Pachete

Pachete in *PL/SQL*

Pachetul *(package*) permite incapsularea intr-o unitate logica in baza de date a procedurilor, functiilor, cursoarelor, tipurilor, constantelor, variabilelor si exceptiilor.

Pachetele sunt unitati de program care sunt compilate, depanate si testate, sunt obiecte ale bazei de date care grupeaza tipuri, obiecte si subprograme *PL/SQL* avand o legatura logica intre ele.

De ce sunt importante?

Atunci cand este referentiat un pachet (cand este apelata pentru prima data o constructie a pachetului), intregul pachet este incarcat in *SGA*, zona globala a sistemului, si este pregatit pentru executie. Plasarea pachetului in *SGA* (zona globala sistem) reprezinta avantajul vitezei de executie, deoarece *server-ul* nu mai trebuie sa aduca informatia despre pachet de pe disc, aceasta fiind deja in memorie. Prin urmare, apeluri ulterioare ale unor constructii din acelasi pachet, nu solicita operatii *I/O* de pe disc. De aceea, ori de cate ori apare cazul unor proceduri si functii inrudite care trebuie sa fie executate impreuna, este convenabil ca acestea sa fie grupate intr-un pachet stocat. Este de subliniat ca in memorie exista o singura copie a unui pachet, pentru toti utilizatorii.

Spre deosebire de subprograme, pachetele nu pot:

- fi apelate,
- transmite parametri,
- fi incuibarite.

Un pachet are doua parti, fiecare fiind stocata separat in dictionarul datelor.

- Specificarea pachetului *(package specification)* - partea „vizibila”, adica interfata cu aplicatii sau cu alte unitati program. Se declara tipuri, constante, variabile, exceptii, cursoare si subprograme folositoare utilizatorului.
- Corpul pachetului *(package body)* - partea „acunsa”, mascata de restul aplicatiei, adica realizarea specificatiei. Corpul defineste cursoare si subprograme, implementand specificatia. Obiectele continute in corpul pachetului sunt fie private, fie publice.

Prin urmare, specificatia defineste interfata utilizatorului cu pachetul, iar corpul pachetului contine codul care implementeaza operatiile definite in

specificaţie. Crearea unui pachet se face in doua etape care presupun crearea specificaţiei pachetului si crearea corpului pachetului.

Un pachet poate cuprinde, fie doar partea de specificatie, fie specificatia si corpul pachetului. Daca contine doar specificatia, atunci evident pachetul contine doar definitii de tipuri si declaratii de date.

Corpul pachetului poate fi schimbat fara schimbarea specificatiei pachetului. Daca specificatia este schimbata, aceasta invalideaza automat corpul pachetului, deoarece corpul depinde de specificatie.

Specificatia si corpul pachetului sunt unitati compilate separat. Corpul poate fi compilat doar dupa ce specificatia a fost compilata cu succes.

Un pachet are urmatoarea forma generala:

```
CREATE PACKAGE nume_pachet {IS | AS} -- specificatia
    /* interfata utilizator, care contine: declaratii de tipuri si obiecte
       publice, specificatii de subprograme */
END [nume_pachet];
CREATE PACKAGE BODY nume_pachet {IS | AS} -- corpul
    /* implementarea, care contine: declaratii de obiecte si tipuri private,
       corpuri de subprograme specificate in partea de interfata */
[BEGIN]
    /* instructiuni de initializare, executate o singura data cand pachetul
       este invocat prima oara de catre sesiunea utilizatorului */
END [nume_pachet];
```

Specificaţia unui pachet

Specificatia unui pachet cuprinde declararea procedurilor, functiilor, constantelor, variabilelor si exceptiilor care pot fi accesibile utilizatorilor, adica declararea obiectelor de tip *PUBLIC* din pachet. Acestea pot fi utilizate in proceduri sau comenzi care nu apartin pachetului, dar care au privilegiul *EXECUTE* asupra acestuia.

Variabilele declarate in specificatia unui pachet sunt globale pachetului si sesiunii. Ele sunt initializate (implicit) prin valoarea *NULL*, evident daca nu este specificata explicit o alta valoare.

```
CREATE [OR REPLACE] PACKAGE [schema.]nume_pachet
[AUTHID {CURRENTUSER | DEFINER}] {IS | AS}
specificatie_PL/SQL;
```

Specificatie_PL/SQL poate include declaraţii de tipuri, variabile, cursoare, excepţii, funcţii, proceduri, pragma etc. În secţiunea declarativa, un obiect trebuie declarat inainte de a fi referit.
Optiunea *OR REPLACE* este specificata daca exista deja corpul pachetului. Clauzele *IS* si *AS* sunt echivalente, dar daca se foloseste *PROCEDURE BUILDER* este necesara optiunea *IS.*
Clauza *AUTHID* specifica faptul ca subprogramele pachetului se executa cu drepturile proprietarului (implicit) sau ale utilizatorului curent. De asemenea, aceasta clauza precizeaza daca referintele la obiecte sunt rezolvate in schema proprietarului subprogramului sau a utilizatorului curent.

