Саратовский государственный технический университет им.

Гагарина Ю.А.

Институт прикладных информационных технологий и коммуникаций

Кафедра прикладные информационные технологии

Практическая работа №6

Выполнил студент 3 курса

Группы б1ИФСТ-31

Духов Илья

Преподаватель: Кузьмин Алексей Константинович

Саратов 2023

****

**Занятие 6. Введение в индексирование и основы оптимизации**

**Практическое задание**

Практическое задание сдается в виде отчета, в котором для каждого задания, подразумевающего практическую работу с экземпляром Postgres, приводятся снимки экрана, на которых видны фрагменты соответствующих экранных форм, комментированные исходные коды запросов, а также результатов их выполнения на экземпляре. Также сдаётся текстовый файл \*.sql, в который включены комментированные исходные коды всех запросов по всему заданию. В случае, если задание требует использования возможности, поддержка которой в Postgres (или используемой версии Postgres) отсутствует, то необходимо отразить это в отчете и использовать альтернативный путь решения задачи.

В качестве «полигона» используйте таблицы (если необходимо), созданные в рамках подготовки ДЗ с одного из предыдущих занятий.

1. Посмотрите на имеющиеся у вас таблицы. Продумайте, какие индексы надо добавить в таблицы БД и почему? Добавьте индексы, которые продумали.

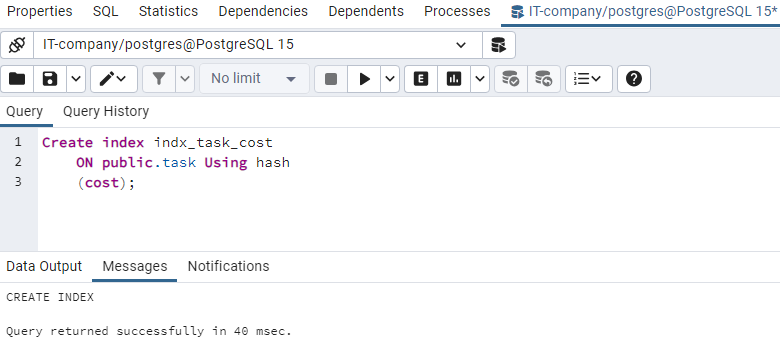
При создании индекса в PostgreSQL существует два основных типа индексов: hash и btree.

Hash-индексы используют хеш-функцию для преобразования значения столбца индекса в хеш-код. Хеш-код используется для определения положения строки в индексе. Hash-индексы обеспечивают очень быстрое выполнение запросов на равенство, но неэффективны для выполнения запросов на диапазон.

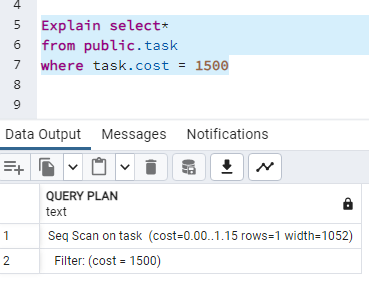
B-Tree-индексы используют древовидную структуру для хранения значений столбца индекса. B-Tree-индексы обеспечивают быстрое выполнение запросов на равенство, диапазон и порядок.

В целом, hash-индексы лучше подходят для случаев, когда требуется быстрое выполнение запросов на равенство. B-Tree-индексы лучше подходят для случаев, когда требуется быстрое выполнение запросов на диапазон и порядок.

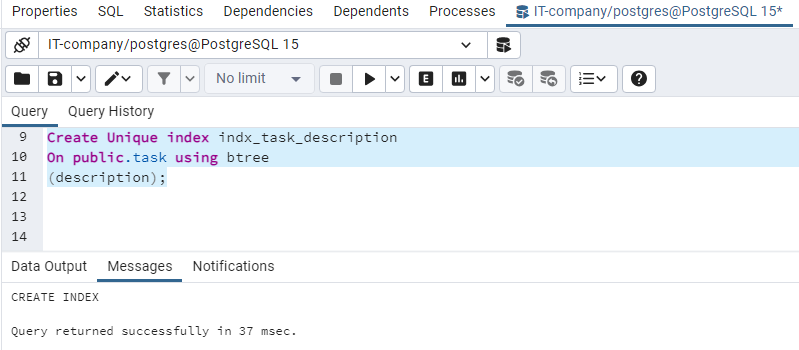
Добавим индекс, который будет искать цену задачи и тип hash-index для быстрого выбора.

