1. Дайте определение понятия “триггер”. Опишите основной sql синтаксис при определении триггера. В каких случаях при работе с БД применяются триггеры?

Триггер (trigger) - это функция, которая автоматически вызывается в ответ на определенные события, такие как вставка, обновление или удаление данных в таблице.

Основной sql синтаксис при определении триггера:

CREATE TRIGGER ***триггер***BEFORE | AFTER } { ***событие***[ OR ***событие*** ] } ON ***таблица***  
FOR EACH { ROW |  STATEMENT }   
EXECUTE PROCEDURE ***функция***( ***аргументы***)

Случаи применения триггеров:

* обновление данных кэширующих полей и таблиц
* контроль и изменение таких значений полей, которые находятся в строгой зависимости от значений других полей
* контроль операций модификации данных для обеспечения выполнения правил бизнес-логики
* контроль мощности связей в случае, когда «стандартных» вариантов недостаточно, и необходимо обеспечить более сложное поведение
* контроль формата и значений данных в случае, если СУБД не поддерживает проверки
* прозрачное исправление ошибок в данных

1. Транзакции. Свойства транзакций (атомарность, консистентность, изолированность и устойчивость)

**Транзакция**–неделимая с точки зрения воздействия на БД последовательность операторов манипулирования данными (чтения, удаления, вставки, модификации),

Свойства:

* Атомарность (Atomicity): транзакция выполняется как одно целое и либо все операции в транзакции будут успешно завершены, либо ни одна операция не будет выполнена.
* Консистентность: транзакция должна обеспечивать целостность данных.
* Изолированность (Isolation): транзакция должна быть изолирована от других транзакций
* Устойчивость (Durability): после завершения транзакции ее изменения должны быть сохранены в базе данных и должны быть доступны даже в случае сбоя системы или отключения питания.

Транзакция — это набор операций по работе с базой данных (БД)

1. Атомарность: транзакция должна быть выполнена целиком или не выполнена совсем. Если одна из операций в транзакции не может быть выполнена, то все изменения, сделанные в рамках этой транзакции, должны быть отменены (откат).

2. Консистентность: транзакция должна приводить базу данных из одного согласованного состояния в другое. Если транзакция выполняется успешно, то база данных должна оставаться в согласованном состоянии.

3. Изолированность: транзакции должны быть изолированы друг от друга, чтобы изменения, сделанные одной транзакцией, не мешали другой транзакции. Это важно для того, чтобы избежать конфликтов при одновременном доступе к одним и тем же данным.

4. Устойчивость: изменения, сделанные в рамках транзакции, должны сохраняться в базе данных навсегда. Если транзакция выполнена успешно, то изменения, сделанные в рамках этой транзакции, должны быть сохранены в базе данных даже в случае сбоя системы.

Эти свойства обеспечивают надежность и целостность данных в базах данных и помогают избежать ошибок при одновременном доступе к данным. Важно, чтобы при проектировании баз данных учитывались эти свойства и чтобы система управления базами данных обеспечивала их реализацию.

1. Аномалии при выполнении транзакции (Потерянное обновление, «Грязное» чтение, Неповторяющееся чтение, Фантомное чтение)

* Потерянное обновление (Lost update): Множественное изменение одного блока приводит к сохранению только последнего изменения.
* «Грязное» чтение (Dirty read): Чтение данных, которые впоследствии отменяются ( откатываются).
* Неповторяющееся чтение (Non-repeatable read): Получение разных данных при повторном чтении в рамках одной транзакции
* Фантомное чтение (Phantom reads): Получение в рамках одной транзакции разного набора строк с использованием одного и того же условия выборки.

1. Уровни изоляции транзакции (Чтение неподтверждённых данных, Чтение подтверждённых данных, Повторяющееся чтение, Снимок, Сериализация)

Table

Description automatically generated

Уровни изоляции транзакции - это специальные режимы, которые определяют, как одна транзакция может видеть изменения, вносимые другими транзакциями в общую базу данных.

