• *Titlul cursului*: **SGBD Oracle**
Autori: Prof. univ. dr. Manole Velicanu, Asist. univ. dr. Vlad Diaconița

• *Introducere*

Vă felicităm pentru faptul că ați ajuns în anul 2 semestrul al doilea şi vă dorim succes !

Cursul **SGBD Oracle** se adresează studenților din anul doi de la facultatea CSIE şi este continuarea disciplinei *Baze de date* din semestrul întâi.

Obiectivul general al disciplinei SGBD Oracle este însuşirea de către studenți a unor noțiuni fundamentale despre: Sistemele de Gestiune a Bazelor de Date - SGBD, sistemele relaționale, sistemul Oracle, modul de lucru cu limbajul PL/SQL, limbajul SQL avansat, interfețele Forms şi Reports din Oracle.

Obiectivele specifice principale ale acestui curs, concretizate în *competențele* pe care le veți dobândi după parcurgerea şi asimilarea lui, sunt:

- familiarizarea cu posibilitățile de utilizare ale unui SGBD indiferent de tipul lui;
- dobândirea abilității de a dezvolta aplicații cu baze de date relaționale;
- cunoaşterea sistemului Oracle şi utilizarea unor componente ale sale (PL-SQL, Forms, Report) pentru a realiza baze de date relaționale.

Structurarea cursului SGBD Oracle este făcută în trei unități de învățare (capitole), fiecare dintre acestea cuprinzând cate un test de verificare, care va fi rezolvat de către student.

Evaluarea cunoştințelor se va realiza sub două forme:

- evaluarea continuă – testele de la sfârşitul fiecărei unități de învățare, precum şi un proiect la seminar;
- evaluarea finală - examenul susținut în perioada de sesiune.

Notă. Structura obligatorie pentru proiect va fi:

- 1 pagina care să conțină: Tema, Descrierea problemei, Schema conceptuală a BD;
- programe PL-SQL care să conțină: cele 3 structuri fundamentale de programare procedurală, cursorul explicit, două tipuri de subprograme;
- o aplicație simplă cu Oracle Forms sau Reports.

Criteriile de evaluare constau în:

1. Punctajul obținut la cele trei teste menționate mai sus.
2. Punctajul obținut la proiectul de la seminar.
3. Punctajul obținut la examenul susținut în sesiune.

Ponderile asociate fiecărui criteriu precizat sunt următoarele:

- criteriul 1(C1) – câte 1 punct pentru fiecare dintre cele trei teste (total C1= 3 puncte);
- criteriul 2 (C2) – 2 puncte pentru proiectul de la seminar;
- criteriul 3 (C3) – 5 puncte pentru examenul susținut în sesiune.

Cuprinsul cursului

Unitatea de învățare 1

ASPECTE FUNDAMENTALE SGBD CU APLICARE ÎN ORACLE

Cuprins

1.1. Conceptul de SGBD.

Noțiunea de SGBD (Sistem de Gestiune a Bazelor de Date – DataBase Management System), trebuie studiată în contextul unui Sistem de Bază de Date - SBD.

a) Definirea unui SGBD (vezi arhitectura unui SBD din cursul Baze de date)
SGBD este un ansamblu complex de programe care asigură interfața între o bază de date şi utilizatorii acesteia.
SGBD este componenta software a unui sistem de bază de date care interacționează cu toate celelalte componente ale acestuia, asigurând legătura şi interdependența între elementele sistemului.

b)Rolul unui SGBD
Rolul unui SGBD într-un context de sistem de bază de date este de a:
1. *defini şi descrie* structura bazei de date, printr-un limbaj propriu specific (LDD), conform unui anumit model de date;
2. *încărca/valida* datele în baza de date respectând nişte restricțiile de integritate impuse de modelul de date utilizat;
3. *realiza accesul* la date pentru diferite operații (consultare, interogarea, actualizare etc) - operatorii modelului de date;
4. *întreține* BD cu ajutorul unor instrumente specializate (editoare, utilitare - shells, navigatoare – browsers etc.);
5. asigura *protecția* bazei de date sub cele două aspecte: securitatea şi integritatea datelor.

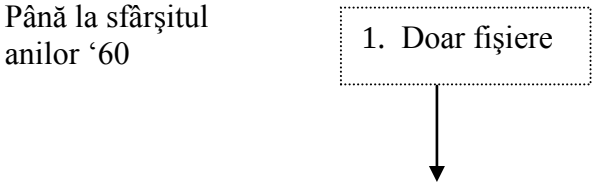
c) Evoluția SGBD
Evoluția SGBD a fost determinată, în principal, de modelul de date pe care-l implementează la organizarea datelor în BD.
Etapele în evoluția SGBD sunt prezentate în continuare.

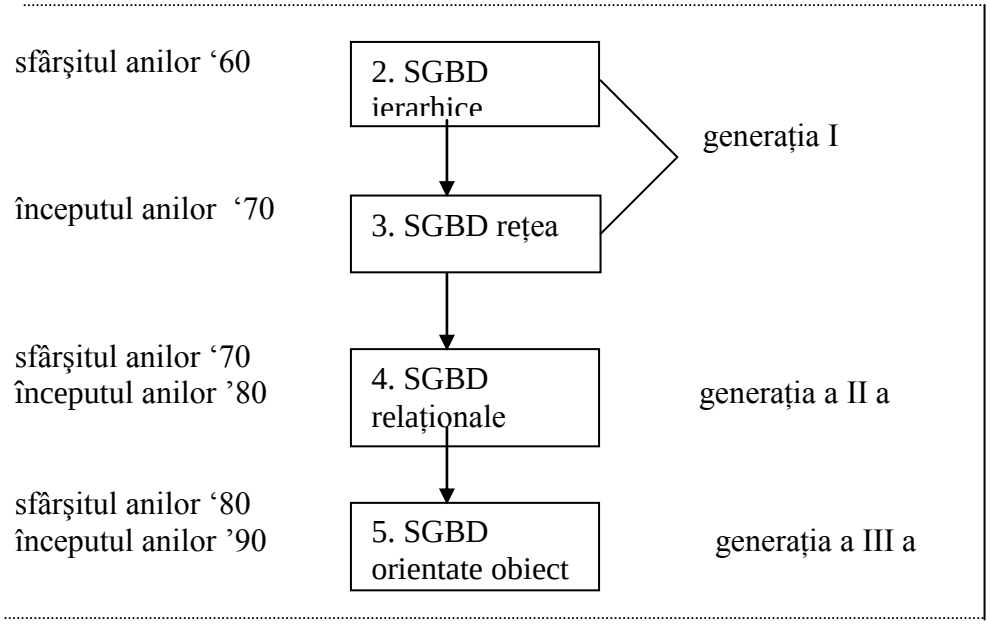
1. Până în anii şaizeci datele erau organizate doar în fişiere, gestionate de programe scrise în diferite limbaje de programare universale (exemple: Cobol, Fortran etc.).

2. La sfârşitul anilor şaizeci a apărut modelul arborescent de organizare a datelor în BD şi primele *SGBD* care erau *ierarhice* şi implementau acest model (exemplu: IMS).

3. La începutul anilor şaptezeci a apărut modelul rețea de organizare a datelor şi *SGBD rețea* ce implementau acest model (exemple: IDMS, SOCRATE).

SGBD arborescente şi rețea fac parte din *prima generație* şi ele constituie pionieratul în domeniu.
4. La sfârşitul anilor şaptezeci şi începutul anilor optzeci a apărut modelul relațional de organizare a datelor şi ulterior s-au realizat SGBD relaționale ce implementau acest model (exemple: Oracle, Informix, DB2, SQLServer, Visual Foxpro, Access etc.). Este *generația a doua* de SGBD, care prin simplitate, interactivitate şi neproceduralitate domină piața actuală.
5. La sfârşitul anilor optzeci, începutul anilor nouăzeci a apărut modelul orientat obiect de organizare a datelor şi SGBD orientate obiect ce implementau acest model (exemple: Gemstone, O2, Jasmine etc.). Aceasta este *generația a treia* de SGBD care este în plină dezvoltare acum.





Notă. În acest moment, în lume, cea mai mare parte a bazelor de date sunt realizate cu SGBD relaționale, o foarte mică parte cu SGBD de generația întâi şi câştigă tot mai mult teren cele realizate cu SGBD orientate obiect.
Oracle este un SGBD relațional extins cu numeroase alte tehnologii informatice.
Notă. La trecerea de la o generație la alta de SGBD s-au urmărit următoarele **aspecte***:*

1. Păstrarea aspectelor *fundamentale* care dau conceptul de SGBD: obiectivele, funcțiile, componentele. Desigur, la fiecare generație apar şi o serie de elemente specifice, de nuanță, care însă nu schimbă fundamentele conceptului.

2. Încadrarea unui SGBD într-o generație se face în funcție de *modelul de date* implementat. Sunt luate în considerare toate cele trei elemente care caracterizează modelul de date: definirea structurii modelului (entitățile şi legăturile dintre ele), operatorii de prelucrare, restricțiile de integritate.

3. Îmbunătățirea *organizării datelor* în memoria externă prin implementarea unui model mai performant. Se urmăreşte creşterea independenței logice şi fizice (până la total), asigurarea unor limbaje de descriere a datelor (LDD) cât mai performante şi automatizate, asigurarea unor limbaje de manipulare a datelor (LMD) puternice, asigurarea unor limbaje de regăsire ne-procedurale (exemplu SQL), reducerea şi controlul redundanței.

4. Îmbunătățirea *accesului* la date prin: acces după mai multe chei, acces concurent, optimizarea accesului, creşterea securității datelor.

5. Oferirea unor facilități de *utilizare* tot mai performante: generatoare specializate (Forms, Reports etc.), interfețe cu alte limbaje de programare, interactivitatea.

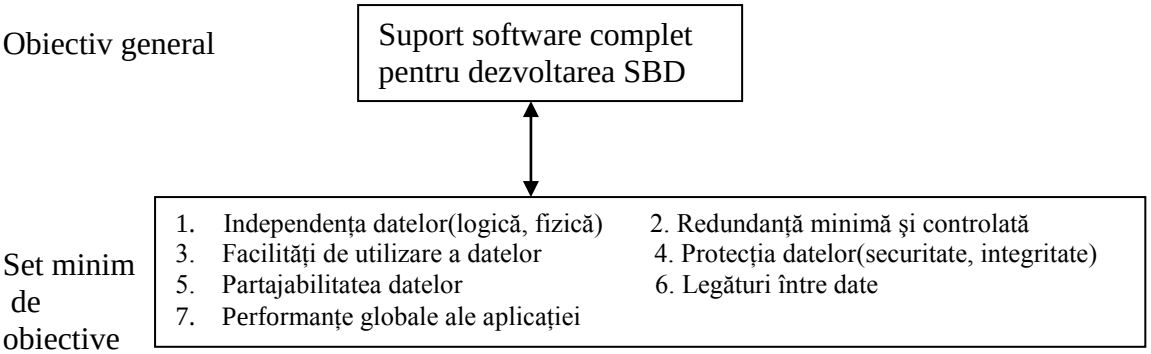
6. *Diversificarea* tipurilor de date ce pot fi utilizate (clasice, multimedia etc.), diversificarea modului de lucru (local sau rețea), precum şi a tipurilor de aplicații ce pot fi dezvoltate (locale, distribuite, multimedia etc). Acest lucru înseamnă că SGBD se doresc a avea o extensibilitate cât mai mare, care să le permită adaptarea la nou.

