

Cursul 2 - PL/SQL. Concepte generale

SQL	PL/SQL
Este un limbaj non-procedural (declarativ) - transmite server-ului de baze de date ce să facă, nu cum să facă.	Reprezintă extensia procedurală a limbajului SQL - transmite server-ului de baze de date cum să facă.
Este un limbaj 4GL (mai apropiat de limbajul natural decât de limbajul de programare)	Este un limbaj 3GL (de nivel înalt, convertibil în limbaj mașină de un compilator)
În SQL poate fi executată o singură interogare la un moment dat	În PL/SQL poate fi executat un bloc de cod
În SQL, interogările pot utiliza instrucțiuni LMD, LDD, LCD.	În PL/SQL, blocurile de cod pot utiliza proceduri, funcții, pachete, variabile, etc.
Interogările SQL sunt utilizate pentru a extrage date din baza de date, dar și pentru a adăuga, șterge sau modifica date.	PL/SQL este utilizat pentru a crea aplicații care pot afișa informația extrasă de SQL.
Interogările SQL nu pot conține cod PL/SQL	Blocurile PL/SQL pot conține cod SQL
Interacționează direct cu server-ul bazei de date	Nu interacționează direct cu server-ul bazei de date

Motorul PL/SQL

- compilează și execută codul PL/SQL;
- se află pe server-ul Oracle sau în unele utilitare Oracle;
- unele utilitare au propriul motor PL/SQL, care execută local blocurile care nu conțin comenzi SQL;
- indiferent de mediu, motorul PL/SQL execută comenzile procedurale, dar trimite comenzile SQL către motorul SQL de pe server-ul Oracle;
- apelurile procedurilor care sunt stocate pe server sunt trimise pentru procesare motorului PL/SQL de pe server.

Cursul 3 - Blocuri. Variabile. Instrucțiuni

Comenzi SQL în PL/SQL

Comenzile LMD (SELECT, INSERT, UPDATE, DELETE, MERGE) și comenzile LCD COMMIT, SAVEPOINT și ROLLBACK pot fi utilizate direct în PL/SQL. (Comanda SELECT poate fi utilizată doar cu clauza INTO).

Comenzile LDD (CREATE, ALTER, DROP) și comenzile LCD GRANT și REVOKE nu pot fi folosite direct în PL/SQL deoarece sunt construite și executate la runtime, nu la compilare (sunt dinamice). Pot fi utilizate în PL/SQL doar cu SQL Dinamic.

Cursul 4 - Tipuri de date

Tipuri de date compuse

1. Înregistrare – RECORD - nu poate fi definit decât local (într-un bloc PL/SQL sau pachet)
2. Colecții

Tablourile indexate

- sunt mulțimi (locale) de perechi cheie-valoare, în care fiecare cheie este unică și utilizată pentru a putea localiza valoarea asociată;

Tablourile imbricate

- indecșii și ordinea liniilor dintr-un tablou imbricat ar putea să nu rămână stabile în timp ce tabloul este stocat sau regăsit din baza de date;
- inițial, un tablou imbricat este dens, dar în urma prelucrării este posibil să nu mai aibă indici consecutivi;
- inițializarea se realizează cu ajutorul unui constructor.

Vectorii

- sunt utilizați în special pentru modelarea relațiilor one-to-many, atunci când numărul maxim de elemente copil este cunoscut și ordinea elementelor este importantă;
- reprezintă structuri dense
- trebuie inițializați și extinși pentru a li se adăuga elemente.

Procedeeul BULK COLLECT

Execuția comenzilor SQL specificate în programe determină comutări ale controlului între motorul PL/SQL și motorul SQL. Prea multe astfel de schimbări de context afectează performanța.

Pentru a reduce numărul de comutări între cele două motoare se utilizează procedeul bulk collect, care permite transferul liniilor între SQL și PL/SQL prin intermediul colecțiilor.

