Запросы корректировки данных

Преподаватель:

канд. тех. наук, доц. Озерова Г.П.

Запросы корректировки

SQL позволяет корректировать информацию в таблицах. Для этого используются запросы корректировки данных. С их помощью можно:

- создать пустую таблицу;
- добавить в таблицу записи как совокупность значений;
- добавить записи из другой таблицы;
- изменить значения в одном или нескольких столбцах;
- удалить записи из таблицы;
- создать таблицу на основе данных других таблиц.

Создание пустой таблицы

```
CREATE TABLE имя_таблицы(
    столбец_1 тип_столбца характеристики,
    столбец_2 тип_столбца характеристики,
    …,
    столбец_N тип_столбца характеристики
);
```

Выбор типов данных для полей

MySQL поддерживает множество типов данных.

Выбор правильного типа для хранения вашей информации критичен с точки зрения увеличения производительности.

Рекомендации по выбору типов данных

Меньше — обычно лучше. Типы данных должны быть минимального размера, достаточного для их хранения и представления.

Причины:

- меньшие по размеру типы данных обычно быстрее, поскольку занимают меньше места на диске, в памяти и кэше процессора;
- для их обработки обычно требуется меньше процессорного времени.

Рекомендации по выбору типов данных

Просто — значит хорошо. Желательно использовать встроенный типы данных, а не моделировать их с помощью других (например, не представлять дату в виде текста)

Причины:

- для выполнения операций с более простыми типами данных обычно требуется меньше процессорного времени.

Рекомендации по выбору типов данных

Насколько это возможно, избегайте значений NULL.

Причины:

- MySQL тяжело оптимизировать запросы, содержащие допускающие **NULL** столбцы, поскольку из-за них усложняются индексы, статистика индексов и сравнение значений;
- столбец, допускающий **NULL**, занимает больше места на диске и требует специальной обработки.

Алгоритм выбора типов данных

Шаг 1. Определение общего класса типов: числовые, строковые, временные и т. п.

Шаг 2. Выбор конкретного типа. Многие типы данных MySQL позволяют хранить данные одного и того же вида, но с разными диапазоном значений и точностью. Кроме того, некоторые типы данных обладают специальным поведением или свойствами.

Алгоритм выбора типов данных

Особенности выбора различных типов данных SQL

Логический тип данных

Примеры: TRUE (1), FALSE (0)

Тип данных	Количество байт	Диапазон
BOOLEAN **BOOL**	1	0 или 1

Рекомендации по использованию:

- это самый экономичный и эффективный при использовании тип данных, если какое-то поле может принимать только два значения (например, М или Ж), то всегда нужно выбирать тип данных **BOOLEAN**.

Типы данных для целых чисел

Примеры: 2367, -347829, 0

Тип данных	Количество бит (n)	Диапазон
TINYINT	8	от -128 до 127
SMALLINT	16	от -32768 до 32757
MEDIUMINT	24	от -8388608 до 8388607
INT	32	от -2147483648 до 2147483647
BIGINT	64	от -9223372036854775808 до 9223372036854775807

Формула для вычисления диапазона:

$$[-2^{(n-1)}; 2^{(n-1)} - 1]$$

Типы данных для целых чисел

Примеры: 2367, -347829, 0

Тип данных	Количество бит (n)	Диапазон
`TINYINT UNSIGNED`	8	от 0 до 255
`SMALLINT UNSIGNED`	16	от 0 до 65535
`MEDIUMINT UNSIGNED`	24	от 0 до 16777215
`INT UNSIGNED`	32	от 0 до 4294967295
`BIGINT UNSIGNED`	64	от 0 до $9 \cdot 10^{21}$

Формула для вычисления диапазона:

$[0; 2^n - 1]$

Типы данных для целых чисел

Рекомендации по выбору:

- используйте принцип «**Чем меньше, тем лучше**», тип для целых чисел влияет на размер базы на диске, выбор типа целых чисел не влияет на производительность, так как для целочисленных вычислений обычно используются 64-разрядные целые типа BIGINT;
- знаковые и беззнаковые типы требуют одинаковой памяти и обладают одинаковой производительностью, так что используйте тот тип, который больше подходит для диапазона ваших данных.

Типы данных для вещественных чисел

Примеры: 23.111, -347829.9, 0.0001

Тип данных	Описание
DECIMAL(m,d) **NUMERIC(m,d)**	Вещественное число (с фиксированной точкой). В скобках указывается максимальная длина числа **m** (включает символы слева и справа от десятичной запятой и символ точки) и количество знаков после запятой **d**. **max(m) = 64, max(d) = 30** **Пример:** **DECIMAL(5,2)** - будет хранить числа от -99.99 до 99.99.
FLOAT(m,d) **DOUBLE(m,d)**	Вещественное число (с плавающей точкой). Может иметь параметр **UNSIGNED**, запрещающий отрицательные числа. **m** - количество отводимых под число символов; **d** - количество символов дробной части. **Пример:** **FLOAT (5,2)** - будет хранить числа из 5 символов, 2 из которых будут идти после запятой (например: 46,58).

Типы данных для вещественных чисел

Примеры: 23.111, -347829.9, 0.0001

Тип данных	Тип вычислений	Скорость выполнения	Количество байт	Диапазон
DECIMAL(m,d) **NUMERIC(m,d)**	точные	медленнее	m + 2	от $-(10^{(m-d-1)}-1).(10^{d}-1)$ до $(10^{(m-d-1)}-1).(10^{d}-1)$
FLOAT(m,d)	приближенные	быстрее	4	от $-1.175494351 \cdot 10^{-39}$ до $1.175494351 \cdot 10^{39}$
DOUBLE(m,d)	приближенные	быстрее	8	от $2.225073859 \cdot 10^{-308}$ до $2.225073859 \cdot 10^{308}$

Типы данных для вещественных чисел

Рекомендации по выбору:

- **DECIMAL** и **NUMERIC** необходимо использовать только тогда, когда нужны точные результаты при вычислениях с дробными числами, например при хранении финансовых данных;
- в остальных случаях используйте **FLOAT** или **DOUBLE**;
- хранение и обработка данных эффективнее, если применяются целые типы данных, при возможности, особенно если используются большие числа, переходите к целым типам (например, нам нужно хранить информацию в рублях и копейках, а диапазон – миллиарды рублей, тогда рекомендуется умножить значение на 100 и хранить данные как целые числа, а при вычислениях делить на 100).

