Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»

ОТЧЁТ ПО

Домашнему заданию

Выполнил: Мишакин А.О.

студент группы ИУ5-31Б

_____

Подпись и дата:

Проверил:

преподаватель каф. ИУ5

Нардид А.Н.

Подпись и дата:

Москва

2024

Оглавление:

I. **Введение**
   1. **История развития SQL**
   2. **Важность SQL в современном мире**

II. **Основная часть**
   1. **Основы SQL**
      - Структура SQL
      - Основные команды и их использование
      - Типы данных в SQL
   2. **Работа с базами данных**
      - Создание базы данных
      - Управление таблицами
      - Вставка, обновление и удаление данных
   3. **Запросы и выбор данных**
      - Команды SELECT, WHERE, ORDER BY, GROUP BY
      - Объединение таблиц (JOIN)
      - Подзапросы
   4. **Оптимизация и администрирование баз данных**
      - Индексы и их влияние на производительность
      - Транзакции и их свойства
      - Резервное копирование и восстановление данных

III. **Заключение**
   1. **Перспективы развития SQL**
   2. **Роль SQL в карьере IT-специалиста**

1. Введение

I. История развития SQL.

SQL (Structured Query Language) был первоначально разработан в начале 1970-х годов исследователями из компании IBM в рамках проекта System R. Целью этого проекта было создание языка для управления и манипулирования данными в реляционных базах данных.

Этапы развития SQL:

1. 1970-е годы: Начало разработки

- В 1970 году Эдгар Ф. Кодд, сотрудник IBM, опубликовал статью "A Relational Model of Data for Large Shared Data Banks", в которой представил концепцию реляционной модели данных.
- В середине 1970-х годов команда IBM начала работу над проектом System R, целью которого было создание системы управления реляционными базами данных.
- В ходе работы над этим проектом был разработан первый прототип языка SQL, тогда называвшийся SEQUEL (Structured English Query Language).

2. 1980-е годы: Стандартизация и первые продукты

- В 1980-х годах были выпущены первые коммерческие продукты на базе SQL. Одним из первых продуктов стала система Oracle Database, разработанная компанией Oracle Corporation (тогда называвшейся Relational Software).
- В 1986 году Американский национальный институт стандартов (ANSI) принял первую версию стандарта SQL.
- В течение 1980-х годов появились и другие коммерческие СУБД, такие как Ingres, Informix и Sybase.

3. 1990-е годы: Расширение функционала и популярность

- В 1990-х годах SQL продолжал активно развиваться. Были добавлены новые функции, такие как поддержка вложенных запросов (подзапросов), расширенные возможности объединения таблиц и поддержка больших объемов данных.
- Появились новые версии стандарта ANSI SQL, такие как SQL-92 и SQL-99.
- Развитие интернета привело к широкому распространению SQL-базированных систем управления контентом (CMS) и других веб-приложений.

4. 2000-е годы и далее: Современные тенденции

- В 2000-х годах SQL продолжил эволюционировать, адаптируясь к новым технологическим вызовам, таким как большие данные (Big Data) и облачные технологии.

- Появились NoSQL базы данных, предлагающие альтернативные подходы к хранению и обработке данных, однако SQL остается основным инструментом для большинства реляционных баз данных.

- Современные версии SQL включают поддержку распределенных вычислений, аналитики в реальном времени и интеграции с другими технологиями, такими как искусственный интеллект и машинное обучение.

II. Важность SQL в современном мире.

SQL (Structured Query Language) играет ключевую роль в современном мире благодаря следующим аспектам:

1. **Управление данными**: SQL является стандартным языком для работы с реляционными базами данных, что позволяет эффективно управлять большими объемами структурированных данных.
2. **Критическая технология**: SQL необходим для множества приложений, начиная от банковских систем и заканчивая веб-сайтами электронной коммерции. Без него невозможно было бы эффективно хранить и извлекать данные.
3. **Широкая применимость**: SQL используется в различных отраслях, от финансов до здравоохранения, что делает его важным навыком для специалистов в области ИТ и аналитики данных.
4. **Аналитика и бизнес-интеллект**: SQL позволяет проводить анализ данных, что критично для принятия обоснованных бизнес-решений и стратегического планирования.
5. **Интеграция и автоматизация**: SQL помогает интегрировать различные системы и автоматизировать процессы, что повышает эффективность работы организаций.

2. Основная часть

I. Основы SQL

1. Структура SQL

1. **Команды (Statements)**

   DDL (Data Definition Language): Команды для определения структуры данных, такие как создание, изменение и удаление таблиц и других объектов базы данных. Например:

   CREATE TABLE: создает новую таблицу.

   ALTER TABLE: изменяет структуру существующей таблицы.

