MySQL

CRUD

Insert

Добавление данных.

Для добавления данных в БД в MySQL используется команда **INSERT**, которая имеет следующий формальный синтаксис:

```
INSERT [INTO] имя_таблицы [(список_столбцов)] VALUES
(значение1, значение2, ... значениеN)
```

После выражения **INSERT INTO** в скобках можно указать список столбцов через запятую, в которые надо добавлять данные, и в конце после слова **VALUES** скобках перечисляют добавляемые для столбцов значения.

Insert

Добавление данных.

Например, пусть в базе данных productsdb есть следующая таблица Products:

```
CREATE TABLE Products
(
    Id INT AUTO_INCREMENT PRIMARY KEY,
    ProductName VARCHAR(30) NOT NULL,
    Manufacturer VARCHAR(20) NOT NULL,
    ProductCount INT DEFAULT 0,
    Price DECIMAL NOT NULL
);
```

Добавим в эту таблицу одну строку с помощью следующего кода:

```
INSERT Products(ProductName, Manufacturer, ProductCount, Price)
VALUES ('iPhone X', 'Apple', 5, 76000);
```

Insert

Добавление данных.

- В данном случае значения будут передаваться столбцам по позиции. То есть стобцу ProductName передается строка "iPhone X", столбцу Manufacturer - строка "Apple" и так далее.
- Важно, чтобы между значениями и типами данных столбцов было соответствие. Так, столбец ProductName представляет тип varchar, то есть строку. Соответственно этому столбцу мы можем передать строковое значение в одинарных кавычках. А стобец ProductCount представляет тип int, то есть целое число, поэтому данному столбцу нужно передать целые числа, но никак не строки.

Insert

Добавление данных.

- Необязательно при добавлении данных указывать значения абсолютно для всех столбцов таблицы. Например, в примере выше не указано значение для стобца Id. Но поскольку для данного столбца определен атрибут AUTO_INCREMENT, то его значение будет автоматически генерироваться.
- Также мы можем опускать при добавлении такие столбцы, которые поддерживают значение NULL или для которых указано значение по умолчанию, то есть для них определены атрибуты NULL или DEFAULT. Так, в таблице Products столбец ProductCount имеет значение по умолчанию - число 0.

Insert

Добавлени е данных.

Поэтому мы можем при добавлении опустить этот столбец, и ему будет передаваться число 0:

```
INSERT Products(ProductName, Manufacturer, Price)
VALUES ('Galaxy S9', 'Samsung', 63000);
```

С помощью ключевых слов DEFAULT и NULL можно указать, что в качестве значения будет использовать значение по умолчанию или NULL соответственно:

```
INSERT Products(ProductName, Manufacturer, Price,
ProductCount)
VALUES ('Nokia 9', 'HDM Global', 41000, DEFAULT);
```

Или

```
INSERT Products(ProductName, Manufacturer, Price,
ProductCount)
VALUES ('Nokia 9', 'HDM Global', 41000, NULL);
```

Insert

Добавление данных.

Множественное добавление

Также мы можем добавить сразу несколько строк:

```
INSERT Products(ProductName, Manufacturer, Price,
ProductCount)
VALUES
('iPhone 8', 'Apple', 51000, 3),
('P20 Lite', 'Huawei', 34000, 4),
('Galaxy S8', 'Samsung', 46000, 2);
```

В данном случае в таблицу будут добавлены три строки.

Select

Выборка данных

Для выборки данных из БД в MySQL применяется команда **SELECT**. В упрощенном виде она имеет следующий синтаксис:

```
SELECT список_столбцов FROM имя_таблицы
```

Select

Выборка данных

Например, пусть ранее была создана таблица Products, и в нее добавлены некоторые начальные данные:

```
CREATE TABLE Products
(
   Id INT AUTO_INCREMENT PRIMARY KEY,
   ProductName VARCHAR(30) NOT NULL,
   Manufacturer VARCHAR(20) NOT NULL,
   ProductCount INT DEFAULT 0,
   Price DECIMAL
);

INSERT INTO Products (ProductName, Manufacturer, ProductCount, Price)
VALUES
('iPhone X', 'Apple', 3, 76000),
('iPhone 8', 'Apple', 2, 51000),
('Galaxy S9', 'Samsung', 2, 56000),
('Galaxy S8', 'Samsung', 1, 41000),
('P20 Pro', 'Huawei', 5, 36000);
```

