**Цель работы**: Научиться работать с транзакциями в СУБД MySQL и PostgreSQL, создавать пользователей базы данных с разными привилегиями, а также создавать check- ограничения.

**Задание**

- Создать транзакцию, включающую 2-3 оператора INSERT,UPDATE или DELETE, вносящие изменения в таблицу БД. Убедится в присутствии изменений в таблице БД (командой SELECT). Затем откатить транзакцию и убедится, что таблица возвратилась к состоянию, в котором она находилась до начала транзакции. Создать транзакцию, вносящую изменения в таблицу БД. Убедится в присутствии изменений в таблице БД (командой SELECT). Затем фиксировать транзакцию и убедится, что изменения сохранились в таблице. Задание выполнить в консольных клиентах MySQL и PostgreSQL.

- Создать двух пользователей. Установить пользователям привилегии:

первому – SELECT, второму – SELECT, INSERT и UPDATE. Проверить установленные привилегии. Отменить у второго пользователя привилегию UPDATE. Проверить установленные привилегии. Выполнить от имени нового пользователя некоторые выборки и изменения (с ограниченными правами доступа) в таблицах БД. Убедится в правильном контроле прав доступа. Задание выполнить в консольных клиентах MySQL и PostgreSQL.

- Наложить check- ограничения на столбцы таблиц баз данных, созданных в л. р. № 3. Проверить работоспособность ограничений путем добавления в таблицы данных удовлетворяющих и не удовлетворяющих условиям ограничений. Задание выполнить в консольном клиенте PostgreSQL.

- **Дополнительное задание для повышения рейтинга:** Подготовить SQL-скрипты для проверки наличия аномалий (потерянных изменений, грязных чтений, неповторяющихся чтений, фантомов) при параллельном исполнении транзакций на различных уровнях изолированности (READ UNCOMMITTED, READ COMMITTED, REPEATABLE READ, SERIALIZABLE). Скрипты должны работать с одной из таблиц БД, созданной в л. р. № 3. Для проверки наличия аномалий потребуются два параллельных сеанса, операторы в которых выполняются пошагово.

* Установить в обоих сеансах уровень изоляции READ UNCOMMITTED. Выполнить сценарии проверки наличия аномалий потерянных изменений и грязных чтений.
* Установить в обоих сеансах уровень изоляции READ COMMITTED. Выполнить сценарии проверки наличия аномалий грязных чтений и неповторяющихся чтений.
* Установить в обоих сеансах уровень изоляции REPEATABLE READ. Выполнить сценарии проверки наличия аномалий неповторяющихся чтений и фантомов.
* Установить в обоих сеансах уровень изоляции SERIALIZABLE. Выполнить сценарии проверки наличия фантомов.

Дополнительное задание выполнить в консольном клиенте MySQL или PostgreSQL.

**// MySQL**

Создать транзакцию, включающую 2-3 оператора INSERT,UPDATE или DELETE, вносящие изменения в таблицу БД.

START TRANSACTION;

update musikalnaya\_gruppa set God\_obrazovaniya = 1200

where Nazvanie\_gruppi = 'blaky';

update musikalnaya\_gruppa set Strana = 'Russia'

where Nazvanie\_gruppi = 'Bastille';

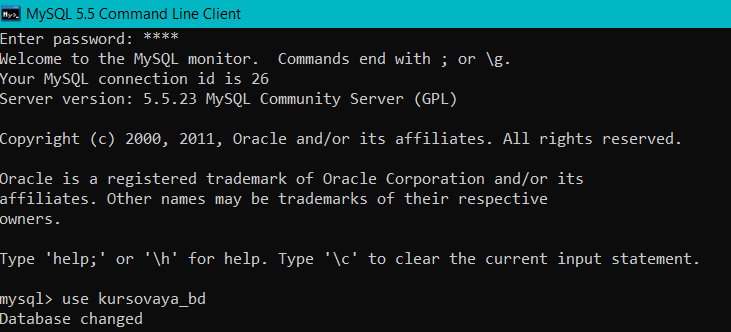


Рисунок 1 - подключение к базе данных

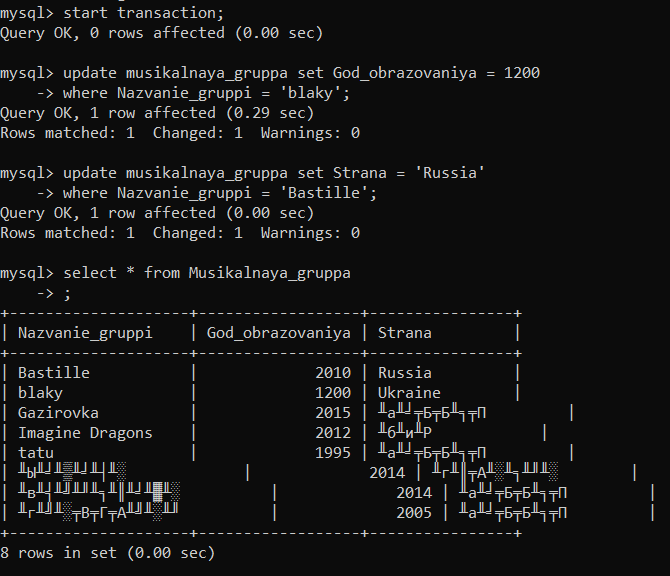


Рисунок 2 - создание транзакции, включающей в себя два оператора UPDATE и проверка оператором SELECT

Затем откатить транзакцию и убедится, что таблица возвратилась к состоянию, в котором она находилась до начала транзакции.

Чтобы откатить транзакцию, необходимо воспользоваться оператором ROLLBACK.

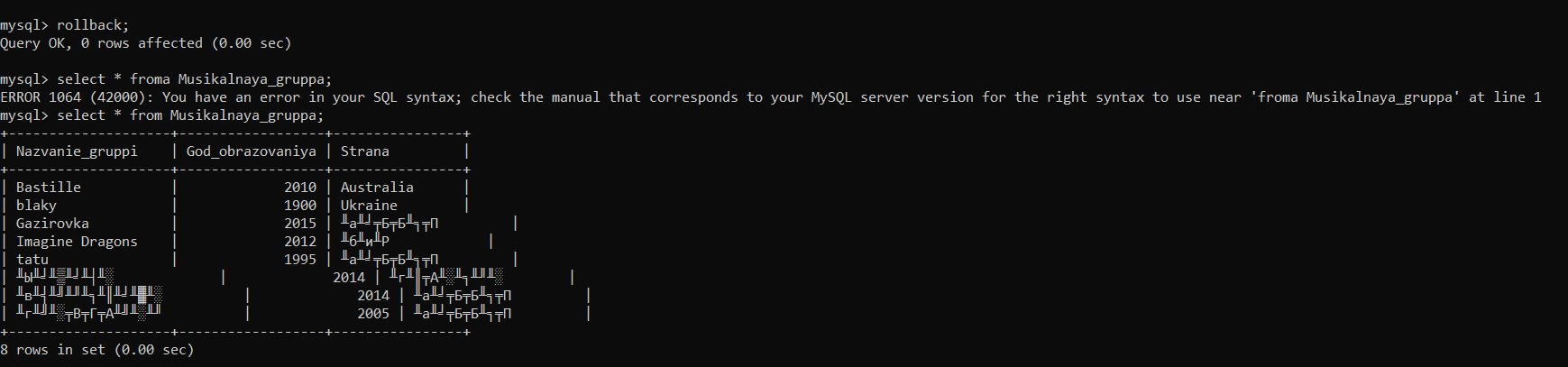


Рисунок 3 - откат транзакции и проверка возврата к исходным данным оператором SELECT

Затем фиксировать транзакцию и убедится, что изменения сохранились в таблице.

Чтобы фиксировать транзакцию, необходимо воспользоваться оператором COMMIT.

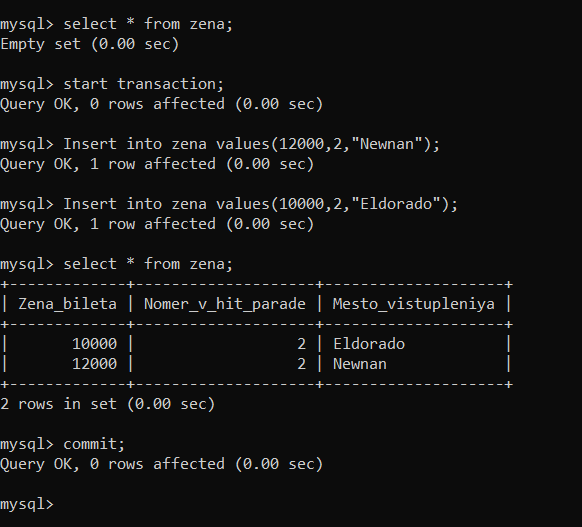


Рисунок 4 - создание транзакции, включающей два оператора INSERT, проверка оператором SELECT, а также последующая фиксация транзакции

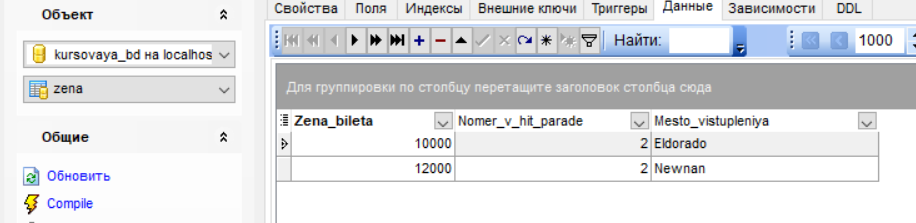


Рисунок 5 - проверка фиксации транзакции в менеджере

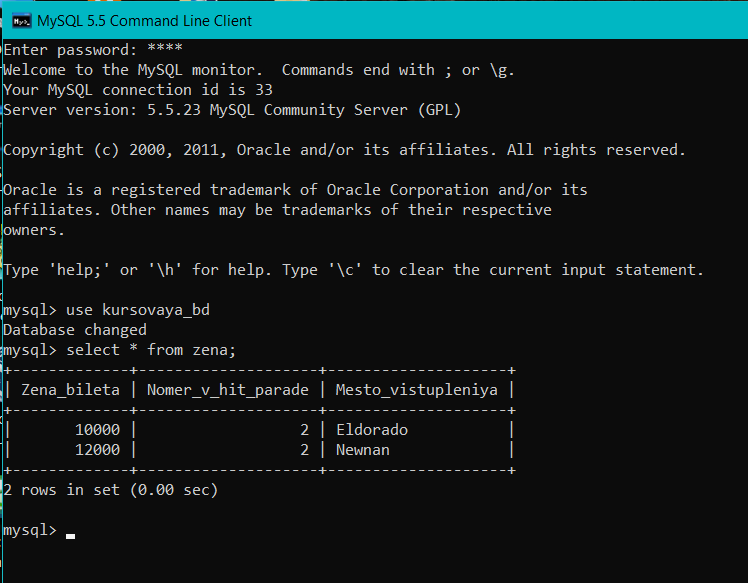


Рисунок 6 - проверка фиксации транзакции в консоли

Создать двух пользователей.

Установить пользователям привилегии:

первому – SELECT, второму – SELECT, INSERT и UPDATE. Проверить установленные привилегии. Отменить у второго пользователя привилегию UPDATE. Проверить установленные привилегии. Выполнить от имени нового пользователя некоторые выборки и изменения (с ограниченными правами доступа) в таблицах БД. Убедится в правильном контроле прав доступа.