7. Utilizarea unor SGBD din generațiile precedente *în paralel* cu dezvoltarea unei noi generații.

8. Tendința de *fundamentare teoretică* şi *standardizare* a conceptelor din fiecare nouă generație de SGBD apărută, ceea ce oferă acestora robustețe şi deschidere.

1.2. Obiectivele unui SGBD

Obiectivul general al unui SGBD este de a furniza suportul software complet pentru dezvoltarea de aplicații informatice cu baze de date. În acest sens, pentru ca un produs software să fie SGBD, el trebuie să asigure *un set minim de obiective*, care va fi prezentat în continuare.



1. Asigurarea independenței datelor față de programe. Se spune că o aplicație informatică depinde de date, dacă modificarea structurii de memorare a datelor sau a strategiei de acces la date afectează şi aplicația. Independența datelor față de aplicație poate fi:

- **fizică**, adică modul de memorare a datelor şi tehnicile fizice de memorare (strategia de acces), pot fi schimbate fără a rescrie programele (exemplu în Oracle: se poate face acces secvențial la date, apoi se poate indexa şi se face un acces direct);
- **logică,** adică structura de date poate fi schimbată fără a rescrie programele (exemplu în Oracle: se poate adăuga în structură un nou câmp, prin comanda ALTER TABLE).

2. Redundanță minimă şi controlată a datelor. Spre deosebire de sistemele clasice (cu fişiere) de prelucrare automată a datelor, stocarea informațiilor în BD se face astfel încât datele să nu fie multiplicate. Cu toate acestea, uneori, pentru a realiza performanțe sporite, în ceea ce priveşte timpul de răspuns se acceptă o anumită redundanță a datelor. Aceasta va fi însă controlată pentru a se asigura coerența (corectitudinea datelor) BD. Exemplul de redundanță controlată acceptată este cea apărută la proiectare BD relaționale prin tehnica de normalizare.

3. Facilități de utilizare a datelor. Această facilitate presupune ca SGBD să aibă nişte componente specializate pentru diferite operații de utilizare:
- folosirea datelor de către *mai mulți utilizatori* în diferite scopuri (aplicații). Acest lucru reduce spațiul de memorare necesar şi efortul de încărcare / validare a datelor;
- *accesul cât mai simplu* al utilizatorilor la date, fără ca ei să fie nevoiți să cunoască structura întregii BD, acest lucru rămânând în sarcina administratorului BD (ex.: asistenți tip Wizard);
- existența unor *limbaje performante* de regăsirea a datelor care permit exprimarea interactivă a unor cereri de regăsire a datelor şi indicarea unor reguli pentru obținerea informațiilor solicitate (ex.: limbajul relațional SQL);
- oferirea posibilității unui *acces multicriterial* la date. SGBD stochează datele în entitățile BD şi permite mai multe căi de acces. Pentru diferite moduri de adresare SGBD creează dinamic, la momentul execuției, o serie de fişiere anexe (de index etc.) care lasă neschimbate entitățile BD.

4. Protecția datelor. În sistemele de BD, protecția datelor se asigură sub două aspecte: securitatea şi integritatea.

Securitatea (confidențialitatea) datelor semnifică faptul că accesul la date se face numai printr-o autorizare corespunzătoare şi doar controlat (sarcina administratorului BD). În acest sens, SGBD permite: autorizarea şi controlul accesului la date, utilizarea viziunilor, realizarea unor proceduri speciale, criptarea datelor.

a) *Autorizarea şi controlul accesului* la date este realizat de SGBD prin intermediul parolelor. Acestea identifică clasele de utilizatori, cu anumite drepturi de acces, la anumite date. Privilegiile diferiților utilizatori sunt gestionate de SGBD astfel: un anumit subiect (utilizator) poate realiza anumite acțiuni, asupra anumitor obiecte, în limita anumitor restricții (condiții suplimentare). Profilul utilizator este dat de nume (NAME), parola (PASS), nume grup, număr nivel de acces. Oracle deține în acest sens, pachetul OEM, dar şi comenzi SQL specializate.

b) *Utilizarea viziunilor* (view) este asigurată de SGBD pentru reprezentarea schemelor externe ale bazei de date. Cu ajutorul viziunilor, SGBD permite să se definească partiții logice ale bazei de date, pentru diferiți utilizatori, în raport cu cerințele acestora de acces la date. Securitatea datelor este asigurată de SGBD prin definirea tuturor drepturilor necesare unui utilizator pentru o viziune (GRANT) şi anularea unor drepturi pentru obiectele sale (REVOKE).

c) Realizarea unor *proceduri speciale* de acces asupra datelor este permisă de SGBD. Aceste proceduri sunt scrise în LMD, se păstrează în formă precompilată, iar anumitor utilizatori li se va acorda dreptul de execuție şi li se va interzice accesul direct la obiectele BD.

d) *Criptarea* este asigurată de SGBD prin oferirea unor rutine de criptare (codificare) a datelor, apelate automat sau la cerere şi prin existența unor instrumente care permit utilizatorului să realizeze propriile rutine de criptare. Criptarea şi decriptarea se realizează după algoritmi specifici, cu o cheie (parolă) de acces la rutină (Oracle foloseşte mai mulți algoritmi de criptare şi decriptare).

Integritatea datelor se referă la corectitudinea (coerența) datelor şi este asigurată prin protejarea acestora împotriva unor incidente intenționate sau neintenționate.

Componentele SGBD asigură integritatea datelor tratând separat cauzele care pot deteriora baza de date: integritatea semantică, controlul accesului concurent, salvarea / restaurarea.

a) *Integritatea semantică* este asigurată prin operații efectuate de SGBD asupra datelor şi a prelucrărilor. Aceste operații alcătuiesc un set de reguli denumit restricții de integritate. SGBD asigură astfel de restricții implicite (rezultă din modelul de date implementat – Oracle implementează cinci) şi explicite (proceduri incluse în programele de aplicație).

b) *Accesul concurent* asigură coerența datelor şi este un obiectiv al SGBD care se pune cu acuitate mai ales la baze de date distribuite. În acest sens SGBD foloseşte o unitate distinctă de prelucrare a datelor denumită *tranzacție*, care este constituită dintr-o secvență de operații – care se execută în totalitate sau deloc - marcată de puncte de început şi sfârşit. Tranzacția poate fi controlată de SGBD implicit, când punctele de început şi de sfârşit sunt automat definite, sau explicit, când punctele de început şi de sfârşit sunt definite prin comenzi specifice.

La execuția concurentă a tranzacțiilor SGBD trebuie să asigure *blocarea* datelor utilizate la un moment dat. Aceasta înseamnă că se interzice accesul celorlalte tranzacții concurente la aceleaşi date, până se termină tranzacția curentă. Tehnica de blocare utilizată de SGBD se poate aplica la nivelul întregii baze de date, a unui fişier, a unei înregistrări sau chiar a unui câmp (de exemplu în Oracle). Ea poate fi pentru citire (partajabilă) sau pentru scriere (exclusivă). Cele mai multe SGBD realizează blocarea la nivel de înregistrare şi fişier, prin diferite metode: setarea unui bit pentru resursa respectivă, construirea unei liste cu resursele blocate, menținerea resurselor blocate într-o zonă specială etc.

Inter-blocarea este situația în care două tranzacții blochează anumite resurse, apoi solicită fiecare resursele blocate de cealaltă. La nivelul de SGBD trebuie să existe facilitatea de prevenire sau rezolvare a inter-blocării.

*Prevenirea inter-blocării presupune că programele blochează toate resursele de care au nevoie încă de la începutul fiecărei tranzacții (greu de precizat).

*Soluționarea inter-blocării presupune că există nişte mecanisme pentru detectarea şi eliminarea inter-blocării (de exemplu graful dependențelor proceselor de executat).

c) *Salvarea / restaurarea* (backup/recovery) ca facilitate a SGBD permite refacerea consistenței datelor care au fost deteriorate fizic din diferite motive.

Salvarea datelor este un proces de stocare prin realizarea de copii de siguranță şi prin jurnalizarea tranzacțiilor şi a imaginilor. SGBD poate asigura salvarea automat şi la cererea administratorului bazei de date (de exemplu în Oracle).

Restaurarea porneşte de la colecțiile de date stocate prin salvare şi reface consistența bazei de date, minimizând prelucrările pierdute. Restaurarea este asigurată automat de SGBD, dar se poate realiza şi manual. *Restaurarea automată* a BD este realizată de SGBD cu ajutorul fişierelor jurnal. La nivelul SGBD pot exista o serie de parametri de configurare care influențează procesul de restaurare automată. Aceşti parametri se referă la: intervalul de restaurare, indicatorul de restaurare (ce informații vor fi scrise în fişierul de erori) etc.

Restaurarea manuală a BD implică intervenția administratorului pentru refacerea bazei de date de pe un suport tehnic care a fost distrus. Cea mai recentă copie de siguranță efectuată pentru o BD afectată este încărcată şi se reiau prelucrările efectuate din momentul copierii până la producerea defecțiunii. Restaurarea manuală se face prin deconectarea tuturor utilizatorilor de la BD, încărcarea copiei şi reluarea lucrului.

5. Partajabilitatea datelor

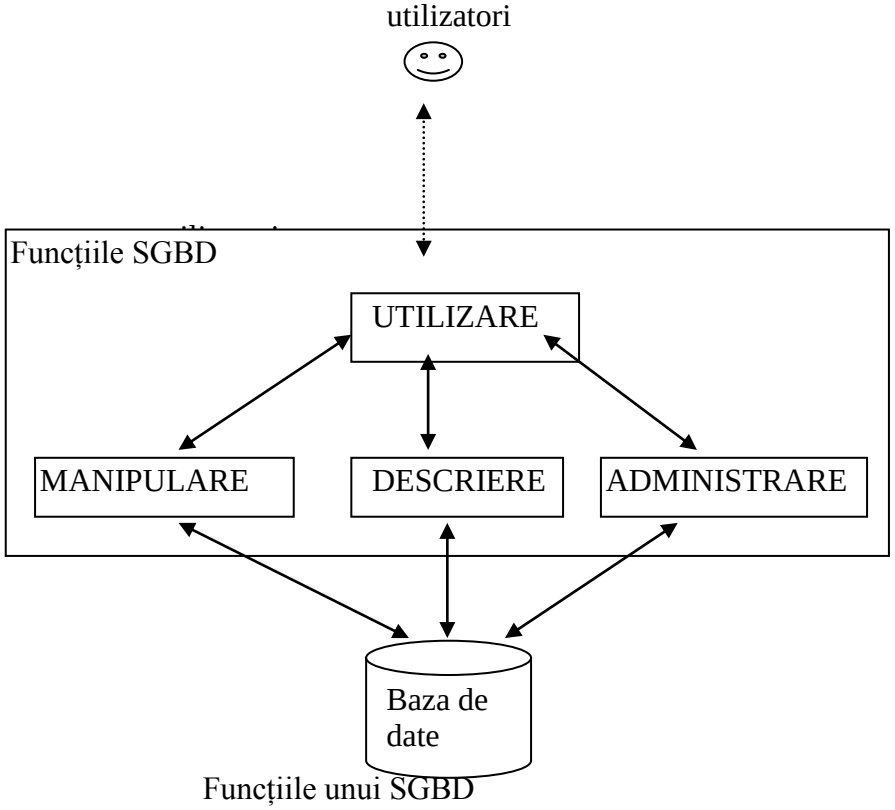
Partajabilitatea datelor se referă nu numai la aspectul asigurării accesului mai multor utilizatori la aceleaşi date, ci şi la posibilitatea dezvoltării unor aplicații fără a se modifica structura bazei de date. Problema partajabilității se pune la un nivel superior pentru SGBD-urile care permit lucrul în rețea.

