SQL teikuma izpildes plāns

Gints Plivna
gints.plivna@gmail.com

Kas es esmu?

- Pieredze darbā ar Oracle kopš 1997
- Oficiālais “amats” – sistēmanalītiķis Rix Technologies
- Oracle sertificēto kursu pasniedzējs Affecto Latvija (Mebius IT)
- Autors vienīgajam pastāvīgajam online informācijas avotam par datubāzēm latviešu valodā (http://datubazes.wordpress.com)
- LVOUG biedrs

SQL teikuma izpildes plāns

- Lēni strādā – **pārbaudi izpildes plānu**! (*execution plan*)

- Kā iegūt?
- Kā attēlot?
- Kā saprast?
- Kā uzlabot?

Kā iegūt?

Pieejams:

- DML teikumiem (Select, Insert, Update, Delete, Merge)
- Arī dažiem DDL teikumiem (Create table, Create index, Alter index .. rebuild)

Iegūstams:

Komanda	**Rezultāts**
explain plan for	plan_table
autotrace	
sql teikuma izpilde	vsql, vsql_plan
AWR aizpilde (automātiski)	dba_hist_sql_plan
SQL trace	treisfails

Kā attēlot?

Rezultāts	**Attēlo ar**
plan_table	dbms_xplan.display
vsql, vsql_plan	dbms_xplan.display_cursor
dba_hist_sql_plan	dbms_xplan.display_awr
treisfails	tkprof

```
SELECT * FROM table(dbms_xplan.display());
```

dbms_xplan.display_cursor rezultāts I

```
SQL> select * from dual where 1=2;
no rows selected
SQL> select * from table(dbms_xplan.display_cursor());
PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------
SQL_ID  c3nfy2116z2pb, child number 0
-------------------------------------
select * from dual where 1=2
Plan hash value: 3752461848
-----------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |       |       |     1 (100)|          |
|*  1 |  FILTER            |      |       |       |            |          |
|   2 |   TABLE ACCESS FULL| DUAL |     1 |     2 |     2   (0)| 00:00:01 |
-----------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   1 - filter(NULL IS NOT NULL)

SQL> select sql_text, sql_id, plan_hash_value, child_number from V$SQL
  2  where sql_id = 'c3nfy2116z2pb';
SQL_TEXT                       SQL_ID        PLAN_HASH_VALUE CHILD_NUMBER
------------------------------ ------------- --------------- --------------
select * from dual where 1=2   c3nfy2116z2pb      3752461848              0
```

Tālākie uzlabojumi

- v$sql_plan – **reālā** izpildes plāna soļi, bet ierakstu skaits (*Rows*) un laiks (*Time*) no **novērtējuma** (*estimation*)
- ir divi veidi kā iegūt **pilnu reālo info**:
  - hints gather_plan_statistics vai
  - parametram statistics_level jābūt ALL
- Informācija glabājas v$sql_plan_statistics
- Attēlo ar dbms_xplan.display_cursor(null, null, 'ALLSTATS LAST‘)

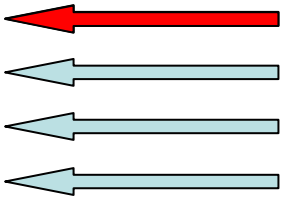
Pilna reālā informācija

```
SQL> alter session set statistics_level = ALL;
Session altered.
SQL> select count(*) from user_objects;
       COUNT(*)
--------------
            105
SQL> select * from table(dbms_xplan.display_cursor(null, null, 'ALLSTATS LAST'));
-------------------------------------------------------------------------------------------------
|Id|Operation                         |Name        |Starts|E-Rows|A-Rows |   A-Time   | Buffers |
-------------------------------------------------------------------------------------------------
| 1| SORT AGGREGATE                   |            |     1|     1|     1 |00:00:00.04 |     663 |
| 2|  VIEW                            |USER_OBJECTS|     1|  3209|   105 |00:00:00.04 |     663 |
| 3|   UNION-ALL                      |            |     1|      |   105 |00:00:00.04 |     663 |
|*4|    FILTER                        |            |     1|      |   105 |00:00:00.03 |     662 |
|*5|     TABLE ACCESS FULL            |OBJ$        |     1|  3391|   111 |00:00:00.02 |     610 |
|*6|     TABLE ACCESS BY INDEX ROWID|IND$          |    25|     1|    24 |00:00:00.01 |      52 |
|*7|      INDEX UNIQUE SCAN           |I_IND1      |    25|     1|    25 |00:00:00.01 |      27 |
|*8|    INDEX RANGE SCAN              |I_LINK1     |     1|     1|     0 |00:00:00.01 |       1 |
-------------------------------------------------------------------------------------------------
```

