ВАРИАТИВНОЕ ЗАДАНИЕ 4.1.
КУРСА
«БАЗЫ ДАННЫХ»

УПРАВЛЯЮЩИЕ ЗАПРОСЫ В ACCESS: СОЗДАНИЕ, ИЗМЕНЕНИЕ И УДАЛЕНИЕ ТАБЛИЦ

В данной лабораторной мы рассмотрим новый тип запросов Access – управляющие запросы. Они служат для создания, изменения и удаления таблиц или индексов.

***Задание №1**: 1. Создайте новую базу данных*

2. В окне конструктора запросов (не добавляя никаких таблиц) нажмите на кнопку «Управление».

3. В открывшемся окне в режиме SQL, введите запрос:

```
CREATE TABLE [Элемент]
(ElemID COUNTER PRIMARY KEY,
ElemName VARCHAR (50) NOT NULL);
```

4. Запустите код на выполнение (на панели конструктора нажмите на кнопку «Выполнить»)

(Сохраните запрос под именем 1_CREATE Элемент)

Выполнение данного запроса приведет к созданию таблицы «Элемент» с первичным ключом *ElemID* типа Счетчик и текстовым полем *ElemName* длиной до 50 символов.

В общем виде, формат оператора CREATE TABLE предполагает, что в круглых скобках идет перечень определений столбцов и ограничений. Причем ограничения уровня столбца записываются в определении столбца, а ограничения уровня таблицы идут после определения столбцов. В приведенном выше примере оба ограничения (PRIMARY KEY и NOT NULL записаны как ограничения уровня столбца).

Если создается составной первичный ключ, его можно описать только как ограничение уровня таблицы. Различие в приведенных ниже выражениях SQL в том, что в первом из приведенных примеров название (имя) ограничения *явно* указывается, во втором - назначается СУБД автоматически.

```
CREATE TABLE [Элемент1]
(ElemID1 INTEGER,
ElemID2 INTEGER,
ElemName VARCHAR (50) NOT NULL,
CONSTRAINT pk1 PRIMARY KEY (ElemID1, ElemID2));
```
(CONSTRAINT – ограничение, pk1 – имя ограничения)

или

```
CREATE TABLE [Элемент1]
(ElemID1 INTEGER,
ElemID2 INTEGER,
ElemName VARCHAR (50) NOT NULL,
PRIMARY KEY (ElemID1, ElemID2));
```

Задание №2*: 1.Выберите один из двух запросов и запустите на выполнение. Проанализируйте результат.*
(Сохраните запрос под именем 2_CREATE Элемент1)

Задание №3*: 1. Постройте запрос, создающий таблицу, ссылающуюся на другую.*

```
CREATE TABLE [Элементы]
(ElementsID COUNTER PRIMARY KEY,
ElemID INTEGER REFERENCES [Элемент] NOT NULL, Comment
MEMO);
```

(Сохраните запрос под именем 3_CREATE Элементы)

Примечание: ограничение внешнего ключа указывается с помощью конструкции *REFERENCES*, после чего следует указание таблицы, на потенциальный ключ которой мы ссылаемся.

Подготовка к заданию №4. Если Вы запустили запрос «3_CREATE Элементы» на выполнение и создали соответствующую таблицу, то удалите ее (правой кнопкой мыши кликните по таблице и удалите ее путем выбора соответствующей операции из консольного меню). Если выполнение не осуществлялось - переходите сразу к заданию №4.