6. Legăturile între date. Legăturile între date corespund asocierilor care se pot realiza între obiectele unei aplicații informatice. Orice SGBD trebuie să permită definirea şi descrierea structurii de date, precum şi a legăturilor dintre acestea, conform unui model de date. Fiecare tip de model de date permite anumite legături între date. Un SGBD, care implementează un anumit model de date, va trebui să asigure şi realizarea legăturilor dintre datele corespunzătoare în conformitate cu schema conceptuală.

7. Performanțele globale. Performanțele globale ale aplicației sunt influențate de SGBD. Acesta trebuie să gestioneze un volum mare de date de o complexitate ridicată, într-un anumit timp de acces util pentru diferiți utilizatori. Pentru toate aceste lucruri SGBD foloseşte diferite metode de acces, tehnici de optimizare, tipuri de date.

1.3. Funcțiile unui SGBD

Realizarea obiectivelor prezentate anterior este asigurată de SGBD printr-o serie de componente ce permit efectuarea unor operații specifice. În funcție de natura lor şi de scopul urmărit, operațiile pot fi grupate pe activități. Activitățile acceptă şi ele o grupare pe funcții astfel încât, una sau mai multe activități, relativ omogene, vor realiza o anumită funcție.



Funcțiile unui SGBD

1. Descrierea datelor

SGBD, prin această funcție, permite definirea structurii bazei de date cu ajutorul limbajului de definire a datelor - LDD. Definirea datelor poate fi realizată la nivel conceptual, logic şi fizic. Se descriu atributele (câmpurile) din cadrul structurii bazei de date, legăturile dintre entitățile bazei de date sau dintre atributele aceleiaşi entități, se definesc eventualele criterii de validare a datelor, metodele de acces la date, aspectele referitoare la asigurarea integrității datelor. Concretizarea acestei funcții este schema bazei de date, memorată în cod intern în dicționarul BD. Această funcție a fost mult automatizată în timp, LDD având acum puține comenzi (exemplu în SQL din Oracle sunt toate comenzile de tip CREATE şi ALTER). LDD este specific fiecărui SGBD, dar el întotdeauna realizează descrierea datelor conform elementelor modelului de date pe care îl implementează SGBD respectiv. După realizarea funcției de descriere, într-un SGBD, entitățile bazei de date există create deja ca fişiere, dar nu conțin datele propriu-zise, ci numai structura bazei de date (schema bazei de date).

2. Manipularea datelor

Funcția de manipulare a datelor este cea mai complexă şi realizează actualizarea şi regăsirea datelor din baza de date, cu ajutorul limbajului de manipulare a datelor (LMD). Sunt realizate următoarele activități referitoare la date: încărcarea, actualizarea, prelucrarea, regăsirea.
a) *Încărcarea* datelor în baza de date se realizează prin operații automatizate (restricțiile de integritate) sau programate ce asigură criteriile de validare a datelor.
b**)** *Actualizarea* bazei de date constă în operațiile de: adăugare, modificare, ştergere de înregistrări. La operațiile de adăugare şi de modificare se păstrează aceleaşi criterii de validare care s-au folosit şi la activitatea de încărcare a datelor. Actualizarea se realizează numai autorizat, prin asigurarea unei protecții corespunzătoare a datelor.
c) *Prelucrarea* datelor se realizează prin operațiile de: selecție, ordonare, inter-clasare (compunere), ventilare (descompunere) efectuate asupra entităților bazei de date. Acestea sunt, de obicei, operații pregătitoare activității de regăsire a datelor. Multe dintre operațiile de prelucrare sunt realizate cu ajutorul operatorilor din modelul de date implementat de SGBD.
d) *Regăsirea* (interogarea) datelor constă în operațiile de: vizualizare (afişare pe ecran, imprimare pe hârtie), răsfoire, editarea unor situații de ieşire. Situațiile de ieşire pot fi intermediare sau finale şi se pot obține pe diferiți suporți tehnici de informație (ecran, hârtie, mediu magnetic, mediu optic). Ele pot avea cele mai diferite forme (punctuale, liste, rapoarte, grafice, imagini, sunet, video) şi se pot obține după cele mai diferite criterii de regăsire.
Note.
- LMD pot fi cu limbaj gazdă sau cu limbaj propriu. Cele *cu limbaj gazdă* sunt dezvoltate prin adaptarea unor limbaje universale de programare (Cobol, Pascal, C etc.) la cerințele de lucru ale SGBD. Se îmbină astfel puterea unui limbaj universal cu necesitățile de regăsire a datelor (exemplu: ORACLE are limbajul PL-SQL). Cele *cu limbaj propriu* sunt dezvoltate printr-un limbaj specific capabil să unească puterea proceduralului cu regăsirea datelor dintr-un anumit tip de bază de date (exemplu: limbajul propriu din Visual Foxpro).

3. Utilizarea datelor

Funcția de utilizare a datelor asigură mulțimea interfețelor necesare pentru comunicarea tuturor utilizatorilor cu baza de date. Pentru a realiza această funcție SGBD trebuie să asigure facilități pentru mai multe categorii de utilizatori ai BD: neinformaicieni, specialişti, administratori.
a) Utilizatorii **neinformaticieni** reprezintă principala categorie a beneficiarilor de informații (utilizatorii finali şi intensivi) din baza de date. Aceşti utilizatori nu trebuie să cunoască structura bazei de date şi nu trebuie să ştie să programeze. În acest sens, SGBD oferă: meniuri cu opțiuni sugestive, ferestre, şabloane pentru diferite forme, asistenți tip Wizard, autodocumentarea (help, mesaje/ferestre explicative etc.).
b) Utilizatorii **specialişti** în informatică creează structura bazei de date şi realizează proceduri complexe de exploatare a bazei de date. SGBD oferă acestor utilizatori limbajul de descriere şi limbajul de manipulare a datelor, precum şi interfețe cu limbaje universale. Cu aceste elemente el descrie schema bazei de date şi asigură manipularea complexă a datelor (exemplu SQL şi PL-SQL în Oracle). Pentru realizarea bazei de date SGBD oferă specialistului şi elemente de CASE (Computer Aidede Software Engineering). Acestea îl ajută în diferitele activități care intervin în etapele de realizare a bazei de date (exemplu Oracle Designer).
c)**Administratorul** bazei de date, care este un utilizator special şi are un rol hotărâtor în ceea ce priveşte funcționarea optimă a întregului sistem. Datorită importanței acestei categorii de utilizatori, SGBD are o funcție distinctă în acest sens (exemplu Oracle Enterprise Manager).

4. Funcția de administrare

Funcția de administrare a datelor este de competența administratorului bazei de date. *Administratorul*, care are o bogată experiență de analiză, proiectare şi programare, organizează şi administrează baza de date în toate etapele de realizare a acesteia.
Astfel, el *organizează* baza de date conform unei anumite metodologii, realizează schema (conceptuală) bazei de date, coordonează proiectarea bazei de date. Pentru toate aceste lucruri SGBD oferă o serie de elemente de CASE, precum şi o serie de utilitare specializate.
În etapa de *exploatare* a bazei de date, administratorul are rolul de a autoriza accesul la date (acordă conturi, parole etc.), de a reface baza de date în caz de incidente (prin jurnalizare, copii), de a utiliza eficient spațiul de memorie internă şi externă (prin organizare, rutine de optimizare), de a realiza o serie de analize statistice din baza de date (număr şi tip de utilizatori, număr de accese, număr de actualizări etc.). Pentru fiecare din aceste activități SGBD oferă instrumente şi tehnici de lucru.
În cazul *lucrului în rețea* de calculatoare cu baze de date distribuite, SGBD are dezvoltate foarte mult componentele destinate administratorului. Acest lucru este determinat de faptul că baza de date este, în acest caz, de mare complexitate, datele sunt distribuite pe calculatoarele din rețea, iar utilizatorii sunt de toate tipurile şi în număr mare (exemplu Oracle RAC – Real Application Clusters).

1.4. Clasificarea SGBD

Diversele SGBD, care au fost şi care sunt în exploatare pe diferite calculatoare şi sub diferite sisteme de operare, impun o clasificare a lor după diferite criterii.

1) După *sistemele de calcul* pe care se implementează

- SGBD pentru calculatoare mari se folosesc pentru baze de date foarte complexe şi foarte mari (exemple: Oracle, DB2, IMS).
- SGBD pentru minicalculatoare se folosesc pentru baze de date complexe şi mari şi au cunoscut o dezvoltare puternică în anii '80 (exemplu: Oracle).
- SGBD pentru microcalculatoare se folosesc pentru baze de date de complexitate şi de mărime mici şi medii. Au o mare răspândire în momentul actual (exemple: Oracle, DB2 etc.).

Tendința actuală este ca SGBD să fie compatibil pe cât mai multe sisteme de calcul sub cât mai multe sisteme de operare.

2) După *limbajul de programare* utilizat

- SGBD *cu limbaj gazdă* este cel care are un limbaj de manipulare a datelor bazat pe unul de nivel înalt (universal). Avantajul acestei soluții este acela că se pot dezvolta proceduri complexe de program, se pot realiza interfețe om-maşină foarte bune, se valorifică experiența de programare din limbajele de nivel înalt (toate rezultă din avantajele programării procedurale). Dezavantajul major este acela că formularea cererilor de regăsire se face mai greu, de multe ori într-un mod inaccesibil utilizatorilor finali. (exemplu PL-SQL din Oracle).
- SGBD *cu limbaj propriu* (autonom) este cel care are un limbaj de manipulare a datelor specific. Acest limbaj de programare propriu este procedural şi are marele avantaj că permite implementarea tuturor facilităților oferite de SGBD. În el se pot programa proceduri complexe şi interfețe puternice ca într-un limbaj universal, dar în plus se realizează un acces uşor şi optimizat la baza de date. Dezavantajul este că un astfel de limbaj nu poate fi utilizat decât de specialiştii în informatică (exemplu limbajul din Visual FoxPro).

