Optimalizace dotazu

(Ovlivnění) optimalizace dotazu v SŘBD Oracle

Upozornění!

Než začnete pracovat na příkladech, zadejte nejdříve tyto příkazy :

```
CREATE TABLE predmety  AS SELECT DISTINCT * FROM predmety;
CREATE TABLE rozvrh    AS SELECT DISTINCT * FROM rozvrh;
CREATE TABLE mistnosti AS SELECT DISTINCT * FROM mistnosti;
```

Během celého cvičení se budeme snažit ovlivňovat optimalizaci vyhodnocování tohto dotazu v SŘBD Oracle:

```
SELECT p.NAZEV,
       m.ZKRATKA,
       r.SEMESTR,
       r.DEN,
       r.H_OD,
       r.H_DO,
       p.KATEDRA,
       p.ZKRATKA
  FROM predmety p,
       mistnosti m,
       rozvrh r
 WHERE p.KREDITY > 2
   AND r.ID_PREDM = p.ID
   AND r.ID_MISTN = m.ID;
```

SŘBD Oracle již sám používá optimalizačních technik pro vyhodnocení jakéhokoliv dotazu. Těmi technikami jsou **RBO** (tj. *Rule Based Optimization*) a **CBO** (*Cost Based Optimization*). Od verze 9i SŘBD Oracle preferuje cenově založený optimalizátor.

Abychom mohli zjistit, jak ve skutečnosti daná optimalizace vyhodnocení dotazu funguje, potřebujeme vytvořit tabulku `PLAN_TABLE` (podle skriptu *utlxplan.sql*), kam optimalizátor ukládá své vítězné plány právě vyhodnocovaného dotazu.

```
CREATE TABLE plan_table (
        statement_id       varchar2(30),
        plan_id            number,
        timestamp          date,
        remarks            varchar2(4000),
        operation          varchar2(30),
        options            varchar2(255),
        object_node        varchar2(128),
        object_owner       varchar2(30),
        object_name        varchar2(30),
        object_alias       varchar2(65),
        object_instance    numeric,
        object_type        varchar2(30),
        optimizer          varchar2(255),
        search_columns     number,
        id                 numeric,
        parent_id          numeric,
        depth              numeric,
        position           numeric,
        cost               numeric,
        cardinality        numeric,
        bytes              numeric,
        other_tag          varchar2(255),
        partition_start    varchar2(255),
        partition_stop     varchar2(255),
        partition_id       numeric,
        other              long,
        distribution       varchar2(30),
        cpu_cost           numeric,
        io_cost            numeric,
        temp_space         numeric,
        access_predicates  varchar2(4000),
        filter_predicates  varchar2(4000),
        projection         varchar2(4000),
        time               numeric,
        qblock_name        varchar2(30),
        other_xml          clob
);
```

Pro vysvětlení (tj. zjištění optimálního plánu) vyhodnocení dotazu použijeme příkaz `EXPLAIN PLAN`:

```
EXPLAIN PLAN FOR
SELECT p.NAZEV,
       m.ZKRATKA,
       r.SEMESTR,
       r.DEN,
       r.H_OD,
       r.H_DO,
       p.KATEDRA,
       p.ZKRATKA
  FROM predmety p,
       mistnosti m,
       rozvrh r
 WHERE p.KREDITY > 2
   AND r.ID_PREDM = p.ID
   AND r.ID_MISTN = m.ID;
```

Vykonáním příkazu `EXPLAIN PLAN` se vítězný plán uloží do tabulky `PLAN_TABLE` v podobě několika záznamů.

```
SELECT * FROM plan_table;
```

Z příslušného výpisu obsahu tabulky moc nevyčteme, potřebuje k získání čitelné informace využít skript *utlxplp.sql*, využívající metody displ z balíku `DBMS_XPLAN`:

