KURSORI BAZE PODATAKA U ORACLE 11g

Zlatko Sirotić, dipl.ing.
Istra informatički inženjering d.o.o.
Pula

SQL - deklarativni jezik

- SQL je (visoko) **deklarativni programski jezik**, koji može vraćati ili ažurirati ne samo jedan redak podataka, već čitav skup redaka
- SQL se u programiranju ne koristi samostalno, već se koristi zajedno sa nekim drugim programskim jezikom, uglavnom **proceduralnim**
- čest način povezivanja SQL-a sa drugim jezikom je taj da se SQL ugrađuje u drugi jezik, koji tada obično nazivamo **jezik-domaćin** (host language)

PL/SQL
– proceduralno proširenje SQL-a

- Oracle korporacija je krajem 80-tih godina prošlog stoljeća napravila svoj jezik-domaćin **PL/SQL** (Procedural Language extensions to the Structured Query Language), na temelju programskog jezika ADA 83
- Oracle je tako dobro integrirao proceduralni jezik PL/SQL i jezik SQL, da u praksi često zaboravimo kako, zapravo, radimo sa dva jezika
- to su ipak **dva odvojena jezika**: kada baza prevodi PL/SQL programski kod, PL/SQL prevoditelj prevodi one naredbe koje on "razumije", a SQL naredbe šalje SQL prevoditelju

Kursori (baze podataka)

- između proceduralnog jezika-domaćina, koji procesira redak po redak, i deklarativnog SQL jezika, koji radi sa skupom redaka, postoji očiti **nesklad u radu**
- taj se nesklad premošćuje pomoću **kursora** baze podataka (database cursors)
- kursor predstavlja neku vrstu **logičkog pokazivača na skup redaka** koji baza vraća kao odgovor na postavljeni upit
- proceduralni jezik koristi kursor da bi obrađivao **redak po redak** odgovora (na upit)

Oracle SUBP - sve je kursor

- SUBP sustavi različitih proizvođača **mogu se značajno razlikovati** ne samo u internoj realizaciji, nego i po svom ponašanju (npr. različit način rada sa transakcijama i zaključavanjem redaka), ali i po dijalektu SQL jezika i po korištenoj terminologiji
- tako, kursori u Oracle relacijskom sustavu ne moraju značiti isto što i kursori u nekom drugom relacijskom sustavu
- **Oracle uvijek radi sa kursorima**, ali ti kursori mogu u programskom kodu biti "skriveni" ili mogu biti "vidljivi" - govorimo o implicitnim i eksplicitnim kursorima

