Sistemi baza podataka

PL/SQL – Indeksi i optimizacija upita

Sadržaj

- Indeksi
- Optimizacija upita
- Naredba EXPLAIN PLAN

Indeksi

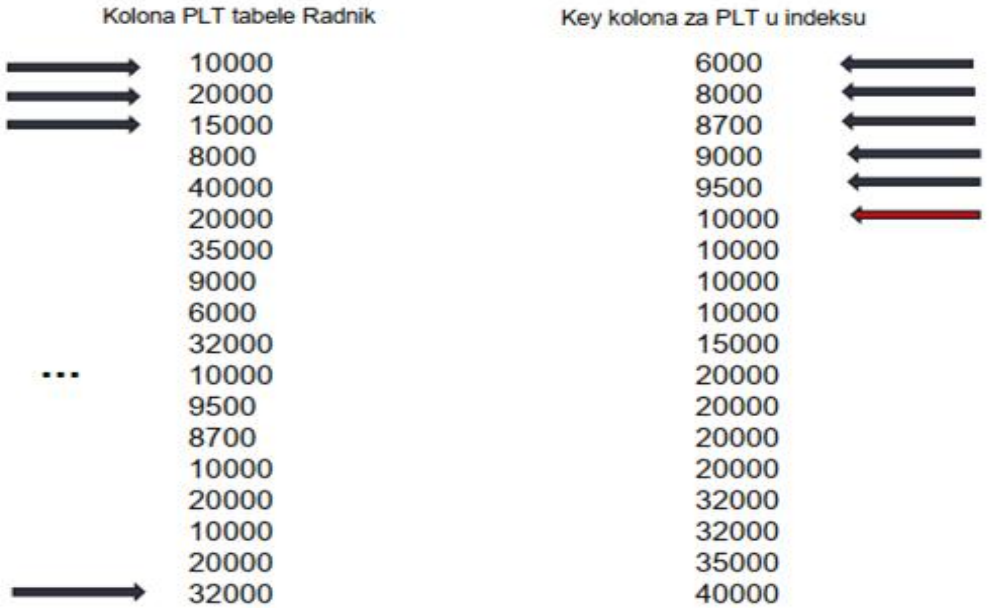
Indeksi

- Objekti SUBP-a koji obezbeđuju brz i direktan pristup torkama u tabeli
- SUBP automatski održava i koristi indekse
- Kreiranje indeksa ne menja podatke u tabeli, već se kreira posebna struktura koja se odnosi na tabelu
- Podaci u indeksu su redudantni u odnosu na podatke u tabeli
- Neki od uzroka loših performansi upita:
  - Nedostatak indeksa
  - Neodgovarajući izbor indeksiranja (engl. Poor Indexing)
  - Neupotrebljeni raspoloživi indeksi

Indeksi

- Podaci u indeksu su sortirani (podrazumevano rastuće) što omogućava bržu i lakšu pretragu

```
SELECT plt
FROM radnik
WHERE plt < 10000;
```



Indeksi

- Indeks se koristi:
  - kada se kolona često koristi u traženju ili u izrazima upita
  - kada se koristi u spojevima tabela
  - kada je velika selektivnost kolone
  - za strani ključ kada se kolone stranog ključa često koriste u uslovima spajanja
- Indeks se ne koristi:
  - kada kolona ili izraz imaju samo nekoliko različitih vrednosti
    - izuzev bitmap indeksa u DW
  - kada se često radi update date kolone
  - kada se kolone koriste samo u okviru funkcija ili izraza

Kreiranje indeksa

```
CREATE [UNIQUE] INDEX index_name
ON table_name (column1, column2, …) [REVERSE];
```

- **Indeksne** (engl. *key*) **kolone** – kolone nad kojima je kreiran indeks
- Oracle Server automatski kreira i koristi indekse za:
  - Ograničenje primarnog ključa
  - Ograničenje jedinstvenosti vrednosti obeležja

```
SELECT index_name
FROM user_indexes
WHERE UPPER(table_name) = 'RADNIK';
```

