**Лабораторная работа №10**

**Невейков Андрей, 2022**

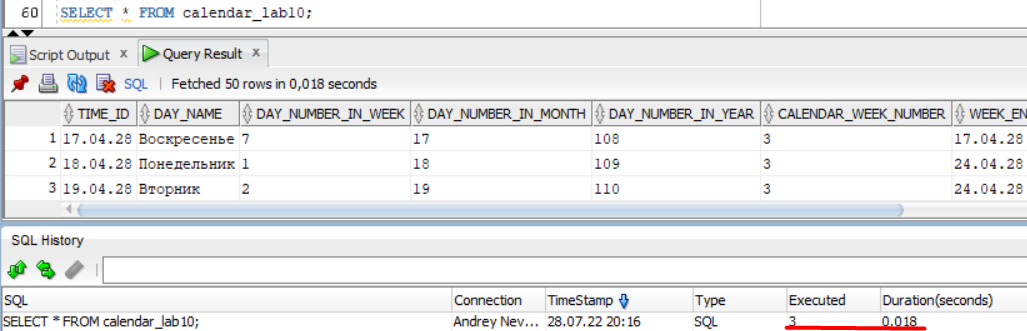
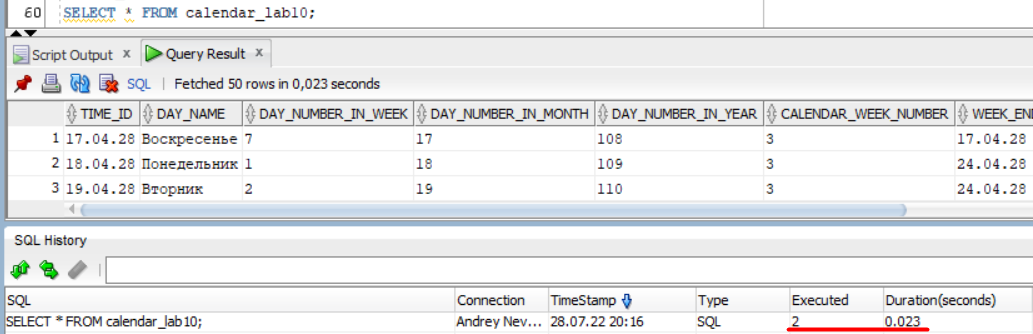
**Ссылка на GitHub со скриптом** <https://github.com/AndreyNeveikov/DataMola_Labs/blob/main/Modul_1/lab10/parallel_calendar_lab10.sql>

**Методология:**

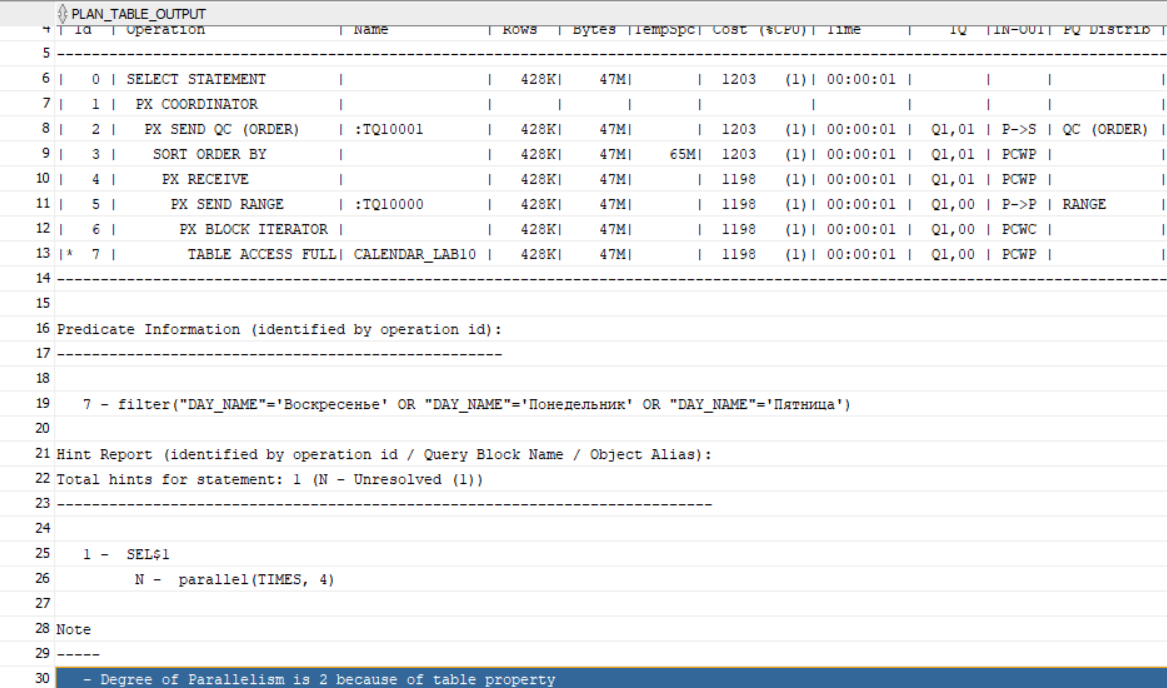
Для проверки эффективности ***Parallel execution*** был взят календарь из лабораторной работы №7. Т.к. parallel execution эффективен на больших объемах данных была создана выборка на 1.000.000 дней (один миллион). Также т.к. при выполнении команд, Oracle подгружает в кэш часто выполняемые операции, то после каждой операции таблица удалялась и пересоздавалась заново.

(В ходе выполнения лабораторной работы я проверил, как повлияет на время выполнения повтор операции без удаления таблицы с ней: с удаленение результат одной и той же операции отличался на одну-две тысячные секунды (0.001-0.002), без значение погрешности могло быть и на порядок больше (в среднем 0.017-0.023) секунды.