DROP TABLE: удаляет таблицу.

DML (Data Manipulation Language): Команды для манипулирования данными, такие как вставка, обновление и удаление записей. Например:

INSERT INTO: вставляет новые записи в таблицу.

UPDATE: обновляет существующие записи.

DELETE FROM: удаляет записи.

DCL (Data Control Language): Команды для управления правами доступа к данным. Например:

GRANT: дает пользователям права на выполнение определенных операций.

REVOKE: отзывает ранее предоставленные права.

TCL (Transaction Control Language): Команды для управления транзакциями. Например:

COMMIT: фиксирует текущую транзакцию.

ROLLBACK: откатывает текущую транзакцию.

2. **Запросы (Queries)**

   Основной элемент SQL, который используется для извлечения данных из базы данных. Пример простого запроса:

   SELECT column_name(s) FROM table_name WHERE condition;

   Ключевые слова:

   SELECT: выбирает данные из одной или нескольких таблиц.

   FROM: указывает таблицу, из которой берутся данные.

   WHERE: фильтрует результаты согласно определенному условию.

3. **Типы данных (Data Types)**
   - SQL поддерживает различные типы данных для хранения различной информации. Некоторые примеры:

     INT: целое число.

     VARCHAR(n): строка переменной длины.

     DATE: дата.

     BOOLEAN: логическое значение.

4. **Таблицы (Tables)** – Основные объекты в реляционных базах данных, состоящие из строк (записей) и столбцов (полей). Таблицы содержат данные, организованные в виде двумерных структур.
5. **Первичные ключи (Primary Keys)** – Столбцы, которые однозначно идентифицируют каждую запись в таблице. Первичные ключи обычно уникальны и не допускают NULL-значений.
6. **Вторичные ключи (Foreign Keys)** – Столбцы, которые ссылаются на первичные ключи в других таблицах, обеспечивая связь между таблицами и целостность данных.
7. **Индексы (Indexes)** – Специальные структуры данных, которые ускоряют поиск и сортировку данных в таблицах. Индексы создаются на основе

определенных столбцов и помогают улучшить производительность запросов.

Пример SQL-запроса

```
SELECT first_name, last_name
FROM employees
WHERE department = 'Marketing'
AND salary > 50000;
```

Этот запрос выберет имена и фамилии сотрудников отдела маркетинга, чья зарплата превышает 50000.

II.Основные команды и их использование

1. **CREATE TABLE – Использование:** создаёт новую таблицу в базе данных. **Синтаксис**:

```
CREATE TABLE table_name (
column1_name datatype,
column2_name datatype,
 ...
PRIMARY KEY (one or more columns)
);
```

2. **INSERT INTO – Использование**: добавляет новую запись в таблицу. **Синтаксис**:

```
INSERT INTO table_name (column1, column2, ...)
VALUES (value1, value2, ...);
```

3. **SELECT – Использование**: извлекает данные из одной или нескольких таблиц. **Синтаксис**:

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

4. **UPDATE – Использование**: обновляет существующие записи в таблице. **Синтаксис**:

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

5. **DELETE FROM – Использование:** удаляет записи из таблицы. **Синтаксис**:

```
DELETE FROM table_name
WHERE condition;
```

6. **ALTER TABLE – Использование**: изменяет структуру существующей таблицы. **Синтаксис**:

```
ALTER TABLE table_name
ADD COLUMN column_name datatype;
```

7. **DROP TABLE – Использование**: удаляет таблицу из базы данных.
   Синтаксис:

```
DROP TABLE table_name;
```

3.Типы данных

I. Числовые типы данных

INTEGER (INT): Целое число. Диапазон зависит от конкретной реализации SQL, но обычно это 32-битное целое число (-2^31 to 2^31-1).

SMALLINT: Короткое целое число. Обычно занимает 16 бит и имеет диапазон от -32,768 до 32,767.

BIGINT: Длинное целое число. Обычно занимает 64 бита и имеет большой диапазон значений.

NUMERIC (DECIMAL): Десятичное число с фиксированной точностью. Может хранить как целую, так и дробную часть.

FLOAT (DOUBLE PRECISION): Число с плавающей точкой. Хранит числа с плавающей точкой двойной точности.

2. Символьные типы данных

CHAR (N): Фиксированная длина строки длиной N символов. Если строка короче, то она дополняется пробелами.

VARCHAR (N): Переменная длина строки. Максимальная длина строки ограничена значением N.

TEXT: Большой объем текста. Может хранить большое количество символов, например, статьи или документы.

3. Логические типы данных

BOOLEAN: Логическое значение (TRUE или FALSE).

4. Дата и время

DATE: хранит дату без времени. Формат: ГГГГ-ММ-ДД.

