1. **Политика информационной безопасности.**

Политика информационной безопасности (ИБ) — это совокупность принципов, правил и процедур, направленных на защиту информации в организации. Она определяет цели, задачи, стратегии и меры по обеспечению конфиденциальности, целостности и доступности информации.

**Пример:** В организации устанавливается политика, согласно которой доступ к конфиденциальным данным базы данных имеют только сотрудники с соответствующими правами доступа.

1. **Доверенная вычислительная база.**

Доверенная вычислительная база (Trusted Computing Base, TCB) — это совокупность аппаратных и программных средств, обеспечивающих выполнение политики безопасности и защищающих систему от угроз.

**Пример:** В контексте SSMS TCB включает сервер SQL, операционную систему и компоненты безопасности, которые контролируют доступ к данным и предотвращают несанкционированные действия.

1. **Субъект доступа.**

Субъект доступа — это лицо или процесс, имеющий возможность взаимодействовать с объектами системы и запрашивать доступ к ресурсам.

**Пример:** В SSMS субъектами доступа являются пользователи базы данных, такие как администраторы, разработчики и конечные пользователи.

1. **Правила разграничения доступа.**

Правила разграничения доступа определяют, какие субъекты имеют право доступа к каким объектам и в каком объеме.

**Пример:** В SSMS можно настроить роли и разрешения, чтобы только определенные пользователи могли изменять структуру базы данных, а другие могли только читать данные.

1. **Объект доступа. Конфиденциальность информации. Целостность информации. Доступность информации.**
   * **Объект доступа:** Ресурс или элемент системы, к которому субъекты могут запрашивать доступ.
   * **Конфиденциальность информации:** Состояние, при котором информация доступна только авторизованным субъектам.
   * **Целостность информации:** Состояние, при котором информация остается точной и неизменной, защищенной от несанкционированных изменений.
   * **Доступность информации:** Состояние, при котором информация доступна и может быть использована теми пользователями которые имеют к ней доступ.

**Пример:** В SSMS объекты доступа включают таблицы и представления. Настройка разрешений обеспечивает, чтобы только авторизованные пользователи могли изменять данные (целостность) и что данные доступны при необходимости (доступность).

1. **Отличительные особенности данных и информации.**

Данные — это необработанные факты и цифры, которые сами по себе не имеют смысла. Информация — сведения независимо от их представления.

**Пример:** В базе данных данные могут быть представлены как набор чисел в таблице. Когда эти данные обрабатываются и представлены в виде отчета с графиками, они становятся информацией, полезной для принятия решений.

1. **Понятие автоматизированной информационной системы.**

Автоматизированная информационная система (АИС) — это совокупность программных и аппаратных средств, а также персонала, обеспечивающих автоматизацию обработки, хранения и передачи информации.

**Пример:** SSMS является частью АИС, предоставляя инструменты для разработки, управления и обеспечения безопасности баз данных.

1. **Понятие защищенной автоматизированной информационной системы.**

Защищенная АИС — это система, в которой реализованы меры по защите информации от несанкционированного доступа, утраты и других угроз, обеспечивая ее безопасность и доверенность.

**Пример:** В SSMS использование шифрования данных, настройки ролей и разрешений, а также регулярный аудит доступа помогают создать защищенную АИС.

1. **Среда управления SSMS. Организация панелей и перемещение по ним. Использование SSMS с компонентом Database Engine.**

SSMS предоставляет графический интерфейс для управления экземплярами SQL Server. Панели, такие как "Обозреватель объектов", "Результаты запросов" и "Сообщения", позволяют эффективно работать с базами данных.

**Пример:** В "Обозревателе объектов" можно просматривать и управлять объектами базы данных, а в "Результатах запросов" видеть результаты выполнения T-SQL запросов.

1. **SSMS. Создание баз данных. Управление БД посредством обозревателя объектов.**

В SSMS базы данных можно создавать с помощью мастера создания базы данных или с помощью T-SQL команд. "Обозреватель объектов" позволяет управлять базами данных, включая создание, изменение и удаление объектов.

