|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.03 Прикладная информатика**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 2 |

**Название:**

Создание БД для приложения

**Дисциплина:** Базы данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-35 Б |  |  | Дулина И.А. |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | Кудрявцев А.П. |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2023

**Вариант 8**

**Цель:**

Данная лабораторная работа призвана сформировать у студента понимание особенностей хранения данных приложения в РСУБД, а также настройка и поддержка хранения данных.

**Задачи:**

• Получить теоретические знания по концептуальным картам.

• Ознакомится с понятием нормализации в БД.

• Изучить типы связей между сущностями и таблицами БД.

• Ознакомится с операторами создания БД.

• Изучение типов данных.

• Научится добавлять записи в таблицы.

• Научиться удалять и изменять записи в таблице.

• Ознакомиться с механизмами контроля согласованности БД, транзакциями и триггерами.

**Практическое задание №1**

**Задание:**

К первоначальной модели предметной области необходимо добавить не менее 2-х дополнительных сущностей (таблиц), необходимых для более полного решения поставленной задачи. Необходимо создать концептуальную схему БД (в виде ER-диаграммы, содержащей таблицы и связи между ними, с уточнением типов полей, с описанием внешних и первичных ключей). При сдаче задания студент должен обосновать соответствие созданной схемы поставленной задаче.

Предметная область для практических заданий: Управление проектом

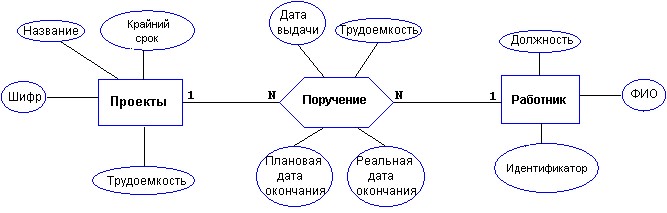


Рисунок 1.1 – предметная область



Рисунок 1.2 – ER-диаграмма

**Практическое задание №2**

**Задание:**

Необходимо определить первичные и внешние ключи, а также ограничения полей (возможность принимать неопределенное значение, уникальные ключи, проверочные ограничения и т. д.). Таблицы следует создавать в отдельной базе данных.

create table assignment

(assignment\_id integer not null generated always as identity,

project\_no integer not null,

worker\_id integer not null,

handout\_date date not null,

planned\_end\_date date not null,

real\_end\_date date not null,

assignment\_complexity double precision not null,

primary key (assignment\_id),

foreign key(project\_no)references project(project\_no),

foreign key(worker\_id)references worker(worker\_id));

Кроме того, нужно подготовить данные для заполнения созданных таблиц. Объем подготовленных данных должен составлять не менее 10 экземпляров для каждой из стержневых сущностей и 1000 экземпляров для целевой сущности. На основе этих данных необходимо создать SQL-скрипт для вставки соответствующих строк в таблицы БД. Python -> icreate -> MAIN!!

Необходимо подготовить два запроса:

* Запрос к одной таблице, содержащий фильтрацию по нескольким полям.
* Запрос к нескольким связанным таблицам, содержащий фильтрацию по нескольким полям.

Для каждого из этих запросов необходимо провести следующие шаги:

* Получить план выполнения запроса без использования индексов.
* Получить статистику (IO и Time) выполнения запроса без использования индексов.
* Создать нужные индексы, позволяющие ускорить запрос.
* Получить статистику выполнения запроса с использованием индексов и сравнить с первоначальной статистикой.

Первый запрос к таблице assignment, содержащий фильтрацию по нескольким полям:

select \* from assignment where worker\_id=2 and project\_no=1;

Второй запрос к таблицам assignment и worker, содержащий фильтрацию по нескольким полям:

select a.assignment\_id,

a.project\_no,

a.worker\_id,

worker.worker\_name,

worker.position,

a.assignment\_complexity

from assignment as a

inner join worker using (worker\_id) – on a.worker\_id=worker.worker\_id

where assignment\_complexity between '10' and '20'

and worker.position = 'дизайнер'

order by assignment\_complexity;