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

```
-------------------------------------------------------------------------------
| Id  | Operation           | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |           |     6 |   786 |    10  (10)| 00:00:01 |
|*  1 |  HASH JOIN          |           |     6 |   786 |    10  (10)| 00:00:01 |
|*  2 |   HASH JOIN         |           |     6 |   666 |     7  (15)| 00:00:01 |
|*  3 |    TABLE ACCESS FULL| PREDMETY  |     3 |   159 |     3   (0)| 00:00:01 |
|   4 |    TABLE ACCESS FULL| ROZVRH    |    10 |   580 |     3   (0)| 00:00:01 |
|   5 |   TABLE ACCESS FULL | MISTNOSTI |     4 |    80 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("R"."ID_MISTN"="M"."ID")
   2 - access("R"."ID_PREDM"="P"."ID")
   3 - filter("P"."KREDITY">2)
```

Z výše uvedné tabulky je vidět, že celá operace stála 10 jednotek a ke všem tabulkám se provádí plný přístup (tj. prochází se všechny záznamy). Ten plný přístup k tabulkám je příliš drahý, je vhodné pro atributy, které jednoznačně identifikují záznam tabulky vytvořit unikátní index:

```
CREATE UNIQUE INDEX mistnosti_pk ON mistnosti(id);
CREATE UNIQUE INDEX predmety_pk ON predmety(id);
```

Nyní musíme znovu spustit příkaz `EXPLAIN PLAN` pro náš dotaz, následovaný strukturovaným výstupem z tabulky `PLAN_TABLE` ukazující informace jen pro tuto optimalizaci.

```
EXPLAIN PLAN FOR
SELECT p.NAZEV,
       m.ZKRATKA,
       r.SEMESTR,
       r.DEN,
       r.H_OD,
       r.H_DO,
       p.KATEDRA,
       p.ZKRATKA
  FROM predmety p,
       mistnosti m,
       rozvrh r
 WHERE p.KREDITY > 2
   AND r.ID_PREDM = p.ID
   AND r.ID_MISTN = m.ID;

SELECT plan_table_output FROM table(dbms_xplan.display());

-----------------------------------------------------------------------------------------------
| Id  | Operation                     | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |             |     6 |   786 |    10  (20)| 00:00:01 |
|*  1 |  HASH JOIN                    |             |     6 |   786 |    10  (20)| 00:00:01 |
|   2 |   MERGE JOIN                  |             |     6 |   666 |     6  (17)| 00:00:01 |
|*  3 |    TABLE ACCESS BY INDEX ROWID| PREDMETY    |     3 |   159 |     2   (0)| 00:00:01 |
|   4 |     INDEX FULL SCAN           | PREDMETY_PK |     5 |       |     1   (0)| 00:00:01 |
|*  5 |    SORT JOIN                  |             |    10 |   580 |     4  (25)| 00:00:01 |
|   6 |     TABLE ACCESS FULL         | ROZVRH      |    10 |   580 |     3   (0)| 00:00:01 |
|   7 |   TABLE ACCESS FULL           | MISTNOSTI   |     4 |    80 |     3   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("R"."ID_MISTN"="M"."ID")
   3 - filter("P"."KREDITY">2)
   5 - access("R"."ID_PREDM"="P"."ID")
       filter("R"."ID_PREDM"="P"."ID")
```

