<https://habr.com/ru/post/564390/#команды-sql>

**Общая структура запроса**

**SELECT** ('столбцы или \* для выбора всех столбцов; обязательно')

**FROM** ('таблица; обязательно')

**WHERE** ('условие/фильтрация, например, city = 'Moscow'; необязательно')

**GROUP** **BY** ('столбец, по которому хотим сгруппировать данные; необязательно')

**HAVING** ('условие/фильтрация на уровне сгруппированных данных; необязательно')

**ORDER** **BY** ('столбец, по которому хотим отсортировать вывод; необязательно')

|  |
| --- |
| **SELECT, FROM** — обязательные элементы запроса, которые определяют выбранные столбцы, их порядок и источник данных. |
| **WHERE –** необязательный элемент запроса, который используется, когда нужно отфильтровать данные по нужному условию. Очень часто внутри элемента where используются IN / NOT IN для фильтрации столбца по нескольким значениям, AND / OR для фильтрации таблицы по нескольким столбцам.  **SELECT** \* **FROM** Customers  **WHERE** Country = 'Germany' **AND** City **NOT IN** ('Berlin', 'Aachen') **AND** CustID > 15  SQL использует специальные операторы: **IN, BETWEEN, LIKE и IS NULL.**  **IN** – Для списков  **BETWEEN** ‘A’ **AND** ‘G’  **LIKE** или **REGEX** (смотри регулярные операторы)  \_ - одиночный символ  % - любое количество символов  [] – внутри можно указать последовательность символов  **ESCAPE** ‘!’ все что стоит после этого символа будет учитываться  Вывести из таблицы Users имена пользователей, чей прогресс составляет 44% или 45%.   |  |  | | --- | --- | |  | **SELECT** name  **FROM** Users  **WHERE** name **LIKE** '4[45]!%' **ESCAPE** '!'  **NOT** city **IS NULL** | |
| **GROUP BY**  **GROUP BY** — необязательный элемент запроса, с помощью которого можно задать агрегацию по нужному столбцу (например, если нужно узнать какое количество клиентов живет в каждом из городов).  При использовании **GROUP BY** обязательно:   1. перечень столбцов, по которым делается разрез, был одинаковым внутри **SELECT** и внутри **GROUP** **BY**, 2. агрегатные функции **(SUM, AVG, COUNT, MAX, MIN)** должны быть также указаны внутри **SELECT** с указанием столбца, к которому такая функция применяется.   Группировка количества клиентов по стране и городу:  **SELECT** Country, City, **count**(CustomerID) **FROM** Customers  **GROUP** **BY** Country, City |
| **HAVING**  **HAVING** — необязательный элемент запроса, который отвечает за фильтрацию на уровне сгруппированных данных (по сути, **WHERE**, но только на уровень выше).  **select** City, **count**(CustomerID) **as** number\_of\_clients **from** Customers  **WHERE** CustomerName **not** **in** ('Around the Horn','Drachenblut Delikatessend')  **group** **by** City  **HAVING** number\_of\_clients >= 5 |
| Обратная сортировка по одному столбцу и сортировка по умолчанию по второму  **select** \* **from** Customers  **order** **by** Country **DESC**, City |
| **JOIN**  **JOIN** — необязательный элемент, используется для объединения таблиц по ключу, который присутствует в обеих таблицах. Перед ключом ставится оператор **ON**.  **select** \* **from** Orders  **JOIN** Customers **ON** Orders.CustomerID = Customers.CustomerID  https://i.pinimg.com/originals/8a/fb/a5/8afba5dabf68de29e78bd1b528ddb3ec.png |

**Вложенные запросы**

SELECT поля\_таблиц

FROM список\_таблиц

WHERE конкретное\_поле IN (

SELECT поле\_таблицы FROM таблица)

