Калимуллина Зимфира Камиловна

Уфимский колледж статистики,

информатики и вычислительной техники

город Уфа

Преподаватель специальных дисциплин

**Лекция «Проектирование и модификация таблиц командами SQL»**

**План – конспект урока**

**Цель:** Научиться создавать базу данных и таблицы средствами языка SQL.

 **Задачи:**

* ПК 2.2 Программировать в соответствии с требованиями технического задания.
* ОК1. Понимать сущность и социальную значимость своей будущей профессии, проявлять к ней устойчивый интерес.
* ОК 2 Организовывать собственную деятельность, определять методы и способы выполнения профессиональных задач, оценивать их эффективность и качество.
* ОК 5 Использовать информационно-коммуникационные технологии в профессиональной деятельности.

**Цели занятия:**

**обучающие:**

* Изучить основные команды языка SQL.
* Научиться применять полученные знания для проектирования и модификации таблиц.

**воспитательные:**

* создание условия, обеспечивающие воспитание интереса к изучаемому предмету, к будущей профессии.

**развивающие:**

* способствовать развитию логического мышления,
* способствовать развитию творческих способностей.

**Тип занятия:** урок изучения нового материала

**Методы обучения:** объяснительно-иллюстративный, исследовательский, практический.

**Оборудование и программное обеспечение:**

* презентация по теме «Проектирование и модификация таблиц командами SQL»;
* проектор и экран для демонстрации лекции;
* маркерная доска..

**Методическое обеспечение:**

* дидактический раздаточный материал.

**План урока:**

1. Организационный момент. (3 мин)
2. Повторение ранее изученного материала. (17 мин)
3. Изучение нового материала с закреплением знаний (Решение задач). (55 мин)
4. Подведение итогов урока. (2 мин)
5. Домашнее задание (3 мин)

**Ход занятия:**

1. **Организационный момент** (проверка присутствующих, проверка готовности к работе)

Здравствуйте. Сегодня мы проведем открытый урок на тему «Проектирование и модификация таблиц командами SQL». Какие задачи стоят перед нами на этом уроке:

* ПК 2.2 Научиться программировать в соответствии с требованиями технического задания.
* ОК1. Понимать сущность и социальную значимость своей будущей профессии, проявлять к ней устойчивый интерес.
* ОК 2 Организовывать собственную деятельность, определять методы и способы выполнения профессиональных задач, оценивать их эффективность и качество.
* ОК 5 Использовать информационно-коммуникационные технологии в профессиональной деятельности.

**Какие цели мы поставим?**

**обучающие:**

* Изучить основные команды языка SQL.
* Научиться применять полученные знания для проектирования и модификации таблиц.

**воспитательные:**

* создание условия, обеспечивающие воспитание интереса к изучаемому предмету, к будущей профессии.

**развивающие:**

* способствовать развитию логического мышления,
* способствовать развитию творческих способностей.
1. **Повторение ранее изученного материала**

Вы уже знаете, что язык SQL (Structured Query Language - структурированный язык запросов) в настоящее время является промышленным стандартом, который поддерживает любая реляционная СУБД. На лабораторных работах мы с вами работаем с MS SQL Server 2008, команды которой строятся на стандартном языке запросов. Но прежде чем мы начнём с ними знакомиться, давайте вспомним некоторые определения и понятия связанные с языком SQL.