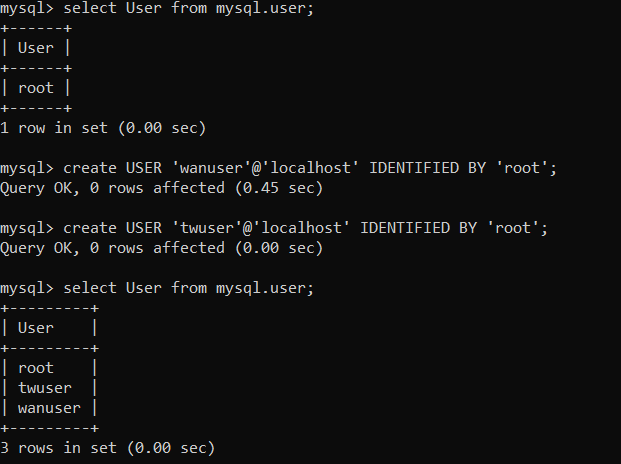


Рисунок 7 - создание двух пользователей

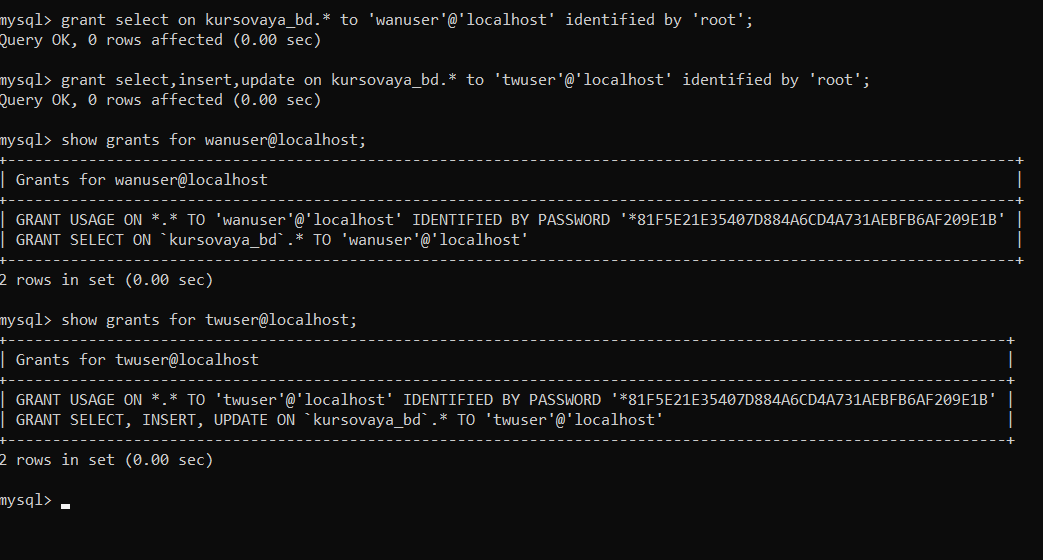


Рисунок 8 - добавлений привилегий пользователям

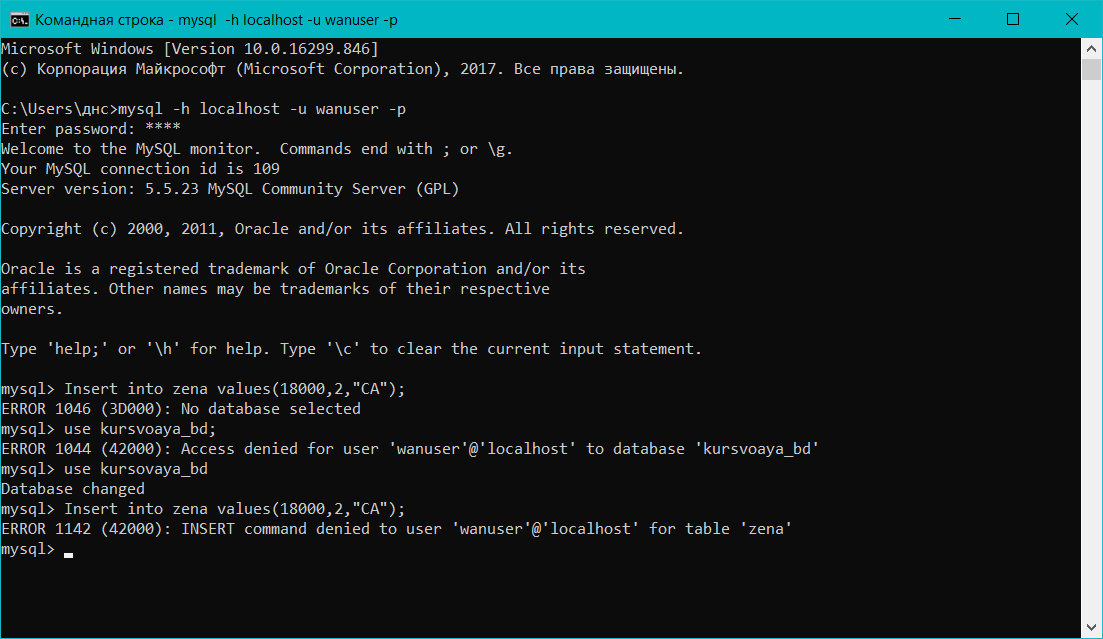


Рисунок 9 - проверка привилегий первого пользователя

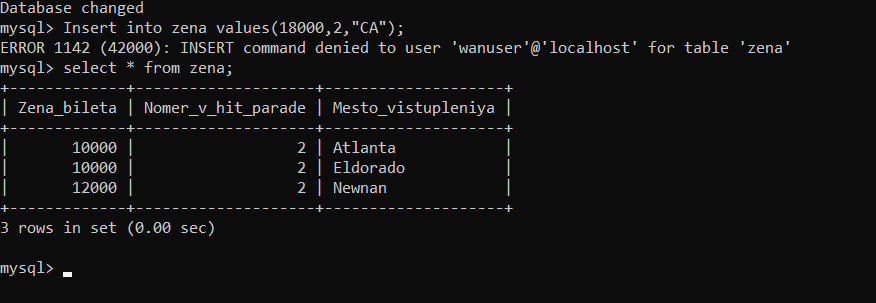


Рисунок 10 - SELECT первому пользователю доступно

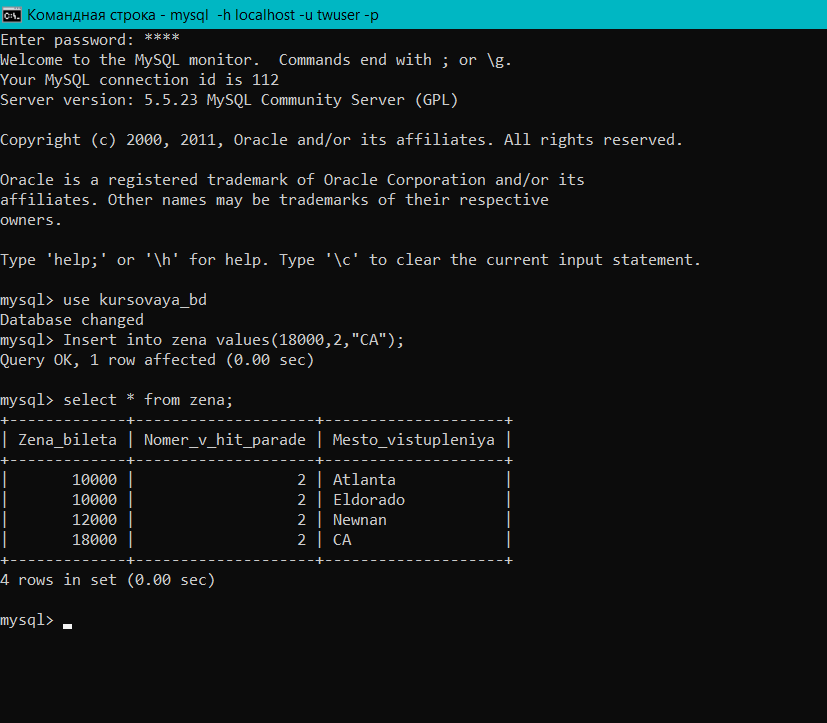


Рисунок 11 - проверка привилегий второго пользователя

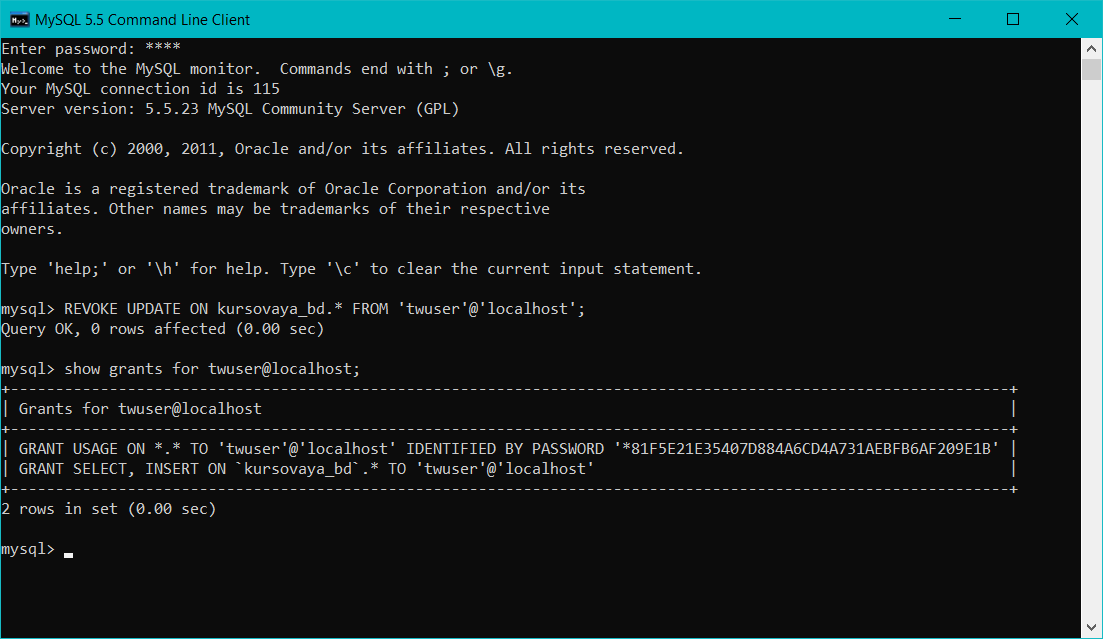


Рисунок 12 - удаление привилегии UPDATE у второго пользователя

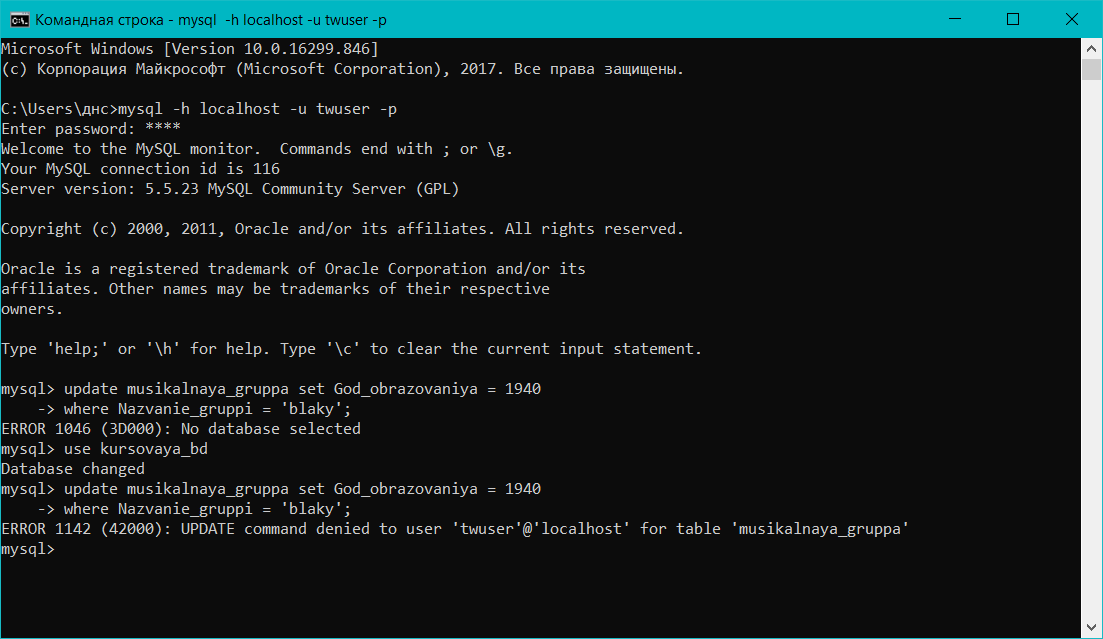


Рисунок 13 - проверка удалённой привилегии у второго пользователя

**// PostgreSQL**

Создать транзакцию, включающую 2-3 оператора INSERT,UPDATE или DELETE, вносящие изменения в таблицу БД.

BEGIN;

update musikalnaya\_gruppa set Strana = 'Черногория'

where Nazvanie\_gruppi = 'tatu';

update musikalnaya\_gruppa set God\_obrazovaniya = 1970

where Nazvanie\_gruppi = 'black';

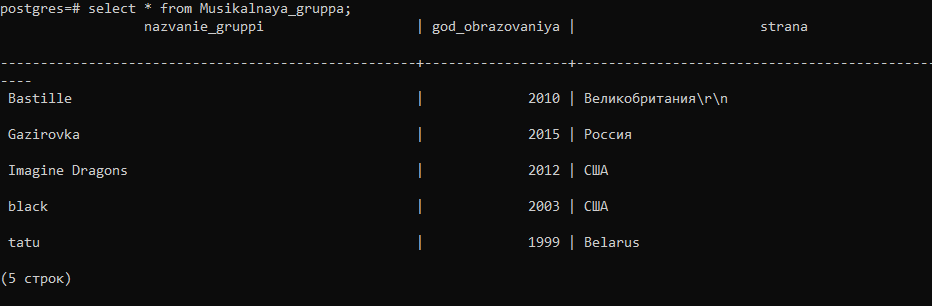


Рисунок 14 - состояние таблицы «Музыкальная группа» до транзакции

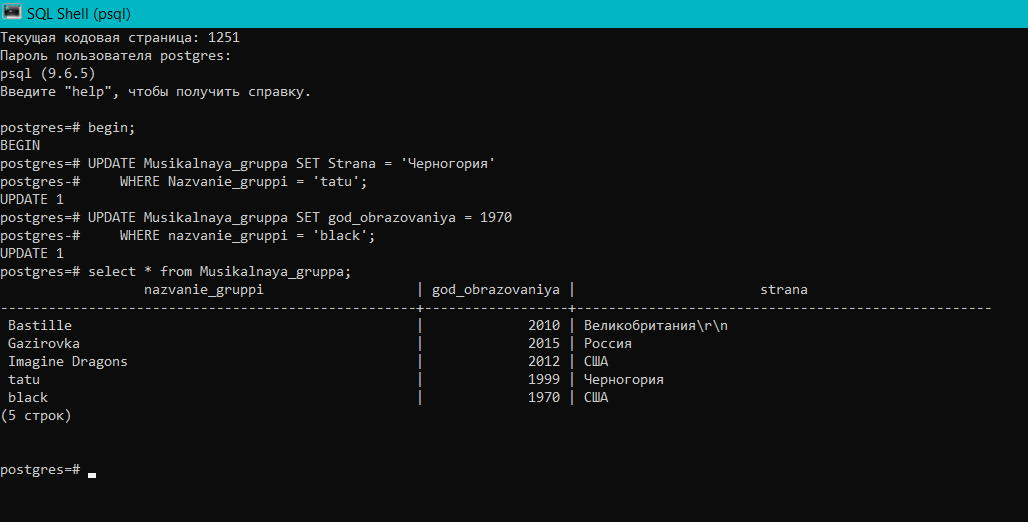


Рисунок 14 - внесение изменений в таблицу оператором UPDATE и проверка оператором SELECT

Затем откатить транзакцию и убедится, что таблица возвратилась к состоянию, в котором она находилась до начала транзакции.

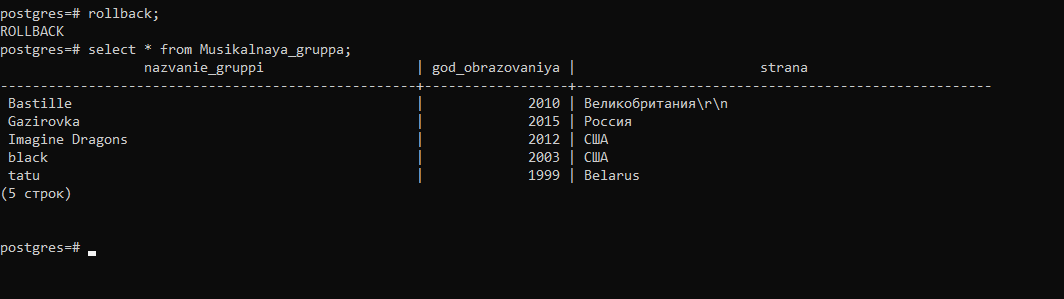
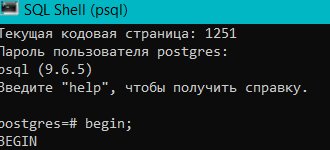


Рисунок 15 - откат транзакции оператором ROLLBACK и проверка возврата к изначальному состоянию таблицы оператором SELECT

Создать транзакцию, вносящую изменения в таблицу БД. Убедится в присутствии изменений в таблице БД (командой SELECT). Затем фиксировать транзакцию и убедится, что изменения сохранились в таблице.



