**Лабораторная работа №4**

**Невейков Андрей, 2022**

**/\*Task\_1\*/**

-- Step 1

CREATE TABLE t2 AS

SELECT TRUNC( rownum / 100 ) id, rpad( rownum,100 ) t\_pad

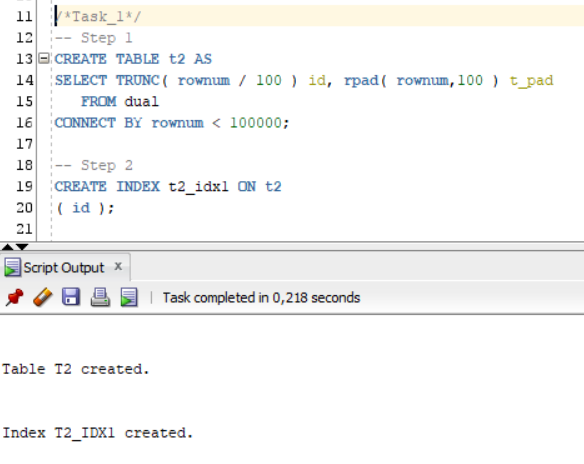
FROM dual

CONNECT BY rownum < 100000;

-- Step 2

CREATE INDEX t2\_idx1 ON t2

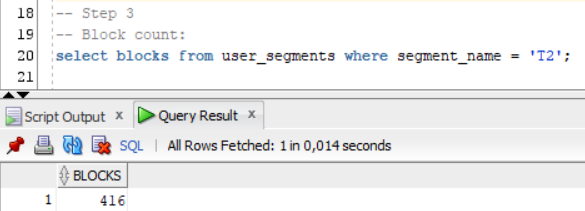
( id );



-- Step 3

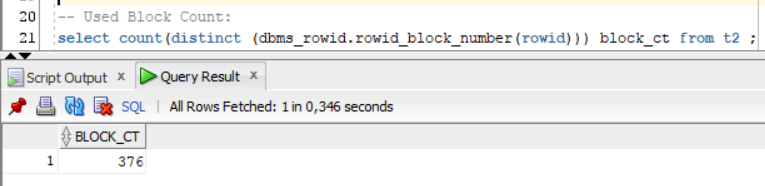
-- Block count:

select blocks from user\_segments where segment\_name = 'T2';



-- Used Block Count:

select count(distinct (dbms\_rowid.rowid\_block\_number(rowid))) block\_ct from t2 ;

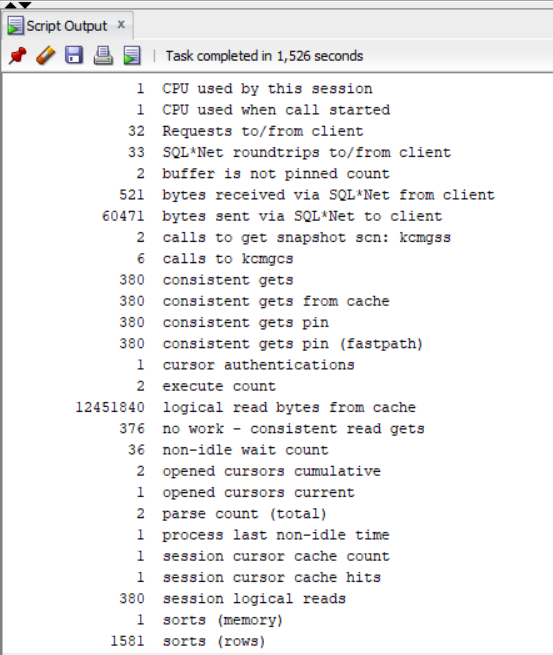


-- Explain Plan:

SET autotrace ON;

SELECT COUNT( \* )

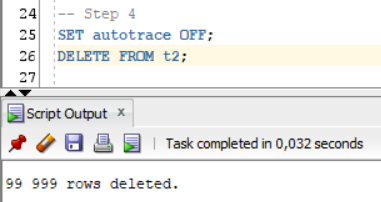
FROM t2 ;



-- Step 4

SET autotrace OFF;