Procedeul bulk collect implică doar două comutări între cele două motoare:

- o motorul PL/SQL comunică motorului SQL să obțină mai multe linii odată și să le plaseze într-o colecție;

- o motorul SQL regăsește toate liniile și le încarcă în colecție, după care predă controlul motorului PL/SQL.

Procedeul BULK BIND

Permite transferul liniilor dintr-o colecție printr-o singură operație, micșorându-se numărul de comutări între motorul SQL și motorul PL/SQL.

În loc de:

```
FOR i IN t.FIRST..t.LAST LOOP
```

```
    INSERT INTO tablou VALUES t(i);
```

```
END LOOP;
```

unde colecția este parcursă folosind comanda FOR, și la fiecare iterație o comandă INSERT este transmisă motorului SQL,

```
FORALL i IN t.FIRST..t.LAST LOOP
```

```
    INSERT INTO tablou VALUES t(i);
```

```
END LOOP;
```

Cursul 5 - Cursoare

Cursoare implicite

- PL/SQL deschide automat un cursor implicit la nivel de sesiune de fiecare dată când este rulată o comandă SELECT sau LMD.
- Mai sunt denumite și cursoare SQL.
- Cursorul implicit este închis automat, atunci când comanda se încheie.
- Valorile atributelor asociate cursorului rămân disponibile până când este rulată o altă comandă SELECT sau LMD.

Cursoare explicite

- Sunt cursoare la nivel de sesiune definite și gestionate de către utilizatori.

Deschiderea unui cursor explicit

La deschiderea unui cursor (OPEN nume_cursor;) se realizează următoarele operații:

- se alocă resursele necesare pentru a procesa cererea
- se procesează cererea:
- se evaluează comanda SELECT asociată (sunt examinate valorile variabilelor de legătură ce apar în declarația cursorului)
- se identifică mulțimea activă prin execuția cererii SELECT, având în vedere valorile de la pasul anterior;
- dacă cererea include clauza FOR UPDATE, atunci liniile din mulțimea activă sunt blocate;
 - se poziționează pointer-ul înaintea primei linii din mulțimea activă.

Încărcarea datelor dintr-un cursor explicit

Se realizează cu FETCH.

FETCH realizează următoarele operații:

- avansează pointer-ul la următoarea linie în mulțimea activă (pointer-ul poate avea doar un sens de deplasare de la prima înregistrare spre ultima);
 - citește datele liniei curente în variabile PL/SQL;
 - dacă pointer-ul este poziționat la sfârșitul mulțimii active, atunci se iese din bucla cursorului.
- Atunci când un cursor încarcă o linie, acesta realizează o „schimbare de context” – controlul este preluat de motorul SQL care va obține datele. Motorul SQL plasează datele în memorie și are loc o altă „schimbare de context” - controlul este preluat înapoi de motorul PL/SQL. Procesul se repetă până când nu mai sunt date de încărcat. Schimbările de context sunt foarte rapide, dar numărul prea mare de astfel de operații poate implica performanță scăzută (vezi BULK COLLECT).

Închiderea unui cursor explicit

După ce a fost procesată mulțimea activă, cursorul trebuie închis: CLOSE nume_cursor;

- PL/SQL este informat că programul a terminat folosirea cursorului și resursele asociate acestuia pot fi eliberate:
- spațiul utilizat pentru memorarea mulțimii active;
- spațiul temporar folosit pentru determinarea mulțimii active.

Cursoare parametrizate

- Unei variabile de tip cursor îi corespunde o comandă SELECT, care nu poate fi modificată pe parcursul programului.
- Cursoarele parametrizate sunt cursoare ale căror comenzi SELECT depind de parametri ce pot fi modificați la momentul execuției.
- Transmiterea de parametri unui cursor parametrizat se face în mod similar procedurilor stocate.