**Опрос студентов:**

1. Расскажите об истории создания языка SQL.
В начале 70-х годов в компании IBM была разработана экспериментальная СУБД System R на основе языка SEQUEL (Structured English Qeury Language - структурированный английский язык запросов), который можно считать непосредственным предшественником SQL. Целью разработки было создание простого непроцедурного языка, которым мог воспользоваться любой пользователь, даже не имеющий навыков программирования. В 1981 году IBM объявила о своем первом, основанном на SQL программном продукте, SQL/DS. Чуть позже к ней присоединились Oracle и другие производители. Первый стандарт языка SQL был принят Американским национальным институтом стандартизации (ANSI) в 1987 (так называемый SQL level /уровень/ 1) и несколько уточнен в 1989 году (SQL level 2). Дальнейшее развитие языка поставщиками СУБД потребовало принятия в 1992, 1999, 2003, 2006 нового расширенного стандарта. В настоящее время действует стандарт, принятый в 2003 году ([SQL:2003](https://ru.wikipedia.org/wiki/SQL%3A2003)) с небольшими модификациями, внесёнными позже ([SQL:2008](https://ru.wikipedia.org/wiki/SQL%3A2008)).
2. Что включает в себя язык Transact-SQL?
СУБД Microsoft SQL Server использует свое расширение языка, которое называется Transact-SQL и включает в себя следующее: все возможности стандартного языка, управляющие конструкции, локальные переменные, дополнительные функции (математические, строковые и другие), поддержка аутентификации Windows.
3. Какие в SQL определены подмножества языка.
В SQL определены два подмножества языка:
**SQL-DDL** (Data Definition Language) - язык определения структур и ограничений целостности баз данных. Сюда относятся команды создания и удаления баз данных; создания, изменения и удаления таблиц; управления пользователями и т.д.
**SQL-DML** (Data Manipulation Language) - язык манипулирования данными: добавление, изменение, удаление и извлечение данных, управления транзакциями
4. Какие основные типы данных поддерживает MS SQL Server? (спросить 3 студентов)
* - Символьные типы данных - содержат буквы, цифры и специальные символы.
	1. **CHAR** или **CHAR(n)** -символьные строки фиксированной длины. Длина строки определяется параметром **n**. **CHAR** без параметра соответствует **CHAR(1)**. Для хранения таких данных всегда отводится **n** байт вне зависимости от реальной длины строки.
	2. **VARCHAR(n)** - символьная строка переменной длины. Для хранения данных этого типа отводится число байт, соответствующее реальной длине строки.
* Целые типы данных - поддерживают только целые числа (дробные части и десятичные точки не допускаются). Над этими типами разрешается выполнять арифметические операции и применять к ним агрегирующие функции (определение максимального, минимального, среднего и суммарного значения столбца реляционной таблицы).
	1. **INTEGER** или **INT**- целое, для хранения которого отводится, как правило, 4 байта. *(Замечание: число байт, отводимое для хранения того или иного числового типа данных зависит от используемой СУБД и аппаратной платформы, здесь приводятся наиболее "типичные" значения)* Интервал значений от - 2147483647 до + 2147483648
	2. **SMALLINT** - короткое целое (2 байта), интервал значений от - 32767 до +32768
* Вещественные типы данных - описывают числа с дробной частью.
	1. **FLOAT** и **SMALLFLOAT** - числа с плавающей точкой (для хранения отводится обычно 8 и 4 байта соответственно).
	2. **DECIMAL(p)** - тип данных аналогичный **FLOAT** с числом значащих цифр **p**.
	3. **DECIMAL(p,n)** - аналогично предыдущему, **p** - общее количество десятичных цифр, **n** - количество цифр после десятичной запятой.
* Денежные типы данных - описывают, естественно, денежные величины. Если ваша система такого типа данных не поддерживает, то используйте DECIMAL(p,n).
	1. **MONEY(p,n)** - все аналогично типу **DECIMAL(p,n)**. Вводится только потому, что некоторые СУБД предусматривают для него специальные методы форматирования.
* Дата и время - используются для хранения даты, времени и их комбинаций. Большинство СУБД умеет определять интервал между двумя датами, а также уменьшать или увеличивать дату на определенное количество времени.
	1. **DATE** - тип данных для хранения даты.
	2. **TIME** - тип данных для хранения времени.
	3. **INTERVAL** - тип данных для хранения верменного интервала.
	4. **DATETIME** - тип данных для хранения моментов времени (год + месяц + день + часы + минуты + секунды + доли секунд).
* Двоичные типы данных - позволяют хранить данные любого объема в двоичном коде (оцифрованные изображения, исполняемые файлы и т.д.). Определения этих типов наиболее сильно различаются от системы к системе, часто используются ключевые слова:
	1. **BINARY**
	2. **BYTE**
	3. **BLOB**
* Последовательные типы данных - используются для представления возрастающих числовых последовательностей.
	1. **SERIAL** - тип данных на основе **INTEGER**, позволяющий сформировать уникальное значение (например, для первичного ключа). При добавлении записи СУБД автоматически присваивает полю данного типа значение, получаемое из возрастающей последовательности целых чисел.

В заключение следует сказать, что для всех типов данных имеется общее значение **NULL** - "не определено". Это значение имеет каждый элемент столбца до тех пор, пока в него не будут введены данные. При создании таблицы можно явно указать СУБД могут ли элементы того или иного столбца иметь значения **NULL** (это не допустимо, например, для столбца, являющего первичным ключом).

1. Какой командой можно создать базу данных?
CREATE DATABASE <имя\_базы\_данных>

**Пример:** CREATE DATABASE UKSIVT

1. Какой командой можно создать базу данных?
DROP DATABASE <имя\_базы\_данных>

**Пример:** DROP DATABASE UKSIVT

Молодцы! Материал прошлой лекции мы повторили, приступим к изучению нового материала.