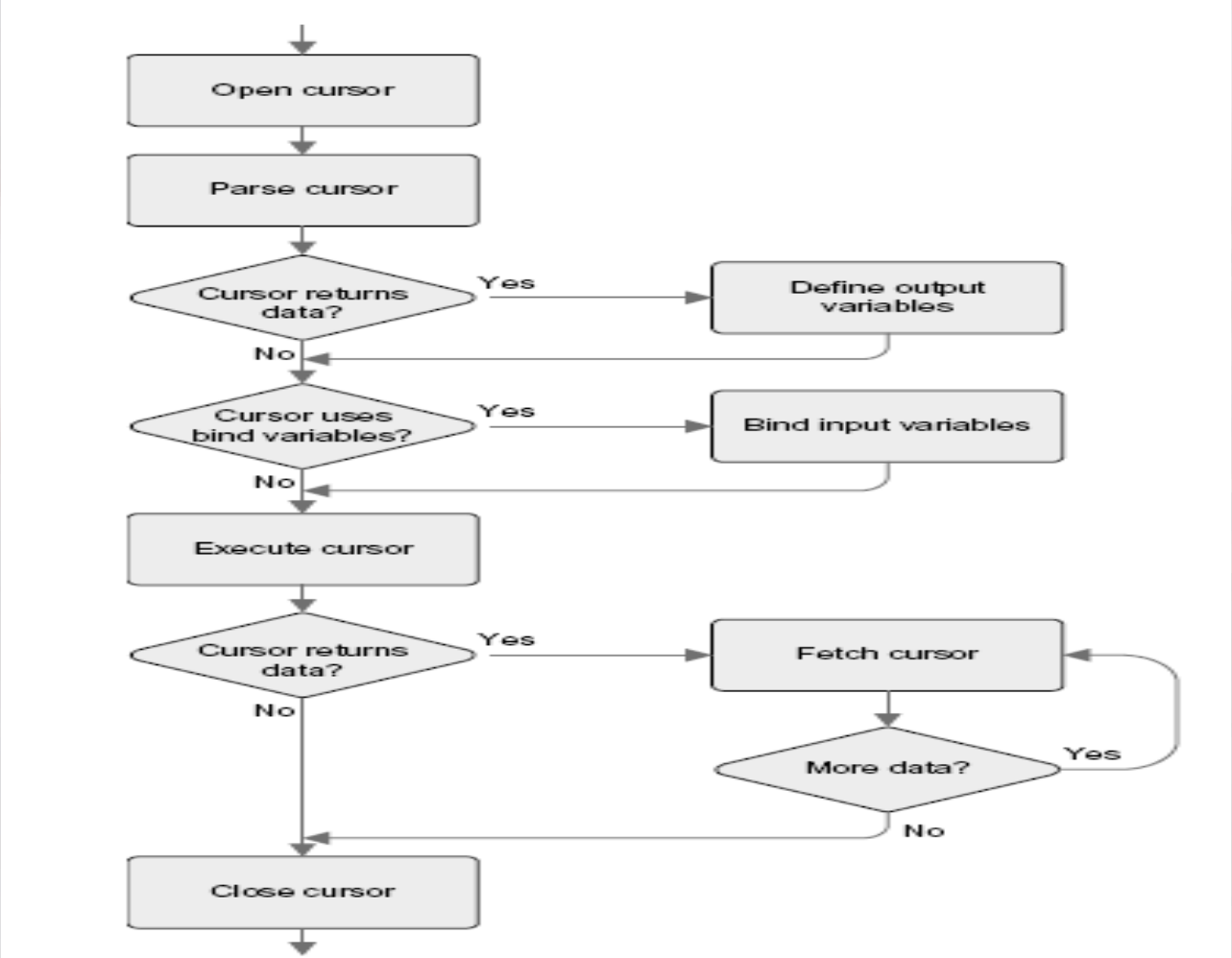
Procesiranje SQL naredbe

- U Oracle sustavu kursor baze podataka je privatno SQL područje u kojem se čuvaju informacije za procesiranje određene SQL naredbe. Kod procesiranja SQL naredbe dešava se sljedeće:
- Otvaranje (**OPEN**) kursora. Alociraju se memorijske strukture za kursor u privatnoj memoriji serverskog procesa sesije – UGA (user global area). SQL naredba još nije pridružena kursoru.
- Parsiranje (**PARSE**) kursora. SQL naredba pridružuje se kursoru. Njena parsirana reprezentacija, koja uključuje plan izvršenja (execution plan) učitava se u dio SGA memorije (system global area), tzv. **library cache**. Strukture u UGA memoriji se ažuriraju tako da pokazuju na djeljivi kursor u library cache-u.

Procesiranje SQL naredbe - nastavak

- Definiranje izlaznih (**DEFINE OUTPUT**) varijabli.
- Povezivanje ulaznih (**BIND INPUT**) varijabli.
- Izvršavanje (**EXECUTE**) kursora. Kod nekih naredbi, npr. kod najvećeg broja upita, realno procesiranje zapravo čeka sljedeću fazu.
- Dohvaćanje (**FETCH**) kursora. Ako SQL naredba vraća retke, to se dešava upravo u ovom koraku.
- Zatvaranje (**CLOSE**) kursora. Resursi koji su pridruženi kursoru u UGA memoriji se oslobađaju. Međutim, djeljivi kursor u library cache memoriji se ne briše, već ostaje za eventualno ponovno korištenje u budućnosti.

Procesiranje SQL naredbe - grafički prikaz



Implicitni kursori

- PL/SQL deklarira implicitni kursor onda kada u programskom kodu koristimo DML naredbe ili SELECT naredbu koja vraća samo jedan redak:

```
SELECT stupac1, stupac2
   INTO varijabla1, varijabla2
   FROM tablica …
```

- u kodu se nigdje ne vidi riječ CURSOR, ali interno Oracle PL/SQL prevoditelj izvodi različite radnje kao što su otvaranje kursora, ugrađivanje varijabli jezika-domaćina (bind variables) u SQL upit, slanje upita SQL prevoditelju, čitanje redaka, zatvaranje kursora

Eksplicitni kursori – "najeksplicitnija verzija"

```
DECLARE
  CURSOR art_c IS
    SELECT sifra, naziv FROM m_artikli; …
BEGIN
  OPEN art_c;
  LOOP
    FETCH art_c INTO sifra_l, naziv_l;
    EXIT WHEN art_c%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE …
  END LOOP;
  CLOSE art_c; …
```