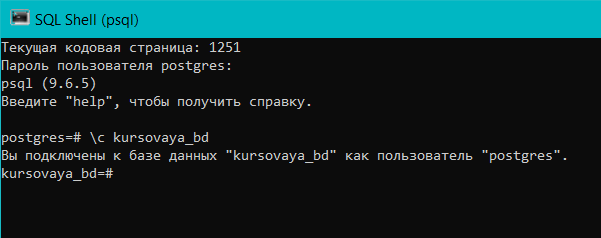


Рисунок 16 - создание новой транзакции и подключение к базе данных

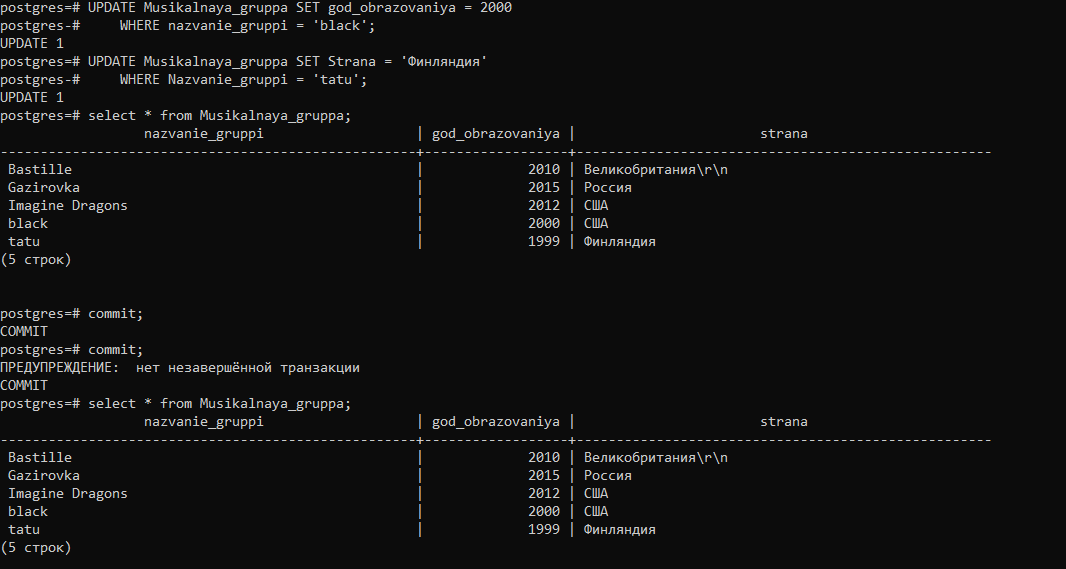


Рисунок 17 - создание транзакции, внесение изменений в базу данных и проверка оператором SELECT, а также фиксация транзакции и проверка фиксации

Создать двух пользователей. Установить пользователям привилегии:

первому – SELECT, второму – SELECT, INSERT и UPDATE. Проверить установленные привилегии. Отменить у второго пользователя привилегию UPDATE. Проверить установленные привилегии. Выполнить от имени нового пользователя некоторые выборки и изменения (с ограниченными правами доступа) в таблицах БД. Убедится в правильном контроле прав доступа.

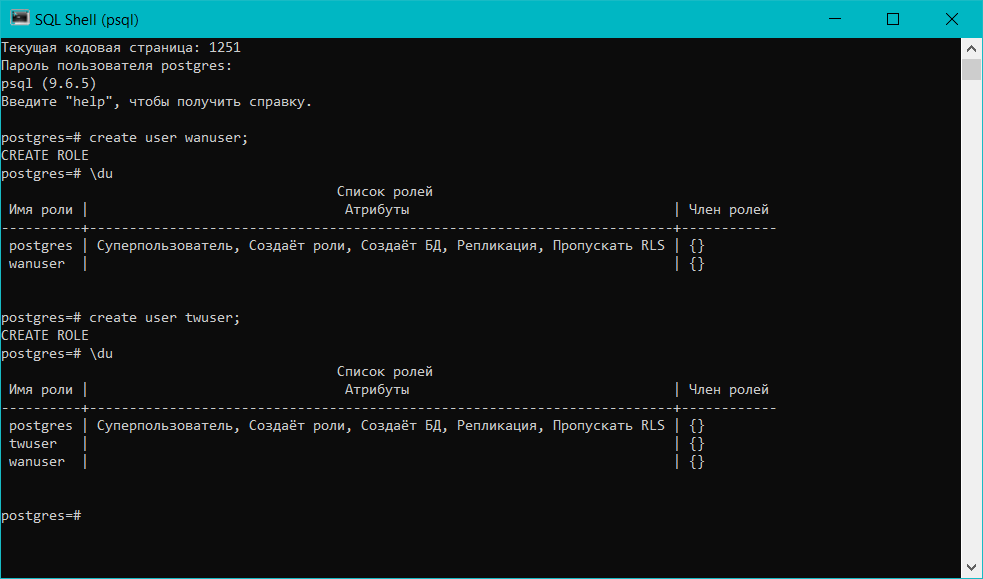


Рисунок 18 - создание двух пользователей в PostgreSQL

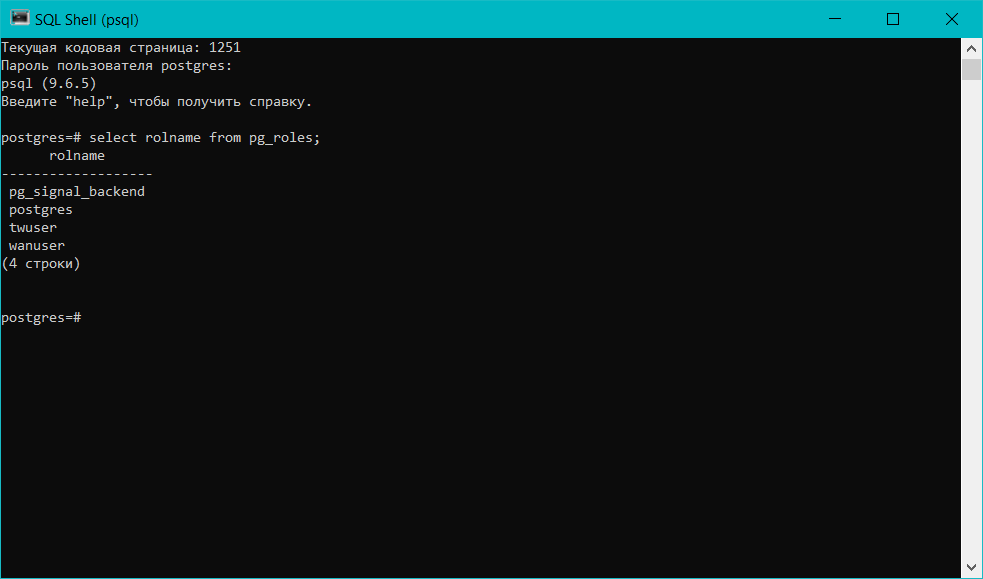
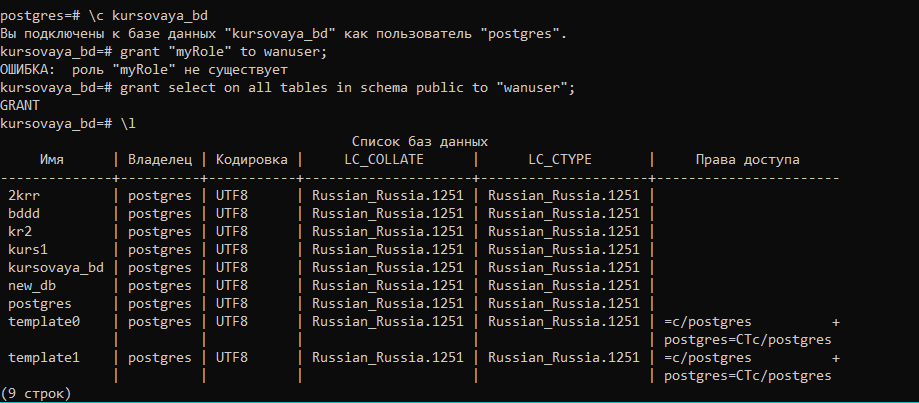


Рисунок 19 - просмотр существующих ролей



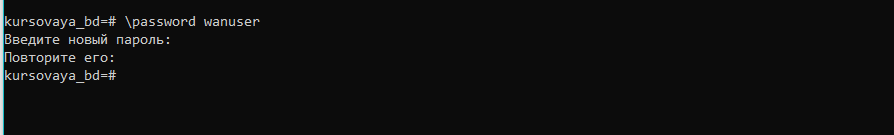


Рисунок 20 - подключение к базе данных и задание новому пользователю wanuser пароля

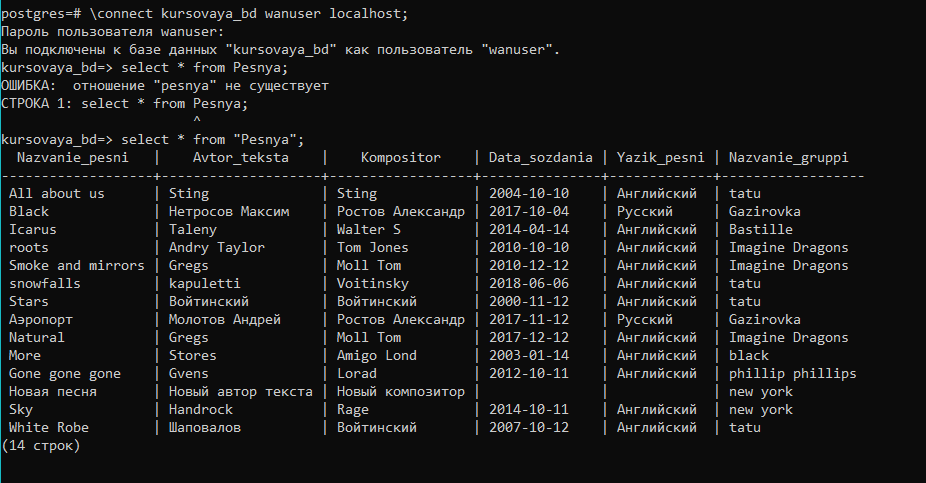


Рисунок 21 - вход под другим пользователем и подключение к базе данных, а также проверка привилегий

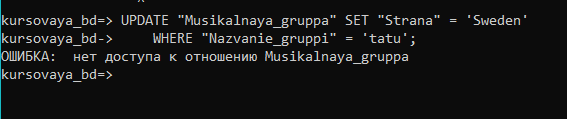


Рисунок 22 - проверка привилегий (у данного пользователя нет привилегий UPDATE)

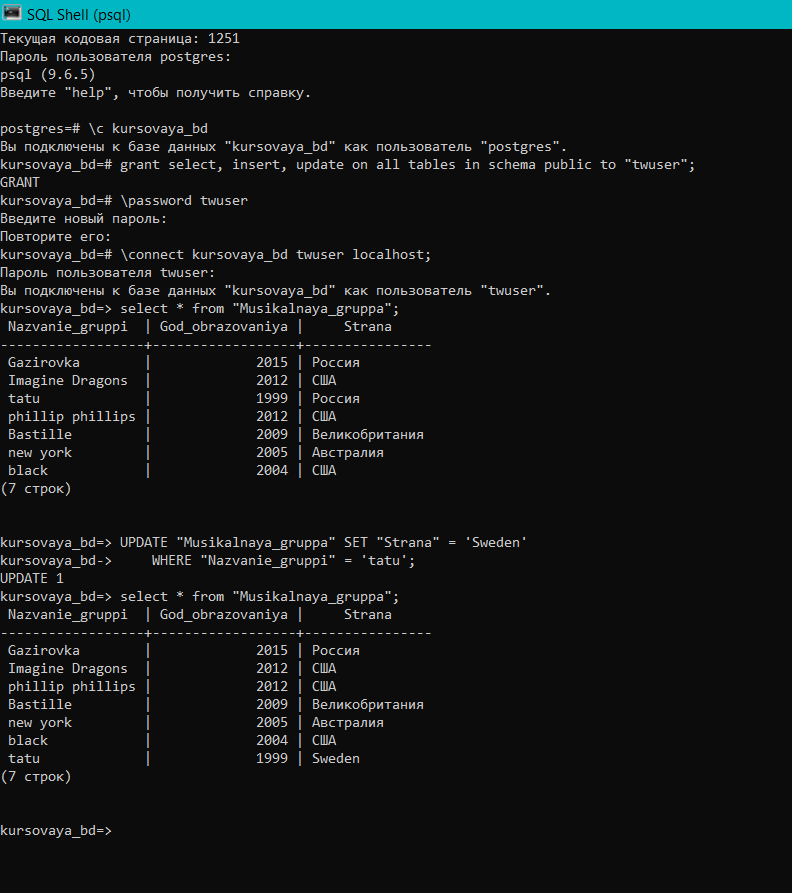


Рисунок 23 - вход под вторым пользователем и подключение к базе данных, а также установка привилегий и их проверка

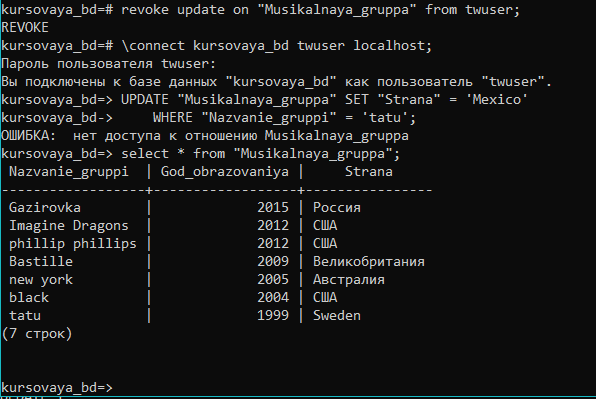


Рисунок 24 - отмена привилегий у второго пользователя (UPDATE), вход под вторым пользователем в базу данных, проверка работы оператора revoke, проверка работы других привилегий (которые остались у второго пользователя, например, SELECT)

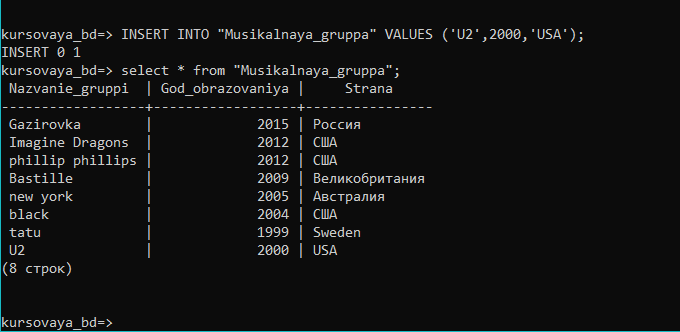


Рисунок 25 - второй пользователь также имеет привилегию INSERT (после отмены привилегии UPDATE это никак не отразилось на других привилегиях)

Наложить check- ограничения на столбцы таблиц баз данных, созданных в л. р. № 3. Проверить работоспособность ограничений путём добавления в таблицы данных удовлетворяющих и не удовлетворяющих условиям ограничений. Задание выполнить в консольном клиенте PostgreSQL.