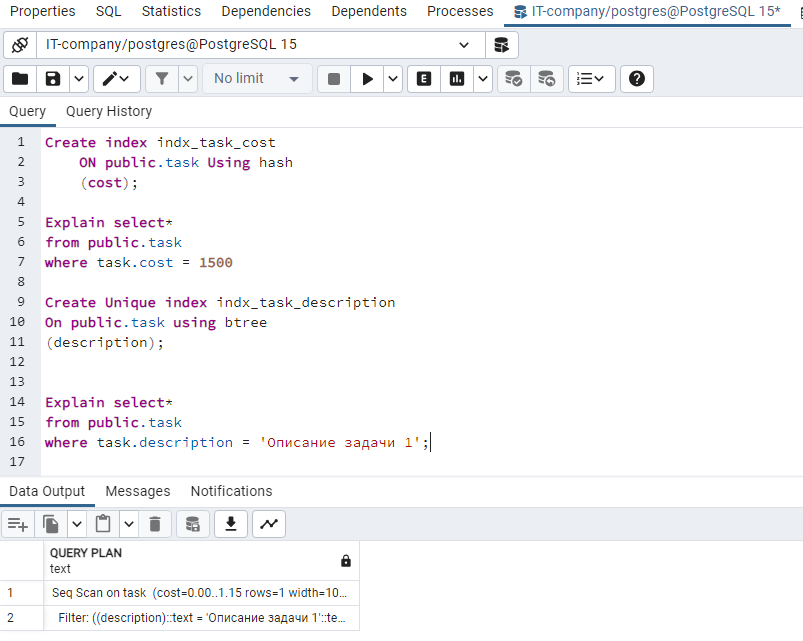


С помощью ключевого слова EXPLAIN вы можете многое сделать для определения стоимости и эффективности своих запросов. Оно помогает найти места, где потребуется тонкая настройка запросов.

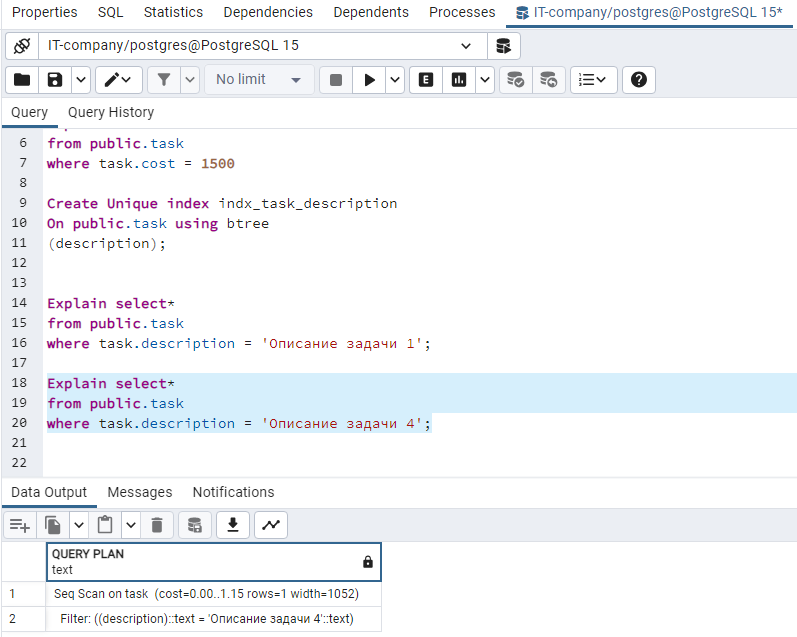


Далее создадим индекс и будем использовать сбалансированное дерево.