**Пример:** Для создания базы данных можно использовать команду T-SQL:

CREATE DATABASE TestDB;

Затем в "Обозревателе объектов" можно увидеть и управлять этой базой данных.

1. **Язык Transact-SQL. Основные объекты SQL.**

Transact-SQL (T-SQL) — это расширение SQL, разработанное Microsoft для управления данными в SQL Server. Основные объекты SQL включают базы данных, таблицы, представления, индексы, хранимые процедуры и функции.

**Пример:** Создание таблицы в T-SQL:

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

Name NVARCHAR(100),

Position NVARCHAR(100));

1. **Язык Transact-SQL. Типы данных.**

В Transact-SQL (T-SQL) используются различные типы данных для определения характеристик столбцов в таблицах. Основные категории типов данных включают:

* + **Числовые типы:** INT, BIGINT, SMALLINT, TINYINT, DECIMAL, NUMERIC, FLOAT, REAL.
  + **Строковые типы:** CHAR, VARCHAR, TEXT, NCHAR, NVARCHAR, NTEXT.
  + **Дата и время:** DATE, TIME, DATETIME, SMALLDATETIME, DATETIME2, DATETIMEOFFSET.
  + **Логический тип:** BIT.
  + **Двоичные типы:** BINARY, VARBINARY.
  + **Уникальные идентификаторы:** UNIQUEIDENTIFIER.

**Пример:** Создание таблицы с различными типами данных:

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

Name NVARCHAR(100),

BirthDate DATE,

Salary DECIMAL(18, 2),

IsActive BIT

);

1. **Язык Transact-SQL. Функции языка SQL.**

T-SQL предоставляет обширный набор встроенных функций для работы с данными. Функции можно разделить на несколько категорий:

* + **Агрегатные функции:** COUNT(), SUM(), AVG(), MIN(), MAX().
  + **Строковые функции:** LEN(), SUBSTRING(), UPPER(), LOWER(), REPLACE().
  + **Функции даты и времени:** GETDATE(), DATEADD(), DATEDIFF(), FORMAT().
  + **Логические функции:** CASE, COALESCE(), NULLIF().
  + **Конвертации типов:** CAST(), CONVERT().

**Пример:** Использование функции GETDATE() для получения текущей даты и времени:

SELECT GETDATE() AS CurrentDateTime;

1. **Язык Transact-SQL. Скалярные операторы.**

Скалярные операторы в T-SQL используются для выполнения операций над одиночными значениями. Основные категории скалярных операторов:

* + **Арифметические операторы:** +, -, \*, /, %.
  + **Операторы сравнения:** =, <>, >, <, >=, <=.
  + **Логические операторы:** AND, OR, NOT.
  + **Операторы строк:** + (конкатенация строк).

**Пример:** Использование арифметического оператора для вычисления зарплаты с учетом бонуса:

SELECT Name, Salary, Salary \* 0.10 AS Bonus

FROM Employees;

1. **Язык Transact-SQL. Значения NULL.**

В T-SQL значение NULL представляет отсутствие данных или неизвестное значение. Работа с NULL имеет особенности:

* + **Сравнение с NULL:** Операторы сравнения не работают с NULL. Для проверки на NULL используются IS NULL и IS NOT NULL.
  + **Функции обработки NULL:** ISNULL(), COALESCE().

**Пример:** Использование функции ISNULL() для замены NULL на значение по умолчанию:

SELECT Name, ISNULL(Salary, 0) AS Salary

FROM Employees;

1. **Стандарты ISO/IEC 17799:2002 (BS 7799:2000). Германский стандарт BSI. Международный стандарт ISO 15408 «Общие критерии безопасности информационных технологий». Российские стандарты безопасности информационных технологий.**
   * **ISO/IEC 17799:2002 (BS 7799:2000):** Международный стандарт, предоставляющий рекомендации по управлению безопасностью информации. Он охватывает организационные аспекты, такие как политика безопасности, организация безопасности, управление активами и т.д.

