Триггеры

Реляционная модель

Преподаватель:

канд. тех. наук, доц. Озерова Г.П.

Триггеры SQL

Триггер позволяет описать действия, которые должны быть выполнены при наступлении определенных действий с данными в таблице.

Особенность триггеров заключается в том, что SQL код, написанный в теле триггера, будет исполнен после того, как в базе данных произойдет какое-либо событие.

События в базах данных происходят в результате выполнения запросов корректировки данных: UPDATE, INSERT, DELETE.

Триггеры SQL

Основное назначение триггеров заключается в обеспечение целостности данных в базе данных, а также для реализации довольно сложной бизнес-логики.

Например, при добавлении новой записи в одну таблицу, можно автоматически изменить данные в других таблицах.

Триггеры SQL

Самые распространенные функции триггеров:

1. **Функция журнализации**. Часто при помощи триггеров разработчики создают таблицы-журналы, в которых фиксируются различные изменения в базе данных, а также кто их вносил.
2. **Функция согласования данных**. При изменении некоторой таблицы, триггер может проверять связанные таблицы на согласованность данных, при необходимости вносить какие-то изменения в эти таблицы.
3. **Функция очистки данных**. Если связанные данные хранятся в несвязанных внешним ключом таблицах, триггеры могут выполнить действия по очистке таких таблиц.

Триггеры SQL

Триггер – объект базы данных, поэтому имя триггера должно быть уникальным во всей базе данных.

У триггеров в SQL есть **момент запуска**. Момент запуска триггера можно разделить на два вида: **BEFORE** и **AFTER**.

Момент запуска триггера **AFTER** говорит о том, что триггер будет запущен **после** выполнения какого-либо события в базе данных.

Соответственно, момент запуска триггера **BEFORE** говорит о том, что триггер будет запущен **до** выполнения события в базе данных.

Триггеры SQL

Триггеры можно разделить на два вида по их применению.

- Триггер **BEFORE**, который срабатывает ДО выполнения какого-либо события в базе данных, например, вставляется строка в таблицу – триггер запускается ДО момента ее добавления.
- Триггер **AFTER**, который срабатывает ПОСЛЕ выполнения события в базе данных, например, вставляется строка в таблицу – триггер запускается ПОСЛЕ того, как она будет добавлена в таблицу.

Триггеры SQL

Триггеры можно разделить по типам команд SQL:

- **DELETE** триггер - запускается при попытке удаления строк из таблицы базы данных;
- **UPDATE** триггер - запускается при попытке обновления данных в таблице базы данных;
- **INSERT** триггер- запускается при попытке добавить строку в таблицу базы данных.

Триггеры SQL

Таким образом можно выделить 6 различных типов триггеров:

- триггер **BEFORE DELETE** - запускается ДО попытки УДАЛЕНИЯ строки из таблицы базы данных;
- триггер **AFTER DELETE** - запускается ПОСЛЕ УДАЛЕНИЯ строки из таблицы базы данных;
- триггер **BEFORE UPDATE** - запускается ДО обновления данных в таблице базы данных;
- триггер **AFTER UPDATE** - запускается ПОСЛЕ обновления данных в таблице базы данных;
- Триггер **BEFORE INSERT** - запускается ДО попытки вставки строки в таблицу базы данных;
- Триггер **AFTER INSERT** - запускается ПОСЛЕ попытки вставки строки в таблицу базы данных (ТАКОГО ТРИГГЕРА СОЗДАТЬ НЕЛЬЗЯ).

Триггеры SQL

С триггерами допустимы **следующие действия**:

- создание;
- удаление, используется запрос:

```
DROP TRIGGER имя_триггера;
```

Триггеры SQL

Синтаксис запроса на создание триггера:

```
CREATE TRIGGER имя_триггера
[AFTER | BEFORE] [INSERT | UPDATE | DELETE]
ON имя_таблицы
FOR EACH ROW
BEGIN
   запрос_или_выражение_sql_1;
   ...
   запрос_или_выражение_sql_N;
END;
```

Триггеры SQL

Механизм выполнения триггера:

1. Триггер связывается с некоторой таблицей.
2. До любых изменений данных в таблице проверяется тип триггера: является ли он **BEFORE** или **AFTER**.
3. Определяется тип изменения данных:
   - если добавляются строки, проверяется тип команды, если команда не **INSERT**, выполнение триггера завершается;
   - если обновляются данные и тип команды не **UPDATE**, выполнение триггера завершается;
   - если удаляются строки и тип команды не **DELETE**, выполнение триггера завершается;
4. Триггер выполняет действия, описанные в его теле.