Tendința actuală este ca SGBD să aibă implementat, pe lângă un limbaj procedural, şi un limbaj de regăsire neprocedural, care să permită formularea de cereri de regăsire uşor, de către toți utilizatorii bazei de date. În acest sens, majoritatea SGBD pentru microcalculatoare au implementat, parțial sau total, limbajul SQL, care este şi standardizat internațional.

3) După *modelul logic de date* implementat

- SGBD *ierarhice* sunt cele care implementează modelul de date arborescent (ierarhic) şi au fost primele care s-au utilizat pentru gestionarea bazelor de date. Ele au o serie de avantaje pentru domenii precise din lumea reală înconjurătoare, de exemplu tehnologia construcției de maşini, dar au limite pentru alte domenii (exemplu: IMS).
- SGBD *rețea* sunt cele care implementează modelul de date rețea şi care au eliminat multe din limitele celor ierarhice. Ele au o largă aplicabilitate pentru numeroase probleme din lumea reală, dar sunt dificil de utilizat datorită complexității ridicate (exemplu: IDMS).
- SGBD *relaționale* sunt cele care implementează modelul de date relațional şi au aplicabilitate în majoritatea domeniilor din lumea reală. Ele pot fi folosite de o gamă largă de utilizatori datorită facilităților oferite (generatoare, limbaj neprocedural etc.) (exemple: Oracle, Visual FoxPro, Paradox, Access, Informix, Progress etc.).
- SGBD *orientate obiect* sunt cele care implementează modelul de date orientat obiect. Ele se pretează bine la problemele foarte mari, de complexitate ridicată, precum şi pentru tipurile noi de aplicații (proiectarea asistată, multimedia, sisteme deschise) (exemple: O2, Orion, Jasmin etc.).

Notă.

Tipurile de mai sus de SGBD, având drept criteriu modelul de date implementat, sunt de bază (fundamentale). Pornind de la acestea, prin extensia cu noi tehnologii informatice, există şi alte tipuri de SGBD: deductive, distribuite, multimedia, spațiale etc.

4) După *localizarea* bazei de date

- SGBD *centralizate* sunt cele care gestionează datele amplasate într-o singură bază de date centrală. La acestea au acces toți utilizatorii autorizați pentru a efectua diferite operații de manipulare a datelor. Toate calculatoarele care nu sunt legate în rețea şi lucrează cu baze de date au instalat un SGBD centralizat. Tot un SGBD centralizat, dar cu facilități de lucru în rețea, trebuie instalat şi în rețelele de calculatoare care au plasată baza de date pe un singur calculator (de obicei pe server).(exemplu: Visual FoxPro, Access)
- SGBD *distribuite* sunt cele care gestionează datele amplasate pe mai multe calculatoare dintr-o rețea tratându-le ca un tot unitar. Complexitatea acestor SGBD este ridicată, având componente speciale pentru realizarea conexiunilor şi tratarea distribuită a datelor (exemplu: Oracle, DB2, Informix).

1.5. Arhitecturi de SGBD

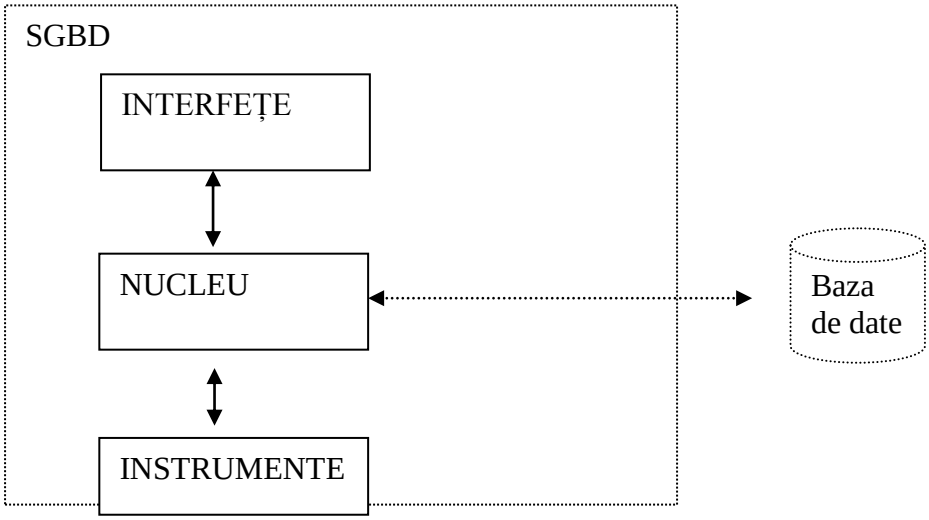
De la apariția lor şi până în prezent, SGBD au cunoscut o mare varietate. Există preocupări de standardizare a arhitecturii SGBD care caută să definească un cadru general al lor.
Dintre acestea, sunt două *arhitecturi de referință* a unui SGBD propuse de grupul de lucru CODASYL şi respectiv ANSI/SPARC.
In ultimul timp, arhitectura de SGBD *a evoluat* spre o configurație cu trei categorii de *componente* (nucleul, interfața, instrumentele), situație întâlnită la ultimele versiuni de sisteme comerciale (exemplu vezi arhitectura unui SGBDR).

Arhitectura pe componente (niveluri) a unui SGBD

Nu orice software care gestionează date în memoria externă este un SGBD:

- în primul rând, produsul software trebuie să fie un sistem, adică un ansamblu de programe intercorelate între ele care lucrează pentru un scop comun;
- mai departe, sistemul de programe trebuie să gestioneze date în memoria externă;
- datele nu pot exista oricum, ci organizate conform unui model de date într-o bază de date;
- în sfârşit, dacă sistemul de programe gestionează o bază de date şi în plus îndeplineşte funcțiile şi obiectivele specifice, atunci acel sistem este un SGBD.

Rezultă că un SGBD conține o serie de componente, care sunt instrumente software ce au scopul de a realiza funcțiile specifice: nucleul, interfețele, instrumentele.



Arhitectura pe componente a unui SGBD

Majoritatea arhitecturilor actuale de SGBD pot fi aduse la forma de mai sus, pe trei niveluri. Diferitele componente din diferitele SGBD (fiecare tip a venit cu una sau mai multe propuneri de arhitecturi) pot fi încadrate (uneori discutabil) în unul dintre cele trei niveluri.
Nivelurile din arhitectura de mai sus, pot conține următoarele *componente* ale unui SGBD:

• *Nucleul* (motorul) conține limbajul de descriere a datelor (LDD), limbajul de manipulare a datelor (LMD), componente obligatorii în *kit*-ul minim de SGBD. Componenta este destinată analiştilor, programatorilor şi administratorilor BD.

• *Interfețele* sunt formate din: generatoarele de diferite tipuri (de meniuri, de video-formate, de rapoarte etc.), elementele de CASE (Computer Aided Software Engeenering), interfețe cu limbaje de programare universale, interfețe cu alte sisteme etc. Componenta este destinată tuturor categoriilor de utilizatori: finali, intensivi, specialişti.

• *Instrumentele* sunt formate din: editoarele, navigatoarele (browsers), utilitarele (shells) de diferite tipuri. Componenta este destinată, în principal, administratorului bazei de date, dar şi altor categorii de utilizatori.

Notă. Arhitectura pe niveluri este simplă dar completă:

- arhitecturile standardizate (CODASYL şi ANSI) pot fi aduse pe cele trei niveluri;
- noile tehnologii de baze de date (de exemplu tehnologia orientată obiect) au determinat apariția unor noi tipuri de SGBD. Arhitecturile propuse pentru acestea, pot fi adaptate pe cele trei niveluri;
- noile tehnologii informatice (de exemplu multimedia, Internet etc.) au interferat cu tehnologia bazelor de date rezultând SGBD derivate corespunzătoare. Pentru acestea s-au propus arhitecturi care, însă, pot fi adaptate la structurarea pe trei niveluri.

1.6. Teste de autoevaluare

1. *Care din următoarele sunt obiective ale unui SGBD?*
   A. Redundanță minimă şi controlată a datelor;
   B. Administrarea bazei de date;
   C. Asigurarea facilităților de utilizare a datelor;
   D. Asigurarea dependenței datelor;
   E. Asigurarea protecției datelor;
2. *Care din următoarele nu este un model de date implementat de SGBDuri la organizare datelor:*
   A. Modelul ierarhic;
   B. Modelul claselor;
   C. Modelul orientat pe servicii;
   D. Modelul relațional;
   E. Modelul orientat obiect;
3. *In evolutia SGBD-urilor se constată:*
   A. Două generații;
   B. Patru generații;
   C. Schimbarea fundamental a aspectelor fundamentale;
   D. Reducerea accesului concurent;
   E. Restricționarea tipurilor de date folosite;
4. *După limbajul de programare utilizat, SGBDuri sunt:*
   A. Cu limbaj gazdă;
   B. Cu limbaj propriu;
   C. Fără niciun limbaj;
   D. Ierarhice;
   E. Relaționale;
5. *Arhitecturi de referință pentru un sistem de baze de date sunt:*
   A. LMD;
   B. CASE;
   C. ANSI;
   D. Pe rețea;
   E. Pe componente;

Răspunsuri la testele de autoevaluare

1. A, C, E
2. B,C
3. –
4. A, B
5. C, E

Bibliografia unității de învățare 1

1. M. Velicanu, I. Lungu s.a. – *Sisteme de baze de date – teorie şi practică*, ed. Petrion, Bucureşti, 2003.
2. J. Date – *An introduction to database systems*, Ed. Addison Wesley, 2004.
3. M. Velicanu – *Dicționar explicativ al sistemelor de baze de date*, Ed. Economică, Bucureşti, 2005.

Cuprinsul cursului

Unitatea de învățare 2

SISTEME DE GESTIUNE A BAZELOR DE DATE RELAȚIONALE (SGBDR) CU EXEMPLIFICARE ÎN ORACLE

2.1. Definirea SGBDR

SGBDR este un sistem software complet care implementează modelul de date relațional, precum și cel puțin un limbaj de programare relațional.
Teoria relațională este un ansamblu de concepte, metode şi instrumente care a dat o fundamentare riguroasă realizării de SGBDR performante.
Paralela între conceptele utilizate în evoluția organizării datelor în memoria externă până la sistemele relaționale.

