Cursul 1 - Recapitulare BD

bază de date = ansamblu structurat de date coerente, fără redundanţă inutilă, astfel încât acestea pot fi prelucrate eficient de mai mulţi utilizatori într-un mod concurent

dicţionarul datelor = catalog de sistem, structurat şi administrat ca o bază de date (metabază de date) care conţine informaţii despre date (descrierea obiectelor, stărilor, constrângerilor)

SGBD = produs software care asigură interacţiunea cu o bază de date, permiţând definirea, consultarea şi actualizarea datelor din baza de date

baze de date distribuite (BDD) = colecţii de baze de date corelate logic între ele care rezidă pe mai multe calculatoare interconectate printr-o reţea de comunicaţie

baze de date depozit = baze de date ce constă în date istorice

Regulile modelului relaţional:
1. Unicitatea cheii (cheia primară trebuie să fie unică şi minimală);
2. Integritatea entităţii (atributele cheii primare trebuie să fie diferite de valoarea null);
3. Integritatea referirii (o cheie externă trebuie să fie ori null în întregime, ori să corespundă unei valori a cheii primare asociate).

Componentele unui sistem de baze de date:
- baza de date (memorează datele);
- SGBD-ul (gestionează şi prelucrează datele);
- dicţionarul bazei de date (stochează informaţii despre date);
- componente hardware;
- reglementări administrative;
- personalul implicat.

Componentele unui SGBD:
- interfaţa cu utilizatorul (interpretează comenzile, formatează rezultatele);
- controlul semantic al datelor (verifică dacă cererile utilizatorului pot fi procesate);
- optimizorul şi analizorul cererilor (determină strategiile de execuţie);
- administratorul tranzacţiilor (coordonează execuţia cererii);
- administratorul recuperărilor (asigură consistenţa bazei de date în cazul defecţiunilor la nivel de sistem);
- procesorul execuţiei (accesează fizic baza de date în concordanţă cu comenzile generate de optimizor).

Tipuri de BDD (baze de date distribuite)
- omogene (compuse din baze de date locale de acelaşi tip, administrate de acelaşi SGBD);
- eterogene (compuse din baze de date locale de acelaşi tip sau de tipuri diferite, administrate de SGBD-uri diferite).

OLTP (Online-Line-Transactional-Processing) vs DW (Data Warehouse)
(Pe scurt, OLTP = customer-oriented; DW = market-oriented)

OLTP	DW
Read Write	Read Only
Trazacţii LMD	Nu există tranzacţii LMD
Blocări (dispută pe resurse)	Nu există blocări
Cantitate citită mică	Cantitate citită foarte mare
Identifică prezentul companiei	Identifică predicţii referitoare la viitorul companiei (după analiza datelor istorice)
Informații stocate	
operaţionale	centralizate sau derivate din datele operaţionale, pentru asistarea deciziei
de detaliu, cu un anumit grad de volatilitate	nevolatile

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 instrucţiuni LMD, LDD, LCD.	În PL/SQL, blocurile de cod pot utiliza proceduri, funcţii, pachete, variabile, etc.
Interogările SQL sunt utilizate pentru a extrage date din baza de date, dar şi pentru a adăuga, șterge sau modifica date.	PL/SQL este utilizat pentru a crea aplicaţii care pot afișa informaţia extrasă de SQL.
Interogările SQL nu pot conţine cod PL/SQL	Blocurile PL/SQL pot conţine cod SQL
Interacţionează direct cu server-ul bazei de date	Nu interacţionează direct cu server-ul bazei de date

Caracteristici PL/SQL
- este integrat cu server-ul Oracle şi utilitarele Oracle;
- este puternic integrat cu SQL;
- extinde SQL prin construcţii specifice limbajelor procedurale (definirea constantelor, variabilelor, declararea tipurilor, definirea şi utilizarea structurilor de control, definirea procedurilor şi funcţiilor, modularizarea programelor, detectarea şi gestiunea erorilor de execuţie şi a situaţiilor excepţionale, etc.)
- permite definirea şi utilizarea declanşatorilor;
- asigură securitatea informaţiei;
- măreşte performanţa aplicaţiei (permite înglobarea mai multor instrucţiuni într-un singur bloc şi trimiterea acestuia către baza de date, reducându-se astfel traficul dintre aplicaţie şi bază);
- are suport pentru dezvoltarea aplicaţiilor WEB;
- este portabil (nu depinde de platformă sau de sistemul de operare, programele PL/SQL pot rula pe orice suport pe care există un server Oracle).