* **BSI (Bundesamt für Sicherheit in der Informationstechnik):** Германское федеральное ведомство по безопасности информационных технологий, разрабатывающее стандарты и рекомендации по обеспечению ИБ.
* **ISO 15408:** Международный стандарт, известный как "Общие критерии безопасности информационных технологий", предоставляет основу для оценки и сертификации безопасности ИТ-продуктов.
* **Российские стандарты:** В России вопросы безопасности информации регулируются стандартами, такими как ГОСТ Р 50922-96 "Система защиты информации. Основные положения" и другими нормативными документами.

1. **Проблемы информационной безопасности баз данных.**

Основные проблемы ИБ баз данных включают:

* + **Неавторизованный доступ:** Неправомерный доступ к данным может привести к утечке конфиденциальной информации.
  + **Атаки на базы данных:** SQL-инъекции и другие виды атак могут эксплуатировать уязвимости в приложениях и СУБД.
  + **Нарушение целостности данных:** Несанкционированные изменения данных могут привести к искажению информации.
  + **Отказ в обслуживании:** Атаки, направленные на недоступность базы данных для пользователей.

1. **Особенности защиты БД.**

Защита баз данных (БД) является ключевым аспектом обеспечения информационной безопасности в организациях. Основные особенности защиты БД включают:

* + **Защита от несанкционированного доступа:** Ограничение доступа к данным только для авторизованных пользователей с использованием механизмов аутентификации и авторизации.
  + **Обеспечение целостности данных:** Предотвращение несанкционированных изменений и обеспечение точности и согласованности данных.
  + **Обеспечение доступности данных:** Гарантия постоянного и надежного доступа к данным для авторизованных пользователей, включая защиту от атак типа "отказ в обслуживании" (DoS).
  + **Защита от программно-аппаратных ошибок:** Обеспечение устойчивости БД к сбоям и ошибкам, предотвращение потери или повреждения данных из-за технических проблем.
  + **Аудит и мониторинг:** Ведение журналов доступа и изменений данных для отслеживания действий пользователей и обнаружения возможных нарушений безопасности.
  + **Резервное копирование и восстановление:** Регулярное создание резервных копий данных и обеспечение их восстановления в случае утраты или повреждения.

Эти меры направлены на всестороннюю защиту данных и поддержание доверия к системам управления базами данных.

1. **Основные требования к безопасности БД.**

Основные требования к безопасности баз данных включают:

* + **Конфиденциальность:** Обеспечение доступа к данным только для авторизованных пользователей.
  + **Целостность:** Гарантия точности и полноты данных, предотвращение их несанкционированных изменений.
  + **Доступность:** Обеспечение постоянного и надежного доступа к данным для авторизованных пользователей.
  + **Аудит:** Ведение журнала операций с данными для отслеживания действий пользователей и выявления возможных нарушений.
  + **Резервное копирование:** Регулярное создание копий данных для восстановления информации в случае ее утраты или повреждения.

**Пример:** Внедрение системы управления доступом на основе ролей (RBAC) в SSMS позволяет эффективно управлять правами пользователей, обеспечивая конфиденциальность и целостность данных.

1. **Аспекты создания защищённых БД.**

Создание защищенных баз данных требует учета следующих аспектов:

* + **Проектирование безопасности:** Внедрение механизмов аутентификации и авторизации, использование шифрования данных и обеспечение физической безопасности серверов.
  + **Управление доступом:** Определение и контроль прав доступа пользователей, применение принципа наименьших привилегий.
  + **Мониторинг и аудит:** Регулярный анализ активности базы данных, ведение логов и проведение аудита для обнаружения подозрительных действий.
  + **Обучение персонала:** Повышение осведомленности сотрудников о принципах информационной безопасности и регулярное проведение тренингов.

**Пример:** В SSMS можно настроить аудит доступа к чувствительным данным, что позволит своевременно обнаруживать и реагировать на несанкционированные попытки доступа.

