

# CURS 9

## PL/SQL II



# Manipularea înregistrărilor 1

- Tipul de dată RECORD permite manipularea simplă a datelor care se pot referi ca un întreg.
- În exemplul următor se consideră datele de bilanț corespunzătoare tabelelor de active și pasive ale unei companii și, folosind analiza ratelor se face compararea a două filiale ale acesteia:

DECLARE

TYPE BIL\_Rec IS RECORD (

cont varchar2(10) ,

cash REAL ...);

fil1\_bil bil\_Rec;

fil2\_bil bil\_Rec;

FUNCTION acid\_test (bil Bil\_Rec) RETURN REAL IS ...

BEGIN

SELECT cont, cash ... INTO fil1\_bil FROM active,pasive WHERE active.fil = 1 AND pasive.fil = 1;

SELECT cash, notes, ... INTO fil2\_bil FROM active,pasive WHERE active.fil = 2 AND pasive.fil = 2;

IF acid\_test(fil1\_bil) > acid\_test(fil2\_bil) THEN ... ..

END;

- este simplu a folosi datele de tip RECORD ca parametru pentru o funcție.



# Manipularea înregistrărilor 2

1. Se definește tipul obiect Passenger:

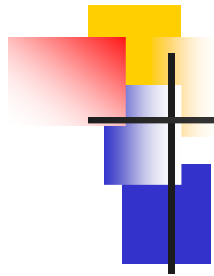
```
CREATE TYPE t_Passenger AS OBJECT(  
    flight_no NUMBER(3),  
    name VARCHAR2(20),  
    seat CHAR(5));
```

2. Se definește PassengerList de tip VARRAY care memorează obiectele Passenger:

```
CREATE TYPE t_PassengerList AS VARRAY(300) OF t_Passenger;
```

3. Se creează tabelul relațional *flights*, care are o coloană de tip t\_PassengerList:

```
CREATE TABLE flights (  
    flight_no NUMBER(3),  
    gate CHAR(5),  
    departure CHAR(15),  
    arrival CHAR(15),  
    passengers t_PassengerList);
```



## Manipularea înregistrărilor 3

- Fiecare coloană *passengers* este un varray care va stoca lista pasagerilor unui anumit zbor.
4. Se poate popula tabelul *flights*, astfel:

BEGIN

```
INSERT INTO flights VALUES(109, '80', 'DFW 6:35PM', 'HOU  
7:40PM', t_PassengerList(t_Passenger(109, 'Paula Trusdale', '13C'),  
t_Passenger(109, 'Louis Jemenez', '22F'), t_Passenger(109, 'Joseph  
Braun', '11B'), ...));
```

```
INSERT INTO flights VALUES(114, '12B', 'SFO 9:45AM', 'LAX  
12:10PM', t_PassengerList(t_Passenger(114, 'Earl Benton', '23A'),  
t_Passenger(114, 'Alma Breckenridge', '10E'), t_Passenger(114, 'Mary  
Rizutto', '11C'), ...));
```

```
INSERT INTO flights VALUES(27, '34', 'JFK 7:05AM', 'MIA 9:55AM',  
t_PassengerList(t_Passenger(27, 'Raymond Kiley', '34D'),  
t_Passenger(27, 'Beth Steinberg', '3A'), t_Passenger(27, 'Jean Lafevre',  
'19C'), ...));
```

END;



# Manipularea înregistrărilor 4

- Datele din tabelul flights pot fi extrase în înregistrarea flight\_info. -> se pot trata toate informațiile despre un zbor ca o unitate logică.