|  |
| --- |
| ANY — ключевое слово, которое должно следовать за операцией сравнения (>, <, <>, = и т.д.), возвращающее TRUE, если хотя бы одно из значений столбца подзапроса удовлетворяет обозначенному условию.  SELECT поля\_таблицы\_1  FROM таблица\_1  WHERE поле\_таблицы\_1 <= ANY (SELECT поле\_таблицы\_2 FROM таблица\_2); |
| ALL — ключевое слово, которое должно следовать за операцией сравнения, возвращающее TRUE, если все значения столбца подзапроса удовлетворяет обозначенному условию.  SELECT поля\_таблицы\_1  FROM таблица\_1  WHERE поле\_таблицы\_1 > ALL (SELECT поле\_таблицы\_2 FROM таблица\_2); |
| IN — ключевое слово, являющееся псевдонимом ключевому слову ANY с оператором сравнения = (эквивалентность), либо <> ALL для NOT IN. Например, следующие запросы равнозначны:  WHERE поле\_таблицы\_1 = ANY (SELECT поле\_таблицы\_2 FROM таблица\_2);  WHERE поле\_таблицы\_1 IN (SELECT поле\_таблицы\_2 FROM таблица\_2); |
| Строковые подзапросы — это подзапрос, возвращающий единственную строку с более чем одной колонкой. Например, следующий запрос получает в подзапросе единственную строку, после чего по порядку попарно сравнивает полученные значения со значениями во внешнем запросе.  SELECT поля\_таблицы\_1  FROM таблица\_1  WHERE (первое\_поле\_таблицы\_1, второе\_поле\_таблицы\_1) =  (  SELECT первое\_поле\_таблицы\_2, второе\_поле\_таблицы\_2  FROM таблица\_2  WHERE id = 10  ); |
| Связанные подзапросы - Внутренний ссылается на внешний  SELECT поля\_таблицы\_1 FROM таблица\_1  WHERE поле\_таблицы\_1 IN  (  SELECT поле\_таблицы\_2 FROM таблица\_2  WHERE таблица\_2.поле\_таблицы\_2 = таблица\_1.поле\_таблицы\_1  ); |

**Нормализация БД**

Нормализация — это процесс эффективной организации данных в БД. Существует две главных причины, обуславливающих необходимость нормализации:

* предотвращение записи в БД лишних данных, например, хранения одинаковых данных в разных таблицах
* обеспечение "оправданной" связи между данными

|  |
| --- |
| **1 нф**  Первая нормальная форма гласит, что таблица базы данных – это представление сущности вашей системы, которую вы создаете. Примеры сущностей: заказы, клиенты, заказ билетов, отель, товар и т.д. Каждая запись в базе данных представляет один экземпляр сущности. Например, в таблице клиентов каждая запись представляет одного клиента. **Первичный ключ.** Правило: каждая таблица имеет первичный ключ, состоящий из наименьшего возможного количества полей.**Атомарность.** Правило: поля не имеют дубликатов в каждой записи и каждое поле содержит только одно значение.**Порядок записей не должен иметь значение.** Правило: порядок записей таблицы не должен иметь значения. |
| **2 нф**  Боремся с избыточностью данных  Столбец зависит от первичного ключа, такое надо убирать |
| **3 нф**  Столбец зависит от другого столбца, такое надо убирать |

|  |  |
| --- | --- |
| **Создание БД**  **CREATE** **DATABASE** **IF** **NOT** **EXISTS** dbName;  **DROP** **DATABASE** **IF** **EXISTS** dbName;  **Выбор БД**  **USE** dbName;  **Создание уникального индекса**  **CREATE** **UNIQUE** **INDEX** indexName  **ON** tableName (colName);  **DROP** **INDEX** indexName; | **Создание таблицы**  **CREATE** **TABLE** employees  ( employee\_number **int NOT NULL**,  employee\_name **char(50) NOT NULL**,  department\_id **int**,  salary **int**,  **CONSTRAINT** employees\_pk **PRIMARY KEY** (employee\_number),  **CONSTRAINT** fk\_departments  **FOREIGN KEY** (department\_id)  **REFERENCES** departments(department\_id)  ); |
| **Добавление записей**  **INSERT** **INTO** tableName (col1, col2, ...colN)  **VALUES**  (val1, val2, ...valN)  (val1, val2, ...valN);  (val1, val2, ...valN);  **Удаление всех записей**  **TRUNCATE** **TABLE** tableName; | **Заполнение таблицы с помощью другой таблицы**  **INSERT** **INTO** tableName [(col1, col2, ...colN)]  **SELECT** col1, col2, ...colN  **FROM** anotherTable  [**WHERE** condition]; |
| **Изменение конкретного поля**  **UPDATE** **users**  **SET** age = 30  **WHERE** username = 'Igor'; | **Удаление строк**  **DELETE** **FROM** **users**  **WHERE** **status** = 'inactive'; |