Motorul PL/SQL
- compilează şi 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 conţin comenzi SQL;
- indiferent de mediu, motorul PL/SQL execută comenzile procedurale, dar trimite comenzile SQL către motorul SQL de pe server-ul Oracle;
- comenzile procedurale pot fi executate pe staţia client fără interacţiune cu server-ul Oracle sau în întregime 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. Instrucţiuni

Blocuri anonime	Subprograme stocate
Blocuri PL/SQL fără nume	Blocuri PL/SQL cu nume
Compilate de fiecare dată când aplicația este executată	Compilate o singură dată
Nu sunt stocate în baza de date	Stocate în baza de date
Nu pot fi invocate de alte aplicații	Pot fi invocate de alte aplicații
Nu întorc valori	Funcţiile trebuie să întoarcă o valoare
Nu acceptă parametri	Acceptă parametri

Comenzi SQL în PL/SQL

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

Comenzile LDD (CREATE, ALTER, DROP) şi comenzile LCD GRANT şi REVOKE nu pot fi folosite direct în PL/SQL deoarece sunt construite şi 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 specifice PL/SQL

- BOOLEAN (stochează valorile true, false sau null; nu are un tip SQL echivalent);
- PLS_INTEGER/ BINARY_INTEGER (stochează numere întregi cu semn pe 32 biţi, necesită mai puţin spaţiu de stocare faţă de NUMBER şi deoarece folosesc aritmetica maşinii, operaţiile sunt efectuate mai rapid);
- referinţă (are ca valoare un pointer care face referinţă către un obiect);
- subtipuri definite de utilizator.

Conversii între tipuri de date

- implicite (realizate automat de sistem);
- explicite (realizate folosind explicit funcţii de conversie - CAST, CONVERT, TO_CHAR, TO_NUMBER, TO_DATE, etc.).

Dezavantaje conversii implicite:

- pot fi lente;
- se pierde controlul asupra programului (dacă Oracle modifică regulile de conversie, atunci codul poate fi afectat);
- depind de mediul în care sunt utilizate;
- codul devine mai greu de înţeles.

Tipuri de date compuse

1. Înregistrare - RECORD

- se defineşte în doi paşi: definirea unui tip RECORD, declararea unei variabile de acest tip;
- câmpurile unei înregistrări au implicit valoarea null şi se referă prin prefixare cu numele înregistrării;
- atribuirea de valori unei înregistrări se poate realiza cu instrucţiunea de atribuire, comenzile SELECT sau FETCH;
- înregistrările nu pot fi comparate, pot fi parametru în subprograme, pot să apară în clauza RETURN a unei funcţii;
- tipul RECORD nu poate fi definit decât local (într-un bloc PL/SQL sau pachet)

2. Colecţii

<u>Tablourile indexate</u>

- sunt mulţimi de perechi cheie-valoare, în care fiecare cheie este unică şi utilizată pentru a putea localiza valoarea asociată;

- atunci când o valoare este asociată pentru prima oară unei chei, cheia este adăugată în tablou;
- iniţial sunt vide (nu iniţializate cu null);

Tablourile imbricate
- stochează mulţimi de valori;
- indecşii şi ordinea liniilor dintr-un tablou imbricat ar putea să nu rămână stabile în timp ce tabloul este stocat sau regăsit din baza de date;
- iniţial, un tablou imbricat este dens, dar în urma prelucrării este posibil să nu mai aibă indici consecutivi;
- iniţializarea se realizează cu ajutorul unui constructor.

