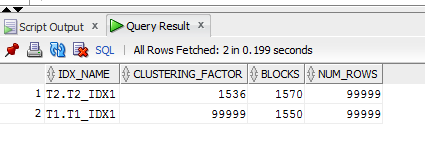
Ex1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Count of Blocks | Count of Used Blocks | Count of Rows | Consistent gets | Description |
| 1 | 1664 | 1536 | 99999 | 1539 | Create(all data in 1536 blocks) |
| 2 | 1664 | 0 | 0 | 1541 | Delete(no data, 0 blocks is used) |
| 3 | 1664 | 1 | 1 | 1541 | Insert(insert 1 row, it is situated in 1 block) |
| 4 | 8 | 0 | 0 | 3 | truncate |

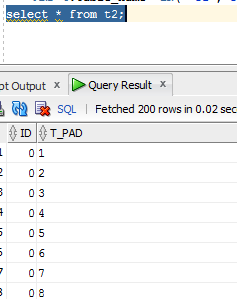
Ex2

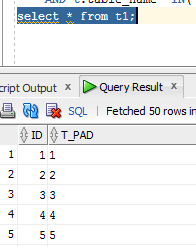
Оптимизатор при расчете стоимости индексного доступа к данным таблицы использует показатель под названием кластерный фактор (CF). Этот показатель выражает степень распределения записей в таблице относительно индексных ключей.

Кластерный фактор выражает, какое соотношение имеется между записями индекса и расположением записей в блоках данных. Фактор - это количество чтений другого блока данных при переходе от одной записи индекса к другой.

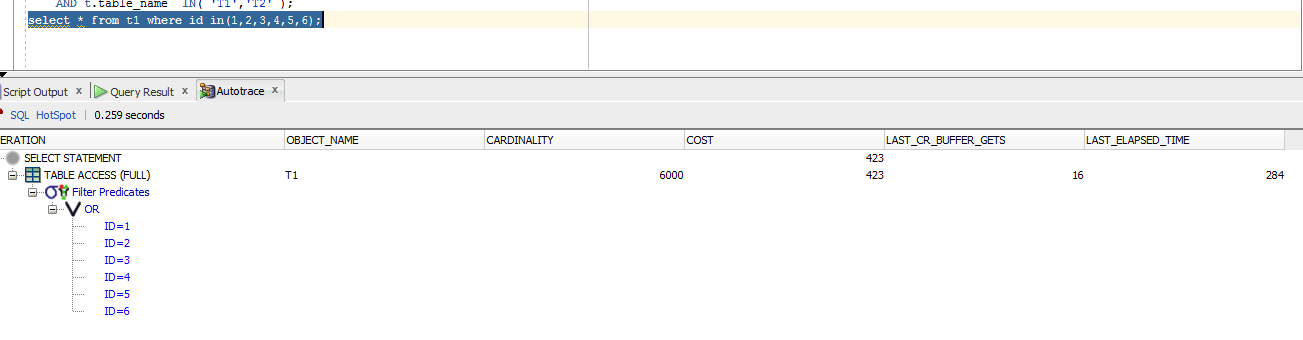


Мы имеем различные факторы из-за различного порядка строк в таблицах. Если значение CF близкое к количеству блоков таблицы, то это хороший индекс означающий, что записи таблицы отсортированы относительно значений индексных ключей( Что видно на примере таблицы 2). Если же записи распределены случайным образом, а значит, имеется большое CF значение, которое близко к количеству записей в таблице, то это плохой индекс(Что видно на примере таблицы 1).

