

Министерство образования Республики Беларусь

Учреждение образования
Белорусский государственный университет информатики и
радиоэлектроники

Факультет компьютерного проектирования
Кафедра инженерной психологии и эргономики
Дисциплина: Базы данных

Лабораторная работа № 7
«Триггеры. Механизм реализации транзакций»

Выполнил:
ст.гр. 113802
Разумов Д.А.

Проверила:
Василькова А.Н.

Минск 2022

Цель работы – изучить принципы работы триггеров в реляционных СУБД, формирование умений создания триггеров на добавление, изменение и удаление данных для создания пользовательских ограничений целостности данных, изучение механизма реализации транзакций.

Триггер — это специальный тип хранимой процедуры, которая автоматически вызывается, когда данные в определенной таблице добавляются, удаляются или изменяются с помощью SQL-предложений INSERT, DELETE или UPDATE. В зависимости от того, какое из событий, связанных с изменением данных, инициирует запуск триггера, он называется insert trigger, delete trigger или update trigger.

Создать новый триггер позволяет оператор:

```
CREATE TRIGGER trigger_name trigger_time trigger_event  
ON tbl_name FOR EACH ROW trigger_stmt;
```

Оператор создает триггер с именем `trigger_name`, привязанный к таблице `tbl_name`. Не допускается привязка триггера к временной таблице или представлению. Конструкция `trigger_time` указывает момент выполнения триггера и может принимать два значения:

- before – действия триггера производятся до выполнения операции изменения таблицы;
- after – действия триггера производятся после выполнения операции изменения таблицы.

Конструкция `trigger_event` показывает, на какое событие должен реагировать триггер, и может принимать три значения:

- insert – триггер привязан к событию вставки новой записи в таблицу;
- update – триггер привязан к событию обновления записи таблицы;
- delete – триггер привязан к событию удаления записей таблицы.

В таблице drivers создадим триггер, удаляющий записи об исполнителях, информация которых была удалена. Триггер delete_driver и его результат приведены на рисунке 1.

```

5 • CREATE TRIGGER delete_driver BEFORE DELETE ON drivers FOR EACH ROW
7 DELETE FROM traffics WHERE idDrivers = OLD.idDrivers;
3 • SET SQL_SAFE_UPDATES = 0;
9 • DELETE FROM traffics WHERE idDrivers = 3;
9
1

```

ult Grid Filter Rows: <input type="text"/> Edit: Export/Import:						
idTraffics	startPoint	endPoint	trafficLength	countStops	idCars	idDrivers
2	Брест	Москва	1062	5	2	3
3	Могилев	Берлин	1313	7	3	4
4	Минск	Прага	1297	7	4	5
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 1 – Триггер удаления водителя

В таблице drivers создадим триггер update_count_stops, который вызывается после обновления опыта и обновляет количество остановок. Исходный вид таблицы приведен на рисунке 2.

idDrivers	fullName	category	address	reputation	familyStatus	phone	callCode	dateOfBirth	experience	citizenship
1	Рогалевич Виктор Семенович	Крупногабаритный груз	г.Минск ул.Волоха 3-17	NULL	NULL	447104585	+375	1975-12-01	24	NULL
2	Ефренов Геннадий Викторович	Крупногабаритный груз	г.Минск ул.Маяковского 23-102	NULL	NULL	445629451	+375	1991-05-22	8	NULL
3	Чернов Юрий Павлович	Древесина	г.Минск ул.Дзержинского 88-137	Профи	NULL	331144185	+375	1971-03-12	27	NULL
4	Керножицкий Степан Дмитриевич	Техника	г.Минск ул.Случкое шоссе 61-67	Профи	NULL	294104582	+375	1968-01-22	26	NULL
5	Соколовский Валерий Александрович	Стекло	г.Минск ул.Карпова 39-14	NULL	NULL	295012127	+375	1988-10-13	9	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 2 – Исходный вид таблицы drivers

Изменение таблицы приведено на рисунке 3.

```
49 • CREATE TRIGGER update_experience BEFORE UPDATE ON drivers FOR EACH ROW
50 UPDATE experience.drivers
51 SET NEW.experience=10
52 WHERE (experience > 2 AND experience > 14);
53 • SET SQL_SAFE_UPDATES = 0;
54 • UPDATE drivers SET category="Крупногабаритный груз" WHERE reputation=NULL;
```

idDrivers	fullName	category	address	reputation	familyStatus	phone	callCode	dateOfBirth	experience	citizenship
1	Рогалевич Виктор Семенович	Крупногабаритный груз	г.Минск ул.Волоха 3-17	NULL	NULL	447104585	+375	1975-12-01	24	NULL
2	Ефремов Геннадий Викторович	Крупногабаритный груз	г.Минск ул.Маяковского 23-102	NULL	NULL	445629451	+375	1991-05-22	8	NULL
3	Чернов Юрий Павлович	Древесина	г.Минск ул.Дзержинского 88-137	Профи	NULL	331144185	+375	1971-03-12	27	NULL
4	Керножицкий Степан Дмитриевич	Техника	г.Минск ул.Случкое шоссе 61-67	Профи	NULL	294104582	+375	1968-01-22	26	NULL
5	Соколовский Валерий Александрович	Стекло	г.Минск ул.Карпова 39-14	NULL	NULL	295012127	+375	1988-10-13	9	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 3 – Триггер изменения опыта

