### 

[1-1.1 Обзор SQL и его применения](#_w7w8jmalnf5t)

[1.1-2 Общее определение типов Баз Данных](#_o7w2uskadkpt)

[1.1-3 Реляционные Базы Данных (RDBMS)](#_texmykf8fhfg)

[1-1.4 Нереляционные (NoSQL) Базы Данных](#_mqm4vymar7z9)

[1-1.5 Объектно-ориентированные Базы Данных](#_x74hnw6orx7a)

[1.1-6 Иерархические Базы Данных](#_5mqsza2vvp5t)

[1.1-7 Сетевые Базы Данных](#_fm3pfy55c7l)

[1-2.1 SQL Select](#_s0ippzf7p08t)

[1.2-2 Использование звездочки \*](#_pijrnzoff3pn)

[Использование псевдонима AS](#_vx2rfxn7nkkk)

[1.2-3 SQL Select Distinct](#_ipk8cxt5dxn4)

[1.2-4 SQL LIMIT](#_wc3tpbay02xm)

[1-3.1 SQL Comments](#_eekju9ln0x3)

[1.3-2 Основы синтаксиса SQL](#_kvu10vscf7nu)

[2.1-1 SQL Where](#_4mjgbuqghb2s)

[Математические операторы сравнения](#_fiykqemfg14b)

### 

### 

### 

### **1-1.1 Обзор SQL и его применения**

#### **Определение:**

SQL (Structured Query Language) — это язык запросов, используемый для управления и манипуляции данными в базах данных.

#### **Объяснение:**

Данные - книги, а SQL - библиотека. В ней можно легко находить, добавлять и удалять данные.

#### **Примеры из практики**

* **Бизнес-Аналитика**: Компании используют SQL для анализа продаж, например, для поиска самых популярных продуктов или определения регионов с наивысшими продажами.
* **Веб-Разработка**: Веб-сайты используют базы данных для хранения информации о пользователях, товарах, заказах и т.д.
* **Наука о Данных**: Аналитики и ученые используют SQL для извлечения данных из баз для анализа и построения моделей.

**Уникальность SQL:**

* **Специализация на Данных:** В отличие от языков программирования, таких как Java или Python, SQL специализируется исключительно на управлении данными и базами данных. Он не предназначен для создания приложений или автоматизации задач.
* **Декларативный Подход:** SQL относится к декларативным языкам, что означает, что пользователь определяет, что нужно сделать с данными, а не как выполнить эту операцию.

**Преимущества SQL:**

* **Универсальность:** SQL поддерживается большинством систем управления базами данных, таких как MySQL, PostgreSQL, SQL Server, и Oracle.
* **Мощь в Анализе Данных:** SQL является незаменимым инструментом в аналитике данных, позволяя эффективно обрабатывать большие объемы данных и извлекать ценную информацию.
* **Легкость изучения:** Основы SQL достаточно просты для понимания, что делает его доступным для начинающих в области работы с данными.

SELECT name, age

FROM employees

WHERE department = 'Маркетинг';

Этот запрос извлекает имена и возраст сотрудников из таблицы employees, работающих в отделе маркетинга.

### 

### 

### **1.1-2 Общее определение типов Баз Данных**

**Типы Баз Данных:**

1. **Реляционные Базы Данных (RDBMS)**
2. **Нереляционные (NoSQL) Базы Данных**
   * Документо-ориентированные;
   * Ключ-значение;
   * Столбцово-ориентированные;
   * Графовые;
3. **Объектно-ориентированные Базы Данных**
4. **Иерархические Базы Данных**
5. **Сетевые Базы Данных**

**Зачем нужны разные типы Баз Данных?**

* **Соответствие специфике приложений:** Разные приложения имеют различные требования к хранению и обработке данных.
* **Оптимизация производительности:** Некоторые типы баз данных лучше подходят для больших объемов данных или для операций, требующих быстрого чтения/записи.
* **Удовлетворение технических требований:** В зависимости от задачи, может потребоваться определенный тип базы данных, например, графовая база для анализа социальных сетей.