TIME: хранит время без даты. Формат: ЧЧ: ММ: СС.

DATETIME (TIMESTAMP): хранит одновременно дату и время. Формат: ГГГГ-ММ-ДД ЧЧ: ММ: СС.

INTERVAL: хранит временной интервал.

5. Специальные типы данных

BLOB (Binary Large Object): Большие двоичные объекты, такие как изображения, аудиофайлы и др.

CLOB (Character Large Object): Большие символьные объекты, например, тексты большого размера.

ENUM: Перечисление. Хранит список возможных значений.

SET: Множество. Хранит набор значений из предопределенного списка.

6. Другие типы данных

SERIAL: Автоинкрементное целое число. Аналог AUTO_INCREMENT в некоторых системах.

UUID: Универсальный уникальный идентификатор.

Пример использования типов данных

```
-- Создаем таблицу с различными типами данных
CREATE TABLE example_table (
   id SERIAL PRIMARY KEY, -- Автоинкрементное целое число
   integer_field INT,      -- Целое число
   decimal_field NUMERIC(10, 2), -- Десятичное число с фиксированной
точностью
   varchar_field VARCHAR(50), -- Строка переменной длины
   text_field TEXT,         -- Большой объем текста
   date_field DATE,         -- Дата
   time_field TIME,         -- Время
   datetime_field DATETIME   -- Дата и время
);
```

II. Работа с базами данных

Создание базы данных

CREATE DATABASE: Команда для создания новой базы данных.

```
CREATE DATABASE database_name;
```

Параметры:

database_name: Имя новой базы данных.

Управление таблицами

CREATE TABLE: Команда для создания новой таблицы в базе данных.

```
CREATE TABLE table_name (
        column1_name datatype,
        column2_name datatype,
         ...
        PRIMARY KEY (one or more columns)
);
```

ALTER TABLE: Команда для изменения структуры существующей таблицы.

```
ALTER TABLE table_name
ADD COLUMN column_name datatype;
```

DROP TABLE: Команда для удаления таблицы из базы данных.

```
DROP TABLE table_name;
```

Вставка, обновление и удаление данных

INSERT INTO: Команда для добавления новых записей в таблицу.

```
INSERT INTO table_name (column1, column2, ...)
VALUES (value1, value2, ...);
```

UPDATE: Команда для обновления существующих записей в таблице.

```
UPDATE table_name
```

```
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

DELETE FROM: Команда для удаления записей из таблицы.

```
DELETE FROM table_name
WHERE condition;
```

III. Запросы на выбор данных

Команды SELECT, WHERE, ORDER BY, GROUP BY

SELECT: Команда для выбора данных из одной или нескольких таблиц.

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Параметры:

column1, column2, ...: Столбцы, которые нужно выбрать.

table_name: Таблица, из которой нужно выбрать данные.

condition: Условие фильтрации данных.

WHERE: Условие для фильтрации данных.

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Параметры: condition: Условие, которому должны соответствовать выбранные записи.

ORDER BY: Сортировка выбранных данных.

```
SELECT column1, column2, ...
FROM table_name
WHERE condition
ORDER BY column1 ASC|DESC, column2
```

Параметры:

column1, column2, ...: Столбцы, по которым производится сортировка.

ASC|DESC: Способ сортировки (по возрастанию или убыванию).

GROUP BY: Группировка данных по одному или нескольким столбцам.

```
SELECT column1, column2, ...
FROM table_name
WHERE condition
GROUP BY column1, column2, ...;
```

Параметры: column1, column2, ...: Столбцы, по которым производится группировка.

Объединение таблиц (JOIN)

JOIN: объединяет данные из двух или более таблиц на основании общего столбца.

```
SELECT column1, column2, ...
FROM table1
JOIN table2 ON table1.column = table2.column
WHERE condition;
```

Параметры:

table1, table2: Таблицы, которые объединяются.

ON: Условие объединения (общий столбец).

INNER JOIN: возвращает только те записи, которые существуют в обеих таблицах.

```
SELECT column1, column2, ...
FROM table1
INNER JOIN table2 ON table1.column = table2.column
WHERE condition;
```

LEFT JOIN: возвращает все записи из первой таблицы и соответствующие записи из второй таблицы.

```
SELECT column1, column2, ...
FROM table1
LEFT JOIN table2 ON table1.column = table2.column
WHERE condition;
```

RIGHT JOIN: возвращает все записи из второй таблицы и соответствующие записи из первой таблицы.

```
SELECT column1, column2, ...
FROM table1
RIGHT JOIN table2 ON table1.column = table2.column
WHERE condition;
```

FULL OUTER JOIN: возвращает все записи, когда есть соответствие в любой из таблиц.