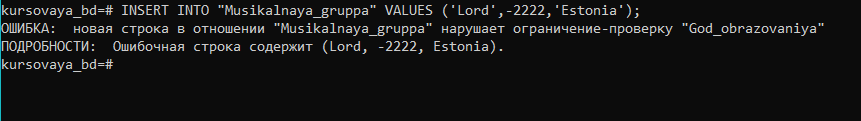


Рисунок 26 - наложение check-ограничения на столбец «Год\_образования» музыкальной группы (год образования не может быть отрицательным и равным нулю), а также проверка данного ограничения

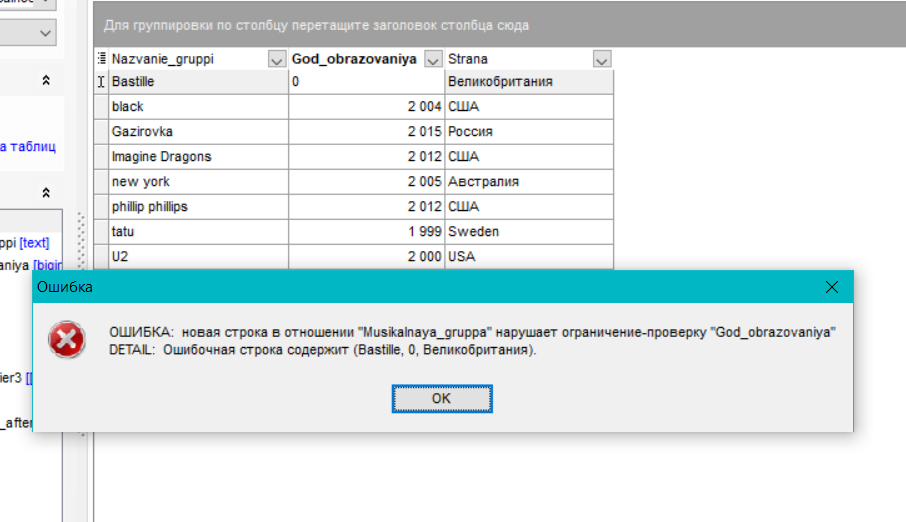


Рисунок 27 - проверка check-ограничения в менеджере

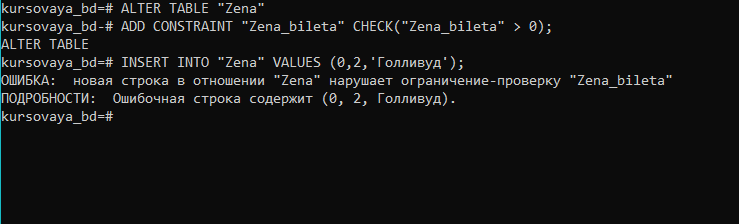


Рисунок 28 - check-ограничение на цену билета (цена не может быть равной нулю)

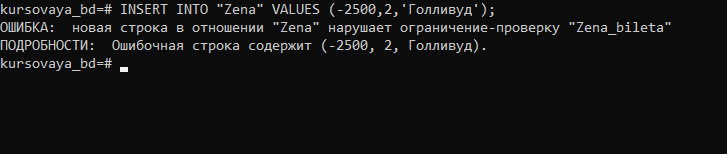


Рисунок 29 - check-ограничение на цену билета (цена не может быть отрицательной)

Подготовить SQL-скрипты для проверки наличия аномалий (потерянных изменений, грязных чтений, неповторяющихся чтений, фантомов) при параллельном исполнении транзакций на различных уровнях изолированности (READ UNCOMMITTED, READ COMMITTED, REPEATABLE READ, SERIALIZABLE). Скрипты должны работать с одной из таблиц БД, созданной в л. р. № 3. Для проверки наличия аномалий потребуются два параллельных сеанса, операторы в которых выполняются пошагово.

* Установить в обоих сеансах уровень изоляции READ UNCOMMITTED. Выполнить сценарии проверки наличия аномалий потерянных изменений и грязных чтений.
* Установить в обоих сеансах уровень изоляции READ COMMITTED. Выполнить сценарии проверки наличия аномалий грязных чтений и неповторяющихся чтений.
* Установить в обоих сеансах уровень изоляции REPEATABLE READ. Выполнить сценарии проверки наличия аномалий неповторяющихся чтений и фантомов.
* Установить в обоих сеансах уровень изоляции SERIALIZABLE. Выполнить сценарии проверки наличия фантомов.

**// MySQL**

Уровень изоляции : READ UNCOMMITTED

- Потерянные изменения

T1

use kursovaya\_bd

SET GLOBAL TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

start transaction;

update zena set Zena\_bileta = 1111111

where Mesto\_vistupleniya = 'CA';

commit;

T2

use kursovaya\_bd

SET GLOBAL TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

start transaction;

update zena set Zena\_bileta = 2222222

where Mesto\_vistupleniya = 'CA';

commit;

select \* from zena;

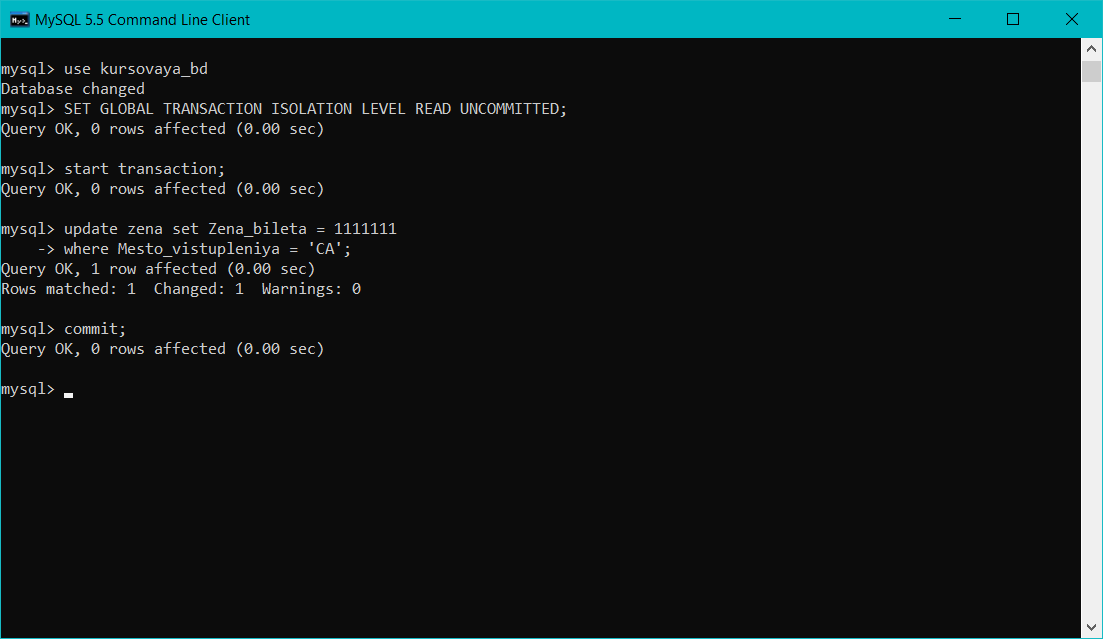


Рисунок 30 - Т1

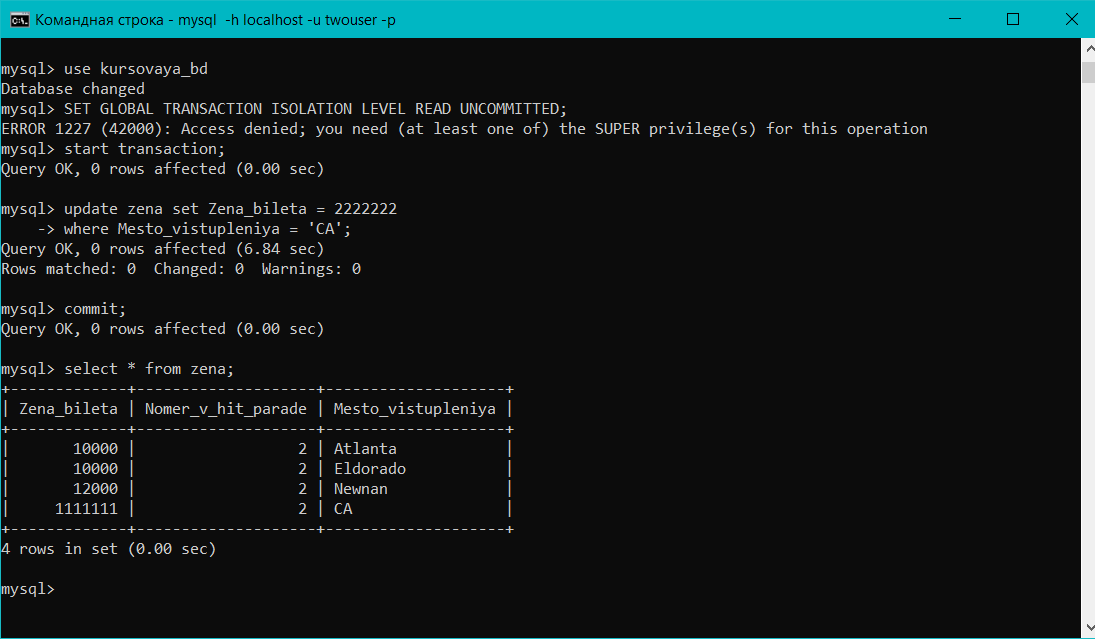


Рисунок 31 - Т2



Рисунок 32 - данные в базе

- Грязное чтение

T1

use kursovaya\_bd

SET GLOBAL TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

start transaction;

select \* from zena where Mesto\_vistupleniya = 'CA';

T2

use kursovaya\_bd

SET GLOBAL TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

start transaction;

update zena set Zena\_bileta = 88888 where Mesto\_vistupleniya = 'CA';

T1

select \* from zena where Mesto\_vistupleniya = 'CA';

T2

rollback;

T1

select \* from zena where Mesto\_vistupleniya = 'CA';

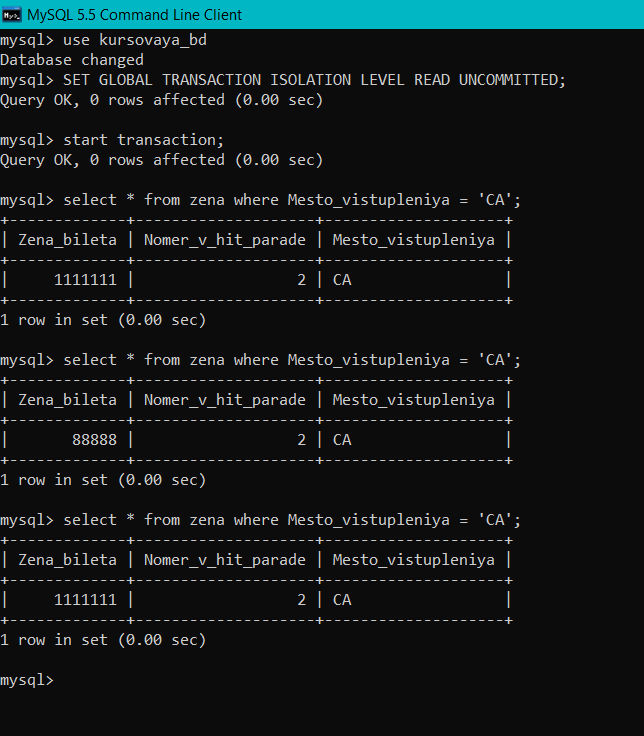


Рисунок 33 - Т1

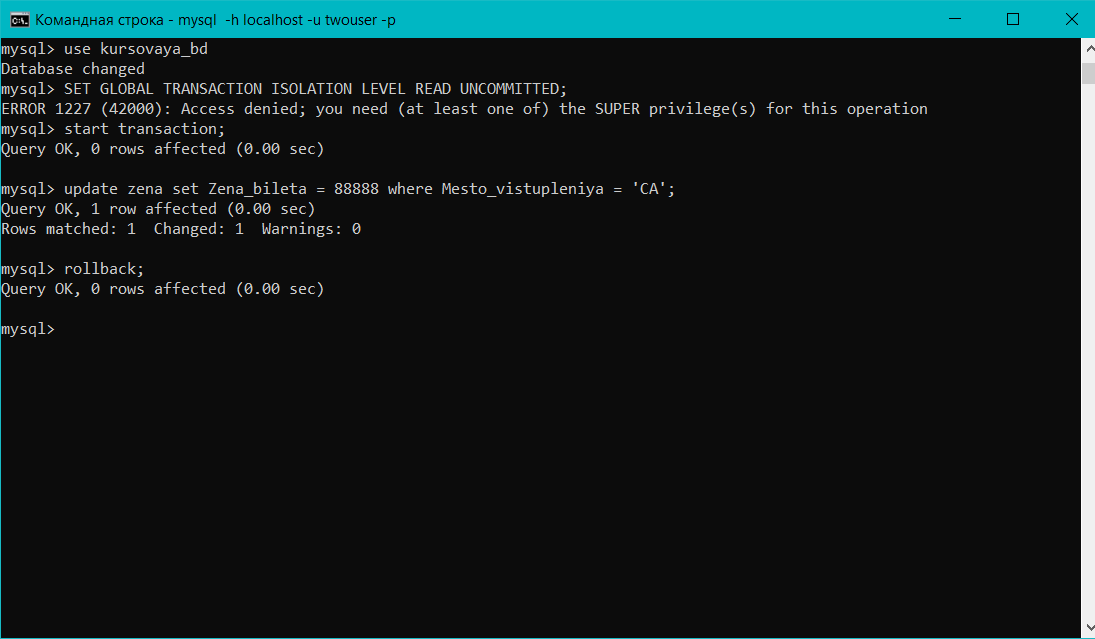


Рисунок 34 - Т2

Уровень изоляции : READ COMMITTED

- Грязное чтение

T1

use kursovaya\_bd

SET GLOBAL TRANSACTION ISOLATION LEVEL READ COMMITTED;

start transaction;

select \* from zena where Mesto\_vistupleniya = 'CA';

T2

use kursovaya\_bd

SET GLOBAL TRANSACTION ISOLATION LEVEL READ COMMITTED;

start transaction;

update zena set Zena\_bileta = 123456789 where Mesto\_vistupleniya = 'CA';

T1

select \* from zena where Mesto\_vistupleniya = 'CA';

T2

rollback;