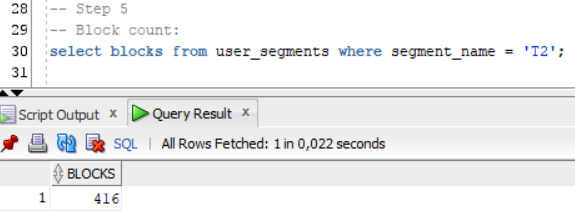
DELETE FROM t2;



-- Step 5

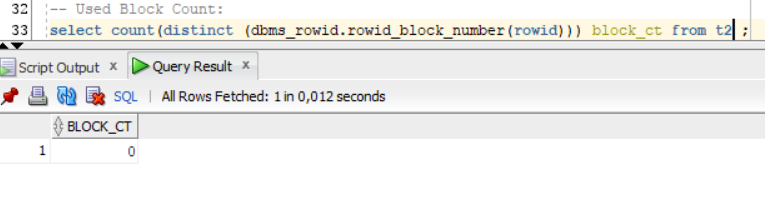
-- Block count:

select blocks from user\_segments where segment\_name = 'T2';



-- Used Block Count:

select count(distinct (dbms\_rowid.rowid\_block\_number(rowid))) block\_ct from t2 ;

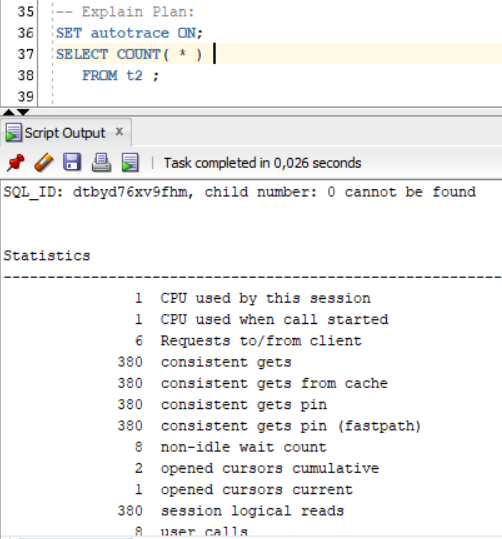


-- Explain Plan:

SET autotrace ON;

SELECT COUNT( \* )

FROM t2 ;



-- Step 6

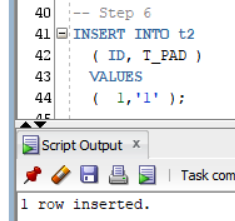
INSERT INTO t2

( ID, T\_PAD )

VALUES

( 1,'1' );

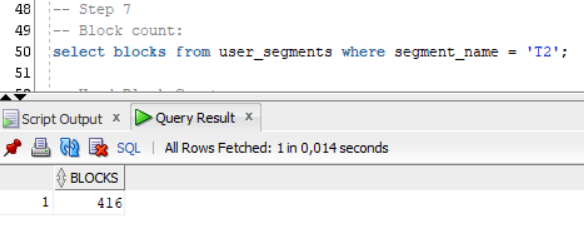
COMMIT;



-- Step 7

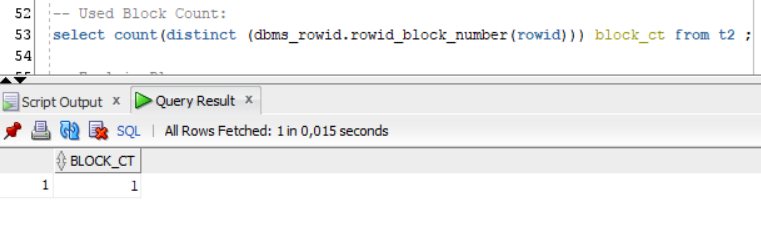
-- Block count:

select blocks from user\_segments where segment\_name = 'T2';



-- Used Block Count:

select count(distinct (dbms\_rowid.rowid\_block\_number(rowid))) block\_ct from t2 ;



-- Explain Plan:

SET autotrace ON;

SELECT COUNT( \* )

FROM t2 ;

-- Step 8

TRUNCATE TABLE t2;

-- Step 9

-- Block count:

select blocks from user\_segments where segment\_name = 'T2';

-- Used Block Count:

select count(distinct (dbms\_rowid.rowid\_block\_number(rowid))) block\_ct from t2 ;

-- Explain Plan:

SET autotrace ON;

SELECT COUNT( \* )

FROM t2 ;

Summary table with all result and text description of analyses this results.

