Саратовский государственный технический университет им.

Гагарина Ю.А.

Институт прикладных информационных технологий и коммуникаций

Кафедра прикладные информационные технологии

Практическая работа №5

Выполнил студент 3 курса

Группы б1ИФСТ-31

Духов Илья

Преподаватель: Кузьмин Алексей Константинович

Саратов 2023

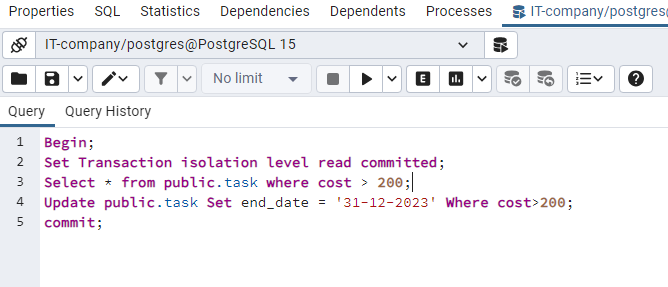
1. ***Организуйте пару параллельно работающих подключений. Последовательно продемонстрируйте возможность и невозможность появления проблем многопользовательского доступа на разных уровнях изолированности транзакций:***

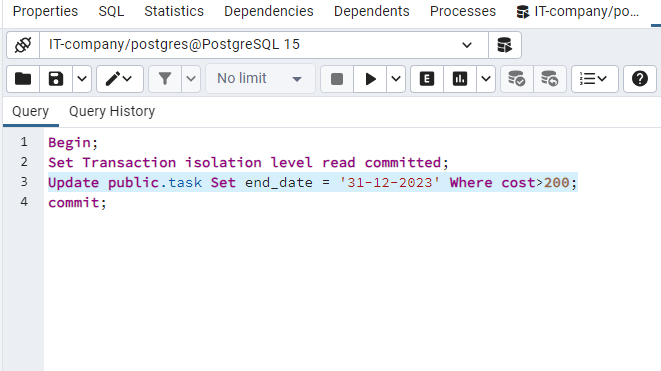
**READ COMMITTED**

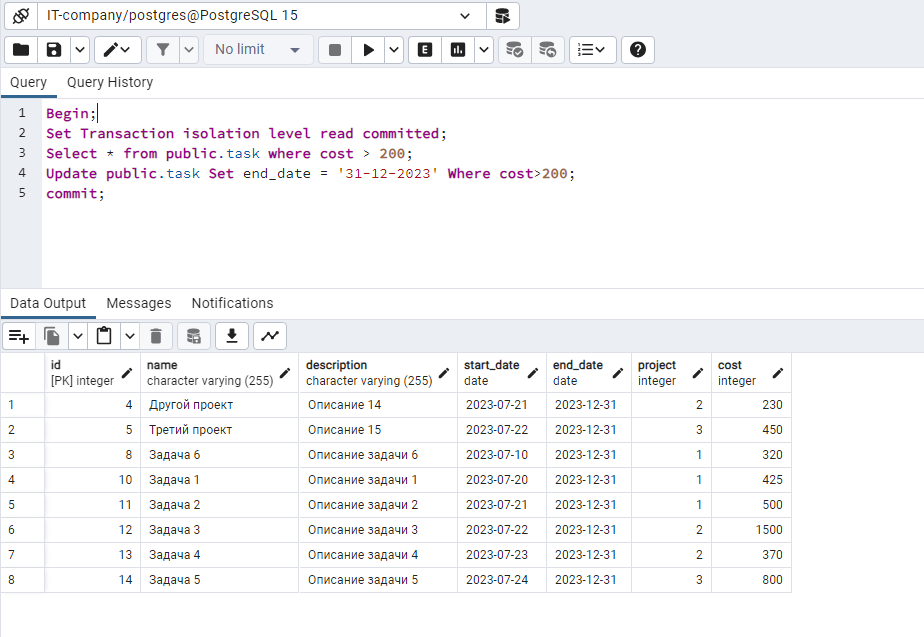
**Dirty read (нет)**

В данном примере транзакция 1 начинает транзакцию, выполняет чтение данных из таблицы и фиксирует изменения. Транзакция 2 также начинает транзакцию, выполняет обновление данных в той же таблице и фиксирует изменения.

Уровень изолированности READ COMMITTED предотвращает проблему "Dirty reads", когда одна транзакция видит изменения, сделанные другой транзакцией, которая еще не завершилась.

