

Laboratorio di PL/SQL 3

Esercizi svolti e valutati

Prof. Alessandra Lumini
Alma Mater Studiorum - Università di Bologna

Per la sintassi PL/SQL:
➤ ORACLE 11g Rel. 2 – PL/SQL Language Reference

1

Perché usare una stored procedure?

- Partiamo da un esempio:
 - Il sistema informativo della scuola di sci *DiscesaLibera* si basa sul seguente DB
 - **TIPICORSI**(IDCorso, Nome, Livello, EtaMin, EtaMax, MinPartecipanti)
 - **ALLIEVI**(IDAllievo, Nome, Eta, Livello, SettimanaRichiesta)
 - **ASSEGNAMENTI**(Corso:TIPICORSI, Allievo:ALLIEVI)
 - ciascun allievo è associabile a un solo corso
 - le richieste sono divise per settimane
 - Si deve progettare la procedura *Assegna(vSettimana)* che ogni venerdì
 - azzera la tabella *Assegna*
 - scrive gli assegnamenti ai corsi della settimana successiva (*SettimanaRichiesta*) in base alle proprie caratteristiche: l'età dell'allievo deve ricadere nel range del corso e i livelli devono coincidere.
 - nel caso in cui per uno specifico corso non si raggiunga il numero minimo di partecipanti il corso non viene effettuato e nessuna tupla deve essere inserita.

2

Soluzione con connessione al DB

- ❑ Apri una connessione con il DB
- ❑ Leggi la tabella dei TIPI Corsi
 - Per ogni corso attivabile 
 - Interroga il DB per contare il numero NI di potenziali iscritti per la settimana 
 - Se NI > MinPartecipanti
 - Scorri i potenziali iscritti sulla tabella ALLIEVI 
 - Inseriscili nella tabella ASSEGNAMENTI 
- ❑ Chiudi la connessione

❑ Complessità

- DB: scansione tabella TIPI Corsi, $NT_{TIPI Corsi}^*$ query su tabella ALLIEVI, $NT_{TIPI Corsi}^*$ scansione tabella ALLIEVI, scritture su ASSEGNAMENTI
- RETE: intera tabella TIPI Corsi, $NT_{TIPI Corsi}^*$ intera tabella ALLIEVI, intera tabella ASSEGNAMENTI

3

Soluzione con stored procedure

- ❑ Dichiarare un cursore tutti i TIPI Corsi
- ❑ Dichiarare un cursore parametrico che seleziona gli allievi in base alle condizioni del corso
 - Per ogni corso nel cursore 
 - Verifica della condizione richiesta 
 - Per ogni allievo nel cursore 
 - Inseriscili nella tabella ASSEGNAMENTI 

❑ Complessità

- DB: scansione dei due cursori, $NT_{TIPI Corsi}^*$ query su tabella ALLIEVI, scritture su ASSEGNAMENTI
- RETE: --

4

```

create or replace procedure Assegna(vSettimana int) IS
--cursori
cursor cCorsi is
select * from TIPICORSI
order by T_Livello,T_EtaMin;

cursor cAllievi(iLivello int,iEtaMin int,iEtaMax int) is
select * from ALLIEVI
where A_SettimanaRichiesta=vSettimana and A_Eta>=iEtaMin and A_Eta<=iEtaMax
and A_Livello=iLivello;

vNumAllievi int;

begin
--primo cursore
FOR vCorsi IN cCorsi
LOOP
select count(*) into vNumAllievi from ALLIEVI
where A_SettimanaRichiesta=vSettimana and A_Eta>=vCorsi.T_EtaMin
and A_Eta<=vCorsi.T_EtaMax and A_Livello=vCorsi.T_Livello;

if (vNumAllievi>vCorsi.T_MinPartecipanti) then
--secondo cursore
FOR vAllievi IN cAllievi(vCorsi.T_Livello, vCorsi.T_EtaMin,vCorsi.T_EtaMax)
LOOP
INSERT INTO ASSEGNAMENTI VALUES (vCorsi.T_IDCorso,vAllievi.A_IDAllievo);
END LOOP;
end if;
END LOOP;
end;