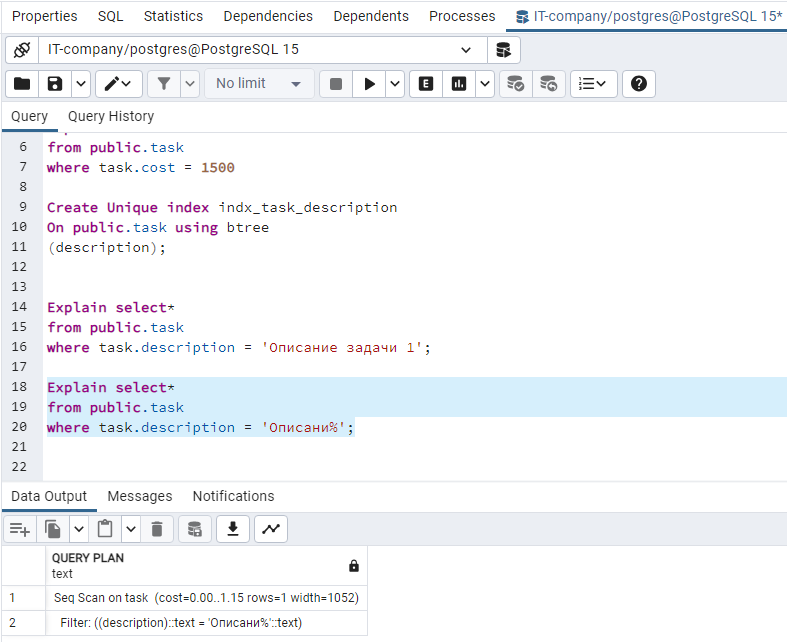




1. Попробуйте придумать такие запросы к таблицам, которые а) будут игнорировать индекс б) будут задействовать индекс. Продумайте и объясните, почему для случаев (а) индекс игнорируется, а для случаев (б) индекс используется.

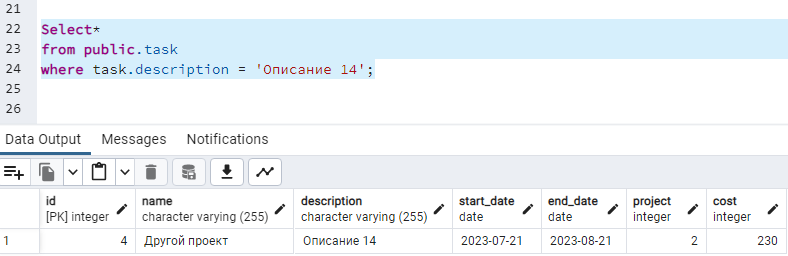


(2 пример):

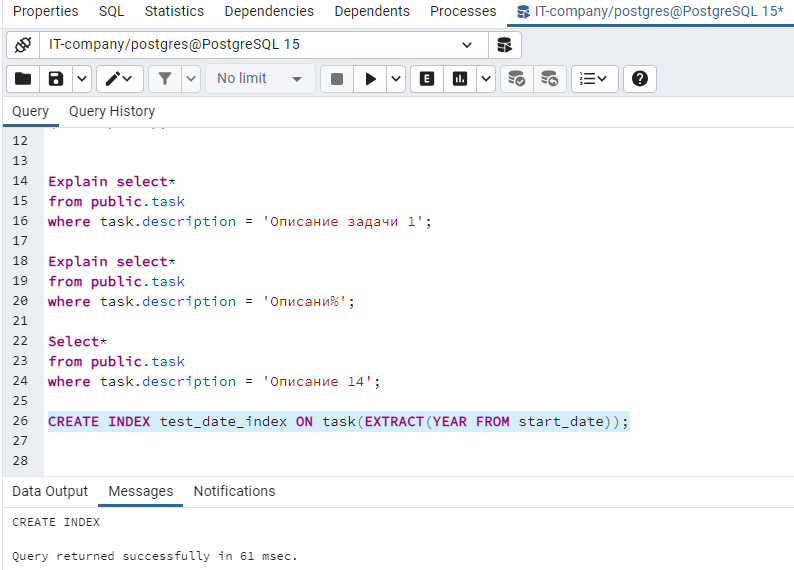


Игнорирование индекса:

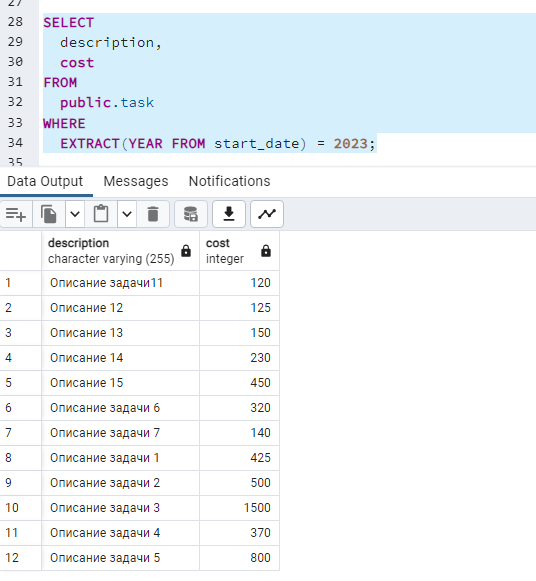
Запрос, использующий функцию или оператор, несовместимые с индексом:



1. Для одной из Ваших таблиц реализуйте индекс по некоторой функции. Постройте запрос, который бы его использовал.