T1

select \* from zena where Mesto\_vistupleniya = 'CA';

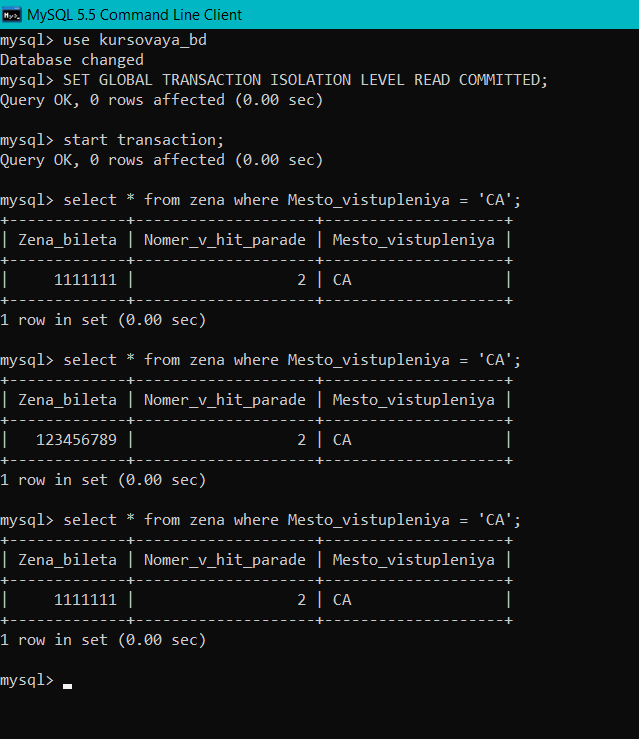


Рисунок 35 - T1

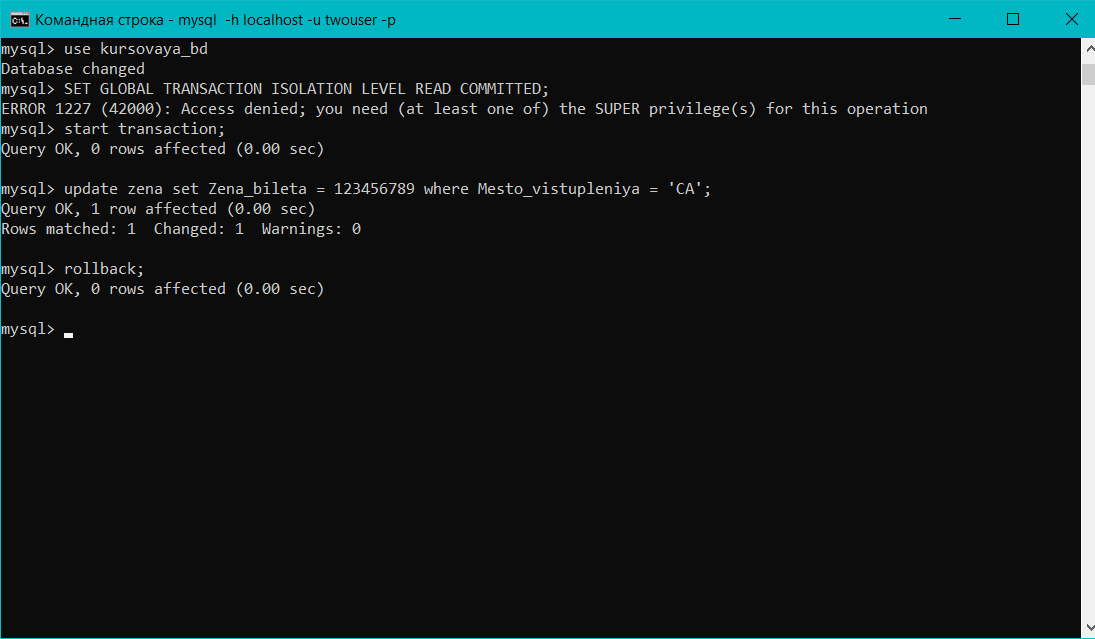


Рисунок 36 - T2

- Неповторяющиеся чтения

T1

use kursovaya\_bd

SET GLOBAL TRANSACTION ISOLATION LEVEL READ COMMITTED;

start transaction;

select \* from zena where Mesto\_vistupleniya = 'CA';

T2

use kursovaya\_bd

SET GLOBAL TRANSACTION ISOLATION LEVEL READ COMMITTED;

start transaction;

update zena set Zena\_bileta = 33333333 where Mesto\_vistupleniya = 'CA';

commit;

T1

select \* from zena where Mesto\_vistupleniya = 'CA';

commit;

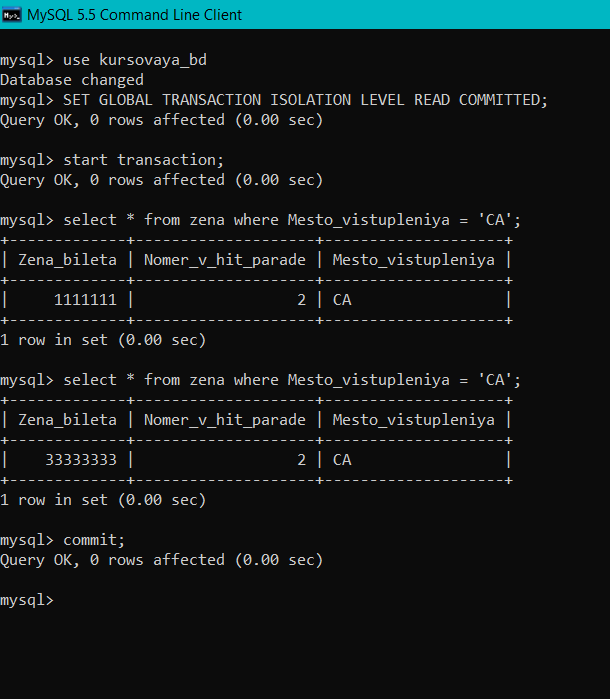


Рисунок 37 - Т1

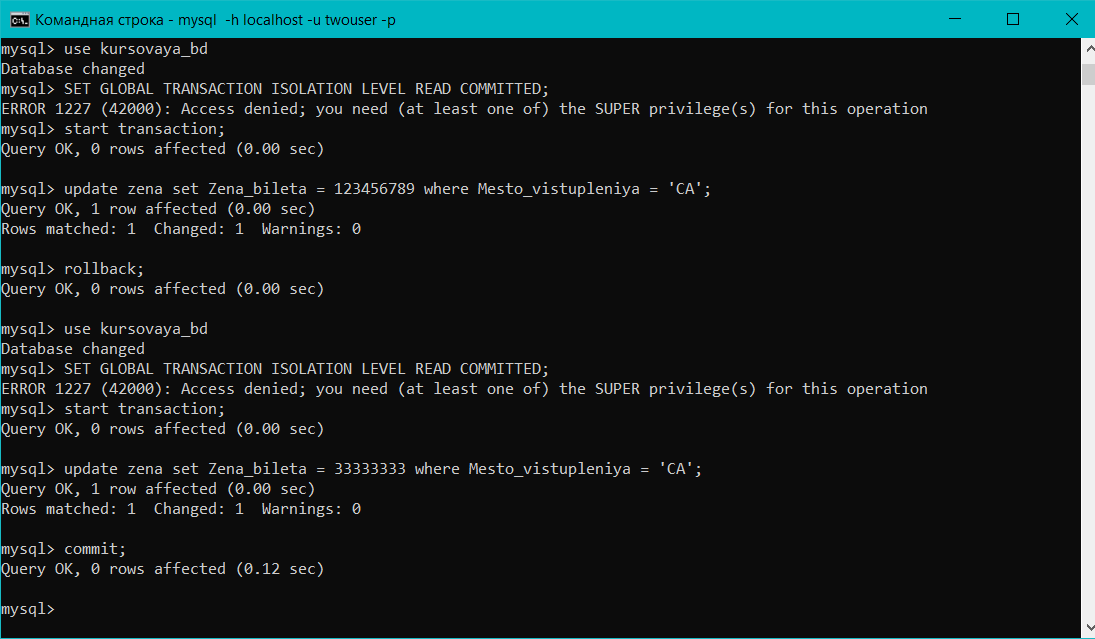


Рисунок 38 - T2

Установить в обоих сеансах уровень изоляции REPEATABLE READ. Выполнить сценарии проверки наличия аномалий неповторяющихся чтений и фантомов.

Уровень изоляции : REPEATABLE READ

- Неповторяющиеся чтения

T1

use kursovaya\_bd

SET GLOBAL TRANSACTION ISOLATION LEVEL REPEATABLE READ;

start transaction;

select \* from zena where Mesto\_vistupleniya = 'CA';

T2

use kursovaya\_bd

SET GLOBAL TRANSACTION ISOLATION LEVEL REPEATABLE READ;

start transaction;

update zena set Zena\_bileta = 17171717 where Mesto\_vistupleniya = 'CA';

commit;

T1

select \* from zena where Mesto\_vistupleniya = 'CA';

commit;

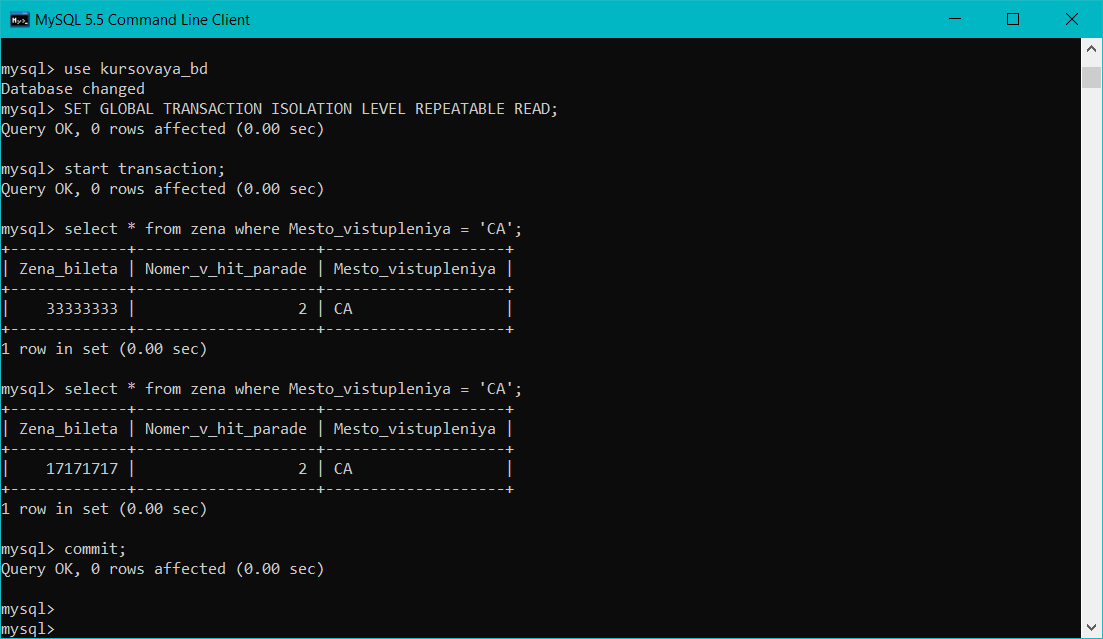


Рисунок 39 - T1

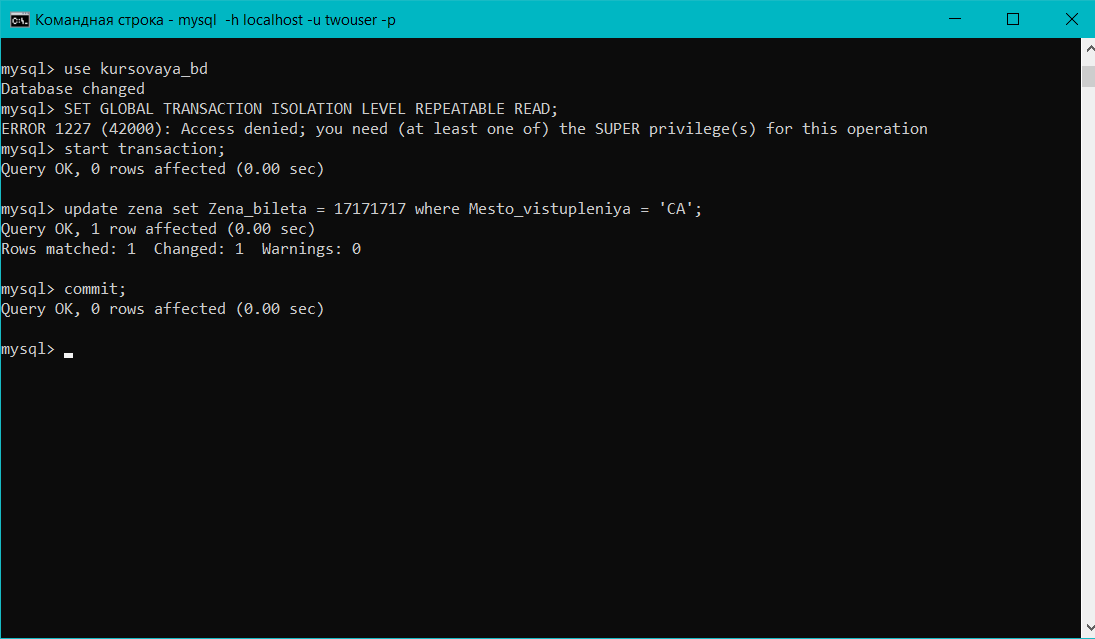


Рисунок 40 - Т2

- Фантомы

T1

use kursovaya\_bd

SET GLOBAL TRANSACTION ISOLATION LEVEL REPEATABLE READ;

select \* from zena where Zena\_bileta>10000;

T2

use kursovaya\_bd

SET GLOBAL TRANSACTION ISOLATION LEVEL REPEATABLE READ;

update zena set Zena\_bileta = 1000 where Mesto\_vistupleniya = 'CA';

commit;

T1

select \* from zena where Zena\_bileta>10000;

commit;

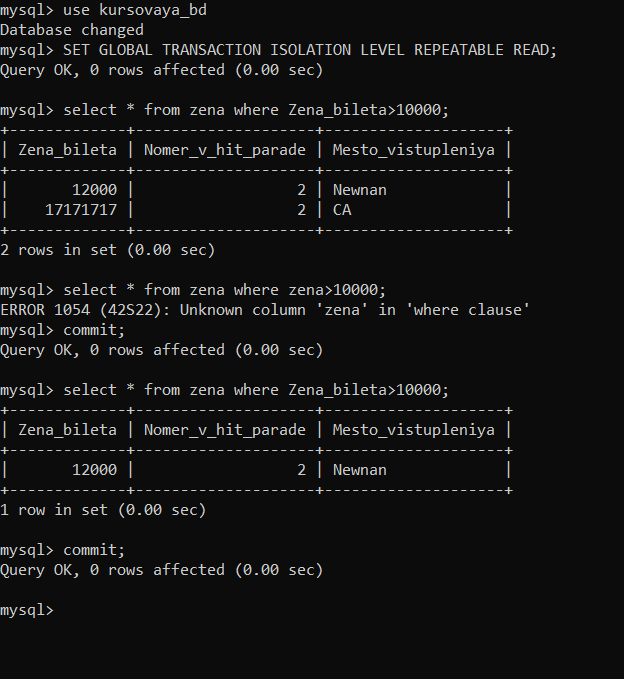


Рисунок 41 - Т1

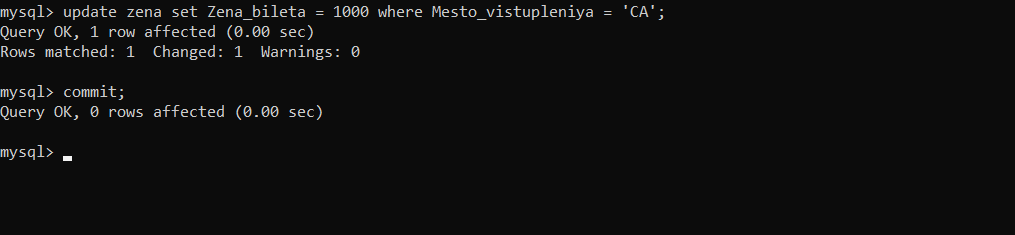


Рисунок 42 - Т2

Установить в обоих сеансах уровень изоляции SERIALIZABLE. Выполнить сценарии проверки наличия фантомов.

