PROGRAMARE ORACLE 10g

SOLUȚII INFORMATICE PENTRU DEZVOLTAREA APLICAȚIILOR ECONOMICE UTILIZÂND PL/SQL și ORACLE DEVELOPER

Lect. univ. dr. IONEL IACOB

PROGRAMARE ORACLE 10g

SOLUȚII INFORMATICE PENTRU DEZVOLTAREA APLICAȚIILOR ECONOMICE UTILIZÂND PL/SQL și ORACLE DEVELOPER

EDITURA UNIVERSITARĂ
București, 2012

Redactor: Gheorghe Iovan
Tehnoredactor: Ionel Iacob
Coperta: Angelica Mălăescu

Editură recunoscută de Consiliul Național al Cercetării Științifice (C.N.C.S.)

Descrierea CIP a Bibliotecii Naționale a României
IACOB, IONEL
Programare Oracle 10g : soluții informatice pentru dezvoltarea aplicațiilor economice utilizând PL/SQL și Oracle Developer / Ionel Iacob. -
București : Editura Universitară, 2011
Bibliogr.
ISBN 978-606-591-313-4

004.4

DOI: (Digital Object Identifier): 10.5682/9786065913134

Editura Universitară
Director: Vasile Muscalu
B-dul. N. Bălcescu nr. 27-33, Sector 1, București
Tel.: 021 – 315.32.47 / 319.67.27
www.editurauniversitara.ro
e-mail: redactia@editurauniversitara.ro

Distribuție: tel.: 021-315.32.47 /319.67.27 / 0744 EDITOR / 07217 CARTE
comenzi@editurauniversitara.ro
O.P. 15, C.P. 35, București
www.editurauniversitara.ro

CUPRINS

PREFAȚĂ

Într-un mediu de afaceri modern, adaptabilitatea rapidă la cerințele complexe şi diversificate ale pieței, poate fi factorul definitoriu care plasează o companie înaintea altor zeci sau poate sute de societăți de profil, care ignoră competitivitatea, flexibilitatea şi eficiența. Dar, eforturile manageriale sau strategiile de marketing, fie ele şi de ultimă generație, pot deveni ineficiente sau zadarnice, dacă partea economică nu este susținută şi de partea informatică din cadrul societății. Eficiența folosirii informațiilor de afaceri depinde de strategia managerială şi de suportul hardware şi software implementat. Ambele cu aceleaşi priorități!

La peste 10 ani de secolul XXI, punctele de reper ale unei companii moderne se identifică cu: capacitatea de a se adapta şi organiza rapid şi eficient la cerințele pieței, flexibilitatea de a prelua şi prelucra un volum foarte mare şi cu caracter complex de date, analiza într-un timp foarte scurt a rezultatelor obținute în urma prelucrării datelor cu ajutorul aplicațiilor informatice moderne şi îmbunătățirea permanentă a platformelor hardware şi software.

Soluția rezolvării acestor deziderate este şi, cu certitudine...subiectivă, va fi: platforma ORACLE! ORACLE...ORACLE 10g, ORACLE 11g, ORACLE 11g WebLogic SUITE sau, pe viitor...alte versiuni „ORACLE n", unde {n/n>11, n<k, pt k<=100}g/i ?!

SGBD ORACLE 10g, păstrat în continuare şi în acest manual, ca versiune de lucru pentru exemplificarea fundamentelor şi realizarea aplicațiior utilizând platforma ORACLE, permite utilizatorilor dezvoltarea unor soluții informatice complexe, integrarea şi utilizarea eficientă a conceptelor de programare precum şi derularea şi administarea aplicațiilor economice.

Pentru a ordona volumul foarte mare de informații prelucrate (coroborat cu dorința „perpetuum mobile" a autorului de a îmbunătății şi diversifica cunoştințele studenților în domeniu) – cuprinsul lucrării a fost structurat în 3 secțiuni reprezentative, fiecare dintre acestea tratând o arie specifică: „SQL AVANSAT", „PROGRAMARE SQL", „ORACLE DEVELOPER-FORMS&REPORTS".

La rândul lor, fiecare dintre aceste părți include un număr distinct de capitole, repezentând în fapt „mini"aplicații informatice care, dezvoltate pe baza exemplelor implementate, pot constitui ulterior un punct viabil de plecare în construirea unor produse software complexe.