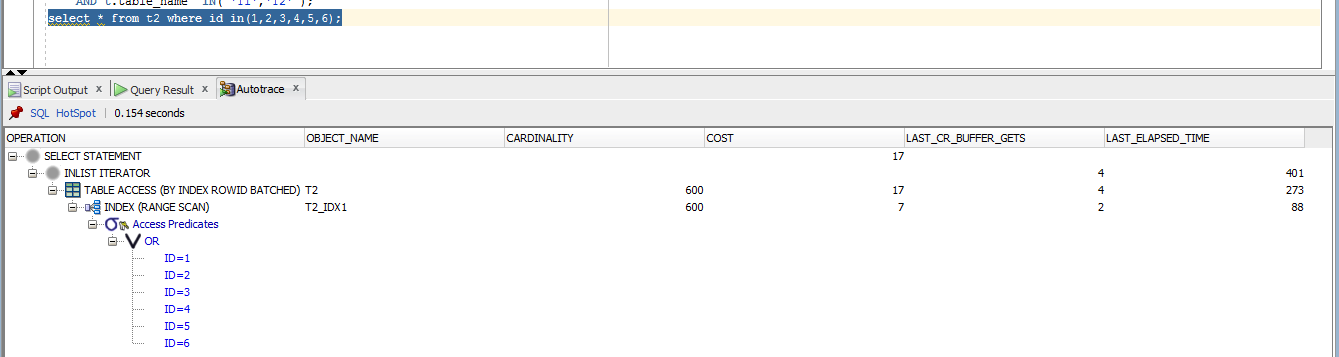




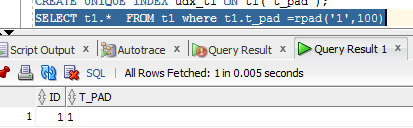
Т1:

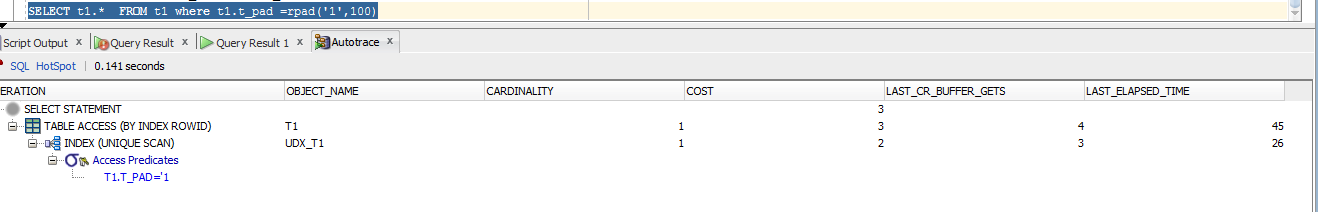


Т2:



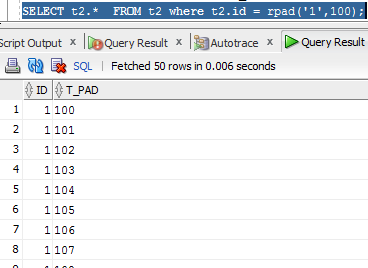
Ex3

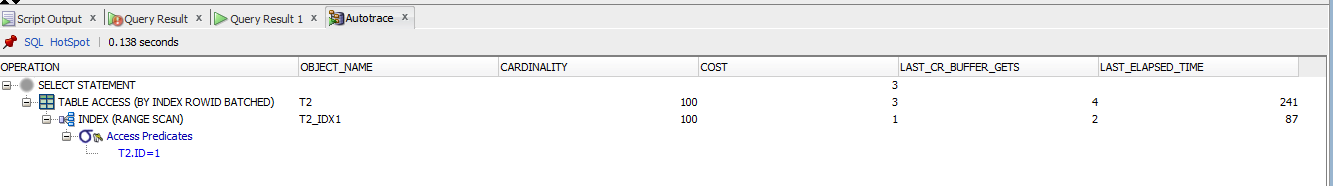




Так как индекс уникальный, то и ссылается он только на одно значение.Используя поиск по уникальному индексу мы сразу переходим в блок с нужной строкой по rowed