### **1.1-3 Реляционные Базы Данных (RDBMS)**

**Описание:**

Реляционные базы данных (Relational Database Management Systems, RDBMS) основаны на реляционной модели данных, предложенной. В реляционных базах данных информация хранится в виде отдельных таблиц, состоящих из строк (записей) и столбцов (атрибутов). Отношения между таблицами определяются через ключи, позволяя эффективно организовывать и связывать различные данные.

**Примеры СУБД:**

* **MySQL:** Популярная открытая СУБД, широко используемая в веб-разработке.
* **PostgreSQL:** Мощная, открытая СУБД, поддерживающая как традиционные, так и новаторские функции.
* **Oracle:** Коммерческая СУБД, известная своей надежностью и масштабируемостью.
* **Microsoft SQL Server:** Коммерческая СУБД от Microsoft, широко используемая в корпоративных решениях.

**Использование:**

* **Универсальность:** RDBMS подходят для большинства типов приложений благодаря их гибкости, мощности и надежности. Они могут использоваться для всего, от простых веб-приложений до крупных корпоративных систем.
* **Сложные запросы и анализ данных:** Поддержка сложных SQL-запросов, включая соединения (joins), подзапросы и агрегатные функции, делает RDBMS идеальными для комплексного анализа данных.
* **Транзакционность и надежность:** RDBMS обеспечивают согласованность данных и поддержку транзакций, что критически важно для банковских, финансовых и любых других систем, где точность данных является приоритетом.
* **Строгая схема данных:** Наличие строго определенной схемы данных обеспечивает четкую организацию и целостность данных.

**Преимущества:**

* **Стандартизация и совместимость:** SQL как язык запросов является стандартом для RDBMS, что обеспечивает совместимость и переносимость знаний между различными системами.
* **Интеграция с различными инструментами и технологиями:** Широкое распространение и поддержка реляционных баз данных обеспечивает легкую интеграцию с множеством инструментов и технологий, от отчетности и аналитики до веб-разработки и бизнес-приложений.

Реляционные базы данных остаются важным компонентом в мире управления данными, предлагая структурированный, мощный и надежный способ хранения и обработки информации.

### **1-1.4 Нереляционные (NoSQL) Базы Данных**

**Описание:**

NoSQL (Not Only SQL) охватывает широкий спектр баз данных, предназначенных для решения специфических задач, где традиционные реляционные базы данных могут быть неэффективны. Эти базы данных часто оптимизированы для работы с большими объемами данных.

**Типы NoSQL Баз Данных:**

1. **Документо-ориентированные Базы Данных**
   * **Описание:** Хранят данные в виде документов, чаще всего в форматах JSON или XML. Эти базы данных предлагают гибкость в структуре данных, позволяя легко добавлять и изменять поля.
   * **Примеры:** MongoDB, CouchDB.
   * **Использование:** Идеально подходят для веб-приложений, CMS, платформ электронной коммерции, где структура данных может часто изменяться.
2. **Базы Данных ключ-значение**
   * **Описание:** Простейшая форма NoSQL баз данных, где каждое значение ассоциировано с уникальным ключом. Они обеспечивают быстрый доступ к данным и масштабируемость.
   * **Примеры:** Redis, DynamoDB.
   * **Использование:** Подходят для кэширования, сеансов пользователей, чатов, рекомендательных систем.
3. **Столбцово-ориентированные Базы Данных**
   * **Описание:** Оптимизированы для чтения и записи больших объемов данных, храня данные не по строкам, а по столбцам. Это улучшает производительность при работе с большими наборами данных и аналитическими запросами.
   * **Примеры:** Cassandra, HBase.
   * **Использование:** Идеально подходят для аналитических приложений, Big Data, временных рядов.
4. **Графовые Базы Данных**
   * **Описание:** Специализированные на хранении и обработке данных, представленных в виде графов (узлы - это объекты, а ребра - отношения между ними). Это обеспечивает эффективное управление связанными данными.
   * **Примеры:** Neo4j, Amazon Neptune.
   * **Использование:** Очень эффективны для социальных сетей, рекомендательных систем, сетевого анализа.

**Преимущества NoSQL:**

* **Гибкость структур:** NoSQL базы данных обеспечивают гибкость в работе со структурами данных, что позволяет легко адаптироваться к изменяющимся требованиям.
* **Масштабируемость:** Отлично подходят для горизонтального масштабирования и работы с большими объемами данных.
* **Высокая производительность:** Некоторые типы NoSQL баз данных оптимизированы для быстрых операций чтения и записи.

