IT CAREER HUB

Урок 3.1. Типы данных. Изменение типов данных

Введение в типы данных

 **Важно!**

На уроке будет использоваться база данных с доступом на чтение:
hostname: ich-db.edu.itcareerhub.de
username: ich1
password: password

На этом уроке работаем с БД northwind

Типы данных — это категории данных, которые определяют, какой тип значения может храниться в поле базы данных.

При создании таблицы каждому столбцу назначается определенный тип данных, который определяет, какие операции можно выполнять с данными этого столбца.

Например, если в таблице есть столбец для хранения имен людей, мы указываем, что этот столбец будет хранить текст. Если в другом столбце хранятся числа, мы указываем, что этот столбец будет хранить числа.

Зачем нужны типы данных

1. **Правильное хранение информации:** Типы данных помогают правильно хранить информацию. Например, числа хранятся по-другому, чем текст, чтобы их можно было использовать в вычислениях.
2. **Эффективность:** Правильный тип данных помогает базе данных работать быстрее. Например, если вы храните даты в специальном формате, база данных может быстрее выполнять операции с датами, такие как сравнения или вычисления.
3. **Избежание ошибок:** Указывая тип данных, мы уменьшаем риск ошибок. Например, вы не сможете случайно записать текст в столбец, который должен хранить только числа.
4. **Корректные операции:** Разные типы данных поддерживают разные операции. Например, текст можно соединять (например, «Иван» + «Иванов»), а числа можно складывать и вычитать.

Таким образом, типы данных помогают базе данных хранить информацию правильно и эффективно, а также предотвращают ошибки и обеспечивают корректность операций с данными.

Основные типы данных

1. **Числовые типы данных:**
   - **INT**: Целочисленные значения (например, 1, 100, -25).
   - **FLOAT, DOUBLE:** Числа с плавающей точкой (например: 5.23E+05, -4.678678)
   - **DECIMAL/NUMERIC**: Точные числа с фиксированной точкой (например, 10.99, 1000.50).

2. **Строковые типы данных:**
   - **CHAR**: Строки фиксированной длины (например, 'ABC').
   - **VARCHAR**: Строки переменной длины (например, 'Hello World').
   - **TEXT**: Большие текстовые блоки (например, статьи, комментарии).

3. **Дата и время:**
   - **DATE**: Дата (например, '2023-08-25').
   - **TIME**: Время (например, '14:30:00').
   - **DATETIME**: Дата и время (например, '2023-08-25 14:30:00').
   - **TIMESTAMP**: Момент времени с учетом временной зоны.

4. **Булевы значения:**
   - **BOOLEAN**: Логическое значение TRUE или FALSE.

Как посмотреть типы данных у столбцов

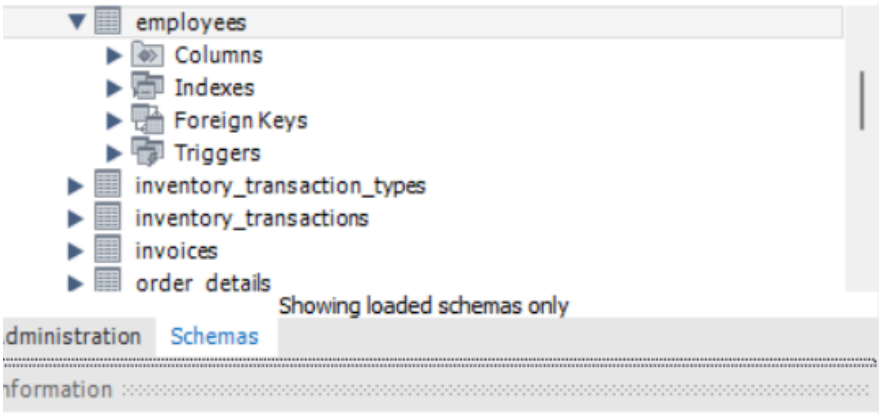

Table: employees

Columns:

id	int AI PK
company	varchar(50)
last_name	varchar(50)
first_name	varchar(50)
email_address	varchar(50)
job_title	varchar(50)
business_phone	varchar(25)
home_phone	varchar(25)
mobile_phone	varchar(25)
fax_number	varchar(25)
address	longtext
city	varchar(50)
state_province	varchar(50)
zip_postal_code	varchar(15)
country_region	varchar(50)
web_page	longtext
notes	longtext
attachments	longblob