```

5

Soluzione alternativa

- ❑ Dichiarare un cursore che seleziona solo i TIPICORSI che soddisfano la condizione richiesta
- ❑ Dichiarare un cursore parametrico che seleziona gli allievi in base alle condizioni del corso
 - Per ogni corso attivabile nel cursore 
 - Per ogni allievo nel cursore 
 - Inseriscili nella tabella ASSEGNAMENTI 
- ❑ Complessità
 - DB: scansione dei due cursori, scritture su ASSEGNAMENTI
 - RETE: --

6

```

create or replace procedure Assegna(vSettimana int) IS
--cursori
cursor cCorsi is
select T_IdCorso, T_Livello, T_EtaMin,T_EtaMax from Tipicorsi, Allievi
where T_Livello = A_Livello and T_EtaMin <= A_Eta and T_EtaMax >= A_Eta and
A_SettimanaRichiesta = vSettimana
group by T_IdCorso, T_Livello, T_EtaMin,T_EtaMax
having count(*) >= T_MinPartecipanti;

cursor cAllievi(iLivello int,iEtaMin int,iEtaMax int) is
select * from ALLIEVI
where A_SettimanaRichiesta=vSettimana and A_Eta>=iEtaMin
and A_Eta<=iEtaMax and A_Livello=iLivello;

vNumAllievi int;
begin
FOR vCorsi IN cCorsi
LOOP
FOR vAllievi IN cAllievi(vCorsi.T_Livello, vCorsi.T_EtaMin,vCorsi.T_EtaMax)
LOOP
INSERT INTO ASSEGNAMENTI VALUES (vCorsi.T_IDCorso,vAllievi.A_IDAllievo);
END LOOP;
END LOOP;
end;

```

7

Soluzione più compatta

- Dichiarare un cursore che seleziona gli ALLIEVI e i relativi TIPICORSI (solo per i TIPICORSI che soddisfano la condizione richiesta)
 - Per ogni allievo nel cursore 
 - Inseriscili nella tabella ASSEGNAMENTI 

- Complessità
 - DB: scansione di un cursore, scritture su ASSEGNAMENTI
 - RETE: --

8

```

create or replace procedure Assegna (vSettimana int) IS

cursor cAllievi is
select * from ALLIEVI, TIPICORSI
where A_SettimanaRichiesta=vSettimana and A_Eta>=T_EtaMin and A_Eta<=T_EtaMax
and A_Livello=T_Livello AND
T_IDCorso IN (select T.T_IDCorso from ALLIEVI A, TIPICORSI T
where A.A_SettimanaRichiesta=vSettimana and A.A_Eta>=T.T_EtaMin and
A.A_Eta<=T.T_EtaMax and A.A_Livello=T.T_Livello
group by T.T_IDCorso
having count(*)>T.T_MinPartecipanti);

begin
FOR vAllievi IN cAllievi
LOOP
INSERT INTO ASSEGNAMENTI VALUES (vAllievi.T_IDCorso,vAllievi.A_IDAllievo);
END LOOP;
end;

```

9

Quale soluzione?

- La soluzione migliore è in genere quella che sfrutta meglio la potenza del DB = fa fare il lavoro di ottimizzazione all'ottimizzatore

10

Vantaggi dell'uso di Stored Procedure

- ❑ Sfruttare al meglio le potenzialità dell'ottimizzatore
- ❑ Riduzione del traffic di rete
- ❑ Possibilità di modulare meglio i problemi
- ❑ Operazioni che sono più efficienti in una stored procedure
 - Trovare massimo o minimo di una funzione
 - Eseguire un ordinamento
 - Eseguire aggiornamenti correlati
 - Eseguire query in cui ci siano sottoquery ripetute

11

Ottimizzazione del codice

- ❑ Statement LOOP e IF
 - Minimizzare il numero di iterazioni
 - Usare la clausola EXIT se necessario
 - Rimuovere dall'interno le LOOP le istruzioni che possono essere eseguite fuori
 - In caso di IF innestati specificare prima la condizione più probabile
- ❑ Evitare la ricorsione
- ❑ Array processing
 - Lo statement FORALL permette di ottimizzare gli inserimenti

```
FORALL i IN 1 .. 30000
  INSERT INTO insert_test (x, y)
  VALUES (l_numbers (i), l_varchars (i));
COMMIT;
```