Vectorii
- sunt utilizaţi în special pentru modelarea relaţiilor one-to-many, atunci când numărul maxim de elemente copil este cunoscut şi ordinea elementelor este importantă;
- reprezintă structuri dense;
- au o dimensiune maximă specificată la declarare;
- trebuie iniţializaţi şi extinşi pentru a li se adăuga elemente.

Procedeul BULK COLLECT

Execuţia comenzilor SQL specificate în programe determină comutări ale controlului între motorul PL/SQL şi motorul SQL. Prea multe astfel de schimbări de context afectează performanţa.

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

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

o motorul PL/SQL comunică motorului SQL să obţină mai multe linii odată şi să le plaseze într-o colecţie;

o motorul SQL regăseşte toate liniile şi le încarcă în colecţie, după care predă controlul motorului PL/SQL.

Procedeul BULK BIND

Permite transferul liniilor dintr-o colecţie printr-o singură operaţie, micşorându-se numărul de comutări între motorul SQL şi motorul PL/SQL.

În loc de:

```
FOR i IN t.FIRST..t.LAST LOOP
        INSERT INTO tablou VALUES t(i);
END LOOP;
```

unde colecţia este parcursă folosind comanda FOR, şi la fiecare iteraţie 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

cursor = pointer către o zonă de memorie care stochează informaţii necesare pentru procesarea unei comenzi SELECT sau LMD.

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 şi 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 şi gestionate de către utilizatori.
• Un cursor explicit are specificat un nume. Acesta este asociat cu o comandă SELECT ce întoarce de obicei mai multe linii.
• Mulţimea rezultat a cererii asociate poate fi procesată folosind una dintre variantele următoare:

- se deschide cursorul (comanda OPEN), se încarcă liniile cursorului în variabile (comanda FETCH), se închide cursorul (comanda CLOSE);
- se utilizează cursorul într-o comandă FOR LOOP.

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 mulţimea activă prin execuția cererii SELECT, având în vedere valorile de la pasul anterior;
▪ dacă cererea include clauza FOR UPDATE, atunci liniile din mulţimea 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);
- citeşte datele liniei curente în variabile PL/SQL;
- dacă pointer-ul este poziționat la sfârşitul 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 şi 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 operaţii poate implica performanţă 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 şi 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 şterse sau reactualizate, atunci blocarea se poate realiza cu ajutorul clauzei FOR UPDATE a comenzii SELECT din definiţia cursorului.
• În momentul deschiderii unui cursor FOR UPDATE, liniile din mulțimea activă, determinată de clauza SELECT, sunt blocate pentru operații de scriere (reactualizare sau ştergere). 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 şi 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 nişte 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 eşua.

Cursul 6 - Subprograme

Avantajele utilizării subprogramelor stocate:
- codul este uşor de întreţinut;
- codul este reutilizabil;
- asigură securitatea datelor;
- asigură integritatea datelor;
- îmbunătățesc performanţa

Etape parcurse de server-ul Oracle atunci când este apelat un subprogram stocat:
- verifică dacă utilizatorul are privilegiul de execuţie 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 aparţine 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 iniţializare al pachetului.

Utilizarea în expresii SQL a funcţiilor definite de utilizator

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

Funcţiile 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.

Restricţii de utilizare a funcţiilor în comenzi SQL:
- funcţiile invocate într-o comandă SELECT nu pot conţine comenzi LMD;
- funcţiile invocate într-o comandă UPDATE sau DELETE asupra unui tabel T nu pot utiliza comenzi SELECT sau LMD care referă acelaşi tabel T;
- nu pot termina tranzacţii;
- 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 unităţi de program compilate şi obiecte ale bazei de date care grupează tipuri, obiecte şi subprograme PL/SQL având o legătură logică între ele.

Importanţa 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) şi este pregătit pentru execuție.