1. **Характеристики качества систем баз данных.**

Качество систем баз данных оценивается по следующим характеристикам:

* + **Производительность:** Способность системы эффективно обрабатывать запросы и транзакции.
  + **Масштабируемость:** Возможность системы адаптироваться к увеличению объема данных и нагрузки.
  + **Надежность:** Способность системы сохранять работоспособность и корректность данных при сбоях.
  + **Безопасность:** Защищенность данных от несанкционированного доступа и модификаций.
  + **Удобство использования:** Интуитивно понятный интерфейс и доступность инструментов для администрирования и разработки.

**Пример:** SSMS предоставляет инструменты для мониторинга производительности и управления ресурсами, что способствует поддержанию высокого качества работы базы данных.

1. **Структура свойств информационной безопасности баз данных.**

Свойства информационной безопасности баз данных включают:

* + **Конфиденциальность:** Защита данных от несанкционированного доступа.
  + **Целостность:** Обеспечение точности и полноты данных, предотвращение их искажений.
  + **Доступность:** Гарантия своевременного доступа к данным для авторизованных пользователей.
  + **Аудитируемость:** Возможность отслеживания и записи операций с данными для обеспечения прозрачности и контроля.
  + **Восстановимость:** Способность быстро восстановить данные после сбоев или утрат.

**Пример:** Внедрение механизмов шифрования и резервного копирования в SSMS способствует обеспечению конфиденциальности и доступности данных.

1. **Концепция баз данных.**

База данных — это организованная совокупность данных, хранимая и управляемая системой управления базами данных (СУБД). Основные концепции баз данных включают:

* + **Модели данных:** Определяют структуру и организацию данных (например, реляционная, объектно-ориентированная модели).
  + **СУБД:** Программное обеспечение, обеспечивающее создание, управление и обслуживание баз данных.
  + **Языки запросов:** Интерфейсы для взаимодействия с базой данных, такие как SQL.

**Пример:** SSMS использует реляционную модель данных и предоставляет инструменты для управления базами данных SQL Server.

1. **Система управления базами данных. Основные функции.**

Система управления базами данных (СУБД) — это комплекс программных средств, обеспечивающих создание, хранение, изменение и извлечение данных из базы данных. Основные функции СУБД:

* + **Определение структуры данных:** Проектирование схемы базы данных и определение типов данных.
  + **Управление данными:** Вставка, обновление, удаление и извлечение данных.
  + **Обеспечение целостности:** Поддержание корректности и согласованности данных.
  + **Управление доступом:** Контроль прав доступа пользователей к данным.
  + **Резервное копирование и восстановление:** Защита данных от потери и обеспечение их восстановления после сбоев.

**Пример:** SSMS выполняет все перечисленные функции для баз данных SQL Server, обеспечивая эффективное управление данными.

1. **Архитектура СУБД.**

Архитектура системы управления базами данных (СУБД) описывает ее структурные компоненты и их взаимодействие, обеспечивая эффективное управление данными и их обработку. Основные аспекты архитектуры СУБД включают:

* **Уровни архитектуры:** СУБД обычно проектируются с учетом многослойной архитектуры, разделяя функциональные компоненты на различные уровни для обеспечения независимости и гибкости.
* **Компоненты системы:** Ключевые элементы СУБД включают ядро системы, процессор языка запросов, менеджер памяти, менеджер транзакций и другие подсистемы, каждая из которых отвечает за определенные аспекты работы с базой данных.
* **Взаимодействие с пользователем:** Предоставление интерфейсов для пользователей и приложений, включая командные оболочки, графические интерфейсы и API, для эффективного взаимодействия с данными.

Понимание архитектуры СУБД важно для администрирования, разработки и оптимизации работы с базами данных, обеспечивая их производительность, надежность и безопасность.