Brisanje indeksa

```
DROP INDEX index_name;
```

Tipovi indeksa

- **Simple index**
  - Indeksi nad jednom kolonom

```
CREATE INDEX idx_plt ON radnik (plt);
```

- **Composite index**
  - Indeksi nad više kolona
  - Obično kolone koje se često koriste zajedno
  - Prvo navesti kolonu sa najviše jedinstvenih vrednosti radi bolje selektivnosti

```
CREATE INDEX idx_ip ON radnik (ime, prz);
```

- **Unique index**
  - Indeksi nad kolonom koja treba da ima jedinstvene vrednosti

```
CREATE UNIQUE INDEX idx_uq ON projekat (nap);
```

Tipovi indeksa

- **Reverse index**
  - Koristi se za rešavanje problema neuravnoteženosti indeksa
  - Vrednosti indeksa čuvaju se u obrnutom redosledu kako bi se obezbedila uravnotežena raspodela i sprečilo gubljenje prostora u strukturi indeksa
  - Uobičajeni tipovi vrednosti za indeks su primarni ključevi čije se vrednosti dodeljuju u rastućem redosledu (desna strana stabla), a postoji tendencija za brisanjem starih vrednosti što dovodi do neuravnoteženosti indeksa
  - Vrednost ključa 123 se čuva kao 321

```
CREATE INDEX ind_rv ON radnik (mbr) REVERSE;
```

Tipovi indeksa

- **Function based index**
  - Indeks nad izrazima ili funkcijama
  - Izračuna se vrednost izraza pomoću jedne ili više kolona
  - Izraz može biti:
    - SQL funkcija
    - PL/SQL funkcija
    - Funkcije iz paketa
  - Izraz **ne sme** da sadrži null vrednost (koristiti nvl ili coalesce funkciju)

```
CREATE INDEX idx_fn ON radnik (NVL(plt + pre), plt);
```

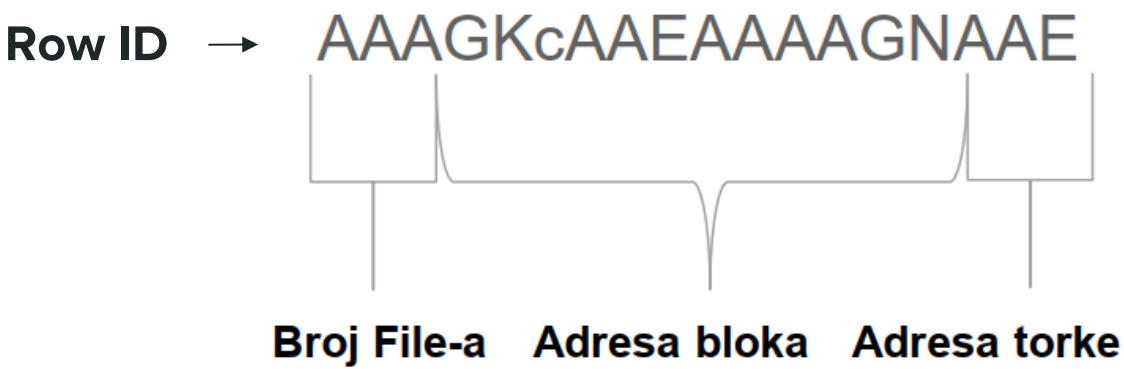
Optimizacija upita

Pristupni putevi

- Full table scans
  - Čitanje svih torki iz tabele uz filtriranje onih koje zadovoljavaju uslov selekcije
- Row ID scans
- Index scans