Типы данных для строк

Примеры: '', 'Python', 'Фамилия', '23-23-01'

Тип данных	Количество байт	Макс. размер	Описание
CHAR (m)	**m**	зависит от кодировки	Позволяет хранить строку фиксированной длины **m**. Значение **m** - от 0 до 255. **Примеры:** **CHAR (8)** – при однобайтовой кодировке хранит строки из 8 символов и занимает 8 байтов: • '', 'SQL', 'MySQL' - занимают по 8 байтов памяти; • 'Python and SQL' → 'Python a' , т.е. до 8 символов. **CHAR (8)** – при двухбайтовой кодировке хранит строки из 4 символов и занимает 8 байтов: • '', ‘Ас', ‘Коля' - занимают по 8 байтов памяти; • ‘программирование' → 'прог' , т.е. до 4 символов. **CHAR (8) –** при кодировках типа UTF-8 количество символов в строке может быть разным, занимает 8 байтов. 'я и SQL' → 'я и SQ' (2 английские буквы и два пробела по 1 байту и две русские буквы по два байта)

Типы данных для строк

Особенности типа **CHAR**:

- тип **CHAR** имеет фиксированную длину;
- MySQL удаляет все пробелы в конце строки;
- к значениям меньшей длины для сравнения добавляются пробелы.

Типы данных для строк

Примеры: '', 'Python', 'Фамилия', ‘23-23-01'

Тип данных	Количество байт	Макс. размер	Описание
VARCHAR (m)	для строк, длиной <= 255 **k+1**, для строк, длиной > 255 **k+2**, где **k=0..m**	зависит от кодировки	Позволяет хранить строки с изменяемой длиной **k**, **k = 0..m.** Значение **m** - от 0 до 65 535. **Примеры:** **VARCHAR(3)** – для однобайтовой кодировки хранит строки максимум из 3 символов: • пустая строка '' занимает 1 байт, • строка 'а' - 2 байта, • строк 'аа' - 3 байта, • строка 'ааа' - 4 байта. Значение более 3 символов будет усечено до 3. **VARCHAR(3)** – для двухбайтовой кодировки хранит строки максимум из 1 символа: • пустая строка '' занимает 1 байт, • строка 'д' - 3 байта. Значение более 1 символа будет усечено до 1.

Типы данных для строк

Особенности типа **VARCHAR**:

- строки этого типа могут занимать меньше места, чем строки фиксированной длины **CHAR**, поскольку для **VARCHAR** используется столько места, сколько действительно необходимо;
- с другой стороны, поскольку это строки переменной длины, они способны увеличиваться при обновлении, что требует дополнительной работы, может влиять на производительность;
- при обновлении строки могут уменьшаться, приводит к фрагментации данных.

Типы данных для строк

Тип данных	Количество байт	Макс. размер	Описание
TEXT **BLOB**	k+2, k – длина текста	2^{16}-1 байт	Это **VARCHAR** большого размера, позволяют хранить большие объемы текста. Причем тип **TEXT** используется для хранения именно текста, а **BLOB** - для хранения двоичных данных: изображений, звука, электронных документов и т.д.
MEDIUMTEXT **MEDIUMBLOB**	k+3	2^{24}-1 байт	Аналогично предыдущему, но с большим размером.
LONGTEXT **LONGBLOB**	k+4	2^{32}-1 байт	Аналогично предыдущему, но с большим размером.

Типы данных для строк

Особенности типов **TEXT** и **BLOB**:

- различие между **BLOB** и **TEXT** заключается в том, что типы **BLOB** хранят двоичные данные без учета схемы упорядочения и кодировки, а типы **TEXT** используют схемы упорядочения и кодировку;
- MySQL сортирует столбцы **BLOB** и **TEXT** иначе, чем столбцы других типов: вместо сортировки строки по всей ее длине она сортирует только по первым **max_sort_length** байтам каждого столбца.

Типы данных для строк

Рекомендации по выбору:

- Использовать принцип «чем меньше, тем лучше»;
- тип **CHAR** полезен, когда требуется сохранять очень короткие строки или все значения имеют приблизительно одинаковую длину;
- тип **CHAR** имеет преимущество над **VARCHAR** при часто меняющихся данных, поскольку строка фиксированной длины не подвержена фрагментации;
- целесообразно использовать тип **VARCHAR**, если максимальная длина строки в столбце значительно больше средней, обновление поля выполняется редко.

Тип данных ENUM

Иногда вместо строковых типов можно задействовать тип **ENUM**.

Столбец **ENUM** может хранить предопределенный набор различных строковых значений.

MySQL сохраняет их очень компактно, упаковывая в 1 или 2 байта в зависимости от количества значений в списке.

В базе данных значение типа **ENUM** хранится как целое число, отражающее позицию его значения в списке значений поля, и сохраняет справочную таблицу, которая определяет соответствие между числом и строкой.