Cena dotazu (vzhledem k malému množství dat) nesnížila, optimalizátor použil jen jeden z nabídnutých unikátních indexů. Stále nám vadí plný přístup do tabulek `ROZVRH` a `MISTNOSTI`. Zkusíme se tohoto přístupu zbavit vytvořením indexů pro atributy cizích klíčů:

```
CREATE INDEX rozvrh_index1 ON rozvrh(id_predm);
CREATE INDEX rozvrh_index2 ON rozvrh(id_mistn);

EXPLAIN PLAN FOR
SELECT p.NAZEV,
       m.ZKRATKA,
       r.SEMESTR,
       r.DEN,
       r.H_OD,
       r.H_DO,
       p.KATEDRA,
       p.ZKRATKA
  FROM predmety p,
       mistnosti m,
       rozvrh r
 WHERE p.KREDITY > 2
   AND r.ID_PREDM = p.ID
   AND r.ID_MISTN = m.ID;

SELECT plan_table_output FROM table(dbms_xplan.display());

-----------------------------------------------------------------------------------------------
| Id  | Operation                     | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |               |     6 |   786 |     9  (12)| 00:00:01 |
|*  1 |  HASH JOIN                    |               |     6 |   786 |     9  (12)| 00:00:01 |
|   2 |   NESTED LOOPS                |               |       |       |            |          |
|   3 |    NESTED LOOPS               |               |     6 |   666 |     5   (0)| 00:00:01 |
|*  4 |     TABLE ACCESS FULL         | PREDMETY      |     3 |   159 |     3   (0)| 00:00:01 |
|*  5 |     INDEX RANGE SCAN          | ROZVRH_INDEX1 |     3 |       |     0   (0)| 00:00:01 |
|   6 |    TABLE ACCESS BY INDEX ROWID| ROZVRH        |     2 |   116 |     1   (0)| 00:00:01 |
|   7 |   TABLE ACCESS FULL           | MISTNOSTI     |     4 |    80 |     3   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
```

```
   1 - access("R"."ID_MISTN"="M"."ID")
   4 - filter("P"."KREDITY">2)
   5 - access("R"."ID_PREDM"="P"."ID")
```

Plný přístup k tabulce `ROZVRH` zmizel, ale objevil se u zbývajících dvou. To je dáno zvolením jiného způsobu spojení tabulek (použita operace `NESTED LOOPS` místo `MERGE JOIN`), protože to vychází laciněji (stejě jako v předchozím případě). Pokud se podíváme na na třetí operaci, zjistíme že se jedná o podmínku selekce, která potřebuje také optimalizovat. Vhodný způsob je vytvořit index pro atribut `KREDITY` tabulky `PREDMETY`.

```
CREATE INDEX predmety_index1 ON predmety(kredity);

EXPLAIN PLAN FOR
SELECT p.NAZEV,
       m.ZKRATKA,
       r.SEMESTR,
       r.DEN,
       r.H_OD,
       r.H_DO,
       p.KATEDRA,
       p.ZKRATKA
  FROM predmety p,
       mistnosti m,
       rozvrh r
 WHERE p.KREDITY > 2
   AND r.ID_PREDM = p.ID
   AND r.ID_MISTN = m.ID;

SELECT plan_table_output FROM table(dbms_xplan.display());

-----------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name            | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                 |     6 |   786 |     8  (13)| 00:00:01 |
|*  1 |  HASH JOIN                    |                 |     6 |   786 |     8  (13)| 00:00:01 |
|   2 |   NESTED LOOPS                |                 |       |       |            |          |
|   3 |    NESTED LOOPS               |                 |     6 |   666 |     4   (0)| 00:00:01 |
|   4 |     TABLE ACCESS BY INDEX ROWID| PREDMETY        |     3 |   159 |     2   (0)| 00:00:01 |
|*  5 |      INDEX RANGE SCAN         | PREDMETY_INDEX1 |     3 |       |     1   (0)| 00:00:01 |
|*  6 |     INDEX RANGE SCAN          | ROZVRH_INDEX1   |     3 |       |     0   (0)| 00:00:01 |
|   7 |    TABLE ACCESS BY INDEX ROWID | ROZVRH          |     2 |   116 |     1   (0)| 00:00:01 |
|   8 |   TABLE ACCESS FULL           | MISTNOSTI       |     4 |    80 |     3   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("R"."ID_MISTN"="M"."ID")
   5 - access("P"."KREDITY">2)
   6 - access("R"."ID_PREDM"="P"."ID")
```