***Задание №4**: Для запроса по созданию таблицы Элементы (ранее рассмотренный) модифицируйте код запроса, чтобы он описывал внешний ключ как именованное ограничение уровня таблицы (имя fk1). Откройте схему данных, добавьте на нее созданные таблицы. Проверьте, что на схеме отображается связь между таблицами типа «1 ко многим».*

```
CREATE TABLE [Элементы]
(ElementsID COUNTER PRIMARY KEY,
ElemID INTEGER NOT NULL,
Comment MEMO,
CONSTRAINT fk1 FOREIGN KEY  (ElemID) REFERENCES
[Элемент] (ElemID) );
```

(Сохраните запрос под именем 4_CREATE Элементы)

***Задание №5**: 1. Напишите три запроса, создающие структуру данных нашей библиотечной базы (см. лабораторную номер 1). Для текстовых данных используйте тип varchar(N), где N – максимально допустимое количество символов, но не более 255. Определите первичные и внешние ключи, NOT NULL, уникальности значения – UNIQUE.*

2. Добавьте созданные Вами таблицы на схему данных и просмотрите связи.

3. Сравните созданные Вами таблицы с таблицами из лабораторной №1.

(Сохраните запрос под именами: 5_CREATE Book; 6_CREATE BookStatus, 7_CREATE BookInLib)

Изменение в структуре уже созданной таблицы можно сделать с помощью оператора ALTER TABLE. Его формат

```
ALTER TABLE <имя таблицы> <действие>
```

где выражение, описывающее выполняемое действие, может принимать значения, перечисленные в табл.

Действия оператора ALTER TABLE

Выражение	Описание
`ADD COLUMN <поле> <тип данных> [NOT NULL] [CONSTRAINT <ограничение>]`	Добавление столбца и ограничений для него
`ADD CONSTRAINT <ограничение>`	Добавление ограничения
`ALTER COLUMN <поле> <тип данных>`	Изменение типа данных в столбце
`DROP COLUMN <поле>`	Удаление столбца
`DROP CONSTRAINT <имя_ограничения>`	Удаление ограничения

Нужно отметить, что в конструкциях `ADD COLUMN` и `ADD CONSTRAINT` надо приводить описание ограничения (подобно тому, как ранее это сделалось для `CREATE TABLE`), а в `DROP CONSTRAINT` надо указать только имя удаляемого ограничения.

Рассмотрим два примера.

В первом мы создаем новый столбец в таблице «Элемент1», название – info, тип – varchar(200).

```
ALTER TABLE [Элемент1]
ADD COLUMN info varchar(200);
```

Второй фрагмент кода создает в этой же таблице ограничение с названием un_info, требующее уникальности значения созданного поля.

```
ALTER TABLE [Элемент1]
ADD CONSTRAINT un_info UNIQUE(info);
```

***Задание №6**: 1. Создайте запросы, представленные выше.*

(Сохраните запросы под именами 8_ALTER и 9_add constr)

2. Напишите запрос, удаляющий созданное ограничение.

(Сохраните запрос под именем 10_drop constr)

***Задание №7**: 1.Для созданного Вами аналога таблицы BookStatus напишите запрос, создающий поле Comment типа varchar(200), значение этого поля должно быть обязательно определено.*

(Сохраните запрос под именем 11_ALTER add)

Конструкция `SELECT … INTO …` позволяет создать таблицу и сразу же внести в нее какие-то данные из другой таблицы. Пусть, например, нам нужен список авторов книг, размещенный в отдельной таблице. Откройте конструктор запросов, выберите таблицу Book и из нее столбец Author. Найдите на панели инструментов кнопку Создание таблицы. Задайте имя таблицы Auth. Теперь в режиме SQL можно увидеть такой текст запроса:

```
SELECT Book.Author INTO Auth FROM Book;
```

Задание №8: *1. Используя конструкцию SELECT … INTO … создайте таблицу NewBook, содержащую всю информацию из Book о книгах, изданных после 2000 года.*
(Сохраните запрос под именем 12_SELECT)

Удалить таблицу можно с помощью управляющего запроса, содержащего оператор DROP TABLE <имя таблицы>.

Задание №9: *1. Удалите только что созданную таблицу NewBook*
(Сохраните запрос под именем 13_DROP)

Задание №10: *1. Выведите в схеме данных все таблицы и связи между ними.*