Уровень изоляции : SERIALIZABLE

- Фантомы

T1

use kursovaya\_bd

SET GLOBAL TRANSACTION ISOLATION LEVEL SERIALIZABLE;

select \* from zena where Zena\_bileta>10000;

T2

use kursovaya\_bd

SET GLOBAL TRANSACTION ISOLATION LEVEL SERIALIZABLE;

update zena set Zena\_bileta = 100000 where Mesto\_vistupleniya = 'CA';

update zena set Zena\_bileta = 100000 where Mesto\_vistupleniya = 'Eldorado';

update zena set Zena\_bileta = 100000 where Mesto\_vistupleniya = 'Atlanta';

commit;

T1

select \* from zena where Zena\_bileta>10000;

commit;

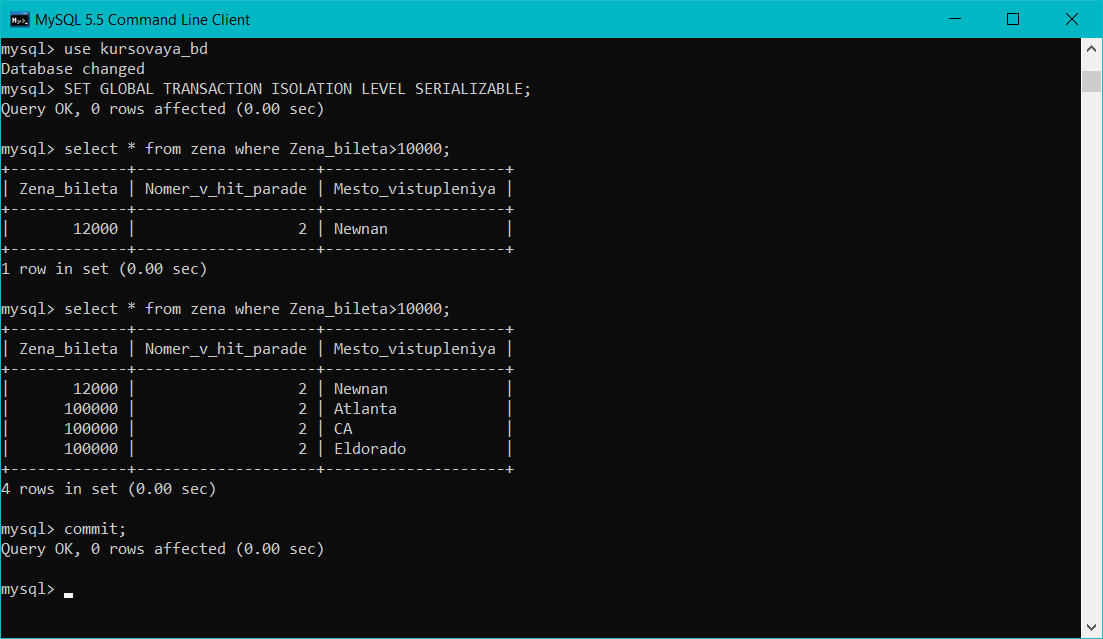


Рисунок 43 - Т1

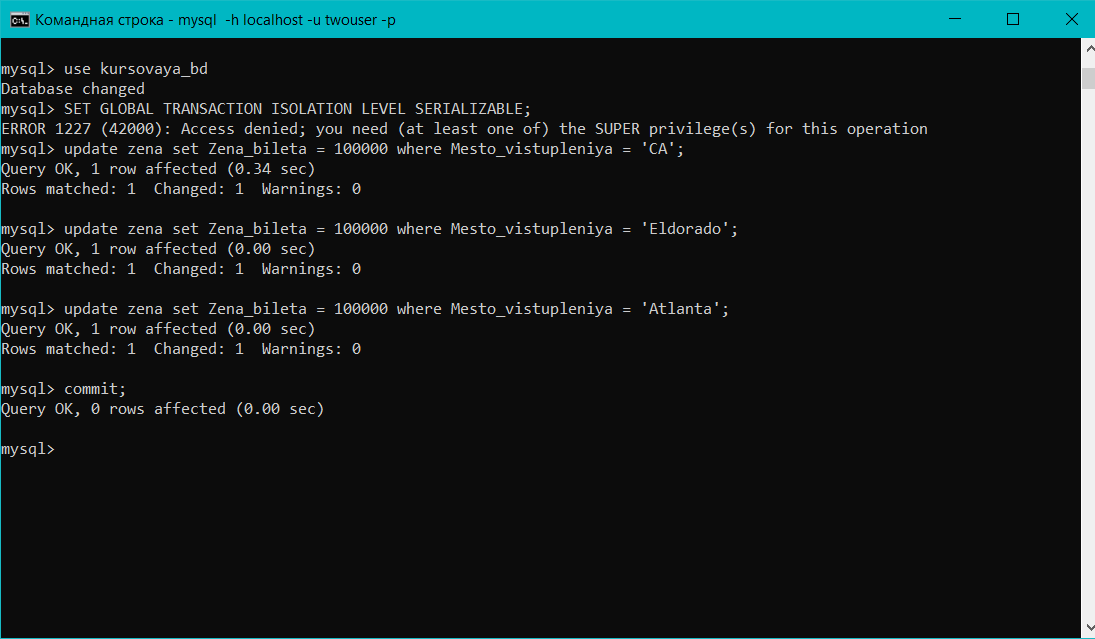
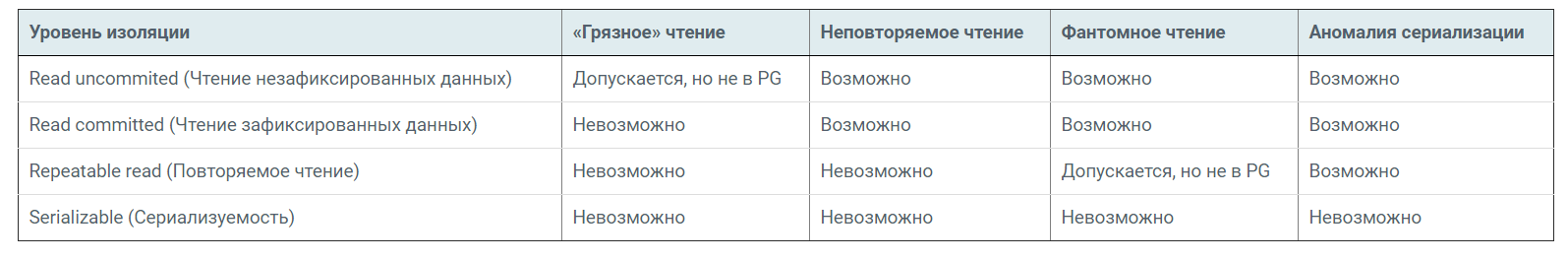


Рисунок 44 - Т2

**// PostgreSQL**

* Установить в обоих сеансах уровень изоляции READ UNCOMMITTED. Выполнить сценарии проверки наличия аномалий потерянных изменений и грязных чтений.
* Установить в обоих сеансах уровень изоляции READ COMMITTED. Выполнить сценарии проверки наличия аномалий грязных чтений и неповторяющихся чтений.
* Установить в обоих сеансах уровень изоляции REPEATABLE READ. Выполнить сценарии проверки наличия аномалий неповторяющихся чтений и фантомов.
* Установить в обоих сеансах уровень изоляции SERIALIZABLE. Выполнить сценарии проверки наличия фантомов.



Уровень изоляции : READ UNCOMMITTED

- Потерянные изменения (в PostgreSQL это Аномалия сериализации)

T1

\c kursovaya\_bd

BEGIN TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

update "Musikalnaya\_gruppa" set "God\_obrazovaniya" = 2000

where "Nazvanie\_gruppi" = 'tatu';

T2

\c kursovaya\_bd

BEGIN TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

update "Musikalnaya\_gruppa" set "God\_obrazovaniya" = 1920

where "Nazvanie\_gruppi" = 'tatu';

commit;

T1

commit;

select \* from "Musikalnaya\_gruppa";

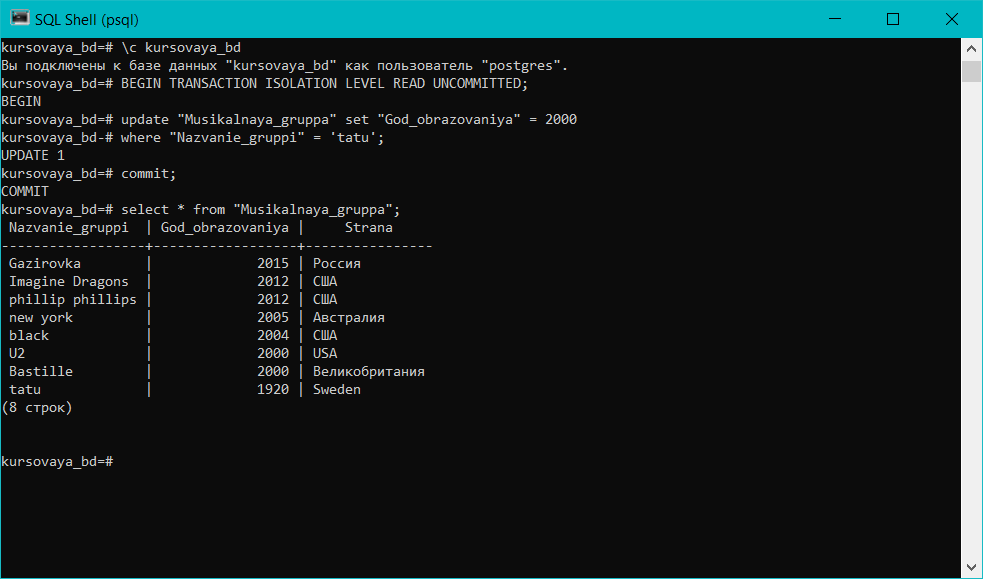


Рисунок 45 - Т1

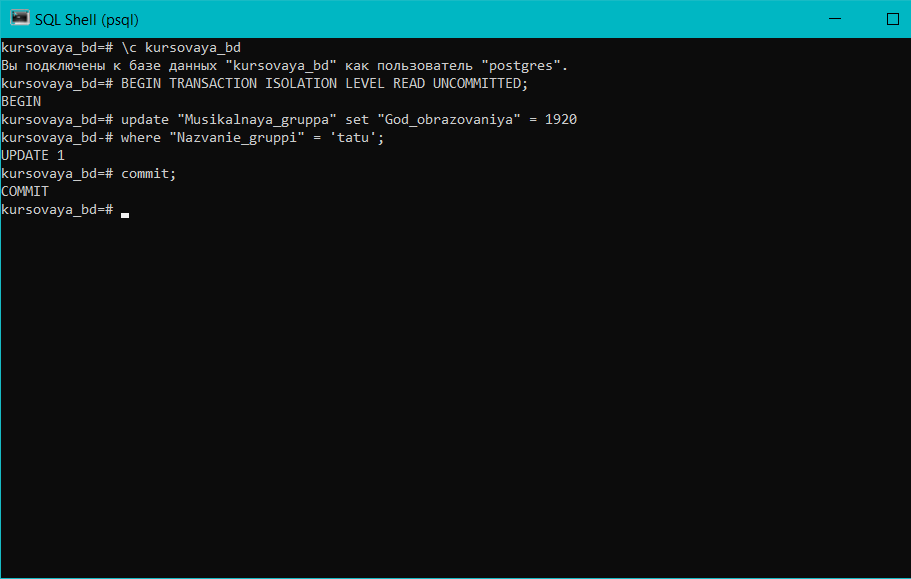


Рисунок 46 - Т2

- Грязное чтение (в PostgreSQL при уровне изоляции READ UNCOMMITTED НЕ допускается)