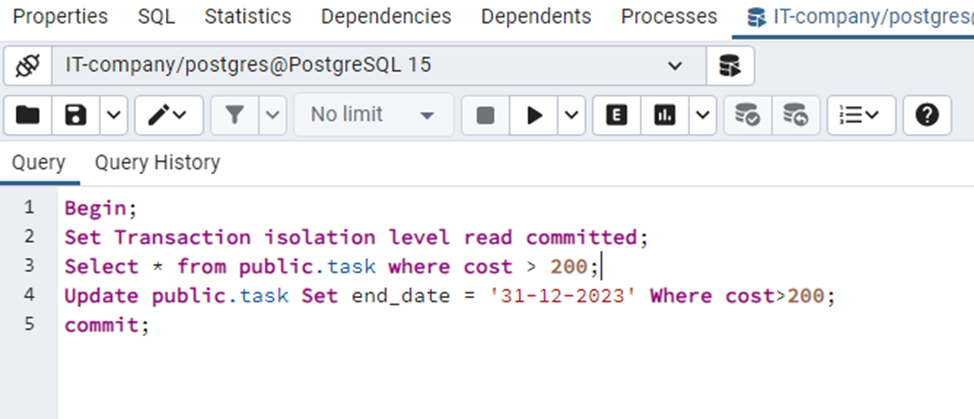


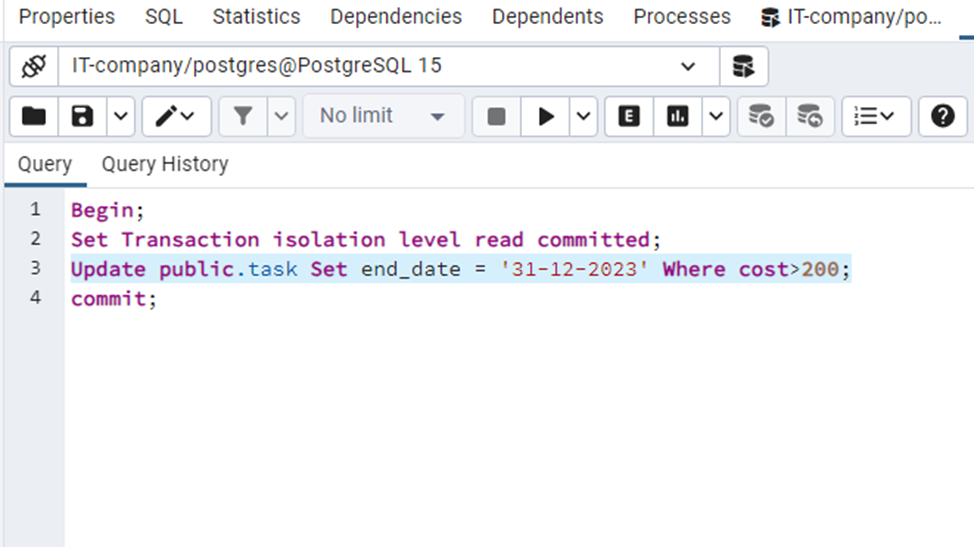


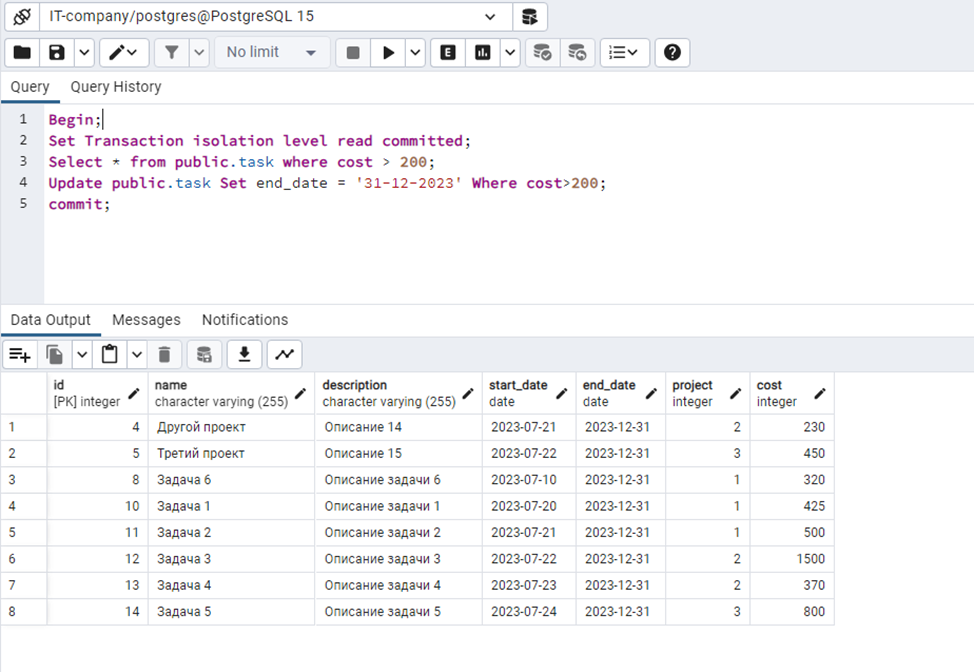


**Lost updates (нет)**

На уровне изолированности READ COMMITTED невозможна проблема "Lost updates". Если оба подключения одновременно обновляют одну и ту же запись, то одно из обновлений не может быть потеряно при фиксации транзакций. Этого не