*Consistent gets - это метрика представляет собой количество раз, когда за транзакцию запрашивалось последовательное чтение для блока.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Count of Blocks | Count of Used Blocks | Count of Rows | Consistent gets | Description |
| Step3 | 416 | 376 | 99999 | 380 | Была создана таблица t2 и заполнена данными. Данные для хранения используют память, соответственно, чем больше данных, тем больше памяти надо выделить. Выделение памяти – процесс медленный, поэтому мы можем настраивать на сколько блоков прирастет хранилище за каждое выделение памяти. Поэтому всего в базе 416 блока, а используем мы 376. (Суть: сэкономить время на операции выделения памяти, не каждый раз по блоку, а один раз несколько блоков). Чтобы выполнить запрос потребовалось 380 запросов последовательного чтения для блоков. |
| Step5 | 416 | 0 | 0 | 380 | В шаге 4 мы очистили таблицу при помощи конструкции DELETE FROM. Эта конструкция удаляет хранимую в блоках информацию, но не освобождает память, выделенную под блоки. Соответственно в таблице 416 блоков, но не один не используется. **Не смотря на отсутствие данных в блоках также потребовалось 380 чтений.** |
| Step7 | 416 | 1 | 1 | 380 | В шаге 6 была вставлена одна строка. Oracle не сократил общую выделенную память под базу данных, **для вывода статистики опять потребовалось 380 чтений.** |
| Step9 | 6 | 0 | 0 | 1 | В шаге 9 использовалась команда *TRUNCATE TABLE,* она не только очищает данные, но и освобождает память из неиспользуемых блоков. Их общее количество сократилось до стандартного состояния, на выполнение запроса потребовалось одно чтение. |

**SUMMARY “TASK\_1”**

1. **Я научился** оптимально использовать вычислительные ресурсы и память. Использовать различные команды очистки, в зависимости от поставленнойзадачи.
2. **Я узнал**, что команды очистки могут освобождать память полностью, а могут только удалять хранимые данные.
3. **Я понял** для каких целей могут применяться различные команды (DELETE FROM – когда нам поступает примерно одинаковый объем информации за единицу времени, например: пишем логи за час, потом перегружаем их на более медленный носитель. Если каждый час кол-во логов примерно одинаково, то нет смысла использовать *TRUNCATE TABLE* т.к. все равно придется выделять такое же кол-во блоков. *TRUNCATE TABLE* используем когда неизвестно сколько данных появится после очистки или известно что их будет меньше, чем тех, что были изначально).

**/\*Task\_2\*/**

-- Step 1

CREATE TABLE t2 AS

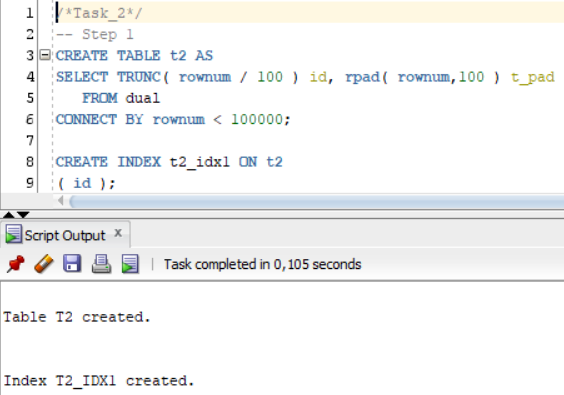
SELECT TRUNC( rownum / 100 ) id, rpad( rownum,100 ) t\_pad

FROM dual

CONNECT BY rownum < 100000;

CREATE INDEX t2\_idx1 ON t2

( id );



-- Step 2

CREATE TABLE t1 AS

SELECT MOD( rownum, 100 ) id, rpad( rownum,100 ) t\_pad

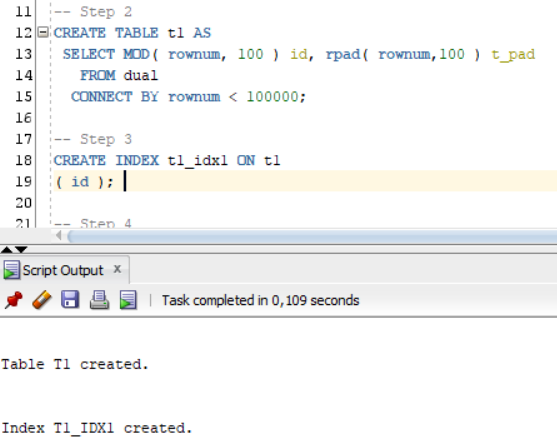
FROM dual

CONNECT BY rownum < 100000;