Триггеры SQL

Для доступа к полям старой записи таблицы (которая была ДО ПРИМЕНЕНИЯ триггера) используется алиас **OLD**, для обращения к полям новой - алиас **NEW**.

Например, **OLD.столбец** - значение **столбца** до выполнения действий, описанных в триггере, **NEW.столбец** - значение того же **столбца** после выполнения действий, описанных в триггере.

Значение столбцов **NEW** доступны для изменения в триггерах, запускаемых по событию **BEFORE**.

Триггеры SQL

Перед созданием триггера необходимо ответить на следующие вопросы:

1. С какой таблицей он будет связан?
2. В какой момент времени он должен быть "запущен"? С какой операцией ассоциируется?
3. Что именно он должен делать?

Выполнение триггера BEFORE INSERT

Пример. Создать триггер, который автоматически увеличивает цену новой книги, добавляемой в таблицу **book**, на 20%.

Описание триггера:

1. Триггер связан с таблицей **book**.
2. Он должен быть запущен перед (**BEFORE**) тем, как добавить новую запись в таблицу (**INSERT**).
3. Триггер увеличивает цену товара (**NEW.price**) на 20%.

Выполнение триггера BEFORE INSERT

Пример. Создать триггер, который автоматически увеличивает цену новой книги, добавляемой в таблицу **book**, на 20%.

Код:

```
CREATE TRIGGER insert_book
BEFORE INSERT ON book
FOR EACH ROW
BEGIN
    SET NEW.price = NEW.price * 1.2;
END;
```

Запрос:

```
INSERT INTO book(title, author, price, amount)
VALUES ("Стихи", "Цветаева М.И.", 300, 5);
```

Выполнение триггера BEFORE INSERT

*Таблица **book**:*

book_id	title	author	price	amount
1	Мастер и Маргарита	Булгаков М.А.	670.99	3
2	Белая гвардия	Булгаков М.А.	540.50	5
…				
28	Скрюченный домишко	Агата Кристи	150.01	13

```
INSERT INTO book(title, author, price, amount)
VALUES ("Стихи", "Цветаева М.И.", 300, 5);
```

➔ выполняется триггер

При этом:

- «старых» значений в таблице нет (строка только вставляется);
- «новые» значения:

  NEW.title = "Стихи“

  NEW.author = "Цветаева М.И.“

  NEW.price = 300

  NEW.amount = 5

Выполнение триггера BEFORE INSERT

Действия триггера:

```
SET NEW.price = NEW.price * 1.2;
```

было: NEW.title = "Стихи“
NEW.author = "Цветаева М.И.“
NEW.price = 300
NEW.amount = 5

стало: NEW.title = "Стихи“
NEW.author = "Цветаева М.И.“
NEW.price = 360
NEW.amount = 5

*Измененная таблица **book**:*

book_id	title	author	price	amount
1	Мастер и Маргарита	Булгаков М.А.	670.99	3
2	Белая гвардия	Булгаков М.А.	540.50	5
…				
28	Скрюченный домишко	Агата Кристи	150.01	13
29	Стихи	Цветаева М.И.	360.00	5

Выполнение триггера BEFORE INSERT

Особенности использования триггера:

1. Доступны только значения полей **NEW**.
2. Можно изменять данные в текущей таблице (поля с префиксом NEW) с помощью операторов присваивания или запросов.
3. Можно запускать запросы на изменение других таблиц.

Выполнение триггера BEFORE UPDATE

Пример. Создать триггер, который при изменении количество экземпляров проверяет количество, если количество меньше 5 - уменьшает цену книги на 20%.