NoSQL базы данных находят свое применение в самых разных областях, от веб-разработки до анализа больших данных, предоставляя гибкие и масштабируемые решения для хранения и обработки информации.

Каждый тип NoSQL базы данных имеет свои уникальные особенности и лучше всего подходит для определенных видов задач, что делает их важным инструментом в современном мире управления данными.

### **1-1.5 Объектно-ориентированные Базы Данных**

**Описание:**

Объектно-ориентированные базы данных (OODBMS) представляют данные в виде объектов, аналогично тому, как это делается в объектно-ориентированных языках программирования. В таких базах данных данные и их отношения представлены в форме объектов, которые могут включать в себя как данные (атрибуты), так и методы (функции). Этот подход обеспечивает естественное и интуитивно понятное представление данных, особенно в сложных приложениях.

**Примеры СУБД:**

* **db4o:** Объектно-ориентированная база данных, предназначенная для платформы Java и .NET.
* **ObjectDB:** База данных, которая поддерживает язык запросов JPA/JDO и предназначена для Java-приложений.

**Использование:**

* **Тесная интеграция с ООП:** Особенно полезны в системах, где требуется тесная интеграция базы данных с объектно-ориентированным программированием. Это позволяет разработчикам работать с данными более естественным и привычным образом.
* **Комплексные приложения:** Хорошо подходят для приложений с сложной бизнес-логикой и структурой данных, где объекты и их взаимодействия являются ключевыми элементами.
* **Объектно-ориентированный дизайн:** Удобны для использования в среде, где уже применяются принципы объектно-ориентированного дизайна, облегчая проектирование и поддержку программного обеспечения.

**Преимущества:**

* **Согласованность с ООП:** Объектно-ориентированные базы данных облегчают разработку, поскольку структура данных в базе напрямую соответствует структуре объектов в приложении.
* **Избегание импедансного расхождения:** Снижается проблема импедансного расхождения, которая возникает, когда данные в приложениях представлены в виде объектов, в то время как в базе данных они хранятся в табличной форме.
* **Упрощение разработки:** Уменьшается необходимость в преобразовании данных между базой данных и приложением, упрощая процесс разработки.

Объектно-ориентированные базы данных предоставляют уникальный подход к управлению данными, который может быть особенно полезен в средах, где объектно-ориентированное программирование является доминирующим парадигмой.

### **1.1-6 Иерархические Базы Данных**

**Описание:**

Иерархические базы данных представляют данные в виде дерева, где каждая запись имеет один родительский элемент и может иметь множество дочерних элементов. Эта модель напоминает структуру дерева с ветвями, идеально подходящую для данных, естественно организованных в иерархическом порядке. Отношения в таких базах данных обычно представляются как "родитель-потомок".

**Примеры СУБД:**

* **IBM IMS (Information Management System):** Одна из первых систем баз данных, разработанная IBM, которая до сих пор используется в крупных корпоративных и банковских системах.

**Использование:**

* **Организационные структуры:** Иерархические базы данных подходят для управления данными, которые имеют естественную иерархическую структуру, например, структуры организаций, каталоги файлов, и т.д.
* **Системы управления контентом:** Используются в CMS для организации контента в иерархическом виде, например, в структурах веб-сайтов.
* **Транзакционные системы:** Подходят для сложных транзакционных систем, где иерархия и скорость доступа к данным являются важными факторами.

**Преимущества:**

* **Четкая иерархия:** Иерархическая структура упрощает понимание и навигацию по данным.
* **Быстрый доступ к данным:** Доступ к данным часто быстрее, чем в реляционных базах данных, особенно когда необходимо навигировать по иерархическим путям.
* **Эффективность для определенных типов запросов:** Запросы, которые естественно соответствуют иерархии, могут быть выполнены очень эффективно.

Иерархические базы данных являются одним из старейших типов баз данных и до сих пор используются в специализированных приложениях, где их уникальные особенности и преимущества могут быть наиболее полезными.