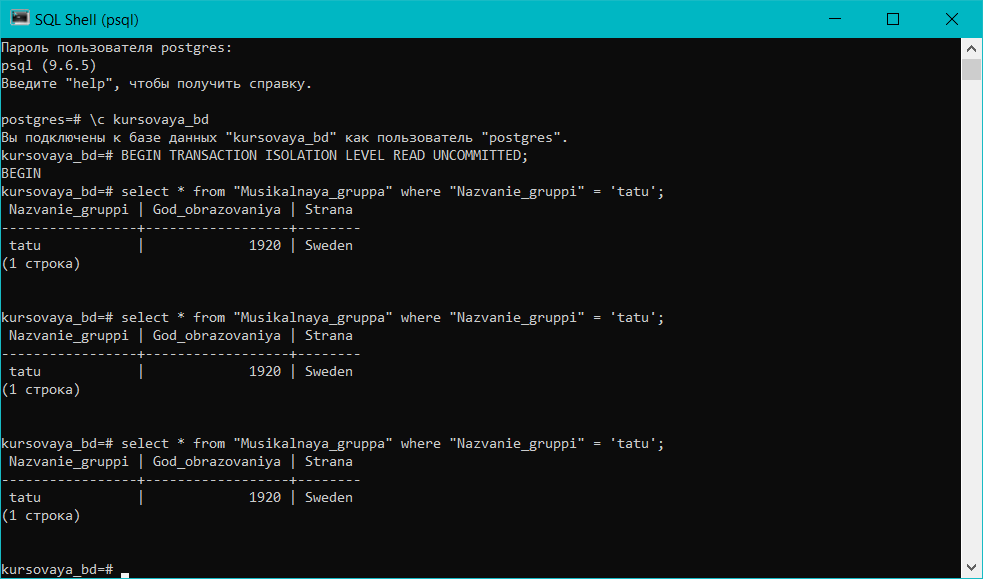


Рисунок 47 - Т1

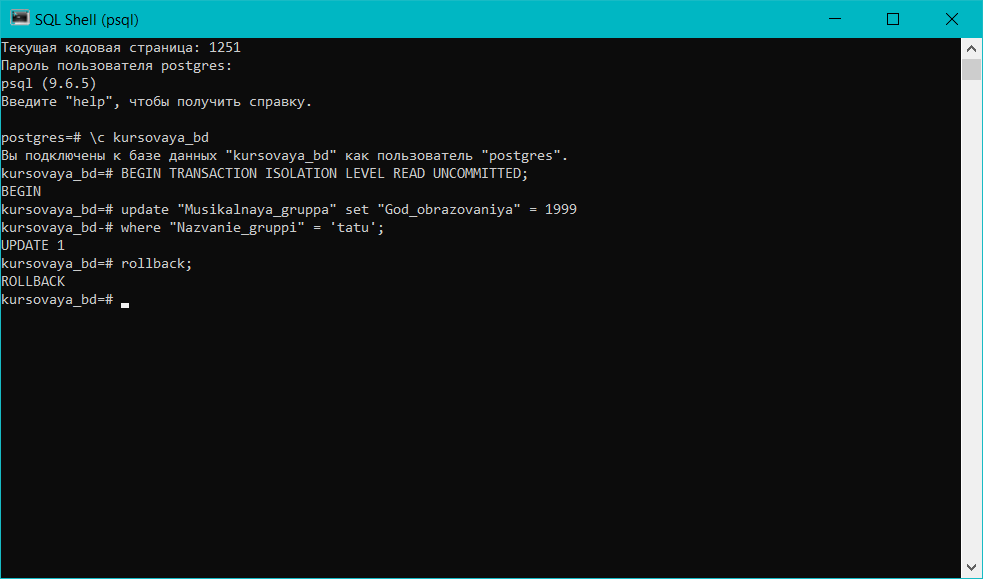


Рисунок 48 - Т2

Установить в обоих сеансах уровень изоляции READ COMMITTED. Выполнить сценарии проверки наличия аномалий грязных чтений и неповторяющихся чтений.

Уровень изоляции : READ COMMITTED

- Грязное чтение (в PostgreSQL при уровне изоляции READ COMMITTED НЕ допускается)

T1

\c kursovaya\_bd

BEGIN TRANSACTION ISOLATION LEVEL READ COMMITTED;

select \* from "Musikalnaya\_gruppa" where "Nazvanie\_gruppi" = 'tatu';

T2

\c kursovaya\_bd

BEGIN TRANSACTION ISOLATION LEVEL READ COMMITTED;

update "Musikalnaya\_gruppa" set "God\_obrazovaniya" = 1999

where "Nazvanie\_gruppi" = 'tatu';

T1

select \* from "Musikalnaya\_gruppa" where "Nazvanie\_gruppi" = 'tatu';

T2

rollback;

T1

select \* from "Musikalnaya\_gruppa" where "Nazvanie\_gruppi" = 'tatu';

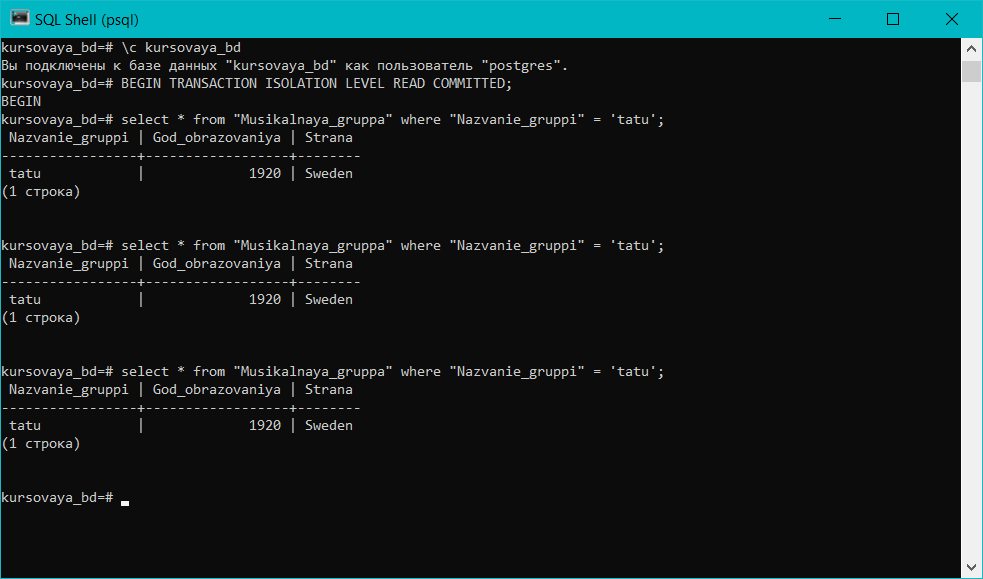


Рисунок 49 - Т1

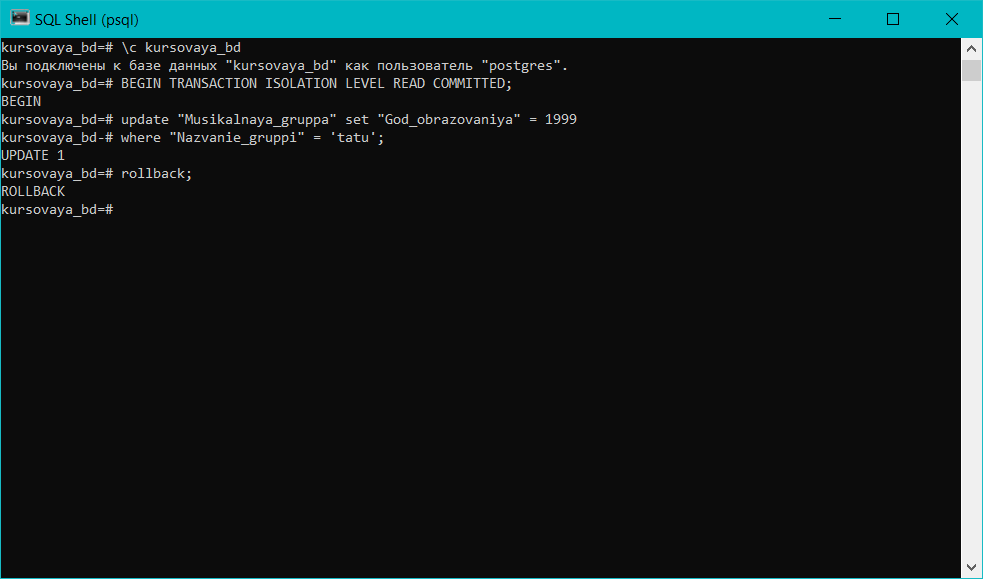


Рисунок 50 - Т2

- Неповторяющиеся чтения

T1

\c kursovaya\_bd

BEGIN TRANSACTION ISOLATION LEVEL READ COMMITTED;

select \* from "Musikalnaya\_gruppa" where "Nazvanie\_gruppi" = 'tatu';

T2

\c kursovaya\_bd

BEGIN TRANSACTION ISOLATION LEVEL READ COMMITTED;

update "Musikalnaya\_gruppa" set "God\_obrazovaniya" = 1999

where "Nazvanie\_gruppi" = 'tatu';

commit;

T1

select \* from "Musikalnaya\_gruppa" where "Nazvanie\_gruppi" = 'tatu';

commit;

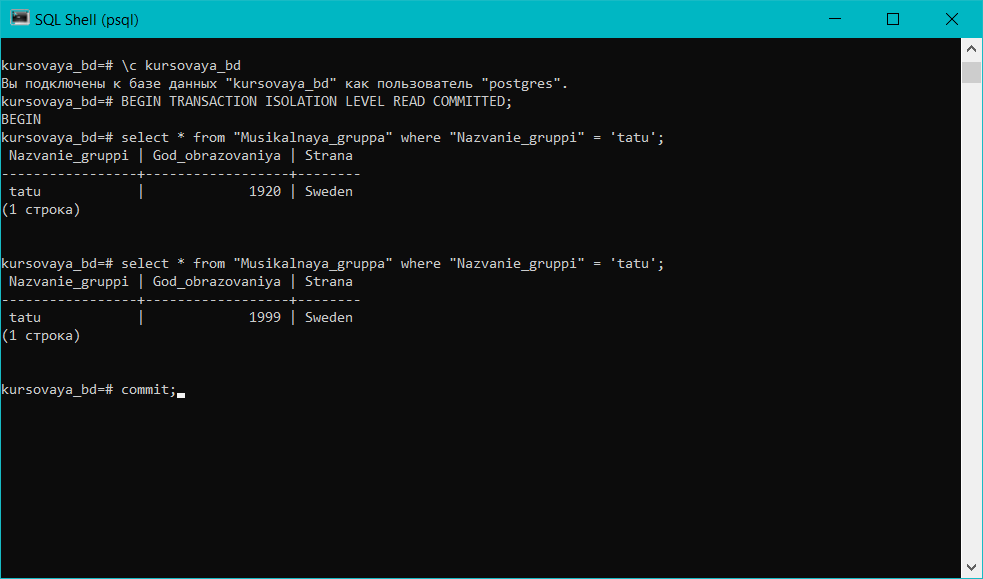


Рисунок 51 - Т1

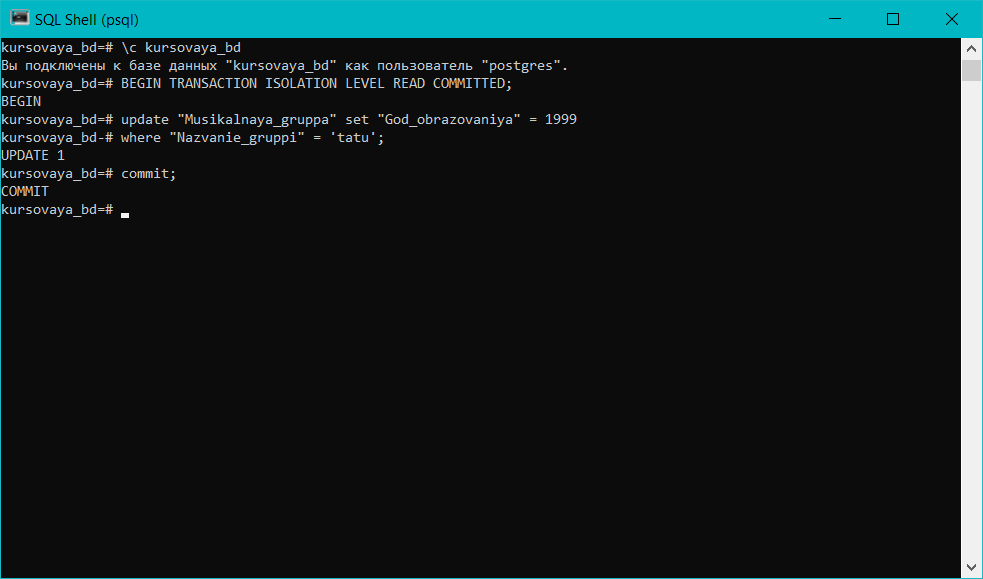


Рисунок 52 - Т2

* Установить в обоих сеансах уровень изоляции REPEATABLE READ. Выполнить сценарии проверки наличия аномалий неповторяющихся чтений и фантомов.

Уровень изоляции : REPEATABLE READ (в PostgreSQL при уровне изоляции REPEATABLE READ НЕ допускается)

- неповторяющиеся чтения

T1

\c kursovaya\_bd

BEGIN TRANSACTION ISOLATION LEVEL REPEATABLE READ;

select \* from "Musikalnaya\_gruppa" where "Nazvanie\_gruppi" = 'tatu';

T2

\c kursovaya\_bd

BEGIN TRANSACTION ISOLATION LEVEL REPEATABLE READ;

update "Musikalnaya\_gruppa" set "God\_obrazovaniya" = 2000

where "Nazvanie\_gruppi" = 'tatu';

commit;

T1

select \* from "Musikalnaya\_gruppa" where "Nazvanie\_gruppi" = 'tatu';

commit;

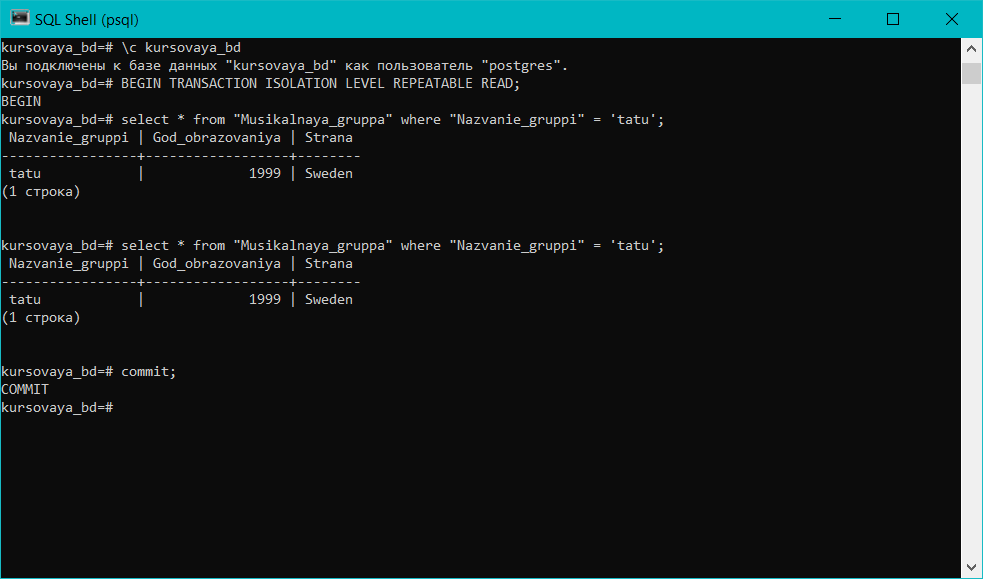


Рисунок 53 - Т1

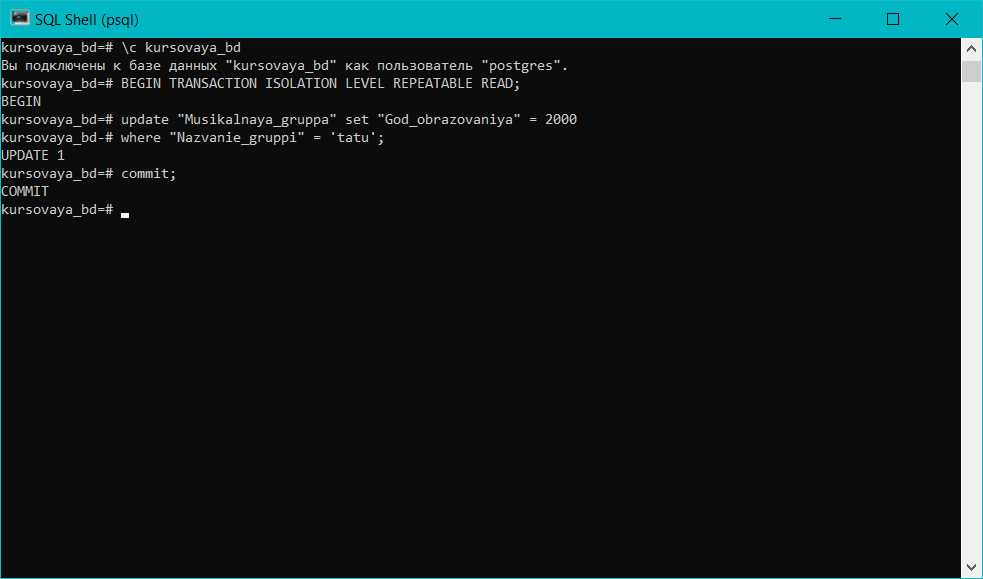


Рисунок 54 - Т2

- фантомы (в PostgreSQL НЕ допускаются)