Fundamentul informatic al lucrării (din punct de vedere didactic) sau „cireaşa de pe tort " (într-o abordare specifică camerelor TV) – este reprezentat(ă) de Limbajul de Programare SQL (PL/SQL), ce se regăseşte exemplificat şi utilizat în peste 75% din conținutul informațional al materialului propus pentru studiu.

Aplicațiile propuse în această lucrare acoperă o paletă largă de cerințe, atât în ceea ce priveşte tematica economică abordată, cât mai ales a modului de realizare a lor, prin prisma conceptelor, noțiunilor şi exemplelor informatice implementate.

Noutatea acestei lucrări constă în integrarea, pe de o parte a conceptelor specifice „SQL AVANSAT” (concretizate în tipuri de funcții ORACLE ce permit o seroe de analize economice complexe) şi, pe de altă parte, realizarea unui capitol special al „Programării SQL”, dedicat „tehnicilor de programare, inclusiv cu problemele specifice metodelor „Backtracking” sau „Divide et impera”, şi exercițiilor de logică matematică.

Datorită simbiozei unitare dintre aspectele teoretice şi numeroasele exemple prezentate, lucrarea se adresează studenților şi tuturor cititorilor dornici de a explora tărâmul vast, dar şi ....periculos de atractiv numit: ORACLE. Acest „ocol de minimum 4 ani, 3 luni şi 2 zile” poate cauza pasionaților, „dependență” sau mai grav „virusul programatorului ORACLE”: acesta se infiltrează subtil în aplicațiile ORACLE şi la intervale de 12 ore fără pauză, virusează aplicația programatorului ORACLE şi, printr-un mesaj de avertizare din secțiunea „EXECEPTION” a unui bloc PL/SQL, lansează atacul: „ORA 30000 - LA MASĂĂĂĂĂĂĂĂĂĂĂ!

Acest sindrom se poate trata doar printr-un „consum suplimentar de programare ORACLE, în cantități moderate de până la 2-3 capitole pe zi de funcții, proceduri, triggere şi exemple, oricând în timpul zilei şi doar la nevoie noaptea, atunci când se susțin testele şi examene la SGBD ORACLE”.

În speranța captării interesului privind programarea şi a inițierii în domeniul Programării SQL, doresc să urez studenților succes la examenele universitare, să li se îndeplinească visul de a fi programatori de succes şi închei, prin povesti următoarea andecdotă cu tâlc: *„într-o grădină stăteau 3 pitici vorbitori; la un moment dat proprietarul se hotărăşte să-i vândă; vine un cumpărător şi întreabă cât costă primul pitic: „100” răspunde proprietarul; „Dar de ce aşa de mult?” întreabă curios cumpărătorul - „Păi e special: ştie programare Visual Basic”; „Dar al doilea?” - „200” răspunde proprietarul, şi el este un pitic special: ştie programare Visual C”. Dar ultimuul? întrebă descumpănit cumpărătorul. „1000” răspunde sec vânzătorul. „1000!” Dar de ce? – “Păi scrie pe spatele lui „ORACLE” răspunde plin el vânzătorul. „Păi şi dacă scrie, ce este? întrebă cumpărătorul; „Păi nu ştiu nici eu”, zise vânzătorul, dar îi tot aud pe ceilalți doi zicându-i „şefu”! Deci e mai scump!*

AUTORUL
APRILIE 2012

CAPITOLUL I

PROGRAMARE ORACLE UTILIZÂND FUNDAMENTE S.Q.L*PLUS AVANSAT PENTRU ANALIZĂ ECONOMICĂ

Pentru exemplificarea fundamentelor SQL*PLUS avansat care stau la baza interpretării rezultatelor economice obținute în urma prelucării datelor din sistemul economic, se propun funcțiile **„ROLLUP”**, **„CUBE”** şi **„GROUPING”**, dar şi clauza **„GROUP BY”** foarte des utilizată - care permit o analiză economică detaliată a rezultatelor obținute.

În acest context, s-a optat pentru proiectarea unei structuri mai puțin detaliate a unei baze de date aferente gestionării producției (în contextul exemplificării programării şi nu a proiectării), care să permită reliefarea funțiilor SQL avansat ale platformei Oracle în ceea ce priveşte analiza economică a datelor prelucrate.