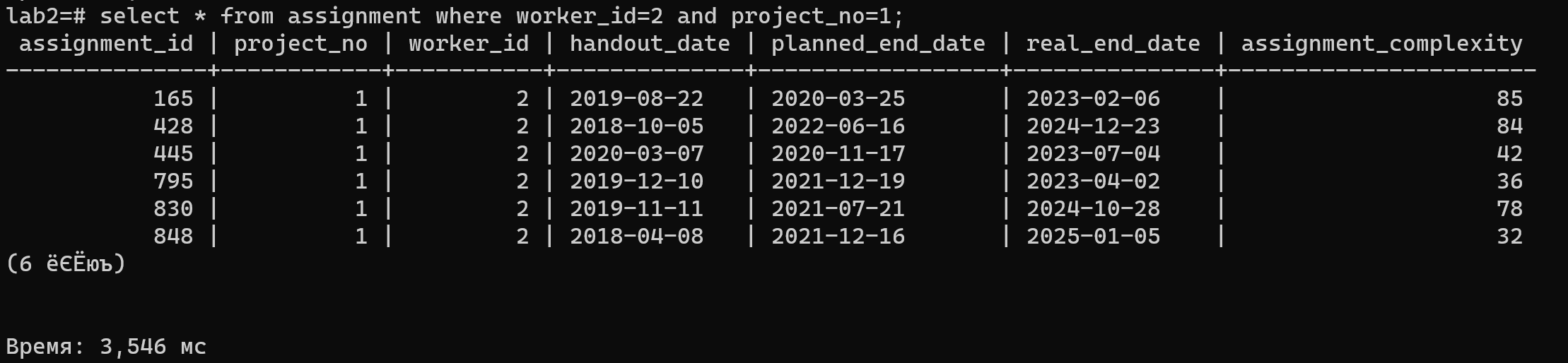


Рисунок 2.1 – время выполнения первого запроса без индексов

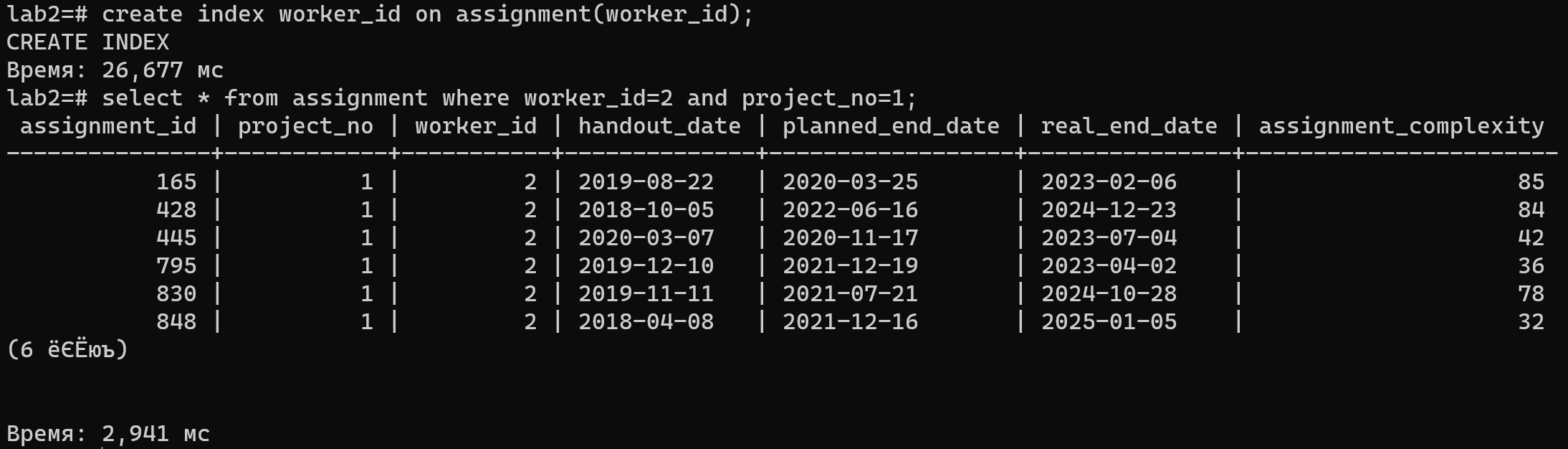


Рисунок 2.2 – время выполнения первого запроса с индексами

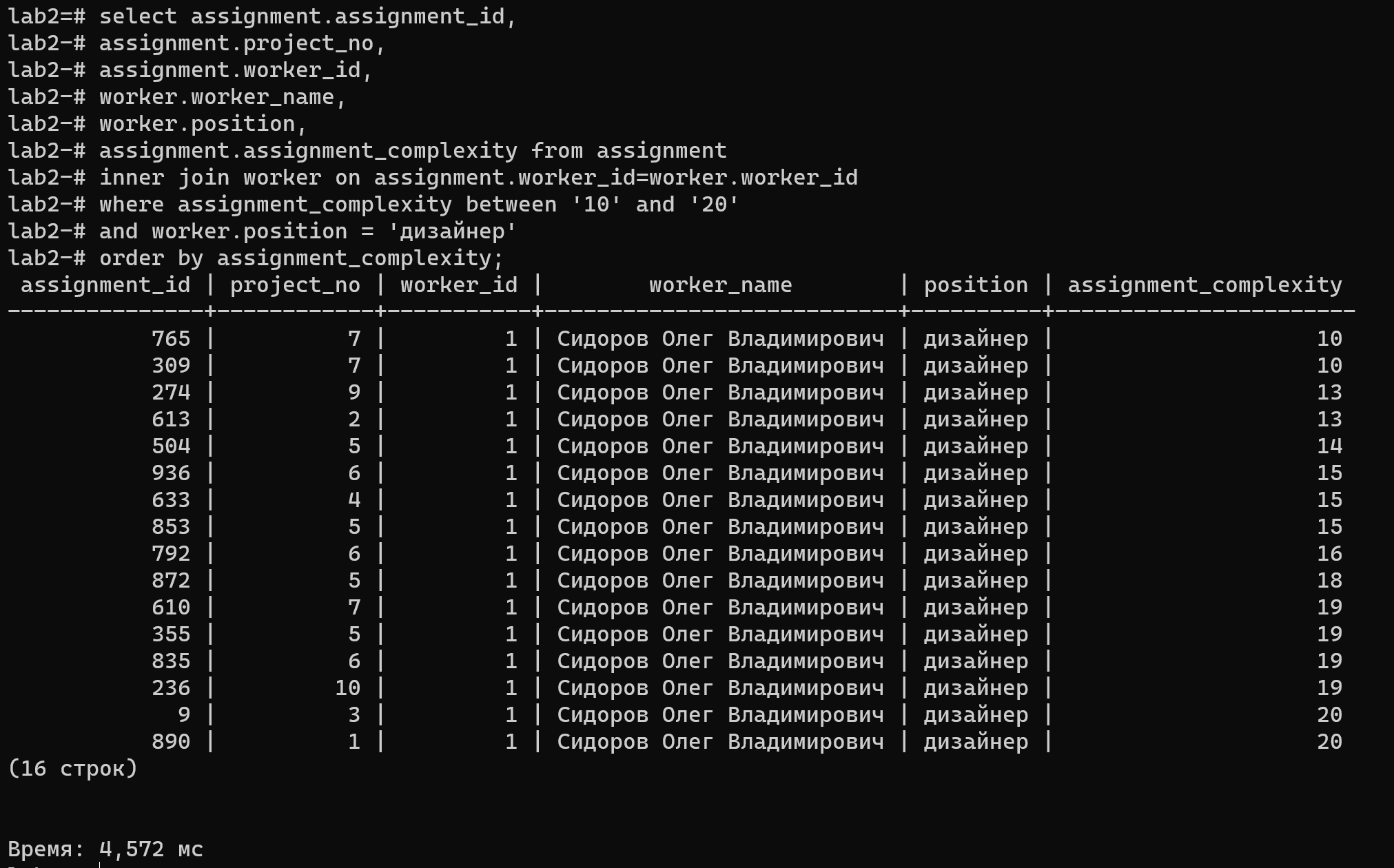


Рисунок 2.3 – время выполнения второго запроса без индексов

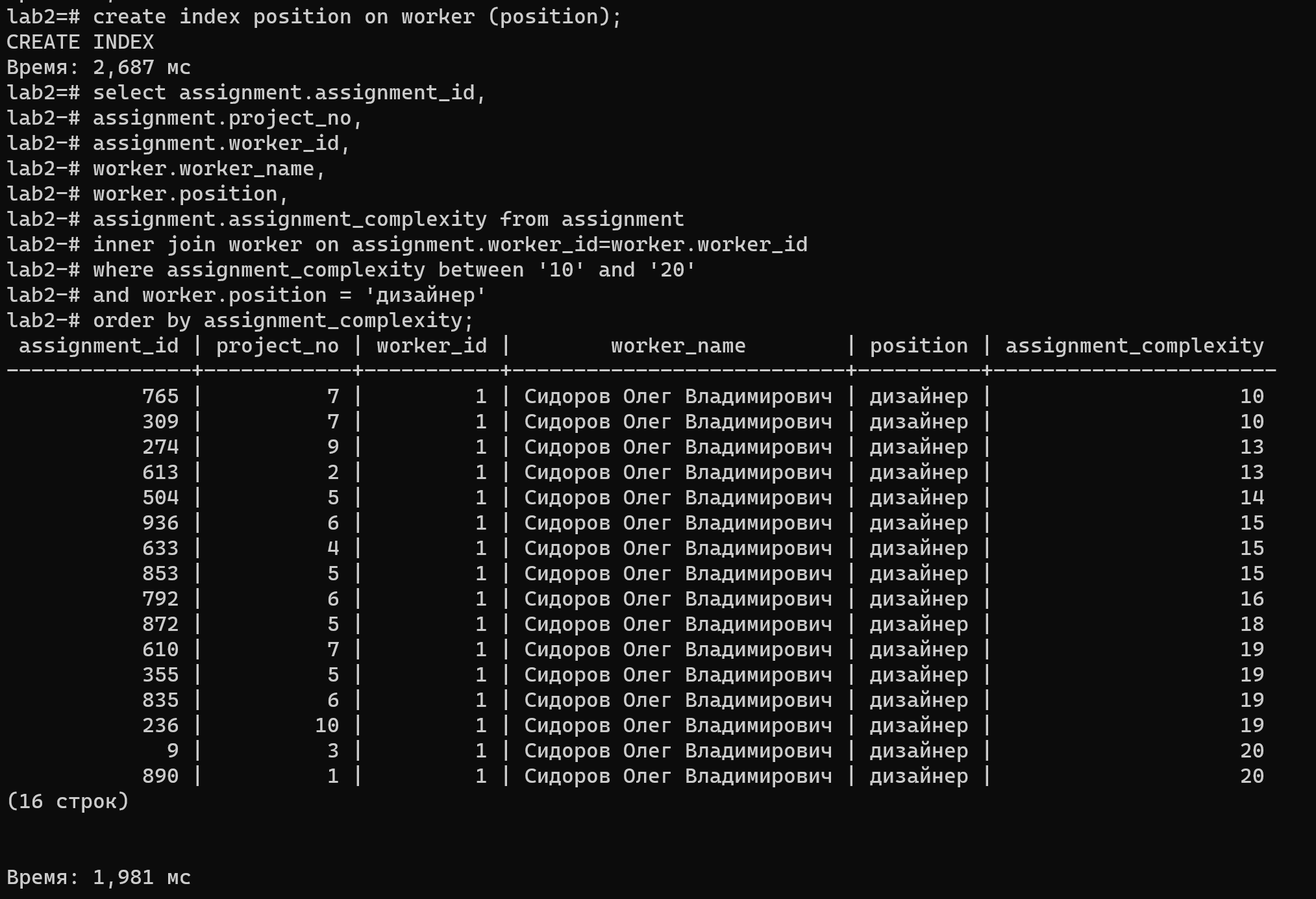


Рисунок 2.4 – время выполнения второго запроса с индексом

**Практическое задание №3**

**Задание:**

* Подготовить 3-4 выборки, которые имеют осмысленное значение для предметной области, и также составить для них SQL-скрипты.
* Сформулировать 3-4 запроса на изменение и удаление из базы данных. Запросы должны быть сформулированы в терминах предметной области. Среди запросов обязательно должны быть такие, которые будут вызывать срабатывание ограничений целостности. Составить SQL-скрипты для выполнения этих запросов.

**Выборка 1:**

select \* from project

where project\_complexity between '40' and '80'

order by deadline;

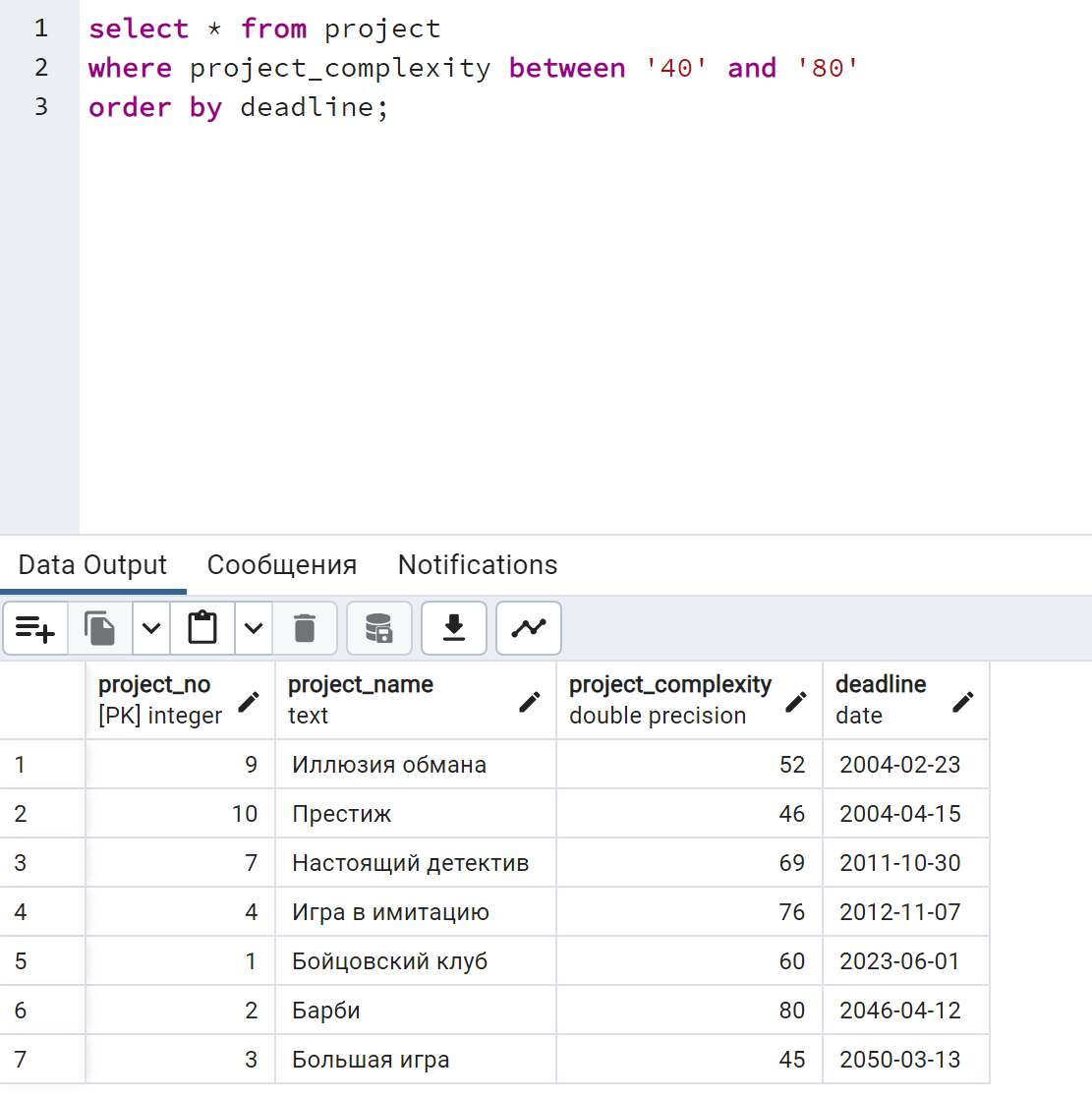


Рисунок 3.1 – результат выборки 1

**Выборка 2:**

select \* from assignment

where handout\_date between '2018-01-01' and '2019-01-01'

order by assignment\_complexity limit 100;