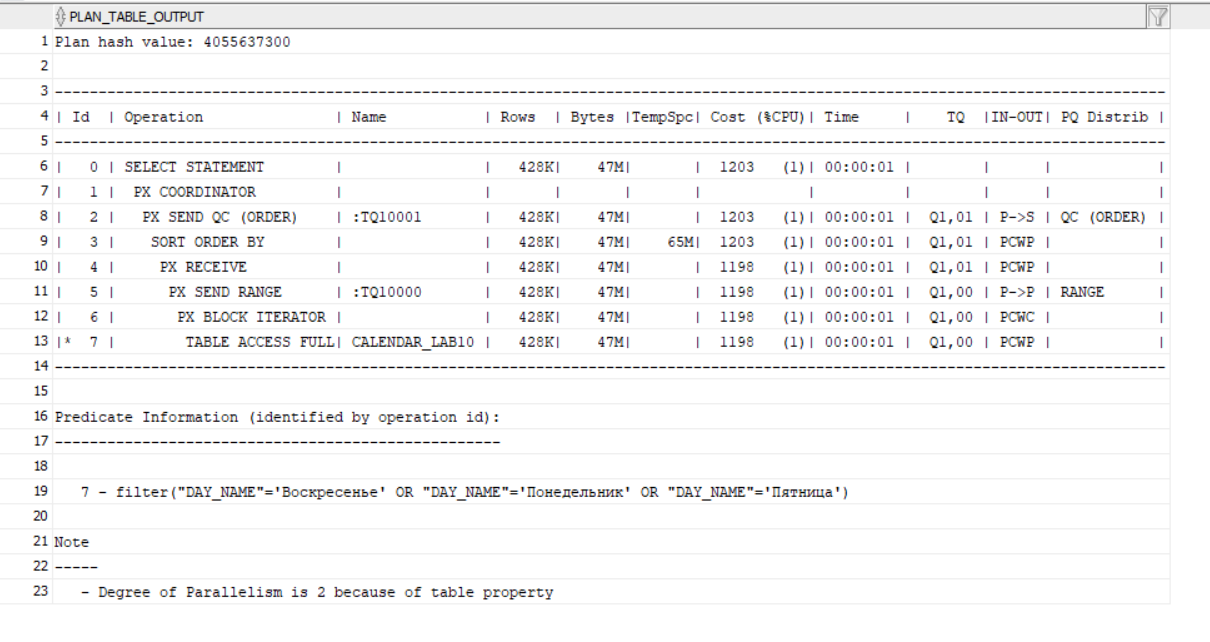
На скриншотах ниже показано как изменяется время выполнения, если выполнить запрос 2 и 3 раза подряд.



**TASK\_1** Select Parallel execution

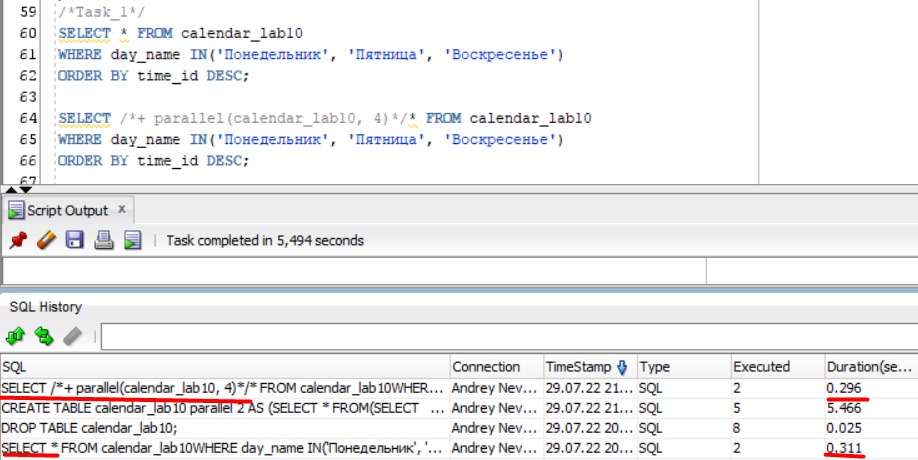


Execution Plan with parallel

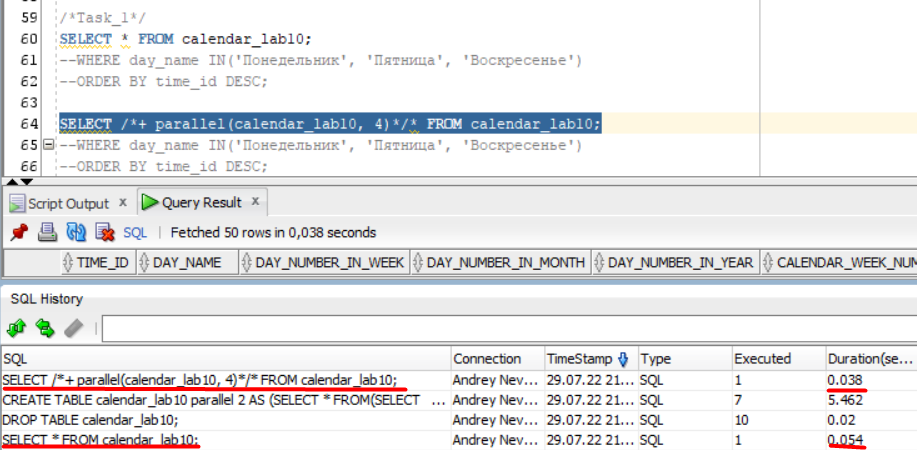


Execution Plan without parallel

Как следует из Execution Plan-ов, Oracle для больших выборок может сам использовать параллельные вычисления т.к. они почти идентичны.

