**Практическая работа 48**
**Создание базы данных с помощью команд SQL.**
**Редактирование данных средствами языка SQL**

**Цель работы:** Получить практический опыт создания таблиц базы данных с помощью команд SQL.

**Перечень оборудования и программного обеспечения**

Персональный компьютер
Microsoft Office (Word, Visio, Access)

**Краткие теоретические сведения**

Новые элементы базы данных создаются с помощью предложения SQL CREATE. Чтобы создать таблицу, используйте команду CREATE TABLE, за которой введите поля и типы данных, предназначенные для добавления в таблицу. В качестве разделителей используйте запятые, а весь список заключите в круглые скобки.

Это одна из команд языка определения данных DDL. Команды DDL являются подмножеством команд SQL и используются для создания, изменения и удаления структур базы данных.

**Синтаксис:**

| | | |
|---|---|---|
| **Создание таблицы** | **CREATE TABLE** ***имя_таблицы*** | |
| **Определение столбцов** | **(ПОЛЕ тип данных,** | **[DEFAULT значение NOT NULL]**<br>*Значение по Нет*<br>*Умолчанию отсутствующих* |
| **Первичный ключ** | **Primarykey (поле, …),** | |
| **Определение ограничения** | **Constraint имя ограничения** | |
| **Определение внешнего ключа** | **Foreign key (поле, …) references имя таблицы(поле, …));** | **On delete Cascade**<br>**Set null**<br>**No action**<br>**Set default**<br>**On update Cascade**<br>**Set null**<br>**No action**<br>**Set default** |

Используем следующие правила:

– имя таблицы указывается после ключевого слова CREATE TABLE (если имя состоит из нескольких слов, то его следует заключить в одинарные кавычки);

– далее в круглых скобках следуют имена столбцов таблицы (полей), после которых указывается тип данных, которому будет принадлежать поле;

– не обязательно: затем указывается может ли поле содержать пустые значения (NULL— может быть пустым или NOT NULL — обязательно для заполнения);

– одно из полей назначается первичным ключом (Primary key);

– поля отделяются запятыми.

**SQL - Типы данных**

Тип данных определяет, какое значение может содержать столбец: целочисленные данные, символьные данные, денежные данные, данные даты и времени, двоичные строки и т. д.

**Типы данных SQL**

Каждый столбец в таблице базы данных должен иметь имя и тип данных.

Разработчик SQL должен решить, какой тип данных будет храниться внутри каждого столбца при создании таблицы. Тип данных является ориентиром для SQL, чтобы понять, какой тип данных ожидается внутри каждого столбца, а также определяет, как SQL будет взаимодействовать с сохраненными данными.

Типы данных SQL разделяются на три группы:

— строковые;

— с плавающей точкой (дробные числа);

— целые числа, дата и время.

Типы данных SQL строковые

| Типы данных SQL | Описание |
|---|---|
| CHAR(size) | Строки фиксированной длиной (могут содержать буквы, цифры и специальные символы). Фиксированный размер указан в скобках. Можно записать до 255 символов |
| VARCHAR(size) | Может хранить не более 255 символов. |
| TINYTEXT | Может хранить не более 255 символов. |
| TEXT | Может хранить не более 65 535 символов. |
| BLOB | Может хранить не более 65 535 символов. |
| MEDIUMTEXT | Может хранить не более 16 777 215 символов. |
| MEDIUMBLOB | Может хранить не более 16 777 215 символов. |

| | |
|---|---|
| LONGTEXT | Может хранить не более 4 294 967 295 символов. |
| LONGBLOB | Может хранить не более 4 294 967 295 символов. |
| ENUM(x,y,z,etc.) | Позволяет вводить список допустимых значений. Можно ввести до 65535 значений в SQL Тип данных ENUM список. Если при вставке значения не будет присутствовать в списке ENUM, то мы получим пустое значение.<br>Ввести возможные значения можно в таком формате: ENUM ( 'X', 'Y', 'Z') |
| SET | SQL Тип данных SET напоминает ENUM за исключением того, что SET может содержать до 64 значений. |