Select

Выборка данных

Получим все объекты из этой таблицы:

SELECT * FROM Products;

```
1  USE productsdb;
2
3  SELECT * FROM Products;
```

Result Grid | Filter Rows: | Edit:

Id	ProductName	Manufacturer	ProductCount	Price
1	iPhone X	Apple	3	76000
2	iPhone 8	Apple	2	51000
3	Galaxy S9	Samsung	2	56000
4	Galaxy S8	Samsung	1	41000
5	P20 Pro	Huawei	5	36000
NULL	NULL	NULL	NULL	NULL

Select

Выборка данных

Стоит отметить, что применение звездочки * для получения данных считается не очень хорошей практикой, так как обычно необходимо получить данные по небольшому набору столбцов. Поэтому более оптимальный подход заключается в указании всех необходимых столбцов после слова SELECT. Исключение составляет тот случай, когда надо получить данные по абсолютно всем столбцам таблицы. Также использование символа * может быть предпочтительно тогда, когда названия столбцов не известны.

Select

Выборка данных

Если необходимо получить данные не из всех, а из каких-то конкретных столбцов, тогда спецификации этих столбцов перечисляются через запятую после SELECT:

SELECT ProductName, Price FROM Products;

```
1  USE productsdb;
2
3  SELECT ProductName, Price FROM Products;
```

Result Grid | Filter Rows: | Export:

ProductName	Price
iPhone X	76000
iPhone 8	51000
Galaxy S9	56000
Galaxy S8	41000
P20 Pro	36000

Select

Выборка данных

Спецификация столбца необязательно должна представлять его название. Это может быть любое выражение, например, результат арифметической операции. Так, выполним следующий запрос:

```
SELECT ProductName, Price * ProductCount FROM Products;
```

Здесь при выборке будут создаваться два столбца. Причем второй столбец представляет значение столбца Price, умноженное на значение столбца ProductCount, то есть совокупную стоимость товара.