Тип данных ENUM

Пример.

```
CREATE TABLE  test(
      genre ENUM ("Роман", "Поэзия", "Приключения", "Фантастика")
);

 INSERT INTO test
 VALUES ("Поэзия"), ("Роман"), ("Фантастика");

SELECT genre, genre + 0
FROM test;
```

Тип данных ENUM

Пример.

```
CREATE TABLE  test(
      genre ENUM ("Роман", "Поэзия", "Приключения", "Фантастика")
);

 INSERT INTO test
 VALUES ("Поэзия"), ("Роман"), ("Фантастика");

SELECT genre, genre + 0
FROM test;
```

```
Affected rows: 0

Affected rows: 3

Query result:
+------------+-----------+
| genre      | genre + 0 |
+------------+-----------+
| Поэзия     | 2.0       |
| Роман      | 1.0       |
| Фантастика | 4.0       |
+------------+-----------+
Affected rows: 3
```

Типы данных для даты и времени

Тип данных	Количество байт	Диапазон	Описание
DATE	3	от **'1000-01-01'** до **'9999-12-31'**	Предназначен для хранения даты. В качестве первого значения указывается год в формате "**YYYY**", через дефис - месяц в формате "**MM**", а затем день в формате "**DD**". В качестве разделителя может выступать не только дефис, а любой символ отличный от цифры.
TIME	3	от **'-838:59:59'** до **'838:59:59'**	Предназначен для хранения времени суток. Значение вводится и хранится в привычном формате - **hh:mm:ss**, где **hh** - часы, **mm** - минуты, **ss** - секунды. В качестве разделителя может выступать любой символ отличный от цифры.
DATETIME	8	от **'1000-01-01 00:00:00'** до **'9999-12-31 23:59:59'**	Предназначен для хранения и даты и времени суток. Значение вводится и хранится в формате - **YYYY-MM-DD hh:mm:ss**. В качестве разделителей могут выступать любые символы отличные от цифры.

Типы данных для даты и времени

Тип данных	Количество байт	Диапазон	Описание
TIMESTAMP	**4**	от **'1970-01-01 00:00:00'** до **'2037-12-31 23:59:59'**	Предназначен для хранения даты и времени суток в виде количества секунд, прошедших с полуночи 1 января 1970 года (начало эпохи UNIX).
YEAR(m)	**1**	от **1970** до **2069** для **m=2** и от **1901** до **2155** для **m=4**	Предназначен для хранения года. **m** - задает формат года. Например, **YEAR (2)** - 70, а **YEAR (4)** - 1970. Если параметр **m** не указан, то по умолчанию считается, что он равен **4**.

Типы данных для даты и времени

Особенности типа **TIMESTAMP** и **DATETIME**:

- можно настроить поведение при вставке и обновлении для каждого столбца этих типов, например, чтобы вставлялось текущее время:

```
DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
```

Типы данных даты и времени

Рекомендации по выбору:

- использовать принцип «чем меньше, тем лучше»;
- выбирать тип **`TIMESTAMP`**, если это возможно, поскольку с точки зрения занимаемого на диске места он намного эффективнее, чем **`DATETIME`**.

Создание пустой таблицы

```
CREATE TABLE имя_таблицы(
    столбец_1 тип_столбца характеристики,
    столбец_2 тип_столбца характеристики,
    …,
    столбец_N тип_столбца характеристики
);
```

Характеристики, описание столбцов

Характеристики:

- **NULL / NOT NULL** – разрешает / запрещает пустое значение в столбце (по умолчанию пустое значение разрешено);
- **DEFAULT** - значение по умолчанию (если не указано, то в поле заносится пустое значение, если это разрешено).

Характеристика, NULL / NOT NULL

NULL - это системное значение, которое занимает один байт памяти и указывает, что значение отсутствует, в отличие от пробела , нуля или любого другого значения.

Поле в базе данных, содержащее значение **NULL**, означает, что содержимое этой ячейки неизвестно на момент ее просмотра.

Характеристика, NULL / NOT NULL

По умолчанию столбец таблицы может содержать пустое значение.

Для того чтобы запретить пустые значения – необходимо установить опцию **NOT NULL**:

```
amount INT NOT NULL
```

Столбец, который допускает значение **NULL**, также позволяет вставлять строки без каких-либо значений в этом столбце.

Характеристика, значение по умолчанию

Для столбца может быть задано значение по умолчанию, т.е. значение, которое будет подставляться в том случае, когда оператор вставки не предоставляет значения для этого столбца. После описания типа столбца необходимо указать:

```
DEFAULT значение
```

Как правило, значением по умолчанию выбирается наиболее часто встречающееся в столбце значение.

Характеристика, значение по умолчанию

Особенности использования значения по умолчанию:

- значение по умолчанию должно соответствовать определенному для столбца типу данных, при этом, если это возможно, происходит преобразование заданного значения по умолчанию к нужному типу:

```
amount INT DEFAULT 1
amount INT DEFAULT 1.0
amount INT DEFAULT "1"
amount INT DEFAULT "abc" – ошибка
```

Характеристика, значение по умолчанию

Особенности использования значения по умолчанию:

- столбцам типа **TEXT** и **BLOB** начальное значение установить нельзя;
- для некоторых типов данных в качестве начального значения можно использовать функции без параметров, например, для **TIMESTAMP** и **DATETIME**:

  CURRENT_TIMESTAMP,

  которая вставляет текущую дату и время

Сравнение NULL и значений по умолчанию

Сравнение значения NULL и значений по умолчанию:

- **NULL** не имеет типа данных, поэтому может быть вставлено в любую структуру данных и любой столбец базы данных, значения по умолчанию должны иметь определенный тип данных.

Сравнение NULL и значений по умолчанию

Сравнение значения NULL и значений по умолчанию:

- **NULL** часто используется в столбцах, где значение является необязательным. Так как значение NULL занимает только 1 байт, они могут быть полезны при оптимизации базы данных (ели эти данные не нужно часто сравнивать и сортировать). Использование этих значений намного эффективнее, чем значения по умолчанию, которое занимает как минимум 1 байт.