Типы данных SQL с плавающей точкой (дробные числа) и целые числа

| Типы данных SQL | Описание |
|---|---|
| TINYINT(size) | Может хранить числа от -128 до 127 |
| SMALLINT(size) | Диапазон от -32 768 до 32 767 |
| MEDIUMINT(size) | Диапазон от -8 388 608 до 8 388 607 |
| INT(size) | Диапазон от -2 147 483 648 до 2 147 483 647 |
| BIGINT(size) | Диапазон от -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807 |
| FLOAT(size,d) | Число с плавающей точкой небольшой точности. |
| DOUBLE(size,d) | Число с плавающей точкой двойной точности. |
| DECIMAL(size,d) | Дробное число, хранящееся в виде строки. |

Типы данных SQL — Дата и время

| Типы данных SQL | Описание |
|---|---|
| DATE() | Дата в формате ГГГГ-ММ-ДД |
| DATETIME() | Дата и время в формате ГГГГ-ММ-ДД ЧЧ:ММ:СС |
| TIMESTAMP() | Дата и время в формате timestamp. Однако при получении значения поля оно отображается не в формате timestamp, а в виде ГГГГ-ММ-ДД ЧЧ:ММ:СС |
| TIME() | Время в формате ЧЧ:ММ:СС |
| YEAR() | Год в двух значной или в четырехзначном формате. |

Так же значение поля, в том случае если это не запрещено, может быть не указано, для этой цели используется ключевое слово NULL.

Пример:

```
CREATE TABLE GROUPS
(id NUMBER CONSTRAINT groups_pk PRIMARY KEY,
```

```
num_gr CHAR(8) not null,
form CHAR(2) not null,
CONSTRAINT groups_uk UNIQUE (num_gr, form),
CHECK (form in('Д','З')));
```

Команда создает таблицу GROUPS. Столбцы
id (уникальный идентификатор),
num_gr (номер группы, например, 3091),
form (форма обучения, дневная (Д) или заочная (З)).

Пример:

```
CREATE TABLE STUDENTS
(id NUMBER CONSTRAINTstud_pk PRIMARY KEY,
surname CHAR(100)not null,
name CHAR(100)not null,
patron CHAR(100)not null,
gr_id number not null,
CONSTRAINT stud_gr_id_fk FOREIGN KEY (gr_id)
REFERENCESgroups( id) ON DELETE CASCADE);
```

Команда создает таблицу STUDENTS. Столбцы
id (уникальный идентификатор),
surname – фамилия,
name – имя,
patron – отчество студента,
gr_id – внешний ключ, ссылающийся на первичный ключ таблицы GROUPS.

Каждый студент из таблицы STUDENTS учится в какой-то группе из таблицы GROUPS. При удалении группы, удаляется и информация о всех студентах, в ней обучающихся (это только для примера, необходимости в этом нет, скорее наоборот, группу с обучающимися в ней студентами удалять не следует).

Отметим, что таблицы создаются пустыми, а данные в них вносятся с помощью команды Insert.

**Обновление таблиц: удаление и добавление полей**

Обновление таблиц выполняется при помощи ключевых слов sql **ALTER TABLE**. Обновляя таблицу можно:

- удалять поля **DROP COLUMN;**
- добавлять поля **ADD;**

Пример:

В таблицу teachers добавить поле phone для номеров телефонов
ALTER TABLE teachers ADD phone CHAR (20);

Пример:

Необходимо удалить поле phone из таблицы teachers
ALTER TABLE teachers DROP COLUMN phone

**Команда SELECT ... INTO**

Команда SELECT ... INTO позволяет создать новую таблицу на основе данных из других таблиц. Эта команда используется для архивирования данных, резервного копирования таблиц.