Jak je vidět, cena vyhodnocení dotazu klesla na 8. Dalším krokem, kterým můžeme ovlivnit optimalizaci je změna optimalizačního módu optimalizátoru SŘBD Oracle. Všechny realizované dotazy byly optimalizovány na nejlepší průchodnost. Je též možnost optimalizátor nastavit na nejrychlejší odezvu, tj. zkrátit čas vyhodnocení dotazu. K tomuto účelu změníme připojení příkazem `ALTER SESSION`.

```
ALTER SESSION SET optimizer_mode = first_rows_1;

EXPLAIN PLAN FOR
SELECT p.NAZEV,
       m.ZKRATKA,
       r.SEMESTR,
       r.DEN,
       r.H_OD,
       r.H_DO,
       p.KATEDRA,
       p.ZKRATKA
  FROM predmety p,
       mistnosti m,
       rozvrh r
 WHERE p.KREDITY > 2
   AND r.ID_PREDM = p.ID
   AND r.ID_MISTN = m.ID;

SELECT plan_table_output FROM table(dbms_xplan.display());

-----------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name            | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                 |     6 |   786 |     6   (0)| 00:00:01 |
|   1 |  NESTED LOOPS                 |                 |       |       |            |          |
|   2 |   NESTED LOOPS                |                 |     6 |   786 |     6   (0)| 00:00:01 |
|*  3 |    HASH JOIN                  |                 |     6 |   666 |     5  (20)| 00:00:01 |
|   4 |     TABLE ACCESS FULL         | ROZVRH          |    10 |   580 |     2   (0)| 00:00:01 |
|   5 |     TABLE ACCESS BY INDEX ROWID| PREDMETY        |     3 |   159 |     2   (0)| 00:00:01 |
|*  6 |      INDEX RANGE SCAN         | PREDMETY_INDEX1 |     3 |       |     1   (0)| 00:00:01 |
|*  7 |    INDEX UNIQUE SCAN          | MISTNOSTI_PK    |     1 |       |     0   (0)| 00:00:01 |
|   8 |   TABLE ACCESS BY INDEX ROWID | MISTNOSTI       |     1 |    20 |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   3 - access("R"."ID_PREDM"="P"."ID")
   6 - access("P"."KREDITY">2)
   7 - access("R"."ID_MISTN"="M"."ID")
```

Finální cena optimalizace našeho dotazu je 6, což téměř polovina ceny, se kterou jsem začínali.

CBO - Cost Based Optimizer

Nyní si ukážeme, jak lze ovlivnit výsledek metody CBO absencí resp. přítomností potřebných statistik pro oba módy optimalizátoru.

Nejvyšší průchodnost, absence statistik

```
ALTER SESSION SET optimizer_mode = all_rows;

ANALYZE TABLE mistnosti DELETE STATISTICS;
ANALYZE TABLE predmety DELETE STATISTICS;
ANALYZE TABLE rozvrh DELETE STATISTICS;

EXPLAIN PLAN FOR
SELECT p.NAZEV,
       m.ZKRATKA,
       r.SEMESTR,
       r.DEN,
       r.H_OD,
       r.H_DO,
       p.KATEDRA,
       p.ZKRATKA
  FROM predmety p,
       mistnosti m,
       rozvrh r
 WHERE p.KREDITY > 2
   AND r.ID_PREDM = p.ID
   AND r.ID_MISTN = m.ID;

SELECT plan_table_output FROM table(dbms_xplan.display());

--------------------------------------------------------------------------------
| Id  | Operation           | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |           |     6 |   786 |    10  (10)| 00:00:01 |
|*  1 |  HASH JOIN          |           |     6 |   786 |    10  (10)| 00:00:01 |
|*  2 |   HASH JOIN         |           |     6 |   666 |     7  (15)| 00:00:01 |
|*  3 |    TABLE ACCESS FULL| PREDMETY  |     3 |   159 |     3   (0)| 00:00:01 |
|   4 |    TABLE ACCESS FULL| ROZVRH    |    10 |   580 |     3   (0)| 00:00:01 |
|   5 |   TABLE ACCESS FULL | MISTNOSTI |     4 |    80 |     3   (0)| 00:00:01 |
--------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("R"."ID_MISTN"="M"."ID")
   2 - access("R"."ID_PREDM"="P"."ID")
   3 - filter("P"."KREDITY">2)
```