Сравнение NULL и значений по умолчанию

Сравнение значения NULL и значений по умолчанию:

- назначение **NOT NULL** столбцам также может помочь в проверке таблицы, так как по этому критерию данные должны быть обязательно вставлены;
- значения по умолчанию необходимо применять, если столбец используется в индексе, часто сортируется или участвует в операторах сравнения.

Предметная область «Книжный склад»

Таблица **book**

book_id	Title	author	Price	amount
1	Мастер и Маргарита	Булгаков М.А.	670.99	3
2	Белая гвардия	Булгаков М.А.	540.50	5
3	Идиот	Достоевский Ф.М.	460.00	10
4	Братья Карамазовы	Достоевский Ф.М.	799.01	2
5	Стихотворения и поэмы	Есенин С.А.	650.00	15

Предметная область «Книжный склад»

SMALLINT	2 байта	от -32768 до 32757
MEDIUMINT	3 байта	от -8388608 до 8388607
INT	4 байта	от -2147483648 до 2147483647

book_id			Price	amount
1	Мастер и Маргарита	Булгаков М.А.	670.99	3
2	Белая гвардия	Булгаков М.А.	540.50	5
3	Идиот	Достоевский Ф.М.	460.00	10
4	Братья Карамазовы	Достоевский Ф.М.	799.01	2
5	Стихотворения и поэмы	Есенин С.А.	650.00	15

Предметная область «Книжный склад»

CHAR(50)	50
VARCHAR(50)	<=50

Таб

book_id	Title	author	Price	amount
INT PRIMARY KEY AUTO_INCREMENT				
1	Мастер и Маргарита	Булгаков М.А.	670.99	3
2	Белая гвардия	Булгаков М.А.	540.50	5
3	Идиот	Достоевский Ф.М.	460.00	10
4	Братья Карамазовы	Достоевский Ф.М.	799.01	2
5	Стихотворения и поэмы	Есенин С.А.	650.00	15

Предметная область «Книжный склад»

CHAR (30)	30
VARCHAR (30)	<=30

Таблица **book**

book_id	Title	author	Price	amount
INT PRIMARY KEY AUTO_INCREMENT	VARCHAR(50)			
1	Мастер и Маргарита	Булгаков М.А.	670.99	3
2	Белая гвардия	Булгаков М.А.	540.50	5
3	Идиот	Достоевский Ф.М.	460.00	10
4	Братья Карамазовы	Достоевский Ф.М.	799.01	2
5	Стихотворения и поэмы	Есенин С.А.	650.00	15

Предметная область «Книжный склад»

DECIMAL(8, 2)	10
FLOAT(8, 2)	4

Таблица **book**

book_id	Title	author	Price	amount
INT PRIMARY KEY AUTO_INCREMENT	VARCHAR(50)	CHAR(30)		
1	Мастер и Маргарита	Булгаков М.А.	670.99	3
2	Белая гвардия	Булгаков М.А.	540.50	5
3	Идиот	Достоевский Ф.М.	460.00	10
4	Братья Карамазовы	Достоевский Ф.М.	799.01	2
5	Стихотворения и поэмы	Есенин С.А.	650.00	15

Предметная область «Книжный склад»

SMALLINT	2 байта	от -32768 до 32757
MEDIUMINT	3 байта	от -8388608 до 8388607
INT	4 байта	от -2147483648 до 2147483647

Таблица b

book_id	Title	author	Price	amount
INT PRIMARY KEY AUTO_INCREMENT	VARCHAR(50)	CHAR(30)	DECIMAL(8,2)	
1	Мастер и Маргарита	Булгаков М.А.	670.99	3
2	Белая гвардия	Булгаков М.А.	540.50	5
3	Идиот	Достоевский Ф.М.	460.00	10
4	Братья Карамазовы	Достоевский Ф.М.	799.01	2
5	Стихотворения и поэмы	Есенин С.А.	650.00	15

Предметная область «Книжный склад»

Таблица **book**

book_id	Title	author	Price	amount
INT PRIMARY KEY AUTO_INCREMENT	VARCHAR(50)	CHAR(30)	DECIMAL(8,2)	SMALLINT UNSIGNED
1	Мастер и Маргарита	Булгаков М.А.	670.99	3
2	Белая гвардия	Булгаков М.А.	540.50	5
3	Идиот	Достоевский Ф.М.	460.00	10
4	Братья Карамазовы	Достоевский Ф.М.	799.01	2
5	Стихотворения и поэмы	Есенин С.А.	650.00	15

Предметная область «Книжный склад»

Таблица **book**

Максимальное количество байт на одну строку:

4 + (50 + 1) + 30 +(8+2) + 2 = 98 байт ≈ 100 байт

Количество байт на 1 000 000 записей:

100 * 1 000 000 = 100 000 000 байт = 100 Мегабайт

Создание пустой таблицы

```
CREATE TABLE book(
    book_id INT PRIMARY KEY AUTO_INCREMENT,
    title VARCHAR(50),
    author CHAR(30),
    price DECIMAL(8, 2),
    amount SMALLINT UNSIGNED
);
```

Добавление записей в таблицу

Для вставки новых записей в таблицу используется SQL запрос **INSERT**.

Два способа вставки записей:

- вставка констант-значений в каждое поле;
- вставка новых записей, «собранных» из других таблиц базы данных.

Добавление записей в таблицу

Добавление одной записи:

```
INSERT INTO таблица
     (столбец_1, ..., столбец_N)
VALUES
     (значение_1, ..., значение_N);
```

Добавление записей в таблицу

Добавление нескольких записей:

```
INSERT INTO таблица
     (столбец_1, ..., столбец_N)
VALUES
     (значение_1_1, ..., значение_N_1),
      ...,
     (значение_M_1, ..., значение_M_1);
```

Добавление записей в таблицу

Правила соответствия списка полей и списка значений:

- количество полей и количество значений в списках должны совпадать;
- должно существовать прямое соответствие между позицией одного и того же элемента в обоих списках;
- типы данных элементов в списке значений должны быть совместимы с типами данных соответствующих столбцов таблицы.