**Синтаксис:**

SELECT Поля INTO НоваяТаблица [INБазаДанных] FROM Таблицы;

- Поля - имена одного или нескольких полей, которые будут скопированы в новую таблицу.
- НоваяТаблица - Имя создаваемой таблицы
- БазаДанных - путь и имя внешней базы данных, в которой содержатся таблицы. Если таблицы находятся в текущей базе данных, то этот аргумент необязателен.
- Таблицы - имена таблиц, из которых выбираются записи.

Если имя новой таблицы совпадает с именем уже существующей, то будет сгенерирована ошибка.

Пример:
SELECT ID, Name, Email, Order INTO CopyOfOrders FROM Orders;

**Задания**

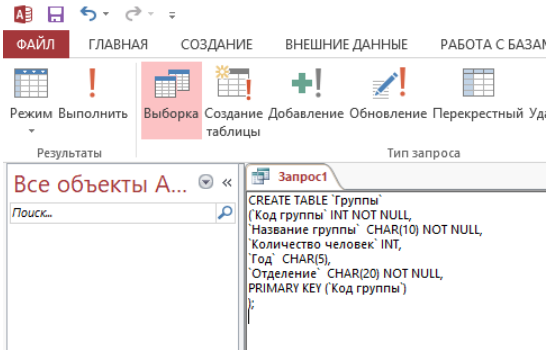
1 Изучить теоретические сведения.
2 В соответствии с вариантом задания создать таблицы базы данных с помощью команд SQL.
3 В соответствии с вариантом задания выполнить ввод данных в таблицы

**Порядок выполнения работы**

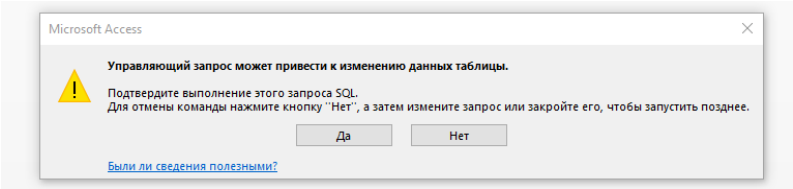
1 **Создадим новую таблицу базы данных**. Чтобы не писать команды, воспользуемся кодом, сгенерированным в DBDesigner
CREATE TABLE `Группы`

(`Код группы` SMALLINT NOT NULL,
`Название группы`  CHAR(10) NOT NULL,
`Количество человек` INT,
`Год`  CHAR(5),
`Отделение`  CHAR(20) NOT NULL,
PRIMARY KEY (`Код группы`)
);

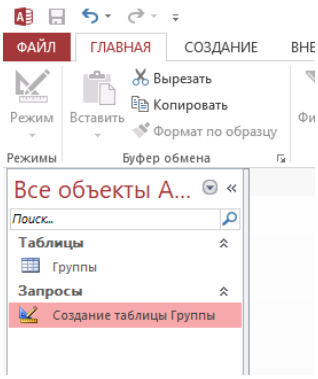
В меню Создание выбираем Конструктор запросов, закрываем окно добавления таблиц, переходим в режим SQL и вставляем команды.