Nejvyšší průchodnost, přítomnost statistik

```
ALTER SESSION SET optimizer_mode = all_rows;

ANALYZE TABLE mistnosti COMPUTE STATISTICS;
ANALYZE TABLE predmety COMPUTE STATISTICS;
ANALYZE TABLE rozvrh COMPUTE STATISTICS;

EXPLAIN PLAN FOR
SELECT p.NAZEV,
       m.ZKRATKA,
       r.SEMESTR,
       r.DEN,
       r.H_OD,
       r.H_DO,
       p.KATEDRA,
       p.ZKRATKA
  FROM predmety p,
       mistnosti m,
       rozvrh r
 WHERE p.KREDITY > 2
   AND r.ID_PREDM = p.ID
   AND r.ID_MISTN = m.ID;

SELECT plan_table_output FROM table(dbms_xplan.display());

-----------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name            | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                 |     5 |   205 |     8  (25)| 00:00:01 |
|*  1 |  HASH JOIN                    |                 |     5 |   205 |     8  (25)| 00:00:01 |
|   2 |   MERGE JOIN                  |                 |     8 |   152 |     5  (20)| 00:00:01 |
|   3 |    TABLE ACCESS BY INDEX ROWID | MISTNOSTI       |     4 |    28 |     2   (0)| 00:00:01 |
|   4 |     INDEX FULL SCAN           | MISTNOSTI_PK    |     4 |       |     1   (0)| 00:00:01 |
|*  5 |    SORT JOIN                  |                 |     8 |    96 |     3  (34)| 00:00:01 |
|   6 |     TABLE ACCESS BY INDEX ROWID| ROZVRH          |     8 |    96 |     2   (0)| 00:00:01 |
|*  7 |      INDEX FULL SCAN          | ROZVRH_INDEX1   |     8 |       |     1   (0)| 00:00:01 |
|   8 |   TABLE ACCESS BY INDEX ROWID | PREDMETY        |     3 |    66 |     2   (0)| 00:00:01 |
|*  9 |    INDEX RANGE SCAN           | PREDMETY_INDEX1 |     3 |       |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("R"."ID_PREDM"="P"."ID")
   5 - access("R"."ID_MISTN"="M"."ID")
       filter("R"."ID_MISTN"="M"."ID")
   7 - filter("R"."ID_PREDM" IS NOT NULL)
   9 - access("P"."KREDITY">2)
```

Nejkratší doba odezvy, absence statistik

```
ALTER SESSION SET optimizer_mode = first_rows_1;
```

```
ANALYZE TABLE mistnosti DELETE STATISTICS;
ANALYZE TABLE predmety DELETE STATISTICS;
ANALYZE TABLE rozvrh DELETE STATISTICS;

EXPLAIN PLAN FOR
SELECT p.NAZEV,
       m.ZKRATKA,
       r.SEMESTR,
       r.DEN,
       r.H_OD,
       r.H_DO,
       p.KATEDRA,
       p.ZKRATKA
  FROM predmety p,
       mistnosti m,
       rozvrh r
 WHERE p.KREDITY > 2
   AND r.ID_PREDM = p.ID
   AND r.ID_MISTN = m.ID;

SELECT plan_table_output FROM table(dbms_xplan.display());

-----------------------------------------------------------------------------------------
| Id  | Operation                    | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |              |     6 |   786 |     8   (0)| 00:00:01 |
|   1 |  NESTED LOOPS                |              |       |       |            |          |
|   2 |   NESTED LOOPS               |              |     6 |   786 |     8   (0)| 00:00:01 |
|*  3 |    HASH JOIN                 |              |     6 |   666 |     7  (15)| 00:00:01 |
|   4 |     TABLE ACCESS FULL        | ROZVRH       |    10 |   580 |     3   (0)| 00:00:01 |
|*  5 |     TABLE ACCESS FULL        | PREDMETY     |     3 |   159 |     3   (0)| 00:00:01 |
|*  6 |    INDEX UNIQUE SCAN         | MISTNOSTI_PK |     1 |       |     0   (0)| 00:00:01 |
|   7 |   TABLE ACCESS BY INDEX ROWID| MISTNOSTI    |     1 |    20 |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   3 - access("R"."ID_PREDM"="P"."ID")
   5 - filter("P"."KREDITY">2)
   6 - access("R"."ID_MISTN"="M"."ID")
```