Plasarea pachetului în SGA reprezintă avantajul vitezei de execuție, deoarece server-ul nu mai trebuie să aducă informația despre pachet de pe disc, aceasta fiind deja în memorie.

o Apelurile ulterioare ale unor construcții din acelaşi pachet, nu solicită operații I/O de pe disc.
o De aceea, ori de câte ori apare cazul unor proceduri şi funcții înrudite care trebuie să fie executate împreună, este convenabil ca acestea să fie grupate într-un pachet stocat.

Definirea pachetelor

Un pachet conţine două părţi, fiecare fiind stocată separat în dicţionarul datelor:
1. Specificaţia:
- este partea vizibilă a pachetului, interfaţa cu aplicaţiile sau cu alte unităţi program;
- poate conţine declaraţii de tipuri, constante, variabile, cursoare, subprograme;
- se declară înaintea corpului pachetului.
2. Corpul:
- este partea ascunsă a pachetului, mascată de restul aplicaţiei;
- conţine codul care implementează obiectele definite în specificaţie (cursoarele şi subprogramele) şi de asemenea, obiecte şi declaraţii proprii;
- poate conţine obiecte publice sau private.

Procesul de creare a specificaţiei şi corpului unui pachet
- sunt verificate erorile sintactice şi semantice, iar modulul este depus în dicţionarul datelor;
- sunt verificate instrucţiunile SQL individuale, adică dacă obiectele referite există şi dacă utilizatorul le poate accesa;
- sunt comparate declaraţiile de subprograme din specificaţia pachetului cu cele din corpul pachetului (dacă au acelaşi număr şi tip de parametri);
- orice eroare detectată la compilarea specificaţiei sau a corpului pachetului este marcată în dicţionarul datelor;
- după ce specificaţia şi corpul pachetului sunt compilate, acestea devin obiecte în schema curentă.

Instanţierea pachetului

Un pachet este instanţiat atunci când este apelat prima dată.

Pachetul este citit de pe disc, depus în memorie şi 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 şi 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 referinţele la obiecte sunt corecte;
- asigurarea că există privilegiile referitoare la acele obiecte (parse);
- obţinerea de valori pentru variabilele de legătură din comandă (binding variables);
- executarea comenzii (execute);
- selectarea liniilor rezultatului;
- încărcarea liniilor rezultatului (fetch).

Cursul 8 - Triggeri

trigger (declanşator) = bloc PL/SQL cu nume, stocat în baza de date (independent), care se execută automat ori de câte ori are loc evenimentul declanşator de care este asociat (modificarea unui tabel sau a unei vizualizări, acţiuni sistem, acţiuni utilizator)

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.

Modul de execuție al triggerilor LMD multipli:

1. Se execută toţi triggerii BEFORE comandă care sunt declanşaţi de comanda LMD;
2. Pentru fiecare linie afectată de comanda LMD:

   i) se execută toţi triggerii BEFORE linie care sunt declanşaţi de comanda LMD;

   ii) se blochează şi se modifică linia afectată de comanda LMD (se execută comanda); sunt verificate constrângerile de integritate şi blocarea rămâne valabilă până în momentul în care tranzacţia este permanentizată);

   iii) se execută toţi triggerii AFTER linia care sunt declanşaţi de comanda LMD;
3. Se execută toţi triggerii AFTER comandă care sunt declanşaţi de comanda LMD.

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 - Excepţii

Importanţa excepţiilor
Mecanismul de gestiune a excepţiilor permite utilizatorului să definească şi să controleze comportamentul programului atunci când acesta generează o eroare.
Într-un program PL/SQL pot să apară:
- erori la compilare: detectate de motorul PL/SQL;
- erori la execuţie (excepţii) - din cauza deficienţelor de proiectare, greşelilor de cod, etc.
Prin utilizarea excepţiilor şi a rutinelor de tratare a excepţiilor, un program PL/SQL devine robust şi capabil să trateze atât erorile aşteptate, cât şi cele neaşteptate ce pot apărea în timpul execuţiei.

Funcţii pentru identificarea excepţiilor:
- SQLCODE - pentru codul excepţiei;
- SQLERRM - pentru mesajul asociat excepţiei.