происходит из-за того, что каждая транзакция видит только фиксированные изменения других транзакций, и если две транзакции обновляют одну и ту же запись одновременно, то одно из обновлений может быть перезаписано другим.



******

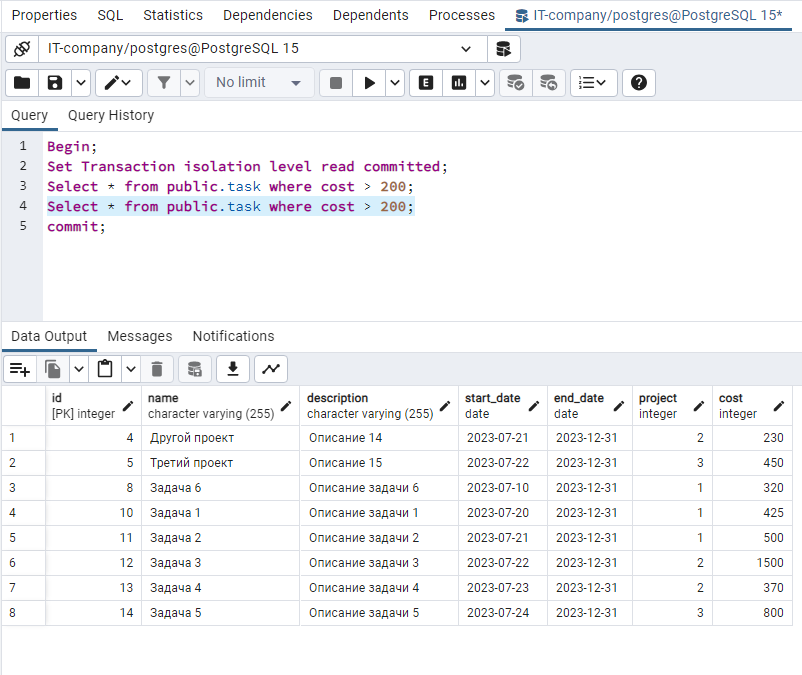
******

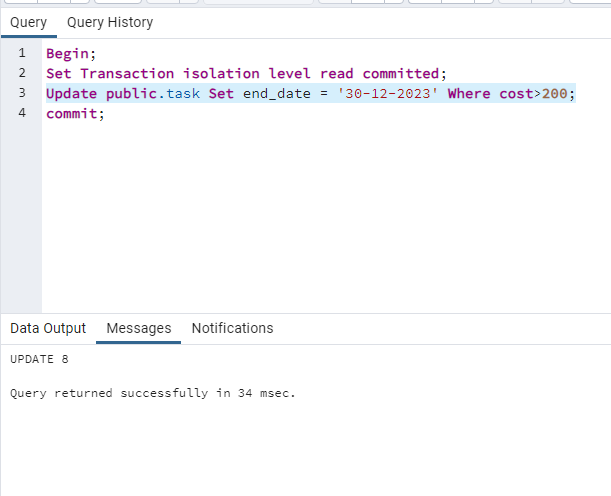
***Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