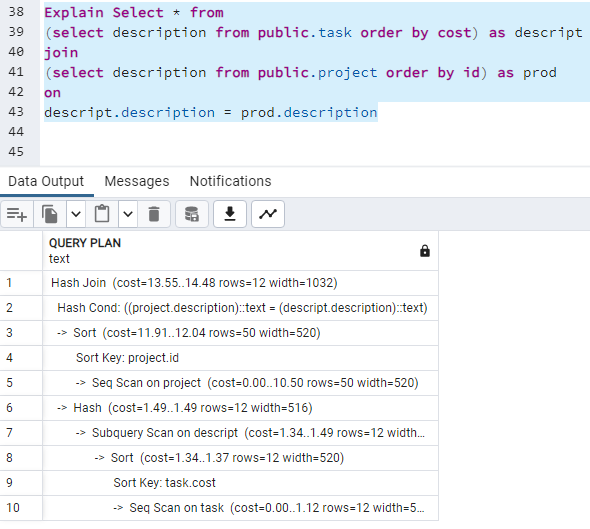


Этот запрос позволяет ускорить поиск строк, содержащих начало задач, совершенные в определенный год.



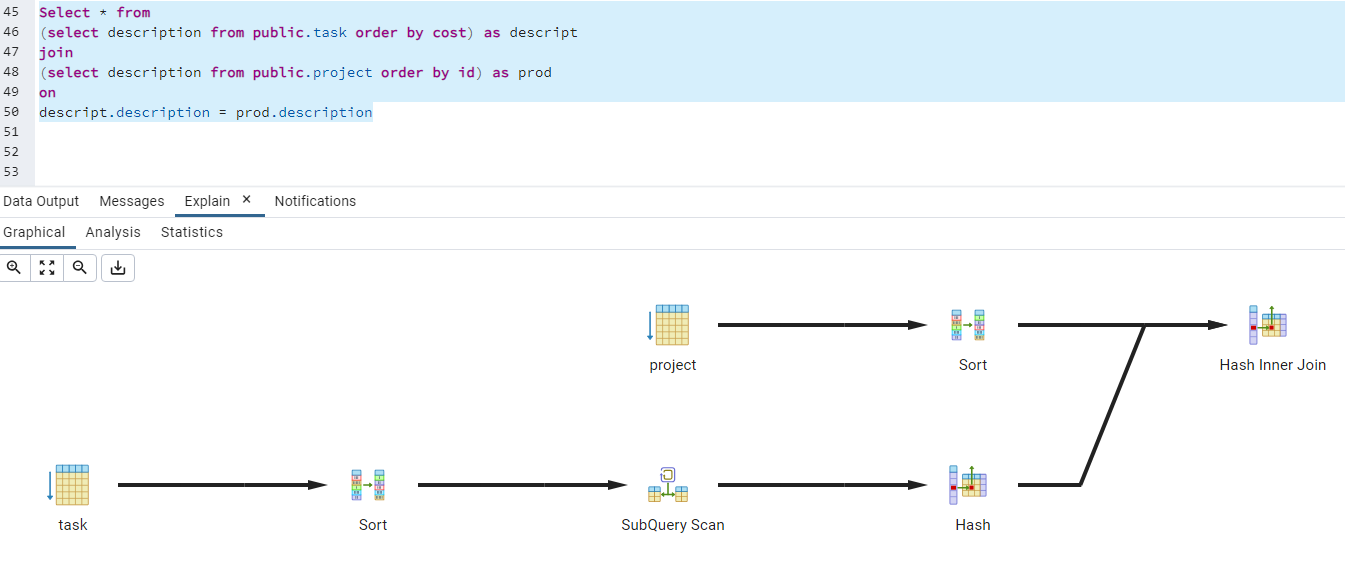
1. Сделайте так, чтобы для каких-либо двух связанных таблиц оптимизатор использовал алгоритм соединения merge join. Продемонстрируйте такой запрос и план его выполнения.

Оптимизатор будет использовать соединение набора строк с помощью merge join, когда оба набора должны будут предварительно отсортированы по тем столбцам, по которым производится соединение



1. Самостоятельно найдите в интернете описание способа получения плана выполнения запроса без использования GUI. Попробуйте его реализовать.

