

Использование языка SQL-DDL для создания структур данных

Высшая Школа Цифровой Культуры

Университет ИТМО

[dc@itmo.ru](mailto:dc@itmo.ru)

Оглавление

[**Работа с таблицами (создание, изменение, удаление таблиц)** 3](#_Toc203349782)

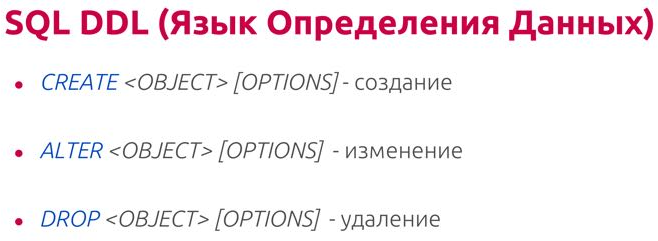
[**Ограничения целостности (ограничение неопределенных значений, допустимого диапазона значений, ограничение уникальности, ограничение первичного ключа и ссылочная целостность)** 8](#_Toc203349783)

## **Работа с таблицами (создание, изменение, удаление таблиц)**

Мы рассмотрели, как можно отобразить концептуальную модель базы данных в виде ER-диаграммы на реляционную, табличную модель. На следующем этапе нужно описать таблицы и ограничения целостности в контексте конкретной СУБД. Для этого в любой СУБД существует специальный язык, который дает возможность создавать объекты и работать с ними. Наиболее распространенный – структурированный язык запросов SQL. Именно он чаще всего используется для работы с реляционными базами данных и поддерживается большинством производителей СУБД. Поэтому мы будем рассматривать именно его.

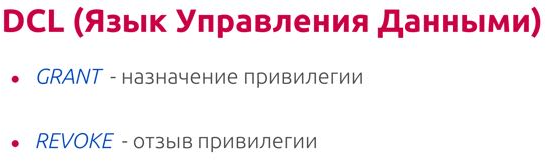
Все операторы языка можно поделить на три группы:

* операторы определения данных;
* операторы манипулирования данными;
* операторы управления данными.

Для описания структуры таблиц и других объектов нужно использовать операторы DDL – язык определения данных. Этих операторов всего три: операторы CREATE, ALTER и DROP – для создания, изменения и удаления объектов базы данных. Опции операторов существенным образом зависят от специфики создаваемого объекта, и от контекста используемой СУБД.

Язык манипулирования данными содержит операторы, позволяющие отображать и модифицировать содержимое таблиц: добавлять и изменять строки, удалять и искать их по заданным критериям.

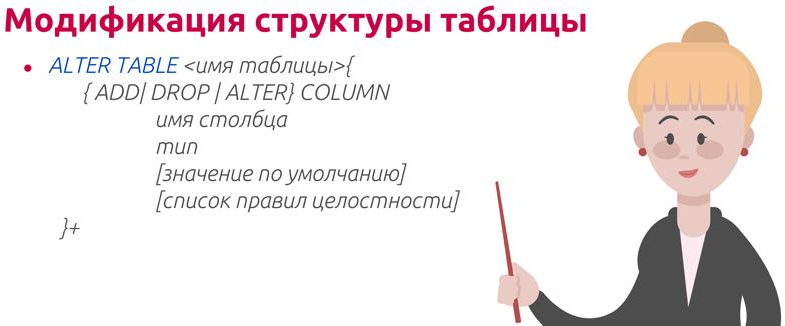
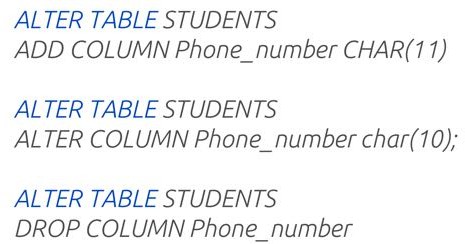


Язык контроля данных состоит из операторов, которые управляют правами пользователя в базе данных. Например, разрешить ли пользователю прочитать данные из таблицы, исполнить хранимую процедуру и т.п.

СУБД предоставляют среду для работы с базой данных, в которой можно писать и выполнять команды. Создание базы данных начинается со специальной команды, при выполнении которой резервируется некоторая область хранилища, где будут сохраняться все объекты базы. Конечно, при создании можно указать не только имя базы данных, а место ее хранения, задать форматы по умолчанию и указать еще много полезных параметров.