Cursoare SELECT FOR UPDATE

- Dacă este necesară blocarea liniilor înainte ca acestea să fie șterse sau reactualizate, atunci blocarea se poate realiza cu ajutorul clauzei FOR UPDATE a comenzii SELECT din definiția cursorului.
 - În momentul deschiderii unui cursor FOR UPDATE, liniile din mulțimea activă, determinată de clauza SELECT, sunt blocate pentru operații de scriere (reactualizare sau ștergere). În felul acesta este realizată consistența la citire a sistemului.
 - De exemplu, această situație este utilă atunci când se reactualizează o valoare a unei linii și trebuie avută siguranța că linia nu este schimbată de un alt utilizator înaintea reactualizării. Astfel, alte sesiuni nu pot schimba liniile din mulțimea activă până când tranzacția nu este permanentizată sau anulată.
- ! Deoarece cursorul lucrează doar cu niște copii ale liniilor existente în tabele, după închiderea cursorului este necesară comanda COMMIT pentru a realiza scrierea efectivă a modificărilor. Deoarece blocările implicate de clauza FOR UPDATE vor fi eliberate de comanda COMMIT, nu este recomandată utilizarea comenzii COMMIT în interiorul ciclului în care se fac încărcări de date. Orice FETCH executat după COMMIT va eșua.

Cursul 6 - Subprograme

Avantajele utilizării subprogramelor stocate:

- codul este ușor de întreținut;
- codul este reutilizabil;
- asigură securitatea datelor;
- asigură integritatea datelor;
- îmbunătățesc performanța

Etape parcurse de server-ul Oracle atunci când este apelat un subprogram stocat:

- verifică dacă utilizatorul are privilegiul de execuție asupra subprogramului;
- verifică dacă p-code-ul (forma compilată) a subprogramului este în shared pool; dacă este prezent, va fi executat, altfel va fi încărcat de pe disc în shared pool;
- verifică dacă starea subprogramului este VALID sau INVALID (dacă au fost detectate erori la compilare, dacă structura unui obiect s-a schimbat când subprogramul a fost executat ultima oară). Dacă starea subprogramului este INVALID, atunci este recompilat automat. Dacă nu a fost detectată nicio eroare, atunci va fi executată noua versiune a subprogramului;
- dacă subprogramul aparține unui pachet, atunci toate subprogramele pachetului sunt de asemenea încărcate în shared pool (dacă acestea nu erau deja acolo); dacă pachetul este activat pentru prima oară într-o sesiune, atunci server-ul va executa blocul de inițializare al pachetului.

Utilizarea în expresii SQL a funcțiilor definite de utilizator

O funcție stocată poate fi referită într-o comandă SQL la fel ca orice funcție standard furnizată de sistem, dar cu anumite restricții.

Funcțiile ce pot fi utilizate în comenzi SQL trebuie:

- să aibă numai parametri de tip IN, al căror tip de date este un tip valid SQL;
- să întoarcă o valoare al cărei tip să fie un tip valid SQL, cu dimensiunea maximă admisă în SQL.

Restricții de utilizare a funcțiilor în comenzi SQL:

- funcțiile invocate într-o comandă SELECT nu pot conține comenzi LMD;
- funcțiile invocate într-o comandă UPDATE sau DELETE asupra unui tabel T nu pot utiliza comenzi SELECT sau LMD care referă același tabel T;
- nu pot termina tranzacții;
- nu pot utiliza comenzi LDD, deoarece acestea realizează COMMIT automat;
- nu pot să apară în clauza CHECK a unei comenzi CREATE/ ALTER TABLE;
- nu pot fi folosite pentru a specifica o valoare implicită a unei coloane în cadrul unei comenzi CREATE/ ALTER TABLE;

Cursul 7 - Pachete

Pachetele sunt unități de program compilate și obiecte ale bazei de date care grupează tipuri, obiecte și subprograme PL/SQL având o legătură logică între ele.

