

Триггеры

Триггер — это специальный тип процедуры, который автоматически запускается («срабатывает») в ответ на определенное событие в базе данных, такое как вставка, обновление или удаление записи в таблице.

Принцип работы:

1. Событие: Триггер ассоциируется с таблицей и срабатывает при определенном событии, таком как `INSERT` , `UPDATE` , `DELETE` , или даже при комбинации этих событий.
 2. Триггерная функция: Когда срабатывает триггер, он вызывает специальную функцию, называемую триггерной функцией. Эта функция содержит логику, которая должна быть выполнена при срабатывании триггера.
 3. Контекст выполнения: Триггерная функция может обрабатывать информацию о текущей операции, такую как значения столбцов в строке, которая была вставлена, обновлена или удалена.
- Уровень строки и уровень операции: Триггеры могут быть настроены на срабатывание для каждой строки, затронутой операцией, или один раз для всей операции.
 - Передача параметров: Триггерные функции могут получать дополнительные параметры, определяемые при создании триггера.
 - Порядок выполнения: Если на одну и ту же таблицу настроено несколько триггеров, можно управлять порядком их выполнения.
 - Рекурсивные триггеры: Можно настроить триггеры так, чтобы они вызывали другие триггеры (рекурсивный вызов).

Типы триггеров:

- BEFORE: Срабатывают перед выполнением операции (например, перед вставкой или обновлением).
- AFTER: Срабатывают после выполнения операции.
- INSTEAD OF: Используются в представлениях для переопределения стандартного поведения операций `INSERT` , `UPDATE` , или `DELETE` .

Типичные кейсы использования триггеров:

1. Аудит и логирование изменений

Требуется отслеживать все изменения, происходящие в определенной таблице, чтобы иметь полный аудит этих действий.

- Без триггера: После каждой операции `INSERT` , `UPDATE` или `DELETE` в приложении необходимо вручную записывать соответствующие действия в таблицу аудита. Это может привести к упущениям или ошибкам, если разработчик забудет добавить код логирования.
- С триггером:

```
CREATE TRIGGER audit_trigger AFTER INSERT OR UPDATE OR DELETE ON target_table FOR EACH ROW EXECUTE FUNCTION record_audit();
```

Триггер `audit_trigger` автоматически срабатывает после любой из указанных операций и вызывает функцию `record_audit()` , которая занимается записью соответствующей информации в таблицу аудита.

2. Поддержание целостности данных

Автоматическое обновление или проверка связанных данных при изменении основной таблицы.

- Без триггера: Необходимо ручное обновление связанных данных в разных таблицах каждый раз, когда основные данные изменяются. Это увеличивает риск ошибок и нарушения целостности данных.
- С триггером:

```
CREATE TRIGGER update_related_trigger AFTER UPDATE ON main_table FOR EACH ROW EXECUTE FUNCTION update_related_data();
```

Каждый раз, когда данные в `main_table` обновляются, триггер `update_related_trigger` автоматически вызывает `update_related_data()` , которая обеспечивает синхронизацию или проверку связанных данных.

3. Автоматическое заполнение полей

Необходимость автоматического заполнения или обновления определенных полей, например, временных меток или счетчиков.

- Без триггера: Разработчику приходится вручную устанавливать эти значения в коде приложения, что может привести к ошибкам или неконсистентности данных.

- С триггером:

```
CREATE TRIGGER autofill_trigger BEFORE INSERT OR UPDATE ON some_table FOR EACH ROW EXECUTE FUNCTION autofill_fields();
```

Перед вставкой или обновлением каждой строки в `some_table`, триггер `autofill_trigger` вызывает функцию `autofill_fields()`, которая автоматически заполняет или обновляет необходимые поля.

4. Каскадное удаление или обновление

Автоматическое удаление или обновление связанных записей при изменении или удалении основной записи.

- Без триггера: При удалении записи разработчикам необходимо вручную удалять все связанные записи, что увеличивает сложность кода и риск ошибок.

- С триггером:

```
CREATE TRIGGER cascade_delete_trigger AFTER DELETE ON parent_table FOR EACH ROW EXECUTE FUNCTION cascade_delete();
```

При удалении записи из `parent_table`, триггер `cascade_delete_trigger` автоматически вызывает `cascade_delete()`, которая удаляет все связанные записи.

5. Автоматическое вычисление агрегированных данных

Необходимость поддержания актуальных агрегированных данных, например, сумм или средних значений, в сводных таблицах.

- Без триггера: Расчет и обновление агрегированных данных требует отдельных запросов после каждой соответствующей операции.

- С триггером:

```
CREATE TRIGGER aggregate_data_trigger AFTER INSERT OR UPDATE OR DELETE ON transaction_table FOR EACH ROW EXECUTE FUNCTION update_aggregates();
```

При любых изменениях в `transaction_table` , триггер `aggregate_data_trigger` вызывает `update_aggregates()` , которая обновляет агрегированные данные в сводных таблицах.

6. Валидация данных

Проверка правильности и соответствия данных определенным критериям перед их вставкой или обновлением в базе данных.

- Без триггера: Валидация данных осуществляется на стороне приложения, что может привести к пропуску некоторых проверок.
- С триггером:

```
CREATE TRIGGER validate_data_trigger BEFORE INSERT OR UPDATE ON user_table FOR EACH ROW EXECUTE FUNCTION validate_data();
```

Перед вставкой или обновлением каждой строки в `user_table` , триггер `validate_data_trigger` вызывает `validate_data()` , которая осуществляет необходимые проверки данных.

Среди недостатков триггеров можно отметить все недостатки обычных хранимых процедур (ведь триггеры это и есть процедуры). Также можно выделить несколько отдельных недостатков:

- Триггеры, выполняющие тяжелые или сложные операции на каждую вставленную, обновленную или удаленную строку, могут значительно снизить производительность системы, особенно в высоконагруженных или больших базах данных.
- Триггеры, особенно те, которые изменяют другие таблицы, могут вызывать каскадные изменения, что может привести к непреднамеренным и сложным для отслеживания побочным эффектам. Внесение изменений в одну таблицу может неожиданно повлиять на данные в других частях базы данных.
- Триггеры могут усложнить управление транзакциями, особенно в сценариях с множественными вложенными операциями. Триггеры могут вызывать дополнительные транзакции или блокировки, что увеличивает сложность и вероятность взаимных блокировок.