Nejkratší doba odezvy, přítomnost statistik

```
ALTER SESSION SET optimizer_mode = first_rows_1;

ANALYZE TABLE mistnosti COMPUTE STATISTICS;
ANALYZE TABLE predmety COMPUTE STATISTICS;
ANALYZE TABLE rozvrh COMPUTE STATISTICS;

EXPLAIN PLAN FOR
SELECT p.NAZEV,
       m.ZKRATKA,
       r.SEMESTR,
       r.DEN,
       r.H_OD,
       r.H_DO,
       p.KATEDRA,
       p.ZKRATKA
  FROM predmety p,
       mistnosti m,
       rozvrh r
 WHERE p.KREDITY > 2
   AND r.ID_PREDM = p.ID
   AND r.ID_MISTN = m.ID;

SELECT plan_table_output FROM table(dbms_xplan.display());

-----------------------------------------------------------------------------------------------
| Id  | Operation                     | Name             | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                  |     1 |    41 |     4   (0)| 00:00:01 |
|   1 |  NESTED LOOPS                 |                  |       |       |            |          |
|   2 |   NESTED LOOPS                |                  |     1 |    41 |     4   (0)| 00:00:01 |
|   3 |    NESTED LOOPS               |                  |     1 |    34 |     3   (0)| 00:00:01 |
|   4 |     TABLE ACCESS BY INDEX ROWID| PREDMETY         |     2 |    44 |     2   (0)| 00:00:01 |
|*  5 |      INDEX RANGE SCAN         | PREDMETY_INDEX1  |     3 |       |     1   (0)| 00:00:01 |
|   6 |     TABLE ACCESS BY INDEX ROWID| ROZVRH           |     1 |    12 |     1   (0)| 00:00:01 |
|*  7 |      INDEX RANGE SCAN         | ROZVRH_INDEX1    |     1 |       |     0   (0)| 00:00:01 |
|*  8 |    INDEX UNIQUE SCAN          | MISTNOSTI_PK     |     1 |       |     0   (0)| 00:00:01 |
|   9 |   TABLE ACCESS BY INDEX ROWID | MISTNOSTI        |     1 |     7 |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   5 - access("P"."KREDITY">2)
   7 - access("R"."ID_PREDM"="P"."ID")
       filter("R"."ID_PREDM" IS NOT NULL)
   8 - access("R"."ID_MISTN"="M"."ID")
```

CBO s využitím nápovědy - hints

Prostřednictvím tzv. nápovědy (*hintu*) mohu optimalizátoru vnutit některou operaci, protože si myslím, že její užití přispěje k lepší optimalizaci. Tato nápověda se zapisuje jako komentář specifického tvaru. Budu se snažit vnutit užití indexu pro atribut `KREDITY` v tabulce `PREDMETY`.