### **1.1-7 Сетевые Базы Данных**

**Описание:**

Сетевые базы данных являются расширением иерархической модели баз данных. В этом типе баз данных каждый узел (запись) может иметь несколько родителей, что позволяет создавать сложные сетевые структуры данных. Эта модель предлагает большую гибкость в определении отношений между различными элементами данных по сравнению с иерархической моделью, где каждый узел имеет только одного родителя.

**Примеры СУБД:**

* **IDMS (Integrated Database Management System):** Один из первых коммерческих сетевых систем управления базами данных, который до сих пор используется в некоторых корпоративных системах.

**Использование:**

* **Сложные приложения:** Сетевые базы данных хорошо подходят для приложений, где требуются многоуровневые и многомерные отношения, таких как сложные производственные процессы, системы управления запасами, и сложные организационные структуры.
* **Крупные корпоративные системы:** Их использование оправдано в крупных корпоративных системах, где необходимо управление сложными сетями данных и отношений.
* **Системы, требующие высокой гибкости в определении отношений:** Подходят для систем, где отношения между данными не могут быть легко представлены в иерархической или реляционной моделях.

**Преимущества:**

* **Гибкость в определении отношений:** Позволяет более естественно представлять сложные отношения между данными.
* **Эффективный доступ к связанным данным:** Поскольку отношения между данными являются частью структуры базы данных, доступ к связанным данным может быть более эффективным.
* **Управление сложными структурами:** Лучше подходит для приложений, где структура данных не является строго иерархической и требует большей гибкости в определении связей.

Сетевые базы данных предлагают уникальный подход к управлению данными, который особенно полезен в приложениях, где требуется управление сложными и многомерными отношениями. Несмотря на то, что сетевая модель менее популярна сегодня по сравнению с реляционными или NoSQL системами, она по-прежнему имеет свое место в определенных специализированных областях.

### **1-2.1 SQL Select**

#### **Определение:**

SELECT в SQL — это команда, используемая для выборки данных из одной или нескольких таблиц базы данных.

#### **Объяснение:**

Представьте, что таблица базы данных — это список всех книг в библиотеке. Используя SELECT, вы можете выбрать просмотреть только определенную информацию из этого списка, например названия книг и их авторов, не видя остальных данных.

#### **Применение на практике:**

* **Отчеты по продажам**: Выборка данных о продажах из таблицы продаж для анализа общего объема продаж или продаж по конкретным товарам.
* **Управление клиентской базой**: Получение списка клиентов с определенными характеристиками, например, из определенного города или в определенном возрастном диапазоне.
* **Аналитика веб-сайтов**: Выборка данных о посещениях веб-страниц для анализа поведения пользователей.

#### **Синтаксис:**

Основной синтаксис команды SELECT выглядит следующим образом:

SELECT column1, column2, ...

FROM table\_name;

Здесь column1, column2 — это столбцы, которые вы хотите выбрать, а table\_name — имя таблицы, из которой производится выборка.

**Пример запроса:**

SELECT Name, Age

FROM Employees;

Этот запрос выберет данные из столбцов Name (имя) и Age (возраст) всех записей в таблице Employees (сотрудники). Он полезен, если вам нужно получить только имена и возраст сотрудников, без другой информации, содержащейся в таблице.

### **1.2-2 Использование звездочки \***

**Определение:**

В SQL \* (звездочка) в команде SELECT используется для выбора всех столбцов из указанной таблицы.

**Объяснение:**

Использование звезды в запросе SELECT подобно запросу к библиотекарю дать вам список всех глав каждой книги в разделе. Это удобный способ получить полное содержимое таблицы без необходимости перечислять все названия столбцов.

**Применение на практике:**

* **Быстрый обзор данных**: При исследовании новой таблицы данных можно использовать SELECT \* для получения общего представления о всех столбцах и данных.
* **Администрирование баз данных**: Для проверки структуры и содержимого таблиц после выполнения операций вставки, обновления или удаления.

**Синтаксис**

SELECT \* FROM table\_name;

### **Использование псевдонима AS**

**Определение:**

AS в SQL используется для создания псевдонима (alias) для столбца или таблицы в запросе.

**Объяснение:**

Использование AS в запросе SQL похоже на то, как вы могли бы дать кличку своему домашнему питомцу. Вместо использования полного или сложного имени, вы используете короткое и понятное обозначение.