1. **Изучение нового материала**

Запишите, пожалуйста, тему нашего сегодняшнего урока**:** **«Проектирование и модификация таблиц командами SQL»**

Для того чтобы создать таблицу в нашей БД мы должны выполнить команду:

Запишем ее шаблон:

 **CREATE TABLE <имя\_таблицы>**

 **(<имя\_столбца> <тип\_столбца>**

 **[NOT NULL]**

 **[UNIQUE | PRIMARY KEY]**

 **[REFERENCES <имя\_мастер\_таблицы> [<имя\_столбца>]]**

 **, ...)**

Пользователь обязан указать имя таблицы и список столбцов. Для каждого столбца обязательно указываются его имя и тип, а также опционально могут быть указаны параметры:

* **NOT NULL** - в этом случае элементы столбца всегда должны иметь определенное значение (не NULL)
* один из взаимоисключающих параметров **UNIQUE** - значение каждого элемента столбца должно быть уникальным или **PRIMARY KEY** - столбец является первичным ключом.
* **REFERNECES <имя\_мастер\_таблицы> [<имя\_столбца>]** - эта конструкция определяет, что данный столбец является внешним ключом и указывает на ключ какой (мастер\_таблицы) он ссылается.

Контроль за выполнением указанных условий осуществляет СУБД.

*Пример: Давайте создадим базы данных* ***publications*** *и таблицы* ***authors , publishers, titles, titleautors, wwwsites, wwwsites*** *согласно концептуальной схеме, которая есть у вас на карточках****:***

Сначала создадим БД: CREATE DATABASE publications;

Затем будем создавать таблицы ориентируясь на шаблон заполнения и нашу концептуальную схему:

 CREATE TABLE authors (au\_id INT PRIMARY KEY,

 author VARCHAR(25) NOT NULL);

 CREATE TABLE publishers (pub\_id INT PRIMARY KEY,

 publisher VARCHAR(255) NOT NULL,url VARCHAR(255));

 CREATE TABLE titles (title\_id INT PRIMARY KEY,

 title VARCHAR(255) NOT NULL,

 yearpub INT,

 pub\_id INT REFERENCES publishers(pub\_id));

 CREATE TABLE titleautors (au\_id INT REFERENCES authors(au\_id),

 title\_id INT REFERENCES titles(title\_id));

 CREATE TABLE wwwsites (site\_id INT PRIMARY KEY,

 site VARCHAR(255) NOT NULL,

 url VARCHAR(255));

 CREATE TABLE wwwsiteauthors (au\_id INT REFERENCES authors(au\_id),

 site\_id INT REFERENCES wwwsites(site\_id));

Проверите, у всех получилось как на слайде? Почему у нас не 5, а 6 таблиц. Правильно, потому что при связи многие ко многим между таблицами Издательство и Сайты (т.к. у Издательства м.б. много сайтов и в разных сайтах могут быть данные об издательстве), вставляют дополнительную таблицу, чтобы связь была один ко многим и многие к одному.

**Удаление таблицы** производится с помощью команды:
**DROP TABLE <имя\_таблицы>**

**Рассмотрим команду модификации таблицы:** Посмотрите на слайд презентации

|  |  |
| --- | --- |
| Добавить столбцы | **ALTER TABLE <имя\_таблицы> ADD** **(<имя\_столбца> <тип\_столбца>** **[NOT NULL]** **[UNIQUE | PRIMARY KEY]** **[REFERENCES <имя\_мастер\_таблицы> [<имя\_столбца>]]** **,...)** |
| Удалить столбцы | **ALTER TABLE <имя\_таблицы> DROP (<имя\_столбца>,...)** |
| Модификация типа столбцов | **ALTER TABLE <имя\_таблицы> MODIFY** **(<имя\_столбца> <тип\_столбца>** **[NOT NULL]** **[UNIQUE | PRIMARY KEY]** **[REFERENCES <имя\_мастер\_таблицы> <имя\_столбца>]]** **,...)** |

Давайте попробуем добавить столбец «Кол-во страниц» в таблицу Книга – titles:

**ALTER TABLE** titles  **ADD (kol\_str INT NOT NULL);**

Теперь попробуем убрать созданный нами столбец:

**ALTER TABLE** titles  **DROP (kol\_str INT);**

После того как мы с вами научились работать со столбцами БД приступим к изучению операторов создания индексов.