Importanța pachetelor

Atunci când este referențiat un pachet (atunci când este apelată pentru prima dată o construcție a pachetului), întregul pachet este încărcat în SGA (zona globală sistem) și este pregătit pentru execuție.

Plasarea pachetului în SGA reprezintă avantajul vitezei de execuție, deoarece serverul nu mai trebuie să aducă informația despre pachet de pe disc, aceasta fiind deja în memorie. O Apelurile ulterioare ale unor construcții din același pachet, nu solicită operații I/O de pe disc. O De aceea, ori de câte ori apare cazul unor proceduri și funcții înrudite care trebuie să fie executate împreună, este convenabil ca acestea să fie grupate într-un pachet stocat.

Definirea pachetelor

Un pachet conține două părți, fiecare fiind stocată separat în dicționarul datelor:

1. Specificația:

- este partea vizibilă a pachetului, interfața cu aplicațiile sau cu alte unități program;
- poate conține declarații de tipuri, constante, variabile, cursoare, subprograme;
- se declară înaintea corpului pachetului.

2. Corpul:

- este partea ascunsă a pachetului, mascată de restul aplicației;
- conține codul care implementează obiectele definite în specificație (cursoarele și subprogramele) și de asemenea, obiecte și declarații proprii;
- poate conține obiecte publice sau private.

Procesul de creare a specificației și corpului unui pachet

- sunt verificate erorile sintactice și semantice, iar modulul este depus în dicționarul datelor;

- sunt verificate instrucțiunile SQL individuale, adică dacă obiectele referite există și dacă utilizatorul le poate accesa;
- sunt comparate declarațiile de subprograme din specificația pachetului cu cele din corpul pachetului (dacă au același număr și tip de parametri);
- orice eroare detectată la compilarea specificației sau a corpului pachetului este marcată în dicționarul datelor;
- după ce specificația și corpul pachetului sunt compilate, acestea devin obiecte în schema curentă.

Instanțierea pachetului

Un pachet este instanțiat atunci când este apelat prima dată.

Pachetul este citit de pe disc, depus în memorie și este executat codul compilat al subprogramului apelat.

În acest moment, memoria este alocată tuturor variabilelor definite în pachet.

Invalidarea modulelor PL/SQL

Dacă un subprogram independent apelează un subprogram definit într-un pachet, atunci apar următoarele situații:

- atunci când corpul pachetului este modificat, dar specificația acestuia nu, subprogramul care referă o construcție a pachetului rămâne valid;
- dacă specificația pachetului este modificată, atunci subprogramul care referă o construcție a pachetului, precum și corpul pachetului devin invalide.

Dacă un subprogram independent referit de un pachet se modifică, atunci întregul corp al pachetului devine invalid, dar specificația pachetului rămâne validă.

Etapele unei comenzi SQL

- analizarea sintactică;
- validarea;
- asigurarea că toate referințele la obiecte sunt corecte;
- asigurarea că există privilegiile referitoare la acele obiecte (parse);
- obținerea de valori pentru variabilele de legătură din comandă (binding variables);
- executarea comenzii (execute);
- selectarea liniilor rezultatului;
- încărcarea liniilor rezultatului (fetch).

Cursul 8 - Triggeri

Tabele mutating

Un tabel mutating este un tabel care este modificat curent de o comandă LMD (tabelul este aflat în proces de modificare).

Un trigger la nivel de linie nu poate obține informații (SELECT) dintr-un tabel mutating, deoarece ar "vedea" date inconsistente (datele din tabel ar fi în proces de modificare în timp ce trigger-ul ar încerca să le consulte).

Dacă o comandă INSERT afectează numai o înregistrare, trigger-ii la nivel de linie pentru înregistrarea respectivă nu tratează tabelul ca fiind mutating.

Un trigger nu poate apela un subprogram care conține comenzile COMMIT, ROLLBACK, SAVEPOINT.