Описание триггера:

1. Триггер связан с таблицей **book**.
2. Он должен быть запущен перед (**BEFORE**) тем, как обновить данные в таблице (**UPDATE**).
3. Триггер должен проверить количество экземпляров и, если значение меньше 5, то уменьшить цену на 10%.

Выполнение триггера BEFORE UPDATE

Пример. Создать триггер, который при изменении количество экземпляров проверяет количество, если количество меньше 4 - уменьшает цену книги на 20%.

Код:

```
CREATE TRIGGER update_book_amount
BEFORE UPDATE ON book
FOR EACH ROW
BEGIN
    IF NEW.amount < 5  THEN
       SET NEW.price = OLD.price * 0.8;
    END IF;
END;
```

Запрос:

```
UPDATE book
SET amount = 2
WHERE book_id = 2;
```

Выполнение триггера BEFORE UPDATE

*Таблица **book**:*

book_id	title	author	price	amount
1	Мастер и Маргарита	Булгаков М.А.	670.99	3
2	**Белая гвардия**	**Булгаков М.А.**	**540.50**	**5**
…				
28	Скрюченный домишко	Агата Кристи	150.01	13

```
UPDATE book
SET amount = 2
WHERE book_id = 2;
```

➔ выполняется триггер

Выполнение триггера BEFORE UPDATE

Запрос:

```
UPDATE book
SET amount = 2
WHERE book_id = 2;
```

Тело триггера:

```
IF NEW.amount < 5  THEN
      SET NEW.price = OLD.price * 0.8;
 END IF;
```

При этом:

- «старые» значений обновляемой строки;
  OLD.title = “Белая гвардия“
  OLD.author = "Булгаков М.А.“
  OLD.price = 540.50
  OLD.amount = 5 – то что было в таблице перед выполнением запроса
- «новые» значения:
  NEW.title = “Белая гвардия“
  NEW.author = "Булгаков М.А.“
  NEW.price = 432.40 – вычислялась новая цена, так как NEW.amount < 5
  NEW.amount = 2 – то что стало в таблице после выполнения запроса

Выполнение триггера BEFORE UPDATE

*Таблица **book**:*

book_id	title	author	price	amount
1	Мастер и Маргарита	Булгаков М.А.	670.99	3
2	**Белая гвардия**	**Булгаков М.А.**	**540.50**	**5**
…				
28	Скрюченный домишко	Агата Кристи	150.01	13

*Измененная таблица **book**:*

book_id	title	author	price	amount
1	Мастер и Маргарита	Булгаков М.А.	670.99	3
2	**Белая гвардия**	**Булгаков М.А.**	**432.40**	**2**
…				
28	Скрюченный домишко	Агата Кристи	150.01	13

Выполнение триггера BEFORE UPDATE

Особенности использования триггера:

1. Доступны все значения полей **OLD** и **NEW**.
2. Можно изменять данные в текущей таблице (поля с префиксом **NEW**) с помощью операторов присваивания или запросов.
3. Можно запускать запросы на изменение других таблиц.

Выполнение триггера AFTER UPDATE

Пример. На складе есть отдельная таблица **book_amount**, в которой хранится информация о всех книгах и количество проданных экземпляров. Если уменьшается количество экземпляров книги, то на это же значение увеличивается количество экземпляров той же книги в таблице **book_amount**. Создать триггер, который автоматически обновляет количество книг в таблице **book_amount**.

Описание триггера:

1. Триггер связан с таблицей **book**.
2. Он должен быть запущен либо после (**AFTER**) обновления записи в таблице (**UPDATE**). Можно использовать любое событие, так как данные изменяются в ДРУГОЙ таблице.
3. Действия триггера: увеличить количество книги в таблице **book_amount**.

Выполнение триггера AFTER UPDATE

Пример. На складе есть отдельная таблица **book_amount**, в которой хранится информация о всех книгах и количество проданных экземпляров. Если уменьшается количество экземпляров книги, то на это же значение увеличивается количество экземпляров той же книги в таблице **book_amount**. Создать триггер, который автоматически обновляет количество книг в таблице **book_amount**.