12

Ottimizzazione del codice

□ WHERE CURRENT OF cursor

- In un LOOP di un cursore la clausola "WHERE CURRENT OF" permette di aggiornare direttamente il record corrente (più rapido che non la ricerca per chiave primaria)
- Il cursore deve essere definito "FOR UPDATE"

```
CURSOR sales_csr
IS SELECT *
   FROM sales
   WHERE sale_date > '01-DEC-99'
   FOR UPDATE; -- Need FOR UPDATE since using
               -- WHERE CURRENT OF
BEGIN
  FOR sale_row IN sales_csr
  LOOP
    IF sale_row.sale_value > 4000 THEN
      l_gstflag:=complex_function(sale_row.sale_value);
      UPDATE sales
      SET gst_flag=l_gstflag
      WHERE CURRENT OF sales_csr;
      l_sales_tot:=l_sales_tot+sale_row.sale_value;
    END IF;
  END LOOP;
```

13

Ottimizzazione del codice

□ Cursori espliciti

- Sebbene si possa usare direttamente una query SQL in una procedura, la creazione di un cursore è più efficiente

```
/* Implicit cursor */
BEGIN
  SELECT customer_id
     INTO g_customer_id
     FROM customers c
     WHERE contact_surname = g_surname
     AND contact_firstname = g_firstname;
END;

/* Explicit cursor */
DECLARE
  CURSOR customer_csr
  IS
    SELECT customer_id
       FROM customers c
       WHERE contact_surname = g_surname
       AND contact_firstname = g_firstname;
BEGIN
  OPEN customer_csr;
  FETCH customer_csr INTO g_customer_id;
  CLOSE customer_csr;
END;
```

Un cursore implicito (query SQL) richiede un secondo fetch per assicurare che il risultato sia una sola riga

14

Un altro esempio: il calcetto

- Una partita di calcetto è bella se è equilibrata! Per questo motivo un gruppo di amici ha costruito il seguente DB per generare automaticamente le formazioni
 - **GIOCATORI**(ID, Nome, Cognome, LivelloTecnico, LivelloAtletico, Ruolo)
 - **PARTITE**(IDPartita, DataPartita)
 - **DISPONIBILITA**(IDPartita:PARTITE, IDGiocatore:GIOCATORI)
 - **FORMAZIONI**(IDPartita:PARTITE, IDGiocatore:GIOCATORI,IDSquadra)

- Si scriva la procedura GeneraSquadra(IDPartita) che suddivide i giocatori disponibili per la partita IDPartita in due squadre (IDSquadra ∈ [1,2]) popolando la relazione FORMAZIONI.
 - I ruoli possibili sono 3: *Attaccante, Difensore, Portiere*.
 - Devono essere disponibili almeno 10 giocatori altrimenti la procedura viene interrotta e viene sollevata una exception con relativo messaggio di warning.
 - Per ogni ruolo i giocatori disponibili sono ordinati in base alla formula (LivelloTecnico * 1.2 + LivelloAtletico) e assegnati alternativamente alla squadra 1 e 2 (se per gli attaccanti il primo assegnamento è fatto alla squadra 1, per i difensori il primo assegnamento dovrà essere fatto alla squadra 2 e così via).

15

Requisiti di funzionamento

- Requisiti di funzionamento
 - Progettare un cursore per scorrere i giocatori disponibili
 - Ordinare i giocatori in base a 2 condizioni: Ruolo e Capacità di gioco (LivelloTecnico * 1.2 + LivelloAtletico)
 - Gestire l'alternanza tra le squadre
 - Gestire il cambio di ruoli
 - Effettuare correttamente l'inserimento
 - Inserire una condizione di uscita (exception)

- Requisiti di ottimalità
 - Uso corretto del LOOP per il cursore
 - Evitare di duplicare il codice per le 2 squadre
 - Evitare l'uso di query inutili
 - ...

16

Traccia di Soluzione

- ❑ Dichiarare un cursore che seleziona i giocatori disponibili ordinati per ruolo e capacità
- ❑ Consegna i giocatori disponibili 
 - Se sono troppo pochi solleva un'eccezione
 - Altrimenti
 - Inizializza variabili per assegnamento (SquadraCorrente, RuoloCorrente)
 - Per ogni giocatore nel cursore 
 - Se c'è un cambio di ruolo
 - » Modifica le variabili di assegnamento
 - Inserisci il giocatore nella squadra corrente 
 - Modifica la squadra corrente
- ❑ Gestisci l'eccezione
- ❑ **Complessità**
 - scansione dei cursori, query su tabella **DISPONIBILITA**, inserimento **FORMAZIONI**.