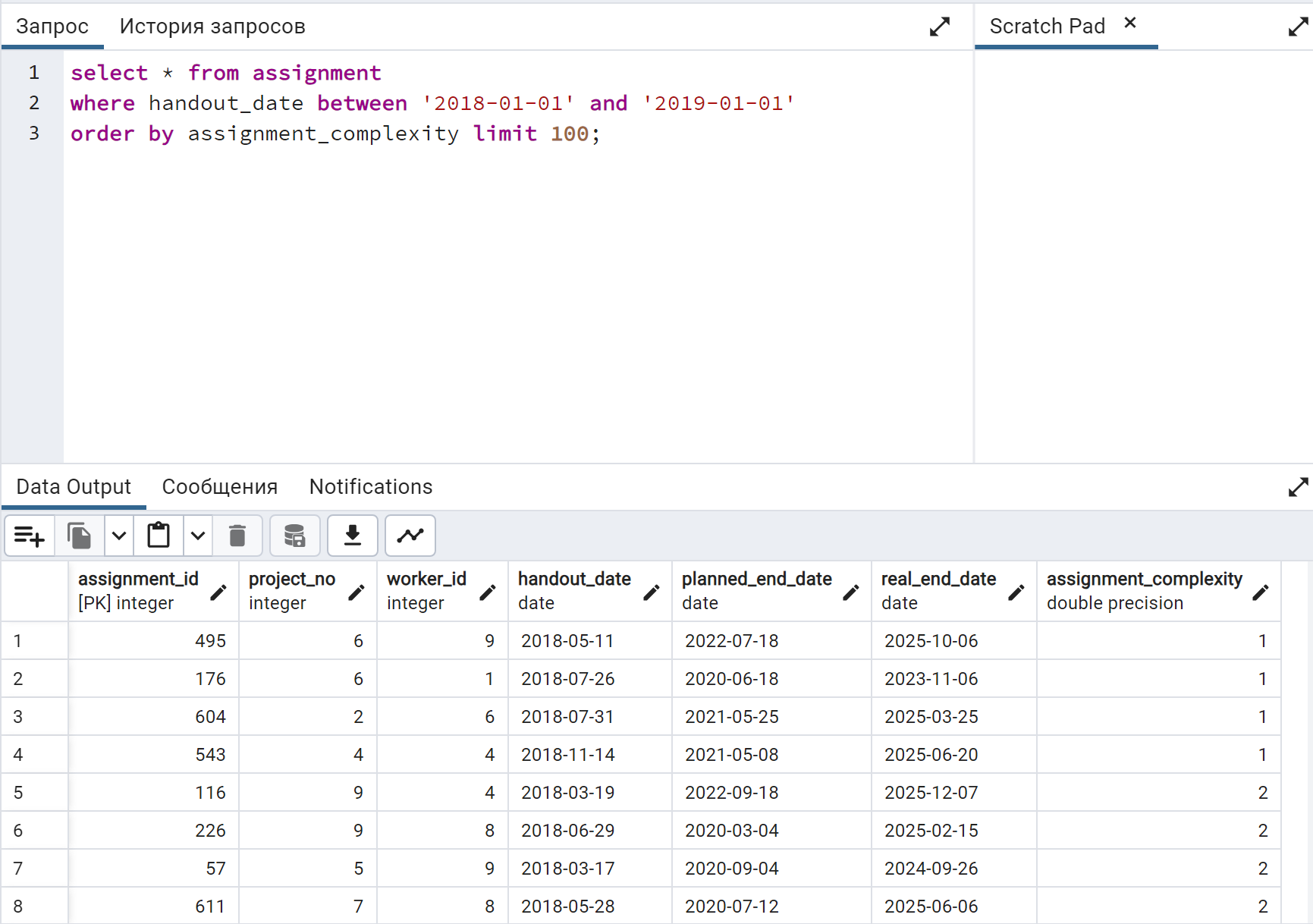


Рисунок 3.2 – результат выборки 2

**Выборка 3:**

select \* from assignment

where planned\_end\_date between '2020-01-01' and '2020-02-02'

and worker\_id=2

order by assignment\_complexity limit 100;

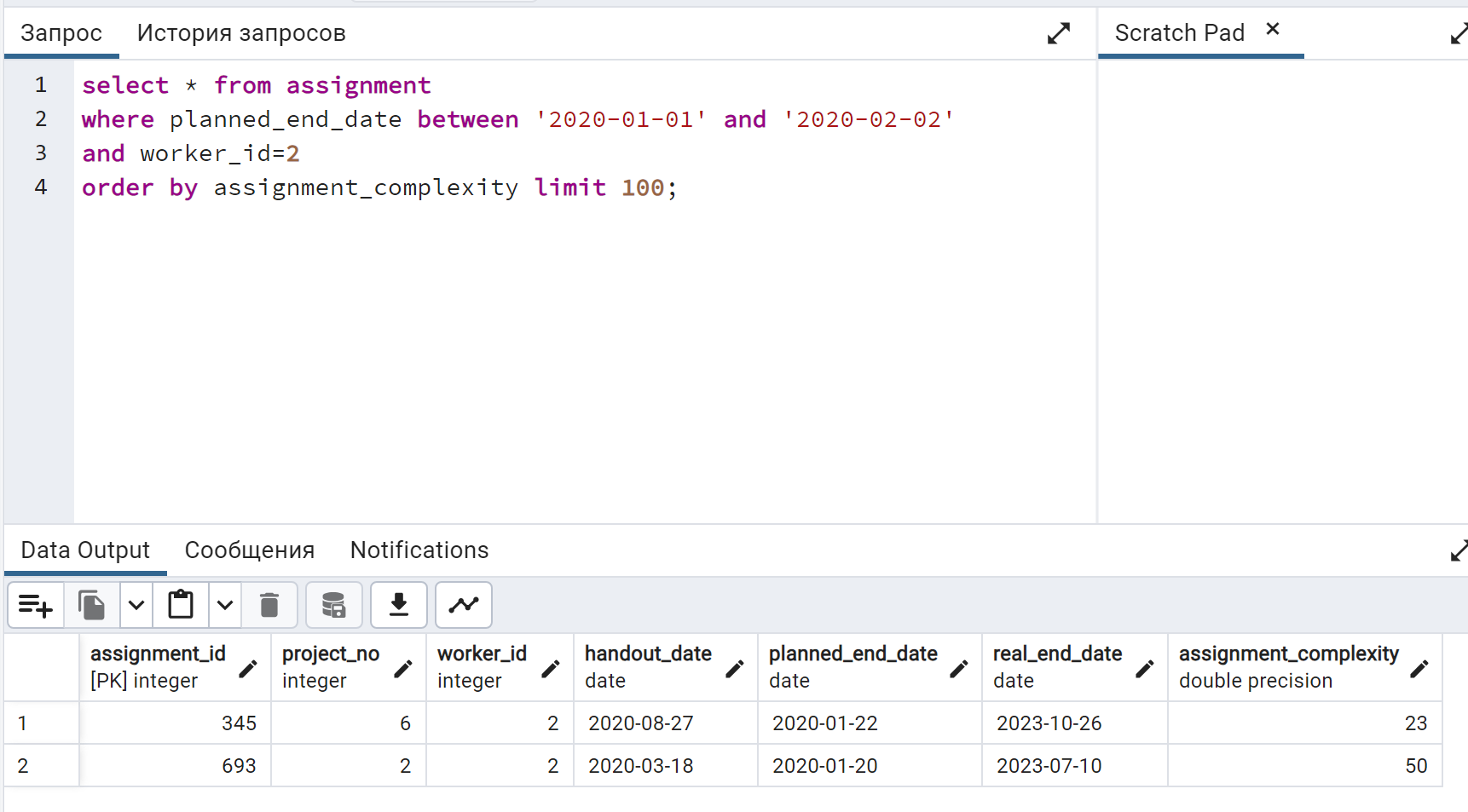


Рисунок 3.3 – результат выборки 3

**Запрос 1:**

insert into assignment

values

('1200', '11', '2', '2020-02-02', '2023-12-21', '2024-01-01', '30');



Рисунок 3.4 – результат запроса 1

*Исправленный запрос:*

insert into assignment

values

('1200', '10', '2', '2020-02-02', '2023-12-21', '2024-01-01', '30');