Продолжим ответы на вопросы, начиная с 26-го:

1. **Проектирование баз данных: Уровни представления моделей данных.**

Проектирование баз данных включает несколько уровней представления данных, каждый из которых играет свою роль в создании эффективной и надежной базы данных:

* + **Концептуальный уровень:** Отражает общую логическую структуру базы данных, описывая сущности, их атрибуты и связи между ними, без учета технических деталей реализации.
  + **Логический уровень:** Представляет данные в терминах выбранной модели данных (например, реляционной), уточняя структуру таблиц, ключей и связей.
  + **Физический уровень:** Определяет, как данные физически хранятся на носителях, включая индексы, распределение данных и другие аспекты производительности.

**Пример:** При проектировании базы данных для учета сотрудников на концептуальном уровне определяются сущности "Сотрудник", "Отдел" и их связи. На логическом уровне эти сущности преобразуются в таблицы с соответствующими атрибутами и ключами. На физическом уровне учитываются вопросы хранения данных для обеспечения быстрого доступа и надежности.

1. **Концептуальное (инфологическое) моделирование предметной области: Предметная область.**

Предметная область — это область реального мира, информацию о которой необходимо хранить и обрабатывать в базе данных. Понимание предметной области критично для создания эффективной модели данных.

**Пример:** В системе учета библиотеки предметная область включает книги, авторов, читателей и заказы.

1. **Концептуальное (инфологическое) моделирование предметной области: Требования, предъявляемые к концептуальной модели.**

Концептуальная модель должна быть:

* + **Полной:** Отражать все важные аспекты предметной области.
  + **Консистентной:** Не содержать противоречий.
  + **Понятной:** Быть легко воспринимаемой и понятной для всех заинтересованных сторон.
  + **Гибкой:** Легко адаптироваться к изменениям в предметной области.

1. **Компоненты инфологической модели.**

Инфологическая модель описывает структуру данных на концептуальном уровне и включает:

* + **Сущности:** Объекты или понятия, имеющие значение в предметной области.
  + **Атрибуты:** Характеристики сущностей.
  + **Связи:** Отношения между сущностями.
  + **Ограничения:** Правила, определяющие допустимые значения атрибутов и связи между сущностями.

1. **Построение модели «сущность-связь».**

Модель «сущность-связь» (Entity-Relationship, ER-модель) используется для графического представления инфологической модели. В этой модели:

* + **Сущности** представляются прямоугольниками.
  + **Атрибуты** — овалами, соединенными с соответствующими сущностями.
  + **Связи** — ромбами, соединяющими сущности, с указанием типа связи (один к одному, один ко многим и т.д.).

**Пример:** В ER-диаграмме для библиотеки сущности "Книга" и "Автор" могут быть связаны отношением "написана", где указывается, что один автор может написать несколько книг, а каждая книга имеет одного автора.

1. **Основные понятия ER-диаграмм.**

ER-диаграммы являются инструментом визуализации структуры базы данных и включают следующие основные компоненты:

* + **Сущности:** Объекты, о которых собирается информация.
  + **Атрибуты:** Свойства или характеристики сущностей.
  + **Связи:** Логические отношения между сущностями.
  + **Кардинальность связи:** Определяет количество экземпляров одной сущности, связанных с экземплярами другой сущности (например, один к одному, один ко многим).

1. **Даталогическое проектирование: общие понятия.**

Даталогическое проектирование фокусируется на преобразовании концептуальной модели в логическую, учитывая особенности выбранной модели данных (например, реляционной). Основные понятия включают:

* + **Таблицы:** Представление сущностей в реляционной модели.
  + **Столбцы и строки:** Атрибуты и записи таблиц соответственно.
  + **Ключи:** Первичные и внешние ключи для идентификации записей и установления связей между таблицами.

1. **Нормальные формы ER-схем.**

Нормальные формы (НФ) — это уровни нормализации базы данных, направленные на устранение избыточности и предотвращение аномалий при обновлении данных. Основные нормальные формы:

* + **Первая нормальная форма (1НФ):** Требует, чтобы все атрибуты содержали атомарные (неделимые) значения.
  + **Вторая нормальная форма (2НФ):** База данных находится в 1НФ и все атрибуты зависят от всего первичного ключа.
  + **Третья нормальная форма (3НФ):** База данных находится во 2НФ и нет транзитивной зависимости между атрибутами.

