Лабораторная работа №7

Задание 1

Используя базу, полученную в лабораторной 2, создать транзакцию, произвести ее откат и фиксацию. Показать, что данные существовали до отката, удалились после отката, снова были добавлены, и затем были успешно зафиксированы. При необходимости используйте точки сохранения и вложенные транзакции.

```sql
-- Исходные данные в таблице Employee
SELECT * FROM Employee;
```

	id	employee_fio	role
1	1	Петров Иван Степанович	Кассир
2	2	Сидорова Мария Николаевна	Кассир
3	3	Козлов Алексей Александрович	Администратор
4	4	Николаева Елена Сергеевна	Кассир
5	5	Васильев Дмитрий Олегович	Старший кассир
6	6	Федорова Ольга Семёновна	Кассир
7	7	Иванов Максим Романович	Мерчандайзер
8	8	Павлова Анна Константиновна	Кассир
9	9	Смирнов Валентин Петрович	Администратор
10	10	Орлова Татьяна Львовна	Кассир
11	11	Морозов Виктор Михайлович	Грузчик
12	12	Волкова Наталья Васильевна	Кассир
13	13	Алексеев Павел Дмитриевич	Охранник
14	14	Лебедева Юлия Харитоновна	Кассир
15	15	Новиков Яков Андреевич	Уборщик

```sql
-- Начинаем транзакцию
BEGIN TRANSACTION;

-- Добавляем нового сотрудника
INSERT INTO Employee (employee_fio, role) VALUES
('Борисова Екатерина Андреевна', 'Стажёр');

-- Создаем точку сохранения
SAVE TRANSACTION save_point;

-- Добавляем ещё одного сотрудника
INSERT INTO Employee (employee_fio, role) VALUES
('Цветков Евгений Иванович', 'Товаровед');

-- Проверяем, что данные новых сотрудников добавлены
SELECT * FROM Employee WHERE role IN ('Стажёр', 'Товаровед');
```

Результаты | Сообщения

	id	employee_fio	role
1	16	Борисова Екатерина Андреевна	Стажёр
2	17	Цветков Евгений Иванович	Товаровед

```sql
-- Откатываем транзакцию до точки сохранения
ROLLBACK TRANSACTION save_point;

-- Проверяем, что данные первого сотрудника остались, а данные второго сотрудника были удалены
-- после отката
SELECT * FROM Employee WHERE role IN ('Стажёр', 'Товаровед');
```

	id	employee_fio	role
1	16	Борисова Екатерина Андреевна	Стажёр

```sql
-- После отката снова добавляем ещё одного сотрудника
INSERT INTO Employee (employee_fio, role) VALUES
('Цветков Евгений Иванович', 'Товаровед');

-- Завершаем транзакцию
COMMIT TRANSACTION;

-- Проверяем, что данные обоих сотрудников были сохранены
SELECT * FROM Employee WHERE role IN ('Стажёр', 'Товаровед');
```

	id	employee_fio	role
1	16	Борисова Екатерина Андреевна	Стажёр
2	18	Цветков Евгений Иванович	Товаровед

Задание 2

Подготовить SQL-скрипты для выполнения проверок изолированности транзакций. Ваши скрипты должны работать с одной из таблиц, созданных в лабораторной работе №2.

1. Уровень изоляции **READ UNCOMMITTED**

1) Проверка сценария потерянных данных:

```
-- Начальные данные:
```

Результаты | Сообщения

	id	employee_fio	role
1	1	Петров Иван Степанович	Обычный кассир
2	2	Сидорова Мария Николаевна	Кассир

```
-- T1 (окно 1) --
-- Устанавливаем уровень изоляции READ UNCOMMITTED
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

BEGIN TRANSACTION;

-- Чтение данных
SELECT role FROM Employee WHERE id = 1;

-- Ждём 5 секунд (в это время параллельно выполняется чтение и изменение в T2)
WAITFOR DELAY '00:00:05';

UPDATE Employee SET role = 'Кассир 1' WHERE id = 1;
COMMIT TRANSACTION;



-- T2 (окно 2) --
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
BEGIN TRANSACTION;

SELECT role FROM Employee WHERE id = 1;

UPDATE Employee SET role = 'Кассир 2' WHERE id = 1;
COMMIT TRANSACTION;



-- Результат: изменения, которые были ЗАФИКСИРОВАНЫ ПОСЛЕДНИМИ (в T1), сохранились в таблице, а
изменения в T2 были потеряны.
```

Результаты | Сообщения

	id	employee_fio	role
1	1	Петров Иван Степанович	Кассир 1
2	2	Сидорова Мария Николаевна	Кассир

2) Проверка грязного чтения:

```
-- Начальные данные:
```

Результаты | Сообщения

	id	employee_fio	role
1	1	Петров Иван Степанович	Обычный кассир
2	2	Сидорова Мария Николаевна	Кассир

```
-- T1 (окно 1) --
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
BEGIN TRANSACTION;

-- Меняем данные, но не фиксируем
UPDATE Employee SET role = 'Грязная роль' WHERE id = 1;
```

```sql
-- Пауза 10 секунд (запускаем T2)
WAITFOR DELAY '00:00:10';

-- Откатываем обновления назад
ROLLBACK;



-- T2 (окно 2) --
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

-- Ждём 3 секунды чтобы T1 успел изменить данные
WAITFOR DELAY '00:00:03';

BEGIN TRANSACTION;

-- Читаем незафиксированные данные
SELECT role as 'Грязное чтение' -- увидим 'Грязная роль'
FROM Employee WHERE id = 1;
COMMIT TRANSACTION;
```