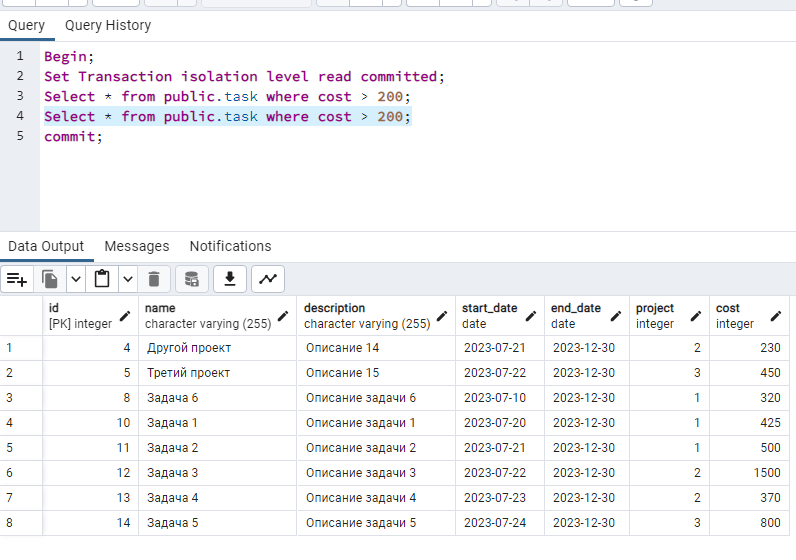
Автоматически созданное описание***

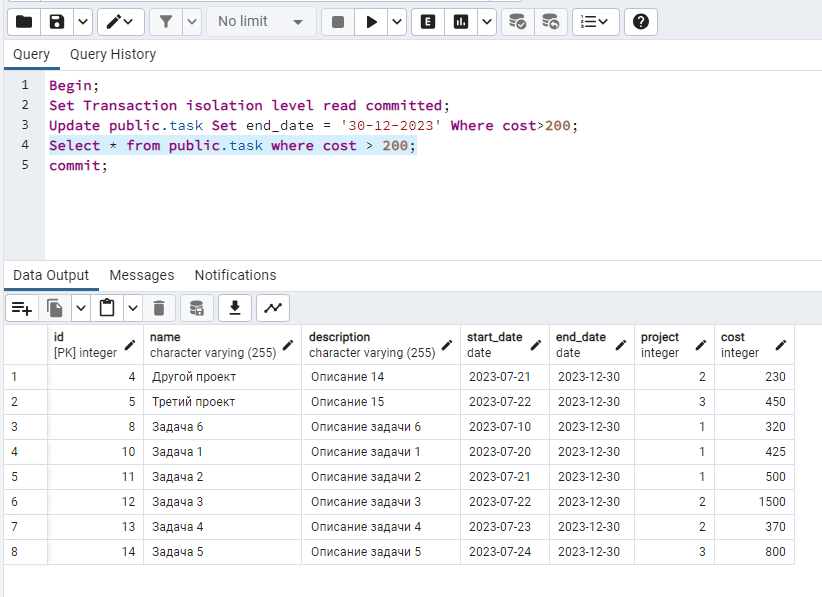
**Nonrepeatable read (есть)**

На уровне изолированности READ COMMITTED возможна проблема "Nonrepeatable read". Это происходит, когда одна транзакция читает данные из таблицы несколько раз в течение одной транзакции, и при каждом чтении видит разные значения. В нашем сценарии, если транзакция 1 читает данные до фиксации изменений, а транзакция 2 обновляет данные, то при повторном чтении данных транзакция 1 может увидеть разные значения, что называется "Nonrepeatable read".