Row ID scans

- Row ID je jedinstveni identifikatori svake torke u tabeli
  - Sadrži informacije o fajlu i bloku tražene torke uz adresu torke unutar bloka
- Row ID scans predstavnja čitanje torki direktno preko Row ID-a
- Najbrži način za dobavljanje jedne torke
- Ukoliko su sve tražene kolone iz upita deo indeksa, ne pristupa se tabeli



Index scans

- **Index Range scan**
  - Skeniranje opsega vrednosti po sortiranim vrednostima indeksa

- **Index Full scan**
  - Čitanje svih vrednosti indeksa redom bez obzira na uslove pretrage; moguća eliminacija operacija sortiranja zato što je indeks sortiran po Key koloni

- **Index Fast-full scan**
  - Čitanje blokova indeksa u redosledu smeštanja na disk (**nesortirani** redosled)
  - Koristi se za čitanje indeksa umesto tabele, odnosno za pristupanje samo onim atributima koji su u indeksu

Index scans

- **Index Join scan**
  - Spajanje više indeksa i vraćanje svih kolona zahtevanih upitom
  - Eliminiše se pristupanje tabelama jer se svi podaci dobijaju iz indeksa

- **Index Skip scan**
  - Kada inicijalna (prva) kolona kompozitnog indeksa nije specificirana u upitu
  - Kreiran indeks:

```
CREATE INDEX idx_ime_prz ON radnik (ime, prz);
```

  - Upit:

```
SELECT * FROM radnik WHERE prz='Peric';
```

Optimizator

- Optimizator određuje najefikasniji način za izvršavanje SQL naredbe
- U velikoj meri utiče na vreme izvršavanja
- Pronalazi optimalni pristupni put
  - Na osnovu SQL naredbe generiše skup mogućih **planova izvršavanja**
  - Procenjuje koštanje plana na osnovu statistike (broj torki, selektivnost indeksa...)
  - Bira „najjeftiniji“ pristupni put

Naredba EXPLAIN PLAN

Plan izvršavanja (EXPLAIN PLAN)

- Naredba explain plan:
  - Generiše plan izvršavanja optimizatora
  - Čuva plan u tabeli PLAN_TABLE
  - Ne izvršava naredbu već analizira plan njenog izvršavanja

```
EXPLAIN PLAN
[SET STATEMENT_ID = ‘text’]
[INTO tabela_za_plan]
FOR sql naredba;
```

  - STATEMENT_ID - identifikator naredbe

EXPLAIN PLAN

- Pregled:

```
SELECT PLAN_TABLE_OUTPUT
FROM TABLE (DBMS_XPLAN.DISPLAY([‘tabela’ , ‘statemend_id’]));
```

- DBMS_XPLAN paket
  - funkcija DISPLAY - formatiranje i prikazivanje naredbe
- Statement_id
  - Ako je korištena opcija SET STATEMENT_ID = ‘text’
  - Ukoliko se ne navede prikazuje se poslednja naredba smeštena u tabeli planova

EXPLAIN PLAN-indeksi i strani ključ

```
EXPLAIN PLAN
SET STATEMENT_ID = 'primer1'
FOR
SELECT ime, prz
FROM radnik
WHERE mbr IN (SELECT ruk FROM projekat);

SELECT PLAN_TABLE_OUTPUT FROM TABLE
(DBMS_XPLAN.DISPLAY('plan_table','primer1'));
```