Eksplicitni kursori – "srednja verzija"

```
DECLARE
  CURSOR art_c IS
    SELECT sifra, naziv FROM m_artikli;
BEGIN
  FOR redak IN art_c LOOP
    DBMS_OUTPUT.PUT_LINE …
  END LOOP;
END;
```

Eksplicitni kursori – "najjednostavnija verzija"

```
BEGIN
  FOR redak IN (
    SELECT sifra, naziv
      FROM m_artikli)
  LOOP
    DBMS_OUTPUT.PUT_LINE …
  END LOOP;
END;
```

Česta greška – koristi se kursor, umjesto obične DML naredbe

```
FOR redak IN (
  SELECT sifra, cijena FROM m_artikli
   WHERE cijena < 100)
LOOP
  UPDATE m_artikli
     SET cijena = cijena * 1.1
   WHERE sifra = redak.sifra;
END LOOP;
-- a može i jednostavno ☺
UPDATE m_artikli
   SET cijena = cijena * 1.1
 WHERE cijena < 100;
```

Kursor varijable (REF kursori)

- **kursor varijable** su varijable čiji je **tip REF kursor** (REF kao referenca ili pokazivač), pa se u Oracle literaturi same varijable često neformalno nazivaju REF kursori
- kursor varijable predstavljaju **pokazivač na "obični" kursor**
- budući da je "obični" kursor sam po sebi neka vrsta (logičkog) pokazivača, moglo bi se reći da je kursor varijabla "pokazivač na pokazivač“ (naravno, ne kao u jezicima C / C++)

Kursor varijable - mogućnosti

- za razliku od eksplicitnih kursora, kursor varijable mogu se otvoriti u jednom potprogramu, pa se u njemu mogu (ali ne moraju) čitati, a nakon toga se **čitanje može prebaciti na drugi potprogram** (koji se može nalaziti čak i na drugom računalu
- kursor varijable omogućavaju i da ista kursor varijabla jedanput pokazuje na jedan kursor, a drugi put na drugi, čime se omogućava **fleksibilnije programiranje**
- kursor varijable tipa **jaki REF kursor** moraju uvijek pokazivati na kursore istih struktura, a one tipa **slabi REF kursor** mogu pokazivati na kursore različitih struktura

Kursor varijable – sa njima se radi slično kao sa eksplicitnim kursorima

```
DECLARE
  TYPE art_rc_type IS REF CURSOR
    RETURN m_artikli%ROWTYPE; -- jaki tip
  art_rc art_rc_type; -- kursor varijabla
BEGIN
  OPEN art_rc FOR SELECT * FROM m_artikli;
  LOOP
    FETCH art_rc INTO art_row;
    EXIT WHEN art_rc%NOTFOUND; …
  END LOOP;
  CLOSE art_rc;
```

Kursor varijable – slabi REF kursor tip

```
DECLARE
   TYPE art_rc_type IS REF CURSOR; -- slabi
   art_rc art_rc_type; …
BEGIN
   OPEN art_rc FOR
      SELECT * FROM m_artikli; …
   CLOSE art_rc;
   OPEN art_rc FOR
      SELECT * FROM m_dobavljaci; …
   CLOSE art_rc; …
```

Kursor varijable – najčešće korištenje: funkcija otvara i vraća kursor varijablu

```
CREATE FUNCTION funkcija
  RETURN SYS_REFCURSOR
IS
  art_rc SYS_REFCURSOR;
BEGIN
  OPEN art_rc FOR
    SELECT * FROM m_artikli;
  RETURN art_rc;
END;
```

Kursor varijable – najčešće korištenje: … a procedura ju koristi

```
CREATE PROCEDURE procedura IS
  art_rc SYS_REFCURSOR;
  art_row m_artikli%ROWTYPE;
BEGIN
  art_rc := funkcija; -- poziv funkcije
  LOOP
    FETCH art_rc INTO art_row;
    EXIT WHEN art_rc%NOTFOUND; …
  END LOOP;
  CLOSE art_rc; -- funkcija je otvorila
END;
```

Dinamički SQL i PL/SQL

- u dosadašnjim primjerima programski kod bio je **statički**, tj. cjelokupan kod bio je formiran već u vrijeme prevođenja programa
- za razliku od toga, **dinamički** programski kod se kreira za vrijeme izvođenja programa
- prednost je dinamičkog koda da omogućava vrlo **fleksibilno (generičko) programiranje**
- mana je dinamičkog koda da se on **provjerava tek u fazi izvođenja**, što znači da se u toku izvođenja mogu pojaviti neke greške koje bi se u statičkom kodu otkrile unaprijed, u vrijeme prevođenja