-- Step 3

CREATE INDEX t1\_idx1 ON t1

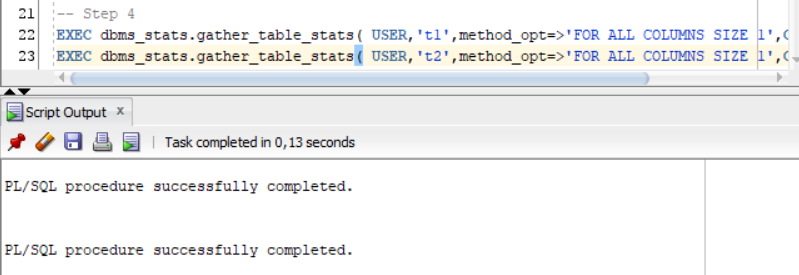
( id );



-- Step 4

EXEC dbms\_stats.gather\_table\_stats( USER,'t1',method\_opt=>'FOR ALL COLUMNS SIZE 1',CASCADE=>TRUE );

EXEC dbms\_stats.gather\_table\_stats( USER,'t2',method\_opt=>'FOR ALL COLUMNS SIZE 1',CASCADE=>TRUE );



-- Step 5

SELECT t.table\_name||'.'||i.index\_name idx\_name,

i.clustering\_factor,

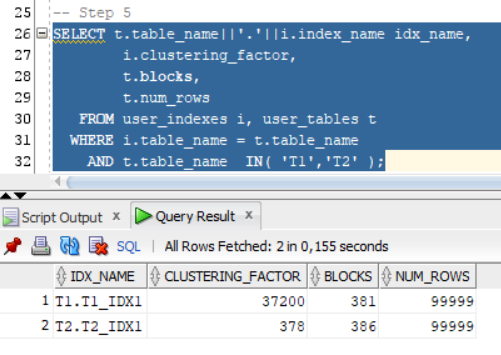
t.blocks,

t.num\_rows

FROM user\_indexes i, user\_tables t

WHERE i.table\_name = t.table\_name

AND t.table\_name IN( 'T1','T2' );



**Итоги “TASK\_2”**

Коэффициент кластеризации — это мера упорядоченности индекса по сравнению с таблицей, на которой он основан. Он используется для проверки стоимости поиска в таблице после доступа к индексу (умножение коэффициента кластеризации на селективность индекса дает стоимость операции).

Значения отличаются т.к. таблицы t1 и t2 были созданы по-разному.

CREATE TABLE t1 AS

SELECT **MOD( rownum, 100 )** id, rpad( rownum,100 ) t\_pad

FROM dual

CONNECT BY rownum < 100000;

CREATE TABLE t2 AS

SELECT **TRUNC( rownum / 100 )** id, rpad( rownum,100 ) t\_pad

FROM dual

CONNECT BY rownum < 100000;

TRUNC - возвращает число, усеченное до определенного количества знаков после запятой.

MOD - возвращает остаток от деления числа на другое число.

Таким образом получилось, что в таблице t2 все значения уникальны т.к. rownum всегда изменяется, а в таблице t1 есть ограниченное количество возможных остатков от деления (т.к. делим на 100, всего 99 вариантов). Значит индексы не уникальны т.к. в таблице 99999 записей. Соответственно в таблице t1 селективность больше и при умножении на коэффициент кластеризации мы получим большее число (больше стоимость поиска в таблице).

Таким образом selective performance лучше у индекса из таблицы t2.

**SUMMARY “TASK\_2”**

1. **Я научился** создавать индексы для оптимального поиска, оценивать

стоимость поиска.