Добавление записей в таблицу

Особенности выполнения запроса **INSERT**, если в списке полей пропущен столбец:

- в столбец, для которого установлено значение по умолчанию, вставляется это значение;
- в столбец, для которого не установлено **NOT NULL**, будет занесено значение **NULL**;
- если не установлено значение по умолчанию, но установлена опция **NOT NULL**, будет выдана ошибка.

Добавление записей в таблицу

Задание. Занести новую запись в таблицу **book**: книгу Мастер и Маргарита, автора Булгаков М.А., по цене 670.99 и в количестве 3 экземпляра.

```
INSERT INTO book
    (title, author, price, amount)
VALUES
    ("Мастер и Маргарита", "Булгаков М.А.",
     670.99, 3);
```

Добавление записей в таблицу

Не рекомендуется добавлять значения в поля, объявленные как **PRIMARY KEY AUTO_INCREMENT.**

Если же такое поле присутствует в списке, для него указывается значение **NULL**.

```
INSERT INTO book
VALUES
  (NULL, "Мастер и Маргарита", "Булгаков М.А.",
  670.99, 3);
```

Добавление в таблицу на основе запроса

Если в таблицу нужно занести данные из других таблиц, то сначала отбираются нужные данные и таблиц базы данных запросом **SELECT**, затем эти данные добавляются в таблицу.

При этом количество столбцов в **SELECT** и их типы должны соответствовать столбцам таблицы, указанных в запросе **INSERT**.

Добавление в таблицу на основе запроса

```
INSERT INTO таблица
     (столбец_1, ..., столбец_N)
SELECT
      выражение_1, ..., выражение_N
FROM таблица_другая
WHERE условие_1
GROUP BY список столбцов_1
HAVING условие_2
ORDER BY список_столбцов_2
```

Этот блок - необязателен

Добавление в таблицу на основе запроса

Задание. Дана таблица **supply** следующей структуры:

supply_id	title	author	price	amount
1	Лирика	Пастернак Б.Л.	518.99	2
2	Черный человек	Есенин С.А.	570.20	6
3	Белая гвардия	Булгаков М.А.	540.50	7
4	Идиот	Достоевский Ф.М.	360.80	3

Занести из таблицы **supply** в таблицу **book** все книги, кроме книг написанных Булгаковым и Достоевским.

Добавление в таблицу на основе запроса

Задание. Занести из таблицы **supply** в таблицу **book** все книги, кроме книг написанных Булгаковым и Достоевским.

Алгоритм:

- отобрать из таблицы **supply** все книги, кроме книг Достоевского и Булгакова;
- добавить отобранные записи в таблицу **book**.

Добавление в таблицу на основе запроса

Задание. Занести из таблицы **supply** в таблицу **book** все книги, кроме книг написанных Булгаковым и Достоевским.

Шаг 1. Отобрать из таблицы **supply** все книги, кроме книг Достоевского и Булгакова.

```
SELECT title, author, price, amount
FROM supply
WHERE author NOT IN
      ("Булгаков М.А.", "Достоевский Ф.М.")
```

Добавление в таблицу на основе запроса

Задание. Занести из таблицы **supply** в таблицу **book** все книги, кроме книг написанных Булгаковым и Достоевским.

Шаг 2. Вставить отобранные записи в таблицу **book**.

```
INSERT INTO book
       (title, author, price, amount)
SELECT title, author, price, amount
FROM supply
WHERE author NOT IN
     ("Булгаков М.А.", "Достоевский Ф.М.");
```

Добавление в таблицу на основе запроса

Задание. Занести из таблицы **supply** в таблицу **book** все книги, кроме книг написанных Булгаковым и Достоевским.

book_id	title	author	price	amount
1	Мастер и Маргарита	Булгаков М.А.	670.99	3
2	Белая гвардия	Булгаков М.А.	540.50	5
3	Идиот	Достоевский Ф.М.	460.00	10
4	Братья Карамазовы	Достоевский Ф.М.	799.01	2
5	Стихотворения и поэмы	Есенин С.А.	650.00	15

Affected rows: 5

Query result:

book_id	title	author	price	amount
1	Мастер и Маргарита	Булгаков М.А.	670.99	3
2	Белая гвардия	Булгаков М.А.	540.50	5
3	Идиот	Достоевский Ф.М.	460.00	10
4	Братья Карамазовы	Достоевский Ф.М.	799.01	2
5	Стихотворения и поэмы	Есенин С.А.	650.00	15
6	Лирика	Пастернак Б.Л.	518.99	2
7	Черный человек	Есенин С.А.	570.20	6

Affected rows: 7

Добавление в таблицу на основе запроса

В запросах на добавление можно использовать вложенные запросы.

Добавление в таблицу на основе запроса

Задание. Занести из таблицы **supply** в таблицу **book** только те книги, которые отсутствуют в таблице **book.**

Алгоритм:

1. отобрать названия книг и их авторов из таблицы **book**;
2. отобрать все книги из таблицы **supply**, у которых названия и авторы не совпадают с отобранными на первом шаге;
3. добавить новые (отобранные на втором шаге) книги из таблицы **supply** в таблицу **book**.