- Starts – cik reizes uzsākts darbs pie šī soļa
- E-Rows – paredzamais ierakstu skaits (*estimated rows*)
- A-Rows – reālais ierakstu skaits (*actual rows*)
- Buffers – cik datu bloki nolasīti

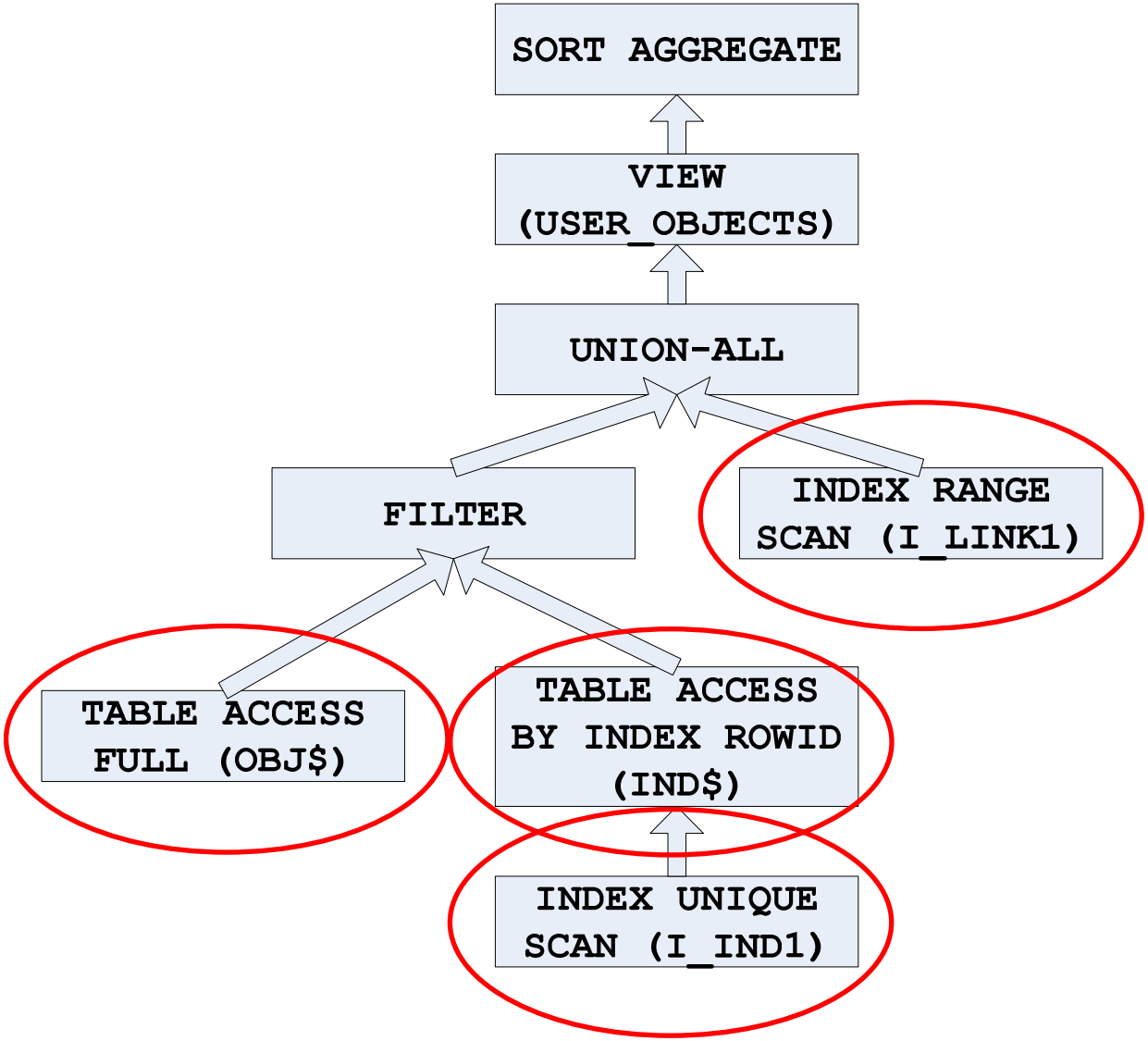
Kā saprast?

```
-----------------------------------------------------------
|Id|Operation                          |Name         |A-Rows |
-----------------------------------------------------------
| 1| SORT AGGREGATE                    |             |     1 |
| 2|  VIEW                             |USER_OBJECTS|   105 |
| 3|   UNION-ALL                       |             |   105 |
|*4|    FILTER                         |             |   105 |
|*5|     TABLE ACCESS FULL             |OBJ$         |   111 |
|*6|     TABLE ACCESS BY INDEX ROWID|IND$         |    24 |
|*7|      INDEX UNIQUE SCAN            |I_IND1       |    25 |
|*8|    INDEX RANGE SCAN               |I_LINK1      |     0 |
-----------------------------------------------------------
```