Код:

```
CREATE TRIGGER update_book_amount
AFTER UPDATE ON book
FOR EACH ROW
BEGIN
    IF NEW.amount < OLD.amount THEN
        UPDATE book_amount
        SET amount_sum = amount_sum + (OLD.amount - NEW.amount)
        WHERE title = OLD.title  AND author = OLD. author;
   END IF;
END;
```

Выполнение триггера AFTER UPDATE

*Таблица **book_amount**:*

book_id	title	author	amount_sum
1	Мастер и Маргарита	Булгаков М.А.	10
2	**Белая гвардия**	**Булгаков М.А.**	**6**
…			
28	Скрюченный домишко	Агата Кристи	1

*Таблица **book**:*

book_id	title	author	price	amount
1	Мастер и Маргарита	Булгаков М.А.	670.99	3
2	**Белая гвардия**	**Булгаков М.А.**	**540.50**	**5**
…				
28	Скрюченный домишко	Агата Кристи	150.01	13

```
UPDATE book
SET amount = amount - 2
WHERE book_id = 2;
```

➔ выполняется триггер

Выполнение триггера AFTER UPDATE

Запрос:

```
UPDATE book
SET amount = amount - 2
WHERE book_id = 2;
```

Тело триггера:

```
IF NEW.amount < OLD.amount THEN
   UPDATE book_amount
   SET amount_sum = amount_sum + (OLD.amount - NEW.amoun
    WHERE title = OLD.title  AND author = OLD. author;
END IF;
```

При этом:

- «старые» значений обновляемой строки;
  OLD.title = “Белая гвардия“
  OLD.author = "Булгаков М.А.“
  OLD.price = 540.50
  OLD.amount = 5 – то что было в таблице перед выполнением запроса
- «новые» значения:
  NEW.title = “Белая гвардия“
  NEW.author = "Булгаков М.А.“
  NEW.price = 540.50
  NEW.amount = 3 – то что стало в таблице после выполнения запроса

Выполнение триггера AFTER UPDATE

*Изменённая таблица **book_amount**:*

book_id	title	author	amount_sum
1	Мастер и Маргарита	Булгаков М.А.	10
2	**Белая гвардия**	**Булгаков М.А.**	**8**
…			
28	Скрюченный домишко	Агата Кристи	1

= 6 + (5 - 3)

*Измененная таблица **book**:*

book_id	title	author	price	amount
1	Мастер и Маргарита	Булгаков М.А.	670.99	3
2	**Белая гвардия**	**Булгаков М.А.**	**432.40**	**3**
…				
28	Скрюченный домишко	Агата Кристи	150.01	13

= 5 - 2

Выполнение триггера AFTER UPDATE

Особенности использования триггера:

1. Доступны все значения полей **OLD** и **NEW**.
2. НЕЛЬЗЯ изменять данные в текущей таблице с помощью операторов присваивания или запросов (поля **NEW** не доступны для изменения).
3. Можно запускать запросы на изменение других таблиц.

Выполнение триггера AFTER DELETE

Пример. Создать триггер, который при удалении книги заносит информацию о ней в таблицу **archive_book**.

Описание триггера

1. Триггер связан с таблицей **book**.
2. Он должен быть запущен после (**AFTER**) удаления записи из таблицы (**DELETE**).
3. Действия триггера: занести информацию об удаляемой книге в архивную таблицу.

Выполнение триггера AFTER DELETE

Пример. Создать триггер, который при удалении книги заносит информацию о ней в таблицу **archive_book**.

Код:

```
CREATE TRIGGER delete_book
 AFTER DELETE ON book
FOR EACH ROW
BEGIN
    INSERT INTO archive_book(title, author_name, price)
    VALUES (OLD.title, OLD.author, OLD.price);
END;
```

Запрос:

```
DELETE FROM book
WHERE book_id = 2;
```

Выполнение триггера AFTER DELETE

*Таблица **archive_book**:*

archive_id	title	author	price
1	Стихи и поэмы	Пушкин А.С.	100.50
2	Черный человек	Есенин С.А.	280.00

*Таблица **book**:*

book_id	title	author	price	amount
1	Мастер и Маргарита	Булгаков М.А.	670.99	3
2	**Белая гвардия**	**Булгаков М.А.**	**540.50**	**5**
…				
28	Скрюченный домишко	Агата Кристи	150.01	13

```
DELETE FROM book
WHERE book_id = 2;
```