Ex4



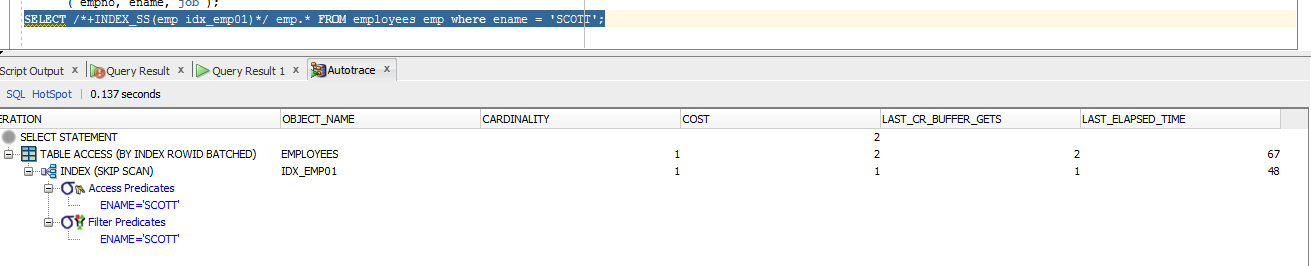


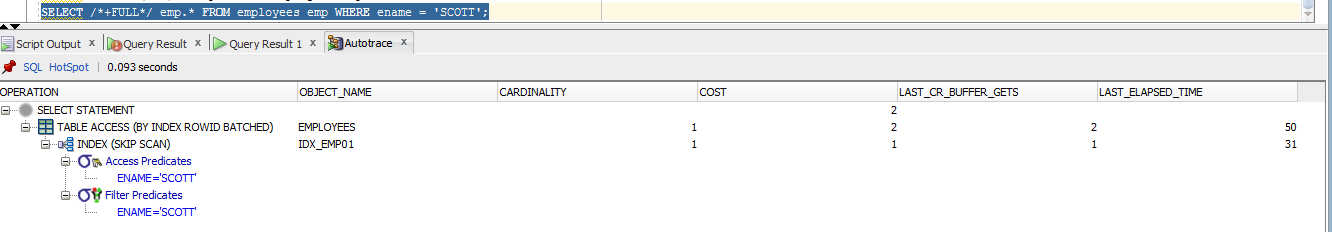
Range scan срабатывает, когда один индекс ссылается на несколько строк.

Ex5

Skip scan используется для составного индекса когда основной (первый) столбец в индексе не используется в WHERE

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Count of Blocks | Count of Used Blocks | Count of Rows | Consistent gets | Description |
| 1 | 8 | 1 | 1 | 2 |  |
| 2 | 8 | 1 | 1 | 2 |  |





Index\_ss Full

|  |  |  |  |
| --- | --- | --- | --- |
| buffer is not pinned count | 2 | buffer is not pinned count | 2 |
| bytes received via SQL\*Net from client | 531 | bytes received via SQL\*Net from client | 550 |
| bytes sent via SQL\*Net to client | 47850 | bytes sent via SQL\*Net to client | 47848 |
| calls to get snapshot scn: kcmgss | 2 | calls to get snapshot scn: kcmgss | 2 |
| calls to kcmgcs | 2 | calls to kcmgcs | 2 |
| consistent gets | 2 | consistent gets | 2 |
| consistent gets from cache | 2 | consistent gets from cache | 2 |
| consistent gets pin | 2 | consistent gets pin | 2 |
| consistent gets pin (fastpath) | 2 | consistent gets pin (fastpath) | 2 |
| CPU used by this session | 6 | CPU used by this session | 5 |
| CPU used when call started | 6 | CPU used when call started | 5 |
| cursor authentications | 1 | DB time | 7 |
| DB time | 5 | execute count | 2 |
| execute count | 2 | logical read bytes from cache | 16384 |
| logical read bytes from cache | 16384 | no work - consistent read gets | 2 |
| no work - consistent read gets | 2 | non-idle wait count | 25 |
| non-idle wait count | 25 | opened cursors cumulative | 2 |
| opened cursors cumulative | 2 | parse count (total) | 2 |
| parse count (total) | 2 | Requests to/from client | 25 |
| Requests to/from client | 25 | session cursor cache count | -2 |
| session cursor cache hits | 1 | session cursor cache hits | 2 |
| session logical reads | 2 | session logical reads | 2 |
| session pga memory | -196608 | sorts (memory) | 2 |
| sorts (memory) | 2 | sorts (rows) | 2356 |
| sorts (rows) | 2356 | SQL\*Net roundtrips to/from client | 25 |
| SQL\*Net roundtrips to/from client | 25 | table fetch by rowid | 1 |
| table fetch by rowid | 1 | user calls | 27 |
| user calls | 27 | workarea executions - optimal | 5 |
| workarea executions - optimal | 5 |  |  |