Турсунов Баходурхон

**1. Приведите примеры команд DDL, таких как CREATE, ALTER, DROP, TRUNCATE, COMMENT и RENAME, и объясните их соответствующие функции.**

1. **CREATE**: Создание нового объекта в базе данных, такого как таблицы, индексы, представления или хранимые процедуры. Пример:

CREATE TABLE Employees ( ID INT PRIMARY KEY, Name VARCHAR(100), Age INT );

1. **ALTER**: Изменение структуры существующего объекта базы данных, например, добавление, удаление или изменение столбцов в таблице. Пример:

ALTER TABLE Employees ADD COLUMN Email VARCHAR(255);

1. **DROP**: Удаление объекта базы данных, такого как таблицы, индекса или представления. Пример:

DROP TABLE Employees;

1. **TRUNCATE**: Удаление всех записей из таблицы, при этом сама структура таблицы остаётся нетронутой. Пример:

TRUNCATE TABLE Employees;

1. **COMMENT**: Добавление комментария к объекту базы данных, чтобы обеспечить дополнительную документацию или описание. Пример:

COMMENT ON TABLE Employees IS 'This table stores information about employees.';

1. **RENAME**: Переименование существующего объекта базы данных. Пример:

ALTER TABLE Employees RENAME TO Staff;

Команды DDL играют важную роль в управлении структурой базы данных, позволяя создавать, изменять и удалять объекты, а также предоставлять комментарии и переименовывать объекты для облегчения работы с базой данных.

**2. Объясните основные функции команд DML, включая SELECT, INSERT, UPDATE и DELETE, и то, как они способствуют управлению данными в реляционной базе данных.**

1. **SELECT**: Команда SELECT используется для извлечения данных из базы данных. Она позволяет выбирать определенные столбцы или строки из одной или нескольких таблиц, а также применять функции агрегирования, сортировку и фильтрацию данных. Пример:

SELECT \* FROM Employees WHERE Department = 'IT';

1. **INSERT**: Команда INSERT используется для добавления новых строк данных в таблицу. Она позволяет вставлять значения в определенные столбцы или вставлять значения по умолчанию для столбцов, которые не требуют явного указания. Пример:

INSERT INTO Employees (Name, Department, Salary) VALUES ('John Doe', 'IT', 50000);

1. **UPDATE**: Команда UPDATE используется для изменения существующих строк данных в таблице. Она позволяет обновлять значения определенных столбцов для одной или нескольких строк, соответствующих определенным условиям. Пример:

UPDATE Employees SET Salary = 55000 WHERE Name = 'John Doe';

1. **DELETE**: Команда DELETE используется для удаления одной или нескольких строк данных из таблицы. Она позволяет удалять строки, соответствующие определенным условиям, либо все строки из таблицы. Пример:

DELETE FROM Employees WHERE Name = 'John Doe';

Команды DML играют ключевую роль в управлении данными в реляционной базе данных, позволяя добавлять, изменять и удалять данные в таблицах. Это основные инструменты для манипулирования данными и обеспечения их целостности и актуальности.

**3.Выделите особенности, отличающие PostgreSQL, такие как поддержка расширенных типов данных и процедурных языков. Также упомяните примечательные особенности MySQL и SQL Server.**

**PostgreSQL:**

1. **Поддержка расширенных типов данных:** PostgreSQL предлагает богатый набор встроенных и расширенных типов данных, включая географические объекты, JSON, XML, массивы, пользовательские типы и многое другое. Это позволяет разработчикам хранить и оперировать данными более гибко.
2. **Процедурные языки:** PostgreSQL поддерживает несколько процедурных языков, таких как PL/pgSQL, PL/Python, PL/Perl, PL/Java и другие. Это позволяет разработчикам создавать хранимые процедуры, функции и триггеры на различных языках программирования.
3. **Транзакционная надежность:** PostgreSQL обеспечивает высокий уровень надежности и целостности данных за счет поддержки ACID-свойств (атомарность, согласованность, изолированность, долговечность) в транзакциях.

**MySQL:**

1. **Простота использования:** MySQL обычно считается более простым в установке, настройке и использовании по сравнению с некоторыми другими СУБД. Он имеет интуитивно понятный синтаксис SQL и хорошо документирован.
2. **Высокая производительность:** MySQL обеспечивает высокую производительность благодаря оптимизации запросов, использованию индексов и другим методам оптимизации.
3. **Широкая поддержка:** MySQL имеет обширное сообщество пользователей и разработчиков, а также большое количество сторонних инструментов и расширений.