Nejdříve pro maximální propustnost:

```
ALTER SESSION SET optimizer_mode = all_rows;
```

```
EXPLAIN PLAN FOR
SELECT /*+ INDEX (predmety predmety_index1) */
       p.NAZEV,
       m.ZKRATKA,
       r.SEMESTR,
       r.DEN,
       r.H_OD,
       r.H_DO,
       p.KATEDRA,
       p.ZKRATKA
  FROM predmety p,
       mistnosti m,
       rozvrh r
 WHERE p.KREDITY > 2
   and r.ID_PREDM = p.ID
   and r.ID_MISTN = m.ID;

SELECT plan_table_output FROM table(dbms_xplan.display());

-----------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name            | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                 |     5 |   205 |     8  (25)| 00:00:01 |
|*  1 |  HASH JOIN                    |                 |     5 |   205 |     8  (25)| 00:00:01 |
|   2 |   MERGE JOIN                  |                 |     8 |   152 |     5  (20)| 00:00:01 |
|   3 |    TABLE ACCESS BY INDEX ROWID | MISTNOSTI       |     4 |    28 |     2   (0)| 00:00:01 |
|   4 |     INDEX FULL SCAN           | MISTNOSTI_PK    |     4 |       |     1   (0)| 00:00:01 |
|*  5 |    SORT JOIN                  |                 |     8 |    96 |     3  (34)| 00:00:01 |
|   6 |     TABLE ACCESS BY INDEX ROWID| ROZVRH          |     8 |    96 |     2   (0)| 00:00:01 |
|*  7 |      INDEX FULL SCAN          | ROZVRH_INDEX1   |     8 |       |     1   (0)| 00:00:01 |
|   8 |   TABLE ACCESS BY INDEX ROWID | PREDMETY        |     3 |    66 |     2   (0)| 00:00:01 |
|*  9 |    INDEX RANGE SCAN           | PREDMETY_INDEX1 |     3 |       |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("R"."ID_PREDM"="P"."ID")
   5 - access("R"."ID_MISTN"="M"."ID")
       filter("R"."ID_MISTN"="M"."ID")
   7 - filter("R"."ID_PREDM" IS NOT NULL)
   9 - access("P"."KREDITY">2)
```

a poté pro nejrychlejší odezvu:

```
ALTER SESSION SET optimizer_mode = first_rows_1;

EXPLAIN PLAN FOR
SELECT /*+ INDEX (predmety predmety_index1) */
       p.NAZEV,
       m.ZKRATKA,
       r.SEMESTR,
       r.DEN,
       r.H_OD,
       r.H_DO,
       p.KATEDRA,
       p.ZKRATKA
  FROM predmety p,
       mistnosti m,
       rozvrh r
 WHERE p.KREDITY > 2
   and r.ID_PREDM = p.ID
   and r.ID_MISTN = m.ID;

SELECT plan_table_output FROM table(dbms_xplan.display());

-----------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name            | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                 |     1 |    41 |     4   (0)| 00:00:01 |
|   1 |  NESTED LOOPS                 |                 |       |       |            |          |
|   2 |   NESTED LOOPS                |                 |     1 |    41 |     4   (0)| 00:00:01 |
|   3 |    NESTED LOOPS               |                 |     1 |    34 |     3   (0)| 00:00:01 |
|   4 |     TABLE ACCESS BY INDEX ROWID| PREDMETY        |     2 |    44 |     2   (0)| 00:00:01 |
|*  5 |      INDEX RANGE SCAN         | PREDMETY_INDEX1 |     3 |       |     1   (0)| 00:00:01 |
|   6 |     TABLE ACCESS BY INDEX ROWID| ROZVRH          |     1 |    12 |     1   (0)| 00:00:01 |
|*  7 |      INDEX RANGE SCAN         | ROZVRH_INDEX1   |     1 |       |     0   (0)| 00:00:01 |
|*  8 |    INDEX UNIQUE SCAN          | MISTNOSTI_PK    |     1 |       |     0   (0)| 00:00:01 |
|   9 |   TABLE ACCESS BY INDEX ROWID | MISTNOSTI       |     1 |     7 |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   5 - access("P"."KREDITY">2)
   7 - access("R"."ID_PREDM"="P"."ID")
       filter("R"."ID_PREDM" IS NOT NULL)
   8 - access("R"."ID_MISTN"="M"."ID")
```