➔ выполняется триггер

Выполнение триггера AFTER UPDATE

Запрос:

```
DELETE FROM book
WHERE book_id = 2;
```

Тело триггера:

```
INSERT INTO archive_book(title, author_name, price)
VALUES (OLD.title, OLD.author, OLD.price);
```

При этом:

- «старые» значений обновляемой строки;
  OLD.title = “Белая гвардия“
  OLD.author = "Булгаков М.А.“
  OLD.price = 540.50
  OLD.amount = 5 – то что было в таблице перед выполнением запроса
- «новые» не доступны значения:

Выполнение триггера AFTER DELETE

*Измененная таблица **archive_book**:*

archive_id	title	author	price
1	Стихи и поэмы	Пушкин А.С.	100.50
2	Черный человек	Есенин С.А.	280.00
3	**Белая гвардия**	**Булгаков М.А.**	**540.50**

*Измененная таблица **book**:*

book_id	title	author	price	amount
1	Мастер и Маргарита	Булгаков М.А.	670.99	3
…				
28	Скрюченный домишко	Агата Кристи	150.01	13

Выполнение триггера AFTER DELETE

Особенности использования триггера:

1. Доступны только значения полей **OLD**.
2. НЕЛЬЗЯ изменять данные в текущей записи с помощью операторов присваивания или запросов.
3. Можно запускать запросы на изменение других таблиц.
4. НЕЛЬЗЯ использовать запрос на выборку для удаляемых строк из таблицы **book**, так как в момент запуска триггера записи из таблицы уже УДАЛЕНЫ:

```
INSERT INTO archive_book(title, author_name, price)
SELECT title, author, price
FROM book
WHERE book_id = OLD.book_id;
```

Выполнение триггера BEFORE DELETE

Пример. Создать триггер, который при удалении книги заносит информацию о ней в таблицу **archive_book**.

Код:

```
CREATE TRIGGER delete_book
BEFORE DELETE ON book
FOR EACH ROW
BEGIN
    INSERT INTO archive_book(title, author_name, price)
   SELECT title, author, price
   FROM book
   WHERE book_id = OLD.book_id;
```

или

```
   INSERT INTO archive_book(title, author_name, price)
    VALUES (OLD.title, OLD.author, OLD.price);
END;
```

Выполнение триггера AFTER DELETE

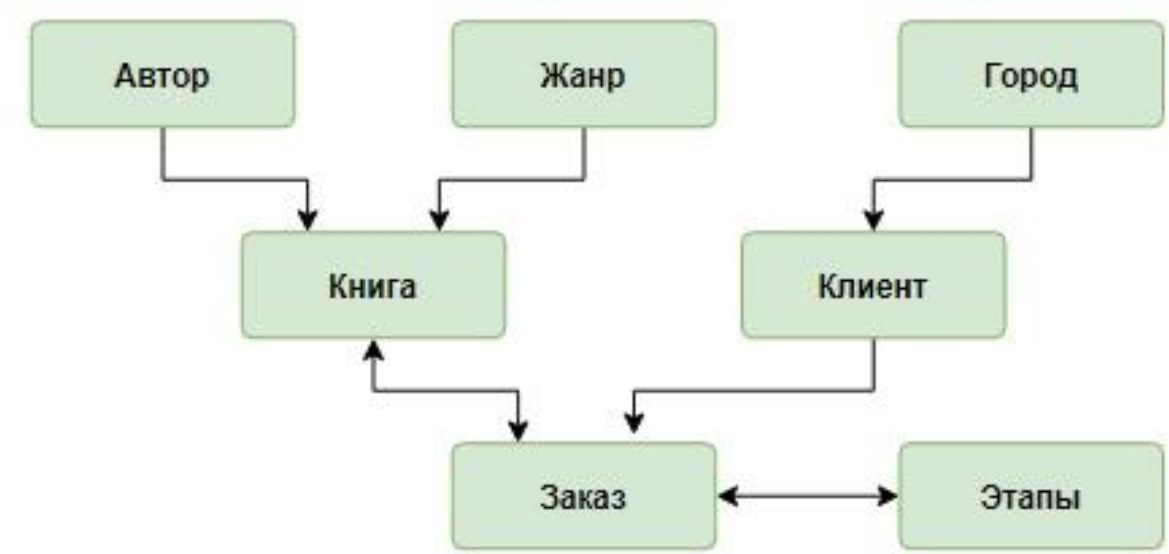
Особенности использования триггера:

1. Доступны только значения полей **OLD**.
2. НЕЛЬЗЯ изменять данные в текущей записи с помощью операторов присваивания или запросов.
3. Можно запускать запросы на изменение других таблиц.
4. МОЖНО использовать запрос на выборку удаляемых строк из таблицы **book**, так как в момент запуска триггера записи из таблицы еще НЕ УДАЛЕНЫ:

Триггеры для реализации бизнес-логики

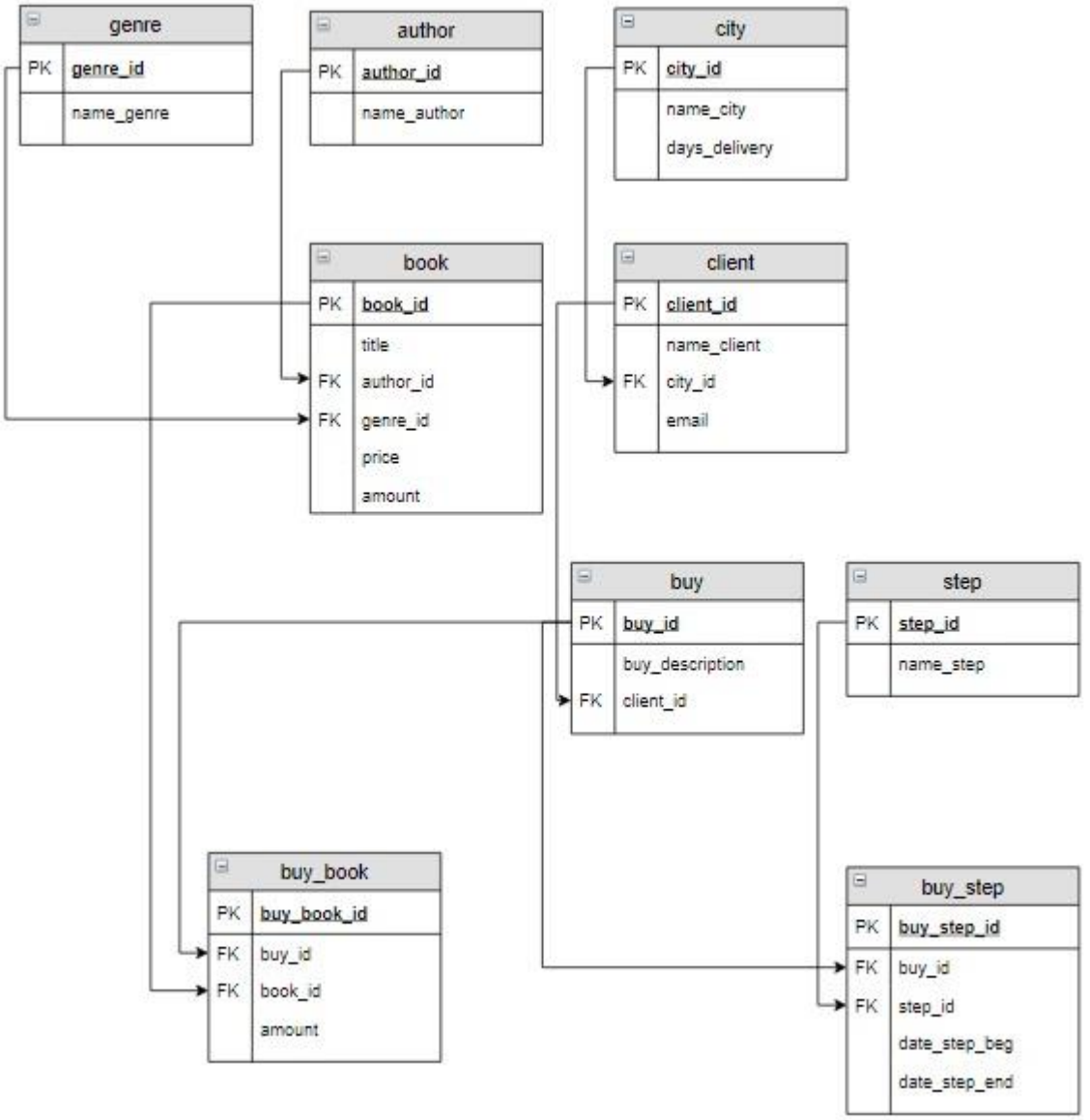
Предметная область: интернет-магазин

Концептуальная схема:



Триггеры для реализации бизнес-логики

Логическая схема:



Типовые операции с данными

Операция 1. В «Интернет-магазин» пришел новый клиент. В базу данных его необходимо добавить запросом на добавление **INSERT.**

Операция 2. Клиент магазина решил сделать новый заказ. Эта операция в базе данных состоит в добавлении новой записи запросом **INSERT** в таблицу **buy**, в которой указывается клиент и его пожелания.
Включение новой записи должно автоматически инициировать следующие действия:

- сохранение **id** созданного заказа;