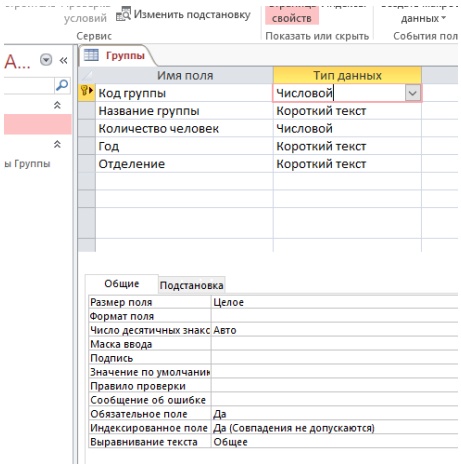

Сохраняем запрос и запускаем на выполнение, подтверждаем выполнение запроса




В результате будет создана таблица



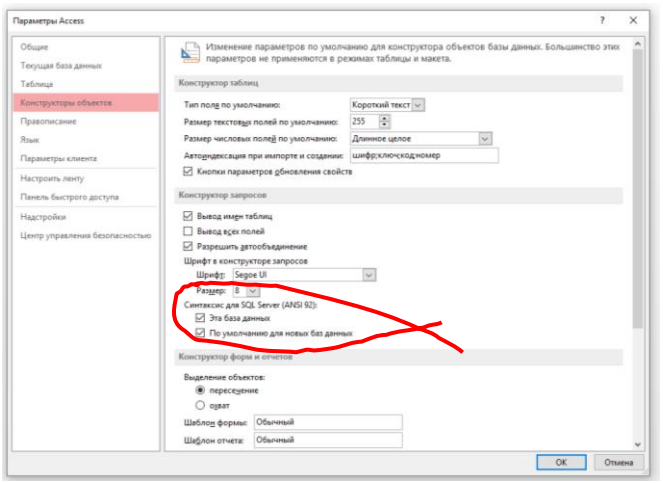

Если просмотрим полученную таблицу в режиме конструктора, то типы и характеристики полей соответствуют требованиям:



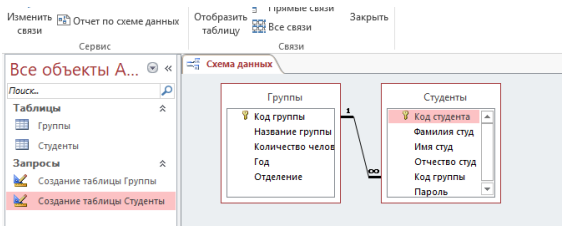

2 **Создадим таблицу с внешними ключами**, для этого добавим к списку полей команду FOREIGN KEY (`Код группы`) REFERENCES `Группы` (`Код группы`)

```
CREATE TABLE `Студенты1` (
`Код студента` SMALLINT NOT NULL,
`Фамилия студ`  CHAR(20) NOT NULL,
`Имя студ`  CHAR(15) NOT NULL,
`Отчество студ`  CHAR(20) NOT NULL,
`Код группы` SMALLINT NOT NULL,
`Пароль`  CHAR(15) NOT NULL,
PRIMARY KEY (`Код студента`),
FOREIGN KEY (`Код группы`) REFERENCES `Группы` (`Код группы`)
ON DELETE CASCADE
ON UPDATE CASCADE
);
```

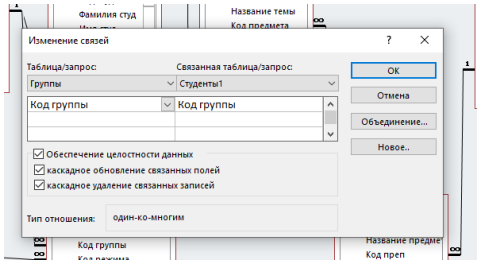
Но прежде, чем создавать такой запрос, необходимо внести изменения в параметры Access. Для этого в меню выбрать Файл→Параметры→Конструкторы объектов. В разделе Синтаксис для SQL Server выбрать Эта база данных и По умолчанию для новых баз данных



После создания таблицы Студенты в разделе Схема данных появятся связанные таблицы




Причем, если открыть окно связи (Правая кнопка мыши, изменить), то каскадное обновление и удаление записей активно):




**3 Создадим таблицу базы данных, используя имеющуюся**, причем, содержащую данные. Предположим, что часть данных из нее необходимо вынести в отдельную вспомогательную таблицу, тогда в основной таблицу эти данные должны быть заменены ссылкой на вспомогательную таблицу.

Определим действия, необходимые для получения результата:

а) скопировать поля из исходной таблицы во вспомогательную с помощью команды SELECT;

б) во вспомогательной таблице добавить ключевое поле код данных с помощью ALTER TABLE … ADD COLUMN, причем определить это поле как счетчик, тогда записи пронумеруются автоматически;

в) в случае необходимости, можно изменить тип первичного ключа на целый с помощью команды ALTER TABLE … ALTER COLUMN;

г) создать основную таблицу из исходной, но вместо полей, вынесенных во вспомогательную, перенести из вспомогательной ключевое поле, а чтобы в основную таблицу попали нужные значения, к команде добавить условия (SELECT …INTO …FROM …WHERE);

д) после того, как обе таблицы созданы, останется определить связи, т.е. добавить внешние ключи с помощью команды ALTER TABLE … ADD CONSTRAINT.