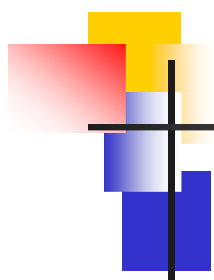
```
DECLARE
TYPE FlightRec IS RECORD (
    flight_no NUMBER(3),
    gate CHAR(5),
    departure CHAR(15),
    arrival CHAR(15),
    passengers t_PassengerList);
flight_info FlightRec;
CURSOR c1 IS SELECT * FROM flights;
seat_not_available EXCEPTION;
BEGIN
    OPEN c1;
    LOOP
        FETCH c1 INTO flight_info;
        EXIT WHEN c1%NOTFOUND;
        FOR i IN 1..flight_info.passengers.LAST
            LOOP
                IF flight_info.passengers(i).seat = 'NA' THEN
                    dbms_output.put_line(flight_info.passengers(i).name);
                    RAISE seat_not_available;
                END IF;
            ...
            END LOOP;
        END LOOP;
        CLOSE c1;
    EXCEPTION WHEN seat_not_available THEN ...
END;
```



# Interacțiunea dintre PL/SQL și SQL

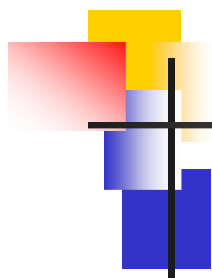
---

- Intr-un bloc PL/SQL se pot folosi instrucțiuni SQL pentru:
  - A regăsi rânduri din baza de date – SELECT
  - A modifica valori în rândurile bazei de date – instrucțiuni DML
  - A controla tranzacțiile – COMMIT, ROLLBACK, SAVEPOINT



# Caracteristici:

- PL/SQL suporta limbajul de manipulare a datelor si comenzile de control al tranzactiilor.
- Cuvantul cheie END semnaleaza sfarsitul unui bloc PL/SQL si NU finalul unei tranzactii.
- PL/QL nu suporta executia directa a instructiunilor din limbajul de definire al datelor (DDL). Acestea sunt instructiuni dinamice SQL. (Instructiunile dinamice SQL sunt construite ca siruri de caractere in timpul rularii si pot contine parametri ). Deci in PL/SQL se poate folosi SQL dinamic pentru a executa instructiuni DDL. In acest scop se foloseste instructiunea EXECUTE IMMEDIATE, care considera sirul ce contine instructiunea SQL ca argument pentru a o executa.
- PL/SQL nu suporta instructiuni de control al limbajului precum GRANT sau REVOKE. Pentru a le executa se foloseste de asemenea EXECUTE IMMEDIATE
- pentru a permanentiza modificarile efectuate in baza de date sau pentru a elimina rezultatele acestor modificari se utilizeaza COMMIT sau ROLLBACK. Se marcheaza un moment intermediar in prelucrarea unei tranzactii ca fiind o stare stabila a bazei de date prin utilizarea instructiunii SAVEPOINT. Aceste instructiuni se pot executa direct in PL/SQL



# Exemplu

Fișier sau URL:

Browse...

Încărcare script

Introducere instrucțiuni:

```
BEGIN
create table tabel1 as select * from employees;
END;
```

Execuție

Salvare script

Ștergere ecran

Renunțare

```
create table tabel1 as select * from employees;
*
```

EROARE la linia 2:

ORA-06550: linia 2, coloana 1:

PLS-00103: A fost întâlnit simbolul "CREATE" când era așteptată una din următoarele:

begin case declare exit for goto if loop mod null pragma

raise return select update while with <an identifier>

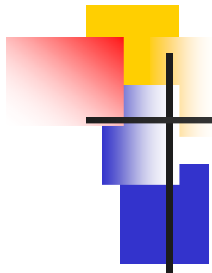
<a double-quoted delimited-identifier> <a bind variable> <<

close current delete fetch lock insert open rollback

savepoint set sql execute commit forall merge

<a single-quoted SQL string> pipe





Fișier sau URL:

Introducere instrucțiuni:

```
BEGIN  
EXECUTE IMMEDIATE 'create table tabell as select * from employees';  
END;
```

Procedură PL/SQL încheiată cu succes.