В таблице traffics создадим триггер, удаляющий записи о маршрутах, информация которых была удалена. Исходная таблица traffics приведена на рисунке 4.

idTraffics	startPoint	endPoint	trafficLength	countStops	idCars	idDrivers
2	Брест	Москва	1062	6	2	3
3	Могилев	Берлин	1313	7	3	4
4	Минск	Прага	1297	7	4	5
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 4 – Исходная таблица traffics

Исходный вид таблицы drivers приведен на рисунке 5.

idDrivers	fullName	category	address	reputation	familyStatus	phone	callCode	dateOfBirth	experience	citizenship
1	Рогалевич Виктор Семенович	Крупногабаритный груз	г.Минск ул.Волоха 3-17	NULL	NULL	447104585	+375	1975-12-01	24	NULL
2	Ефремов Геннадий Викторович	Крупногабаритный груз	г.Минск ул.Маяковского 23-102	NULL	NULL	445629451	+375	1991-05-22	8	NULL
3	Чернов Юрий Павлович	Древесина	г.Минск ул.Дзержинского 88-137	Профи	NULL	331144185	+375	1971-03-12	27	NULL
4	Керножицкий Степан Дмитриевич	Техника	г.Минск ул.Случкое шоссе 61-67	Профи	NULL	294104582	+375	1968-01-22	26	NULL
5	Соколовский Валерий Александрович	Стекло	г.Минск ул.Карпова 39-14	NULL	NULL	295012127	+375	1988-10-13	9	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 5 – Исходный вид таблицы drivers

Триггер delete_driver и его результат приведены на рисунке 6.

```
59 • CREATE TRIGGER delete_driver BEFORE DELETE ON drivers
60   FOR EACH ROW
61   DELETE FROM traffics WHERE idDrivers=OLD.idDrivers;
62 • SET SQL_SAFE_UPDATES = 0;
63 • DELETE FROM drivers WHERE idDrivers = 3;
```

idDrivers	fullName	category	address	reputation	familyStatus	phone	callCode	dateOfBirth	experience	citizenship
1	Рогалевич Виктор Семенович	Крупногабаритный груз	г.Минск ул.Волоха 3-17	NULL	NULL	447104585	+375	1975-12-01	24	NULL
2	Ефренов Геннадий Викторович	Крупногабаритный груз	г.Минск ул.Маяковского 23-102	NULL	NULL	445629451	+375	1991-05-22	8	NULL
4	Керножицкий Степан Дмитриевич	Техника	г.Минск ул.Случокое шоссе 61-67	Профи	NULL	294104582	+375	1968-01-22	26	NULL
5	Соколовский Валерий Александрович	Стекло	г.Минск ул.Карпова 39-14	NULL	NULL	295012127	+375	1988-10-13	9	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 6 – Триггер delete_driver и его результат

Удаления триггеров. Удалить существующий триггер позволяет оператор

```
DROP TRIGGER trigger_name;
```

Триггеры можно использовать для проверки корректности ввода данных.

Транзакции

Транзакция является рабочей единицей работы с базой данных. Это последовательность операций, выполняемых в логическом порядке пользователем, либо программой, которая работает с БД.

Транзакция представляет собой группу запросов SQL, обрабатываемых атомарно, то есть как единое целое. Если подсистема базы данных может выполнить всю группу запросов, она делает это, но, если какой-либо запрос не может быть выполнен в результате сбоя или по иной причине, ни один запрос группы не будет выполнен.

Начинается транзакция ключевым словом BEGIN.

Завершается либо словом COMMIT (применить изменения), либо ROLLBACK (откатить изменения). Пример и результаты транзакции с командой COMMIT приведены на рисунках 7-8.

```
• BEGIN;  
• UPDATE traffics SET countStops = 6 WHERE idDrivers=3;  
• UPDATE cars SET yearOfIssue=2018 WHERE idCars=1;  
• COMMIT;
```

Рисунок 7 – Код транзакции

idTraffics	startPoint	endPoint	trafficLength	countStops	idCars	idDrivers
2	Брест	Москва	1062	6	2	3
3	Могилев	Берлин	1313	7	3	4
4	Минск	Прага	1297	7	4	5
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 8 – Результат транзакции

После завершения транзакции будет гарантировано, что оба запроса либо выполнились успешно, либо, если хотя бы один из них выполнить не удалось, не будет выполнен ни один из них.