FIŞIERE	TEORIA BD	TERIA RELAȚIONALĂ	SGBDR
Fişier	Colecție de date	Relație	Tabela
Înregistrare	Familie de caracteristici	Tuplu	Linie
Câmp	Caracteristică	Atribut	Coloană
Valoare	Domeniu de valori	Domeniu	Domeniu

Regulile lui Codd

E.F. Codd a formulat 13 reguli care exprimă cerințele maximale pentru ca un SGBD să fie relațional.
Regulile sunt utile pentru evoluarea performanțelor unui SGBDR.
R0. Gestionarea datelor la nivel de relație: limbajele utilizate trebuie să opereze cu relații, care constituie unitatea elementară de informație.
R1. Reprezentarea logică a datelor: toate informațiile din BD trebuie stocate şi prelucrate ca tabele.
R2. Garantarea accesului la date: LMD trebuie să permită accesul la fiecare valoare atomică din BD (tabelă, coloană, cheie).
R3. Valoarea NULL: trebuie să se permită declararea şi prelucrarea valorii NULL ca date lipsă sau inaplicabile.
R4. Metadatele: informațiile despre descrierea BD se stochează în dicționar şi se tratează ca tabele, la fel ca datele propriu-zise.
R5. Limbajele utilizate: SGBD trebuie să permită utilizarea mai multor limbaje de programare, dintre care cel puțin unul să permită definirea tabelelor (de bază şi virtuale), definirea restricțiilor de integritate, manipularea datelor, autorizarea accesului, tratarea tranzacțiilor.
R6. Actualizarea tabelelor virtuale: trebuie să se permită ca tabelele virtuale să fie şi efectiv actualizabile, nu numai teoretic actualizabile.
R7. Actualizările în baza de date: manipularea unei tabele trebuie să se facă prin operații de regăsire dar şi de actualizare.
R8. Independența fizică a datelor: schimbarea structurii fizice a datelor (modul de reprezentare (organizare) şi modul de acces) nu afectează programele.
R9. Independența logică a datelor: schimbarea structurii de date (logice) a tabelelor nu afectează programele.

R10. Restricțiile de integritate: acestea, trebuie să fie definite prin LDD şi stocate în dicționarul (catalogul) BD.
R11. Distribuirea geografică a datelor: LMD trebuie să permită ca programele de aplicație să fie aceleaşi atât pentru datele distribuite cât şi pentru datele centralizate (alocarea şi localizarea datelor vor fi în sarcina SGBD).
R12. Prelucrarea datelor la nivel de bază (scăzut): dacă SGBD posedă un limbaj de nivel scăzut (prelucrarea datelor se face la nivel de înregistrare), acesta nu trebuie utilizat pentru a evita restricțiile de integritate.

2.2. Limbajele relaționale

SGBDR *oferă* seturi de comenzi pentru descrierea şi manipularea datelor. Acestea pot fi incluse într-un singur limbaj relațional (cazul cel mai întâlnit) sau separate în LDD şi LMD. În ambele situații, comenzile pentru definirea datelor sunt distincte de cele pentru manipularea datelor. *Exemple* de limbaje relaționale: SQL (Structured Querry Language - standarde începând cu 1985), QUEL, QBE, SQUARE, ALPHA, ISBL.

a) Partea de definire a datelor (LDD)
LDD este simplificat, cu puține comenzi.
Descrierea datelor este memorată în BD, sub formă de tabele, în dicționarul (metabaza) bazei de date. La nivel conceptual

- Crearea unei BD (dicționarul BD): CREATE DATABASE
- Ştergerea unei BD: DROP DATABASE
- Crearea tabelelor de bază: CREATE TABLE
- Ştergerea tabelelor de bază: DROP TABLE
- Crearea de sinonime: CREATE SYNONYM
- Ştergerea sinonimelor: DROP SYNONYM
- Actualizarea structurii unei tabele: ALTER TABLE cu opțiunile ADD, MODIFY, DROP
- Adăugarea restricțiilor de integritate :ASSERT ON. În Oracle restricțiile de integritate sunt: NULL, CHECK, pe cheie (PRIMARY , UNIQUE, REFERENTIAL).

La nivel logic

- Crearea tabelelor virtuale: CREATE VIEW
- Ştergerea tabelelor virtuale: DROP VIEW
- Acordarea drepturilor de acces la BD:

GRANT CONNECT – conectarea la BD a unui utilizator.
GRANT drepturi – acordarea unor drepturi de acces (pentru regăsire, actualizare etc.).

- Retragerea drepturilor de acces la BD:

REVOKE drepturi – retragerea unor drepturi.
REVOKE CONNECT – deconectarea unui utilizator de la BD.
La nivel fizic

- Crearea indecşilor: CREATE INDEX
- Ştergerea indecşilor: DROP INDEX
- Controlul alocării spațiului fizic al BD:

CREATE SPACE –creează un model de alocare a spațiului fizic pentru o BD
ALTER SPACE –actualizarea modelului de alocare a spațiului fizic
DROP S PACE – şterge un model de alocare a spațiului fizic

- Regruparea fizică a datelor dintr-o BD (clustere):

CREATE CLUSTER – creează un cluster dintr-o BD
ALTER CLUSTER– actualizează un cluster
DROP CLUSTER – şterge un cluster

b) Partea de manipulare a datelor (LMD)
Aspectele care pot caracteriza LMD sunt diverse şi din acest motiv, ele pot fi grupate în trei *categorii*: generale, funcționale, calitative.
b1) Caracteristici generale ale LMD

1. *Tratarea datelor la nivel de ansamblu.* Toate LMD relaţionale realizează o tratare la nivel de *ansamblu* a datelor: unitatea de informative pentru lucru este *tabela*. La comunicarea unui LMD relaţional cu un limbaj universal, avantajele se pierd deoarece comunicarea se poate face doar tuplu cu tuplu şi nu la nivel de ansamblu. Deoarece limbajele universale oferă alte avantaje legate de proceduralitate, soluţia este de a integra în acestea un limbaj relaţional. *Cursorul* este soluţia în SGBDR pentru a face trecerea de la tratarea la nivel de ansamblu la cea la nivel de înregistrare (tuplu).

2. *Operatorii relaţionali* implementaţi. SGBDR s-au dezvoltat, din punct de vedere relaţional, având la bază:

- calculul relaţional orientat pe tuplu (ALPHA, QUEL).
- calculul relaţional orientat pe domeniu (QBE).
- algebra relaţională (ISBL)
- transformarea (mapping) (SQL, SQUARE)

Notă. Limbajele bazate pe calculul relaţional sunt neprocedurale, cele bazate pe algebra relaţională sunt procedurale, celelalte sunt combinaţii.

3. *Realizatorii* limbajelor relaţionale s-au orientat pe domenii precise din teoria relaţională. Astfel, au rezultat: limbaje relaţionale standardizate internaţional (exemplu SQL - ANSI), limbaje cu standard de utilizare impus de constructor (exemplu QUEL), limbaje nestandardizate (celelalte limbaje relaţionale).

4. *Utilizatorii* limbajelor relaţionale sunt mult diversificaţi. SGBDR oferă atât elemente procedurale (pentru specialişti) cât şi neprocedurale (pentru nespecialilşti).

b2) Caracteristici funcţionale ale LMD

1. Facilităţile de interogare a datelor. Acestea sunt puternice şi sunt oferite prin comenzi pentru interogarea tabelelor de bază, dar şi a celor virtuale (exemplu SELECT).

2. Facilitătile de actualizare a datelor sunt oferite prin comenzi pentru actualizarea tabelelor de bază, dar şi a celor virtuale:

INSERT INTO – adaugă rânduri la sfârşitul unei tabele.

UPDATE – modifică rânduri dintr-o tabelă.

DELETE FROM - şterge rânduri dintr-o tabelă.

Notă. Unele SGBDR nu permit actualizarea tabelelor virtuale, altele permit acces lucru cu o serie de restricţii pentru ca operaţia să se propage spre tabelele de bază fără ambiguităţi.

3. Alte facilităţi funcţionale. La facilităţile relaţionale de mai sus, SGBDR oferă şi alte facilităţi, pe care le au toate limbajele de programare procedurale:

- Calculul aritmetic prin operatorii specifici: +, -, *, /, **
- Agregarea: prin funcţii standard (exemplu SUM), prin comenzi (exemplu COMPUTE OF expr).
- Comenzi de intrare/ieşire standard: ACCEPT…PROMPT… (în Oracle).

b3) Caracteristici calitative ale LMD

1. *Puterea selectivă* a LMD relaţionale este dată de posibilitatea selectării datelor după criterii (filtre) complexe (exemplu comanda SELECT).

2. *Uşurinţa de învăţare şi utilizare* este nuanţată în funcţie de tipul LMD relaţional.

- Cele bazate pe calculul relaţional sunt neprocedurale (descriptive), deci uşor de învăţat şi utilizat (apropiat, ca stil, de limbajul natural) (exemplu QUEL).

• Cele bazate pe algebra relațională sunt procedurale (algoritmice), deci mai greu de învățat şi utilizat (exemplu ISBL).

• Cele intermediare promovează stilul neprocedural dar acceptă şi elemente de control procedural (exemplu SQL).

• Cele bazate pe grafică oferă primitive grafice pentru machetarea cererilor de regăsire, deci uşor de utilizat (exemplu QBE).

3. *Eficacitatea utilizării* este determinată de posibilitatea optimizării cererilor de regăsire.

LMD bazate pe calculul relațional lasă compilatorul să aleagă ordinea de execuție a operațiilor, deci rezultă o eficiența mare.

LMD bazate pe algebra relațională au o ordine impusă pentru execuția operațiilor, deci rezultă o eficiență mica.

2.3. Mecanisme de optimizare în SGBDR

a) Mecanisme pentru protecția datelor

Pentru atingerea acestui obiectiv, orice SGBDR trebuie să aibă implementate mecanisme specifice pentru ambele aspecte privind datele: integritatea, securitatea.

a1) Controlul integrității (corectitudinea) datelor

- Mecanismul de tranzacții.

Toate SGBDR sunt sisteme tranzacționale, deci unitatea de bază pentru prelucrare este tranzacția, care va respecta automat restricțiile de integritate.

Tranzacția este o secvență de instrucțiuni care formează un tot unitar şi care se execută în totalitate sau deloc.

În Oracle există tranzacții implicite (referirea SQL%) şi explicite (Begin şi End Tranzaction).

- Mecanismul de blocare.

Dacă mai mulți utilizatori doresc simultan să acceseze aceleaşi date atunci SGBDR deține mecanisme specifice pentru a evita conflictele şi apariția unor erori: blocarea, interblocarea.

Blocarea este situația în care mai multe tranzacții accesează în acelaşi timp, aceleaşi date. SGBDR va atribui priorități tranzacțiilor, după anumite criterii, execuția lor fiind făcută în ordinea acestora. Blocarea se poate face la nivel de: BD, tabelă, tuplu, atribut.

Inter-blocarea este situația în care tranzacția T1 accesează datele D1, iar tranzacția T2 accesează datele D2. La un moment dat T1 are nevoie de datele D2, iar T2 are nevoie de date D1. SGBDR rezolvă conflictul prin priorități acordate tranzacțiilor.

- Fişierele jurnal.

O soluție pentru asigurarea corectitudinii datelor este aceea ca la apariția anumitor evenimente să fie salvate (backup) datele în fişiere de rezervă (jurnal). Acestea vor fi restaurate (recovery) atunci când BD va fi stricată, din anumite motive, în vederea refacerii ei.