**Применение на практике:**

* **Отчеты**: Использование псевдонимов для столбцов с длинными или неинтуитивными названиями при формировании отчетов.
* **Сложные запросы**: Упрощение написания и понимания запросов, особенно при соединении нескольких таблиц.

**Синтаксис:**

SELECT column\_name AS alias\_name

FROM table\_name;

**Объяснение**: Этот запрос выбирает столбцы first\_name и last\_name из таблицы employees, но в результате они будут отображаться как name и surname. Это делает результаты запроса более понятными, особенно если исходные названия столбцов не очень информативны или слишком длинны.

### **1.2-3 SQL Select Distinct**

#### **Определение:**

Команда SELECT DISTINCT в SQL используется для выборки уникальных записей из таблицы, исключая дубликаты.

#### **Объяснение:**

Представьте, что у вас есть коробка, наполненная разноцветными шариками, где некоторые цвета повторяются. Если вы хотите узнать, какие уникальные цвета шариков есть в коробке, вы бы выбрали по одному шарику каждого цвета. SELECT DISTINCT работает подобным образом.

#### **Применение на практике:**

* **Маркетинговые Анализы**: Выборка уникальных категорий товаров из таблицы продуктов для определения разнообразия ассортимента.
* **Анализ Клиентской Базы**: Получение списка уникальных городов, из которых есть клиенты, для географического анализа рынка.
* **Исследование Данных**: Определение уникальных типов транзакций в финансовых отчетах.

#### **Синтаксис:**

Основной синтаксис команды SELECT DISTINCT:

SELECT DISTINCT column1, column2, ...

FROM table\_name;

SELECT DISTINCT является важным инструментом для анализа данных, особенно когда требуется понять разнообразие или уникальность значений в наборе данных. Это помогает избежать искажения результатов за счет повторяющихся записей.

### **1.2-4 SQL LIMIT**

#### **Определение:**

Команда LIMIT в SQL, используемая в MySQL, ограничивает количество строк, возвращаемых в результате запроса. Она позволяет выбирать фиксированное количество записей из начала результата запроса.

#### **Объяснение:**

LIMIT в MySQL подобен запросу на получение ограниченного количества книг из большого списка в библиотеке. Например, если вам нужно изучить только несколько первых записей из большого набора данных, вы можете использовать LIMIT для ограничения вывода.

#### **Применение на практике:**

* **Анализ данных**: Выборка топ-5 продуктов по продажам для определения наиболее популярных товаров.
* **Отчетность**: Получение первых 10 записей из большого набора данных для создания сводных отчетов.
* **Ограниченный просмотр**: Показ только первых нескольких записей в пользовательском интерфейсе, чтобы не перегружать экран большим количеством данных.

#### **Синтаксис:**

В MySQL синтаксис LIMIT следующий:

SELECT column\_name(s)

FROM table\_name

LIMIT number;

number здесь указывает на максимальное количество строк, которые должны быть возвращены.

LIMIT является ключевым инструментом в MySQL для управления объемом данных, возвращаемых запросом. Он не только ускоряет выполнение запросов, но и помогает избежать перегрузки при обработке и отображении данных.

Важно отметить, что аналогичная функциональность в других СУБД, например, в SQL Server или MS Access, реализована через команду SELECT TOP.

### **1-3.1 SQL Comments**

**Определение:**

Комментарии в SQL - это фрагменты текста в коде SQL-запросов, которые не выполняются как часть запроса. Они используются для пояснения кода, уточнения запросов или временного исключения определенных частей кода из выполнения.

**Важность комментариев:**

Комментарии в SQL имеют решающее значение для:

* **Повышения читаемости:** Объяснение сложной логики или указание на цели определенных запросов.
* **Облегчения совместной работы:** Помогают другим разработчикам понять ваш код, что критически важно в командной работе.
* **Упрощения отладки и тестирования:** Позволяют временно отключить части кода без удаления их.

**Синтаксис комментариев:**

В SQL существуют два основных типа комментариев:

**1. Однострочные комментарии:**

* Начинаются с -- и продолжаются до конца строки.