Результаты | Сообщ

	Грязное чтение
1	Грязная роль

```sql
-- Ждём отката в T1
WAITFOR DELAY '00:00:08';

-- Проверяем какая информация хранится
SELECT role as 'После ROLLBACK T1' -- увидим исходное значение
FROM Employee WHERE id = 1;
```

	После ROLLBACK T1
1	Обычный кассир

```sql
-- Результат: T2 смогла прочитать незафиксированные в T1 данные и вывести их, после отката в T1
эта информация уже оказывается несуществующей.
```

2. Уровень изоляции **READ COMMITTED**

1) Проверка грязного чтения:

```sql
-- Повторяя проверку из предыдущего сценария, видим, что при таком уровне изоляции “грязное
чтение” невозможно, гарантируется чтение только зафиксированных данных.

-- Транзакция T2, пытаясь прочитать данные, заблокированные через UPDATE транзакцией T1,
переходит в состояние ожидания. SELECT в T2 выполнится только после того, как T1 завершит работу
(в данном примере это через ROLLBACK).
```

Результаты | Сообщен

	Грязное чтение
1	Обычный кассир

	После ROLLBACK T1
1	Обычный кассир

2) Проверка неповторяющегося чтения:

```sql
-- Начальные данные:
```

Результаты | Сообщения

	id	employee_fio	role
1	1	Петров Иван Степанович	Обычный кассир
2	2	Сидорова Мария Николаевна	Кассир

```sql
-- T1 (окно 1) --
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
```

```sql
BEGIN TRANSACTION;

-- Первое чтение
SELECT 'Первое чтение' AS information, role
FROM Employee WHERE id = 1;
```

Результаты | Сообщения

	information	role
1	Первое чтение	Обычный кассир

```sql
-- Пауза 15 секунд (запускаем T2)
WAITFOR DELAY '00:00:15';

-- Второе чтение (уже будут другие данные)
SELECT 'Второе чтение' AS information, role
FROM Employee WHERE id = 1;

COMMIT TRANSACTION;
```

	information	role
1	Второе чтение	Лучший кассир

```sql
-- T2 (окно 2) --
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

-- Ждем 5 секунд чтобы T1 успел прочитать первый раз
WAITFOR DELAY '00:00:05';

-- Меняем данные
BEGIN TRANSACTION;
UPDATE Employee SET role = 'Лучший кассир' WHERE id = 1;
COMMIT TRANSACTION;



-- Результат: T1 выполнила два последовательных SELECT одних и тех же данных, но между этими
чтениями T2 изменила и зафиксировала данные, из-за чего T1 получила разные значения при первом и
втором чтении, что нарушает согласованность данных в рамках одной транзакции.
```

3. Уровень изоляции **REPEATABLE READ**

1) Проверка неповторяющегося чтения:

```sql
-- На данном уровне изоляции неповторяющееся чтение невозможно. Транзакция T1 при повторном
чтении получает идентичные данные, так как изменение этих строк другими транзакциями до
завершения T1 заблокировано.

-- При проверке примера из предыдущего пункта, можно увидеть, что T1 вывел одинаковые данные,
неповторяющееся чтение не происходит.
```

Результаты | Сообщения

	information	role
1	Первое чтение	Обычный кассир

	information	role
1	Второе чтение	Обычный кассир

2) Проверка фантома:

```sql
-- T1 (окно 1) --
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
BEGIN TRANSACTION;

-- Первая проверка
SELECT 'Первая проверка' as information, employee_fio, role
FROM Employee
```

```sql
WHERE role = 'Администратор';
```

Результаты | Сообщения

	information	employee_fio	role
1	Первая проверка	Козлов Алексей Александрович	Администратор
2	Первая проверка	Смирнов Валентин Петрович	Администратор

```sql
-- Время для T2 на изменения
WAITFOR DELAY '00:00:10';

-- Вторая проверка
SELECT 'Вторая проверка' as information, employee_fio, role
FROM Employee
WHERE role = 'Администратор';
COMMIT TRANSACTION;
```

	information	employee_fio	role
1	Вторая проверка	Козлов Алексей Александрович	Администратор
2	Вторая проверка	Смирнов Валентин Петрович	Администратор
3	Вторая проверка	Фантомный Администратор (новый)	Администратор

```sql
-- T2 (окно 2) -
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

-- Время T1 на чтение таблицы
WAITFOR DELAY '00:00:05';

BEGIN TRANSACTION;
INSERT INTO Employee (employee_fio, role)
VALUES ('Фантомный Администратор (новый)', 'Администратор');
COMMIT TRANSACTION;

-- Результат: T1 при повторном выполнении SELECT видит новые строки ('фантомы'), добавленные другой транзакцией T2 между операциями чтения. REPEATABLE READ предотвращает изменение уже прочитанных строк, но не блокирует добавление новых строк, соответствующих критериям выборки.
```

4. Уровень изоляции **SERIALIZABLE**

1) Проверка фантома:

```sql
-- Повторив предыдущий пример для данного уровня изоляции, увидим, что невозможно не только изменение существующих строк, но и добавление новых строк. Транзакция T2, пытающаяся вставить новую фантомную запись, блокируется до завершения транзакции T1.
```

Проблемы не найдены.

Результаты | Сообщения

	information	employee_fio	role
1	Первая проверка	Козлов Алексей Александрович	Администратор
2	Первая проверка	Смирнов Валентин Петрович	Администратор

	information	employee_fio	role
1	Вторая проверка	Козлов Алексей Александрович	Администратор
2	Вторая проверка	Смирнов Валентин Петрович	Администратор