Добавление в таблицу на основе запроса

Задание. Занести из таблицы **supply** в таблицу **book** только те книги, которые отсутствуют в таблице **book.**

Шаг 1. Отобрать названия книг и их авторов из таблицы **book.**

```
SELECT title, author
FROM book
```

```
Query result:
+------------------------+--------------------+
| title                  | author             |
+------------------------+--------------------+
| Мастер и Маргарита     | Булгаков М.А.      |
| Белая гвардия          | Булгаков М.А.      |
| Идиот                  | Достоевский Ф.М.   |
| Братья Карамазовы      | Достоевский Ф.М.   |
| Стихотворения и поэмы  | Есенин С.А.        |
+------------------------+--------------------+
Affected rows: 5
```

Добавление в таблицу на основе запроса

Задание. Занести из таблицы **supply** в таблицу **book** только те книги, которые отсутствуют в таблице **book.**

Шаг 2. Отобрать новые книги из таблицы **supply.**

```
SELECT title, author, price, amount
FROM supply
WHERE (title, author)
        NOT IN (список_название_книга_из_book
                );
```

Добавление в таблицу на основе запроса

Задание. Занести из таблицы **supply** в таблицу **book** только те книги, которые отсутствуют в таблице **book.**

Шаг 2. Отобрать новые книги из таблицы **supply.**

```
SELECT title, author, price, amount
FROM supply
WHERE (title, author)
       NOT IN (SELECT title, author
                FROM book
                );
```

Добавление в таблицу на основе запроса

Задание. Занести из таблицы **supply** в таблицу **book** только те книги, которые отсутствуют в таблице **book.**

Шаг 2. Отобрать новые книги из таблицы **supply.**

```
Query result:
+-----------------+-----------------+---------+---------+
| title           | author          | price   | amount  |
+-----------------+-----------------+---------+---------+
| Лирика          | Пастернак Б.Л.  | 518.99  | 2       |
| Черный человек  | Есенин С.А.     | 570.20  | 6       |
+-----------------+-----------------+---------+---------+
Affected rows: 2
```

Добавление в таблицу на основе запроса

Задание. Занести из таблицы **supply** в таблицу **book** только те книги, которые отсутствуют в таблице **book.**

Шаг 3. Добавить новые книги из таблицы **supply** в **book.**

```
INSERT INTO book
       (title, author, price, amount)
SELECT title, author, price, amount
FROM supply
WHERE (title, author)
       NOT IN (SELECT title, author
                FROM book
                );
```

Добавление в таблицу на основе запроса

Задание. Занести из таблицы **supply** в таблицу **book** только те книги, которые отсутствуют в таблице **book.**

Query result:

book_id	title	author	price	amount
1	Мастер и Маргарита	Булгаков М.А.	670.99	3
2	Белая гвардия	Булгаков М.А.	540.50	5
3	Идиот	Достоевский Ф.М.	460.00	10
4	Братья Карамазовы	Достоевский Ф.М.	799.01	2
5	Стихотворения и поэмы	Есенин С.А.	650.00	15

Affected rows: 5

Query result:

book_id	title	author	price	amount
1	Мастер и Маргарита	Булгаков М.А.	670.99	3
2	Белая гвардия	Булгаков М.А.	540.50	5
3	Идиот	Достоевский Ф.М.	460.00	10
4	Братья Карамазовы	Достоевский Ф.М.	799.01	2
5	Стихотворения и поэмы	Есенин С.А.	650.00	15
6	Лирика	Пастернак Б.Л.	518.99	2
7	Черный человек	Есенин С.А.	570.20	6

Affected rows: 7

Добавление в таблицу на основе запроса

В запросах на добавление можно использовать табличные выражения. Синтаксис:

```
INSERT INTO таблица
     (столбец_1, ..., столбец_N)
WITH табличное_выражение(...)
AS (
    ...
)
SELECT ...
FROM табличное_выражение, ...
...
```

Обновление данных

Обновление данных - это изменение значений в уже существующих записях таблицы.

При этом возможно изменить:

- значения одного или нескольких столбцов во всей таблице;
- значения одного или нескольких столбцов в одной или нескольких строках таблицы.

Обновление данных

```
UPDATE таблица
SET поле = выражение
WHERE условие
```

Обновление данных

Задание. Уменьшить на 30% цену тех книг в таблице **book**, количество которых меньше 5.

```
UPDATE book
SET price = 0.7 * price
WHERE amount < 5;
```

Обновление данных

Задание. Уменьшить на 30% цену тех книг в таблице **book**, количество которых меньше 5.

Query result:

book_id	title	author	price	amount
1	Мастер и Маргарита	Булгаков М.А.	670.99	3
2	Белая гвардия	Булгаков М.А.	540.50	5
3	Идиот	Достоевский Ф.М.	460.00	10
4	Братья Карамазовы	Достоевский Ф.М.	799.01	2
5	Стихотворения и поэмы	Есенин С.А.	650.00	15

Affected rows: 5

Query result:

book_id	title	author	price	amount
1	Мастер и Маргарита	Булгаков М.А.	469.69	3
2	Белая гвардия	Булгаков М.А.	540.50	5
3	Идиот	Достоевский Ф.М.	460.00	10
4	Братья Карамазовы	Достоевский Ф.М.	559.31	2
5	Стихотворения и поэмы	Есенин С.А.	650.00	15

Affected rows: 5

Обновление данных

Запросом **UPDATE** можно обновлять значения нескольких столбцов одновременно.

Обновление данных

```
UPDATE таблица
SET поле_1 = выражение_1,
    …,
    поле_N = выражение_N
WHERE условие
```

Обновление данных

Задание. Для каждой книги уменьшить количество ее экземпляров на указанное в столбце **buy** количество, а в столбец **buy** занести 0.

book_id	title	author	price	amount	buy
1	Мастер и Маргарита	Булгаков М.А.	670.99	3	0
2	Белая гвардия	Булгаков М.А.	540.50	5	3
3	Идиот	Достоевский Ф.М.	460.00	10	8
4	Братья Карамазовы	Достоевский Ф.М.	799.01	2	0
5	Стихотворения и поэмы	Есенин С.А.	650.00	15	12

Обновление данных

Задание. Для каждой книги уменьшить количество ее экземпляров на указанное в столбце **buy** количество, а в столбец **buy** занести 0.

```
UPDATE book
SET amount = amount - buy,
    buy = 0
WHERE buy != 0;
```

Обновление данных

Задание. Для каждой книги уменьшить количество ее экземпляров на указанное в столбце **buy** количество, а в столбец **buy** занести 0.