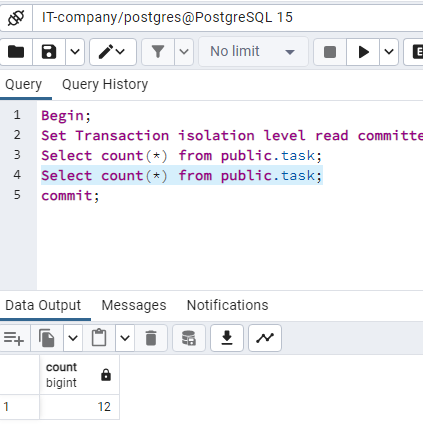


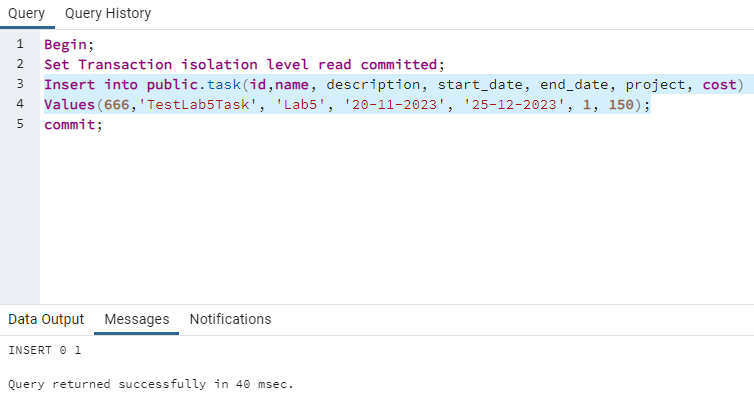


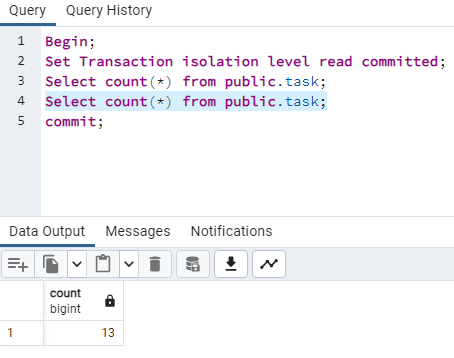
**Phantom values (есть)**

В данном примере транзакция 1 начинает транзакцию, выполняет чтение данных из таблицы и фиксирует изменения. Транзакция 2 также начинает транзакцию, выполняет вставку новой записи в ту же таблицу и фиксирует изменения.

На уровне изолированности READ COMMITTED возможна проблема "Phantom values". Это происходит, когда одна транзакция выполняет запрос, который возвращает набор строк, а затем другая транзакция вставляет новые строки, которые соответствуют условиям запроса первой транзакции. При повторном выполнении запроса первая транзакция может увидеть новые строки, которые появились после первого выполнения запроса, что называется "Phantom values".







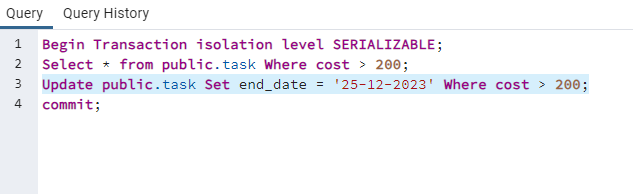


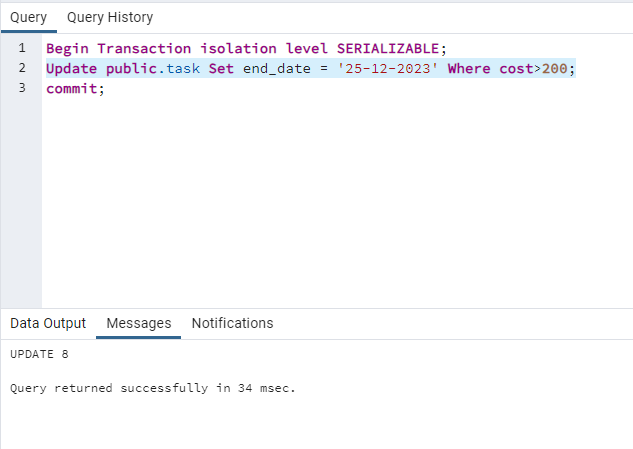
**SERIALIZABLE**

В данном примере транзакция 1 начинает транзакцию с уровнем изолированности SERIALIZABLE, выполняет чтение данных из таблицы и фиксирует изменения. Транзакция 2 также начинает транзакцию с уровнем изолированности SERIALIZABLE, выполняет обновление данных в той же таблице и фиксирует изменения.

**Lost updates (есть)**

На уровне изолированности SERIALIZABLE возможна проблема "Lost updates". Если оба подключения одновременно обновляют одну и ту же запись, то одно из обновлений может быть потеряно при фиксации транзакций. Это происходит из-за того, что уровень изолированности SERIALIZABLE гарантирует последовательное выполнение транзакций, и если две транзакции обновляют одну и ту же запись одновременно, то одно из обновлений может быть перезаписано другим.