****

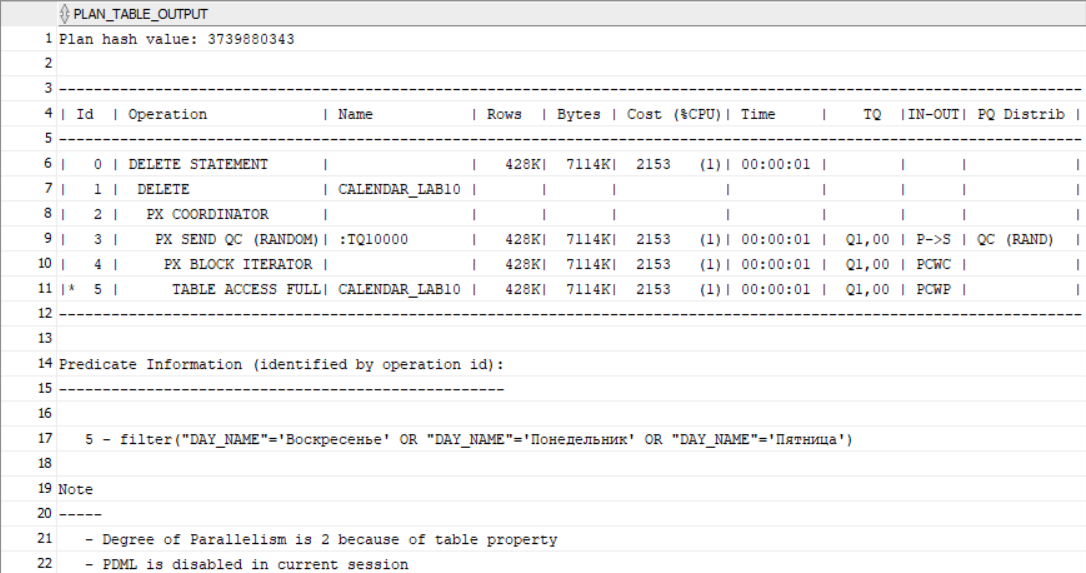
**Вывод:** parallel execution на выборке в 1 млн сток увеличил производительность на 4,9% при сложном запросе.

****

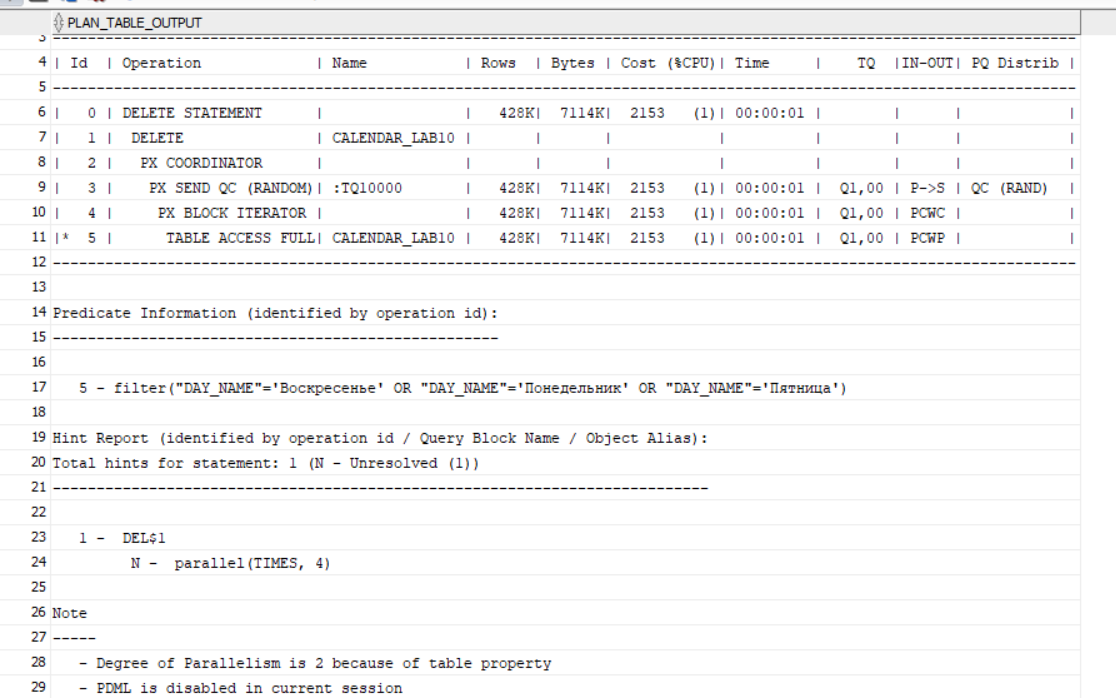
**Вывод:** parallel execution на выборке в 1 млн сток увеличил производительность на 29.7% при простом запросе.

Т.к. в работе часто будет необходимо делать запросы с условиями, то дальше будут использоваться только они.