I.1 PROIECTAREA APLICAȚIEI INFORMATICE PRIVIND EVIDENȚA PRODUSELOR FABRICATE ŞI IMPLEMENTAREA ÎN SQL*PLUS

PREZENTARE ACTIVITATE

Într-o societate economică, al cărui principal obiect de activitate este *„Producția de bunuri”*, se impune, la un moment dat, proiectarea şi implementarea unei baze de date, astfel încât să se asigure informatizarea procesului de gestiune a produselor fabricate, prin evidența „intrărilor” (materiile prime), evidența „ieşirilor” (produsele finite) şi a proceselor „intermediare” tehnologice.

Aplicația solicitată se va dezvolta utilizând standardul SQL, dar implementat în cadrul platformei Oracle (respectiv, SQL*PLUS) şi va asigura o interpretare de ansamblul a rezultatelor obținute în urma prelucării datelor preluate din sistem.

PROIECTAREA STRUCTURII BAZEI DE DATE

Baza de date privind evidența *„Producției de bunuri”* se denumeşte *„ProdData”* şi include *trei tabele*: tabela *„Materii prime”*, tabela *„Procese”* şi tabela *„Produse_Fabr”* – fiecare dintre aceste entități ale sistemului economic real identificând o parte a activității proiectate: *gestiunea materiilor prime pentru fabricație”*, *„prelucarea materiilor prime”* respectiv, *„evidența produselor finite rezultate”*.

Fiecare dintre cele trei tabele proiectate vor „surpinde” *caracteristicile definitorii* ale entităților identificate: *„Materii prime”, „Procese de fabricație”* şi *„Produse finite”*, astfel încât, într-o prezentare succintă a fundamentelor procesului economic privind *„evidența produselor finite rezultate în urma derulării procesului de fabricație”* - să poată fi dezvoltată o bază de date cât mai reprezentativă din punct de vedere informațional, dar şi o aplicație informatică capabilă să permită automatizarea procesului de gestiune.

Produsele finite rezultate în urma implementării proceselor tehnologice de fabricație privesc îndeosebi componentele auto, descrise în mod generic prin „Bloc motor”, „Panou bord” sau „Sisteme ABS”, şi sunt rezultatul implementării etapelor de realizare definite prin procesele „Prelucrare Tehnologică”, „Ansamblare tehnică” şi „Finisare operațională”.

Pentru proiectarea tabelelor bazei de date *„ProdData”* se vor identifica cu *„N”* câmpurile de tip *„numeric”*, cu *„C”*, câmpurile de tip *„caracter”* şi cu *„D”*, câmpurile de tip *„dată”*, urmând ca pentru fiecare dintre aceste tipuri de date să fie stabilită în structura SQL*PLUS identitatea corespunzătoare: *„Number”*, *„Varchar2*” şi respectiv, *„Date”*.

Totodată, atributul unei tabele identificat prin simbolul *„Pk”* (sau *„Primary Key”*) stabileşte *„cheia primară”* a tabelei (care perimite identificarea unică a tuplurilor), iar simbolul *„Fk”* (sau *„Foreign Key”*) stabileşte *„cheia externă”* a tabelei (cea care permite definirea asocierii dintr o tabelă de bază şi o tabelă de referită).

În tabela „Produse_Fabr”, atributul „cheie primară” este compus (fiind definit pe baza valorilor atributelor „CodP” şi „CodMP”), iar atributul „cheie externă” este definit prin „CodProc”; astfel se presupune ipoteza că „un produs este realizat pe baza mai multor categorii de materii prime”; procesele de fabricație sunt identificate generic, astfel încât pot să cuprindă o succesiune variată de faze şi operații de lucru.