Declanşarea excepţiilor:
- execuţia blocului este întreruptă (setul de comenzi care urmează după comanda care a declanşat excepţia nu se mai execută);
- controlul este transferat în secţiunea de tratare a excepţiilor;
- se tratează excepţia (se execută comenzile specificate în handler-ul excepţiei respective);
- se iese din bloc.

Propagarea excepţiilor
Eroare declanşată în secţiunea executabilă:
1. dacă blocul curent are un handler pentru tratare: blocul curent se termină cu succes şi controlul este transmis blocului imediat exterior;
2. dacă blocul curent nu are handler pentru tratare: excepţia se propagă spre blocul "părinte", iar blocul curent se termină fără succes -> procesul continuă până fie se găseşte într-un bloc modalitatea de tratare a erorii, fie se opreşte execuţia şi se semnalează situaţia apărută (unhandled exception error).
Eroare declanşată în secţiunea declarativă sau în secţiunea de tratare a erorilor:
- excepţia este propagată către blocul imediat exterior, chiar dacă există un handler al acesteia în blocul curent.

Cursul 10 - Structura BD

Structura fizică a bazei de date
Structura fizică a bazei de date Oracle presupune existenţa unui set de fişiere binare prin intermediul cărora se realizează stocarea fizică a datelor:
- fişiere de date; - fişiere de reluare; - fişiere de control.

Fişiere de date
- sunt fişiere fizice ale sistemului de operare care stochează datele tuturor structurilor logice ale bazei şi informaţia necesară funcţionării sistemului.
- unul sau mai multe fişiere de date formează o unitate logică numită spaţiu tabel.
- dacă un utilizator interoghează date dintr-un tabel al bazei şi informaţiile cererii solicitate nu sunt încă în memoria cache, atunci acestea vor fi preluate din fişierul de date şi stocate în memorie.

Fişiere de reluare
- înregistrează toate modificările care au loc asupra datelor bazei (indiferent dacă au fost permanentizate sau nu) şi care nu au fost scrise încă în fişierele de date;
- asigură protecţia bazei de date în cazul defecţiunilor (dacă operaţiile bazei sunt întrerupte din cauza unei avarii de curent, datele din memorie nu pot fi scrise în fişierele de date şi sunt pierdute temporar; atunci când baza de date este repornită, informaţiile vor putea fi recuperate din cele mai recente fişiere de reluate).

Fişiere de control
- sunt fişiere binare de dimensiune redusă, necesare pentru pornirea şi funcţionarea bazei de date;
- conţine informaţii despre structura fizică a bazei de date (numele şi data creării bazei, numele şi locaţia fişierelor de date şi a fişierelor de reluare, etc.);
- la pornirea unei instanţe Oracle, sistemul foloseşte fişierul de control pentru a identifica baza şi a determina dacă aceasta este în stare validă pentru utilizare; de asemenea, sunt identificate fişierele de reluare necesare execuţiei operaţiilor bazei de date.

Structura logică a bazei de date
Utilizarea adecvată a structurilor logice determină alocarea optimă a spațiului de pe disc în vederea obținerii unei baze de date eficiente.
Componente ale structurii logice:
- bloc de date = cea mai mică unitate I/O folosită de baza de date, corespunzătoare unui bloc fizic de octeţi de pe disc; conţine un antet (informaţii generale referitoare la bloc), un spaţiu pentru date (informaţii stocate în tabele sau indecşi), un spaţiu liber (pentru inserarea de noi linii sau actualizarea liniilor care necesită spaţiu suplimentar);
- extensie = unitate logică de alocare a spaţiului bazei de date, compusă dintr-o mulțime contiguă de blocuri de date (din acelaşi fişier de date);
- segment = set de extensie care conţine toate datele unei structuri logice dintr-un spaţiu tabel;
- spaţiu tabel = unitate logică de stocare, asociată unei singure baze de date şi formată dintr-unul sau mai multe segmente, utilizată pentru a grupa logic o mulţime de obiecte
- obiect al schemei (tabel, vizualizare, secvenţă, unitate de program, sinonim, index, grupări, etc.).