**TASK\_2** Parallel DML

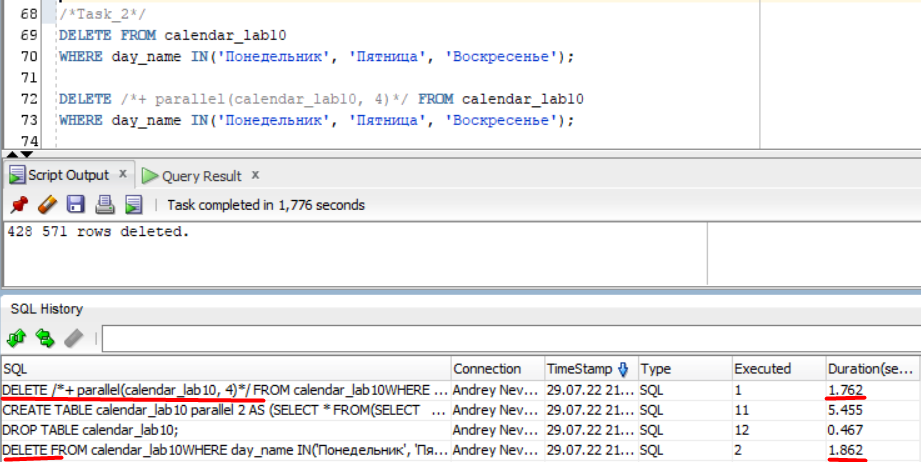


Execution Plan without parallel



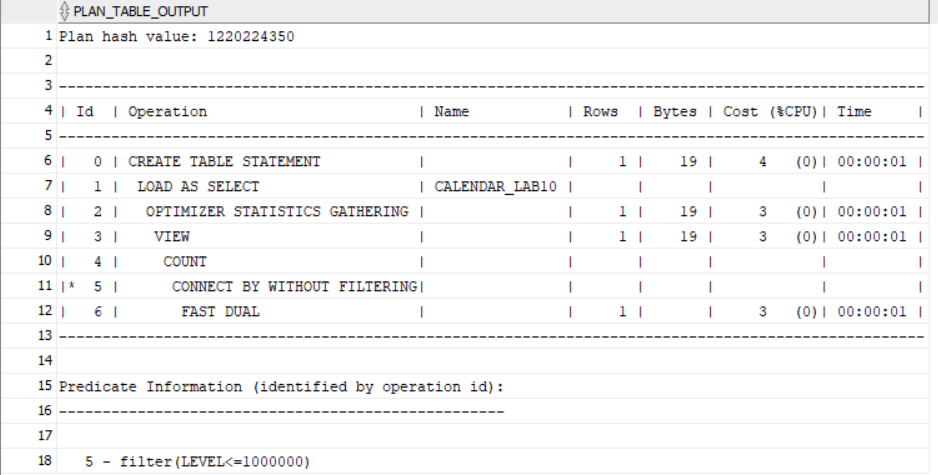
Execution Plan with parallel

Как следует из Execution Plan-ов, Oracle для больших выборок может сам использовать параллельные вычисления т.к. они почти идентичны.