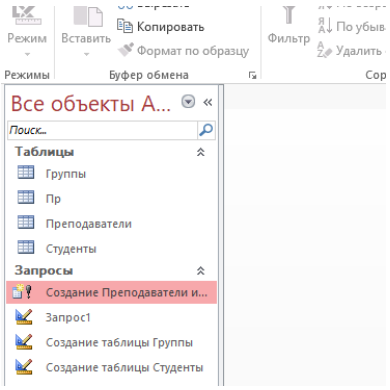
Предположим, имеется таблица Пр, хранящая данные о предметах и преподавателях.

```
CREATE TABLE `Пр`
(`Код предмета` SMALLINT NOT NULL,
`Название предмета`  CHAR(100) NOT NULL,
```

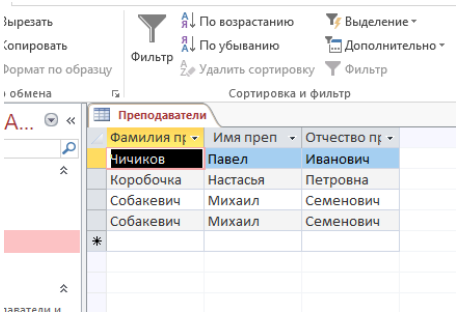
`Фамилия преп`  CHAR(20) NOT NULL,
`Имя преп`  CHAR(15) NOT NULL,
`Отчество преп`  CHAR(20) NOT NULL,
`Семестр` SMALLINT NOT NULL,
PRIMARY KEY (`Код предмета`)
);

Необходимо выделить информацию о преподавателях в отдельную таблицу. Для этого можно использовать команду SELECT ... INTO. Она скопирует указанные поля вместе с данными

SELECT [Фамилия преп], [Имя преп], [Отчество преп] INTO Преподаватели FROM Пр



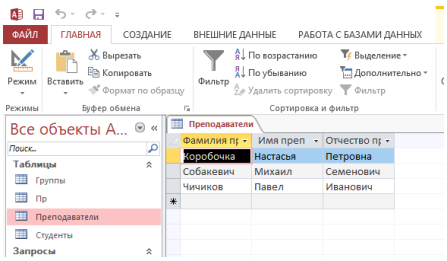

Откроем появившуюся таблицу:




Можно заметить, что одна строка повторяется дважды. Чтобы исключить повторение строк, добавим в команду слово DISTINCT:

SELECT DISTINCT [Фамилия преп], [Имя преп], [Отчество преп] INTO Преподаватели FROM Пр

Получим результат



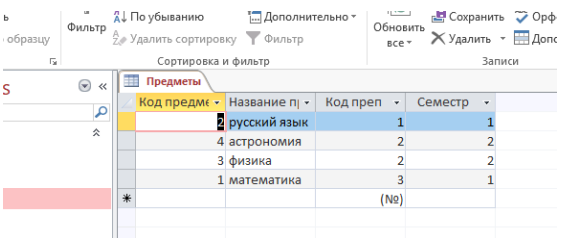

В режиме конструктора в таблице характеристики полей совпадают с Пр, но отсутствует первичный ключ и поле код преподавателя, необходимо внести изменения в таблицу Преподаватели. Добавим поле счетчик и назначим его первичным ключом:

ALTER TABLE Преподаватели ADD COLUMN [Код преп] COUNTER PRIMARY KEY;

**4 Осталось создать таблицу** Предметы. В ней ФИО преподавателя из Пр заменим кодом преподавателя из таблицы Преподаватели. Таким образом для создания таблицы Предметы **используем две таблицы** Пр и Преподаватели, кроме того в поле Код преп необходимо поместить соответствующий код преподавателя. Поэтому в команду создания таблицы добавим условие (иначе вместо 4 в таблице появится 12 строк):