În Oracle fişierele jurnal sunt create, întreținute şi utilizate automat de către sistem, administratorul BD putând folosi aceste fişiere.

a2) Controlul securității (accesului) datelor

- Profilele utilizator.

La o BDR pot avea acces doar utilizatorii autorizați. În acest sens, administratorul BD construieşte un profil utilizator (categorie, grupă, nume, parolă, drepturi) restricționând accesul acestuia la BD.

- Tabelele virtuale.

La un moment dat, un anumit utilizator trebuie să vadă doar acea parte din BD la care are dreptul de acces. Acest lucru poate fi restricționat prin construirea de către proiectantul BD a unor viziuni (tabele virtuale), pe care utilizatorul le va apela.

Viziunea este o construcție logică pentru o cerere de regăsite, memorată în dicționarul BD sub o denumire. La apelul viziunii cererea de regăsire este executată şi se accesează date pe o parte din BD.

În Oracle viziunea se construieşte prin comanda CREATE VIEW şi apoi poate fi utilizată prin comenzi din SQL, la fel ca orice tabelă relațională.

- Bibliotecile de sistem.

O modalitate de restricționare a accesului la date este dată de procedurile stocate. Acestea sunt secvențe de instrucțiuni, stocate sub o denumire, de unde le pot apela utilizatorii autorizați.

În Oracle pot fi scrise proceduri stocate în PL-SQL, atât structurate cât şi orientate obiect. Stocarea se poate face fie în memoria internă (volatile), fie în memoria internă (persistente).

- Algoritmii de criptare.

Anumite date nu trebuie să apară în clar în BD şi atunci ele vor fi codificate (criptate). În acest sens, se utilizează un algoritm de criptare şi o cheie de acces la acesta. De asemenea, există algoritmi pentru decriptare, cu cheie de acces.
Oracle foloseşte mai mulți algoritmi de criptare-decriptare, cu chei de acces.

b) Mecanisme de optimizare a regăsirii/interogării

Operația de regăsire a datelor este una dintre cele mai importante şi mai utilizate în aplicațiile cu BDR. Puterea de regăsire a limbajelor relaționale, în special SQL, nu a fost egalată de nici un alt tip de limbaj de programare.

b1) Transformările relaționale

O cerere de regăsire se scrie într-un limbaj relațional cu ajutorul expresiilor relaționale bazate pe calculul relațional sau pe algebra relațională. Cele două tipuri de expresii sunt echivalente şi pot oricând să fie rescrise în celălalt tip.
SGBDR trebuie să fie capabil să *transforme* o expresie scrisă în calculul relațional în una echivalentă în algebra relațională şi invers, pentru a se putea realiza o regăsire optimă.
Transformarea relațională se poate realiza prin două *strategii de optimizare*: generale, specifice.

- Strategiile *generale*: sunt independente de modul de memorare al datelor şi se bazează pe proprietățile operațiilor din algebra relațională – comutativitatea, asociativitatea, compunerea. Astfel de strategii sunt:
  - ✓ selecția înaintea joncțiunii;
  - ✓ proiecția înaintea joncțiunii;
  - ✓ selecția înaintea proiecției;
  - ✓ combinarea regăsirii multiple.
- Strategiile *specifice* țin cont de modul de memorare al datelor şi sunt caracteristice unui anumit SGBDR. Elementele care influențează executarea operațiilor care intervin la o cerere de regăsire sunt: accesul direct sau secvențial, reguli de ordonare a expresiilor algebrice etc.

Oracle aplică ambele strategii de optimizare pentru transformările relaționale.

b2) Optimizarea alocării

Prin definiție BD sunt mari consumatoare de spațiu calculator, atât în memoria internă cât şi în memoria externă. Pentru ca aceste spații să fie cât mai eficient folosite, SGBDR folosesc diferite *mecanisme*: reutilizarea, realocarea, virtualizarea, zone de memorie (buffers) cu destinații speciale, compactarea datelor (în binar , cu algoritmi de compactare), alocarea dinamică etc.
Toate aceste mecanisme presupun algoritmi specifici.
Oracle face o alocare optimă atât în memoria internă cât şi în memoria externă (vezi Elementele unei BD Oracle şi Mecanismele interne Oracle).

b3) Optimizarea accesului

Căutarea datelor într-o BD a fost întotdeauna o operație care consumă mult timp calculator. Volumul mare de date din BD şi complexitatea ridicată a acesteia determină consum de timp calculator la operația de regăsire.
SGBDR au cele mai bune performanțe d.p.d.v. al regăsirii datelor, dintre toate tipurile de SGBD, datorită limbajelor relaționale (mai ales SQL) pe care le au implementate. Totuşi, dacă volumul de date este foarte mare atunci limbajele relaționale ajung la limită, datorită numărului mare de operatori relaționali pe care trebuie să-i aplice la regăsire.
Oracle implementează o variantă extinsă de SQL standard cu o comandă SELECT deosebit de puternică.

2.4 Avantajele şi limitele sistemelor relaționale

Avantaje

1. Simplitatea conceptelor şi a schemei .
2. Teoria relațională oferă un solid suport teoretic şi o bază pentru cercetări ulterioare.
3. Un grad mare de independență a datelor față de programe.
4. Limbajele relaționale sunt declarative (de nivel înalt) şi au o mare putere de regăsire.
5. Ameliorarea semnificativă a protecției datelor, sub toate aspectele.
6. Optimizarea semnificativă a accesului la date, precum şi a alocării datelor.
7. Manipularea de ansambluri de date prin operatorii din calculul sau algebra relațională, cu implicații importante pentru regăsirea datelor.

Limite

1. Prea marea simplitate a modelului relațional, care pentru tipurile noi de aplicații (Internet, sisteme deschise etc.) conduce la:
   - pierderea unor informații semantice utile (prin multiplicarea tabelelor la normalizare);
   - operațiile relaționale, chiar optimizate, sunt costisitoare (noile aplicații generează multe operații relaționale) din punctul de vedere al resurselor de calcul.
2. LMD relaționale sunt prea limitate, ceea ce generează disfuncționalități :
   - programatorul trebuie să cunoască două tipuri de limbaje (declarativ - relațional şi procedural - universal). De aici rezultă necesitatea conversiilor, o fiabilitate scăzută, necesitatea comunicării, productivitatea scăzută;
   - mecanismele de optimizare privesc doar LMD relațional, deci ceea ce este scris în limbaj procedural trebuie optimizat de către programator.

2.5. Teste de autoevaluare

1. *Noțiuni similare între fişiere şi teoria relațională sunt*
   A. Fişier-tuplu;
   B. Câmp-atribut;
   C. Câmp-relație;
   D. Valoare-domeniu;
   E. Nu există noțiuni similare;
2. *Regulile lui CODD se referă la:*
   A. Valoarea NULL;
   B. Valorile TRUE şi FALSE;
   C. Garantarea accesului la date;
   D. Dependența logică a datelor;
   E. Restricțiile de integritate;
3. *Comenzi LDD la nivel logic sunt:*
   A. CREATE VIEW;
   B. SELECT from VIEW;
   C. GRANT;
   D. COMMIT;
   E. DELETE VIEW;
4. *Comenzi LDD la nivel fizic sunt:*
   A. CREATE INDEX;
   B. UPDATE INDEX;
   C. VIEW CLUSTER;
   D. DROP CLUSTER;
   E. REVOKE;
5. *Avantajele sistemelor relaționale sunt:*
   A. Optimizarea semnificativă a accesului la date, precum şi a alocării datelor;
   B. Ameliorarea semnificativă a protecției datelor, sub toate aspectele;
   C. Un grad mare de dependență a datelor față de programe;
   D. Operațiile relaționale, chiar optimizate, sunt costisitoare;
   E. Necesitatea conversiilor implicite;

Răspunsuri la testele de autoevaluare

1. B, D
2. A, C, E
3. A, C
4. A, D
5. A, B

Bibliografia unității de învățare 2

1. M. Velicanu, I. Lungu s.a. – *Sisteme de baze de date – teorie şi practică*, ed. Petrion, Bucureşti, 2003.
2. J. Date – *An introduction to database systems*, Ed. Addison Wesley, 2004.
3. M. Velicanu – *Dicționar explicativ al sistemelor de baze de date*, Ed. Economică, Bucureşti, 2005.
4. A. Bara, I. Botha, V. Diaconița, I. Lungu, A. Velicanu – *Baze de date. Limbajul PL/SQL*, ed. ASE, Bucureşti, 2009.

Cuprinsul cursului
Unitatea de învățare 3
SISTEMUL ORACLE

3.1. Elementele unei BD Oracle şi mecanismele interne Oracle

În contextul de SGBD relaţional, Oracle foloseşte o mulţime de *concepte şi mecanisme interne,* care au semnificaţia prezentată în continuare.

Elementele unei BD Oracle

Procesul este o prelucrare (job) a sistemului care presupune un program unic, ce rulează, la un moment dat, cu propria lui zonă de lucru.

Zona de lucru este o parte din memoria internă (buffer) alocată de sistemul Oracle pentru diferite operaţii temporare: execuţia tranzacţiilor, conectarea utilizatorilor – sesiunea de lucru etc.

Indexul este o structură fizică de date asociată unei tabele sau unui cluster. Indexarea are rolul de a obţine date ordonate după diferite criterii (chei). Cheia de indexare este formată din una sau mai multe coloane dintr-o tabelă. Oracle foloseşte patru metode de indexare: arbori B simpli, arbori B compuşi, cheie inversă, bitmap.

Viziunea (view) este o tabelă virtuală construită din una sau mai multe tabele de bază, fără a ocupa spaţiul fizic. Ea este, de fapt, o cerere de regăsire SELECT stocată în dicţionarul de date şi contribuie la creşterea protecţiei datelor.

Clusterul este o grupare fizică a informaţiilor din una sau mai multe tabele şi are rolul de a creşte eficienţa regăsirii datelor.

Mecanismele interne Oracle

Contextul o zonă de memorie extensibilă dinamic, alocată şi gestionată de Oracle pentru a putea executa o comandă.

Cursorul este un pointer (identificator) spre zona de context necesar manipulării acesteia. Oracle foloseşte două tipuri de cursor: implicit şi explicit. Cursorul implicit este creat şi gestionat automat de sistem şi utilizat pentru o serie de comenzi pentru definirea sau manipularea datelor (CREATE, ALTER, INSERT, DELETE etc.). Cursorul explicit este creat şi gestionat de programator, în special pentru cererile de regăsire (SELECT).

Tranzacţia este un ansamblu omogen de operaţii (în special actualizări) într-o sesiune utilizator. Ea are rolul de a asigura coerenţa datelor şi funcţionează binar: se validează în totalitate sau deloc. Tranzacţia poate fi descompusă în subtranzacţii prin stabilirea unor puncte de salvare (SAVEPOINT).

Pseudo-coloane şi pseudo-funcţii sunt variabile de sistem utilizate pentru a avea acces la parametrii interni ai sistemului. Ele se pot utiliza pentru tabelele utilizator sau pentru cele standard (exemple: LEVEL, ROWID, NEXVAL şi respectiv USER, UID, SYSDATE).