```
SELECT column1, column2, ...
FROM table1
FULL OUTER JOIN table2 ON table1.column = table2.column
WHERE condition;
```

Подзапросы

Подзапрос: Вложенный запрос, который возвращает одно значение или набор данных.

```
SELECT column1, column2, ...
FROM table_name
WHERE column1 IN (SELECT column2 FROM another_table WHERE condition
```

Параметры: (SELECT column2 FROM another_table WHERE condition): Вложенный запрос.

EXISTS: проверяет существование записей, соответствующих условиям подзапроса.

```
SELECT column1, column2, ...
FROM table_name
WHERE EXISTS (SELECT 1 FROM another_table WHERE condition
```

Correlated Subquery: Подзапрос, который ссылается на внешнюю таблицу.

```
SELECT column1, column2, ...
FROM table_name t1
WHERE column1 > (SELECT AVG(column2) FROM table_name t2 WHERE t1.column = t2.column);
```

IV. Оптимизация и администрирование баз данных

Индексы и их влияние на производительность

Индексы: Специальные структуры данных, которые ускоряют поиск и сортировку данных в таблицах.

CREATE INDEX: Команда для создания индекса.

```
CREATE INDEX index_name ON table_name (column1, column2, ...
```

Параметры:

index_name: Имя нового индекса.

table_name: Таблица, для которой создается индекс.

column1, column2, ...: Столбцы, по которым строится индекс.

Влияние на производительность:

Ускоряют выполнение запросов, особенно тех, которые используют условия WHERE и ORDER BY.

Увеличивают скорость поиска данных.

Требуют дополнительного места на диске.

Обновляются при каждом изменении данных, что может замедлить операции вставки, обновления и удаления.

Транзакции и их свойства

Транзакция: Группа операций, которая должна быть выполнена полностью или не выполнена вообще.

ACID Properties:

Atomicity: Все операции в транзакции либо выполняются целиком, либо не выполняются совсем.

Consistency: после завершения транзакции база данных находится в согласованном состоянии.

Isolation: одновременно выполняющиеся транзакции изолированы друг от друга.

Durability: Результаты успешно завершенной транзакции сохраняются даже при сбоях системы.

Команды управления транзакциями:

START TRANSACTION: начать транзакцию. START TRANSACTION;

COMMIT: зафиксировать транзакцию. COMMIT;

ROLLBACK: откатить транзакцию. ROLLBACK;

Резервное копирование и восстановление данных

Резервное копирование: Процесс создания копий данных для последующего восстановления в случае потери или повреждения.

Full Backup: Полное резервное копирование всей базы данных.

```
BACKUP DATABASE database_name TO DISK = 'path\to\backup\file.bak';
```

Differential Backup: Резервное копирование изменений с момента последнего полного копирования.

```
BACKUP DATABASE database_name TO DISK = 'path\to\backup\file.dif' WITH DIFFERENTIAL;
```

Transaction Log Backup: Резервное копирование журнала транзакций.

```
BACKUP LOG database_name TO DISK = 'path\to\backup\file.trn';
```

Восстановление данных:

RESTORE DATABASE: восстанавливает базу данных из резервной копии.

```
RESTORE DATABASE database_name FROM DISK = 'path\to\backup\file.bak';
```

RESTORE LOG: восстанавливает журнал транзакций.

```
RESTORE LOG database_name FROM DISK = 'path\to\backup\file.trn';
```

3. Заключение

1. Перспективы развития SQL

SQL продолжает оставаться одним из наиболее востребованных и перспективных языков программирования. Его развитие идет в ногу с современными технологическими трендами, такими как:

Большие данные (Big Data): SQL-запросы становятся всё более мощными и эффективными для обработки огромных объемов данных.

Облачные технологии: Многие облачные платформы предлагают сервисы баз данных, управляемые SQL, что расширяет возможности использования языка.

Машинное обучение и искусственный интеллект: SQL интегрируется с этими новыми технологиями, помогая анализировать и обрабатывать данные.

Будущее SQL связано с дальнейшим развитием и адаптацией к новым вызовам и возможностям, что делает его незаменимым инструментом в арсенале любого IT-специалиста.

2. Роль SQL в карьере IT-специалиста

SQL играет ключевую роль в карьере IT-специалиста, независимо от его специализации. Вот почему:

Универсальность: SQL нужен везде, где есть необходимость работать с данными. Это делает его обязательным навыком для разработчиков, аналитиков данных, администраторов баз данных и менеджеров проектов.

Востребованность: Большинство компаний, от стартапов до крупных корпораций, используют реляционные базы данных, что делает владение SQL крайне ценным умением на рынке труда.

Повышение эффективности: SQL позволяет быстро и эффективно получать необходимые данные, что экономит время и ресурсы.