Vizualizările dicţionarului de date
Dicţionarul datelor este compus din tabele de bază şi vizualizări publice asupra acestora. Vizualizările decodifică informaţiile stocate în tabelele de bază şi le sintetizează pentru a fi disponibile utilizatorilor.

Vizualizări DBA_ - relative la toate obiectele din baza de date;
Vizualizări ALL_ - informaţii despre toate obiectele schemei la care are acces utilizatorul curent;
Vizualizări USER_ - informaţii despre utilizatorul curent, obiectele create de acesta, privilegii, etc.
Vizualizări V$ şi GV$ - relative la performanţe.

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 şi informaţii de control;
3. Nivelul fişiere - corespunde structurii bazei de date şi modului în care sunt stocate datele.

Arhitectura proceselor

Procesele sunt mecanisme ale sistemului care permit executarea unor operaţii de calcul sau I/O şi sunt de 2 tipuri:
- procese user, care generează mesaje printr-un program interfaţă care creează o sesiune şi porneşte un proces server.
- procese Oracle (procese server, procese background), care asigură gestiunea informaţiilor dintr-o bază de date.
1. Procesele server - interacţionează cu procesele user şi comunică în mod direct cu server-ul Oracle pentru a transmite cererile acestora.
2. Procesele background - sunt folosite pentru a îmbunătăţi performanţele unui sistem multiprocesor.

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

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 conţin date şi informaţii de control relative la o instanţă, alocată la pornirea instanţei şi eliberată la oprirea acesteia.

- zona globală program (PGA) - zonă de memorie care conţine date şi informaţii de control relative la un singur proces Oracle, alocată la crearea procesului şi eliberată la terminarea acestuia.

Zone SGA

Database buffer cache

- Este o mulţime 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 găseşte un bloc, procesul server îl citeşte din fişierul de date şi plasează o copie a acestuia în database buffer cache. Întrucât datele cel mai frecvent folosite sunt păstrate în memoria cache, se micşorează numărul de operaţii I/O şi se îmbunătăţesc performanţele bazei de date.

Redo log buffer

- Este un buffer circular care conţine informaţiile referitoare la modificările blocurilor de date ale bazei.
- Aceste informaţii pot fi necesare atât pentru recuperarea bazei de date, cât şi pentru întoarcerea la forma anterioară a modificărilor realizate prin operaţii de tip LMD sau LDD.
- Conţinutul buffer-ului este scris în fişierul de reluare.

Shared pool

- Este o porţiune din SGA formată din 3 zone: library cache (informaţii despre cele mai recente comenzi SQL şi unităţi de program PL/SQL), data dictionary cache şi buffere (pentru mesaje relative la execuţii paralele şi structuri de control).

Paşi configurare Oracle în care procesul user şi procesul server rulează pe maşini diferite:

1) Se porneşte o instanţă pe staţia care găzduieşte sistemul Oracle (cunoscută sub denumirea de gazdă sau server de baze de date).

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

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

4) Utilizatorul execută o comandă SQL şi apoi instrucţiunea COMMIT pentru a permanentiza schimbările şi a încheia tranzacţia.

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

6) Procesul server extrage datelor necesare cererii. Dacă datele necesare au fost recent folosite, atunci procesul server le va găsi şi extrage din database buffer cache. În caz contrar, datele vor fi extrase din fişierele de date şi o copie a acestora va fi depusă în database buffer cache.

7) Procesul server modifică datele din database buffer cache. Informaţiile referitoare la modificările blocurilor de date sunt înregistrate în redo log buffer. Deoarece tranzacţia a fost permanentizată, procesul LGWR o înregistrează imediat într-un fişier de reluare. La un moment dat procesul CKPT anunţă procesul DBWR că va avea loc o operaţie checkpoint şi

modifică header-ele fişierelor de date şi de control, pentru a înregistra detalii despre aceasta. DBWR scrie blocurile de date modificate din database buffer cache în fişierele de date.

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