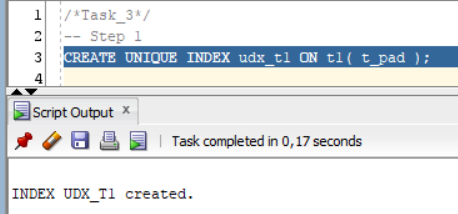
1. **Я узнал**, что такое коэффициент кластеризации, селективность и как на них влияет способ создания индекса.
2. **Я понял,** что для уменьшения стоимости поиска нужно делать индексы

уникальными.

**/\*Task\_3\*/**

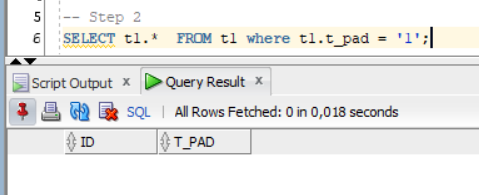
-- Step 1

CREATE UNIQUE INDEX udx\_t1 ON t1( t\_pad );



-- Step 2

SELECT t1.\* FROM t1 where t1.t\_pad = '1';

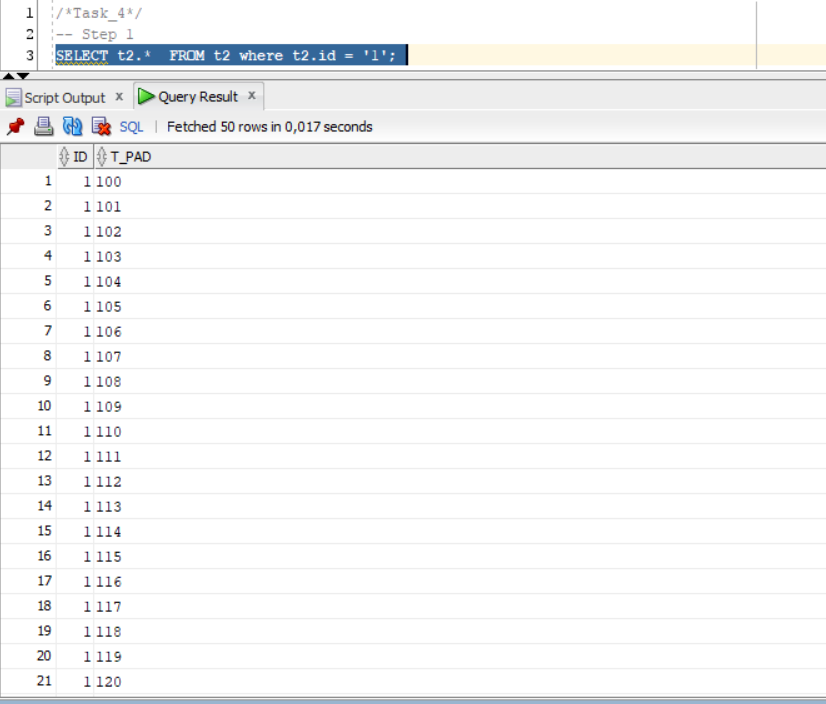


В данном случае выполняется поиск по значению не индексного поля, поэтому oracle должен проверить все блоки на соответствие. Процесс: oracle читает блок, если там значение поля, подходящее условию, потом так по всем блокам.

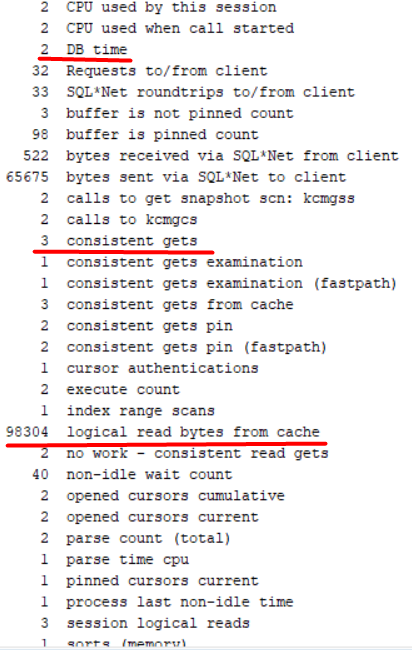
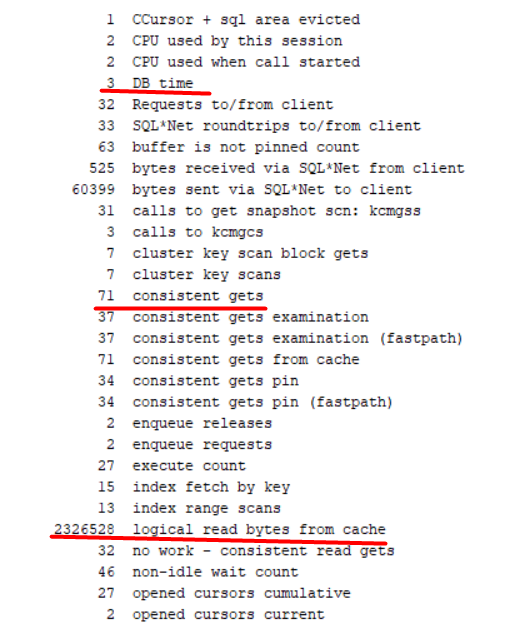
**/\*Task\_4\*/**