1. **Язык описания данных. Создание объектов баз данных.**

В языке SQL для создания объектов базы данных, таких как таблицы, представления и индексы, используются команды языка описания данных (DDL). Команда CREATE позволяет определить структуру и характеристики этих объектов.

**Примеры создания объектов:**

* + **Создание таблицы:**
  + CREATE TABLE Employees (
  + EmployeeID INT PRIMARY KEY,
  + Name NVARCHAR(100) NOT NULL,
  + Position NVARCHAR(100),
  + HireDate DATE
  + );

В этом примере создается таблица Employees с четырьмя столбцами: EmployeeID, Name, Position и HireDate. Столбец EmployeeID является первичным ключом, а Name не может содержать значения NULL.

* + **Создание индекса:**
  + CREATE INDEX IX\_EmployeeName
  + ON Employees (Name);

Здесь создается индекс IX\_EmployeeName на столбце Name таблицы Employees, что ускоряет операции поиска по этому столбцу.

* + **Создание представления:**
  + CREATE VIEW EmployeeView AS
  + SELECT Name, Position
  + FROM Employees
  + WHERE HireDate > '2020-01-01';

В этом примере создается представление EmployeeView, которое отображает имена и должности сотрудников, нанятых после 1 января 2020 года.

1. **Модифицирование объектов баз данных. Удаление объектов баз данных.**

Для изменения и удаления объектов базы данных в SQL используются команды DDL, такие как ALTER и DROP.

**Примеры модификации и удаления объектов:**

* + **Изменение таблицы (добавление столбца):**
  + ALTER TABLE Employees
  + ADD Email NVARCHAR(255);

Эта команда добавляет новый столбец Email в таблицу Employees.

* + **Удаление столбца из таблицы:**
  + ALTER TABLE Employees
  + DROP COLUMN Email;

Команда удаляет столбец Email из таблицы Employees.

* + **Удаление таблицы:**
  + DROP TABLE Employees;

Эта команда полностью удаляет таблицу Employees из базы данных.

1. **Язык описания данных. Создание объектов баз данных.**

Этот вопрос повторяет предыдущий (пункт 34). Рекомендуется ознакомиться с ответом на пункт 34 для получения информации о создании объектов баз данных с помощью DDL.

1. **Инструкция CREATE TABLE и ограничения декларативной целостности.**

Инструкция CREATE TABLE используется для создания новой таблицы в базе данных и определения ее структуры, включая столбцы и их типы данных. Ограничения декларативной целостности, такие как PRIMARY KEY, FOREIGN KEY, NOT NULL, UNIQUE и CHECK, обеспечивают сохранение целостности и корректности данных.

**Пример использования CREATE TABLE с ограничениями:**

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

Name NVARCHAR(100) NOT NULL,

Position NVARCHAR(100),

HireDate DATE CHECK (HireDate >= '2000-01-01'),

Salary DECIMAL(10, 2) CHECK (Salary > 0)

);

В этом примере:

* + EmployeeID является первичным ключом, обеспечивая уникальность каждой записи.
  + Name не может быть NULL.
  + HireDate должно быть не ранее 1 января 2000 года.
  + Salary должно быть положительным числом.

Ограничения CHECK используются для определения допустимых значений для столбцов, например, для проверки диапазона значений.

1. **Предложение UNIQUE.**

Предложение UNIQUE в SQL используется для обеспечения уникальности значений в одном или нескольких столбцах таблицы. Оно гарантирует, что все значения в указанном столбце или комбинации столбцов будут различными.

**Пример использования UNIQUE:**

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

Name NVARCHAR(100) NOT NULL,

Email NVARCHAR(255) UNIQUE

);

В этом примере столбец Email имеет ограничение UNIQUE, что означает, что все адреса электронной почты в таблице должны быть уникальными.

1. **Предложение CHECK.**

Предложение CHECK в SQL используется для определения ограничения, которое проверяет, соответствует ли значение в столбце заданному условию. Это позволяет ограничить диапазон допустимых значений для столбца.