Un trigger poate activa alt trigger, iar acesta la rândul său poate activa alt trigger etc (trigger-i în cascadă). Sistemul permite maximum 32 de triggeri în cascadă. Numărul acestora poate fi limitat (utilizând parametrul de inițializare OPEN_CURSORS), deoarece pentru fiecare execuție a unui trigger trebuie deschis un nou cursor.

Cursul 9 - Excepții

Importanța excepțiilor

Mecanismul de gestiune a excepțiilor permite utilizatorului să definească și să controleze comportamentul programului atunci când acesta generează o eroare.

Declanșarea excepțiilor:

- execuția blocului este întreruptă (setul de comenzi care urmează după comanda care a declanșat excepția nu se mai execută);
- controlul este transferat în secțiunea de tratare a excepțiilor;
- se tratează excepția (se execută comenzile specificate în handler-ul excepției respective);
- se iese din bloc.

Cursul 10 - Structura BD

Structura fizică a bazei de date

Structura fizică a bazei de date Oracle presupune existența unui set de fișiere binare prin intermediul cărora se realizează stocarea fizică a datelor:

- fișiere de date;
- fișiere de reluare;
- fișiere de control.

Fișiere de date

- stochează datele tuturor structurilor logice ale bazei și informația necesară funcționării sistemului; unul sau mai multe fișiere de date formează o unitate logică numită spațiu tabel.
- dacă un utilizator interoghează date dintr-un tabel al bazei și informațiile cererii solicitate nu sunt încă în memoria cache, atunci acestea vor fi preluate din fișierul de date și stocate în memorie.

Fișiere de reluare

- înregistrează toate modificările care au loc asupra datelor bazei (indiferent dacă au fost permanentizate sau nu) și care nu au fost scrise încă în fișierele de date;
- asigură protecția bazei de date în cazul defecțiunilor.

Fișiere de control

- sunt fișiere binare de dimensiune redusă, necesare pentru pornirea și funcționarea bazei de date;

- conțin informații despre structura fizică a bazei de date (numele și data creării bazei, numele și locația fișierelor de date și a fișierelor de reluare, etc.);
- la pornirea unei instanțe Oracle, sistemul folosește fișierul de control pentru a identifica baza și a determina dacă aceasta este în stare validă pentru utilizare; de asemenea, sunt identificate fișierele de reluare necesare execuției operațiilor bazei de date.

Structura logică a bazei de date

- bloc de date = cea mai mică unitate I/O folosită de baza de date, corespunzătoare unui bloc fizic de octeți de pe disc; conține un antet (informații generale referitoare la bloc), un spațiu pentru date (informații stocate în tabele sau indecși), un spațiu liber (pentru inserarea de noi linii sau actualizarea liniilor care necesită spațiu suplimentar);
- extensie = unitate logică de alocare a spațiului bazei de date, compusă dintr-o mulțime contiguă de blocuri de date (din același fișier de date);
- segment = set de extensie care conține toate datele unei structuri logice dintr-un spațiu tabel;
- spațiu tabel = unitate logică de stocare, asociată unei singure baze de date și formată dintr-unul sau mai multe segmente, utilizată pentru a grupa logic o mulțime de obiecte;
- obiect al schemei (tabel, vizualizare, secvență, unitate de program, sinonim, index, grupări, etc.).

Cursul 11 - Arhitectura internă a sistemului Oracle

Cele 3 niveluri ale sistemului Oracle

1. Nivelul procese - corespunde diverselor procese de sistem care asigură gestiunea datelor;
2. Nivelul memorie - corespunde zonelor tampon alocate pentru a stoca date și informații de control;
3. Nivelul fișiere - corespunde structurii bazei de date și modului în care sunt stocate datele.

