**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Казанский государственный энергетический»**

**Отчет по лабораторной работе 3**

**Индексный доступ**

Выполнил:

студент группы ПИ-1-22

Соловьёв Л.А.

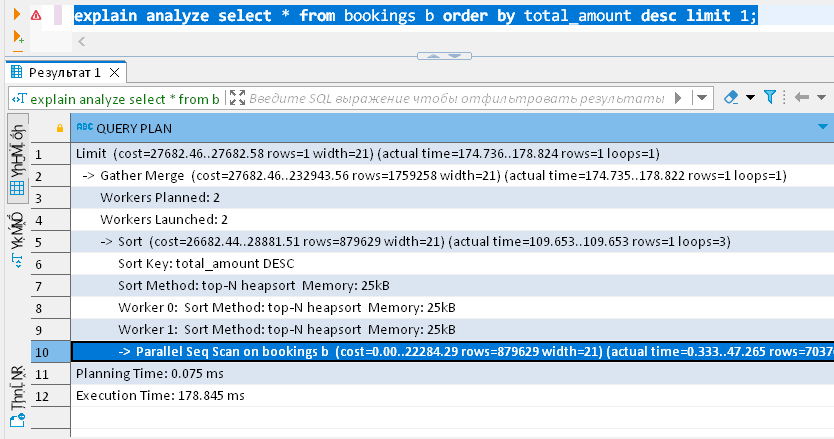
Проверил:

Хамитов Р. М.

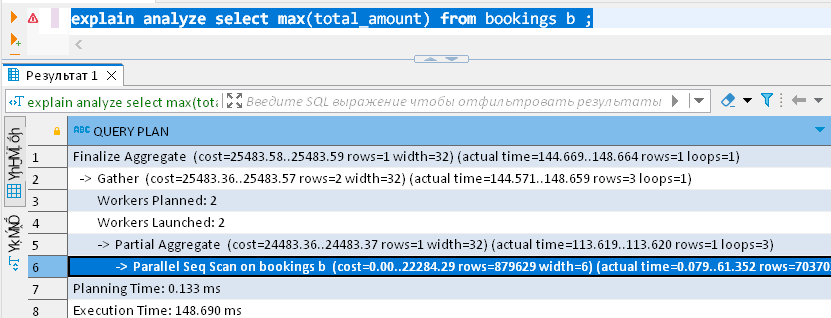
Казань 2024

**1. Напишите запрос, выбирающий максимальную сумму бронирования. Проверьте план выполнения. Какой метод доступа выбрал планировщик? Эффективен ли такой доступ?**

**Этот запрос можно написать как минимум двумя разными способами(order by + limit, max). Попробуйте оба варианта.**

**explain** **analyze** **select** \* **from** bookings b **order** **by** total\_amount **desc** **limit** 1;  


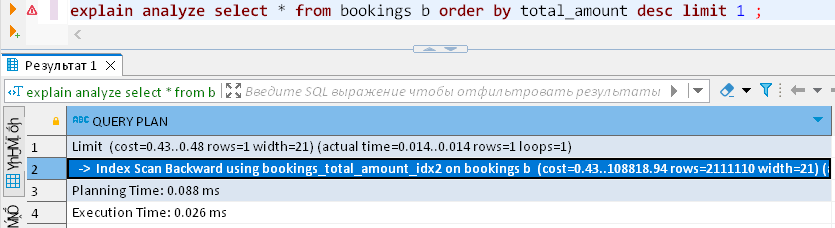
**explain** **analyze** **select** **max**(total\_amount) **from** bookings b ;