**SQL Server:**

1. **Интеграция с экосистемой Microsoft:** SQL Server тесно интегрируется с другими продуктами и технологиями от Microsoft, такими как .NET Framework, Visual Studio, Azure и др. Это обеспечивает удобство в разработке, развертывании и управлении приложениями.
2. **Enterprise функциональность:** SQL Server предоставляет широкий спектр возможностей для корпоративного использования, включая поддержку аналитики, отчетности, репликации, безопасности и масштабируемости.
3. **Высокая производительность и масштабируемость:** SQL Server обладает возможностями для обработки больших объемов данных и высоких нагрузок, а также поддерживает кластеризацию и репликацию для обеспечения высокой доступности и отказоустойчивости.

Каждая из этих систем управления базами данных обладает своими особенностями и преимуществами, которые могут быть важны в различных сценариях разработки и использования.

**4. Объясните концепцию движков хранения данных в MySQL и то, как они влияют на поддержку транзакций и производительность.**

Движки хранения данных (storage engines) в MySQL — это программные модули, которые отвечают за хранение, обработку и доступ к данным в MySQL. Каждый движок предоставляет свои собственные особенности, возможности и ограничения. Два наиболее распространенных движка в MySQL — это InnoDB и MyISAM.

**InnoDB:**

* **Поддержка транзакций:** InnoDB является транзакционным движком, что означает, что он поддерживает транзакции с ACID-свойствами (Atomicity, Consistency, Isolation, Durability). Это позволяет выполнять операции в рамках транзакции и гарантирует целостность данных даже при возникновении ошибок или сбоев.
* **Ссылочная целостность:** InnoDB поддерживает ограничения ссылочной целостности, такие как внешние ключи, что обеспечивает целостность данных и помогает предотвращать вставку некорректных значений в связанные таблицы.
* **MVCC (Multi-Version Concurrency Control):** Этот механизм обеспечивает изоляцию транзакций, позволяя одной транзакции видеть снимок данных на момент ее начала, что позволяет избежать блокировок чтения и улучшает параллелизм запросов.

**MyISAM:**

* **Отсутствие поддержки транзакций:** MyISAM не поддерживает транзакции, что означает, что операции INSERT, UPDATE и DELETE не могут быть отменены или применены частично в случае сбоя.
* **Блокировка на уровне таблиц:** В отличие от InnoDB, который использует блокировки на уровне строк, MyISAM использует блокировки на уровне таблицы. Это может привести к проблемам с параллелизмом запросов и конкуренцией при одновременном доступе к таблице.
* **Лучшая производительность для чтения:** MyISAM часто обладает лучшей производительностью для операций чтения, так как блокировки на уровне таблицы более просты и затраты на управление транзакциями отсутствуют.

В целом, выбор между InnoDB и MyISAM зависит от требований к вашей базе данных. Если вам необходима поддержка транзакций, ссылочной целостности и хорошая производительность для операций записи, то предпочтительнее использовать InnoDB. Однако, если вы работаете с приложением, где операции чтения преобладают над операциями записи, и вам важна максимальная производительность, то MyISAM может быть более подходящим выбором.

**5. Обсудите поддержку типов данных JSON и JSONB в PostgreSQL и их значение. Как SQL Server расширил свою совместимость с различными платформами и каковы его исторические ассоциации с точки зрения операционных систем?**

В PostgreSQL поддерживаются два типа данных для работы с JSON: JSON и JSONB.

**JSON:** это тип данных, который хранит JSON-данные в текстовом формате. Он обеспечивает базовые операции над JSON-данными, такие как извлечение, вставка и обновление. Однако он не обеспечивает эффективного хранения и операций с индексами.

**JSONB:** это бинарное представление JSON-данных, которое обеспечивает более эффективное хранение, компрессию и индексацию JSON-данных. JSONB позволяет выполнять запросы и операции поиска на JSON-данных с использованием индексов, что делает его более производительным для работы с большими объемами JSON-данных.

Значение типов данных JSON и JSONB в PostgreSQL заключается в том, что они позволяют сохранять и манипулировать структурированными данными в формате JSON прямо в базе данных. Это особенно полезно в современных приложениях, где JSON широко используется для обмена данными между клиентской и серверной сторонами, а также для хранения сложных структур данных.

**SQL Server:** SQL Server, начиная с версии 2016, также поддерживает тип данных JSON. Это позволяет хранить JSON-данные в таблицах SQL Server и выполнять операции с ними с помощью специальных функций и запросов.

Относительно исторических ассоциаций SQL Server с операционными системами, следует отметить, что SQL Server был разработан Microsoft и изначально выпускался только для платформы Windows. Однако в более поздних версиях, начиная с SQL Server 2016, Microsoft расширила совместимость SQL Server с различными платформами, включая Linux и контейнеры Docker. Это позволяет разработчикам выбирать платформу, которая лучше всего подходит для их конкретных потребностей и предпочтений.