Dinamički SQL i PL/SQL – paket DBMS_SQL i prirodni dinamički SQL

- dinamički kod (SQL i PL/SQL) u Oracle bazi pojavio se prvi puta u verziji Oracle 7, i to kao **paket DBMS_SQL**
- iako se paket zove DBMS_SQL, on omogućava ne samo dinamički SQL, nego i dinamički PL/SQL
- zbog složenosti rada sa paketom DBMS_SQL, Oracle je u verziji 8i napravio tzv. **prirodni dinamički SQL** (native dynamic SQL – NDS); i on omogućava dinamički SQL i dinamički PL/SQL
- prirodni dinamički SQL čine dvije vrste naredbi: jedna je **EXECUTE IMMEDIATE**, a druga je naredba za kreiranje **dinamičke kursor varijable**

Prirodni dinamički SQL – naredba EXECUTE IMMEDIATE

```
CREATE FUNCTION broj_redaka
  (tablica_p VARCHAR2) RETURN NUMBER
IS
  naredba VARCHAR2 (1000);
  broj_redaka NUMBER;
BEGIN
  naredba := 'SELECT COUNT (*)' ||
             '  FROM ' || tablica_p;
  EXECUTE IMMEDIATE naredba
    INTO broj_redaka;
  RETURN broj_redaka;
END;
```

Prirodni dinamički SQL – dinamička kursor varijabla

```
DECLARE
  cur SYS_REFCURSOR; …
BEGIN
  naredba := 'SELECT ename, sal' ||
             '  FROM emp' ||
             ' WHERE job = :1';
  OPEN cur FOR naredba USING 'SALESMAN';
  LOOP
    FETCH cur INTO name, salary;
    EXIT WHEN cur%NOTFOUND; …
  END LOOP;
  CLOSE cur; …
```

Metoda 1, 2, 3, 4
kod rada sa dinamičkim SQL-om

- **Metoda 1** dozvoljava samo DML naredbe i to bez "bind" varijabli
- **Metoda 2** dozvoljava i DML naredbe sa "bind" varijablama
- **Metoda 3** dozvoljava i SELECT naredbu, koja mora imati fiksan broj stupaca i fiksan broj "bind" varijabli
- **Metoda 4** dozvoljava da broj stupaca i broj "bind" varijabli u SELECT naredbi bude nepoznat sve do trenutka izvođenja naredbe

Što DBMS_SQL može, a prirodni dinamički SQL ne može

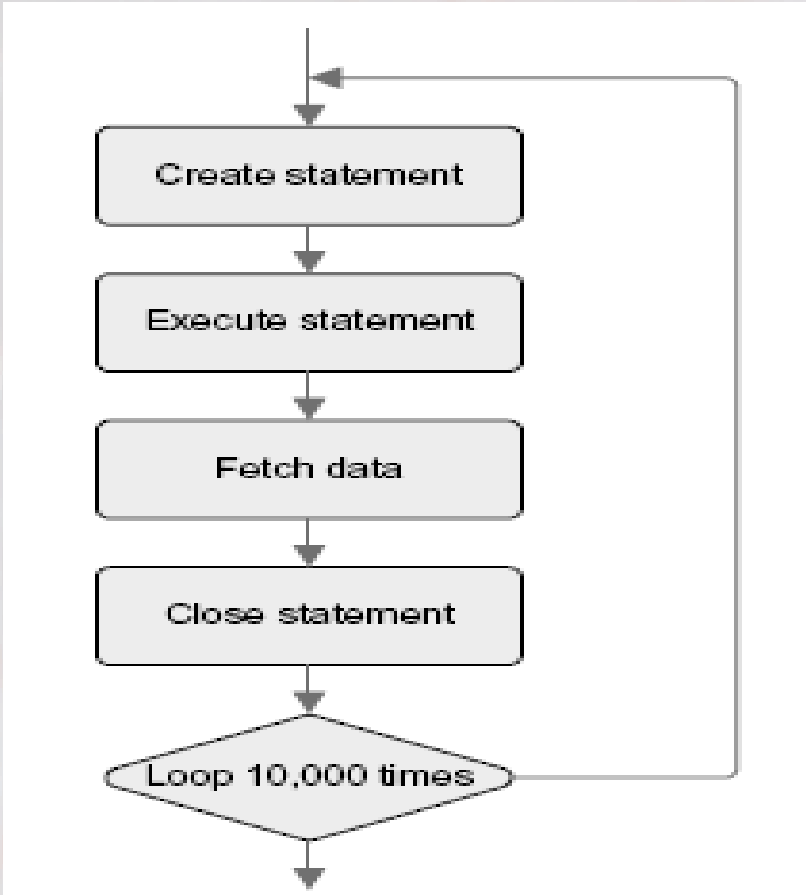
- paket **DBMS_SQL podržava sve 4 metode**
- prirodni dinamički SQL podržava Metodu 4 samo ako koristimo naredbu EXECUTE IMMEDIATE i to sa dinamičkim PL/SQL blokom, a ne sa dinamičkim SQL-om
- **dinamičke kursor varijable ne podržavaju Metodu 4**
- no, verziji 11g Oracle je uveo **nove naredbe** koje omogućavaju da dinamičke kursor varijable **indirektno podrže Metodu 4**