**Пример использования CHECK:**

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

Name NVARCHAR(100) NOT NULL,

Age INT CHECK (Age >= 18 AND Age <= 65)

);

В этом примере ограничение CHECK на столбце Age обеспечивает, что возраст сотрудника будет в пределах от 18 до 65 лет.

1. **Ссылочная целостность.**

Ссылочная целостность в базе данных гарантирует, что отношения между таблицами остаются согласованными. Она обеспечивает, чтобы значения в одном столбце, ссылающемся на другой столбец (обычно в другой таблице), соответствовали существующим значениям в целевой

Продолжим ответы на оставшиеся вопросы:

**41. Язык описания данных. Модифицирование объектов баз данных.**

Для изменения существующих объектов базы данных в SQL используются команды DDL, такие как ALTER. С их помощью можно добавлять, изменять или удалять столбцы в таблицах, а также изменять другие характеристики объектов.

**Примеры модификации объектов:**

* **Изменение таблицы (добавление нового столбца):**

ALTER TABLE Employees

ADD Email NVARCHAR(255);

Эта команда добавляет новый столбец Email в таблицу Employees.

* **Изменение существующего столбца (изменение типа данных):**

ALTER TABLE Employees

ALTER COLUMN Salary DECIMAL(15, 2);

Команда изменяет тип данных столбца Salary на DECIMAL с точностью до 15 знаков и 2 знаков после запятой.

* **Удаление столбца из таблицы:**

ALTER TABLE Employees

DROP COLUMN Email;

Эта команда удаляет столбец Email из таблицы Employees.

**42. Изменение таблиц. Добавление и удаление столбцов.**

Добавление и удаление столбцов в таблицах осуществляется с помощью команды ALTER TABLE.

**Добавление столбца:**

ALTER TABLE Employees

ADD DateOfBirth DATE;

Команда добавляет новый столбец DateOfBirth типа DATE в таблицу Employees.

**Удаление столбца:**

ALTER TABLE Employees

DROP COLUMN DateOfBirth;

Эта команда удаляет столбец DateOfBirth из таблицы Employees.

**43. Изменение свойств столбцов.**

Изменение свойств столбцов, таких как тип данных или ограничения, выполняется с помощью команды ALTER TABLE.

**Пример изменения типа данных столбца:**

ALTER TABLE Employees

ALTER COLUMN Name NVARCHAR(200);

Команда изменяет тип данных столбца Name на NVARCHAR длиной 200 символов.

**44. Язык описания данных. Удаление объектов баз данных.**

Удаление объектов базы данных, таких как таблицы, представления или индексы, осуществляется с помощью команды DROP.

**Примеры удаления объектов:**

* **Удаление таблицы:**

DROP TABLE Employees;

Эта команда удаляет таблицу Employees из базы данных.

* **Удаление представления:**

DROP VIEW EmployeeView;

Команда удаляет представление EmployeeView.

* **Удаление индекса:**

DROP INDEX IX\_EmployeeName ON Employees;

Эта команда удаляет индекс IX\_EmployeeName на таблице Employees.

**45. Инструкция SELECT. Ее предложения и функции.**

Инструкция SELECT используется для извлечения данных из базы данных. Она позволяет выбирать столбцы, фильтровать строки, группировать данные и сортировать результаты.

**Основные компоненты SELECT:**

* **FROM:** Определяет таблицу или представление, из которых будут извлекаться данные.
* **WHERE:** Фильтрует строки на основе заданного условия.
* **GROUP BY:** Группирует строки по указанным столбцам.
* **HAVING:** Фильтрует группы после применения GROUP BY.
* **ORDER BY:** Сортирует результаты по указанным столбцам.

**Пример использования SELECT:**

SELECT Name, Position, Salary

FROM Employees

WHERE Salary > 50000

ORDER BY Salary DESC;

Этот запрос выбирает имена, должности и зарплаты сотрудников с зарплатой более 50 000, сортируя результаты по убыванию зарплаты.

