IT CAREER HUB

Урок 4.1. Создание и изменение таблиц. Ограничения

Создание таблиц с нуля

Важно!

База данных с доступом на запись:
hostname: ich-edit.edu.itcareerhub.de
MYSQL_USER: ich1
MYSQL_PASSWORD: ich1_password_ilovedbs

Определение структуры таблицы включает в себя присвоение таблицы имени, указание какие столбцы будут присутствовать в этой таблице, какой тип данных они будут иметь и какие значения там могут присутствовать.

Ограничения (constraints) — это значения, которые можно вводить в столбцы, определяются специальными условиями.

Ограничения (constraints) в базах данных используются для обеспечения целостности и точности данных. Они помогают поддерживать правила и условия, которые должны соблюдаться в таблицах базы данных. В SQL существуют несколько основных типов ограничений, которые можно применить к столбцам и таблицам. Ниже представлено подробное описание каждого типа ограничения с примерами.

Типы ограничений

1. **PRIMARY KEY**

PRIMARY KEY — это ограничение, которое указывает, что столбец или комбинация столбцов уникально идентифицируют каждую строку в таблице.

Значения в столбце с первичным ключом не могут быть NULL и должны быть уникальными.

Часто используется в сочетании с **AUTO_INCREMENT.**

AUTO_INCREMENT — это специальное свойство, которое используется в базах данных для автоматического увеличения значения в столбце при добавлении новых записей.

Это особенно полезно для создания уникальных идентификаторов для каждой строки в таблице.

Когда вы создаете таблицу в базе данных и задаете одному из столбцов свойство AUTO_INCREMENT, база данных автоматически увеличивает значение этого столбца каждый раз, когда вы добавляете новую строку в таблицу.

Это значит, что вам не нужно вручную указывать уникальные значения для этого столбца — база данных сделает это за вас.

- Изначально: Первый раз, когда вы добавляете запись в таблицу, значение AUTO_INCREMENT обычно начинается с 1 (или другого начального значения, если вы его изменили).
- Автоматическое увеличение: При добавлении следующей записи значение автоматически увеличивается на 1. Так, если предыдущая запись имела значение 1, следующая запись получит значение 2, и так далее.
- Уникальные идентификаторы: Значения в столбце с AUTO_INCREMENT всегда уникальны в пределах таблицы, что делает его отличным выбором для первичных ключей.

2. **UNIQUE**

Обеспечивает уникальность значений в столбце или комбинации столбцов. Значения должны быть уникальными для всех строк, но могут быть NULL (в большинстве СУБД).

3. **NOT NULL**

Указывает, что столбец не может содержать NULL значения. Каждая строка должна иметь значение в этом столбце.

4. **DEFAULT**

Задает значение по умолчанию для столбца, если явно не указано другое значение при вставке новой строки.

5. **CHECK**

Указывает условие, которое должно быть выполнено для значений в столбце. Это ограничение проверяет данные на соответствие определенным условиям.

IT CAREER HUB

Общий шаблон создания таблицы

```
Unset
CREATE TABLE TableName

 ( Column1 DataType Constraints,

Column2 DataType Constraints,

 Column3 DataType Constraints, ...)
```

Перед созданием таблицы продумать, какие данные будет хранить каждый столбец, чтобы правильно выбрать тип данных и ограничения

IT CAREER HUB

Задание для закрепления

1. Создать таблицу Employees со следующими столбцами:

- EmployeeID
- FirstName
- LastName
- BirthDate
- HireDate
- Salary
- Email

2. Указать ограничения

- EmployeeID - первичный ключ, увеличивается автоматически на 1 при добавлении записи
- FirstName и LastName - строка длиной в 50 символов Не может быть пустой
- BirthDate - дата
- HireDate - дата по умолчанию указывается текущая дата
- Salary - число с количеством цифр 2 после запятой Общее число знаков, включая запятую, 10 Должна быть больше 0
- Email - строка длиной в 100 символов Должна быть уникальной

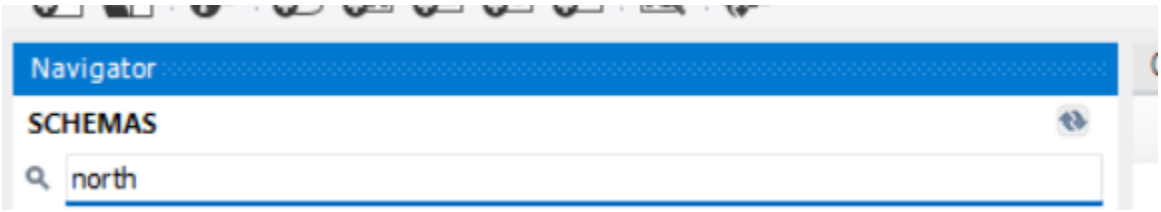
```
Unset

CREATE TABLE Employees (

    EmployeeID INT AUTO_INCREMENT PRIMARY KEY,

    FirstName VARCHAR(50) NOT NULL,

    LastName VARCHAR(50) NOT NULL,

    BirthDate DATE,

    HireDate DATE DEFAULT CURRENT_DATE,

    Salary DECIMAL(10, 2) CHECK (Salary > 0),

    Email VARCHAR(100) UNIQUE

);
```

Объяснение:

- EmployeeID: Столбец типа INT, который автоматически увеличивается при добавлении новых записей. Это первичный ключ таблицы.
- FirstName и LastName: Столбцы типа VARCHAR с максимальной длиной 50 символов, не допускающие NULL значения.
- BirthDate: Столбец типа DATE, который может содержать дату рождения сотрудника.
- HireDate: Столбец типа DATE с значением по умолчанию — текущая дата.
- Salary: Столбец типа DECIMAL с двумя десятичными знаками. Ограничение CHECK обеспечивает, чтобы зарплата была положительным числом.
- Email: Столбец типа VARCHAR с максимальной длиной 100 символов, значение должно быть уникальным.