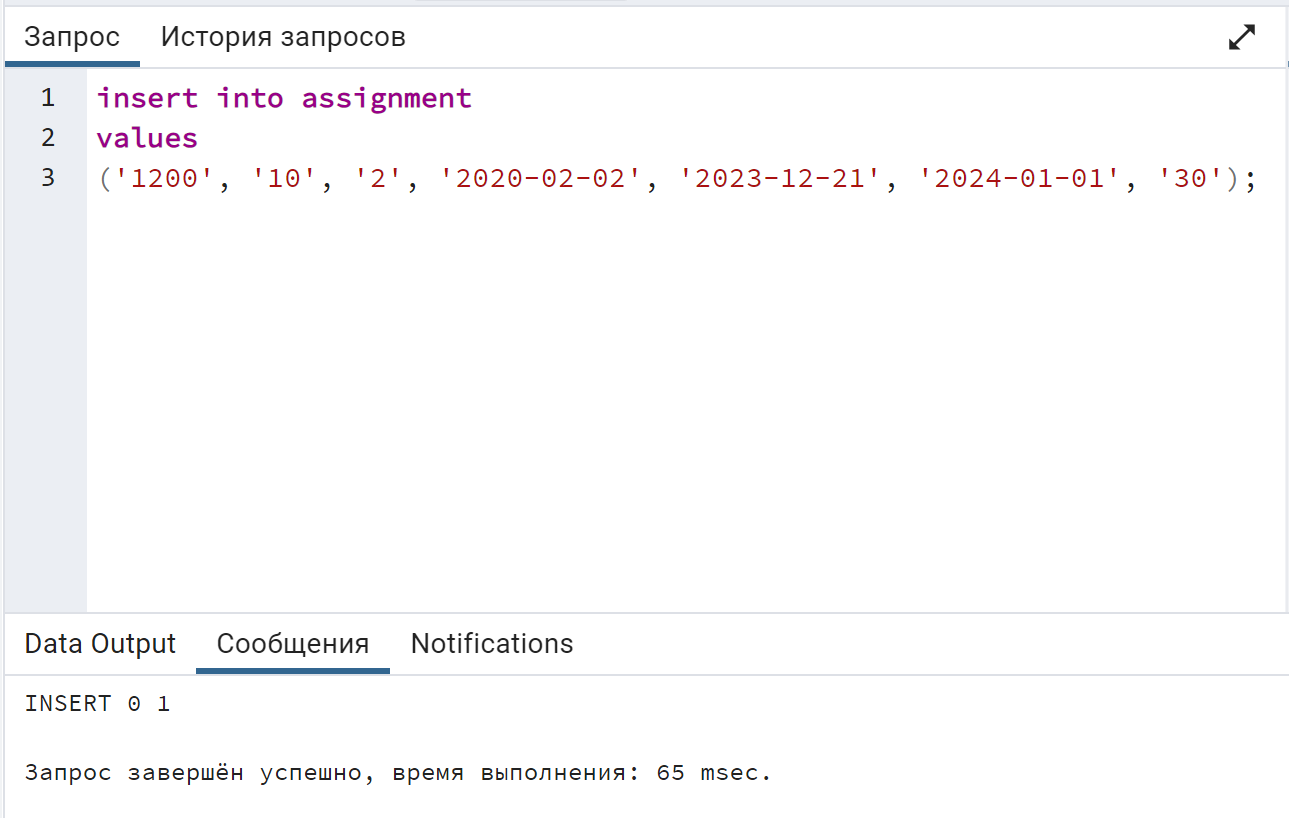


Рисунок 3.5 – результат исправленного запроса

**Запрос 2:**

DELETE

FROM assignment

WHERE assignment\_id = 1200;

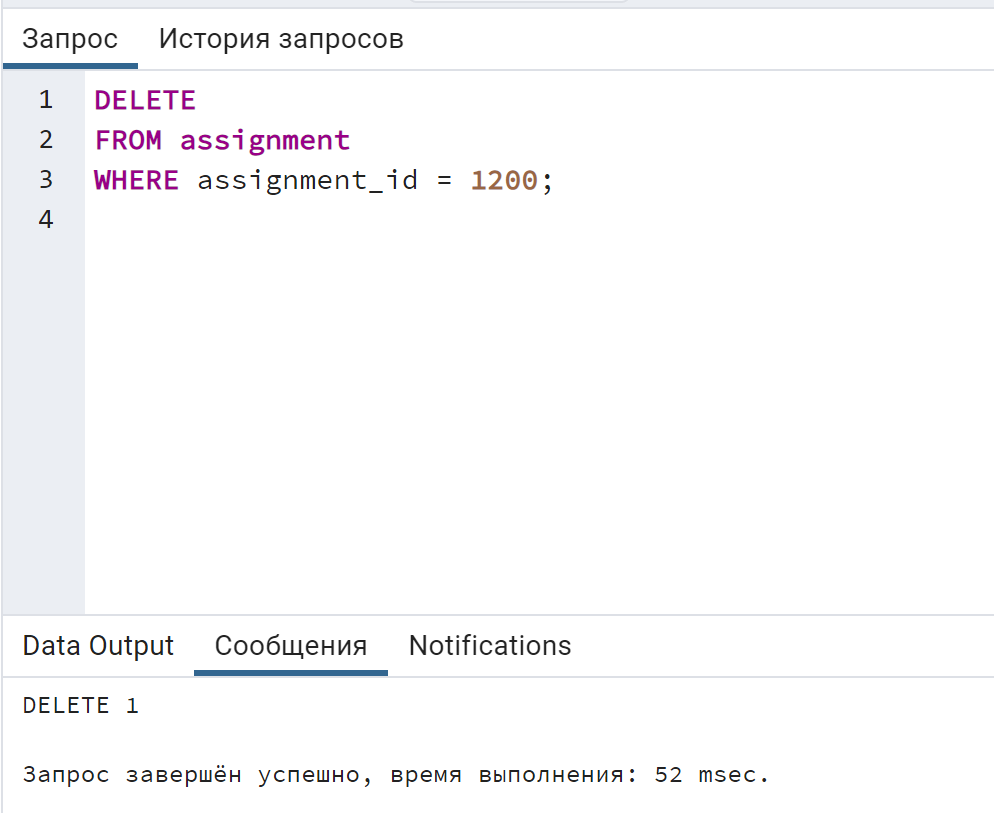


Рисунок 3.6 – результат запроса 2

**Запрос 3:**

update assignment

set assignment\_complexity=99

where worker\_id=2;

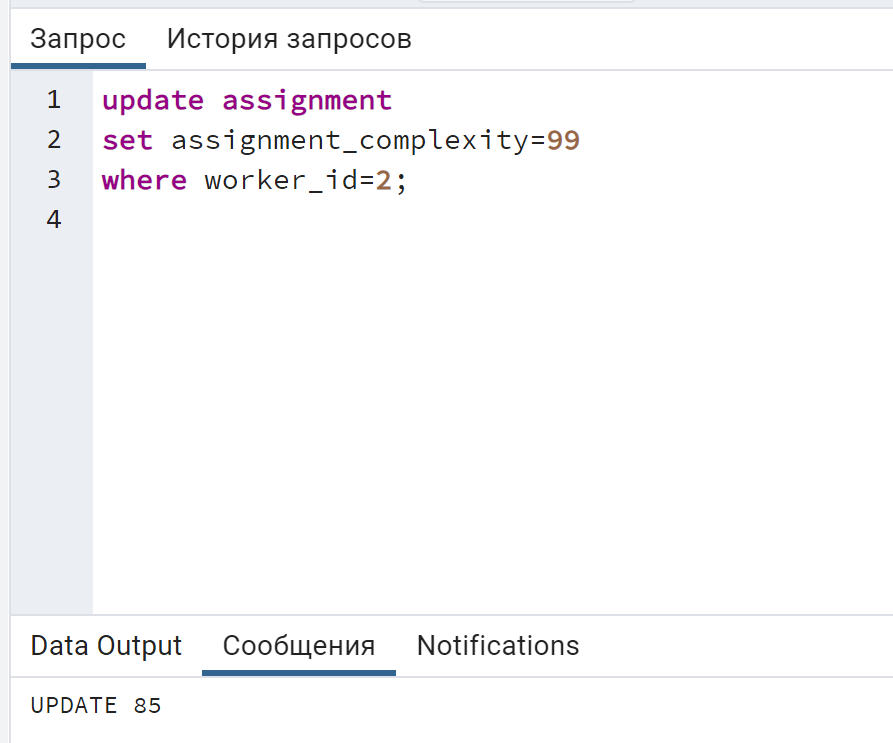


Рисунок 3.7 – результат запроса 3

**Практическое задание №4**

**Задание:**

* Составить SQL-скрипты для создания нескольких представлений, которые позволяли бы упростить манипуляции с данными или позволяли бы ограничить доступ к данным, предоставляя только необходимую информацию.
* Продемонстрировать изменение и вставку данных через представления.
* Продемонстрировать невозможность изменения данных через представление.
* Продемонстрировать полезность материализованного представления.

**Представление 1:**

*SQL-скрипт:*

CREATE OR REPLACE VIEW count\_worker\_and\_project as

SELECT project\_no, worker\_id, count(\*)

FROM assignment

GROUP BY worker\_id , project\_no

ORDER BY worker\_id, project\_no;