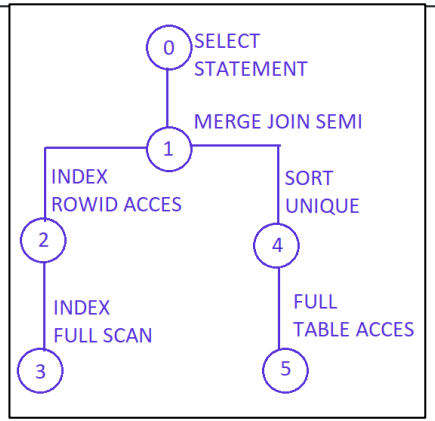
EXPLAIN PLAN-indeksi i strani ključ

- Pre kreiranja indeksa:

```
Plan hash value: 4292270174

-----------------------------------------------------------------------------------------
| Id  | Operation                     | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |            |     3 |    60 |     6  (17)| 00:00:01 |
|   1 |  MERGE JOIN SEMI              |            |     3 |    60 |     6  (17)| 00:00:01 |
|   2 |   TABLE ACCESS BY INDEX ROWID| RADNIK     |    24 |   408 |     2   (0)| 00:00:01 |
|   3 |    INDEX FULL SCAN            | RADNIK PK  |    24 |       |     1   (0)| 00:00:01 |
|*  4 |   SORT UNIQUE                 |            |     8 |    24 |     4  (25)| 00:00:01 |
|   5 |    TABLE ACCESS FULL          | PROJEKAT   |     8 |    24 |     3   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------
```

- Stablo izvršavanja (engl. Parse tree)



EXPLAIN PLAN-indeksi i strani ključ

- Nakon kreiranja indeksa

```
CREATE INDEX ind_fk ON projekat(ruk);
```

```
Plan hash value: 166315348

-------------------------------------------------------------------------------
| Id  | Operation           | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |         |     3 |    60 |     3   (0)| 00:00:01 |
|   1 |  NESTED LOOPS SEMI  |         |     3 |    60 |     3   (0)| 00:00:01 |
|   2 |   TABLE ACCESS FULL| RADNIK  |    24 |   408 |     3   (0)| 00:00:01 |
|*  3 |   INDEX RANGE SCAN  | IND FK  |     1 |     3 |     0   (0)| 00:00:01 |
-------------------------------------------------------------------------------

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

   3 - access("MBR"="RUK")
```

EXPLAIN PLAN-function based indeks

```
CREATE INDEX idx_plt_pre ON radnik (NVL(plt + pre, plt));

EXPLAIN PLAN
SET STATEMENT_ID = 'primer_rng'
FOR
SELECT *
FROM radnik
WHERE NVL(plt + pre, plt ) > 10000;

SELECT PLAN_TABLE_OUTPUT FROM TABLE
(DBMS_XPLAN.DISPLAY('plan_table','primer_rng'));
```

Tabela dolasci radnika

```
CREATE TABLE radnik_dolazak(
    Mbr integer NOT NULL,
    Datum date,
    Dan varchar(20) NOT NULL,
    Prisutan varchar(2),
    Razlog_odsustva varchar(20),
    CONSTRAINT radnik_dolazak_PK PRIMARY KEY (MBR, Datum),
    CONSTRAINT radnik_dolazak_FK FOREIGN KEY (Mbr)
                                        REFERENCES Radnik (Mbr)
);
```

Primer - unique kolone

- Kreirati i prikazati plan izvršavanja za upit koji vraća broj dana godišnjeg odmora rukovodioca projekta sa nazivom ‘Optimizacija mreže’.

Primer - unique kolone

```
EXPLAIN PLAN
SET STATEMENT_ID = 'primer_uq'
FOR
SELECT COUNT(*)
FROM radnik_dolazak
WHERE razlog_odsustva='godisnji' AND mbr IN
(SELECT ruk FROM projekat
WHERE nap = 'Optimizacija mreze');

SELECT PLAN_TABLE_OUTPUT
FROM TABLE (DBMS_XPLAN.DISPLAY('plan_table','primer_uq'));
```

Zadatak za vežbu

- Kreirati i prikazati plan izvršavanja za upit koji prikazuje podatke o radnicima koji imaju platu veću od prosečne plate svih radnika.
- Optimizovati upit.
- Ponovo kreirati i prikazati plan izvršavanja i uporediti korake izvršavanja pre optimizacije.