T1

\c kursovaya\_bd

BEGIN TRANSACTION ISOLATION LEVEL REPEATABLE READ;

select \* from "Musikalnaya\_gruppa" where "God\_obrazovaniya" > 2010;

T2

\c kursovaya\_bd

BEGIN TRANSACTION ISOLATION LEVEL REPEATABLE READ;

update "Musikalnaya\_gruppa" set "God\_obrazovaniya" = 2018 where "Nazvanie\_gruppi" = 'black';

commit;

T1

select \* from "Musikalnaya\_gruppa" where "God\_obrazovaniya" > 2010;

commit;

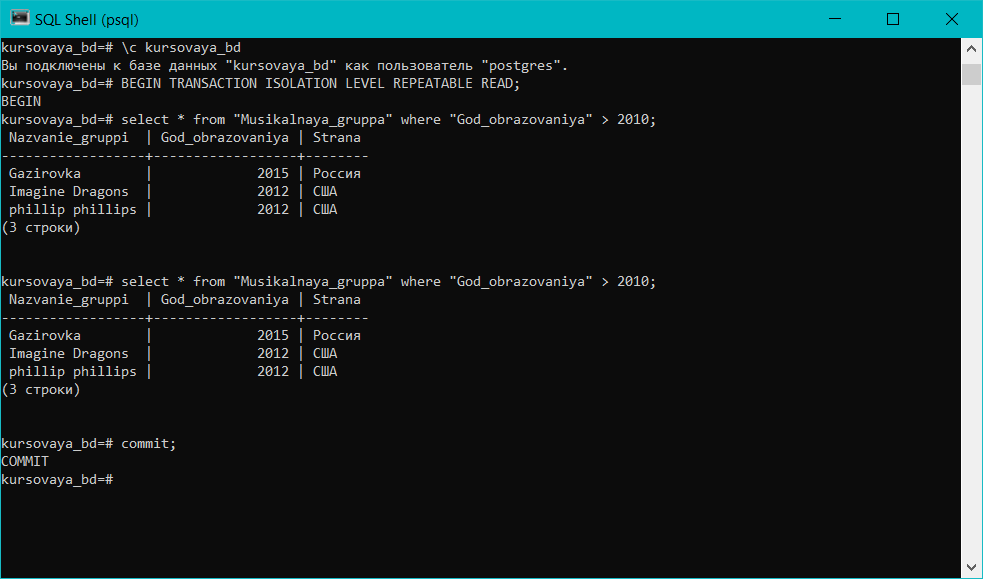


Рисунок 55 - Т1

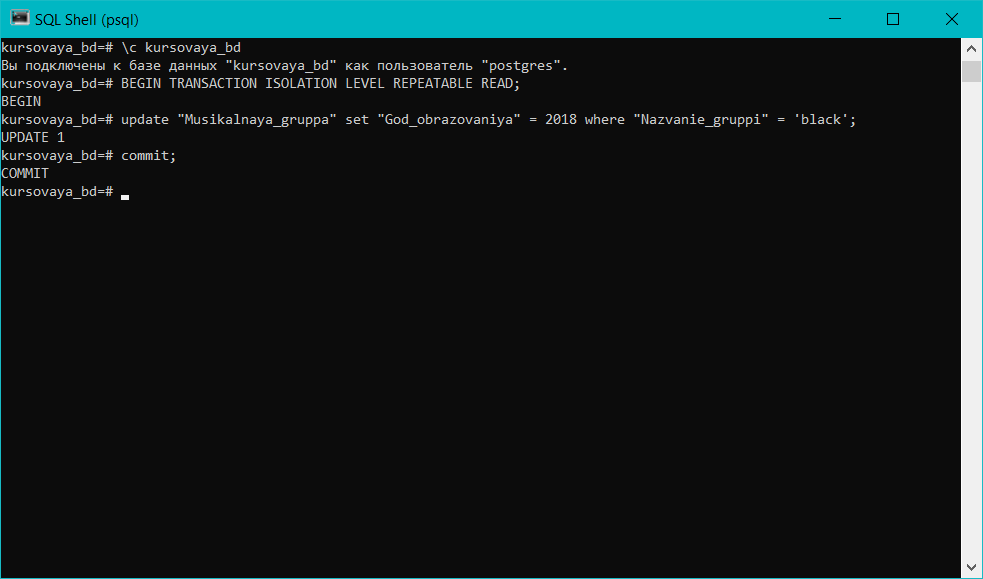


Рисунок 56 - Т2

* Установить в обоих сеансах уровень изоляции SERIALIZABLE. Выполнить сценарии проверки наличия фантомов.

Уровень изоляции : SERIALIZABLE

- Фантомы (в PostgreSQL это не реализуется)

T1

\c kursovaya\_bd

BEGIN TRANSACTION ISOLATION LEVEL SERIALIZABLE;

select \* from "Musikalnaya\_gruppa" where "God\_obrazovaniya" > 2010;

T2

\c kursovaya\_bd

BEGIN TRANSACTION ISOLATION LEVEL SERIALIZABLE;

update "Musikalnaya\_gruppa" set "God\_obrazovaniya" = 2018 where "Nazvanie\_gruppi" = 'tatu';

commit;

T1

select \* from "Musikalnaya\_gruppa" where "God\_obrazovaniya" > 2010;

commit;

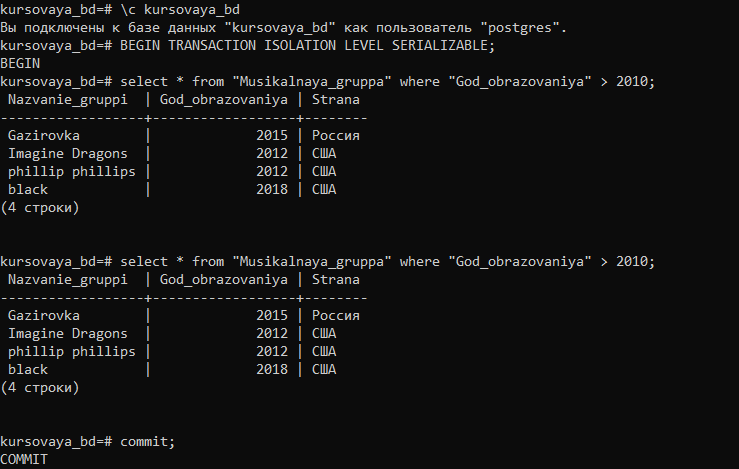


Рисунок 57 - Т1

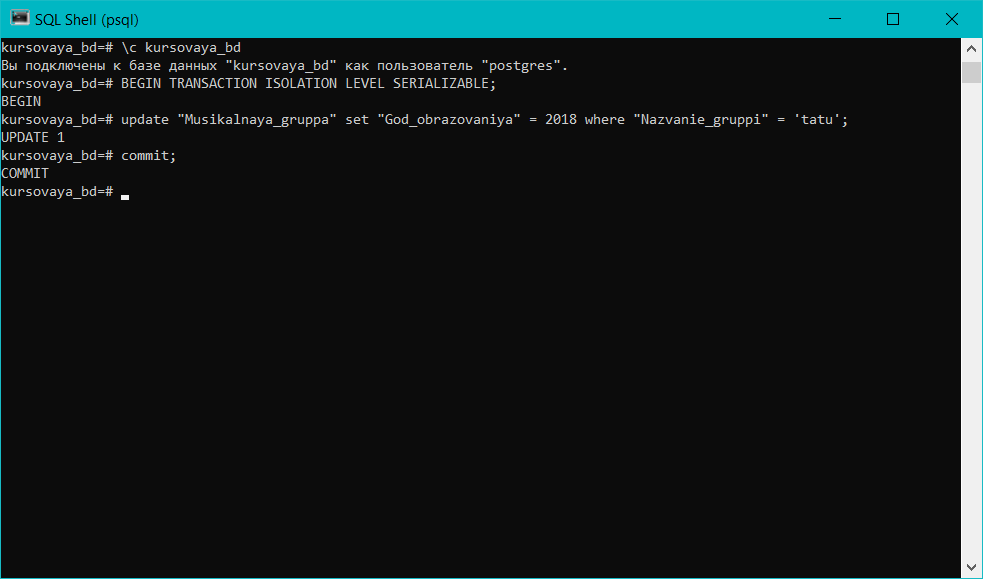


Рисунок 58 - Т2

**// Тупик в Mysql**

Уровень изоляции : SERIALIZABLE

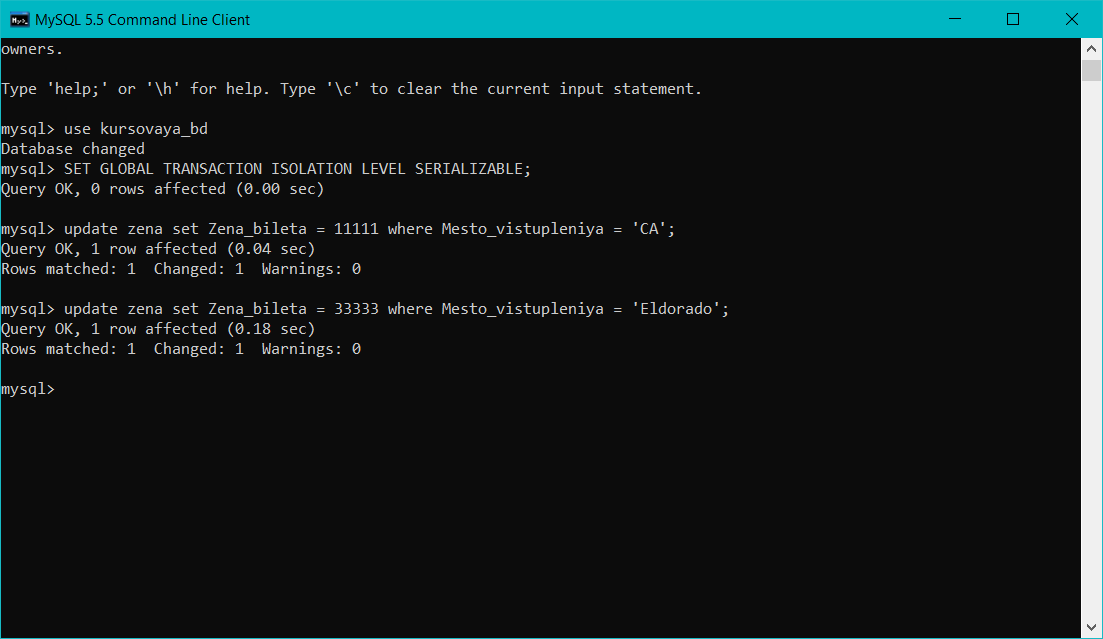


Рисунок 59 - Т1

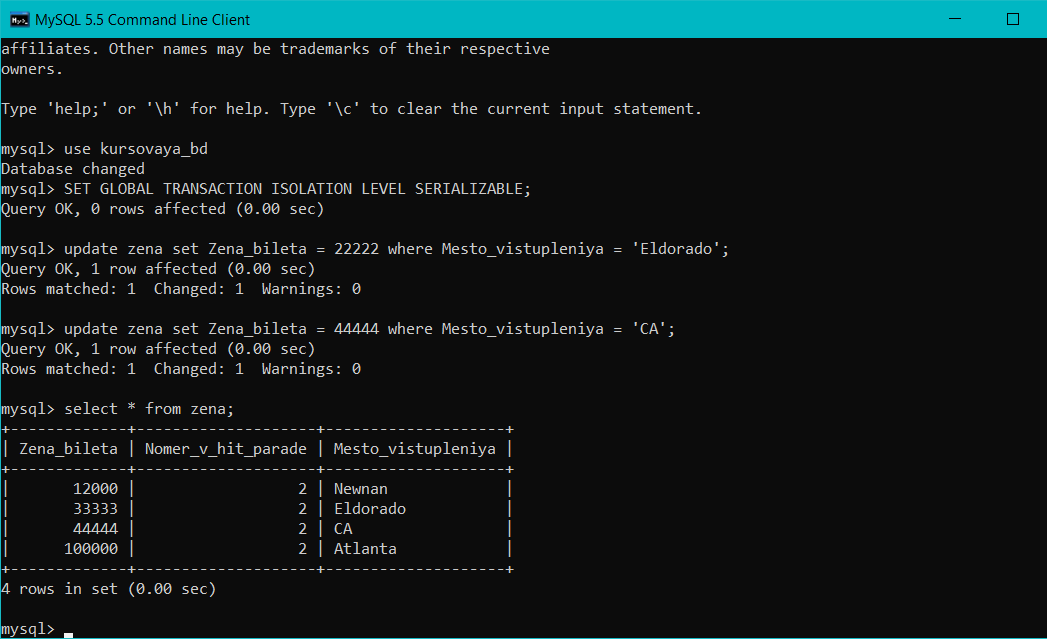


Рисунок 60 - Т2