RBO - Rule Based Optimizer

Tento způsob optimalizace je v SŘBD Oracle již označen jako *deprecated*, tj. zavrhovaný, je možné z důvodu kompatibility jej aktivovat odpovídajícím hintem `RULE`:

Nejdříve pro maximální propustnost:

```
ALTER SESSION SET optimizer_mode = all_rows;

EXPLAIN PLAN FOR
SELECT /*+ RULE */
       p.NAZEV,
       m.ZKRATKA,
```

```
       r.SEMESTR,
       r.DEN,
       r.H_OD,
       r.H_DO,
       p.KATEDRA,
       p.ZKRATKA
  FROM predmety p,
       mistnosti m,
       rozvrh r
 WHERE p.KREDITY > 2
   and r.ID_PREDM = p.ID
   and r.ID_MISTN = m.ID;

SELECT plan_table_output FROM table(dbms_xplan.display());

-----------------------------------------------------------
| Id  | Operation                     | Name            |
-----------------------------------------------------------
|   0 | SELECT STATEMENT              |                 |
|   1 |  NESTED LOOPS                 |                 |
|   2 |   NESTED LOOPS                |                 |
|   3 |    NESTED LOOPS               |                 |
|   4 |     TABLE ACCESS BY INDEX ROWID| PREDMETY        |
|*  5 |      INDEX RANGE SCAN         | PREDMETY_INDEX1 |
|   6 |     TABLE ACCESS BY INDEX ROWID| ROZVRH          |
|*  7 |      INDEX RANGE SCAN         | ROZVRH_INDEX1   |
|*  8 |    INDEX UNIQUE SCAN          | MISTNOSTI_PK    |
|   9 |   TABLE ACCESS BY INDEX ROWID | MISTNOSTI       |
-----------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   5 - access("P"."KREDITY">2)
   7 - access("R"."ID_PREDM"="P"."ID")
   8 - access("R"."ID_MISTN"="M"."ID")

Note
-----
   - rule based optimizer used (consider using cbo)
```

a poté pro nejrychlejší odezvu:

```
ALTER SESSION SET optimizer_mode = first_rows_1;

EXPLAIN PLAN FOR
SELECT /*+ RULE */
       p.NAZEV,
       m.ZKRATKA,
       r.SEMESTR,
       r.DEN,
       r.H_OD,
       r.H_DO,
       p.KATEDRA,
       p.ZKRATKA
  FROM predmety p,
       mistnosti m,
       rozvrh r
 WHERE p.KREDITY > 2
   and r.ID_PREDM = p.ID
   and r.ID_MISTN = m.ID;

SELECT plan_table_output FROM table(dbms_xplan.display());

-----------------------------------------------------------
| Id  | Operation                     | Name            |
-----------------------------------------------------------
|   0 | SELECT STATEMENT              |                 |
|   1 |  NESTED LOOPS                 |                 |
|   2 |   NESTED LOOPS                |                 |
|   3 |    NESTED LOOPS               |                 |
|   4 |     TABLE ACCESS BY INDEX ROWID| PREDMETY        |
|*  5 |      INDEX RANGE SCAN         | PREDMETY_INDEX1 |
|   6 |     TABLE ACCESS BY INDEX ROWID| ROZVRH          |
|*  7 |      INDEX RANGE SCAN         | ROZVRH_INDEX1   |
|*  8 |    INDEX UNIQUE SCAN          | MISTNOSTI_PK    |
|   9 |   TABLE ACCESS BY INDEX ROWID | MISTNOSTI       |
-----------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   5 - access("P"."KREDITY">2)
   7 - access("R"."ID_PREDM"="P"."ID")
   8 - access("R"."ID_MISTN"="M"."ID")

Note
-----
   - rule based optimizer used (consider using cbo)
```

Mnohem více informací nejen o optimalizaci dotazu se dočtete v knize Oracle Database Performance Tuning Guide.