Zadatak za vežbu - rešenje

```
EXPLAIN PLAN
SET STATEMENT_ID = 'primer_plata'
FOR
SELECT *
FROM radnik
WHERE plt > (SELECT AVG(plt) FROM radnik);

CREATE INDEX idx_plt ON radnik (plt);
```

Zadatak za vežbu - rešenje

- Pre kreiranja indeksa

```
-----------------------------------------------------------------------------
| Id  | Operation          | Name   | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |        |     1 |    87 |     6   (0)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL | RADNIK |     1 |    87 |     3   (0)| 00:00:01 |
|   2 |   SORT AGGREGATE   |        |     1 |    13 |            |          |
|   3 |    TABLE ACCESS FULL| RADNIK |    24 |   312 |     3   (0)| 00:00:01 |
-----------------------------------------------------------------------------
```

- Nakon kreiranja indeksa

```
-----------------------------------------------------------------------------------------------
| Id  | Operation                           | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |         |     1 |    87 |     3   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID BATCHED| RADNIK  |     1 |    87 |     2   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN                  | IDX_PLT |     1 |       |     1   (0)| 00:00:01 |
|   3 |    SORT AGGREGATE                   |         |     1 |    13 |            |          |
|   4 |     INDEX FULL SCAN                 | IDX_PLT |    24 |   312 |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------
```

Zadatak za vežbu

- Kreirati i prikazati plan izvršavanja za upit koji računa broj svih radnika koji su u maju 2022. godine bili odsutni.
- Optimizovati upit.
- Ponovo kreirati i prikazati plan izvršavanja i uporediti korake izvršavanja pre optimizacije.

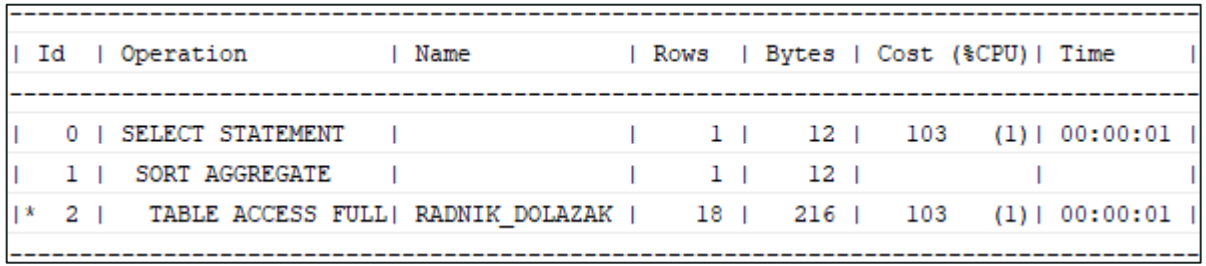
Zadatak za vežbu - rešenje

```
EXPLAIN PLAN
SET STATEMENT_ID = 'primer_odsutni_maj'
FOR
SELECT COUNT(*)
FROM radnik_dolazak
WHERE prisutan = 'ne' AND datum BETWEEN
              to_date('01.05.2022.','dd.mm.yyyy.') AND
                              to_date('31.05.2022.','dd.mm.yyyy.');

CREATE INDEX idx_prisutan_datum ON radnik_dolazak (prisutan, datum);
```