Теперь можно приступить к созданию таблиц. Нам понадобится оператор CREATE TABLE, и затем в круглых скобках нужно описать создаваемую таблицу. Описание таблицы состоит из описания столбцов и ограничений целостности. Столбцы таблицы служат для представления атрибутов хранимых сущностей. Ограничения целостности — это правила, которым должны удовлетворять данные. Например, в фамилии не бывает цифр, оценка за экзамен может быть от 2 до 5. Очень важно при описании таблицы описать, какие значения могут принимать хранимые данные, чтобы предотвратить попадание в базу некорректных значений. Чтобы создать простую таблицу, нужно в круглых скобках просто описать столбцы, указывая название столбца и тип.

К примеру, опишем столбцы таблицы STUDENTS. В результате выполнения оператора CREATE TABLE будет создана пустая таблица с пятью столбцами.



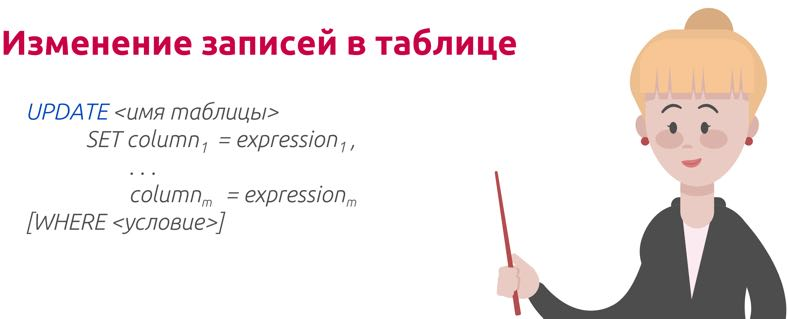
После того, как таблица создана, её структуру можно изменить. Для этого используют оператор ALTER TABLE. После ALTER TABLE указывается название таблицы, которую требуется изменить, а дальше описывается действие, которое нужно предпринять для изменения таблицы. Чтобы добавить колонку, используется фраза ADD COLUMN, в которую передается название колонки. К примеру, можно добавить номер телефона – поле Phone\_number, тип данных CHAR(11), затем сделать длину поля 10 символов, и удалить столбец.

Удалить таблицу можно с помощью оператора DROP TABLE. Для добавления записей используется оператор INSERT INTO.



Следующий пример демонстрирует добавление записей в таблицу STUDENTS. В первый раз названия столбцов не были указаны, т.к. добавляемые данные соответствовали порядку столбцов, заданному при создании таблицы. Во второй раз значения полей StudentId и StudentName были указаны не в том порядке, поэтому после названия таблицы был указан порядок полей добавляемых значений. У столбца BirthDate тип данных – дата. При вводе значений этого типа, надо знать установленный в системе формат по умолчанию – даты могут быть представлены в разном формате. Иногда бывает трудно определить, в каком именно. В этой ситуации на помощь может прийти функция преобразования строки в дату. В некоторых СУБД это функция to\_date. В качестве параметров функции указывают собственно дату в виде строки и ее формат.

Есть два оператора, которые можно использовать для удаления записей: DELETE и TRUNCATE. Первый оператор удаляет из таблицы строки, соответствующие указанным критериям, а второй удаляет все записи из таблицы. В качестве примера приведем команду для удаления из таблицы STUDENTS строки с параметром StudentId = 1294. Если выполнить второй оператор, указанный на рисунке, то из таблицы STUDENTS будут удалены все записи.

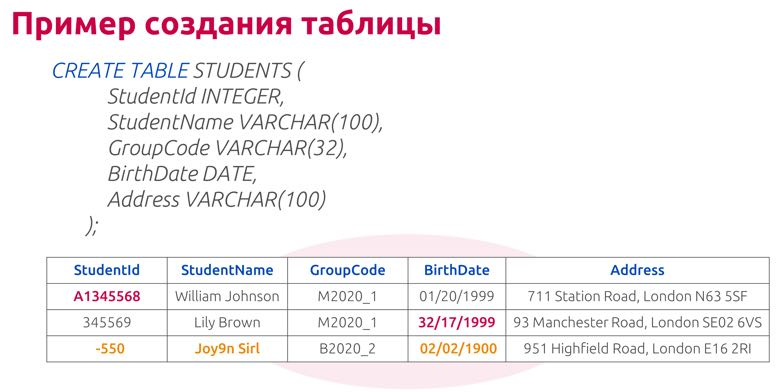
Для редактирования записей используется оператор UPDATE.