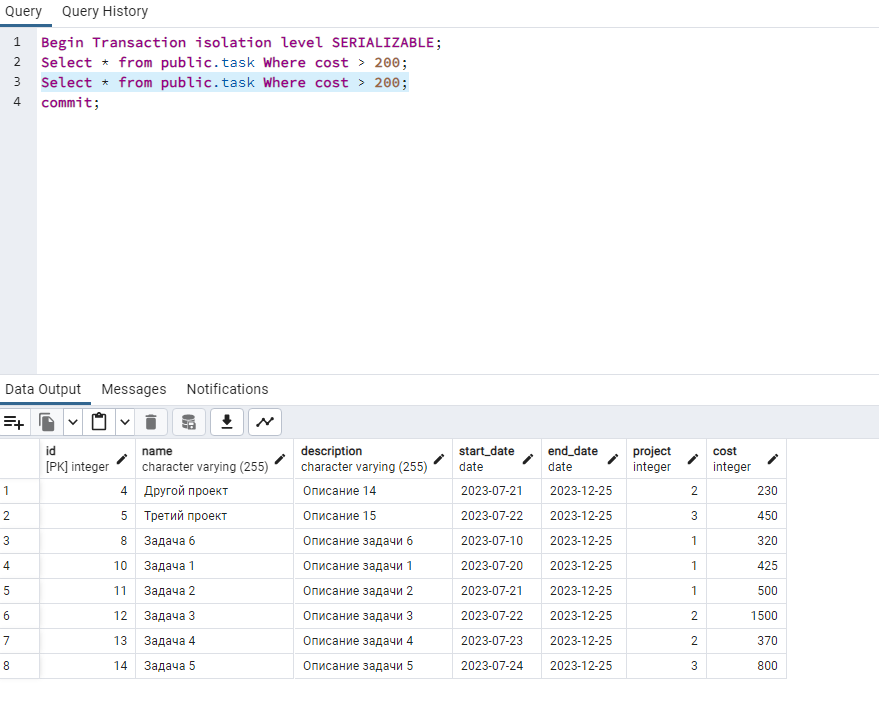


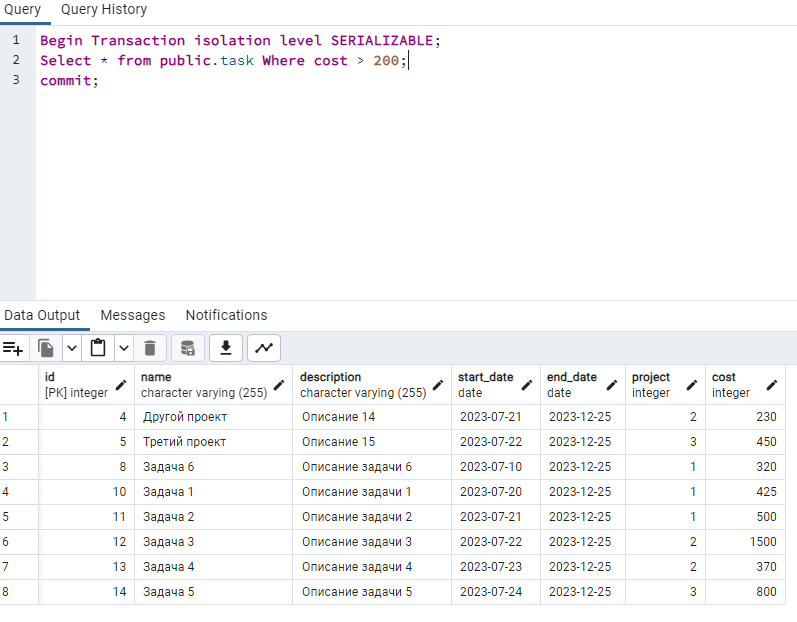
***Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание***