- Operācijas bez bērniem noteikti lasa datus no datubāzes (5, 7, 8)
- Operācijas ar bērniem iegūst datus no saviem bērniem (un varbūt lasa datus no datubāzes) (6)
- Pirmā operācija bez bērniem ir arī tā ar kuru izpilde tiek sākta (5)
- Lielākajai daļai operāciju ir ne vairāk kā 2 bērni

Vieglāk saprast kā koku



- Operācijas bez bērniem noteikti lasa datus no datubāzes (5, 7, 8)
- Operācijas ar bērniem iegūst datus no saviem bērniem (un varbūt lasa datus no datubāzes) (6)
- Pirmā operācija bez bērniem ir arī tā ar kuru izpilde tiek sākta (5)
- Lielākajai daļai operāciju ir ne vairāk kā 2 bērni

Biežāk izpildāmās operācijas I

- TABLE ACCESS
  - FULL
  - BY INDEX ROWID
  - CLUSTER
- INDEX
  - UNIQUE SCAN
  - RANGE SCAN
  - RANGE SCAN (MIN/MAX)
  - FAST FULL SCAN
  - FULL SCAN

Biežāk izpildāmās operācijas II

- NESTED LOOPS
- HASH JOIN
- MERGE JOIN
- MERGE JOIN CARTESIAN

- SORT
  - ORDER BY
  - AGGREGATE
  - GROUP BY
  - UNIQUE
- HASH
  - GROUP BY
  - UNIQUE

Biežāk izpildāmās operācijas III

- FILTER
- UNION-ALL
- PX [...]
- FAST DUAL
- CONNECT BY [...]
- COUNT STOPKEY
- WINDOW SORT

Savienojumi (*join*) – Nested loops

- Algoritms

```
katram ierakstam R1 ārējā tabulā A
    katram ierakstam R2 iekšējā tabulā B
        ja ierakstiem R1 un R2 izpildās savienojuma nosacījums
            tad atgriež {R1, R2}
```

A → *Ieraksts 1*

→ *Ieraksts 2*

→ *Ieraksts 3*

B → *Ieraksts 1*

→ *Ieraksts 2*

→ *Ieraksts 3*

Nested loops iezīmes

- Parasti noderīgs nelielu datu kopu apstrādē
- Iekšējās tabulas savienojošos ierakstus vēlams atlasīt neizmantojot TABLE ACCESS FULL
- Atbalsta visus savienojumu nosacījumus (vienāds, lielāks, mazāks, nevienādības utt.)
- No ātrdarbības viedokļa ir būtiski, kura tabula ārējā, kura iekšējā
  - Bet šo lēmumu atstājiet optimizatora ziņā, ja vien nav ļoti speciāli iemesli, kāpēc tā nedarīt