3.2. Subprogramele PL/SQL

Subprogramele sunt module de program care efectuează o anumită prelucrare şi care se apelează dintr-un program oarecare.
PL/SQL permite dezvoltarea subprogramelor, ca orice limbaj de programare procedural, utilizarea acestora oferind *avantajele* cunoscute:
- încadrarea în tehnica de programare structurată şi respectiv modulară;
- concentrarea prelucrărilor pe operaţii omogene, specifice;
- diminuarea numărului de linii sursă;
- structurarea prelucrărilor mai des utilizate, într-un singur nume şi un singur cod sursă;
- parametrarea prelucrărilor, deci generalizarea programelor;
- simplificarea realizării programelor de aplicaţie precum şi uşurinţă în întreţinerea lor, deci posibilitatea lucrului eficient, în echipă.

Pe lângă tipurile de subprograme cunoscute din toate limbajele de programare procedurale (funcţii şi proceduri), PL/SQL are două tipuri specifice (pachete şi declanşatori).
Aşadar *tipurile* de subprograme suportate de PL/SQL sunt:
- procedurile – module de program cu nume, parametri de intrare şi ieşire;
- funcţiile – module de programe cu nume, parametri de intrare şi care întorc o singură valoare obţinută prin evaluare;
- pachetele – module care reunesc obiecte din bază ce grupează prelucrări cu legături între ele;
- declanşatorii (triggers) – module ce au ca efect declanşarea unei acţiuni automat, în funcţie de anumite evenimente care au loc la momentul execuţiei programului.

Procedurile şi funcţiile
Procedurile (procedure) şi funcţiile (function) utilizator sunt module de program *asemănătoare.* Astfel, ambele sunt scrise de programator, ambele au un nume sub care se definesc şi apoi apelează, ambele au parametri de intrare.
La apelul procedurilor/funcţiilor se părăseşte execuţia programului apelant (principal) se face salt la modulul indicat (programul apelat) prin nume, se execută subprogramul, apoi se revine la programul principal la următoarea instrucţiune de după cea de apel.
Deosebirea este că funcţia poate primi nici unul sau oricâţi parametri de intrare, dar nu are nici un parametru de ieşire. Întotdeauna funcţia întoarce o singură valoare dată de evaluarea expresiei care se atribuie numelui funcţiei.
Procedura poate primi oricâţi parametri de intrare şi poate întoarce oricâţi parametri de ieşire.
Covenţiile privind parametrii (formali - cei din instrucţiunea de apel, reali – cei de la definirea subprogramului), cunoscute din toate limbajele procedurale, rămân valabile şi în PL/SQL:
- trebuie să fie acelaşi număr de parametri;
- parametrii trebuie să fie în aceeaşi ordine;
- tiupul pentru fiecare parametru corespunzător trebuie să fie la fel;
- numele parametrilor din cele două categorii pot fi aceleaşi sau diferite (pentru că referirea parametrilor se face prin adresă).

In PL/SQL se pot construi astfel de subprograme pe două *niveluri:*
- Nivelul aplicaţiei (blocuri cu nume) – sub un generator (de exemplu Forms), în care se dezvoltă aplicaţia, se crează asistat procedurile/funcţiile dorite. Aceste subprograme sunt accesibile doar în cadrul aplicaţiei în care se dezvoltă.

- Nivelul bazei de date – se crează explicit prin cod sursă (CREATE PROCEDURE / FUNCTION) subprogramele dorite. Acestea se stochează în dicționarul bazei de date şi vor fi disponibile tuturor celor care au drept de acces la baza de date.

Pachetele

Ca subprogram, pachetul (package) este un obiect tip Oracle definit la nivelul bazei de date, care reunește un ansamblu de prelucrări efectuate asupra aceluiaşi domeniu sau aceloraşi obiecte.
În PL/SQL subprogramul de tip pachet este compus din *două părți*:
PACKAGE SPECIFICATION – partea publică , declarativă, care poate fi cunoscută de toți utilizatorii care au dreptul de acces.
PACKAGE BODY – partea privată, corpul pachetului, care conține definițiile obiectelor declarate anterior, accesibile doar local prin componentele pachetului.
Subprogramul de tip pachet se poate crea prin instrucțiunea CREATE PACKAGE şi el poate *conține*: proceduri, funcții, variabile, cursoare, excepții.
Necesitatea subprogramelor de tip pachete în aplicațiile cu baze de date dezvoltate în PL/SQL este dată de *avntaje*le acestora:

- întreținerea şi dezvoltarea prelucrărilor sunt facilitate de regruparea şi structurarea prelucrărilor elementare, prin modularizare;
- se permite o gestiune mai fină a drepturilor de acces la proceduri şi funcții (care sunt incluse în pachete), datorită conceptelor de structurare privată şi respecxtiv publică;
- un pachet poate fi compilat atât în întregime, ca un tot unitar, cât şi parțial, ceea ce extinde modularizarea.

Declanşatorii

Subprogramele de tip declanşatori (trigers) sunt blocuri PL/SQL care realizează o anumită acțiune, lansată automat atunci când apare un anumit eveniment.
Declanşatorii pot fi creați şi utilizați pe două *niveluri:*

- nivelul aplicației (blocuri cu nume) – sub un generator (de exemplu Forms), în care se dezvoltă aplicația, se crează asistat declanşatorul dorit (de exemplu cu DESIGNER). În acest caz subprogramul este accesibil doar în cadrul aplicației în care se dezvoltă. Se permite un control sporit al intrărilor;
- nivelul bazei de date – se crează explicit prin cod sursă (CREATE TRIGGER) subprogramele dorite. Acestea se stochează în dicționarul bazei de date şi vor fi disponibile tuturor celor care au drept de acces la baza de date. Permite să se implementeze reguli de gestiune complexe şi să se intensifice (măsuri suplimentare) protecția datelor.

Într-un subprogram de tip declanşator *se specifică:*

- tipul declanşatorului, care poate fi posterior (AFTER) sau anterior (BEFOR);
- evenimentul declanşator, care poate fi: execuția unei instrucțiuni, click de mouse, apăsarea unei taste, apariția unei erori etc.;
- tratamentul efectuat, adică acțiunea care se desfăşoară.

3.3 Introducere în Oracle

Sistemul Oracle a fost, încă de la apariția sa la sfârşitul anilor 70, un SGBD care a respectat în totalitate teoria relațională. Acest lucru înseamnă că sistemul a respectat cel puțin două cerințe minime:

- a implementat modelul de date relațional pentru baze de date;
- a implementat un limbaj de programare relațional (SQL – Structured Query Language).

De asemenea, sistemul Oracle se încadra încă de atunci în categoria SGBD care respectau cam toate regulile lui Codd. E.F.Codd a realizat 13 reguli care stabilesc în ce măsură un SGBD este relațional. Acum, la versiunea Oracle 11g sistemul a devenit mai mult decât un SGBD relațional, a devenit o infrastructură pentru baze de date pe Internet, în arhitectură Grid Computing. Pe parcursul evoluției sistemului Oracle, au fost adăugate noi facilități rezultate atât din noul contextul informatic cât şi din cerințele utilizatorilor, date de dezvoltarea societății informaționale. În mod continuu, noi tehnologii informatice au fost implementate în sistem: lucrul cu baze de date distribuite, abordarea orientată obiect, facilități multimedia, gestionarea bazelor de date pentru Internet, noi limbaje de programare, afaceri electronice, inteligența afacerii, grid computing etc.

Principalele facilități oferite de sistemul Oracle

Pe parcursul evoluției sale, în special în ultimii zece ani, sistemului Oracle i s-au adăugat o serie de noi facilități. În acest fel, el a evoluat de la un SGBD relațional spre o infrastructură de baze de date pentru Internet. Câteva dintre aceste *facilități* sunt:

- a inițiat trecerea de la arhitectura client-server spre arhitectura NC (Network Computing). Versiunile actuale de Oracle pot funcționa în ambele arhitecturi;
- oferă o mare deschidere atât el ca sistem, cât şi pentru aplicațiile cu baze de date dezvoltate;
- pune mare accent pe tot felul de optimizări privind utilizarea resurselor de calcul (timpul şi spațiul);
- pune accent mai mare pe analiză – proiectare (modelare – funcționalitate) față de programare, înclinând balanța în favoarea primei activități;
- a implementat primul sistem de bază de date pentru Internet din lume şi apoi a dezvoltat acest concept cu o mulțime de servicii performante;
- a implementat pentru prima dată în lume tehnologia Grid Computing pentru baze de date şi continuă să o dezvolte;
- a devenit o platformă multiplă, de mare complexitate, cu portabilitate ridicată, care acceptă: orice calculator orice sistem de operare, orice date, orice aplicație, orice utilizator;
- oferă o mare diversitate de interfețe pentru dezvoltarea aplicațiilor cu baze de date: bazate pe modelare (Designer, Developer, Application Server), bazate pe componente (platforma Java), bazate pe HTML (navigatoare, editoare Web, XML), bazate pe servicii (Service Oriented Architecture – SOA), prin programare (fraze SQL, proceduri stocate – PL/SQL şi Java, obiecte standard CORBA, obiecte ODBC, obiecte JDBC) etc.;
- cuvintele de ordine în acest sistem sunt: optimizare (timp şi spațiu calculator) şi integrare (interfețe, aplicații, date, tehnologii etc.).