Arhitectura proceselor

Procesele sunt mecanisme ale sistemului care permit executarea unor operații de calcul sau I/O și sunt de 2 tipuri:

- procese user, care generează mesaje printr-un program interfață care creează o sesiune și pornește un proces server.
 - procese Oracle (procese server, procese background), care asigură gestiunea informațiilor dintr-o bază de date.
1. Procesele server - interacționează cu procesele user și comunică în mod direct cu serverul Oracle pentru a transmite cererile acestora.
 2. Procesele background - sunt folosite pentru a îmbunătăți performanțele unui sistem multiprocesor.

Instanța Oracle

- constă dintr-o structură de memorie numită SGA și procese background;
- pentru a putea accesa baza de date, trebuie pornită o instanță;
- instanța poate fi asociată unei singure baze de date;
- atunci când baza de date este pornită, se localizează structura SGA și sunt lansate procesele background; dacă unul dintre aceste procese se termină anormal, instanța se oprește.

Arhitectura memoriei

Toate structurile de memorie se găsesc în memoria centrală.

Din punct de vedere structural, memoria este compusă din:

- zona globală sistem (SGA) - grup de structuri partajate de memorie care conțin date și informații de control relative la o instanță, alocată la pornirea instanței și eliberată la oprirea acesteia.
- zona globală program (PGA) - zonă de memorie care conține date și informații de control relative la un singur proces Oracle, alocată la crearea procesului și eliberată la terminarea acestuia.

Zone SGA

Database buffer cache - o mulțime de zone tampon care stochează blocurile de date folosite cel mai recent; Atunci când este executată o cerere, procesul server caută blocurile de care are nevoie în database buffer cache. Dacă nu găsește un bloc, procesul server îl citește din fișierul de date și plasează o copie a acestuia în database buffer cache. Întrucât datele cel mai frecvent folosite sunt păstrate în memoria cache, se micșorează numărul de operații I/O și se îmbunătățesc performanțele bazei de date.

Redo log buffer - un buffer circular care conține informațiile referitoare la modificările blocurilor de date ale bazei.

Shared pool - o porțiune din SGA formată din 3 zone: library cache (informații despre cele mai recente comenzi SQL și unități de program PL/SQL), data dictionary cache și buffere (pentru mesaje relative la execuții paralele și structuri de control).

Pași configurare Oracle în care procesul user și procesul server rulează pe mașini diferite:

1) Se pornește o instanță pe stația care găzduiește sistemul Oracle (cunoscută sub denumirea de gazdă sau server de baze de date).

2) Aplicația client încearcă să stabilească o conexiune cu server-ul de baze de date.

3) Server-ul detectează cererea de conectare a aplicației și creează un proces server dedicat procesului user. Procesului server i se alocă o zonă de memorie PGA.

4) Utilizatorul execută o comandă SQL și apoi instrucțiunea COMMIT pentru a permanentiza schimbările și a încheia tranzacția.

5) Procesul server primește comanda și verifică dacă în shared pool (library cache) există o zonă partajată SQL care conține comenzi similare. Dacă este găsită o astfel de zonă, procesul server verifică privilegiile de acces ale utilizatorului la datele interogate și folosește zona respectivă pentru procesarea comenzii. În caz contrar, pentru comandă este alocată o nouă zonă partajată SQL, unde aceasta va putea fi analizată și procesată.

6) Procesul server extrage datelor necesare cererii (din database buffer cache sau din fișierele de date).

7) Procesul server modifică datele din database buffer cache. Informațiile referitoare la modificările blocurilor de date sunt înregistrate în redo log buffer. Deoarece tranzacția a fost permanentizată, procesul LGWR o înregistrează imediat într-un fișier de reluare. La un moment dat procesul CKPT anunță procesul DBWR că va avea loc o operație checkpoint și modifică header-ele fișierelor de date și de control, pentru a înregistra detalii despre aceasta. DBWR scrie blocurile de date modificate din database buffer cache în fișierele de date.

8) Dacă tranzacția s-a derulat cu succes, procesul server trimite un mesaj corespunzător către aplicație. Altfel, este transmis un mesaj de eroare.