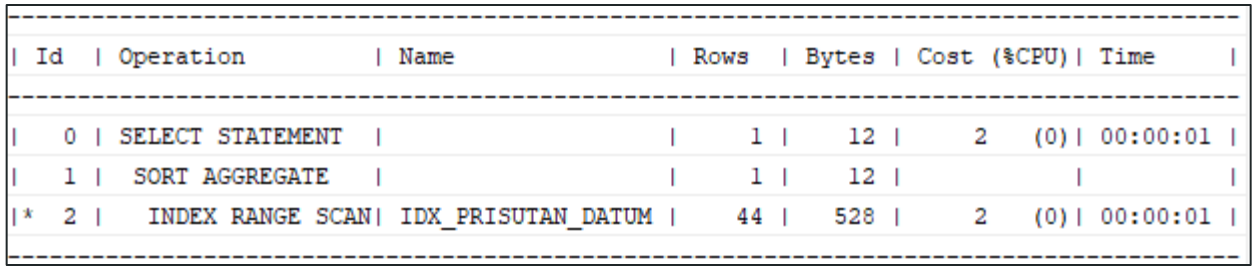
Zadatak za vežbu - rešenje

- Pre kreiranja indeksa



Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	12	103 (1)	00:00:01
1	SORT AGGREGATE		1	12		
* 2	TABLE ACCESS FULL	RADNIK_DOLAZAK	18	216	103 (1)	00:00:01

- Nakon kreiranja indeksa



Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	12	2 (0)	00:00:01
1	SORT AGGREGATE		1	12		
* 2	INDEX RANGE SCAN	IDX_PRISUTAN_DATUM	44	528	2 (0)	00:00:01

Zadatak za vežbu

- Kreirati i prikazati plan izvršavanja za upit koji za svakog radnika koji je nekada bio na bolovanju prikazuje njegovo ime i prezime i broj dana na bolovanju.
- Optimizovati upit.
- Ponovo kreirati i prikazati plan izvršavanja i uporediti korake izvršavanja pre optimizacije.

Zadatak za vežbu - rešenje

```
EXPLAIN PLAN
SET STATEMENT_ID = 'primer_bolovanje'
FOR
SELECT rd.mbr, r.ime, r.prz, COUNT(*)
FROM radnik_dolazak rd INNER JOIN radnik r ON rd.mbr = r.mbr
WHERE razlog_odsustva = 'bolovanje'
GROUP BY rd.mbr, r.ime, r.prz;

CREATE INDEX idx_razlog_odsustva
                          ON radnik_dolazak (razlog_odsustva);
```

Zadatak za vežbu - rešenje

- Pre kreiranja indeksa

```
-------------------------------------------------------------------------------------
| Id  | Operation           | Name            | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |                 |  2009 |   125K|   107   (2)| 00:00:01 |
|   1 |  HASH GROUP BY      |                 |  2009 |   125K|   107   (2)| 00:00:01 |
|*  2 |   HASH JOIN         |                 |  2009 |   125K|   106   (1)| 00:00:01 |
|   3 |    TABLE ACCESS FULL| RADNIK          |    24 |   936 |     3   (0)| 00:00:01 |
|*  4 |    TABLE ACCESS FULL| RADNIK_DOLAZAK  |  2009 | 50225 |   103   (1)| 00:00:01 |
-------------------------------------------------------------------------------------
```

- Nakon kreiranja indeksa

```
-----------------------------------------------------------------------------------------------------------
| Id  | Operation                           | Name               | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |                    |  2432 |   152K|    49   (3)| 00:00:01 |
|   1 |  HASH GROUP BY                      |                    |  2432 |   152K|    49   (3)| 00:00:01 |
|*  2 |   HASH JOIN                         |                    |  2432 |   152K|    48   (0)| 00:00:01 |
|   3 |    TABLE ACCESS FULL                | RADNIK             |    24 |   936 |     3   (0)| 00:00:01 |
|   4 |    TABLE ACCESS BY INDEX ROWID BATCHED| RADNIK_DOLAZAK   |  2432 | 60800 |    45   (0)| 00:00:01 |
|*  5 |     INDEX RANGE SCAN                | IDX_RAZLOG_ODSUSTVA|  2432 |       |     7   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------------
```

Indeksi i operator LIKE

```
SELECT *
FROM radnik_dolazak
WHERE razlog_odsustva like 'b%';
```

- Da li će za ovaj upit biti iskorišćen indeks idx_razlog_odsustva? **DA**
  - Kada se koristi operator like uz karaktere '%' i '_' indeks se ne može iskoristiti ukoliko su ti specijalni karakteri na početku izraza.
  - U drugim slučajevima indeks se može iskoristiti.