**Шаблон создание индекса:**

**CREATE [UNIQUE] INDEX <имя\_индекса> ON <имя\_таблицы> (<имя\_столбца>,...)**

Эта команда создает индекс с заданным именем для таблицы <имя\_таблицы> по столбцам, входящим в список, указанный в скобках. Создание индексов значительно ускоряет работу с таблицами. В случае указания необязательного параметра **UNIQUE** СУБД будет проверять каждое значение индекса на уникальность.

Очень часто встает вопрос, какие поля необходимо индексировать. Обязательно надо строить индексы для первичных ключей, поскольку по их значениям осуществляется доступ к данным при операциях соединения двух и более таблиц. Также в ответе на этот вопрос поможет анализ наиболее частых запросов к базе данных. Например, для БД **publications** можно ожидать, что одним из наиболее частых запросов будет выборка всех публикаций данного автора. Для минимизации времени этого запроса необходимо построить индекс для таблицы **authors** по именам авторов:

CREATE INDEX au\_names ON authors (author);

Создание индексов для первичных ключей:

CREATE INDEX au\_index ON authors (au\_id);
CREATE INDEX title\_index ON titles (title\_id);
CREATE INDEX pub\_index ON publishers (pub\_id);
CREATE INDEX site\_index ON wwwsites (site\_id);

Первоначальное определение структуры индексов производится разработчиком на стадии создания прикладной системы. В дальнейшем она уточняется администратором системы по результатам анализа ее работы, учета наиболее часто выполняющихся запросов и т.д.

**Для удаление индекса используется команда:**

**DROP INDEX <имя\_индекса>**

**Кроме того,** по соображениям безопасности не каждому пользователю прикладной системы может быть разрешено получать информацию из какой-либо таблицы, а тем более изменять в ней данные. Для определения прав пользователей относительно объектов базы данных (таблицы, представления, индексы) в **SQL** определена пара команд GRANT и REVOKE. Синтаксис операции передачи прав на таблицу:

 **GRANT <тип\_права\_на\_таблицу>**

 **ON <имя\_таблицы> [<список\_столбцов>]**

 **TO <имя\_пользователя>**

Права пользователя на уровне таблицы определяются следующими ключевыми словами (как мы увидим чуть позже эти ключевые слова совпадают с командами выборки и изменения данных):

* SELECT - получение информации из таблицы
* UPDATE - изменение информации в таблице
* INSERT - добавление записей в таблицу
* DELETE - удаление записей из таблицы
* INDEX - индексирование таблицы
* ALTER - изменение схемы определения таблицы
* ALL - все права

Давайте в качестве примера зададим права доступа к нашей БД:

В поле <тип\_права\_на\_таблицу> может быть указано либо ключевое слово ALL или любая комбинация других ключевых слов. Например, предоставим все права на таблицу **publishers** пользователю **andy**:

 GRANT ALL ON publishers TO andy;

Пользователю **peter** предоставим права на извлечение и добавление записей на эту же таблицу:

 GRANT SELECT INSERT ON publishers TO peter;

В том случае, когда одинаковые права надо предоставить сразу всем пользователям, вместо выполнения команды GRANT для каждого из них, можно вместо имени пользователя указать ключевое слово PUBLIC:

 GRANT SELECT ON publishers TO PUBLIC;

Отмена прав осуществляется командой REVOKE:

 **REVOKE <тип\_права\_на\_таблицу>**

 **ON <имя\_таблицы> [<список\_столбцов>]**

 **FROM <имя\_пользователя>**

Все ключевые слова данной команды эквивалентны оператору GRANT.

Большинство систем поддерживают также команду GRANT для назначения привилегий на базу данных в целом. В этом случае формат команды:

 **GRANT <тип\_права\_на\_базу\_данных>**

 **ON <имя\_базы данных>**

 **TO <имя\_пользователя>**

Отмена прав на базу данных осуществляется командой:

 **REVOKE <тип\_права\_на\_базу\_данных> FROM <имя\_пользователя>**

**Закрепление знаний.**

Для закрепления знаний полученных на занятии, давайте сначала вспомним чему оно было посвящено? Мы изучили:

1. Создание, удаление и модификация таблиц с решением примеров
2. Операторы создание и удаление индекса.
3. Операторы управления правами доступа

Все выполняли с решением конкретных задач на нашей БД.

Давайте вспомним форматы команд.

 Завершите фразу:

•       Сегодня я узнал…

•       Я научился…

•       У меня получилось …

•       Было трудно…

**Самостоятельная работа:**

Создайте БД по своему усмотрению, содержащую не менее 2-х таблиц, индексы для первичных ключей, операторы для управления правами доступа.

1. **Домашнее задание**
* Операторы создание, удаление и модификация таблиц с решением примеров
* Операторы создание и удаление индекса.
* Операторы управления правами доступа