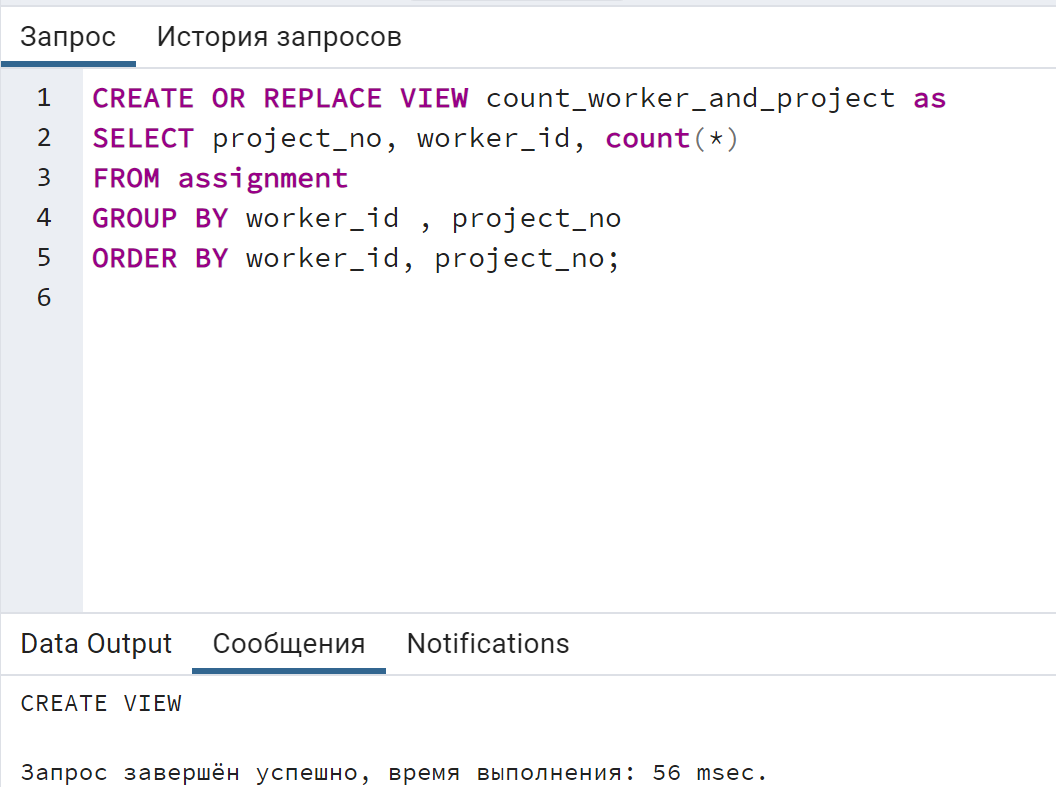


Рисунок 4.1 – результат выполнения

*Невозможность изменения данных через представление:*

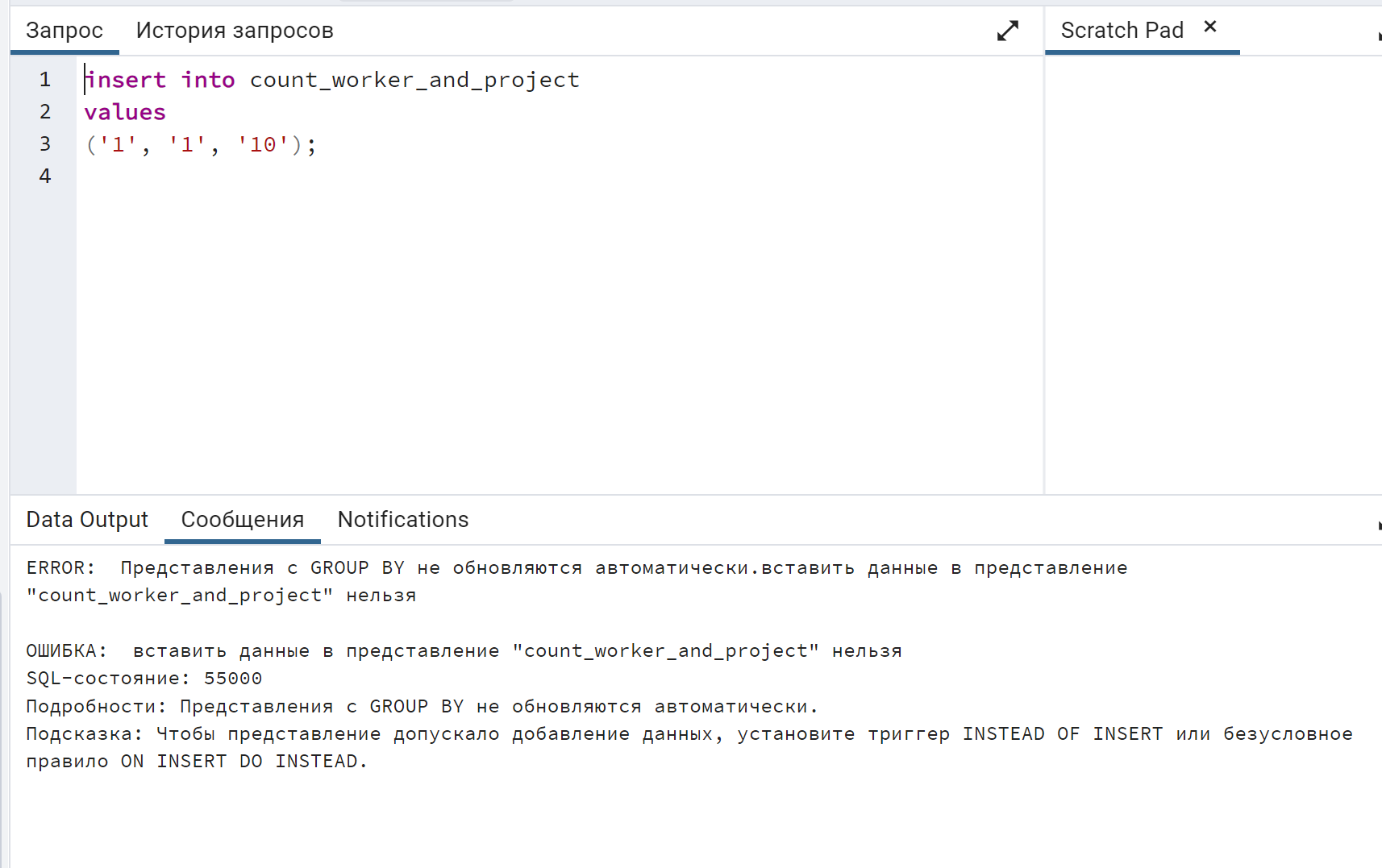


Рисунок 4.2 – результат выполнения

**Представление 2:**

*SQL-скрипт:*

drop view if exists tr;

create or replace view vtoroe as

select \* from assignment where handout\_date>'2019-09-09';

*Изменение и вставка данных через представление:*

update vtoroe

set handout\_date='2020-12-12' where worker\_id='2';

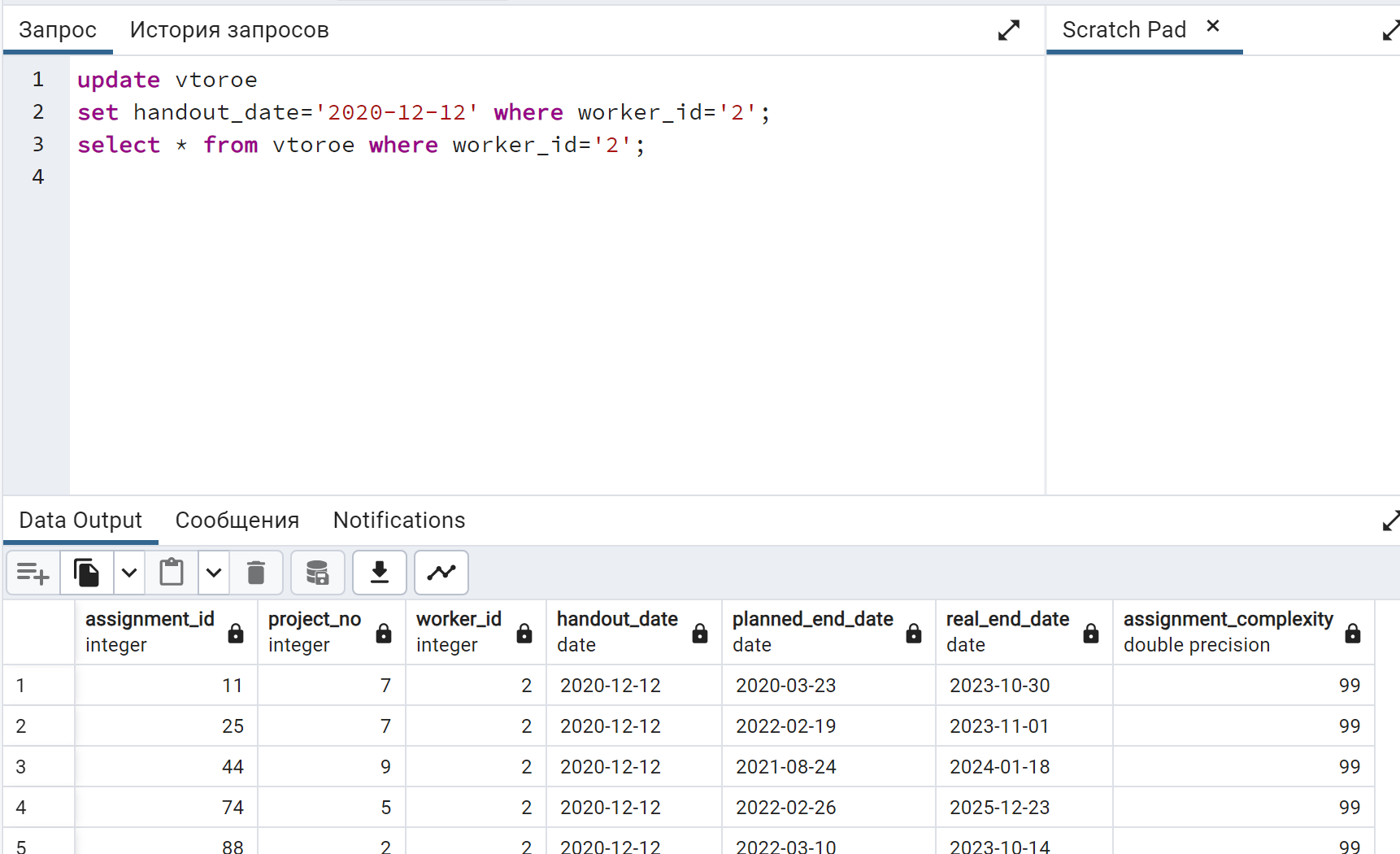


Рисунок 4.3 – измененное представление

*Вставка данных через представление:*

insert into vtoroe

values

('1200', '3', '2', '2020-02-02', '2024-12-12', '2025-05-05', '60')