Query result:

title	author	price	amount	buy
Мастер и Маргарита	Булгаков М.А.	670.99	3	0
Белая гвардия	Булгаков М.А.	540.50	5	3
Идиот	Достоевский Ф.М.	460.00	10	8
Братья Карамазовы	Достоевский Ф.М.	799.01	2	0
Стихотворения и поэмы	Есенин С.А.	650.00	15	12

Affected rows: 5

Query result:

title	author	price	amount	buy
Мастер и Маргарита	Булгаков М.А.	670.99	3	0
Белая гвардия	Булгаков М.А.	540.50	2	0
Идиот	Достоевский Ф.М.	460.00	2	0
Братья Карамазовы	Достоевский Ф.М.	799.01	2	0
Стихотворения и поэмы	Есенин С.А.	650.00	3	0

Affected rows: 5

Обновление данных

Запросом **UPDATE** можно обновлять значения нескольких столбцов из нескольких таблиц одновременно.

Обновление данных

```
UPDATE таблица_1, таблица_2
SET поле_1 = выражение_1,
    таблица_1.поле_2
```

если в двух таблицах столбцы
имеют одинаковое имя

Обновление данных

```
UPDATE таблица_1, таблица_2
SET поле_1 = выражение_1,
    таблица_1.поле_2
```

если в двух таблицах столбцы
имеют одинаковое имя

Обновление данных

```
UPDATE таблица_1, таблица_2
SET поле_1 = выражение_1,
    таблица_1.поле_2
```

если в двух таблицах столбцы
имеют одинаковое имя

Обновление данных

```
UPDATE таблица_1, таблица_2
SET поле_1 = выражение_1,
    таблица_1.поле_2 = выражение_2,
    …,
    таблица_2.поле_N = выражение_N
WHERE условие связи между таблицами
```

Обновление данных

Задание. Если в таблице **supply** есть те же книги, что и в таблице **book**, увеличить их количество в таблице **book** на значение столбца **amount** таблицы **supply**.

Обновление данных

Задание. Если в таблице **supply** есть те же книги, что и в таблице **book**, увеличить их количество в таблице **book** на значение столбца **amount** таблицы **supply**.

```
UPDATE book, supply
SET
    book.amount = book.amount + supply.amount
WHERE
    book.title = supply.title AND
    book.author = supply.author
```

Обновление данных

Задание. Если в таблице **supply** есть те же книги, что и в таблице **book**, увеличить их количество в таблице **book** на значение столбца **amount** таблицы **supply**.

```
Query result:
+---------+-----------------------+------------------+--------+--------+
| book_id | title                 | author           | price  | amount |
+---------+-----------------------+------------------+--------+--------+
| 1       | Мастер и Маргарита    | Булгаков М.А.    | 670.99 | 3      |
| 2       | Белая гвардия         | Булгаков М.А.    | 540.50 | 5      |
| 3       | Идиот                 | Достоевский Ф.М. | 460.00 | 10     |
| 4       | Братья Карамазовы     | Достоевский Ф.М. | 799.01 | 2      |
| 5       | Стихотворения и поэмы | Есенин С.А.      | 650.00 | 15     |
+---------+-----------------------+------------------+--------+--------+
Affected rows: 5

Query result:
+-----------+----------------+------------------+--------+--------+
| supply_id | title          | author           | price  | amount |
+-----------+----------------+------------------+--------+--------+
| 1         | Лирика         | Пастернак Б.Л.   | 518.99 | 2      |
| 2         | Черный человек | Есенин С.А.      | 570.20 | 6      |
| 3         | Белая гвардия  | Булгаков М.А.    | 540.50 | 7      |
| 4         | Идиот          | Достоевский Ф.М. | 360.80 | 3      |
+-----------+----------------+------------------+--------+--------+
Affected rows: 4
```

Обновление данных

Задание. Если в таблице **supply** есть те же книги, что и в таблице **book**, увеличить их количество в таблице **book** на значение столбца **amount** таблицы **supply**.

```
Query result:
+---------+-----------------------+------------------+--------+--------+
| book_id | title                 | author           | price  | amount |
+---------+-----------------------+------------------+--------+--------+
| 1       | Мастер и Маргарита    | Булгаков М.А.    | 670.99 | 3      |
| 2       | Белая гвардия         | Булгаков М.А.    | 540.50 | 12     |
| 3       | Идиот                 | Достоевский Ф.М. | 460.00 | 13     |
| 4       | Братья Карамазовы     | Достоевский Ф.М. | 799.01 | 2      |
| 5       | Стихотворения и поэмы | Есенин С.А.      | 650.00 | 15     |
+---------+-----------------------+------------------+--------+--------+
Affected rows: 5
```

Табличные выражения, обновление данных

В запросах на обновление можно использовать табличные выражения. Синтаксис:

```
WITH табличное_выражение(...)
AS (
    ...
)
UPDATE ..., табличное_выражение
SET ...
...
```

Удаление данных

Запрос на удаление позволяет удалить одну или несколько записей из таблицы.

Удаление данных

```
DELETE FROM таблица
WHERE условие;
```

Удаление данных

Задание. Удалить из таблицы **supply** все книги, названия которых есть в таблице **book**.

Алгоритм:

- получить все различные названия из таблицы **book**;
- удалить из **supply** книги, названия которых получены на первом шаге.

Удаление данных

Задание. Удалить из таблицы **supply** все книги, названия которых есть в таблице **book**.