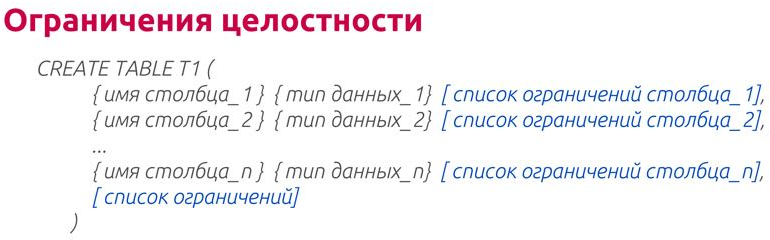
Например, изменим адрес у студента с полем StudentId = 345569. Обратите внимание, что при помощи оператора UPDATE можно изменить значение не только одного поля, а сразу нескольких полей, например адрес и имя, как показано во второй команде.

## **Ограничения целостности (ограничение неопределенных значений, допустимого диапазона значений, ограничение уникальности, ограничение первичного ключа и ссылочная целостность)**

Мы научились создавать таблицы и изменять их структуру. Для каждого столбца таблицы указывается название столбца и тип данных.

Также мы научились заполнять таблицы значениями при помощи оператора INSERT. При добавлении записей в таблицу производится автоматическая проверка соответствия добавляемых таким образом мы можем быть уверены, что в числовое поле не попадет буква, и в поле с типом дата нельзя ввести 32 число 15 месяца. Но может потребоваться задать более серьезные ограничения на возможные значения данных, чем просто соответствие типу данных.

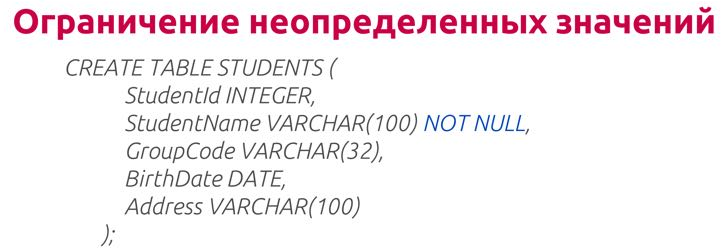


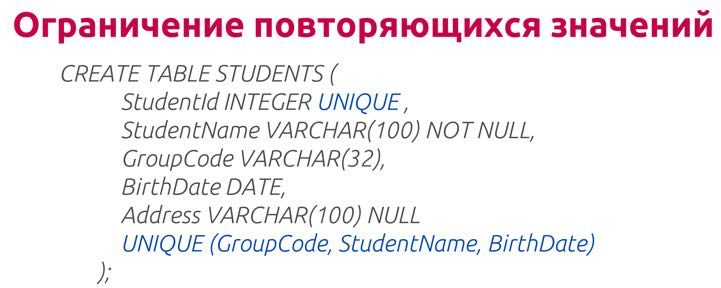
К примеру, может потребоваться указать, что идентификатор – положительное число, а в фамилии не бывает цифр, некоторые поля могут принимать значения из ограниченного множества – например, пол бывает мужской и женский, дней недели всего 7 и т.д.

Для этого помогут ограничения целостности, определяющие возможные значения данных. Существует несколько категорий правил целостности:

* ограничение неопределенных значений;
* ограничение допустимого диапазона значений;
* ограничение уникальности;
* ограничение первичного ключа;
* ссылочная целостность.

Некоторые атрибуты сущности могут иметь неопределенное значение, а другие всегда должны быть заданы. Это обязательные атрибуты, без которых строка таблицы не имеет смысла. Например, сложно представить, что может быть студент, у которого не определено имя или номер зачетки. Для того, чтобы значения поля всегда было определено, для него можно указать ограничение NOT NULL.



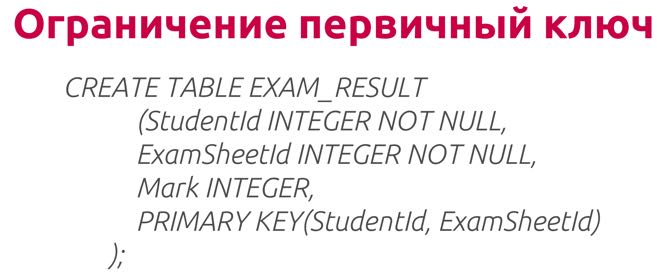
Если ограничение NOT NULL не задано, то по умолчанию значения полей могут быть не заданы. Это можно прописать в явном виде, например для поля адрес – слово NULL рядом с полем означает, что его значения могут быть не определены.