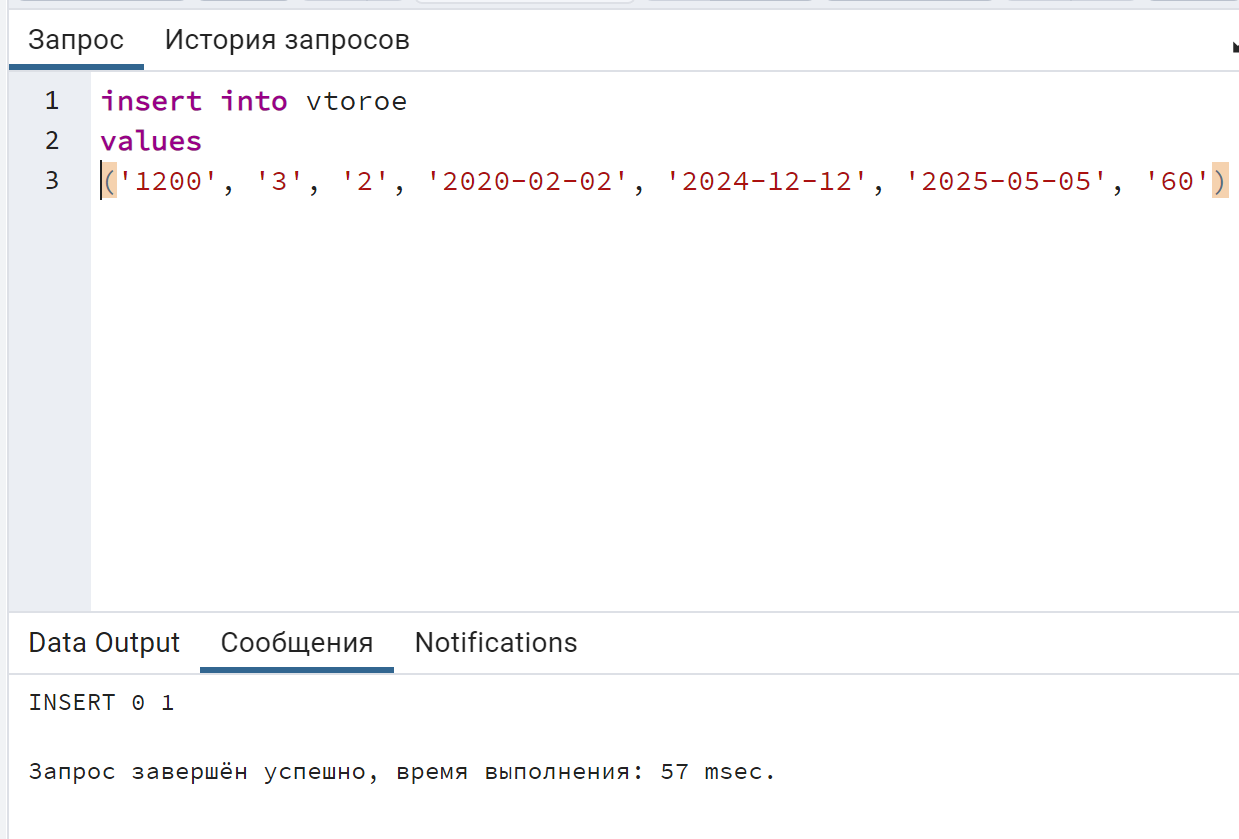


Рисунок 4.4 – результат выполнения

**Практическое задание №5**

**Задание:**

Необходимо подготовить SQL-скрипты для проверки наличия аномалий (потерянных изменений, грязных чтений, неповторяющихся чтений, фантомов) при параллельном исполнении транзакций на различных уровнях изолированности SQL/92 (READ UNCOMMITTED, READ COMMITTED, REPEATABLE READ, SERIALIZABLE). Подготовленные скрипты должны работать с одной из таблиц, созданных в практическом задании №2.1. 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.

Уровень изоляции **READ UNCOMMITED**

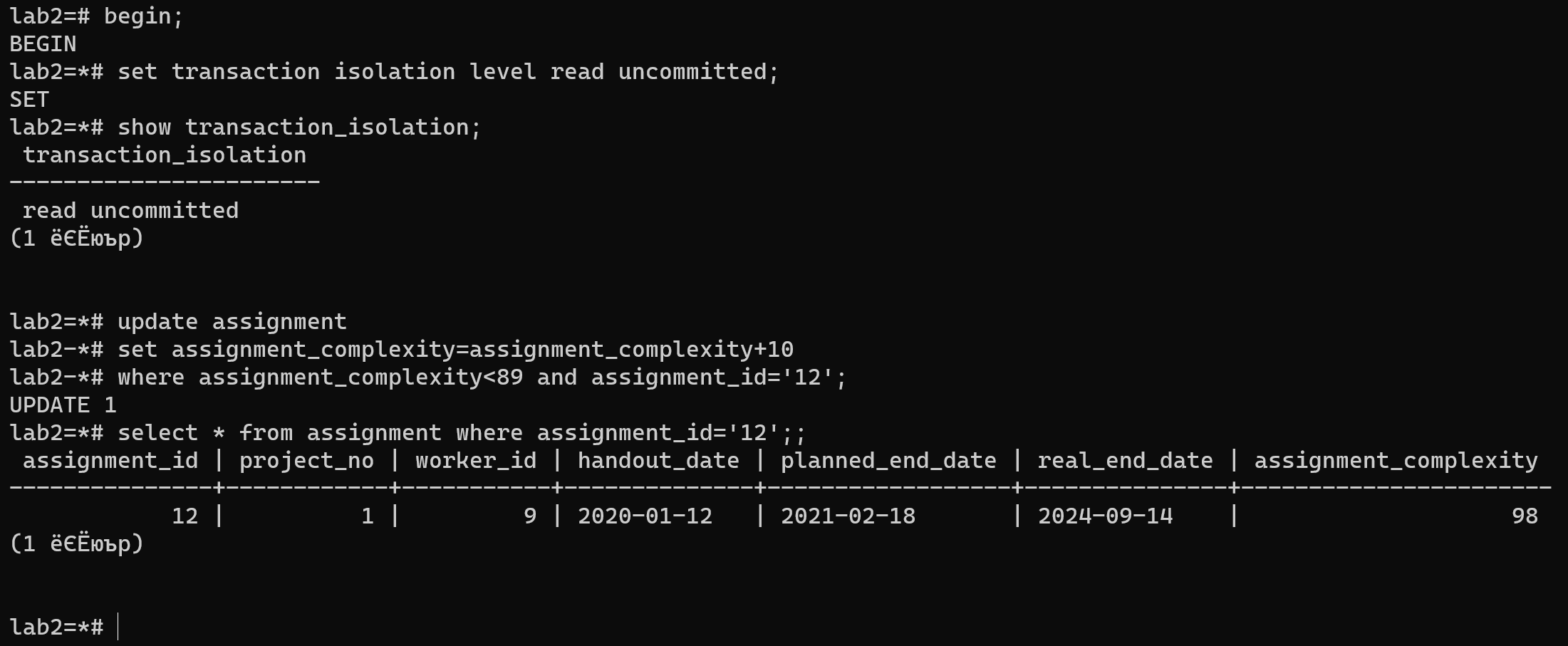


Рисунок 5.1 – Транзакция 1 (Терминал №1)

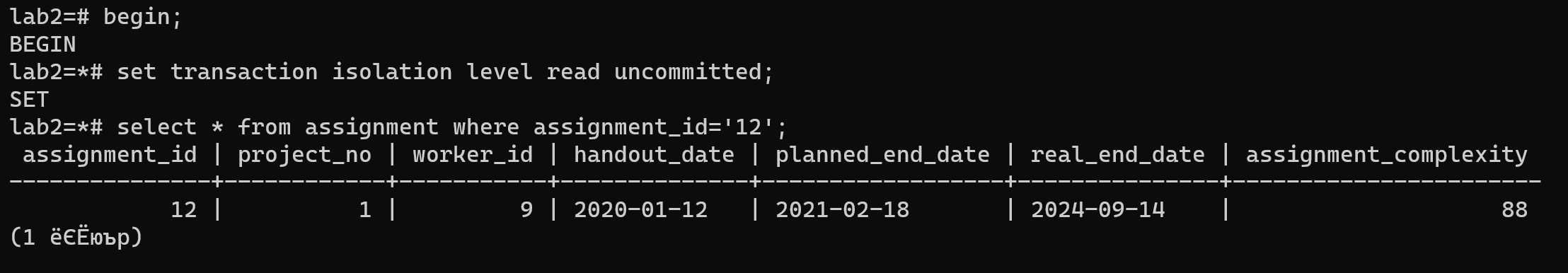


Рисунок 5.2 – Транзакция 1 (Терминал №2)

Вторая транзакция не видит изменение значения атрибута assignment\_complexity, произведенное в первой — незафиксированной — транзакции.