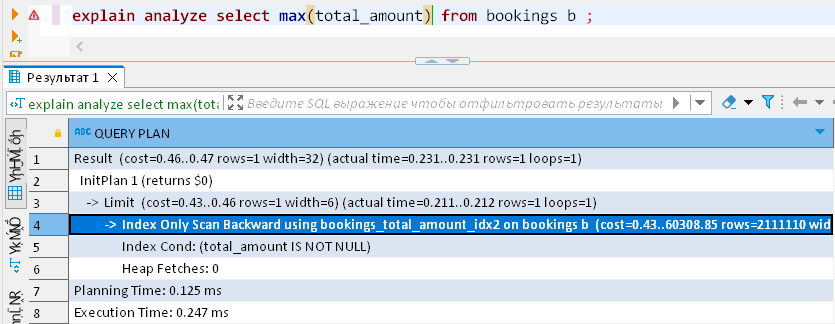


В первом случае планировщик сначала параллельно сканирует всю таблицу, а затем сортирует их по убыванию. Это неэффективно, так как по итогу нам нужно получить только одно значение, а все значения сортируются полностью.  
Во втором случае планировщик сначала параллельно сканирует всю таблицу, затем выполняется агрегатная функция. Это эффективнее, чем первый запрос, потому что нет нужды в дополнительной сортировки и отсутствует условия выбора limit 1

**2. Создайте индекс по столбцу bookings.total\_amount. Снова проверьте план выполнения запроса. Какой метод доступа выбрал планировщик теперь?**

Создание индекса: **create** **index** **on** bookings(total\_amount);



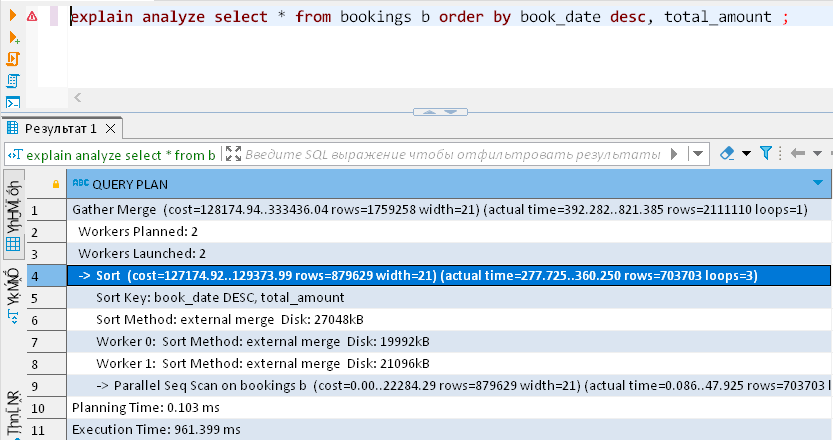
При использовании order by + limit планировщик выбирает индексное сканирование, которое является более эффективным чем последовательное или параллельное сканирование.  


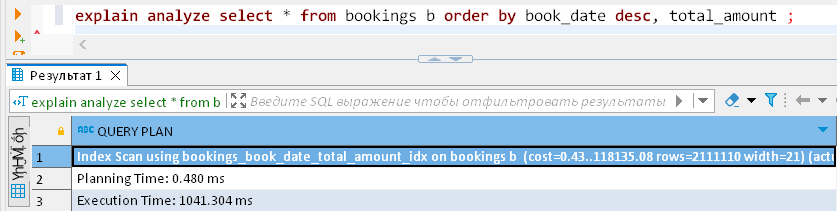
При использовании max планировщик выбирает только индексное сканирование, которое является более эффективным чем последовательное или параллельное сканирование.  
При этом из плана запроса видно, что планировщик так же использует limit, как и в первом запросе, т.е. переформулировал запрос.

**3. При создании индекса можно указать порядок сортировки столбца. Зачем, если индекс можно просматривать в любом направлении?**

Это необходимо для запросов с сортировкой в разных направлениях, например, упорядочить бронирования по убыванию даты и возрастании цены

Без индекса:

  
Создание индекса: **create** **index** **on** bookings(book\_date **desc**, total\_amount);  
С индексом:

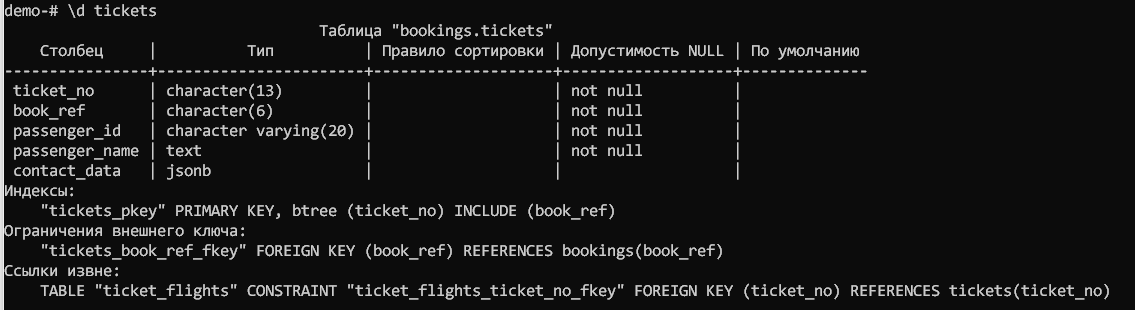


**4. В демонстрации был создан include-индекс для таблицы билетов. Замените им индекс, поддерживающий первичный ключ таблицы.  
Используйте команду ALTER TABLE … DROP CONSTRAINT для удаления старого ограничения целостности вместе с соответствующим индексом, и команду ALTER TABLE … ADD CONSTRAINT … USING INDEX для создания нового ограничения, используя существующий индекс.**

**create** **unique** **index** idx\_ticket\_no\_book\_ref **on** tickets(ticket\_no) **include** (book\_ref);

**alter** **table** tickets **drop** **constraint** tickets\_pkey **cascade**;

**alter** **table** tickets **add** **constraint** tickets\_pkey **primary** **key** **using** **index** idx\_ticket\_no\_book\_ref;

**alter** **table** ticket\_flights **add** **foreign** **key** (ticket\_no) **references** tickets(ticket\_no);  


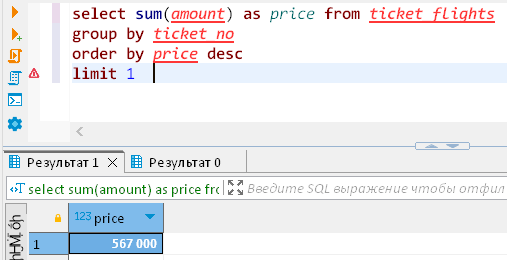
**5. Сколько стоит самый дешевый и самый дорогой билет? Сколько по времени занял этот запрос?  Найдите способ ускорить данный запрос.**

**Максимум: select** **sum**(***amount***) **as** *price* **from** ***ticket\_flights***

**group** **by** ***ticket\_no***

**order** **by** ***price*** **desc**

**limit** 1



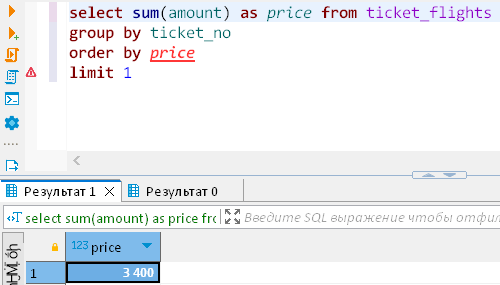
Этот запрос выполнялся 21.3 секунд.  


**Минимум: select** **sum**(amount) **as** price **from** ticket\_flights

**group** **by** ticket\_no

**order** **by** price

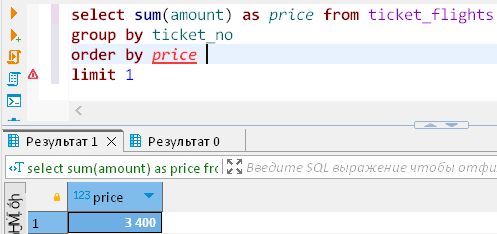
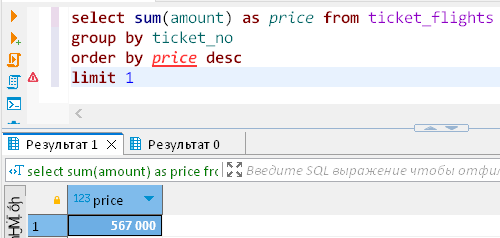
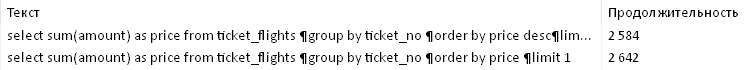
**limit** 1



Этот запрос так же занял 21.6 секунд.

  
  
**Создадим индекс:**

**create** **index** **on** ticket\_flights(ticket\_no, amount)

  
  
  
Оба этих запроса заняли примерно 2.5 секунды, после создания индекса выполнение запросов ускорилось в 21.45/2.6 = 8.25 раз.