17

```
create or replace procedure GeneraSquadra(IDPartita int) IS
cursor cDisp is --cursore
select G_ID, G_Ruolo, (G_LivelloTecnico * 1.2 + G_LivelloAtletico) AS Valore
from DISPONIBILITA, GIOCATORI where D_IDPartita=IDPartita and D_IDGiocatore=G_ID
order by 2, 3 DESC;
vSquadra int; vPrimaSquadra int; nGioc int; --variabili
vRuolo varchar2(10); notEnoughPlayers exception;
begin
select count(*) into nGioc from DISPONIBILITA where D_IDPartita = IDPartita;
if nGioc<10 then --check condizione
raise notEnoughPlayers;
else
vPrimaSquadra:=1; --inizializza
vSquadra:=vPrimaSquadra;
vRuolo:='Attaccante';
FOR vDisp IN cDisp --cicla su cursore
LOOP
if (vDisp.G_Ruolo != vRuolo) then
vPrimaSquadra:=mod(vPrimaSquadra,2)+1; --cambio di ruolo
vSquadra:=vPrimaSquadra;
vRuolo:=vDisp.G_Ruolo;
end if;
INSERT into FORMAZIONI values (IDPartita,vDisp.G_ID,vSquadra);
vSquadra:=mod(vSquadra,2)+1; --squadra successiva
END LOOP;
end if;
EXCEPTION
WHEN notEnoughPlayers THEN
DBMS_OUTPUT.PUT_LINE('Non ci sono abbastanza giocatori disponibili ');
end;
```

18

Soluzione Alternativa

- ❑ Dichiarare 3 cursori (1 per ruolo) che selezionano i giocatori disponibili ordinati per capacità
- ❑ Conteggia i giocatori disponibili 
 - Se sono troppo pochi solleva un'eccezione
 - Altrimenti
 - Inizializza variabili per assegnamento (SquadraCorrente, RuoloCorrente)
 - Ripete per ogni ruolo le seguenti operazioni
 - Per ogni giocatore nel cursore 
 - » Inserisci il giocatore nella squadra corrente 
 - » Modifica la squadra corrente
- ❑ Gestisci l'eccezione
- ❑ **Complessità**
 - scansione di 3 cursori, query su tabella **DISPONIBILITA**, inserimento **FORMAZIONI**.

19

```
create or replace procedure GeneraSquadra(IDPartita int) is
cursor c_attaccanti is select g_id
from disponibilita, giocatori
where d_idgiocatore = g_id and d_idpartita = idpartita and g_ruolo = 'Attaccante'
order by (g_livellotecnico * 1.2 + g_livelloatletico) desc;

cursor c_difensori is . . .
cursor c_portieri is . . .
v_count int := 0; v_squadra int; pochi_giocatori exception;
begin
select count(*) into v_count from disponibilita where d_idpartita = IDPartita;
if v_count < 10 then raise pochi_giocatori; end if;
--assegna gli attaccanti
v_squadra int := 1;
for vr_gioc in c_attaccanti loop
insert into formazioni values (idpartita, vr_gioc.g_id, v_squadra);
if v_squadra = 1 then v_squadra := 2; else v_squadra := 1; end if;
end loop;
--assegna i difensori poi i portieri
v_squadra := 2;
for vr_gioc in c_difensori loop . . .

exception
when NO_DATA_FOUND or pochi_giocatori then
DBMS_OUTPUT.PUT_LINE('Pochi giocatori');
end;
```

20

Errori comuni

- ❑ Errori nelle query SQL
- ❑ Assenza di filtri nel cursore (tutti i giocatori, non solo quelli disponibili)
- ❑ Errore nell'ordinamento dei giocatori
- ❑ Assenza di alternanza tra le squadre nel cambio di ruoli
- ❑ Assenza gestione eccezione