# Funcții SQL în PL/SQL

---

- Disponibile în instrucțiunile procedurale sunt cele de tip:
  - Număr pe un singur rând
  - Caracter pe un singur rând
  - Referitoare la conversii de date
  - Referitoare la date calendaristice
  - Referitoare la marci de timp – timestamp
  - GREATEST și LEAST
- Nu pot fi folosite în instrucțiuni procedurale
  - DECODE
  - Funcții de agregare (AVG, MIN, MAX, COUNT, SUM ) cu clauza GROUP



# Instrucțiunea SELECT în PL/SQL

- Sintaxa:

SELECT listă de câmpuri

**INTO {nume\_var1[, nume\_var2]...|nume\_înreg}**

FROM tabel

[WHERE conditii]

- Clauza INTO este obligatorie
- Interogarea trebuie să întoarcă numai un rând



# Exemplu 1

Introducere instrucțiuni:

```
SET SERVEROUTPUT ON
DECLARE
    fname    varchar2(20);
BEGIN
    SELECT first_name INTO fname
        FROM employees WHERE employee_id=205;
    DBMS_OUTPUT.PUT_LINE('Prenumele este: ' || fname);
END;
```

Execuție

Salvare script

Ștergere ecran

Renunțare

Prenumele este: Shelley

Procedură PL/SQL încheiată cu succes.



## Exemplu 2

Introducere instrucțiuni:

```
DECLARE
    emp_hiredate    employee.hire_date%TYPE;
    emp_salary      employee.salary%TYPE;
BEGIN
    SELECT hire_date, salary
        INTO emp_hiredate, emp_salary
        FROM employee WHERE employee_id=100;
    DBMS_OUTPUT.PUT_LINE (emp_hiredate);
    DBMS_OUTPUT.PUT_LINE (emp_salary);
END;
```

Execuție

Salvare script

Ștergere ecran

Renunțare

17-06-1987

35139,4

Procedură PL/SQL încheiată cu succes.

# Exemplu 3

Introducere instrucțiuni:

```
SET SERVEROUTPUT ON
DECLARE
    sum_sal number(10,2);
    deptno  number not null :=60;
BEGIN
SELECT sum(salary)
    INTO sum_sal
    FROM employee
    WHERE department_id=deptno;
DBMS_OUTPUT.PUT_LINE('Suma salariilor este: '|| sum_sal);
```

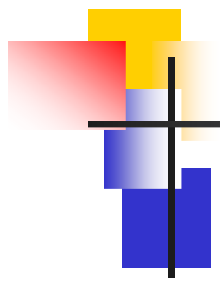
Execuție

Salvare script

Ștergere ecran

Renunțare

Suma salariilor este: 46059,25  
Procedură PL/SQL încheiată cu succes.



# Convenții de nume

- este recomandabil a fi folosite pentru a evita ambiguitățile în clauza WHERE
- trebuie evitată utilizarea numelor coloanelor din baza de date ca identificatori
  - pot sa apară erori de sintaxă datorita faptului ca PL/SQL verifica mai întâi existența coloanelor în tabelele bazei de date
    - numele variabilelor locale și parametrii formali au prioritate față de numele tabelelor bazei de date
    - numele coloanelor din tabelele bazei de date au prioritate asupra numelor variabilelor locale



# Manipularea datelor în PL/SQL

---

- Se folosesc instrucțiunile DML
  - INSERT
  - UPDATE
  - DELETE
  - MERGE





# Inserarea datelor

Fișier sau URL:

Introducere instrucțiuni:

```
BEGIN
  INSERT INTO employee
    (employee_id, first_name, last_name, email, hire_date, job_id,
     salary)
  VALUES (employees_seq.NEXTVAL, 'Ruth', 'Cores', 'RCORES',
sysdate, 'AD_ASST', 4000);
END;
/
```

- într-o instrucțiune INSERT dintr-un bloc PL/SQL se pot folosi:
- funcții – gen sysdate
  - secvențe pentru a genera valori pentru cheile primare



# Actualizarea datelor

Introducere instrucțiuni:

```
DECLARE
    sal_increase    employee.salary%TYPE :=800;
BEGIN
    UPDATE employee
        set salary = salary + sal_increase
        WHERE  job_id = 'ST_CLERK';
END;
/
```

Execuție

Salvare script

Ștergere ecran

Renu

Procedură PL/SQL încheiată cu succes.

# Ștergerea datelor

Fișier sau URL:

Browse...