**46. Подзапросы.**

Подзапрос (или вложенный запрос) — это запрос, вложенный внутри другого SQL-запроса. Он позволяет использовать результат выполнения внутреннего запроса во внешнем запросе.

**Пример подзапроса в условии WHERE:**

SELECT Name, Position, Salary

FROM Employees

WHERE DepartmentID = (

SELECT DepartmentID

FROM Departments

WHERE DepartmentName = 'Sales'

);

Этот запрос выбирает имена, должности и зарплаты сотрудников, работающих в отделе "Sales".

**47. Временные таблицы.**

Временные таблицы используются для хранения промежуточных результатов на время сессии или до закрытия соединения с базой данных.

**Создание временной таблицы:**

CREATE TABLE #TempEmployees (

EmployeeID INT,

Name NVARCHAR(100),

Position NVARCHAR(100)

);

Эта команда создает временную таблицу #TempEmployees с тремя столбцами: EmployeeID, Name и Position.

**Использование временной таблицы:**

INSERT INTO #TempEmployees (EmployeeID, Name, Position)

SELECT EmployeeID, Name, Position

FROM Employees

WHERE Salary > 50000;

Запрос вставляет в временную таблицу #TempEmployees данные сотрудников с зарплатой более 50 000.

Продолжим ответы на оставшиеся вопросы:

**48. Операторы соединения.**

Операторы соединения (JOIN) используются для объединения строк из двух или более таблиц на основе связанного столбца между ними. В SQL существуют различные типы соединений:

* **INNER JOIN:** Возвращает строки, которые имеют совпадения в обеих таблицах.
* **LEFT JOIN (или LEFT OUTER JOIN):** Возвращает все строки из левой таблицы и совпадающие строки из правой таблицы. Если совпадений нет, результат содержит NULL для правой таблицы.
* **RIGHT JOIN (или RIGHT OUTER JOIN):** Возвращает все строки из правой таблицы и совпадающие строки из левой таблицы. Если совпадений нет, результат содержит NULL для левой таблицы.
* **FULL JOIN (или FULL OUTER JOIN):** Возвращает строки, когда есть совпадение в одной из таблиц.

**Пример использования INNER JOIN:**

SELECT Employees.Name, Departments.DepartmentName

FROM Employees

INNER JOIN Departments ON Employees.DepartmentID = Departments.DepartmentID;

Этот запрос возвращает имена сотрудников и названия их отделов, объединяя таблицы Employees и Departments по полю DepartmentID.

**49. Связанные подзапросы.**

Связанные подзапросы (correlated subqueries) — это подзапросы, которые ссылаются на столбцы внешнего запроса. Каждый раз при выполнении внешнего запроса внутренний подзапрос выполняется заново.

**Пример связанного подзапроса:**

SELECT Name, Salary

FROM Employees e1

WHERE Salary > (

SELECT AVG(Salary)

FROM Employees e2

WHERE e1.DepartmentID = e2.DepartmentID

);

В этом примере внешний запрос выбирает имена и зарплаты сотрудников, чья зарплата выше средней в их отделе.

**50. Табличные выражения.**

Табличные выражения, такие как обобщенные табличные выражения (CTE), позволяют создавать временные наборы результатов, которые можно использовать в последующих запросах.

**Пример использования CTE:**

WITH DepartmentCTE AS (

SELECT DepartmentID, AVG(Salary) AS AvgSalary

FROM Employees

GROUP BY DepartmentID

)

SELECT e.Name, e.Salary, d.AvgSalary

FROM Employees e

INNER JOIN DepartmentCTE d ON e.DepartmentID = d.DepartmentID

WHERE e.Salary > d.AvgSalary;

В этом примере CTE DepartmentCTE вычисляет среднюю зарплату по каждому отделу, а основной запрос выбирает сотрудников, чья зарплата выше средней по их отделу.

Эти концепции являются основными при работе с базами данных и их безопасностью, обеспечивая эффективное управление данными и защиту от различных угроз.