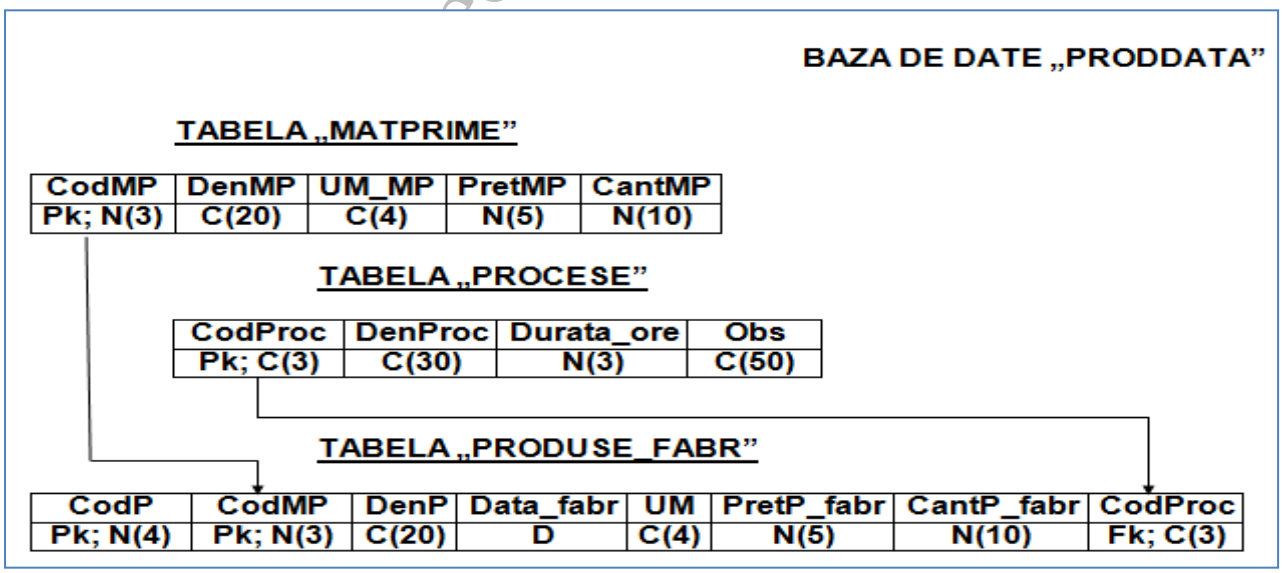

Figura 1: Proiectarea structurii bazei de date „ProdData” pentru evidența produselor finite rezultate în urma procesului de fabricație

PROGRAMAREA SQL*PLUS

Implementarea în mediul de programare SQL*PLUS a cerințelor de proiectare prezentate anterior, presupune: *crearea tabelelor aplicației, vizualizarea structurii tabelelor, inserarea de tupluri în cadrul tabelelor şi vizualizarea înregistrărilor adăugate.*

CREAREA TABELOR APLICAȚIEI

Pentru uşurință în programarea aplicației, se recomandă, preliminar creării tabelelor aplicației, utilizarea unui fişier *„.sql”,* denumit **„proddata.sql”**, în care se vor introduce *toate comenzile aferente procesului de creare tabele şi inserare tupluri.*

PROMPT CREARE TABELE

```
SQL> DROP TABLE Produse_Fabr;
     DROP TABLE MatPrime;
     DROP TABLE Procese;
```

PROMPT CREARE TABELA „MATERII PRIME”

```
SQL> CREATE TABLE MatPrime (
          CodMP     Number(3),
          DenMP     Varchar2(20),
          UM_MP     Varchar2(4),
          PretMP    Number(5),
          CantMP    Number(10),
               CONSTRAINT pk_codmp PRIMARY KEY(CodMP) );
SQL> DESCRIBE MatPrime;
```

PROMPT CREARE TABELA „PROCESE”

```
SQL> CREATE TABLE Procese (
          CodProc        Varchar2 (3),
          DenProc        varchar2(30),
          Durata_ore     Number(3),
          Obs            Varchar2(50),
               CONSTRAINT pk_codproc PRIMARY KEY(CodProc));
SQL> DESCRIBE PROCESE;
```

PROMPT CREARE TABELA „PRODUSE_FABR”

```
SQL> CREATE TABLE Produse_Fabr(
          CodP      Number(4),
          CodMP     Number(3),
          DenP      Varchar2(20),
```

```
        Data_fabr   Date default Sysdate,
        UM_P        Varchar2(4),
        PretP_fabr  Number(5),
        CantP_fabr Number(10),CodPROC varchar2(3),
              CONSTRAINT pk_codp_codmp PRIMARY KEY(CodP, CodMp),
              CONSTRAINT fk_codproc FOREIGN KEY (CodPROC)
                    REFERENCES Procese (CodPROC));
SQL> DESCRIBE Produse_Fabr;
```

INSERAREA TUPLURILOR ÎN TABELE

Adăugarea înregistrărilor în tabele se va realiza în funcție de informațiile prelevate din analiza sistemului, astfel încât datele introduse să reflecte cât mai exact realitatea economică.