**Dirty Read (есть)**

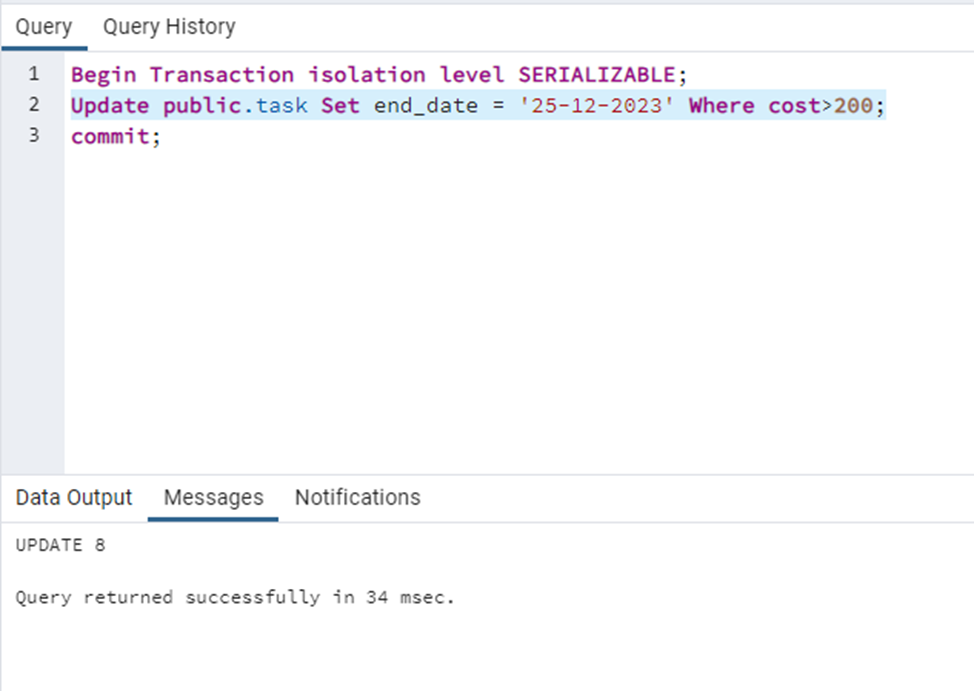
На уровне изолированности SERIALIZABLE возможна проблема "Dirty Read". Это происходит, когда одна транзакция видит изменения, сделанные другой транзакцией, которая еще не завершилась. В нашем сценарии, если транзакция 2 обновляет данные, а транзакция 1 читает данные до фиксации изменений, то транзакция 1 может увидеть "грязные" данные, которые могут быть отменены или изменены транзакцией 2.