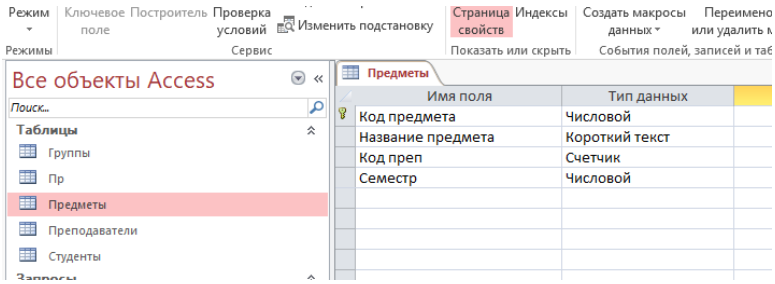
SELECT [Код предмета], [Название предмета], [Код преп], Семестр
INTO Предметы
FROM Пр, Преподаватели
WHERE Пр.[Фамилия преп] = Преподаватели.[Фамилия преп] AND Пр.[Имя преп] = Преподаватели.[Имя преп];

Результат:

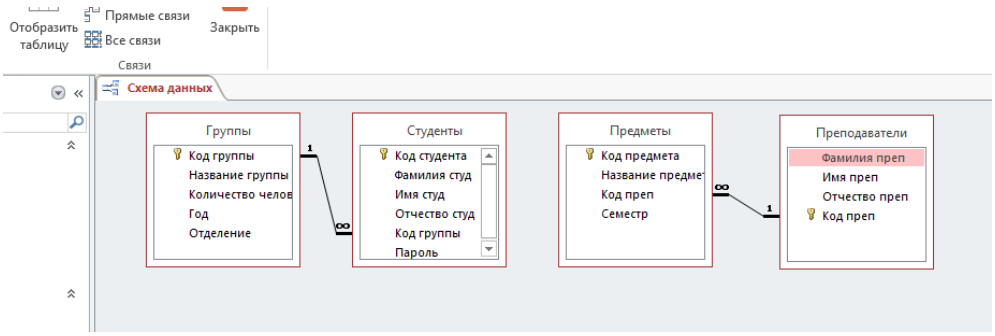



Аналогично предыдущей таблице, здесь отсутствуют ключи, как первичный, так и внешний. Внесем изменения в таблицу с помощью SQL запроса:

ALTER TABLE `Предметы` ADD CONSTRAINT PK_Predm PRIMARY KEY (`Код предмета’);




ALTER TABLE `Предметы` ADD CONSTRAINT FK_Predm_Prep FOREIGN KEY (`Код преп`) REFERENCES `Преподаватели`(`Код преп`) ON DELETE CASCADE ON UPDATE CASCADE;




Если необходимо изменить тип поля в таблице, используем команду:

ALTER TABLE `Предметы` ALTER COLUMN [Код преп] SMALLINT NOT NULL;

**Содержание отчета**

1 Название работы
2 Цель работы
3 Перечень технических средств обучения
4 Порядок выполнения работы
5 Ответы на контрольные вопросы
6 Вывод

**Варианты заданий**

Варианты заданий представлены в практической работе № 39

**Контрольные вопросы:**

1. Основные категории команд языка SQL
2. Для чего используется язык определения данных?

3. Основные команды языка DDL?
4. К каким объектам применяются команды языка DDL?
5. Какие действия выполняет команда CREATE?
6. К каким объектам применяются команда CREATE?
7. Какие действия выполняет команда ALTER?

**Используемая литература**


– Г.Н.Федорова Основы проектирования баз данных. М.: Академия, 2020

– Г.Н.Федорова Разработка, администрирование и защита баз данных. М.: Академия, 2018

– https://infopedia.su/16x10105.html

– https://studfile.net/preview/5917121/page:2

– https://codetown.ru/category/sql