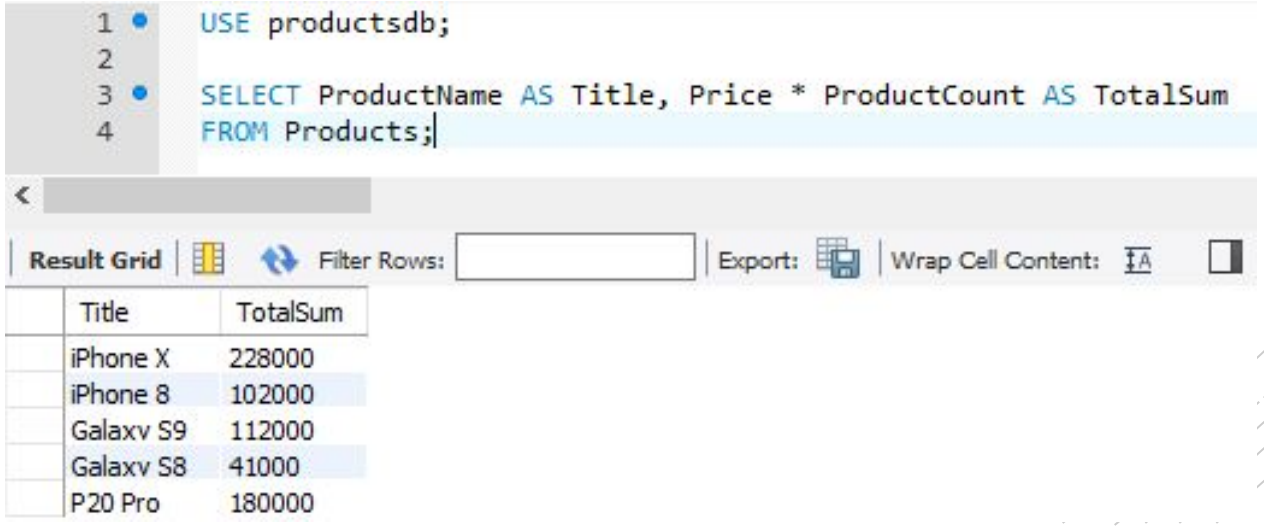
Select

Выборка данных

С помощью оператора **AS** можно изменить название выходного столбца или определить его псевдоним:

```
SELECT ProductName AS Title, Price * ProductCount AS TotalSum
FROM Products;
```

Здесь для первого столбца определяется псевдоним Title, хотя в реальности он будет представлять столбец ProductName. Второй столбец TotalSum хранит произведение столбцов ProductCount и Price.



Where

Фильтр данных

Зачастую необходимо извлекать не все данные из БД, а только те, которые соответствуют определенному условию. Для фильтрации данных в команде SELECT применяется оператор **WHERE**, после которого указывается условие:

WHERE условие

Where

Фильтр данных

Если условие истинно, то строка попадает в результирующую выборку. В качестве можно использовать операции сравнения, которые сравнивают два выражения:

- **=**: сравнение на равенство
- **!=**: сравнение на равенство
- **<>**: сравнение на неравенство
- **<**: меньше чем
- **>**: больше чем
- **<=**: меньше чем или равно
- **>=**: больше чем или равно

Where

Фильтр данных

Если условие истинно, то строка попадает в результирующую выборку. В качестве можно использовать операции сравнения, которые сравнивают два выражения:

- **=**: сравнение на равенство
- **!=**: сравнение на равенство
- **<>**: сравнение на неравенство
- **<**: меньше чем
- **>**: больше чем
- **<=**: меньше чем или равно
- **>=**: больше чем или равно

Where

Фильтр данных

К примеру, выберем всех товары, производителем которых является компания Samsung:

SELECT * FROM Products

WHERE Manufacturer = 'Samsung;

```
1  USE productsdb;
2
3  SELECT * FROM Products
4  WHERE Manufacturer = 'Samsung';
```

Result Grid | Filter Rows: | Edit:

Id	ProductName	Manufacturer	ProductCount	Price
3	Galaxy S9	Samsung	2	56000
4	Galaxy S8	Samsung	1	41000
NULL	NULL	NULL	NULL	NULL

Where

Фильтр данных

Стоит отметить, что для MySQL не важен регистр символов, и, к примеру, строка "Samsung" будет эквивалентна строке "SAMSUNG" или "sumSunG".

Другой пример - найдем все товары, количество которых меньше 3:

```
SELECT * FROM Products
WHERE ProductCount < 3
```

Критерий фильтрации может представлять и более сложное составное выражение. Например, найдем все товары, у которых совокупная стоимость больше 100 000:

```
SELECT * FROM Products
WHERE Price * ProductCount > 100000;
```

Where

Фильтр данных

Логические операторы

Логические операторы позволяют объединить несколько условий. В MySQL можно использовать следующие логические операторы:

AND: операция логического И. Она объединяет два выражения:

выражение1 AND выражение2

Только если оба этих выражения одновременно истинны, то и общее условие оператора AND также будет истинно. То есть если и первое условие истинно, и второе.

Where

Фильтр данных

Логические операторы

OR: операция логического ИЛИ. Она также объединяет два выражения:

```
выражение1 OR выражение2
```

Если хотя бы одно из этих выражений истинно, то общее условие оператора OR также будет истинно. То есть если или первое условие истинно, или второе.

Where

Фильтр данных

Логические операторы

NOT: операция логического отрицания. Если выражение в этой операции ложно, то общее условие истинно.