Управление транзакцией

Следующие команды используются для управления операциями.

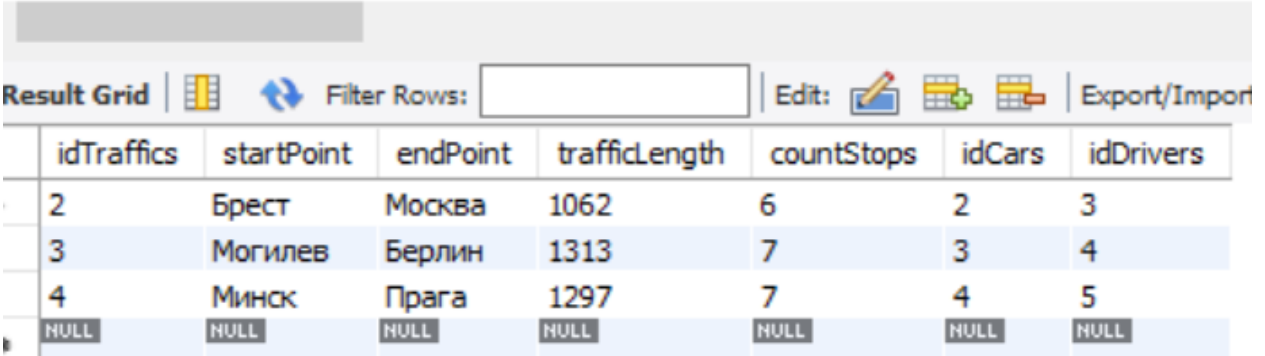
- COMMIT – для сохранения изменений.
- ROLLBACK – откат изменений.
- SAVEPOINT – создает точки внутри групп операций, которые следует откатить.
- SET TRANSACTION – размещает имя транзакции.

Команда ROLLBACK

Команда ROLLBACK является транзакционной командой и используется для отмены операций, которые еще не были сохранены в базе данных. Эта команда может быть использована только для отката транзакции после последних команд commit или rollback.

Применим команду rollback после удаления информации. Синтаксис команды и результат показан на рисунке 9.

```
29 • BEGIN;
30 • UPDATE traffics SET trafficLength = 1313
31   WHERE idTraffics = 3;
32 • UPDATE traffics SET countStops = 7
33   WHERE idTraffics = 3;
34 • ROLLBACK;
```



idTraffics	startPoint	endPoint	trafficLength	countStops	idCars	idDrivers
2	Брест	Москва	1062	6	2	3
3	Могилев	Берлин	1313	7	3	4
4	Минск	Прага	1297	7	4	5
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 9 – Код и результат выполнения команды ROLLBACK

Команда SAVEPOINT

SAVEPOINT это точка в транзакции, когда вы можете откатить транзакцию до определенной точки без отката всех транзакций.

На рисунке 10 приведен пример, в котором мы собираемся обновить 3 записи из таблицы `traffics`. Мы хотим создать `SAVEPOINT` перед каждым обновлением, так что мы можем откатить к любому `SAVEPOINT` в любое время, чтобы вернуть соответствующие данные в исходное состояние.

```
38 • BEGIN;
39 • SAVEPOINT sp1;
40 • UPDATE traffics SET countStops=6 WHERE idTraffics = 2;
41 • SAVEPOINT sp2;
42 • UPDATE traffics SET countStops=8 WHERE idTraffics = 3;
43 • SAVEPOINT sp3;
44 • UPDATE traffics SET countStops=8 WHERE idTraffics = 4;
45 • ROLLBACK TO sp2;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

idTraffics	startPoint	endPoint	trafficLength	countStops	idCars	idDrivers
2	Брест	Москва	1062	6	2	3
3	Могилев	Берлин	1313	7	3	4
4	Минск	Прага	1297	7	4	5
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 10 – Код и результат выполнения команды `ROLLBACK`

Команда `RELEASE SAVEPOINT`

Команда `RELEASE SAVEPOINT` используется для удаления `SAVEPOINT`, который вы создали.

Синтаксис команды `RELEASE SAVEPOINT` выглядит следующим образом:

```
BEGIN;
RELEASE SEVEPOINT sp;
```

После того, как `SAVEPOINT` был опубликован, вы больше не можете использовать команду `ROLLBACK`, чтобы отменить транзакции, выполненные с момента последнего `SAVEPOINT`.

Команда SET TRANSACTION

Команда SET TRANSACTION может быть использована для инициирования транзакции базы данных. Эта команда используется для определения характеристик для транзакции, которая следует. Например, вы можете указать транзакцию только для чтения или чтения и записи.