Если вы предпочитаете использовать графический интерфейс, вы можете воспользоваться инструментом pgAdmin, который предоставляет анализатор плана выполнения запроса. В pgAdmin вы можете выполнить запрос, на результате и выбрать "Explain" для получения плана выполнения.



1. Почитайте про такие не рассмотренные нами на лекции вещи, как:
   1. Видимые и невидимые индексы

В PostgreSQL индексы бывают видимыми и невидимыми. Видимые индексы используются по умолчанию, и они учитываются при выполнении запросов. Невидимые индексы не учитываются при выполнении запросов, но они могут быть использованы для ускорения выполнения некоторых операций, таких как сортировка и объединение таблиц.

Невидимые индексы - невидимые индексы создаются с помощью оператора CREATE INDEX с опцией VISIBLE.

Причины использования невидимых индексов:

* Когда индекс используется только для ускорения определенных операций, таких как сортировка или объединение таблиц.
* Когда индекс используется для ускорения выполнения запросов, которые выполняются редко.
* Когда индекс может повлиять на производительность запросов, которые выполняются часто.
  1. Полнотекстовые индексы

Полнотекстовые индексы в PostgreSQL позволяют ускорить выполнение запросов, в которых используется полнотекстовое сравнение. Полнотекстовое сравнение - это сравнение текста, которое учитывает морфологию слов, например, слияние слов, множественное число и т.д.

Полнотекстовые индексы могут быть использованы для ускорения выполнения следующих типов запросов:

* Запросы на поиск строк, содержащих определенное слово или фразу.
* Запросы на поиск строк, содержащих слово или фразу в определенном порядке.
* Запросы на поиск строк, содержащих слово или фразу в определенном контексте.

Типы полнотекстовых индексов

* GIN индексы. GIN индексы используют древовидную структуру для хранения информации о полнотекстовом сравнении. GIN индексы обеспечивают хорошее соотношение производительности и размера.
* GiST индексы. GiST индексы используют более сложную структуру для хранения информации о полнотекстовом сравнении. GiST индексы обеспечивают лучшее соотношение производительности и точности, чем GIN индексы.

Выбор типа полнотекстового индекса

* Требуемая производительность. Если требуется высокая производительность, то следует использовать GIN индексы.
* Требуемая точность. Если требуется высокая точность, то следует использовать GiST индексы.
* Размер индекса. GIN индексы обычно меньше, чем GiST индексы.

Примеры использования полнотекстовых индексов

* Системы управления документами. Полнотекстовые индексы могут быть использованы для ускорения поиска документов по их содержанию.
* Системы поиска информации. Полнотекстовые индексы могут быть использованы для ускорения поиска информации в Интернете.
* Системы продаж. Полнотекстовые индексы могут быть использованы для ускорения поиска товаров по их описанию.
  1. Column-store индексы

Column-store индексы - это особый тип индексов, которые хранят значения столбцов в виде столбцов, а не строк. Это позволяет ускорить выполнение запросов, которые требуют операций с диапазоном или порядка по нескольким столбцам.

В этом случае column-store индекс будет использоваться для быстрого поиска строк, где значение столбца product\_id находится в заданном диапазоне.

Column-store индексы могут быть использованы для ускорения выполнения следующих типов запросов:

* Запросы на поиск строк, где значения столбцов находятся в заданном диапазоне.
* Запросы на сортировку строк по нескольким столбцам.
* Запросы на агрегирование данных по нескольким столбцам.

Преимущества column-store индексов

* Column-store индексы имеют следующие преимущества:
* Улучшенная производительность запросов с диапазоном и порядка.
* Column-store индексы могут значительно ускорить выполнение запросов с диапазоном и порядка по нескольким столбцам.
* Уменьшение размера индекса. Column-store индексы обычно меньше, чем традиционные B-Tree индексы.
* Улучшенная масштабируемость. Column-store индексы лучше масштабируются на большие объемы данных.

Недостатки column-store индексов

* Увеличение сложности обновления индекса.
* Обновление column-store индекса может быть более сложным, чем обновление традиционного B-Tree индекса.
* Увеличение сложности создания индекса.
* Создание column-store индекса может быть более сложным, чем создание традиционного B-Tree индекса.

1. Немного отдохните. Это сложная и интересная тема. Вы проделали большую работу ☺