Чтение грязных данных не произошло, так как в PostgreSQL требования, предъявляемые к этому уровню, более строгие, чем в стандарте

Уровень изоляции **READ COMMITED**

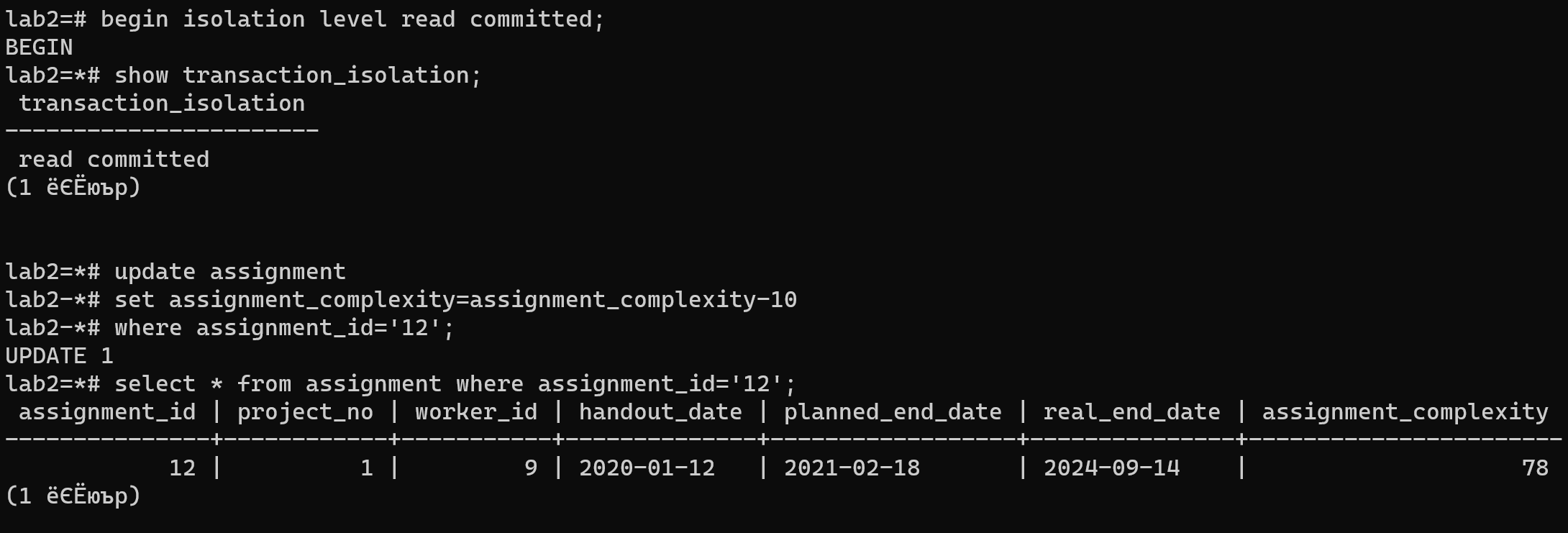


Рисунок 5.3 – Транзакция 2 (Терминал №1)

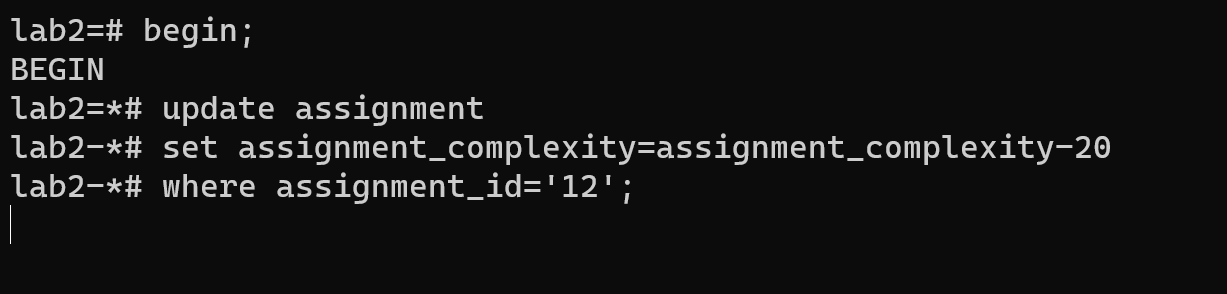


Рисунок 5.4 – Транзакция 2 (Терминал №2)

Команда UPDATE в первой транзакции заблокировала строку в таблице assignment\_tmp. После выполнения команды ROLLBACK (отмена изменений) в первом терминале, команда UPDATE во втором терминале завершилась. Таким образом, грязных чтений не производится, в отличие от неповторяющегося чтения.

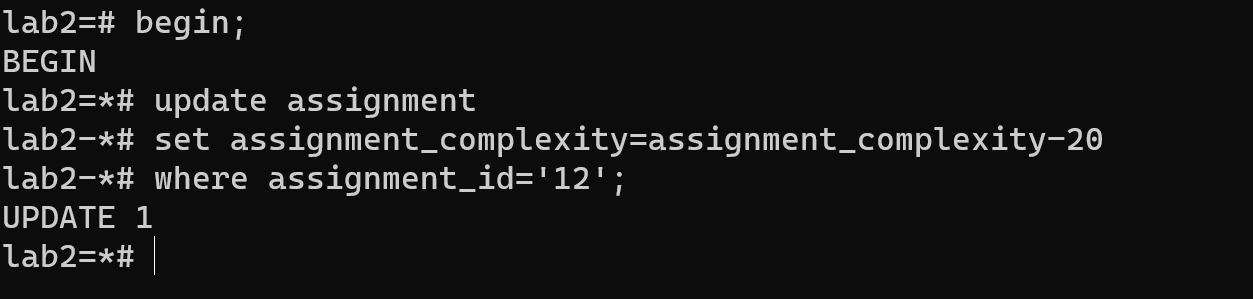


Рисунок 5.5 – транзакция 2 (Терминал №2) после завершения транзакции в первом терминале

Уровень изоляции **REPEATABLE READ**

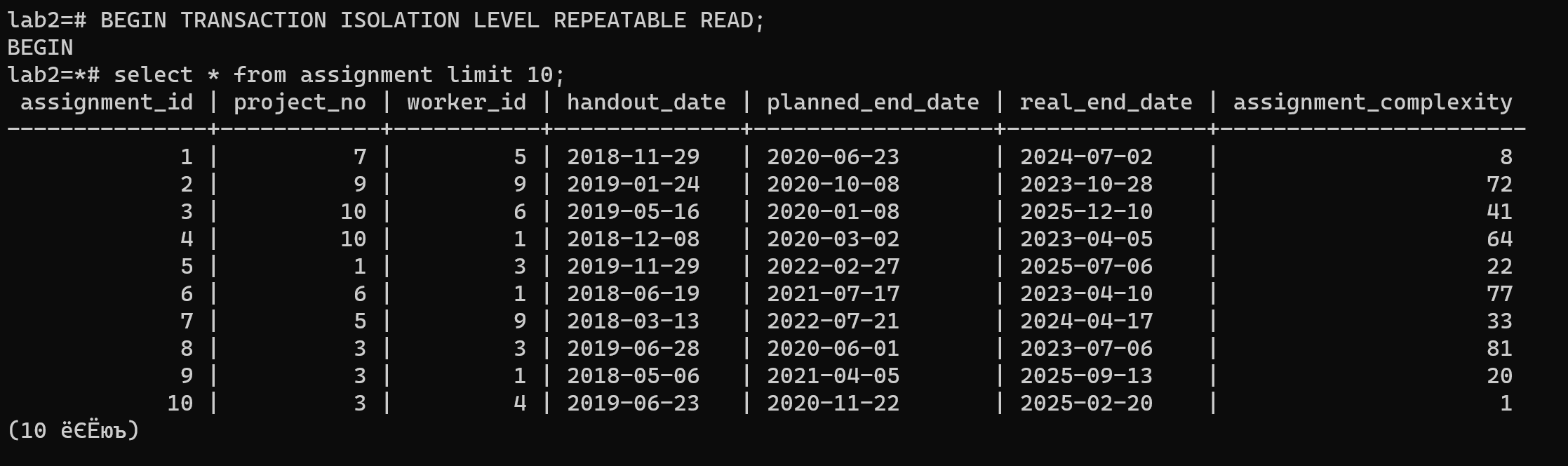
****

Рисунок 5.6 – транзакция 3 (Терминал №1)

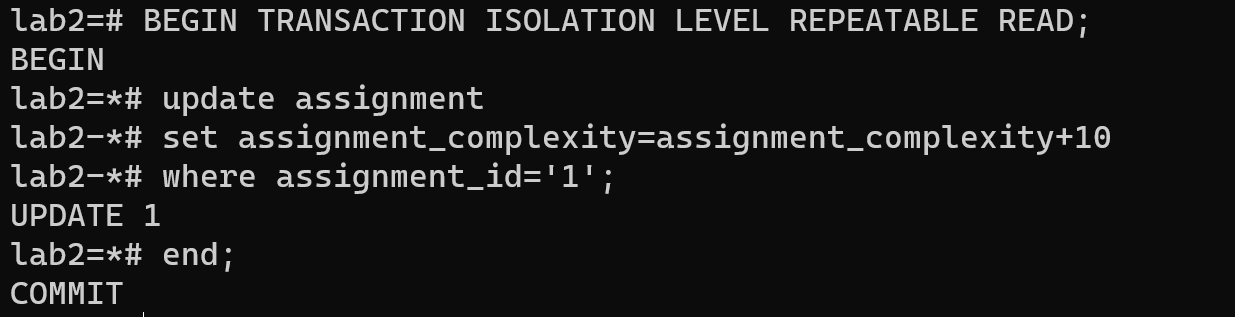


Рисунок 5.7 – транзакция 3 (Терминал №2)

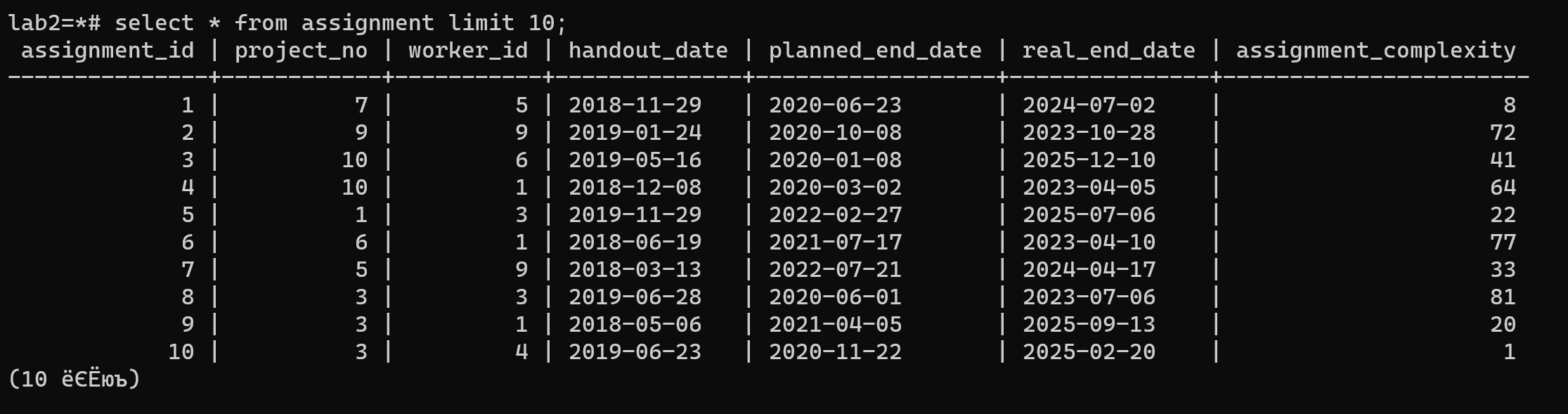


Рисунок 5.8 – вывод таблицы в первом терминале сразу после изменения строки во втором терминале

Изменения во втором терминале не повлияли на выводные данные в первом терминале, так как первый терминал всё ещё использует снимок таблицы, сделанный до преобразований.