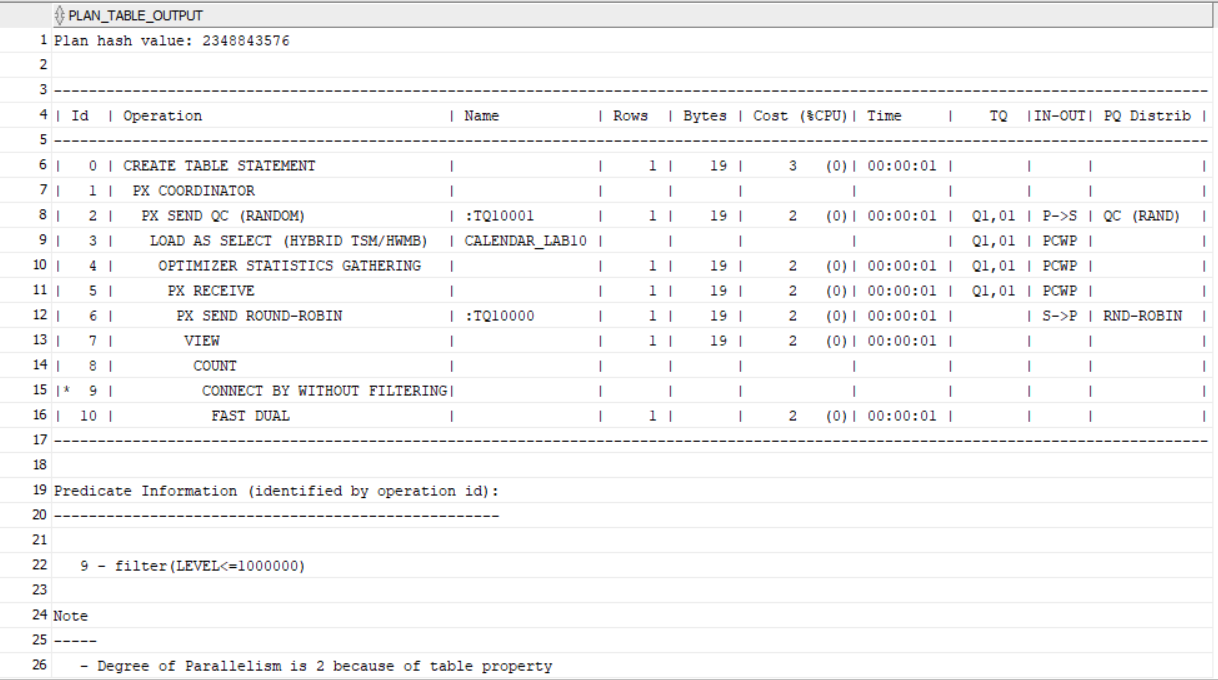
****

**Вывод:** parallel execution на выборке в 1 млн сток увеличил производительность на 5,4% при сложном запросе.

**TASK\_3** Parallel DDL

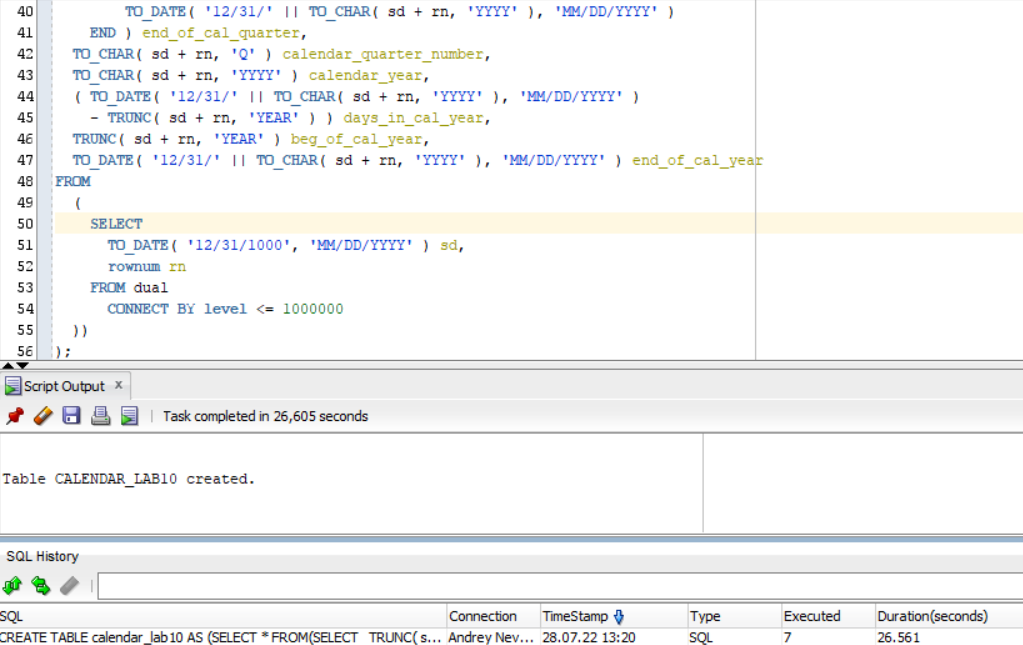


Execution Plan without parallel

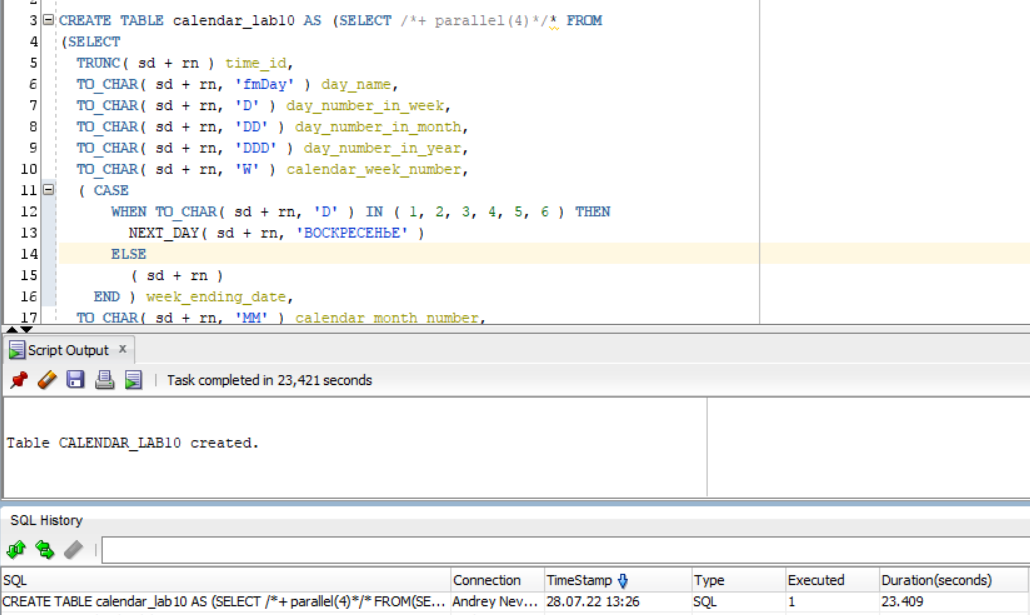
****

Execution Plan with parallel

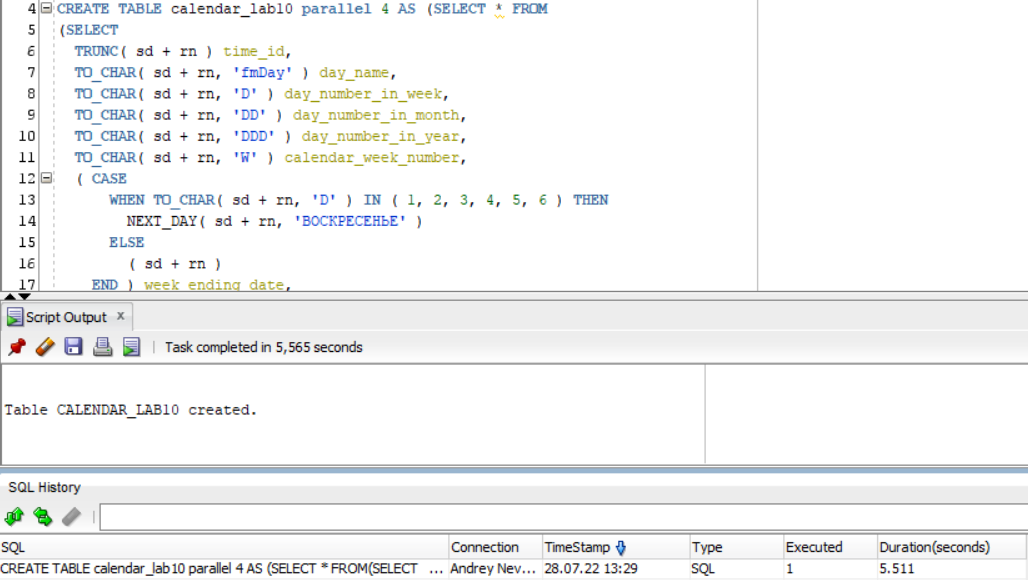
Как следует из Execution Plan-ов, Oracle для больших выборок может сам использовать параллельные вычисления т.к. они почти идентичны.