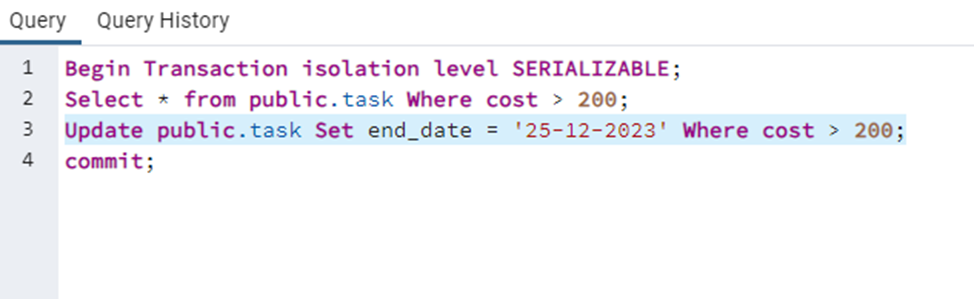


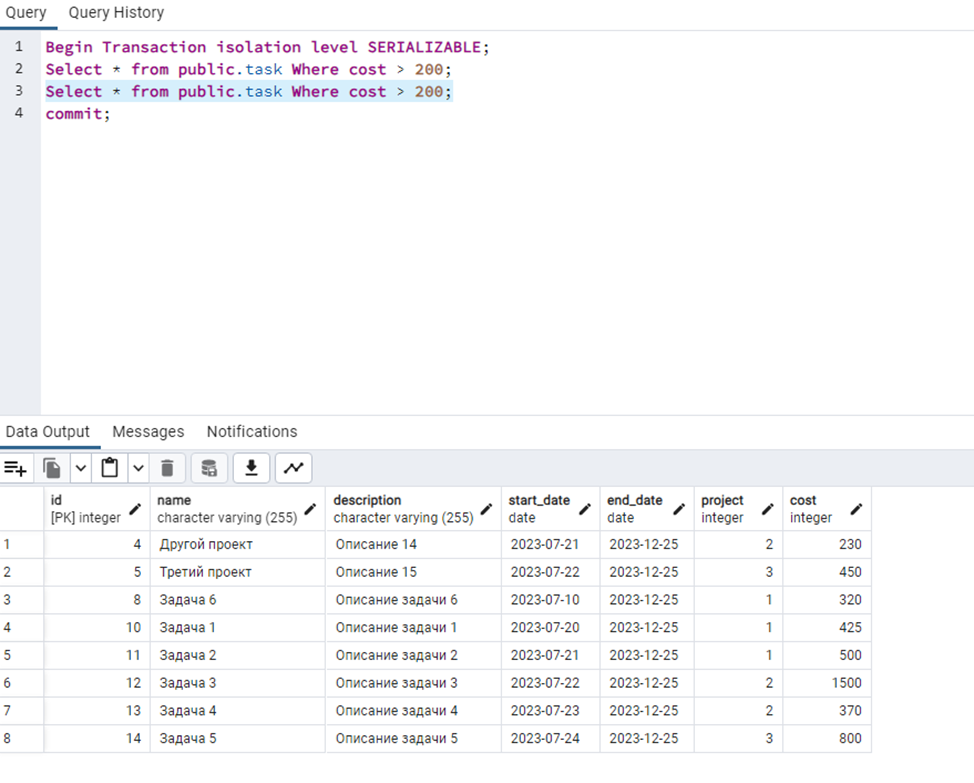


**Nonrepeatable read (есть)**

На уровне изолированности SERIALIZABLE возможна проблема "Non Repeatable Read". Это происходит, когда одна транзакция читает данные из таблицы несколько раз в течение одной транзакции, и при каждом чтении видит разные значения. В нашем сценарии, если транзакция 1 читает данные до фиксации изменений, а транзакция 2 обновляет данные, то при повторном чтении данных транзакция 1 может увидеть разные значения, что называется "Non Repeatable Read".



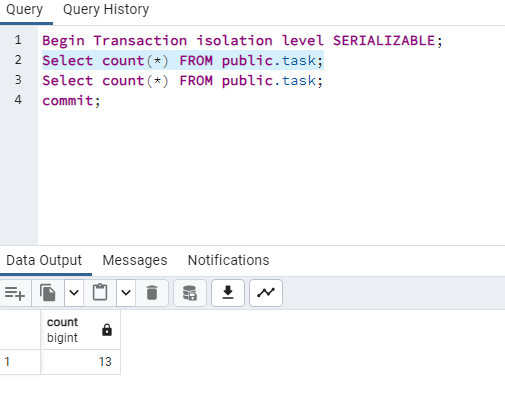


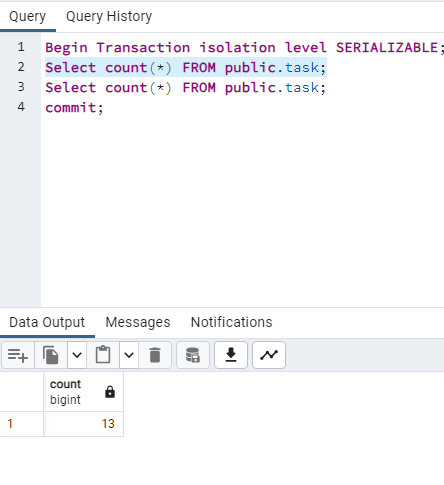


**Phantom values (Нет)**

На уровне изолированности SERIALIZABLE не возможна проблема "Phantom Values". Этого не происходит, когда одна транзакция выполняет запрос, который возвращает набор строк, а затем другая транзакция вставляет новые строки, которые соответствуют условиям запроса первой транзакции. При повторном выполнении запроса первая транзакция не может увидеть новые строки, которые появились после первого выполнения запроса.







1. ***Сначала подумайте, а затем поисследуйте источники в Интернете, чем полезен на практике уровень изолированности READ ONLY. Ведь не для ограничения доступа его специально внедряли.***

Уровень изолированности READ ONLY в PostgreSQL полезен на практике по нескольким причинам:

* **Повышение производительности:** Использование уровня изолированности READ ONLY позволяет транзакциям читать данные без блокировки других транзакций. Это может улучшить производительность в ситуациях, когда множество транзакций выполняют только чтение данных и не требуют блокировки для изменения данных.
* **Гарантированная стабильность данных:** Уровень изолированности READ ONLY гарантирует, что данные, прочитанные в рамках транзакции, не будут изменены другими транзакциями. Это полезно в ситуациях, когда требуется стабильность данных, например, при чтении отчетов или аналитических данных.
* **Предотвращение конфликтов:** Использование уровня изолированности READ ONLY может помочь предотвратить конфликты между транзакциями, особенно в ситуациях, когда несколько транзакций пытаются изменить одни и те же данные. Поскольку транзакции только читают данные, они не будут блокировать другие транзакции, что может снизить вероятность возникновения конфликтов.
* **Улучшение масштабируемости:** Использование уровня изолированности READ ONLY может помочь улучшить масштабируемость системы, поскольку транзакции, которые только читают данные, не блокируют другие транзакции, позволяя большему числу пользователей одновременно получать доступ к данным.