Согласно бизнес-логике некоторых данных, значения некоторых атрибутов таблицы должны быть уникальными. Например, уникальным должно быть значение поля StudentId, которое соответствует номеру зачетки. Чтобы сделать невозможным повторение значений в столбце, нужно задать ограничение UNIQUE.

Иногда уникальным нужно сделать не отдельное поле, а комбинацию полей. Например, потребуем, чтобы в каждой группе не было двух людей с одинаковыми именами и с одной датой рождения.

Когда ограничения затрагивают не одно поле, а несколько, то описание ограничений делают после описания всех полей таблицы. Нужно указать ограничение UNIQUE, затем в скобках перечислить поля, значения которых должны быть уникальной комбинацией. Ограничение уникальности поля можно усилить, добавив ограничение NOT NULL для поля StudentId.

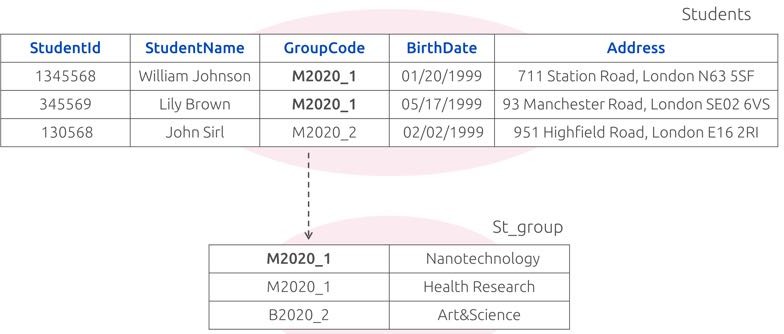


Более сильным, чем уникальность, является ограничение первичный ключ. Это ограничение требует уникальности значений поля, отсутствия не заданных значений. Кроме того, первичный ключ в таблице может быть только один. Но следует отметить, что в качестве первичного ключа может быть комбинация нескольких столбцов таблицы.

Очень важным ограничением, позволяющим хранить связи в таблицах, является ссылочная целостность. Чтобы продемонстрировать ее, рассмотрим еще одну таблицу ST\_GROUP. Каждый студент относится к определенной группе. Посмотрим на содержимое таблиц STUDENTS и ST\_GROUP. Вильям и Лили учатся в группе M2020\_2. Из таблицы ST\_GROUP мы можем найти их специализацию. Но как найти специализацию Джона? Он учится в группе M2020\_2, но такой группы нет в таблице ST\_GROUP. Мы видим, что логика данных нарушена. Нельзя указывать для студента группу, которой нет в списке групп. Чтобы такого не происходило, надо связать столбцы двух таблиц соответствующим правилом целостности.



Зададим для столбца с кодом группы в таблице STUDENTS внешний ключ – столбец GroupCode из таблицы ST\_GROUP. Такие ограничения не позволят добавить в таблицу STUDENTS значения в поле GroupCode, отсутствующие в соответствующем столбце связанной таблицы. Таблицу ST\_GROUP называют еще родительской.

Заметим, что внешний ключ должен быть такого же типа, как и связываемый столбец – в нашем случае они оба имеют тип VARCHAR(32).

Кроме того, поле внешнего ключа должно быть объявлено либо уникальным значением, либо первичным ключом.

Если строки родительской таблицы ST\_GROUP будут меняться или удаляться, то автоматически будет происходить поиск связанных строк в таблице STUDENTS. Предположим, что мы захотим удалить из таблицы ST\_GROUP группу M2020\_1 специальности Nanotechnology.

Тогда в зависимости от настроек ссылочной целостности, возможны несколько вариантов: например, можно запретить удаление этой группы, т.к. есть студенты, связанные с ней; либо можно удалить все связанные записи зависимых таблиц – при удалении группы удалять студентов, которые в ней учатся.

Последний вид ограничений, который мы рассмотрим, связан с указанием допустимого диапазона значений.

Это ограничение указывается ключевым словом CHECK() после описания атрибута. Если нужно проверить выполнение нескольких простых условий, то их можно соединять логическими операторами, например: CHECK(StudentID>5 AND StudentID<500).

При добавлении, удалении и изменении записей в таблицах будут автоматически проверяться все заданные ограничения целостности.

Таким образом ограничения целостности не позволят некорректным значениям попасть в базу данных.