Încărcare script

Introducere instrucțiuni:

```
DECLARE
    deptno  number not null :=10;
BEGIN
    DELETE FROM employees
    WHERE department_id=deptno;
END;
```

Execuție

Salvare script

Ștergere ecran

Renunțare

```
DECLARE
*
```

EROARE la linia 1:

ORA-02292: constrângerea de integritate (HR.DEPT\_MGR\_FK) violată - găsită înregistrarea copil

ORA-06512: la linia 4



# Cursoare SQL

- Oracle folosește zone de memorie pentru a executa instrucțiuni SQL și pentru a memora datele prelucrate
- Cursorul – permite denumirea unei zone de lucru și accesarea datelor conținute e aceasta
- Tipuri de cursoare
  - implicite – declarate de PL/SQL pentru toate instrucțiunile SQL – inclusiv pentru interogările care întorc un singur rând
  - explicite – declarate de utilizator atunci când o interogare întoarce mai multe rânduri rezultat și este necesară prelucrarea fiecărui rând în parte
- Un program PL/SQL:
  - deschide un cursor – care va extrage fiecare rând al mulțimii rezultat al unei interogări
  - prelucrează rândul curent extras în cursor
  - închide cursorul



# Atribute SQL pentru cursoare implicite

- Permit testarea rezultatului unei instrucțiuni SQL

SQL%FOUND	Atribut boolean evaluat ca fiind TRUE dacă ultima instrucțiune SQL întoarce cel puțin un rând
SQL%NOTFOUND	Atribut boolean evaluat ca fiind TRUE dacă ultima instrucțiune SQL nu întoarce nici un rând
SQL%ROWCOUNT	Valoare întreagă care reprezintă numărul rândurilor afectate de cea mai recentă instrucțiune SQL

- Atributele pot fi testate în secțiunea executabilă a unui bloc pentru a primi informații referitoare la comanda DML
  - PL/SQL nu întoarce eroare dacă o instrucțiune DML nu afectează nici un rând în tabel
    - este posibil ca un SELECT care nu întoarce nici o înregistrare să genereze o excepție
- Sunt prefixate cu SQL deoarece cursoarele implicite sunt generate automat de PL/SQL și nu li se cunoaște numele

# Exemplu

Fișier sau URL:

Browse...

Încărcare script

Introducere instrucțiuni:

```
variable randuri_sterse varchar2(30)

DECLARE
empno employee.employee_id%TYPE :=172;
BEGIN
    DELETE FROM employee
    WHERE employee_id=empno;
    :randuri_sterse:=(SQL%ROWCOUNT || ' rand sters. ');
END;
/
```

Execuție

Salvare script

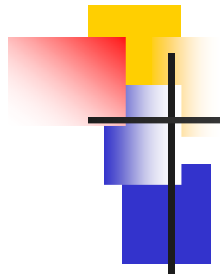
Ștergere ecran

Renunțare

Procedură PL/SQL încheiată cu succes.

RANDURI\_STERSE

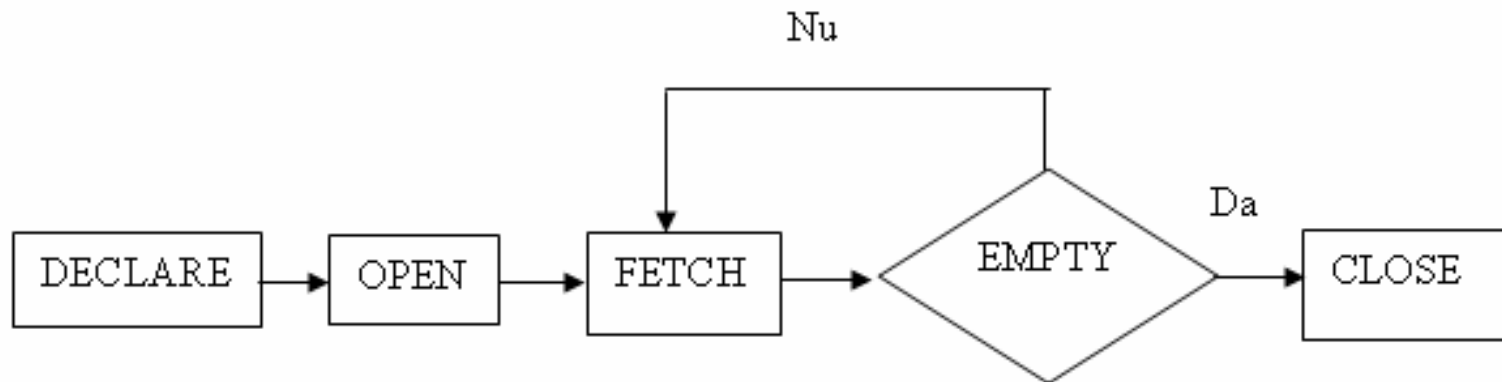
1 rand sters.