**Пример:**

-- Это однострочный комментарий

SELECT \* FROM users; -- Комментарий после кода

**2. Многострочные (блочные) комментарии:**

* Начинаются с /\* и заканчиваются на \*/.
* Могут занимать несколько строк.

**Пример:**

/\* Это многострочный комментарий,

который продолжается на несколько строк \*/

SELECT \* FROM users;

**Применение на практике:**

* **Описание логики запроса:**

-- Получение списка пользователей старше 30 лет

SELECT name, age FROM users WHERE age > 30;

* **Временное отключение кода:**

SELECT name, age FROM users;

-- SELECT email FROM users; -- Этот запрос временно отключен

* **Объяснение сложных запросов:**

/\*

Запрос для получения общего количества покупок

каждого пользователя за последний месяц

\*/

SELECT user\_id, COUNT(\*) FROM purchases

WHERE purchase\_date > '2021-01-01'

GROUP BY user\_id;

Использование комментариев в SQL улучшает понимание и поддержку кода, делая его более доступным для других разработчиков и упрощая процесс отладки и модификации запросов. Комментирование кода является лучшей практикой программирования, и ее стоит применять регулярно.

### **1.3-2 Основы синтаксиса SQL**

#### **Определение:**

Синтаксис SQL — это набор правил, которые определяют структуру и формат команд в SQL, используемых для выполнения различных операций с базами данных, таких как создание таблиц, выборка данных, обновление записей и многие другие.

#### **Объяснение:**

Синтаксис SQL можно сравнить с грамматикой языка. Так же, как в обычном языке есть правила, определяющие, как строить предложения, в SQL есть правила, которые определяют, как формировать команды. Эти команды позволяют "общаться" с базой данных: запрашивать данные, изменять их, удалять и т.д.

#### **Синтаксис**

Основные элементы синтаксиса SQL включают ключевые слова, операторы, выражения и предложения. Ключевые слова, такие как SELECT, INSERT, UPDATE, и DELETE, определяют тип операции. Операторы, например, =, >, <, используются для сравнения значений. Выражения и предложения строятся из этих элементов для формирования полных команд SQL.

**Объяснение**:

Этот запрос выбирает имена сотрудников из таблицы Employees, возраст которых больше 30 лет. SELECT Name FROM Employees указывает, что нужно выбрать столбец Name из таблицы Employees. WHERE Age > 30 является условием, фильтрующим записи, где возраст (Age) больше 30.

Синтаксис SQL является основой для работы с базами данных. Понимание этого синтаксиса позволяет формировать эффективные и точные запросы для управления данными. В последующих разделах мы рассмотрим более детально различные команды и операторы SQL, расширяя ваше понимание и навыки работы с базами данных.

**МОДУЛЬ 2**

### 

### **2.1-1 SQL Where**

#### **Определение:**

Блок WHERE в SQL используется для фильтрации записей, отвечающих определенным условиям, в операторах выборки, таких как SELECT, UPDATE или DELETE. Он определяет критерий, согласно которому строка будет включена в результат или исключена из него.

#### **Объяснение:**

Представьте, что у вас есть таблица с данными о студентах, включающая их имена и оценки. Если вам нужно найти студентов, которые получили оценку выше 75, вы используете блок WHERE в запросе SELECT, чтобы отфильтровать только тех студентов, чьи оценки соответствуют этому условию. Это позволяет точно настраивать запросы, делая их более специфическими и эффективными.

#### **Синтаксис:**

SELECT column\_name\_1, column\_name\_2, ....

FROM table\_name

WHERE condition;

Здесь condition может включать различные критерии, такие как сравнения (>, <, = и т.д.) и логические операторы (AND, OR, NOT), для создания более сложных условий фильтрации.

#### **Пример запроса:**

SELECT Name, Grade FROM Students

WHERE Grade > 75;

Этот запрос выберет имена и оценки студентов из таблицы Students, но только для тех, у кого оценка больше 75. Таким образом, WHERE действует как фильтр, позволяющий выделить интересующую подгруппу данных из общей таблицы.

### **Математические операторы сравнения**

#### **Определение:**

Математические операторы сравнения в SQL используются для сравнения двух значений или выражений. К основным операторам относятся =, >, <, >=, <=, <> (или !=).