Različite varijante kursora i parsiranje

- Parsiranje (PARSE) kursora je jedan od koraka u procesiranju SQL naredbe.
- U ovisnosti od toga koliko dugo traju drugi koraci, vrijeme parsiranja može biti više ili manje značajno u cjelokupnom vremenu procesiranja naredbe.
- Kod parsiranja možemo imati tri situacije.
- Vremenski najgora (tj. najduža) situacija je tzv. tvrdo parsiranje. Tvrdo parsiranje se neizbježno mora desiti barem jedanput, ali ponavljanje tvrdog parsiranja može se izbjeći.
- Tvrdo parsiranje nije moguće dobiti sa statičkim SQL-om, tj. samo kod dinamičkog SQL-a moramo paziti da izbjegnemo tvrdo parsiranje, i to korištenjem bind varijabli.

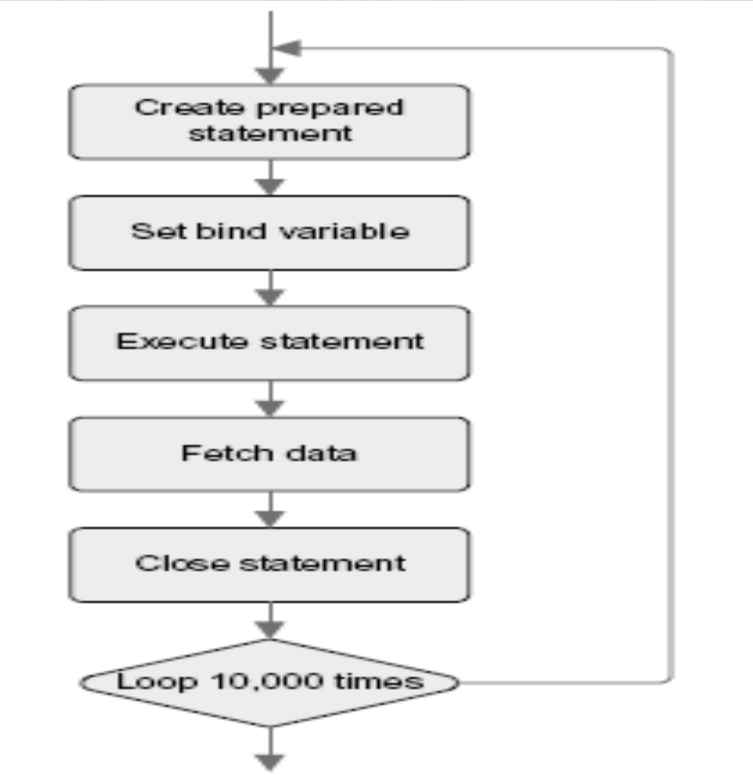
Tvrdo parsiranje

- Slika prikazuje petlju u kojoj se 10 000 puta izvršava naredba koja se svaki put tvrdo parsira:



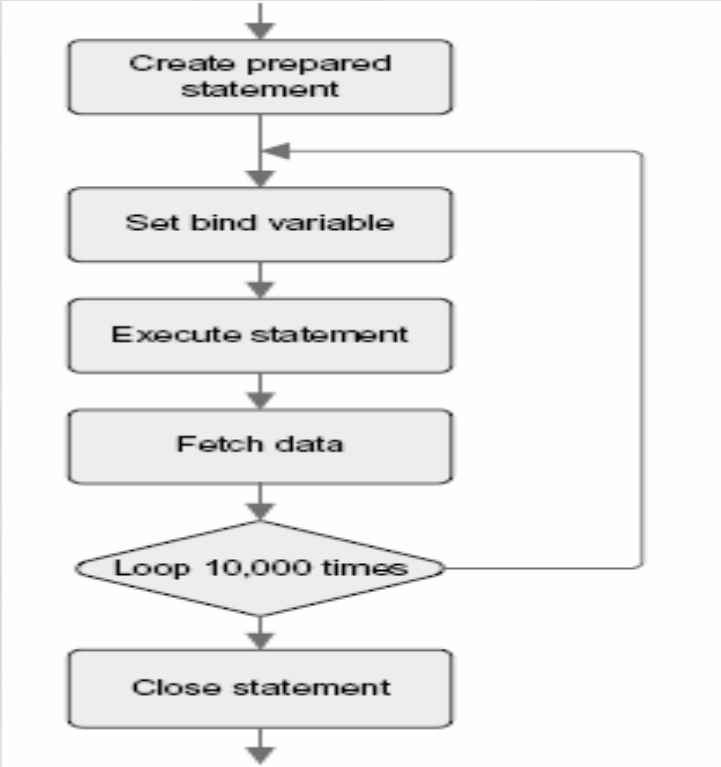
Meko parsiranje

- Kod mekog parsiranja se koriste djeljivi kursori u library cache memoriji. Na ovaj način radi statički SQL, a i dinamički SQL ako se koriste bind varijable:



Elimininiranje i mekog parsiranja

- Nabolja situacija je kada se eliminira i meko parsiranje - ponovno se koristi pripremljena naredba. Ovo je moguće isključivo korištenjem paketa DBMS_SQL:



Kako koristiti kursor varijablu nepoznate strukture

- pretpostavimo da imamo funkciju koja kao povratnu vrijednost (RETURN) vraća kursor varijablu koja je dobivena na temelju dinamički generiranog upita
- željeli bismo ju koristiti, ali kako napraviti FETCH kad **ne znamo kakva je njena struktura**?
- problem nije lako rješiv, jer kursor varijabla ne omogućava (direktan) opis svoje strukture
- do baze 11g jedino rješenje bilo je da funkcija vraća ne samo kursor varijablu, već i upit koji je funkcija (dinamički) generirala
- također, moraju se koristiti DBMS_SQL paket i njegove procedure PARSE i DESCRIBE_COLUMNS da bismo mogli napraviti nepoznatu record varijablu

Primjena novih naredbi Oracle 11g za opisivanje kursor varijable nepoznate strukture

- u bazi 11g također moramo koristiti paket DBMS_SQL i njegove procedure PARSE i DESCRIBE_COLUMNS
- no, funkcija, koja je dinamički generirala upit, više ne mora vratiti taj upit, već samo kursor varijablu
- to se postiže zahvaljujući novim naredbama iz paketa DBMS_SQL:
  TO_CURSOR_NUMBER i
  TO_REFCURSOR,
  pomoću kojih je moguće kursor varijablu (REF kursor) pretvoriti u DBMS_SQL kursor i obrnuto

Primjena novih naredbi Oracle 11g za opisivanje kursor varijable nepoznate strukture

```
…
-- kursor v. pretvara se u DBMS_SQL kursor
l_cur :=
  DBMS_SQL.TO_CURSOR_NUMBER (p_ref_cur);

-- istražuje se struktura DBMS_SQL kursora
DBMS_SQL.DESCRIBE_COLUMNS3
  (l_cur, g_count, l_desc_tab);

-- DBMS_SQL kursor pretvara se u kursor v.
p_ref_cur :=
  DBMS_SQL.TO_REFCURSOR (l_cur);
…
```

Zaključak

- kursor je neka vrsta **logičkog pokazivača** na skup redaka koji baza vraća kao odgovor na postavljeni upit
- u Oracle bazi "sve je kursor", pa i DML naredbe i SELECT naredba koja vraća samo jedan redak – to su **implicitni kursori**
- **eksplicitni kursori** mogu biti **statički ili dinamički**
- **kursor varijable** predstavljaju pokazivače na kursore i isto mogu biti statičke i dinamičke
- **DBMS_SQL kursor** je posebna vrsta dinamičkog k.
- u bazi 11g dobio je nove mogućnosti, a nove naredbe TO_CURSOR_NUMBER i TO_REFCURSOR omogućavaju da se i sa dinamičkim kursor varijablama **indirektno koristi Metoda 4** iz dinamičkog SQL-a