IT CAREER HUB

Операции, которые можно проводить над столбцами в зависимости от типа данных

Тип данных	Описание	Операции	Примеры SQL
Числовые (INT, DECIMAL/NUMERIC, DOUBLE)	Хранят числовые значения (целые, с фиксированной точкой, с плавающей точкой)	Арифметические операции, сравнение	`SELECT quantity * unit_price AS total_price FROM order_details;`
			`SELECT (list_price / standard_cost - 1) * 100 AS markup_percentage FROM products;`
Строковые (CHAR(n), VARCHAR(n), TEXT)	Хранят строковые значения (фиксированной длины, переменной длины, большие текстовые блоки)	Конкатенация, извлечение подстроки, преобразование регистра, замена, поиск	`SELECT CONCAT(first_name, ' ', last_name) AS full_name FROM employees;`
			`SELECT LEFT(product_name, 5) FROM products;`
			`SELECT UPPER(company) FROM customers;`
			`SELECT SUBSTRING(product_name, 19, 30) FROM products;`

BOOLEAN	TRUE или FALSE	Логические операции	`SELECT * FROM employees WHERE is_manager = TRUE;`

IT CAREER HUB

Задание для закрепления

1. **Таблица:** `order_details`

Задача: Выведите информацию о каждом заказе, включая идентификатор заказа (`order_id`), расчетную полную стоимость заказа после применения скидки (`net_price`).

```
None
SELECT order_id, (unit_price * quantity) - (unit_price * quantity * discount)
AS net_price FROM order_details;
```

2. **Таблица:** `customers`

Задача: Выведите полный адрес каждого клиента, объединяя адрес (`address`), город (`city`) и страну (`country`) в одну строку.

```
None
SELECT id, CONCAT(address, ', ', city, ', ', country_region) AS full_address
FROM customers;
```

3. **Таблица:** `employees`

Задача: Выведите информацию о каждом сотруднике, включая идентификатор сотрудника (`id`), имя (`first_name`), фамилию (`last_name`) и роль (`role`), где роль определяется на основе значения поля `is_manager` (если значение 1, то "Manager", иначе "Employee").

```
None
SELECT id, first_name, last_name, CASE WHEN is_manager = 1 THEN 'Manager' ELSE
'Employee' END AS role FROM employees;
```

IT CAREER HUB

Преобразование типов

Преобразованием типов — это преобразование одного типа данных в другой.

Это позволяет вам изменить формат данных в запросе, чтобы использовать их в нужном виде. В SQL существуют несколько способов для преобразования типов данных, в зависимости от используемой системы управления базами данных (СУБД).

Функция CAST — это функция, которая используется для явного преобразования, заданного вами как пользователем, одного типа данных в другой.

```
None
CAST(expression AS target_data_type)
SELECT CAST('123' AS INT) AS Number;
SELECT CAST(OrderDate AS VARCHAR) AS OrderDateString
FROM Orders;
```

Необходимость преобразования типов данных

1. **Совместимость данных:**
    - **Разные форматы:** В одной таблице могут храниться данные в разных форматах. Например, даты могут быть в текстовом формате, а числа — в числовом. Если вы хотите сравнить или объединить такие данные, их нужно преобразовать в совместимый формат.
2. **Арифметические операции:**
    - **Числа и текст:** Если вы храните числа как текст (строки), вы не сможете с ними делать арифметические операции, такие как сложение или умножение. Преобразование текста в числа позволяет вам выполнять такие операции.
3. **Форматирование вывода:**
    - **Человеческое восприятие:** Иногда данные нужно отформатировать для удобного отображения. Например, преобразовать дату в строку для отчета или преобразовать число в строку для объединения с другим текстом.
4. **Фильтрация и сортировка:**
    - **Точные результаты:** Для корректной фильтрации или сортировки данных может потребоваться преобразование их в нужный формат. Например, сортировка дат или поиск чисел.