3. Обновить схему, чтобы убедиться, что таблица появилась в базе данных.

4. Найти базу данных и раскрыть вкладку Tables.



5. Заполнить пустую таблицу значениями. При заполнении стоит учитывать, какие ограничения стоят на том или ином столбце - если Вы попытаетесь вставить значение, не соответствующее ограничению, то SQL покажет ошибку.

```
Unset
INSERT INTO TableName (Column1, Column2, Column3, ...)
VALUES (Value1, Value2, Value3, ...);
```

IT CAREER HUB

Задание для закрепления

1. Вставить в созданную таблицу 5 любых строк.

```
Unset
INSERT INTO Employees (FirstName, LastName, BirthDate, Salary, Email)

VALUES ('Alice', 'Green', '1985-05-15', 55000.00, 'alice.green@example.com'),
 ('Bob', 'Smith', '1990-08-22', 60000.00, 'bob.smith@example.com'),
('Charlie', 'Johnson', '1988-02-10', 52000.00, 'charlie.johnson@example.com'),
('Diana', 'Williams', '1992-11-01', 58000.00, 'diana.williams@example.com'),
('Edward', 'Brown', '1987-09-30', 61000.00, 'edward.brown@example.com');
```

2. Попытайтесь вставить строки, не соответствующие ограничениям.

Как правило, если нам нужна какая-либо таблица, за основу берется уже существующая таблица в БД, из которой выбираются необходимые нам строки.

```
Unset
CREATE TABLE NewTableName AS
SELECT * FROM ExistingTableName
WHERE условие
```

IT CAREER HUB

Задание для закрепления

Создайте таблицу с первыми двумя строками таблицы Employees.

```
Unset
CREATE TABLE Employees_short AS
SELECT * FROM Employees
LIMIT 2
```

IT CAREER HUB

Оператор UPDATE

Иногда при выявлении некорректных данных приходится исправлять значения вручную. Для этого мы можем использовать оператор UPDATE.

Оператор UPDATE — это оператор, который используется для изменения существующих записей в таблице.

Он позволяет обновлять значения одного или нескольких столбцов в строках, которые соответствуют заданному условию.

```
Unset
UPDATE TableName
SET Column1 = NewValue1, Column2 = NewValue2, …
 WHERE SomeCondition;
```

- **TableName** — имя таблицы, в которой вы хотите обновить данные.
- **Column1, Column2, ...** — имена столбцов, значения которых вы хотите изменить.
- **NewValue1, NewValue2, ...** — новые значения для этих столбцов.
- **SomeCondition** — условие, которое определяет, какие строки будут обновлены.

IT CAREER HUB

Задание для закрепления

1. Изменить зарплату сотрудника с EmployeeID = 1 на 65000

```
Unset
UPDATE Employees
SET Salary = 65000
WHERE EmployeeID = 1;
```

2. Увеличить зарплату всех сотрудников, работающих с 2024 года, на 10%

```
Unset
UPDATE Employees
SET Salary = Salary * 1.10
WHERE HireDate >= '2024-01-01';
```

IT CAREER HUB

Основные источники и методы внесения данных в базу данных

1. Импорт данных из внешних файлов

Часто данные в базу данных импортируются из внешних файлов, таких как CSV, Excel, XML или JSON. Это делается для того, чтобы перенести данные из других систем или источников.

Примеры включают:

- CSV (Comma-Separated Values): Простые текстовые файлы, где значения разделены запятыми. Этот формат легко читается и обрабатывается многими СУБД.
- Excel: Cпreadsheets из Microsoft Excel или Google Sheets могут быть импортированы в базу данных.
- XML/JSON: Форматы, используемые для обмена данными между системами.

2. Ввод данных через формы и пользовательские интерфейсы

Веб-приложения и другие пользовательские интерфейсы часто позволяют пользователям вводить данные через формы. Когда пользователь отправляет форму, данные передаются на сервер и сохраняются в базе данных. Это удобно для:

3. Внесение данных через программные интерфейсы (API)

Многие современные системы используют API (Application Programming Interface) для взаимодействия с базами данных. API позволяют автоматически передавать данные между системами без необходимости ручного ввода.

4. Данные из других баз данных

Иногда данные копируются из одной базы данных в другую. Это может быть сделано через ETL (Extract, Transform, Load) процессы или с помощью специальных инструментов миграции.

Примеры:

- Резервное копирование и восстановление данных.
- Синхронизация данных между различными системами.

Почему данные не вносятся вручную

1. Эффективность и масштабируемость: Вручную вносить данные сложно и времязатратно, особенно когда объем данных велик. Автоматизация позволяет обрабатывать большие объемы данных быстрее и с меньшими затратами.
2. Ошибки и точность: Автоматизированные процессы снижают риск человеческих ошибок и обеспечивают более высокую точность данных.
3. Интеграция систем: Автоматизация позволяет интегрировать различные системы и источники данных, обеспечивая более плавный обмен информацией.
4. Обновления и поддержка: Автоматические процессы позволяют поддерживать данные актуальными и легко обновлять их без ручного вмешательства.