- добавление этапов (Оплата, Упаковка и пр.) прохождения заказа в таблицу buy_step для текущего заказа;
- вставка даты начала этапа Оплаты для текущего заказа.

Операция 2 - Новый заказ

Операция 2 – Новый заказ


```
INSERT INTO buy(client_id) VALUES(2);
```

Описание триггера

1. Триггер связан с таблицей **buy**.
2. Он должен быть запущен после (**AFTER**) добавления (**INSERT**).
3. Действия триггера:

```
SET @buy_current = NEW.buy_id;
```

```
INSERT INTO buy_step (buy_id, step_id)
SELECT NEW.buy_id, step_id FROM step;
```

```
UPDATE buy_step
SET date_step_beg = NOW()
WHERE step_id = 1 and buy_id = NEW.buy_id;
```

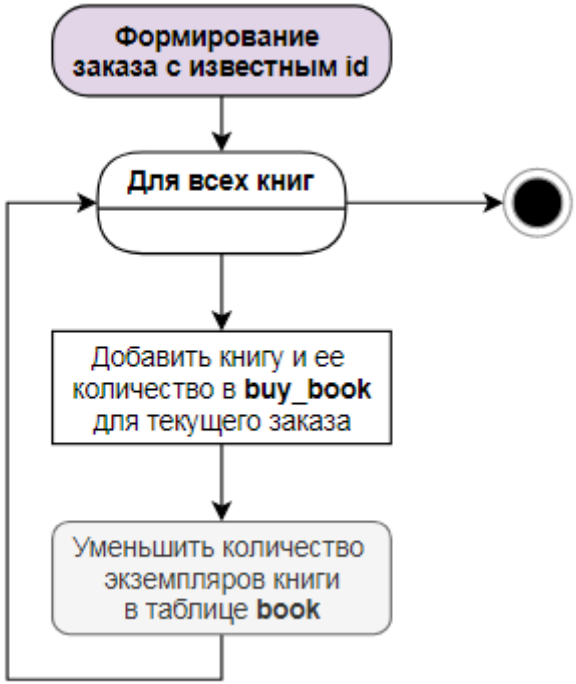
Типовые операции с данными

Операция 3. После создания заказа клиент выбирает необходимые ему книги, указывает количество и включает их в заказ.

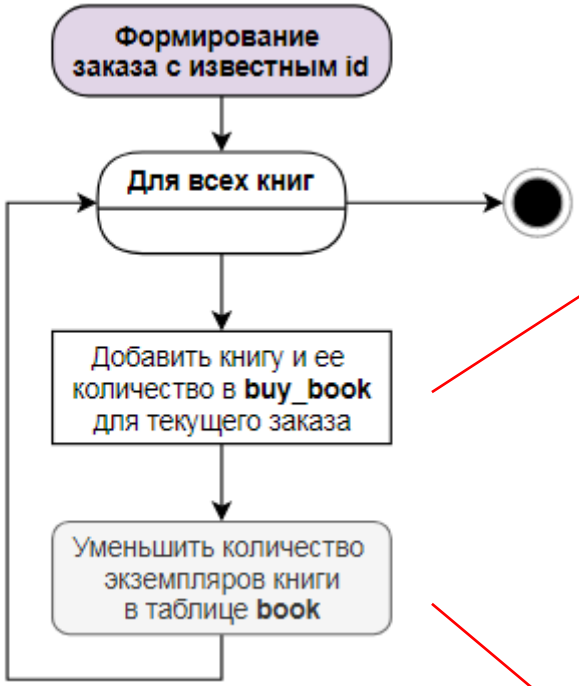
В базе данных для созданного заказа в таблицу **buy_book** добавляются выбранные клиентом книги и количество их экземпляров.

