы данных: проектирование и использование. - М: Финансы и статистика, 2008. – 214с.
3. Марков А.С., Лисовский К.Ю. Базы данных: Введение в теорию и методологию. М: Финансы и статистика, 2009. – 324с.
4. Селко Д. SQL для профессионалов. М: Лори, 2009. – 562с.
5. Ульман Дж. Основы реляционных баз данных. М: Лори, 2010. – 568с.
6. Шумаков П. В «Создание приложений баз данных». М.: Логос, 2008. -846с.