Nested loops ārējās/iekšējās tabulas ilustrācija

```
TABLE a (a number, d number)

100 ieraksti

SELECT /*+ use_nl (a b)
   ordered*/ *
FROM a, b
WHERE a = b;

Operation                      | Name
-------------------------------------
SELECT STATEMENT               |
 TABLE ACCESS BY INDEX ROWID| B
  NESTED LOOPS                 |
   TABLE ACCESS FULL           | A
   INDEX RANGE SCAN            | B_IDX

33 consistent gets
100 rows processed
```

```
TABLE b (b number, d number)
INDEX b_idx (b)
100 ieraksti

SELECT /*+ use_nl (a b)
   ordered*/ *
FROM b, a
WHERE a = b;

Operation             | Name
-----------------------------
SELECT STATEMENT      |
 NESTED LOOPS         |
  TABLE ACCESS FULL| B
  TABLE ACCESS FULL| A

317 consistent gets
100 rows processed
```

Tas nenozīmē, ka FROM klauzā ir svarīga tabulu secība!!

Savienojumi (*join*) – Sort merge join

- Algoritms (1:n gadījumam)

```
sakārto tabulas A ierakstus pēc savienojuma kolonas(-ām)
sakārto tabulas B ierakstus pēc savienojuma kolonas(-ām)
iegūst pirmo ierakstu R1 no A
iegūst pirmo ierakstu R1 no B
kamēr nav sasniegtas A vai B beigas
    ja ierakstiem R1 un R2 izpildās savienojuma nosacījums
        tad atgriež {R1, R2}
        iegūst nākošo ierakstu R2 no B
    citādi ja R1 < R2
        tad iegūst nākošo ierakstu R1 no A
    citādi
        tad iegūst nākošo ierakstu R2 no B
```

Sort merge join iezīmes

- Abām tabulām jābūt sakārtotām (izpildot kārtošanu vai jau iepriekš sakārtotas)
- Parasti noderīgs, ja datu kopas jau sakārtotas iepriekš vai hešsavienojums nav iespējams
- Atbalsta daudzus savienojumu nosacījumus (=, <, >, <=, >=, between), bet ne <>
- No ātrdarbības viedokļa nav svarīgi, kura tabula pirmā, kura otrā

Sort merge join ārējās/iekšējās tabulas ilustrācija

```
TABLE a (a number, d number)
100 ieraksti
```

```
SELECT /*+ use_merge (a b)
   ordered*/ *
FROM a, b
WHERE a = b;
```

```
Operation             | Name
-----------------------------------
SELECT STATEMENT      |
 MERGE JOIN           |
  SORT JOIN           |
   TABLE ACCESS FULL| A
  SORT JOIN           |
   TABLE ACCESS FULL| B
```

```
23376 consistent gets
2  sorts (memory)
100 rows processed
```

```
TABLE b (b number, d number)
10 000 000 ieraksti
```

```
SELECT /*+ use_merge (a b)
   ordered*/ *
FROM b, a
WHERE a = b;
```

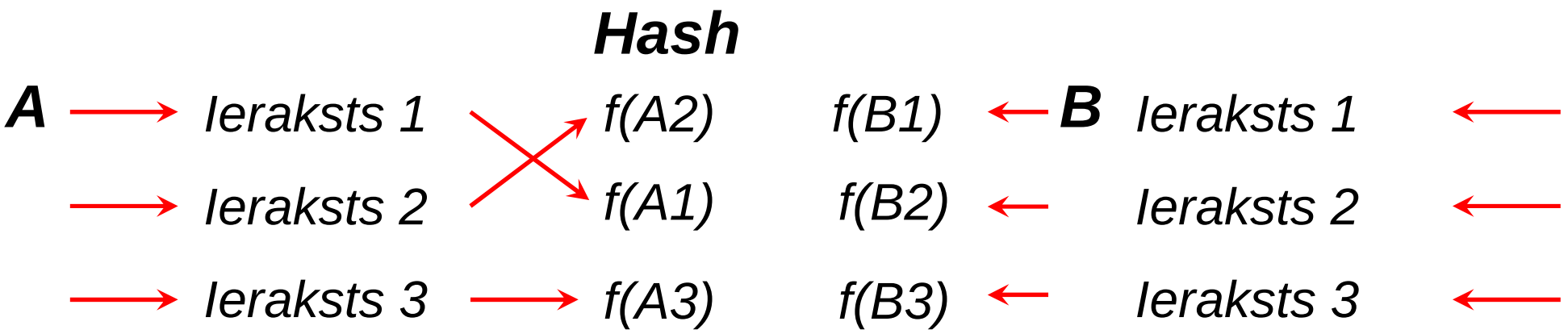
```
Operation             | Name
-----------------------------------
SELECT STATEMENT      |
 MERGE JOIN           |
  SORT JOIN           |
   TABLE ACCESS FULL| B
  SORT JOIN           |
   TABLE ACCESS FULL| A
```

```
23376 consistent gets
2  sorts (memory)
100 rows processed
```