# Utilizarea cursoroarelor explicite

- Permit:
  - procesarea rând cu rând a rezultatului unei interogări
  - Umărește randul care este prelucrat la un moment dat
  - Permite programatorului să controleze manual cursoroarele explicite în blocul PL/SQL

# Comenzi de control pentru cursoare



Se crează o zonă pentru SQL cu un anumit nume

Se identifică mulțimea activă (rezultatul întors de SQL)

Se încarcă rândul curent în variabile

Se testează dacă mai există rânduri în setul activ - dacă DA se reia FETCH

Se eliberează setul activ de rânduri



Introducere instrucțiuni:

```
DECLARE
  CURSOR emp_curs IS
    SELECT employee_id, last_name FROM employee
    WHERE department_id=30;
  empno employee.employee_id%TYPE;
  lname employee.last_name%TYPE;
BEGIN
  OPEN emp_curs;
  FETCH emp_curs INTO empno, lname;
  DBMS_OUTPUT.PUT_LINE(empno || ' ' || lname);
end;
```

Execuție

Salvare script

Ștergere ecran

Renunțare

114 Raphaely

Procedură PL/SQL încheiată cu succes.

Introducere instrucțiuni:

```
empno employee.employee_id%TYPE;  
lname employee.last_name%TYPE;  
BEGIN  
OPEN emp_curs;  
  LOOP  
    FETCH emp_curs INTO empno, lname;  
    DBMS_OUTPUT.PUT_LINE(empno || ' ' || lname);  
    exit when emp_curs%NOTFOUND;  
  end loop;  
end;
```

Execuție

Salvare script

Ștergere ecran

Renunțare

114 Raphaely

115 Khoo

116 Baida

117 Tobias

118 Himuro

# Cursoare si înregistrări

- Se pot procesa rândurile dintr-un set activ prin extragerea valorilor într-o variabila de tip RECORD

Introducere instrucțiuni:

```
CURSOR emp_curs IS
  SELECT employee_id, last_name FROM employee
  WHERE department_id=30;
emp_rec emp_curs%ROWTYPE;
BEGIN
  OPEN emp_curs;
  LOOP
    FETCH emp_curs INTO emp_rec;
    DBMS_OUTPUT.PUT_LINE(emp_rec.employee_id||' '||emp_rec.last_name);
    exit when emp_curs%NOTFOUND;
  end loop;
```

Execuție

Salvare script

Ștergere ecran

Renunțare

114 Raphaely

115 Khoo

116 Baida

117 Tobias

118 Himuro

119 Colmenares

119 Colmenares

Procedură PL/SQL încheiată cu succes.



# Cursoare FOR... LOOP

- Simplifică scrierea codului prin eliminarea instrucțiunilor OPEN, FETCH, LOOP
- declară implicit indexul de buclare ca fiind o înregistrare care reprezintă un rând extras din baza de date
  - apoi deschide cursorul
  - extrage rândurile din baza de date
  - închide cursorul după prelucrarea tuturor rândurilor
- Sintaxa:

```
FOR nume_rec IN nume_cursor LOOP  
    instructiune  
    instructiune  
    .....  
END LOOP;
```

# Exemplu

Introducere instrucțiuni:

```
DECLARE
  CURSOR emp_curs IS
    SELECT employee_id, last_name FROM employee
    WHERE department_id=30;
BEGIN
  FOR emp_r IN emp_curs
  LOOP
    DBMS_OUTPUT.PUT_LINE (emp_r.employee_id||' '||emp_r.last_name);
  end loop;
end;
```

Execuție

Salvare script

Ștergere ecran

Renunțare

114 Raphaely

115 Khoo

116 Baida

117 Tobias

118 Himuro

119 Colmenares

Procedură PL/SQL încheiată cu succes.