Включение каждой записи должно уменьшать количество экземпляров заказанной книги в таблице **book** на выбранное пользователем количество.

Операция 3 – Формирование заказа



Операция 3 – формирование заказа




```
INSERT INTO buy_book (buy_id, book_id, amount)
VALUES (@buy_current, 3, 1);
```

Описание триггера

1. Триггер связан с таблицей **buy_book**.
2. Он должен быть запущен после (**AFTER**) добавления (**INSERT**).
3. Действия триггера:

```
UPDATE book
SET amount = amount - NEW.amount
WHERE book_id = NEW.book_id;
```

Типовые операции с данными

Операция 4. По концу года необходимо удалить все оплаченные заказы за текущий год и занести их в архив.

В базе данных из таблицы **buy** (и следовательно из **buy_book** и **buy_step**) удаляются все заказы, которые уже оплачены.

При удалении каждого заказа необходимо добавить информацию о нем в архивную таблицу **archieve_book**.

Операция 4 – очистка таблиц

```
DELETE FROM buy
    USING buy INNER JOIN buy_step ON buy.buy_id = buy_step.buy_id
WHERE YEAR(date_step_end) = YEAR(NOW())-1 and step_id =1;
```

Описание триггера

1. Триггер связан с таблицей **buy**.
2. Он должен быть запущен перед (**BEFORE**) удалением записей (**DELETE**).
3. Действия триггера:

```
INSERT INTO archive_book(client_id, book_id, date_payment, price, amount)
SELECT client_id, book_id, date_step_end, price, buy_book.amount
FROM book
      INNER JOIN buy_book USING(book_id)
      INNER JOIN buy USING(buy_id)
      INNER JOIN buy_step USING(buy_id)
      INNER JOIN client USING(client_id)
WHERE buy_id = OLD.buy_id and step_id = 1;
```

Резюме

Триггер позволяет описать действия, которые должны быть выполнены при наступлении определенных действий с данными в таблице.

Особенность триггеров заключается в том, что SQL код, написанный в теле триггера, будет исполнен после того, как в базе данных произойдет какое-либо событие.

События в базах данных происходят в результате выполнения запросов корректировки данных**: UPDATE**, **INSERT**, **DELETE**.

Резюме

Триггер – объект базы данных, поэтому имя триггера должно быть уникальным во всей базе данных.

Триггер можно

- создать;
- удалить;

Резюме

Синтаксис создания триггера:

```
CREATE TRIGGER имя_триггера
[AFTER | BEFORE] [INSERT | UPDATE | DELETE]
ON имя_таблицы
FOR EACH ROW
BEGIN
   запрос_или_выражение_sql_1;
   ...
   запрос_или_выражение_sql_N;
END;
```

Резюме

Синтаксис запроса на создание триггера:

```
CREATE TRIGGER имя_триггера
[AFTER | BEFORE] [INSERT | UPDATE | DELETE]
ON имя_таблицы
FOR EACH ROW
BEGIN
   запрос_или_выражение_sql_1;
   ...
   запрос_или_выражение_sql_N;
END;
```

Резюме

Синтаксис запроса на удаление триггера:

DROP TRIGGER имя_триггера;

Триггеры SQL

Перед созданием триггера необходимо ответить на следующие вопросы:

1. С какой таблицей он будет связан?
2. В какой момент времени он должен быть "запущен"? С какой операцией ассоциируется?
3. Что именно он должен делать?

Спасибо за внимание!