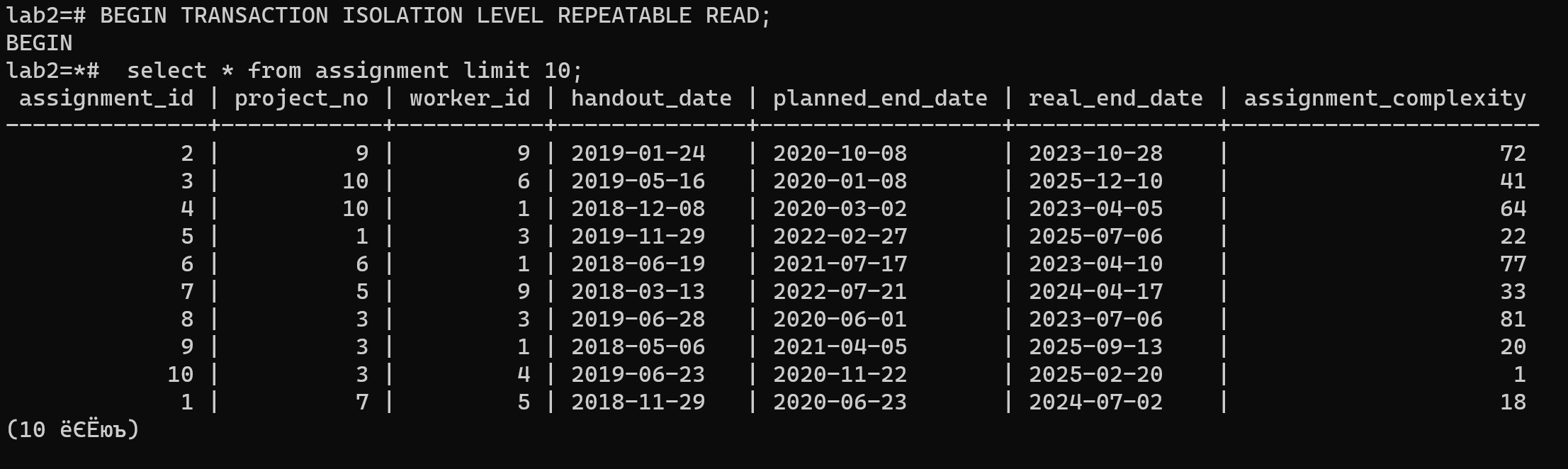


Рисунок 5.9 – вывод таблицы в первом терминале после завершения предыдущей транзакции

Таким образом, феномена неповторяющегося чтения данных не производились. Однако в PostgreSQL на этом уровне не допускается и чтение фантомных строк.

Уровень изоляции **SERIALIZABLE**

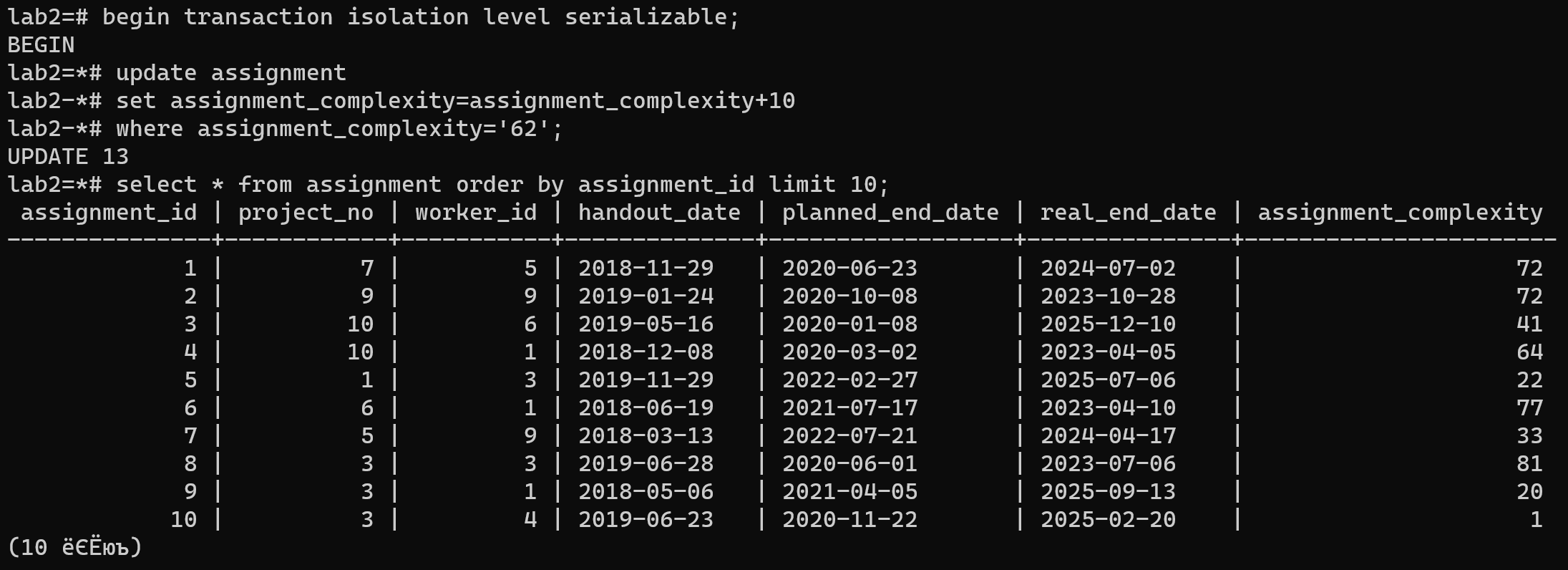


Рисунок 5.10 – Транзакция 4 (Терминал №1)

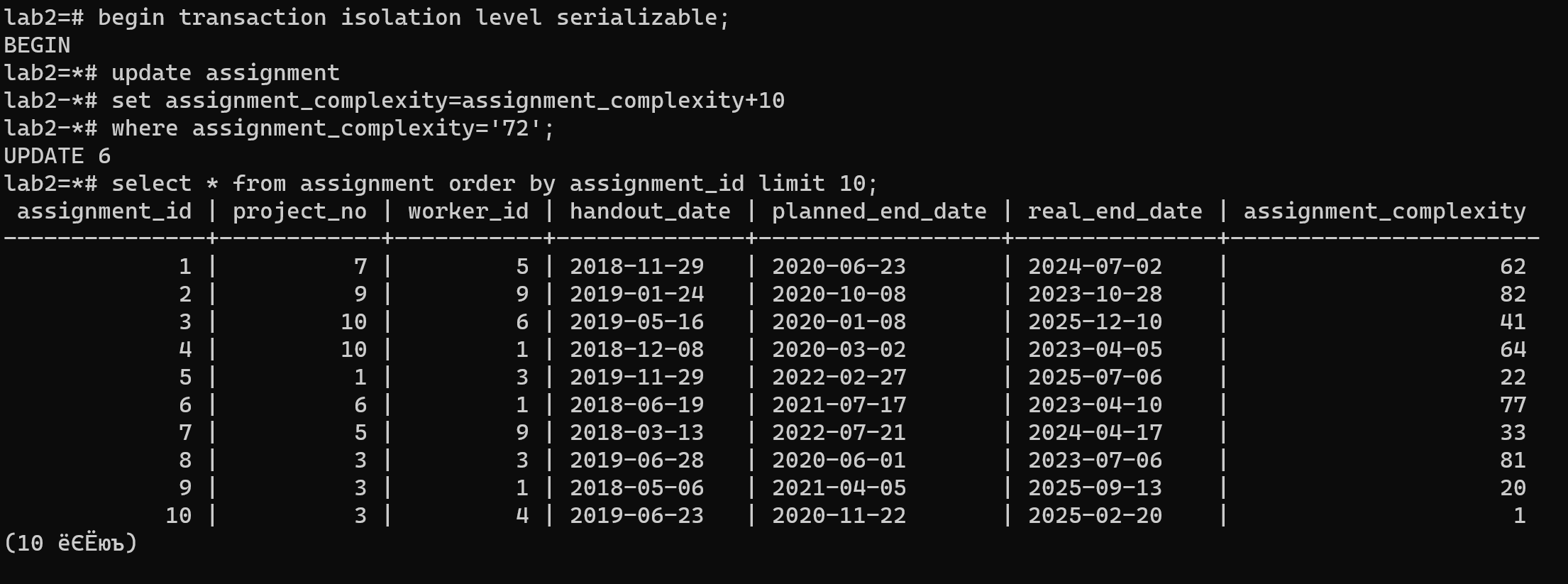


Рисунок 5.11 – Транзакция 4 (Терминал №2)

Изменение, произведенное в первой транзакции, вторая транзакция не видит, поскольку на уровне изоляции SERIALIZABLE каждая транзакция работает с тем снимком базы данных, которых был сделан в ее начале, т. е. непосредственно перед выполнением ее первого оператора. Поэтому обновляется только одна строка.

**Вывод**: мы ознакомились с операторами создания БД, изучили типы данных, научились добавлять записи в таблицы, удалять и изменять их, ознакомились с механизмами контроля согласования БД, транзакциями и триггерами.