-- Step 1

SELECT t2.\* FROM t2 where t2.id = '1';



В данном случае выполняется поиск по значению индексного поля, поэтому oracle очень быстро находит нужный id в balance tree и выводит нужные результаты.



Применив команду “SET autotrace ON;” и выполнив запросы еще раз можно

увидеть, что select из четвертого задания потребовал значительно меньше

ресурсов.

**SUMMARY “TASK\_3-4”**

1. **Я научился** выполнять поиск по значению индексного/не индексного полей.
2. **Я узнал**, что поиск по индексному полю быстрее.
3. **Я понял** почему поиск по индексному полю быстрее (индексы находятся в

balance tree).

**/\*Task\_5\*/**

-- Step 1

CREATE TABLE employees AS

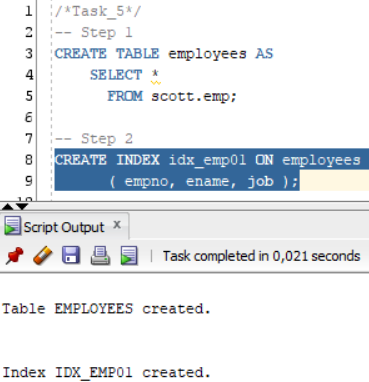
SELECT \*

FROM scott.emp;

-- Step 2

CREATE INDEX idx\_emp01 ON employees

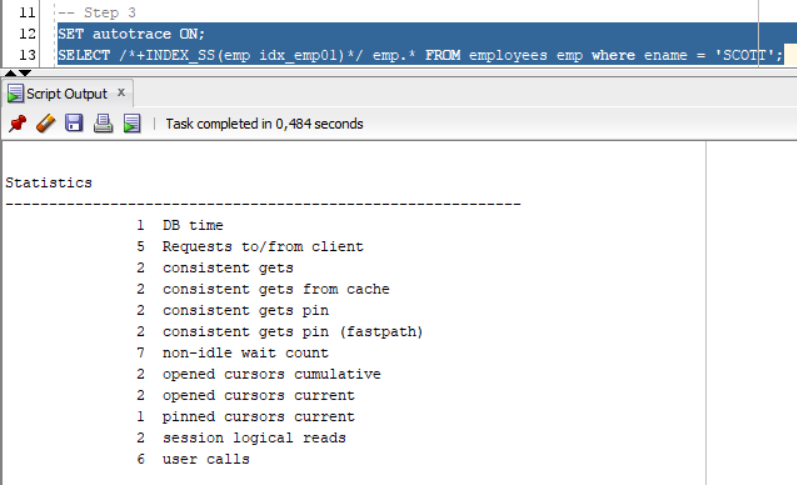
( empno, ename, job );



-- Step 3

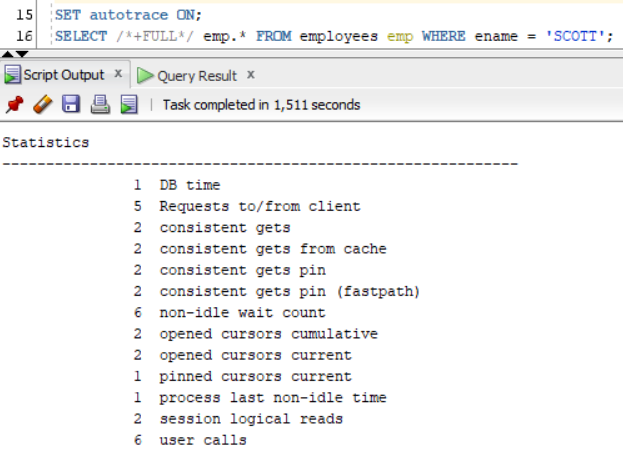
SET autotrace ON;