IT CAREER HUB

Задание для закрепления

Работа с числами

Ситуация: У вас есть числовые данные, которые хранятся в текстовом формате. Например, количество товаров на складе.

Проблема: Вы хотите рассчитать общую стоимость товаров на складе. Если количество товаров хранится как текст, вы не сможете сделать расчет.

Решение: Преобразуйте текстовое значение количества в число, чтобы провести арифметическую операцию.

```
None
SELECT product_name,
       CAST(unit_in_stock AS SIGNED) * list_price AS total_value
FROM products;
```

Объяснение: `CAST(unit_in_stock AS SIGNED)` преобразует количество товаров из текста в число, что позволяет умножить его на цену за единицу и вычислить общую стоимость.

IT CAREER HUB

Задание для закрепления

Создать отчет, который показывает количество и цену продуктов в текстовом формате, чтобы представить информацию в более понятном виде для конечных пользователей.

Проблема: Количество и цена хранятся в числовом формате, но для отчетов вы хотите объединить эти данные в строку, которая будет легко читаться.

Решение: Преобразуйте числовые значения в строки и объедините их с дополнительным текстом для создания более понятного отчета.

```
None
SELECT product_name, CONCAT('Available: ', CAST(unit_in_stock AS CHAR), '
units, Price: $', CAST(list_price AS CHAR)) AS product_report FROM products;
```

Скрытые преобразования в MySQL

Скрытые преобразования типов данных — это автоматическое преобразование данных одного типа в данные другого типа, и MySQL сам решает, как преобразовать данные для выполнения операции.

Эти преобразования могут быть неявными, что означает, что они происходят без явного указания пользователя.

Основные принципы скрытых преобразований в MySQL

1. **Автоматическое преобразование типов данных:**
   - MySQL автоматически преобразует один тип данных в другой, когда выполняется операция с различными типами данных. Это делается для того, чтобы упростить выполнение запросов и предотвратить ошибки.
2. **Правила преобразования:**
   - MySQL следует определенным правилам при преобразовании типов данных. Например, при выполнении арифметических операций над строками, которые содержат числа, строки будут преобразованы в числовые значения.
3. **Тип данных и приоритеты:**
   - При смешивании типов данных MySQL использует правила приоритета типов данных. Например, при комбинировании строк и чисел, число будет преобразовано в строку, чтобы операция могла быть выполнена.

Примеры скрытых преобразований в MySQL:

1. Преобразование строк в числа

Когда вы выполняете арифметическую операцию со строками, которые содержат числа, MySQL автоматически преобразует строки в числа.

Пример:

```
None
SELECT '100' + 50 AS Result;
```

2. Преобразование чисел в строки

Когда вы объединяете строку с числом, число автоматически преобразуется в строку.

```
None
SELECT CONCAT('The total amount is ', 150) AS Message;
```

3. Преобразование типов при сравнении

Когда вы сравниваете строку и число, строка автоматически преобразуется в число.

```
None
SELECT * FROM products WHERE unit_in_stock = '50';
```

Правила и приоритеты преобразования:

1. **Типы данных с плавающей запятой и целые числа:**
    - При смешивании типа данных с плавающей запятой и целого числа, целое число будет преобразовано в число с плавающей запятой.
2. **Строки и числа:**
    - При объединении строки с числом, число будет преобразовано в строку.
    - При выполнении арифметических операций со строками, содержащими числа, строки будут преобразованы в числа.

Скрытые преобразования типов данных в MySQL облегчают выполнение операций и запросов, автоматически обрабатывая несовместимые типы данных. Это упрощает работу с данными, но важно понимать, как эти преобразования работают, чтобы избежать неожиданных результатов.