Savienojumi (*join*) – *Hash join*

- Algoritms

```
katram ierakstam R1 kreisās puses tabulā A
    izrēķina hešfunkciju f(R1) savienojuma kolonai(-ām)
    saglabā ierakstu attiecīgajā heštabulas vietā
katram ierakstam R2 labās puses tabulā B
    izrēķina hešfunkciju f(R2) savienojuma kolonai(-ām)
    katram ierakstam R1, kam f(R1) = f(R2)
        ja ierakstiem R1 un R2 izpildās savienojuma nosacījums
            tad atgriež {R1, R2}
```



Hash join iezīmes

- Parasti noderīgs lielu datu kopu apstrādē vai ja uz tabulām nav indeksu un/vai nepieciešama visa tabulas skanēšana
- Heštabulai (ko izrēķina no kreisās puses tabulas) ļoti vēlams satilpt atmiņā
- Atbalsta tikai vienāds (=) nosacījumu
- No ātrdarbības viedokļa ir būtiski, kura tabula kreisā, kura labā
  - Bet šo lēmumu atstājiet optimizatora ziņā, ja vien nav ļoti speciāli iemesli, kāpēc tā nedarīt

Hash join ārējās/iekšējās tabulas ilustrācija

```
TABLE a (a number, d number)
100 ieraksti
```

```
SELECT /*+ use_hash (a b)
   ordered*/ *
FROM a, b
WHERE a = b;
```

```
Operation           |Name|Used-Mem
-------------------------------------
SELECT STATEMENT    |    |
 HASH JOIN          |    |1140K(0)
  TABLE ACCESS FULL| A   |
  TABLE ACCESS FULL| B   |

Elapsed: 00:00:01.68
100 rows processed
```

```
TABLE b (b number, d number)
10 000 000 ieraksti
```

```
SELECT /*+ use_hash (a b)
   ordered*/ *
FROM b, a
WHERE a = b;
```

```
Operation           |Name|Used-Mem
-------------------------------------
SELECT STATEMENT    |    |
 HASH JOIN          |    |217M(1)
  TABLE ACCESS FULL| B   |
  TABLE ACCESS FULL| A   |

Elapsed: 00:00:16.11
100 rows processed
```

Tas nenozīmē, ka FROM klauzā ir svarīga tabulu secība!!

Kā uzlabot? I

- Tieši neiespaidojot izpildes plānu:
  - SQL teikumā pēc iespējas tabulas datus atlasīt tikai vienreiz
  - Sakolekcionēt reprezentatīvas statistikas;
  - Pārbaudīt vai aptuveni sakrīt CBO paredzētais un reāli atgrieztais ierakstu skaits katrā solī

Kā uzlabot? II

- Tieši veicot izmaiņas izpildes plānā:
  - Sākt datu atlasi ar tabulu, kas visvairāk ierobežo atlasāmo datu apjomu
  - Izvēlēties noderīgāko datu pieejas metodi:
    - Dažas operācijas sāk atgriezt datus neizpildot visu operāciju līdz galam
      - Piemēram nested loops, table access full
    - Dažām operācijām nepieciešams liels darbs pirms atgriež pirmo ierakstu
      - Piemēram sort order by, merge join
  - Izvēlēties noderīgāko savienojumu (*join*) mehānismu
  - Eksperimentēt!

Saistītie informācijas avoti

- http://blog.tanelpoder.com/2009/03/14/new-presentation-slides-plus-aot-seminars-in-hong-kong-and-dallas/
- http://tkyte.blogspot.com/2007/04/when-explanation-doesn-sound-quite.html
- Oracle dokumentācija - Performance Tuning Guide => The Query Optimizer
- PL/SQL Packages and Types Reference => dbms_xplan
- **Šī prezentācija** (un arī citas interesantas lietas) - http://datubazes.wordpress.com