****

**Генерация таблицы без параллелизма**

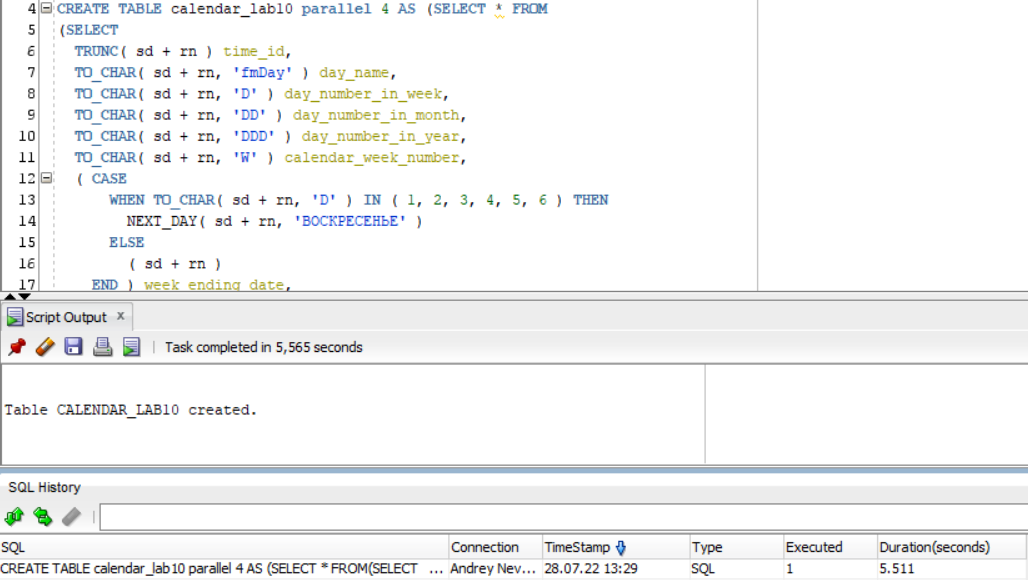
****

**Генерация таблицы используя хинт**

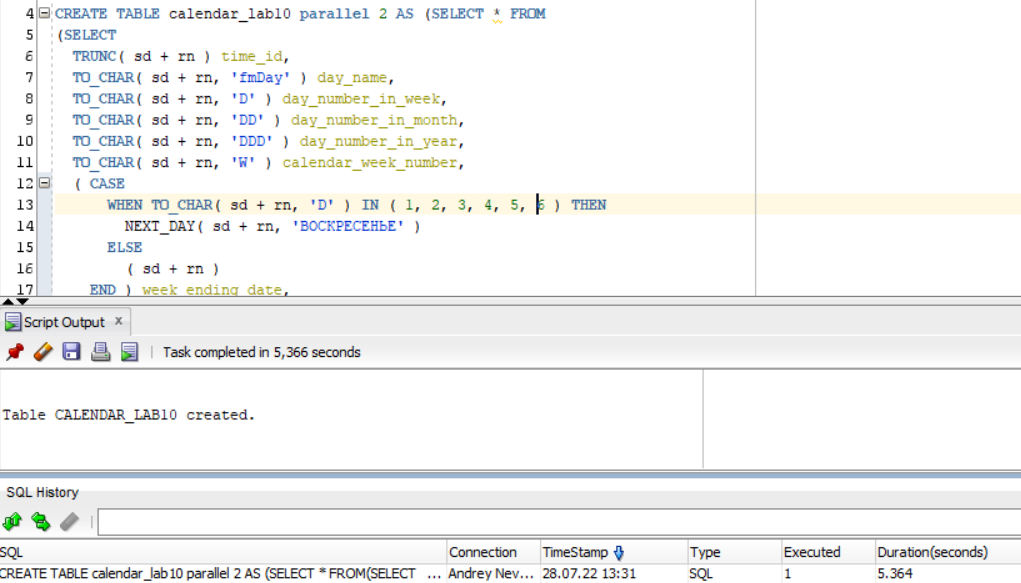
****

**Генерация таблицы используя строгое указание**

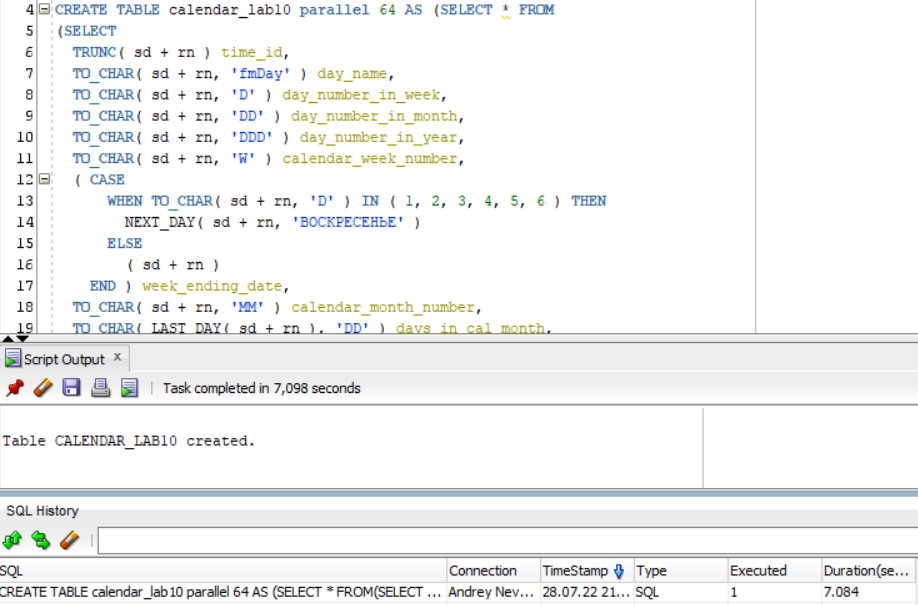
|  |  |  |
| --- | --- | --- |
| **No parallel** | **Parallel hint** | **Parallel comand** |
| **26.561 sec** | **23.409 sec** | **5.511 sec** |

****

**Генерация таблицы используя строгое указание ‘parallel 4'**

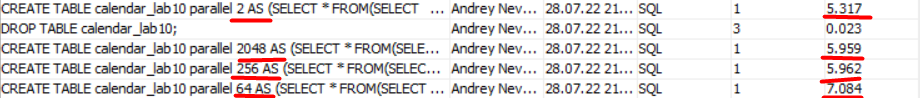
****

**Генерация таблицы используя строгое указание ‘parallel 2'**

****

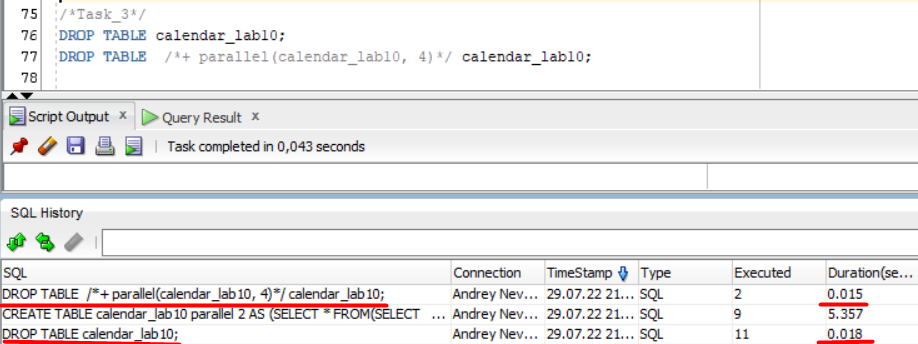
|  |  |  |
| --- | --- | --- |
| **parallel 64** | **parallel 4** | **parallel 2** |
| **7.084 sec** | **5.511 sec** | **5.364 sec** |

**Вывод:** параллелизм надо использовать с умом т.к. большее число потоков может только замедлить выполнение команды из-за ресурсоемкости процесса управления потоками.