#### **Объяснение:**

Операторы сравнения позволяют осуществлять выборку, фильтрацию, обновление и удаление данных на основе заданных условий. Они являются фундаментальной частью SQL-запросов, позволяя проводить анализ и обработку данных в базе.

#### **Пример запроса:**

Выборка заказов с суммой меньше 500 и не равной 0:

SELECT \* FROM Orders

WHERE Amount < 500 AND Amount <> 0;

* SELECT \* FROM Orders указывает на то, что необходимо выбрать все поля (\*) из таблицы Orders.
* Amount < 500 - Это условие говорит о том, что в результаты выборки должны попасть те записи, где значение в поле Amount (сумма заказа) меньше 500.
* AND Amount <> 0 - Это дополнительное условие, соединенное с предыдущим при помощи логического оператора AND. Оно указывает, что сумма заказа не должна быть равна 0. Здесь используется оператор <>, который означает "не равно".

### **2.2-1 Логический оператор AND в SQL**

**Определение:**

Логический оператор AND в SQL используется для комбинирования двух или более условий в одном SQL-запросе.

**Синтаксис:**

SELECT column\_name\_1, column\_name\_2, ....

FROM table\_name

WHERE condition\_1 AND condition\_2;

### **2.2-2 Логический оператор OR в SQL**

**Определение:**

Логический оператор OR в SQL применяется для объединения двух или более условий в одном запросе, где результат будет возвращен, если хотя бы одно из условий истинно.

WHERE condition\_1 OR condition\_2;

### **2.2-3 Логический оператор NOT в SQL**

**Определение:**

Логический оператор NOT в SQL используется для инвертирования условия в запросе. Он выбирает строки, где заданное условие не истинно.

WHERE NOT condition;

### **2.2-4 Использование команды ORDER BY в SQL**

**Определение:**

Команда ORDER BY в SQL используется для сортировки результатов запроса по одному или нескольким столбцам в порядке возрастания или убывания. То есть сортирует вывод по выбранным столбцам

**Практический пример:**

Компания хочет получить отсортированный список всех транзакций по дате, чтобы увидеть самые последние операции в начале списка.

### **Сортировка по одному столбцу без указания ASC или DESC**

Когда ASC или DESC не указаны, SQL по умолчанию применяет сортировку по возрастанию (ASC).

**Пример запроса:**

SELECT Name, Age

FROM Employees

ORDER BY Age;

В этом запросе выбираются имена и возраст сотрудников из таблицы Employees. Результаты будут отсортированы по возрасту сотрудников в порядке возрастания. Поскольку направление сортировки не указано, SQL автоматически применяет сортировку по возрастанию. Это означает, что сначала будут показаны сотрудники младшего возраста, а затем более старшие.

### **Сортировка двух столбцов без явного указания ASC или DESC**

**Синтаксис:**

SELECT column\_name\_1, column\_name\_2, ...

FROM table

ORDER BY column\_name\_1, column\_name\_2 ...;

При отсутствии явного указания, SQL использует сортировку по возрастанию (ASC) по умолчанию.

**Пример запроса:**

SELECT LastName, Department

FROM Employees

ORDER BY Department, LastName;

В этом запросе выбираются фамилии сотрудников и названия их отделов из таблицы Employees. Сначала результаты будут отсортированы по отделам, а затем внутри каждого отдела — по фамилиям сотрудников.

### **Сортировка по алфавиту (DESC)**

**Пример запроса:**

SELECT LastName, FirstName

FROM Employees

ORDER BY LastName DESC;

В этом запросе выбираются фамилии и имена сотрудников из таблицы Employees. Результаты будут отсортированы по фамилиям в обратном алфавитном порядке (от Я до А), благодаря использованию DESC в ORDER BY.

### **Упорядочивание по нескольким столбцам**

**Пример запроса:**

SELECT LastName, FirstName, Department

FROM Employees

ORDER BY Department ASC, LastName DESC;

В этом запросе выбираются фамилии, имена и отделы сотрудников из таблицы Employees. Сначала результаты будут отсортированы по отделам в алфавитном порядке (ASC), а затем внутри каждого отдела — по фамилиям сотрудников в обратном алфавитном порядке (DESC).