|  |
| --- |
| **Ограничения при создании ТАБЛИЦ**   * NOT NULL — колонка не может иметь нулевое значение * DEFAULT — значение колонки по умолчанию * UNIQUE — все значения колонки должны быть уникальными * PRIMARY KEY — первичный или основной ключ, уникальный идентификатор записи в текущей таблице * FOREIGN KEY — внешний ключ, уникальный идентификатор записи в другой таблице (таблице, связанной с текущей) * CHECK — все значения в колонке должны удовлетворять определенному условию * INDEX — быстрая запись и извлечение данных |
| **ALTER** **TABLE**  -- добавление новой колонки  **ALTER** **TABLE** tableName **ADD** colName datatype;  -- удаление колонки  **ALTER** **TABLE** tableName **DROP** **COLUMN** colName;  -- изменение типа данных колонки  **ALTER** **TABLE** tableName **MODIFY** **COLUMN** colName newDatatype;  -- добавление ограничения `NOT NULL`  **ALTER** **TABLE** tableName **MODIFY** colName datatype **NOT** NULL;  -- добавление ограничения `UNIQUE`  **ALTER** **TABLE** tableName  **ADD** **CONSTRAINT** myUniqueConstraint **UNIQUE** (col1, col2, ...colN);  -- добавление ограничения `CHECK`  **ALTER** **TABLE** tableName  **ADD** **CONSTRAINT** myUniqueConstraint **CHECK** (condition);  -- добавление первичного ключа  **ALTER** **TABLE** tableName  **ADD** **CONSTRAINT** myPrimaryKey PRIMARY **KEY** (col1, col2, ...colN);  -- удаление ограничения  **ALTER** **TABLE** tableName  **DROP** **CONSTRAINT** myUniqueContsraint;  -- mysql  **ALTER** **TABLE** tableName  **DROP** **INDEX** myUniqueContsraint;  -- удаление первичного ключа  **ALTER** **TABLE** tableName  **DROP** **CONSTRAINT** myPrimaryKey;  -- mysql  **ALTER** **TABLE** tableName  **DROP** PRIMARY **KEY**; |
| **CREATE** **VIEW** usersView **AS**  **SELECT** userName, age  **FROM** **users**  **WHERE** age **IS** **NOT** NULL  **WITH** **CHECK** **OPTION**;  **WITH CHECK OPTION** — это настройка инструкции CREATE VIEW. Она позволяет обеспечить соответствие всех UPDATE и INSERT условию, определенном в представлении. |
| Обновление представления  Представление может быть обновлено при соблюдении следующих условий:   * SELECT не содержит ключевого слова DISTINCT * SELECT не содержит агрегирующих функций * SELECT не содержит функций установки значений * SELECT не содержит операций установки значений * SELECT не содержит предложения ORDER BY * FROM не содержит больше одной таблицы * WHERE не содержит подзапросы * запрос не содержит GROUP BY или HAVING * вычисляемые колонки не обновляются * все ненулевые колонки из базовой таблицы включены в представление в том же порядке, в каком они указаны в запросе INSERT   **UPDATE** usersView  **SET** age = 31  **WHERE** userName = 'Igor';  **DELETE** **FROM** usersView  **WHERE** age = 26; |

**Транзакции**

Транзакция — это единица работы или операции, выполняемой над БД. Это последовательность операций, выполняемых в логическом порядке. Эти операции могут запускаться как пользователем, так и какой-либо программой, функционирующей в БД.

Транзакция — это применение одного или более изменения к БД. Например, при создании/обновлении/удалении записи мы выполняем транзакцию. Важно контролировать выполнение таких операций в целях обеспечения согласованности данных и обработки возможных ошибок.

На практике, запросы, как правило, не отправляются в БД по одному, они группируются и выполняются как часть транзакции.

**Свойства транзакции**

Транзакции имеют 4 стандартных свойства (ACID):

* атомарность (atomicity) — все операции транзакции должны быть успешно завершены. В противном случае, транзакция прерывается, а все изменения отменяются (происходит откат к предыдущему состоянию)
* согласованность (consistency) — состояние должно изменться в полном соответствии с операциями транзакции
* изоляция или автономность (isolation) — транзакции не зависят друг от друга и не оказывают друг на друга никакого влияния
* долговечность (durability) — результат звершенной транзакции должен сохраняться при поломке системы

Для управления транзакцией используются следующие команды:

* BEGIN|START TRANSACTION — запуск транзакции
* COMMIT — сохранение изменений
* ROLLBACK — отмена изменений
* SAVEPOINT — контрольная точка для отмены изменений
* SET TRANSACTION — установка характеристик текущей транзакции

Выполняем три запроса на удаление данных из users, создавая контрольные точки перед каждый удалением:

**START** **TRANSACTION**

**SAVEPOINT** sp1;

**DELETE** **FROM** **users**

**WHERE** age = 26;

**SAVEPOINT** sp2;

**DELETE** **FROM** **users**

**WHERE** userName = 'Oleg';

**SAVEPOINT** sp3;

**DELETE** **FROM** **users**

**WHERE** **status** = 'inactive';

Отменяем два последних удаления, возвращаясь к контрольной точке sp2, созданной после первого удаления:

**ROLLBACK** **TO** sp2;

Команда SET TRANSACTION используется для инициализации транзакции, т.е. начала ее выполнения. При этом, можно определять некоторые характеристики транзакции. Например, так можно определить уровень доступа транзакции (доступна только для чтения или для записи тоже)

**SET** **TRANSACTION** [**READ** WRITE | **READ** **ONLY**];

Рандомные функции

|  |  |
| --- | --- |
| **AVG** — вычисляет среднее значение  **SUM**— вычисляет сумму значений  **MIN**— вычисляет наименьшее значение  **MAX**— вычисляет наибольшее значение  **COUNT** — вычисляет количество записей в таблице | Также существует несколько встроенных функций для работы со строками:  **CONCAT**— объединение строк  **LENGTH** — возвращает количество символов в строке  **TRIM** — удаляет пробелы в начале и конце строки  **SUBSTRING** — извлекает подстроку из строки  **REPLACE** — заменяет подстроку в строке  **LOWER** — переводит символы строки в нижний регистр  **UPPER** — переводит символы строки в верхний регистр и т.д. |
| с числами:  **ROUND** — округляет число  **TRUNCATE** — обрезает дробное число до указанного количества знаков после запятой  **CEILING** — возвращает наименьшее целое число, которое больше или равно текущему значению  **FLOOR** — возвращает наибольшее целое число, которое меньше или равно текущему значению  **POWER** — возводит число в указанную степень  **SQRT** — возвращает квадратный корень числа  **RAND** — генерирует случайное число с плавающей точкой в диапазоне от 0 до 1 | Выражения для работы с датами  **SELECT CURRENT\_TIMESTAMP -** функция для получения текущей даты и времени (их много, гугли)  **Функции для разбора даты и времени:**  **DAYOFMONTH(date)** — возвращает день месяца в виде числа  **DAYOFWEEK(date)** — возвращает день недели в виде числа  **DAYOFYEAR(date)** — возвращает номер дня в году  **MONTH(date)** — возвращает месяц  **YEAR(date)** — возвращает год  **LAST\_DAY(date)**— возвращает последний день месяца в виде даты  **HOUR(time)** — возвращает час  **MINUTE(time)** — возвращает минуты  **SECOND(time)** — возвращает секунды и др. |
| Функции для манипулирования датами:  **DATE\_ADD(date, interval)**— выполняет сложение даты и определенного временного интервала  **DATE\_SUB(date, interval)** — выполняет вычитание из даты определенного временного интервала  **DATEDIFF(date1, date2)** — возвращает разницу в днях между двумя датами  **TO\_DAYS(date)** — возвращает количество дней с 0-го дня года  **TIME\_TO\_SEC(time)**— возвращает количество секунд с полуночи и др. | Для форматирования даты и времени используются функции **DATE\_FORMAT(date, format)**и **TIME\_FORMAT(date, format)**, соответственно. |

**UNION**

|  |  |
| --- | --- |
| id айди | name название |
| 1 | Беларусь |
| 2 | Россия |
| 3 | Украина |

|  |  |  |
| --- | --- | --- |
| id айди | name название | country\_id айди страны |
| 1 | Минск | 1 |
| 2 | Минск | 1 |
| 3 | Москва | 2 |
| 4 | Киев | 3 |

SELECT id, name FROM countries UNION ALL SELECT id, name FROM cities

|  |  |
| --- | --- |
| **id** айди | **name** название |
| 1 | Беларусь |
| 2 | Россия |
| 3 | Украина |
| 1 | Минск |
| 2 | Минск |
| 3 | Москва |
| 4 | Киев |