****

Через некоторое время я проверил 64, 256, 2048 потоков. (Результаты эксперимента воспроизводимы для двух потоков время всегда в пределах 5.3, для 256 и 2048 в пределах 5,9, для 64 в пределах 7,05-7,15).

Как оказалось, **нет линейной зависимости между кол-вом потоков и временем выполнения**, но 2 потока для данной выборки оказались наиболее эффективными.

****

**Вывод:** функция drop table работает на 17% эффективнее для выборки из 1 миллиона записей. Возможно это из-за того, что oracle хранит ссылки на ячейки памяти с данными (если они заняты, то их не может перезаписать другое приложение) и drop таблицы просто делает память свободной для перезаписи (операция быстрая и одинаковая.

**SUMMARY “TASK\_1-3”**

1. **Я научился** использовать parallel операции с базой данных, использовать для этого метод хинтования и метод строгого декларирования и изменять

их эффективность.

1. **Я сделал** множество выводов на основе экспериментальных наблюдений (

выводы записаны по ходу выполнения); Как следует из Execution Plan-ов, Oracle для больших выборок может сам использовать параллельные вычисления.

1. **Я понял**, принцип работы parallel операций и методологию их применения.

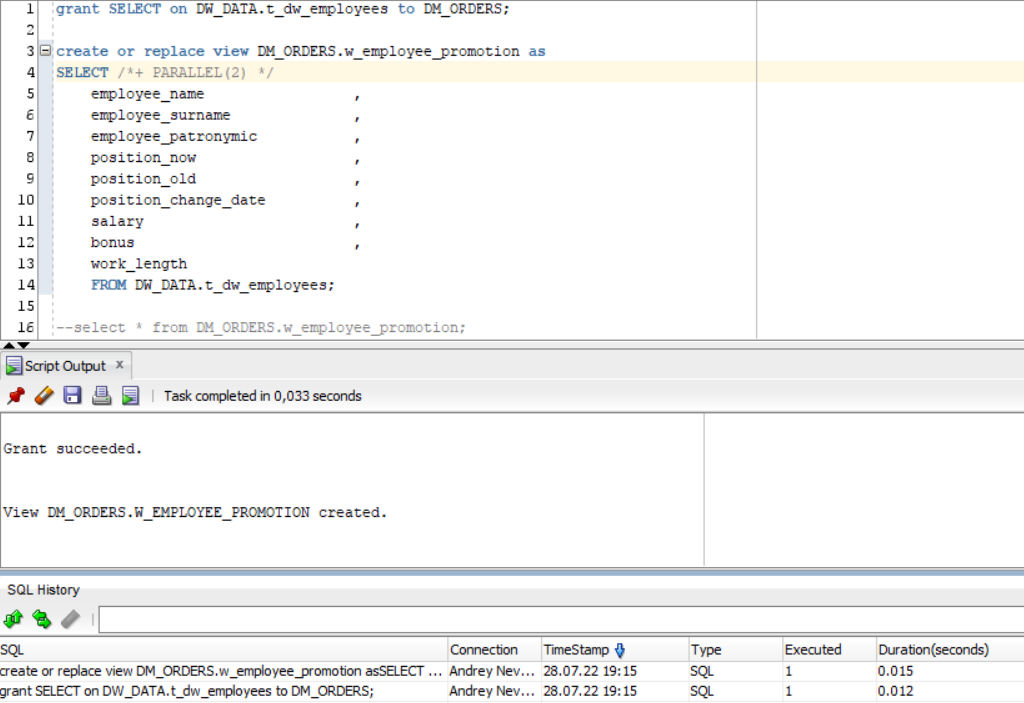
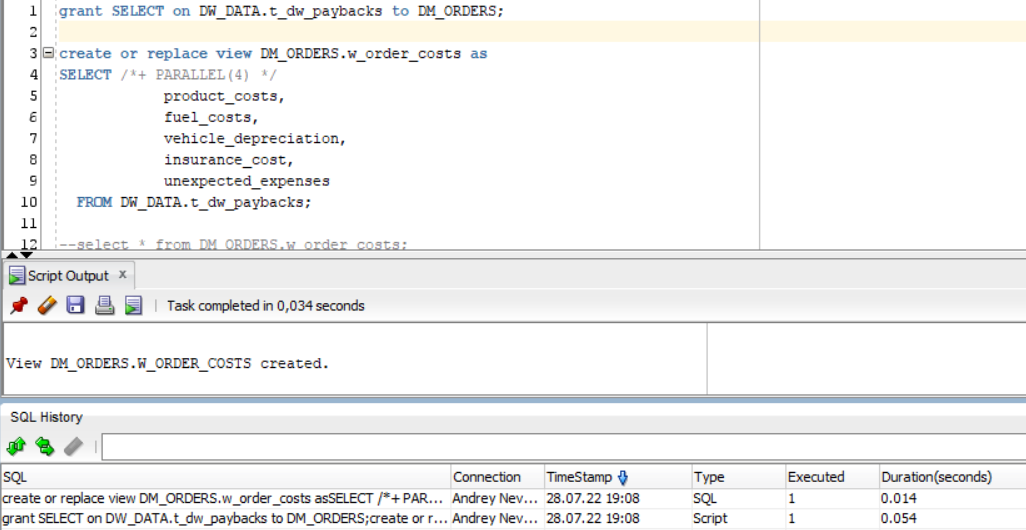
**FINAL SUMMARY**

Данная лабораторная работа …

1. … учит использовать параллельные вычисления и рассчитывать прирост производительность.
2. … на основе экспериментальных результатов дает понять принцип работы параллельных вычислений Oracle.
3. … показывает, что при параллельных вычислениях существуют нелинейные зависимости между производительностью и кол-вом потоков.
4. … позволяет оценить влияние параллельных вычислений на SELECT, DML и DDL операции отдельно.