PROMPT INSERARE TUPLURI IN TABELE

```
SQL> DELETE FROM Produse_Fabr;
SQL> DELETE FROM MatPrime;
SQL> DELETE FROM Procese;
```

PROMPT INSERARE TUPLURI IN TABELA „MATERII PRIME”

```
SQL> INSERT INTO MatPrime VALUES(1, 'Aliaj', 'Tone', 1200, 110);
       INSERT INTO MatPrime VALUES(2, 'Subansamble','Buc', 350, 500);
       INSERT INTO MatPrime VALUES(3, 'Prefabricate','Buc', 2000, 880);
SQL> SELECT * FROM MatPrime;
```

PROMPT INSERARE TUPLURI IN TABELA „PROCESE”

```
SQL> INSERT INTO Procese VALUES ('P1','Prelucrare tehnologica', 10,
            'Se prelucreaza materialul pentru ansamblare');
       INSERT INTO Procese VALUES ('P2','Ansamblare tehnica', 25,
            'Se ansambleaza produsul nefinisat');
       INSERT INTO Procese VALUES ('P3','Finisare operationala', 15,
            'Se finiseaza produsul finit');
SQL> SELECT * FROM Procese;
```

PROMPT INSERARE TUPLURI IN TABELA „PRODUSE_FABR”

```
SQL> INSERT INTO Produse_Fabr VALUES (101, 1, 'Bloc motor',
            TO_DATE('01-01-11','DD-MM-YY'), 'Tone', 1500, 200, 'P1');
       INSERT INTO Produse_Fabr VALUES (102, 1, 'Caroserie',
            TO_DATE('10-10-10','DD-MM-YY'), 'Buc', 1200, 410, 'P1');
       INSERT INTO Produse_Fabr VALUES (102, 1, 'Caroserie',
            TO_DATE('11-11-11','DD-MM-YY'), 'Buc', 11500, 100, 'P2');
```

```
SQL> INSERT INTO Produse_Fabr VALUES (103, 1, 'Panou bord',
        TO_DATE('10-11-11','DD-MM-YY'), 'Buc', 10000, 100, 'P2');
     INSERT INTO Produse_Fabr VALUES (103, 2, 'Panou bord',
        TO_DATE('17-11-11','DD-MM-YY'), 'Buc', 11500, 200, 'P3');
     INSERT INTO Produse_Fabr VALUES (103, 3, 'Panou bord',
        TO_DATE('12-11-11','DD-MM-YY'), 'Buc', 9000, 100, 'P3');
     INSERT INTO Produse_Fabr VALUES (104, 2, 'Sisteme ABS',
        TO_DATE('01-12-11','DD-MM-YY'), 'Buc', 11550, 100, 'P2');
     INSERT INTO Produse_Fabr VALUES (104, 3, 'Sisteme ABS',
         TO_DATE('08-12-11','DD-MM-YY'), 'Buc', 8500, 350, 'P3');
SQL> SELECT * FROM Produse_Fabr;
```

I.2 FUNCȚII ORACLE PENTRU ANALIZA ECONOMICĂ A REZULTATELOR

I.2.1 GRUPAREA LINIILOR TOTALIZATOARE - CLAUZA „GROUP BY”

Clauza „GROUP BY” este utilizată pentru *a diviza liniile unui tabel în grupuri.* Pentru a returna informația corespunzătoare fiecărui astfel de grup, se pot utiliza *funcțiile agregat*. *Funcțiile agregat* pot fi definite în *clauzele SELECT, ORDER BY şi HAVING*.

Server-ul Oracle aplică aceste funcții *fiecărui grup de linii şi returnează un singur rezultat pentru fiecare mulțime*.

Aplicații „GROUP BY”:

În tabela „Produse fabricate” se pot stabili mai multe „grupuri de tupluri” cu ajutorul cărora se pot obține rezultate semnificative şi se pot defini o serie de interpretări economice. Exemplificarea fundamentelor de bază privind gruparea datelor şi interpretarea rezultatelor, se va realiza utilizând facilitățile oferite de clauza „GROUP BY”.

Pentru o mai bună reprezentativitate a informațiilor prelevate, structurarea exemplelor propuse se va face alegând ca puncte de reper atributele cheii primare compuse din tabela „Produse_fabr”: „CodP” şi „CodMP”.

```
SQL> SELECT*FROM Produse_fabr;

SQL> PROMPT GRUPAREA PRODUSELOR IN FUNCTIE DE COD PRODUS
     PROMPT CODP ESTE PRIMUL FACTOR DE GRUPARE
SQL> SELECT Codp, SUM(Cantp_fabr) as "CANT_PROD_FABR", CodMP
     FROM Produse_Fabr
     WHERE CodP=103
     GROUP BY (CodP, CodMP);
```