Шаг 1. Вывести все различные книги из таблицы **book**.

```
SELECT DISTINCT title
FROM book
```

```
Query result:
+-----------------------+
| title                 |
+-----------------------+
| Мастер и Маргарита    |
| Белая гвардия         |
| Идиот                 |
| Братья Карамазовы     |
| Стихотворения и поэмы |
+-----------------------+
Affected rows: 5
```

Удаление данных

Задание. Удалить из таблицы **supply** все книги, названия которых есть в таблице **book**.

Шаг 2. Удалить из **supply** книги, которые есть в **book**.

```
DELETE FROM supply
WHERE title
      IN (SELECT DISTINCT title
          FROM book
          );
```

Удаление данных

Задание. Удалить из таблицы **supply** все книги, названия которых есть в таблице **book**.

```
Query result:
+-----------+-----------------+-------------------+--------+--------+
| supply_id | title           | author            | price  | amount |
+-----------+-----------------+-------------------+--------+--------+
| 1         | Лирика          | Пастернак Б.Л.    | 518.99 | 2      |
| 2         | Черный человек  | Есенин С.А.       | 570.20 | 6      |
| 3         | Белая гвардия   | Булгаков М.А.     | 540.50 | 7      |
| 4         | Идиот           | Достоевский Ф.М.  | 360.80 | 3      |
+-----------+-----------------+-------------------+--------+--------+
Affected rows: 4
```

```
Query result:
+-----------+-----------------+-----------------+--------+--------+
| supply_id | title           | author          | price  | amount |
+-----------+-----------------+-----------------+--------+--------+
| 1         | Лирика          | Пастернак Б.Л.  | 518.99 | 2      |
| 2         | Черный человек  | Есенин С.А.     | 570.20 | 6      |
+-----------+-----------------+-----------------+--------+--------+
Affected rows: 2
```

Табличные выражения, удаление данных

В запросах на удаление можно использовать табличные выражения. Синтаксис:

```
WITH табличное_выражение(...)
AS (
    ...
)
DELETE таблица
USING ..., табличное_выражение
WHERE ...
```

Создание таблицы

Новая таблица может быть **создана** на основе данных из другой таблицы.

Для этого используется запрос **SELECT**, результирующая таблица которого и будет новой таблицей базы данных.

При этом имена столбцов запроса становятся именами столбцов новой таблицы.

Создание таблицы

```
CREATE TABLE
     имя_таблицы AS
SELECT ...
```

Создание таблицы

Задание. Создать таблицу заказ (**ordering**), куда включить авторов и названия тех книг, количество которых в таблице **book** меньше 4. Для всех книг указать одинаковое количество 5.

Алгоритм:

1. отобрать нужные книги из таблицы **book;**
2. создать таблицу на основе результата первого шага.

Создание таблицы

Задание. Создать таблицу заказ (**ordering**), куда включить авторов и названия тех книг, количество которых в таблице **book** меньше 4. Для всех книг указать одинаковое количество 5.

Шаг 1. Отобрать книги, количество которых меньше 4.

```
SELECT author, title, 5 AS amount
FROM book
WHERE amount < 4
```

```
Query result:
+------------------+--------------------+--------+
| author           | title              | amount |
+------------------+--------------------+--------+
| Булгаков М.А.    | Мастер и Маргарита | 5      |
| Достоевский Ф.М. | Братья Карамазовы  | 5      |
+------------------+--------------------+--------+
Affected rows: 2
```

Создание таблицы

Задание. Создать таблицу заказ (**ordering**), куда включить авторов и названия тех книг, количество которых в таблице **book** меньше 4. Для всех книг указать одинаковое количество 5.

Шаг 2. Создать таблицу на основе отобранных записей.

```
CREATE TABLE
     ordering AS
SELECT author, title, 5 AS amount
FROM book
WHERE amount < 4
```

Создание таблицы

Задание. Создать таблицу заказ (**ordering**), куда включить авторов и названия тех книг, количество которых в таблице **book** меньше 4. Для всех книг указать одинаковое количество 5.

```
Query result:
+-------------------+----------------------+---------+
| author            | title                | amount  |
+-------------------+----------------------+---------+
| Булгаков М.А.     | Мастер и Маргарита   | 5       |
| Достоевский Ф.М.  | Братья Карамазовы    | 5       |
+-------------------+----------------------+---------+
Affected rows: 2
```

Табличные выражения, создание таблицы

В запросах на создание таблицы можно использовать табличные выражения. Синтаксис:

```
CREATE TABLE таблица AS
WITH табличное_выражение(...)
AS (
    ...
)
SELECT ...
FROM ..., табличное_выражение
...
```

Резюме

С помощью SQL запросов можно вносить изменения в таблицы базы данных, создавать новые таблицы, удалять существующие.

Резюме

Вставка записей в таблицу осуществляется с помощью запроса **INSERT**. Этот запрос поддерживает как вставку значений непосредственно в таблицу, так и вставку одной или нескольких записей, отобранных из уже существующих таблиц.

Резюме

Обновление записей осуществляется с помощью запроса **UPDATE**. С его помощью можно обновлять одновременно несколько полей одной или нескольких таблиц.

Резюме

С помощью запроса **DELETE** можно **удалить** одну или несколько записей из одной таблицы базы данных

Резюме

Создание таблицы осуществляется с помощью запроса **CREATE**. Этот запрос позволяет описать структуру таблицы (поля, их типы и другие опции), а также создать новую таблицу на основе информации из базы данных.

Резюме

Во всех запросах корректировки данных можно использовать **вложенные запросы** и **табличные выражения**

Задание

«Интерактивный тренажер по SQL»:

- модуль 1, уроки 5, 7

«Расширенные возможности SQL»:

- модуль 1, урок 7

Спасибо за внимание!