SELECT /\*+INDEX\_SS(emp idx\_emp01)\*/ emp.\* FROM employees emp where ename = 'SCOTT';



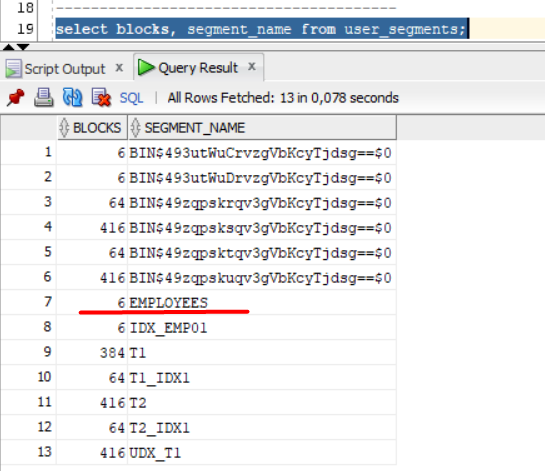
SET autotrace ON;

SELECT /\*+FULL\*/ emp.\* FROM employees emp WHERE ename = 'SCOTT';



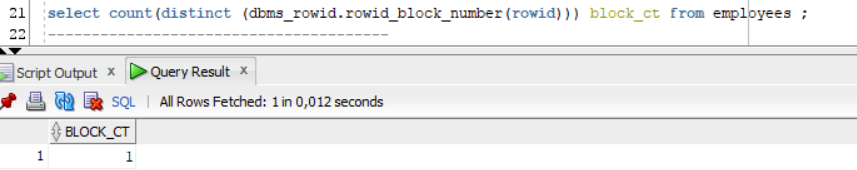
---------------------------------------

select blocks, segment\_name from user\_segments;



---------------------------------------

select count(distinct (dbms\_rowid.rowid\_block\_number(rowid))) block\_ct from employees ;

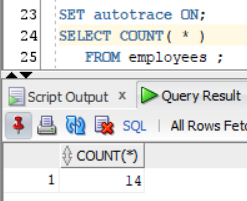


---------------------------------------

SET autotrace ON;

SELECT COUNT( \* )

FROM employees ;



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Count of Blocks | Count of Used Blocks | Count of Rows | Consistent gets | Description |
| 1 | 6 | 1 | 14 | 2 | Была использована подсказка /\*+INDEX\_SS(emp idx\_emp01)\*/ |
| 2 | 6 | 1 | 14 | 2 | Была использована подсказка  /\*+FULL\*/ |
| Статистика обоих запросов показала одинаковый результат, возможно из-за маленькой выборки данных. Метод использования подсказок, чтобы обеспечить оптимальный план выполнения теоретически на больших объемах данных должен быть эффективнее в первом случае т.к. выборка будет происходить сразу по сложному индексу. | | | | | |

В шаге 2 был создан сложный индекс на три колонки. Oracle воспринимает

индекс как единое целое.

**SUMMARY “TASK\_5”**

1. **Я научился** использовать подсказки для обеспечения оптимального плана выполнения.
2. **Я узнал**, можно из нескольких колонок создать один индекс.
3. **Я понял**, что, чем больше экспериментальная выборка данных, тем заметнее различия в затраченных на план выполнения ресурсах.

**FINAL SUMMARY**

Данная лабораторная работа …

1. … учит оптимально использовать вычислительные ресурсы и память. Использовать различные команды очистки, в зависимости от поставленной задачи.
2. … показывает различия в базовых методах создания индексов, показывает, что не оптимально созданный индекс существенно увеличит ресурсы, требуемые на выполнение запроса.
3. … знакомит с понятиями «коэффициент кластеризации», «селективность» и способам расчета «стоимости поиска».
4. … знакомит с использованием подсказок оптимизатора.
5. … знакомит с метрикой «Consistent gets»