- Pentru a referi câmpurile individuale într-o înregistrare se utilizează notația *dot*, unde in simbolul (.) este selector al componentei



# Atribute pentru cursoare explicite

- Furnizează informații de stare despre un cursor

Atribut	Tip	Descriere
%ISOPEN	Boolean	Evaluat ca TRUE dacă cursorul este deschis
%NOTFOUND	Boolean	Evaluat ca TRUE dacă cel mai recent FETCH nu întoarce nici un rând
%FOUND	Boolean	Evaluat ca TRUE dacă ultimul FETCH întoarce un rând
%ROWCOUNT	Număr	Evaluat la numărul de rânduri întoarse



# Variabile cursor

---

- Indică rândul curent din mulțimea rezultat a unei interogări
  - Poate fi deschisă pentru orice interogare compatibilă ca tip
  - Nu este legată de o anume definiție (interogare)
  - Se comporta ca variabile PL/ SQL cărora li se pot atribui valori noi si care pot fi transmise ca parametri procedurilor stocate în baza de date
- În mod obișnuit, o variabilă cursor se deschide prin transmiterea sa unei proceduri stocate care declară o variabilă cursor ca parametru formal



# Exemplu

---

```
PROCEDURE open_vc (generic_vc IN OUT t_GenericCur, choice NUMBER) IS
BEGIN
    IF choice = 1 THEN
        OPEN generic_vc FOR SELECT * FROM employees;
    ELSIF choice = 2 THEN
        OPEN generic_vc FOR SELECT * FROM departments;
    ELSIF choice = 3 THEN
        OPEN generic_vc FOR SELECT * FROM job_history;
    END IF;
    ...
END;
```





# Attribute

---

- Pot fi asociate variabilelor PL/SQL și cursoarelor
  - proprietăți care permit referirea tipului de dată și structurii unui articol fără a-i repeta definiția
  - Sunt specificate cu ajutorul simbolului %



# Atributul %TYPE

---

- Furnizează tipul de data al unei variabile sau al unei coloane
- Este util când se declară variabile ce vor lua valori din baza de date
- Avantaje:
  - Nu trebuie cunoscut exact tipul datei ce se va atribui variabilei
  - Dacă se schimbă tipul datei în baza de date acest lucru va fi reflectat și în variabila declarată cu atributul %TYPE
- Exemplu:



# Atributul %ROWTYPE

---

- În PL/SQL înregistrările sunt folosite pentru a grupa datele
- Constă într-un număr de câmpuri
- Atributul %ROWTYPE furnizează un tip înregistrare care reprezintă un rând într-un tabel
- Înregistrarea poate stoca un rând întreg de date selectate din tabel sau extrase dintr-un cursor sau variabilă cursor
- Coloanele unui rând și câmpurile dintr-o înregistrare au aceleași nume și aceleași tipuri de dată



# Exemplu

DECLARE

depart\_rec department.%ROWTYPE;

Pentru a referi un anumit câmp se folosește notația *dot*

my\_dep\_id := depart\_rec.department\_id;

- Dacă se declară un cursor care regăsește câmpurile: *last\_name*, *hire\_date* și *job\_id* din *employees*, se poate folosi apoi %ROWTYPE pentru a declara o variabilă *record* care stochează aceeași informație

DECLARE

CURSOR c1 IS SELECT last\_name, hire\_date, job\_id FROM employee;

emp\_rec c1%ROWTYPE;      -- se declară o variabilă record care reprezintă  
                              -- un rând extras din tabelul employee

- Când se execută

FETCH c1 INTO emp\_rec;

- valoarea din coloana *last\_name* a tabelului *employee* este asignată câmpului *last\_name* din *emp\_rec*, valoarea din coloana *hire\_date* este asignată câmpului *hire\_date* din *emp\_rec*, s.a.m.d.