**TASK\_4**

В моем бизнесе чаще всего изменяются dimention-ы: client, order и employee(т.к. компания большая). Поэтому логичнее всего использовать параллелизм в DW, CL и SA уровнях для обновления данных о них. Однако применять параллельные вычисления можно и в DM – data mart уровнях, при проектировании представлений (view), связанных с финансами. Т.к. они часто обновляются из-за большого числа транзакций, а бухгалтерам, маркетологам и т.д. необходима достоверная актуальная информация.



**Скрипт основного задания (На GitHub проблемы с отображением русских букв):**

-- DROP TABLE calendar\_lab10;

-- parallel 2

CREATE TABLE calendar\_lab10 parallel 2 AS (SELECT \* FROM

(SELECT

TRUNC( sd + rn ) time\_id,

TO\_CHAR( sd + rn, 'fmDay' ) day\_name,

TO\_CHAR( sd + rn, 'D' ) day\_number\_in\_week,

TO\_CHAR( sd + rn, 'DD' ) day\_number\_in\_month,

TO\_CHAR( sd + rn, 'DDD' ) day\_number\_in\_year,

TO\_CHAR( sd + rn, 'W' ) calendar\_week\_number,

( CASE

WHEN TO\_CHAR( sd + rn, 'D' ) IN ( 1, 2, 3, 4, 5, 6 ) THEN

NEXT\_DAY( sd + rn, 'ВОСКРЕСЕНЬЕ' )

ELSE

( sd + rn )

END ) week\_ending\_date,

TO\_CHAR( sd + rn, 'MM' ) calendar\_month\_number,

TO\_CHAR( LAST\_DAY( sd + rn ), 'DD' ) days\_in\_cal\_month,

LAST\_DAY( sd + rn ) end\_of\_cal\_month,

TO\_CHAR( sd + rn, 'FMMonth' ) calendar\_month\_name,

( ( CASE

WHEN TO\_CHAR( sd + rn, 'Q' ) = 1 THEN

TO\_DATE( '03/31/' || TO\_CHAR( sd + rn, 'YYYY' ), 'MM/DD/YYYY' )

WHEN TO\_CHAR( sd + rn, 'Q' ) = 2 THEN

TO\_DATE( '06/30/' || TO\_CHAR( sd + rn, 'YYYY' ), 'MM/DD/YYYY' )

WHEN TO\_CHAR( sd + rn, 'Q' ) = 3 THEN

TO\_DATE( '09/30/' || TO\_CHAR( sd + rn, 'YYYY' ), 'MM/DD/YYYY' )

WHEN TO\_CHAR( sd + rn, 'Q' ) = 4 THEN

TO\_DATE( '12/31/' || TO\_CHAR( sd + rn, 'YYYY' ), 'MM/DD/YYYY' )

END ) - TRUNC( sd + rn, 'Q' ) + 1 ) days\_in\_cal\_quarter,

TRUNC( sd + rn, 'Q' ) beg\_of\_cal\_quarter,

( CASE

WHEN TO\_CHAR( sd + rn, 'Q' ) = 1 THEN

TO\_DATE( '03/31/' || TO\_CHAR( sd + rn, 'YYYY' ), 'MM/DD/YYYY' )

WHEN TO\_CHAR( sd + rn, 'Q' ) = 2 THEN

TO\_DATE( '06/30/' || TO\_CHAR( sd + rn, 'YYYY' ), 'MM/DD/YYYY' )

WHEN TO\_CHAR( sd + rn, 'Q' ) = 3 THEN

TO\_DATE( '09/30/' || TO\_CHAR( sd + rn, 'YYYY' ), 'MM/DD/YYYY' )

WHEN TO\_CHAR( sd + rn, 'Q' ) = 4 THEN

TO\_DATE( '12/31/' || TO\_CHAR( sd + rn, 'YYYY' ), 'MM/DD/YYYY' )

END ) end\_of\_cal\_quarter,

TO\_CHAR( sd + rn, 'Q' ) calendar\_quarter\_number,

TO\_CHAR( sd + rn, 'YYYY' ) calendar\_year,

( TO\_DATE( '12/31/' || TO\_CHAR( sd + rn, 'YYYY' ), 'MM/DD/YYYY' )

- TRUNC( sd + rn, 'YEAR' ) ) days\_in\_cal\_year,

TRUNC( sd + rn, 'YEAR' ) beg\_of\_cal\_year,

TO\_DATE( '12/31/' || TO\_CHAR( sd + rn, 'YYYY' ), 'MM/DD/YYYY' ) end\_of\_cal\_year

FROM

(

SELECT

TO\_DATE( '12/31/1000', 'MM/DD/YYYY' ) sd,

rownum rn

FROM dual

CONNECT BY level <= 1000000

))

);

/\*Task\_1\*/

SELECT \* FROM calendar\_lab10

WHERE day\_name IN('Понедельник', 'Пятница', 'Воскресенье')

ORDER BY time\_id DESC;

SELECT /\*+ parallel(TIMES, 4)\*/\* FROM calendar\_lab10

WHERE day\_name IN('Понедельник', 'Пятница', 'Воскресенье')

ORDER BY time\_id DESC;

/\*Task\_2\*/

DELETE FROM calendar\_lab10

WHERE day\_name IN('Понедельник', 'Пятница', 'Воскресенье');

DELETE /\*+ parallel(TIMES, 4)\*/ FROM calendar\_lab10

WHERE day\_name IN('Понедельник', 'Пятница', 'Воскресенье');

/\*Task\_3\*/

DROP TABLE calendar\_lab10;

DROP TABLE /\*+ parallel(TIMES, 4)\*/ calendar\_lab10;