NOT выражение

Where

Фильтр данных

Например, выберем все товары, у которых производитель Samsung и одновременно цена больше 50000:

```
SELECT * FROM Products
WHERE Manufacturer = 'Samsung' AND Price > 50000
```

Where

Фильтр данных

Теперь изменим оператор на **OR**. То есть выберем все товары, у которых либо производитель Samsung, либо цена больше 50000:

```
SELECT * FROM Products
WHERE Manufacturer = 'Samsung' OR Price > 50000
```

Where

Фильтр данных

Применение оператора **NOT** - выберем все товары, у которых производитель не Samsung:

```
SELECT * FROM Products
WHERE NOT Manufacturer = 'Samsung';
```

Where

Фильтр данных

Приоритет операций

В одном условии при необходимости мы можем объединять несколько логических операций. Однако следует учитывать, что самой приоритетной операцией, которая выполняется в первую очередь, является NOT, менее приоритетная - AND и операция с наименьшим приоритетом - OR. Например:

```
SELECT * FROM Products WHERE Manufacturer ='Samsung' OR NOT
Price > 30000 AND ProductCount > 2;
```

В данном случае сначала вычисляется выражение NOT Price > 30000, то есть цена должна быть меньше или равна 30000.

Затем вычисляется выражение NOT Price > 30000 AND ProductCount > 2, то есть цена должна быть меньше или равна 30000 и одновременно количество товаров должно быть больше 2.

В конце вычисляется оператор OR - либо цена должна быть меньше или равна 30000 и одновременно количество товаров должно быть больше 2, либо производителем должен быть Samsung

Where

Фильтр данных

С помощью скобок можно переопределить приоритет операций:

```
SELECT * FROM Products
WHERE Manufacturer ='Samsung' OR NOT (Price > 30000
AND ProductCount > 2);
```

В данном случае находим товары, у которых либо производитель Samsung, либо одновременно цена товара меньше или равна 30000 и количество товаров меньше 3.

Update

Обновление данных

Команда **UPDATE** применяется для обновления уже имеющихся строк. Она имеет следующий формальный синтаксис:

```
UPDATE имя_таблицы
SET столбец1 = значение1, столбец2 = значение2, ...
столбецN = значениеN
[WHERE условие_обновления]
```

Update

Обновление данных

Например, увеличим у всех товаров цену на 3000:

```
UPDATE Products
SET Price = Price + 3000;
```

Update

Обновление данных

Используем выражение WHERE и изменим название производителя с "Samsung" на "Samsung Inc.":

```
UPDATE Products
SET Manufacturer = 'Samsung Inc.'
WHERE Manufacturer = 'Samsung';
```

Также можно обновлять сразу несколько столбцов:

```
UPDATE Products
SET Manufacturer = 'Samsung',
    ProductCount = ProductCount + 3
WHERE Manufacturer = 'Samsung Inc.';
```

Update

Обновление данных

При обновлении вместо конкретных значений и выражений мы можем использовать ключевые слова **DEFAULT** и **NULL** для установки соответственно значения по умолчанию или NULL:

```
UPDATE Products
SET ProductCount= DEFAULT
WHERE Manufacturer = 'Huawei';
```

Delete

Удаление данных

Команда **DELETE** удаляет данные из БД. Она имеет следующий формальный синтаксис:

```
DELETE FROM имя_таблицы
[WHERE условие_удаления]
```

Delete

Удаление данных

Например, удалим строки, у которых производитель - Huawei:

```
DELETE FROM Products
WHERE Manufacturer='Huawei';
```

Delete

Удаление данных

Или удалим все товары, производителем которых является Apple и которые имеют цену меньше 60000:

```
DELETE FROM Products
WHERE Manufacturer='Apple' AND Price < 60000;
```

Если необходимо вовсе удалить все строки вне зависимости от условия, то условие можно не указывать:

```
DELETE FROM Products;
```

HW

Домашнее задание

В таблицы авторов и книг добавить любых 20 авторов и 40 любых книг этих авторов. Удалить авторов, которые старше 100 лет, добавить магазины (один из этих магазинов должен иметь имя – «класс»), и обновить все магазины, которые имеют имя – «класс» на «рост»