Oracle Grid Computing

Grid Computing (grilă de calculatoare) este o nouă tehnologie informatică ce presupune utilizarea coordonată a mai multor servere mici, care acționează împreună ca un singur sistem foarte

puternic. Se realizează un progres tehnologic de la Internet la *grid computing* datorat apariției unor componente tot mai puternice şi mai ieftine.
Oracle a introdus tehnologia *grid computing* odată cu dezvoltarea serverelor de baze de date în versiunea 10g (2003) şi apoi a dezvoltat-o în versiunea 11g (2007). Acest lucru a însemnat adăgarea unor *facilități noi* sistemului Oracle, câteva dintre acestea fiind prezentate în continuare.
Virtualizarea pe fiecare nivel. Oracle permite acum adăugarea sau scoaterea discurilor de stocare, cu păstrarea on-line a aplicațiilor. Performanțele serverelor sunt puse la dispoziția mai multor aplicații cu baze de date, într-o structură eficientă de tip cluster. Resursele serverelor se alocă în funcție de necesitățile organizației şi folosind tehnici speciale (load balancing): se urmăreşte încărcarea diferitelor echipamente din rețea şi maşina cu cele mai puține procese active va primi o nouă sarcină. Se realizează statistici privind utilizarea resurselor, care ajută administratorul să construiască dinamic strategii de repartizare a proceselor.
Platformă ieftină. Oracle poate rula pe platforme diverse, inclusiv servere şi alte echipamente de stocare a datelor la un cost redus, oferind aceleaşi funcționalități indiferent de platforma aleasă. Stocarea datelor se realizează folosind componenta denumită ASM (Automatic Storage Management) care permite utilizarea echipamentelor de cost redus. Administratorul alocă discurile de stocare componentei ASM care se ocupă de managementul ulterior, oferind performanțe optime, fără a necesita intervenția umană.
Scalarea. Pentru exploatare, se poate porni inițial de la o aplicație pe două niveluri (2tier), pe un grup de servere de cost redus, continuându-se apoi cu includerea unor noi aplicații şi a altor echipamente.
Managementul tip grid. Oracle a introdus un instrument special pentru managementul tip *grid*, denumit Oracle Grid Control, pe care administratoriul BD îl poate folosi pentru monitorizarea şi întreținerea întregii infrastructuri de baze de date, care include resurse eterogene, distribuite geografic. Resursele aplicațiilor cu BD nu mai sunt administrate individual, aşa cum se realiza în mod tradițional, ci grupat, prin utilizarea unui navigator Web. Astfel se poate realiza managementul resurselor de tip: servere de aplicații, servere de baze de date, servere de tip *firewall*, echipamente de stocare, echipamente de rețea.
Baze de date distribuite. Oracle a introdus conceptul de rulare a unei singure baze de date, integrate logic, pe multiple servere folosind tehnologia Real Application Clusters (RAC). Practic mai multe servere sunt folosite optim de către mai multe aplicații, reducându-se numărul de echipamente şi licențe necesare. Tehnologia de tip cluster nu trebuie să fie achiziționată de la un producător diferit, introducându-se componenta Oracle Portable Clusterware care permite folosirea facilităților de tip cluster pe orice tip de hardware.
Sisteme informatice integrate. Componenta Oracle Streams oferă funcționalități multiple pentru realizarea unei soluții de sistem informatic integrat, cu eliminarea redundanței datelor.
Securitatea. Oracle Enterprise User Security este o componentă de management centralizat a privilegiilor de acces a utilizatorilor. Practic, un utilizator este creat o singură dată, putând avea acces la multiple baze de date existente în arhitectură, conform drepturilor sale de acces.
Oracle Identity Management centralizează managementul autentificării şi autorizării utilizatorilor în cadrul unei soluții integrate, prin intermediul componentei denumite Oracle Internet Directory.

Componentele sistemului Oracle
Sistemul Oracle poate fi structurat funcțional, prin adaptare, conform arhitecturii pe componente a unui SGBD, care grupează componentele în trei niveluri: nucleul, interfețele de dezvoltare, instrumentele de întreținere.

Nucleul Oracle (Oracle Database) conține componentele care se regăsesc în orice kit de instalare: limbajele de programare (SQL, PL/SQL, Java, precompilatoarele), configurarea şi instalarea, instanțele de baze de date generate de sistem. Varianta de lucru cu adevărata bază de date distribuită este Oracle RAC (Real Application Clusters). Aceasta este o nouă generație de tehnologie de clustere, bazată pe o nouă arhitectură de baze de date denumită *îmbinare ascunsă* (Cache Fusion). Acest lucru înseamnă că la adăugarea unui calculator într-o rețea cu baze de date distribuite Oracle, clusterele se adaptează automat la noile resurse, fără să fie necesară redistribuirea datelor sau rescrierea aplicațiilor.
Interfețele pentru dezvoltarea aplicațiilor cu baze de date Oracle (Oracle Developer Suite) permit personalizarea aplicațiilor, oferă un mediu complet pentru dezvoltarea aplicațiilor tip afaceri electronice (e-business) şi tip Web. Câteva dintre produsele software incluse în acest pachet sunt: Oracle Forms, Oracle Reports, Oracle Designer, Oracle JDeveloper etc.
Instrumentele pentru întreținerea bazei de date Oracle sunt destinate, în principal, administratorului de baze de date, dar şi dezvoltatorilor. Oracle Enterprise Manager este pachetul integrat pentru administrarea bazei de date. El conține în acest sens o mulțime de instrumente software: monitorul Grid Control pentru supravegherea întregii activități a sistemului de baze de date, navigatoare pentru informare, editoare pentru actualizări, utilitare pentru întreținere şi acordare de drepturi de utilizatori etc. În Oracle activitatea de administrare a bazelor de date este automatizată în cea mai mare parte şi procesul continuă în aceeaşi direcție. Efectul este că activitatea de administrare a BD este ferită de erori şi accidente umane, iar costul acestei activități scade foarte mult. Tot în componenta de instrumente se poate încadra şi pachetul Oracle Application Server – OAS, care integrează serviciile de Internet şi permit crearea celor mai rapide aplicații Web din lume. Conține printre altele: Oracle E-business Suite destinat implementării aplicațiilor prefabricate cu baze de date pentru întreprinderi; Oracle Portal destinat construirii portalurilor organizaționale etc.

3.4. Limbajul PL/SQL - compendiu

PL/SQL (Procedural Language) este un *limbaj procedural*, propriu sistemului Oracle, care lucrează stil compilator şi care dă posibilitatea să se dezvolte structuri procedurale de program, suportând în acelaşi timp o parte dintre comenzile SQL. El a fost dezvoltat pe o structură inițială de program Pascal, ulterior fiindu-i adăugate numeroase extensii.
Câteva dintre elementele de limbaj din PL/SQL sunt prezentate, în sinteză, în continuare.
Instrucțiunea PL/SQL are drept terminator caracterul punct şi virgulă (;).
Expresiile în PL/SQL sunt formate din operatori şi operanzi. Tipul unei expresii este dat de tipul operatorilor utilizați.
Operatorii sunt de următoarele tipuri: *aritmetici* (+,-, *, /, **), *logici* (AND, OR, NOT), *de comparație* (=, !=, <, >, <=, >=), *alți operatori speciali* (LIKE; IN; BETWEEN etc.).
Operanzii pot fi variabile, constante, atribute (%TYPE, %FOUND etc.), funcții.
Unitatea de bază pentru structurarea unui program în limbajul PL/SQL este *blocul*.

Structura unui bloc este: DECLARE
Instrucțiuni declarative (neexecutabile)
BEGIN
Instrucțiuni executabile (proprii sau din SQL)
EXCEPTION
Rutine pentru tratarea erorilor (excepțiilor)
END;

Notă. Dintre secțiunile de mai sus, doar cea executabilă (între BEGIN şi END) este obligatorie, într-un bloc PL/SQL.
Comentariul se indică ca un text scris între /* şi */ (analog ca în SQL) sau după două caractere liniuță de unire (- -).
Instrucțiunile dintr-un bloc pot fi precedate de <<*etichetă*>>, care se poate folosi apoi pentru referire.
Limbajul PL/SQL acceptă şi *subprograme* care pot fi de următoarele tipuri: *proceduri* (PROCEDURE), *funcții* (FUNCTION), *pachete* (PACKAGE), *declanşatori* (TRIGGER).

Comenzi din SQL suportate în blocuri PL/SQL

INSERT, UPDATE, DELETE, SELECT, COMMIT, ROLLBACK, SAVEPOINT, LOCK, TABLE, SET TRANSACTION.

Instrucțiuni proprii limbajului PL/SQL

:= este instrucțiunea de atribuire.
BEGIN desemnează începutul părții executabile a uni bloc.
CLOSE închide un cursor explicit.
DECLARE marchează începutul unui bloc şi a părții neexecutabile a acestuia.
Pentru a declara variabile şi constante se folosesc clauzele: NUMBER, CHAR, VARCHAR, DATE, BOOLEAN, atributele (%nume).
Pentru a declara un cursor se foloseşte clauza: CURSOR.
Pentru a declara o excepție (tratarea erorilor) se foloseşte clauza: EXCEPTION.
END încheie o serie de instrucțiuni PL/SQL: bloc – END, structură repetitivă – END LOOP, structură alternativă – END IF.
EXCEPTION marchează începutul părții de tratare a erorilor dintr-un bloc.
EXIT forțează ieşirea necondiționată dintr-o structură nesecvențială de program.
FETCH accesează următoarea linie din mulțimea selectată de un cursor explicit.

GOTO salt necondiționat la o etichetă dintr-un bloc.
IF…END IF structura alternativă simplă de program.
LOOP… END LOOP structura repetitivă de program tip WHILE sau FOR
NULL este instrucțiunea care nu are nici un efect şi se foloseşte pentru structurarea şi lizibilitatea blocului.
OPEN deschide un cursor explicit.
RAISE opreşte execuția unui bloc şi transferă controlul unei secțiuni de tratarea excepțiilor.
SELECT…INTO instrucțiune de regăsire, care întoarce o linie dintr-o tabelă şi o plasează într-o variabilă de memorie indicată prin clauza INTO.

Câteva funcții de sistem

SQLCODE() numărul codului unei erori detectate.
SQLERRM() mesajul explicativ al unui cod de eroare specificat.
Majoritatea funcțiilor din SQL sunt suportate şi de PL/SQL.

3.5. Teste de autoevaluare

1. *Elemente ale unei BD Oracle sunt:*
   A. Procesul;
   B. Utilizatorul sistem;
   C. Indexul;
   D. Zona de lucru;
   E. Arborele;
2. *Mecanisme interne Oracle sunt:*
   A. Tabela;
   B. Tabelul;
   C. Cursorul;
   D. Tranzacția;
   E. Spațiul tabelă;
3. Tipurile de subprograme suportate de PL/SQL sunt:
   A. Procedurile;
   B. Pachetele;
   C. Funcțiile;
   D. Declanşatorii;
   E. Secvențialele;
4. *Care dintre următoarele convenții ce privesc parametrii formali şi reali unui subprogram sunt valabile în PL/SQL*
   A. Trebuie să fie acelaşi număr de parametri;
   B. Parametrii trebuie să fie în aceeaşi ordine;
   C. Numele parametrilor din cele două categorii nu pot fi diferite;
   D. Tipul pentru fiecare parametru corespunzător trebuie să fie diferit;
   E. Parametrii trebuie să fie în aceeaşi ordine;
5. *Într-un subprogram de tip declanşator se specifică:*
   A. Subprogramul apelator;
   B. Tipul declanşatorului, care poate fi posterior (AFTER) sau anterior (BEFOR);
   C. Evenimentul declanşator;
   D. Acțiunea de realizat;
   E. Parametrii reali ai declanşatorului

Răspunsuri la testele de autoevaluare

1. A, D
2. C, D
3. A, B, C, D
4. A, B
5. B,C,D

Bibliografia unității de învățare 3

1. M. Velicanu, I. Lungu s.a. – *Sisteme de baze de date – teorie şi practică*, ed. Petrion, Bucureşti, 2003.
2. J. Date – *An introduction to database systems*, Ed. Addison Wesley, 2004.
3. M. Velicanu – *Dicţionar explicativ al sistemelor de baze de date*, Ed. Economică, Bucureşti, 2005.
4. A. Bara, I. Botha, V. Diaconița, I. Lungu, A. Velicanu – *Baze de date. Limbajul PL/SQL*, ed. ASE, Bucureşti, 2009.