Zadatak za vežbu

- Kreirati i prikazati plan izvršavanja za upit koji prikazuje ukupan broj dana godišnjeg odmora svih radnika. Pri poređenju razloga odsustva koristiti funkciju UPPER.
- Optimizovati upit.
- Ponovo kreirati i prikazati plan izvršavanja i uporediti korake izvršavanja pre optimizacije.

Zadatak za vežbu - rešenje

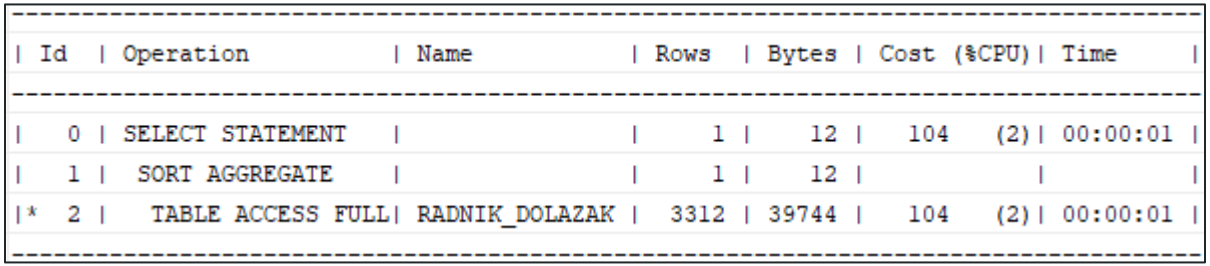
```
EXPLAIN PLAN
SET STATEMENT_ID = 'primer_upper'
FOR
SELECT COUNT(*) as broj_dana_godisnjeg
FROM radnik_dolazak
WHERE UPPER(razlog_odsustva) = 'GODISNJI';

CREATE INDEX idx_razlog_odsustva_upp
             ON radnik_dolazak (UPPER(razlog_odsustva));
```

- **Napomena:** nije iskorišten indeks idx_razlog_odsustva zbog funkcije UPPER

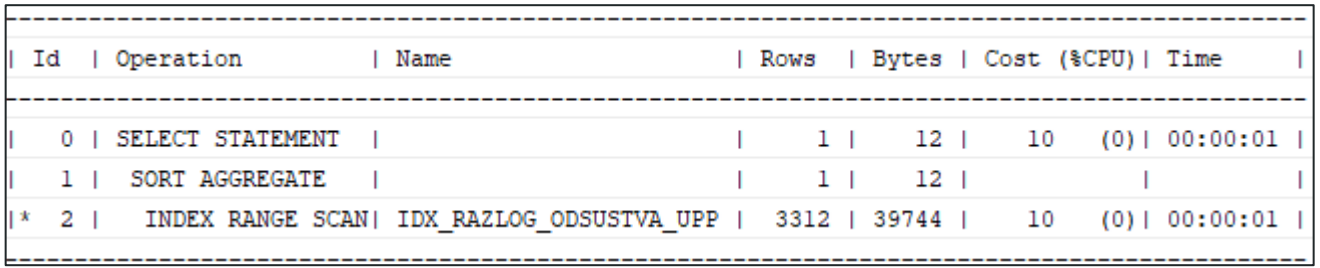
Zadatak za vežbu - rešenje

- Pre kreiranja indeksa



Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	12	104 (2)	00:00:01
1	SORT AGGREGATE		1	12		
* 2	TABLE ACCESS FULL	RADNIK_DOLAZAK	3312	39744	104 (2)	00:00:01

- Nakon kreiranja indeksa



Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	12	10 (0)	00:00:01
1	SORT AGGREGATE		1	12		
* 2	INDEX RANGE SCAN	IDX_RAZLOG_ODSUSTVA_UPP	3312	39744	10 (0)	00:00:01

Kraj!

Hvala na pažnji!