Defintions for Lab 4

Berezin Leonid Yurievich

November 2024

Триггеры

Триггер является указанием, что база данных должна автоматически выполнить заданную функцию всякий раз, когда выполнен определённый тип операции.

Триггеры могут быть настроены на выполнение до или после операции (например, вставка, обновление или удаление), и на различные виды событий (например, изменения в строках таблицы).

Триггеры позволяют:

- Автоматизировать задачи в базе данных (например, проверка данных перед вставкой).
- Обеспечить целостность данных, например, через выполнение проверок при изменении данных.
- Реализовать логику, которая должна быть выполнена каждый раз при изменении данных (например, изменение других таблиц при вставке или обновлении данных).

Триггер состоит из следующих частей:

- **Событие**: Указывает, на какие изменения данных триггер будет реагировать, например, на операцию ‘INSERT‘, ‘UPDATE‘, или ‘DELETE‘.
- **Время выполнения**: Указывает, когда триггер должен быть выполнен: до (BEFORE) или после (AFTER) выполнения операции.
- **Действие**: Функция или выражение, которое выполняется при активации триггера.

Типы триггеров

- **BEFORE**: Триггер выполняется до того, как будет выполнена операция (например, перед вставкой или обновлением строки). Это позволяет, например, выполнить проверки или модификации данных до их сохранения в таблице.
- **AFTER**: Триггер выполняется после выполнения операции. Это полезно для задач, которые должны быть выполнены после изменения данных, например, обновление других таблиц.
- **INSTEAD OF**: Этот триггер заменяет выполнение операции. Например, если вместо стандартной вставки данных требуется выполнить кастомное действие, этот триггер может переопределить стандартное поведение.

Основные особенности триггеров

- **Триггеры могут быть настроены на работу с несколькими строками**: В отличие от обычных операций, триггер может быть настроен для обработки каждой строки, которая изменяется в ходе операции (например, в случае вставки нескольких строк).
- **Необходимо соблюдать производительность**: Триггеры могут повлиять на производительность, так как они добавляют дополнительные вычисления в каждую операцию, с которой они связаны.
- **Может вызывать рекурсию**: Важно помнить, что триггеры могут вызывать другие триггеры. Это может привести к рекурсии, которая иногда может быть нежелательной или вызвать ошибки. Например, триггер, вызывающий изменения в другой таблице, может вызвать другой триггер, который снова изменяет первую таблицу.
- **Ошибки в триггерах могут привести к отмене операции**: Если в триггере возникнет ошибка (например, попытка вставки дублирующего значения), операция, которая вызвала триггер, будет отменена.

Триггеры и транзакции

Триггеры выполняются в рамках транзакции, и их результат влияет на результат самой транзакции. Если триггер вызывает ошибку, вся транзакция будет откатана. Таким образом, триггеры обеспечивают согласованность базы данных в условиях параллельных операций.

Уровни изоляции транзакций и триггеры

При работе с триггерами важно учитывать уровни изоляции транзакций. Уровень изоляции определяет, какие изменения данных видны для других транзакций и какие проблемы могут возникать при параллельном выполнении транзакций. Например:

- При уровне изоляции **Read Uncommitted**, триггеры могут видеть незавершённые изменения других транзакций, что может привести к ошибкам, если данные ещё не были зафиксированы.
- При уровне **Serializable**, транзакции и триггеры будут работать в полностью изолированном режиме, где транзакции выполняются как если бы они были последовательными, исключая все возможные аномалии.

Транзакции

Транзакция — это последовательность операций, выполняемых как единое целое. Она представляет собой набор операций, которые должны быть выполнены полностью или не выполнены вовсе, обладая следующими свойствами:

1. **Атомарность** — либо будут выполнены все действия транзакции, либо никакие.
2. **Согласованность** — транзакция должна переводить базу данных из одного согласованного состояния в другое. Это означает, что все правила и ограничения базы данных должны соблюдаться до и после выполнения транзакции.
3. **Изолированность** — транзакция выполняется так, как будто не знает о существовании других транзакций.
   (a) *Эффект потерянного обновления/изменения (аномалия сериализации)* возникает, когда несколько транзакций обновляют одни и те же данные, не учитывая изменений, сделанных другими транзакциями.
   (b) *Эффект грязного чтения* — транзакция читает данные, записанные параллельной незавершённой транзакцией.
   (c) *Эффект потерянного изменения* — это ситуация, когда при повторном чтении объекта базы данных транзакция не видит своих изменений, произведенных ранее, вследствие перезаписи их другой транзакцией.
   (d) *Эффект неповторяемого чтения* — транзакция повторно читает те же данные, что и раньше, и обнаруживает, что они были изменены другой транзакцией (которая завершилась после первого чтения).
   (e) *Эффект фантомного чтения* — транзакция повторно выполняет запрос, возвращающий набор строк для некоторого условия, и обнаруживает, что набор строк, удовлетворяющих условию, изменился из-за транзакции, завершившейся за это время.
4. **Долговечность** — после завершения транзакции все изменения, сделанные в рамках этой транзакции, должны быть сохранены и оставаться в базе данных даже в случае сбоев системы.
   - *Журналирование*: после выполнения изменения записываются в журнал и потом вступают в силу.

Уровни изоляции

Уровень изоляции ReadUncommitted

Транзакция может видеть результаты других транзакций, даже если они ещё не закоммичены.

Уровень изоляции Read Committed

Этот уровень изоляции гарантирует, что транзакция может читать только данные, которые были зафиксированы (committed) другими транзакциями. Оператор SELECT делает снимок базы данных до начала выполнения запроса. Два SELECT могут выдать разные результаты, если другие транзакции зафиксируют изменения после запуска первого, но до запуска второго.

- Исключает: грязное чтение.
- Не исключает: неповторяемое чтение, фантомное чтение, аномалию сериализации.

Уровень изоляции Repeatable Read

В режиме Repeatable Read видны только те данные, которые были зафиксированы до начала транзакции, но не видны незафиксированные данные и изменения, произведённые другими транзакциями в процессе выполнения данной транзакции. Однако запрос будет видеть эффекты предыдущих изменений в своей транзакции, несмотря на то, что они не зафиксированы.

- Исключает: грязное чтение, неповторяемое чтение, фантомное чтение.
- Не исключает: аномалию сериализации.

Уровень изоляции Serializable

На этом уровне моделируется последовательное выполнение всех зафиксированных транзакций, как если бы транзакции выполнялись одна за другой, последовательно, а не параллельно. Однако, как и на уровне Repeatable Read, на этом уровне приложения должны быть готовы повторять транзакции из-за сбоев сериализации. Фактически этот режим изоляции работает так же, как и Repeatable Read, только он дополнительно отслеживает условия, при которых результат параллельно выполняемых сериализуемых транзакций может не согласовываться с результатом этих же транзакций, выполняемых по очереди. Это отслеживание не привносит дополнительных препятствий для выполнения, кроме тех, что присущи режиму Repeatable Read, но тем не менее создаёт некоторую добавочную нагрузку, а при выявлении исключительных условий регистрируется аномалия сериализации и происходит сбой сериализации.