* Чтение неподтвержденных данных (Read Uncommitted) - это самый низкий уровень изоляции, при котором транзакция может видеть любые изменения в базе данных, даже если они были внесены другими неподтвержденными транзакциями. Этот уровень изоляции обеспечивает максимальную производительность и минимальную защищенность данных, поэтому его использование не рекомендуется.
* Чтение подтвержденных данных (Read Committed) - при этом уровне изоляции транзакция может видеть только те изменения, которые были подтверждены другими транзакциями. Таким образом, транзакция может читать данные только после того, как они были записаны в базу данных другой транзакцией. Этот уровень изоляции является стандартным и обеспечивает достаточный уровень защищенности данных.
* Повторяющееся чтение (Repeatable Read) - при этом уровне изоляции каждая транзакция видит данные так, как они были на момент ее начала. Если другая транзакция внесет изменения в эти данные, то они не будут видны текущей транзакции до ее завершения. Этот уровень изоляции обеспечивает более высокий уровень защищенности данных, но может приводить к блокировкам и конфликтам при параллельном доступе.
* Снимок (Snapshot) - это уровень изоляции, который позволяет выполнять чтение данных в момент времени, когда транзакция началась. Это обеспечивается созданием "снимка" данных, который сохраняется до завершения транзакции.
* Сериализация (Serializable) - это наивысший уровень изоляции, при котором транзакции выполняются последовательно в том порядке, в котором они были запущены. Этот уровень изоляции обеспечивает максимальный уровень защищенности данных, но может существенно снижать производительность в случае большого количества параллельных транзакций.

1. Журналы транзакций. Типы журнальных записей.

Журналы транзакций (transaction logs) в PostgreSQL используются для записи информации о всех изменениях, которые были произведены в базе данных. Они играют важную роль в обеспечении надежности, целостности и восстановления базы данных.

В PostgreSQL существует несколько типов журнальных записей, включая:

* WAL (Write-Ahead Logging) записи - записи, используемые для восстановления базы данных в случае сбоев. Они содержат информацию о транзакциях, включая команды изменения данных и общую информацию о состоянии транзакции.
* Журнальные записи запросов - записи, которые содержат информацию о каждом запросе к базе данных. Они используются для мониторинга и аудита базы данных.
* Журнальные записи ошибок - записи, которые содержат информацию об ошибках, произошедших в базе данных. Они используются для уведомления администраторов о проблемах, связанных с базой данных.
* Журнальные записи блокировок - записи, которые содержат информацию о блокировках, произошедших в базе данных. Они используются для отслеживания проблем, связанных с конкуренцией доступа к данным.
* Журнальные записи репликации - записи, которые содержат информацию о репликации базы данных между несколькими узлами. Они используются для отслеживания состояния репликации и для обеспечения целостности данных при передаче между узлами.

1. Стратегии ведения журнала транзакций

* Synchronous\_commit=on: журнальные записи сначала записываются на диск, а затем подтверждаются в транзакции.
* Synchronous\_commit=remote\_write: в этом режиме журнальные записи сначала записываются на диск локальной системы, а затем отправляются на другую систему для записи на диск.
* Synchronous\_commit=local: в этом режиме журнальные записи сначала записываются на диск локальной системы, а затем подтверждаются в транзакции.
* Synchronous\_commit=off: в этом режиме журнальные записи записываются на диск, но не подтверждаются в транзакции.

В PostgreSQL существует несколько стратегий ведения журнала транзакций:

1. Синхронный режим: в этом режиме сервер остановит выполнение транзакции, пока все данные не будут записаны на диск и подтверждены. Это гарантирует полную сохранность данных, но может привести к увеличению времени выполнения операций.

2. Асинхронный режим: в этом режиме данные сохраняются на диск по мере их поступления, без ожидания подтверждения. Это может ускорить выполнение операций, но есть риск потери данных в случае сбоя системы.

3. Режим архивации журнала транзакций: в этом режиме записи журнала транзакций сохраняются в отдельном месте, что позволяет восстановить данные в случае сбоя системы. Этот режим может использоваться как в синхронном, так и в асинхронном режимах.

4. Резервное копирование: это стратегия, в которой полная копия базы данных создается на регулярной основе и сохраняется на отдельном устройстве. Это обеспечивает дополнительную защиту данных в случае сбоя системы.

1. Аудит событий СУБД. Какие события важны с точки зрения ИБ.

Аудит событий в СУБД - это процесс регистрации событий, производимых базой данных и ее пользователями. Аудит позволяет отслеживать действия пользователей, контролировать безопасность СУБД и реагировать на нарушения безопасности.

Некоторые наиболее важные события, которые имеет смысл отслеживать при аудите СУБД с точки зрения ИБ:

1. Попытки неудачной аутентификации в базе данных.

2. Попытки обойти механизмы безопасности, такие как изменение прав на объекты базы данных, подключение к базе данных с различных адресов и т.д.

3. Попытки внесения изменений данных в базу данных или чтение конфиденциальных данных.

4. Сбои и ошибки в работе СУБД.

5. Выполнение необычных запросов и процедур в СУБД, которые могут указывать на попытки выполнения очень сложных операций.

Это только некоторые из важных событий, которые могут быть отслежены при аудите СУБД. В зависимости от конкретной базы данных, ее пользователей и целей, могут быть выделены и другие события для отслеживания.