**6. Объясните основные компоненты таблицы, такие как столбцы и строки. Какую информацию представляет столбец и из чего состоит строка?**

В реляционных базах данных таблица представляет собой структурированную коллекцию данных, организованную в виде строк и столбцов. Вот основные компоненты таблицы:

1. **Столбцы (Columns):** Столбцы представляют собой вертикальные структуры данных в таблице. Каждый столбец имеет имя и тип данных, определяющий тип информации, которая может быть сохранена в этом столбце. Например, столбец "Name" может содержать имена сотрудников в виде строковых значений, а столбец "Age" может содержать возраст сотрудников в виде числовых значений. Каждый столбец также может иметь свойства, такие как ограничения (например, уникальность или внешние ключи) и атрибуты (например, автоинкремент для генерации уникальных идентификаторов).
2. **Строки (Rows):** Строки представляют собой горизонтальные структуры данных в таблице. Каждая строка представляет отдельную запись или кортеж данных, содержащую значения для каждого столбца в таблице. Например, строка может представлять собой запись о сотруднике, содержащую его имя, возраст, должность и т. д. Количество строк в таблице определяет количество записей данных, которые содержит таблица.

Итак, каждый столбец представляет определенный тип информации, а каждая строка содержит значения этой информации для конкретной записи или объекта. Вместе столбцы и строки образуют таблицу, которая представляет собой коллекцию данных, организованных в структурированную форму для удобного хранения, поиска и обработки.

**7. Дайте определение транзакции в контексте реляционной базы данных и объясните ее важность для поддержания согласованности и целостности.**

В контексте реляционной базы данных транзакция представляет собой логическую единицу работы, которая состоит из одного или нескольких SQL операторов. Транзакции обладают четырьмя основными свойствами, известными как ACID:

1. **Атомарность (Atomicity):** это означает, что транзакция должна быть выполнена целиком либо не выполнена вообще. Если хотя бы один из SQL операторов в транзакции завершается с ошибкой, то все изменения, внесенные этой транзакцией, должны быть отменены (откат), чтобы сохранить базу данных в консистентном состоянии.
2. **Согласованность (Consistency):** Транзакция должна приводить базу данных из одного согласованного состояния в другое согласованное состояние. Это означает, что транзакция должна соблюдать все ограничения целостности данных, определенные в базе данных.
3. **Изолированность (Isolation):** Каждая транзакция должна выполняться изолированно от других транзакций, что означает, что результаты выполнения транзакции не должны быть видны другим транзакциям до ее окончания и фиксации (commit). Это гарантирует, что даже в многопользовательской среде транзакции не будут взаимно влиять друг на друга.
4. **Долговечность (Durability):** после успешного завершения транзакции изменения, внесенные в базу данных, должны оставаться постоянными и доступными даже в случае сбоя системы или перезагрузки. Это обеспечивает надежность базы данных и защищает данные от потери.

Важность транзакций для поддержания согласованности и целостности данных в реляционной базе данных трудно переоценить. Они обеспечивают надежность операций записи, гарантируют, что данные остаются в консистентном состоянии, предотвращают взаимные влияния между параллельными операциями и обеспечивают долговременное сохранение данных даже в случае сбоев системы. Без использования транзакций операции с данными могут стать непредсказуемыми и могут привести к потере или повреждению данных, что недопустимо для большинства приложений и систем.

Practice:

- Напишите SQL-запрос, который выберет все данные из таблицы "student", отсортированные по столбцу "name" в алфавитном порядке. Ограничьте выборку только 5 строками. Кроме того, выполните псевдонимы для столбца "age" и назовите его "student\_age".

Вам необходимо написать SQL-запрос, удовлетворяющий следующим условиям:

Выбрать все данные из таблицы "student":

1. отсортировать выбранные данные по столбцу "name" в алфавитном порядке.

2. Ограничить выборку только 5 строками.

3. Выполните псевдонимы для столбца "age" и назовите его "student\_age".

Необходимо написать SQL-запрос, который выбирает следующую информацию из таблицы "student":

- id (уникальный идентификатор студента)

- name

- age (возраст студента, переименованный в student\_age)

select id, name, age as student\_age from Student

order by name

LIMIT 5

Доп задание:

1. Создать таблицу для сущности Risk из проекта ERM.

CREATE TABLE RiskProfile(

Id INT PRIMARY KEY autoincrement,

RiskName VARCHAR(50) NOT NULL,

Description VARCHAR(500) NOT NULL,

OccurenceProbability SMALLINT NOT NULL,

PotentialBusinessImpact SMALLINT NOT NULL

)