21

Un altro esempio: i compiti

- ❑ Il sistema automatico per la correzione dei compiti universitari si basa su una banca dati di domande con diversi livelli di complessità (1, 2, 3, 4) e diverse aree (Area1, Area2, Area3). Un compito è formato da N domande estratte dal sistema. Per ogni domanda il sistema mostra le diverse possibili risposte e lo studente ne sceglie una che viene registrata sul DB
 - **DOMANDE**(ID, Testo, RispostaCorretta, LivelloComplessità, AreaDomanda)
 - **RISPOSTE**(IDDomanda:DOMANDE, IDRisposta, TestoRisposta)
 - **RISPOSTESTUDENTE**(IDStudente, IDCompito, IDDomanda: DOMANDE, Risposta)
- ❑ Si scriva la procedura **CorreggiCompito**(IDStudente, IDCompito) che calcola e mostra a video il voto dello studente in base alle seguenti regole:
 - Si calcola un voto parziale per ognuna delle tre aree (Area1, Area2, Area3) e il voto complessivo è la media pesata dei voti delle tre aree Area1 – peso 1, Area2 – peso 2, Area3 – peso 3.
 - Per le domande di un'area:
 - Se la risposta è sbagliata il punteggio è -0.5
 - Se la risposta è corretta (RISPOSTESTUDENTE.Risposta = DOMANDE.RispostaCorretta) il punteggio acquisito è dato dal campo LivelloComplessità

22

Requisiti di funzionamento

- ❑ Requisiti di funzionamento
 - Progettare un cursore per scorrere le risposte degli studenti
 - Filtrare solo il compito e lo studente selezionati
 - Ordinare le risposte in base all'area
 - Calcolare i voti parziali
 - Calcolare il totale
 - Effettuare la stampa a video
- ❑ Requisiti di ottimalità
 - Uso corretto del LOOP per il cursore
 - Evitare di duplicare il codice per le diverse aree squadre
 - Evitare l'uso di query/join inutili
 - ...

23

Traccia di Soluzione

- ❑ Dichiarare un cursore che seleziona le soluzioni relative al compito selezionato e allo studente selezionato **ordinate** per area
- ❑ Inizializza variabili per assegnamento (AreaCorrente, ParzialeArea, ...)
- ❑ Per ogni soluzione nel cursore 
 - Se c'è un cambio di area
 - Stampa i dati relativi all'Area corrente
 - Modifica le variabili di assegnamento
 - Aggiorna i risultati dell'area corrente
- ❑ Stampa il risultato finale
- ❑ **IN ALTERNATIVA (senza ordinamento):**
 - Un solo cursore e gestione delle aree con l'uso di **if** in cascata o **case..when**
- ❑ **Complessità**
 - scansione del cursore

24

```

create or replace procedure CorreggiCompito(IDStudiante int, IDCompito int) IS
cursor cDom is --cursore
select * from RISPOSTESTUDENTE, DOMANDE
where RS_IDDomanda=D_ID and RS_IDStudiante=IDStudiante AND RS_IDCompito=IDCompito
order by D_AreaDomanda;
vParzArea float; vArea int; vNumAree int; vNum int; vTot float;
begin
vArea:=1; vParzArea:=0; vTot:=0; vNum:=0; vNumAree:=0;
FOR vDom IN cDom
LOOP
if ((vDom.D_AreaDomanda>vArea) and (vNum>0)) then
DBMS_OUTPUT.PUT_LINE('Area: ' || vArea || 'Parziale:' || vParzArea/vNum);
vTot := vTot + vArea*vParzArea/vNum;
vNumAree:=vNumAree+1; vParzArea:=0; vArea:=vDom.D_AreaDomanda; vNum:=0;
end if;
--aggiorna conteggi parziali
vNum:=vNum+1;
if (vDom.D_RispostaCorretta=vDom.RS_Risposta) then
vParzArea:=vParzArea+vDom.D_LivelloComplessità;
else
vParzArea:=vParzArea-0.5;
end if;
END LOOP;
--stampe finali
DBMS_OUTPUT.PUT_LINE('Area: ' || vArea || 'Parziale:' || vParzArea/vNum);
vTot := vTot + vArea*vParzArea/vNum; vNumAree:=vNumAree+1;
DBMS_OUTPUT.PUT_LINE('Totale: ' || vTot/vNumAree);
end;
```

25

Soluzione alternativa

- ❑ Dichiarare 3 cursori, uno per ciascuna area, che selezionano le soluzioni relative al compito selezionato e allo studente selezionato
- ❑ Ripetere iterativamente per i tre cursori 
 - Inizializza variabili per assegnamento (ParzialeArea)
 - Per ogni soluzione nel cursore
 - Aggiorna i risultati dell'area corrente
 - Stampa i dati relativi all'Area corrente
- ❑ Stampare il risultato finale

- ❑ **Complessità**
 - scansione dei 3 cursori

26