Corpul unui pachet

Corpul unui pachet contine codul *PL/SQL* pentru obiectele declarate in specificatia acestuia si obiectele private pachetului. De asemenea, corpul poate include o sectiune declarativa in care sunt specificate definitii locale de tipuri, variabile, constante, proceduri si functii locale. Obiectele private sunt vizibile numai in interiorul corpului pachetului si pot fi accesate numai de catre functiile si procedurile din pachetul respectiv. Corpul pachetului este optional si nu este necesar sa fie creat daca specificatia pachetului nu contine declaratii de proceduri sau functii.

Este importanta ordinea in care subprogramele sunt definite in interiorul corpului pachetului. O variabila trebuie declarata inainte ca sa fie referita de alta variabila sau subprogram, iar un subprogram privat trebuie declarat sau definit inainte de a fi apelat de alte subprograme.

CREATE [OR REPLACE] PACKAGE BODY [schema.]nume_pachet
{IS | AS} corp_pachet;

Un pachet este instantiat cand este apelat prima data. Aceasta presupune ca pachetul este citit de pe disc in memorie si este executat codul compilat a subprogramului apelat. În acest moment, memoria este alocata tuturor variabilelor definite in pachet.

În multe cazuri este necesar sa se faca o initializare atunci cand pachetul este instantiat prima data intr-o sesiune. Aceasta se realizeaza prin adaugarea unei

secţiuni de iniţializare (opţionala) in corpul pachetului secţiune incadrata intre cuvintele cheie *BEGIN* si *END*. Secţiunea conţine un cod de iniţializare care este executat atunci cand pachetul este invocat pentru prima data.

Crearea pachetului face ca acesta sa fie disponibil pentru utilizatorul care l-a creat sau orice cont de utilizator caruia i s-a acordat privilegiul *EXECUTE*.

Referinţa la o declaraţie sau la un obiect specificat in pachet se face prefixand numele obiectului cu numele pachetului. În corpul pachetului, obiectele din specificaţie pot fi referite fara a specifica numele pachetului.

Procesul de creare a specificatei si corpului unui pachet urmeaza acelasi algoritm ca cel intalnit in crearea subprogramelor *PL/SQL* independente.

- sunt verificate erorile sintactice si semantice, iar modulul este depus in dicţionarul datelor;
- sunt verificate instrucţiunile *SQL* individuale, adica daca obiectele referite exista si daca utilizatorul le poate accesa;
- sunt comparate declaraţiile de subprograme din specificaţia pachetului cu cele din corpul pachetului (daca au acelasi numar si tip de parametri). Orice eroare detectata la compilarea specificaţiei sau a corpului pachetului este marcata in dictionarul datelor.

Dupa ce specificaţia si corpul pachetului sunt compilate, ele devin obiecte in schema curenta. În vizualizarea *USER OBJECTS* din dicţionarul datelor, vor fi doua noi linii:

QBJECTJTYPE OBJECTNAME
PACKAGE nume_pachet
PACKAGE BODY nume_pachet

Modificarea si suprimarea pachetelor

Modificarea unui pachet presupune de fapt recompilarea sa (pentru a putea modifica metoda de acces si planul de execuţie) si se realizeaza prin comanda:

ALTER PACKAGE [schema.]nume_pachet COMPILE [PACKAGE | BODY]

Schimbarea corpului pachetului nu cere recompilarea constructiilor dependente, in timp ce schimbari in specificaţia pachetului solicita recompilarea fiecarui subprogram stocat care referentiaza pachetul.

Daca se doreste modificarea sursei, utilizatorul poate recrea pachetul (cu opţiunea *REPLACE)* pentru a-l inlocui pe cel existent.

DROP PACKAGE [schema.]nume_pachet [PACKAGE | BODY]

Daca in cadrul comenzii apare opţiunea *BODY* este distrus doar corpul pachetului, in caz contrar sunt distruse atat specificaţia, cat si corpul pachetului. Daca pachetul este distrus, toate obiectele dependente de acesta devin invalide. Daca este distrus numai corpul, toate obiectele dependente de acesta raman valide. În schimb, nu pot fi apelate subprogramele declarate in specificatia pachetului, pana cand nu este recreat corpul pachetului.

Pentru ca un utilizator sa poata distruge un pachet trebuie ca fie pachetul sa apartina schemei utilizatorului, fie utilizatorul sa posede privilegiul de sistem *DROP ANY PROCEDURE*.

Una din posibilitatile interesante oferite de pachetele *PL/SQL* este aceea de a crea proceduri/functii *overload*. Procesul implica definirea unui numar de proceduri cu acelasi nume, dar care difera prin numarul si tipul parametrilor pe care le folosesc in fiecare instanta a procedurii implementata separat in corpul pachetului. Acest tip de programare este folositor cand este necesara o singura functie care sa execute aceeasi operatie pe obiecte de tipuri diferite (diferite tipuri de parametri de intrare). Cand este apelata o procedura *overload* sistemul decide pe baza tipului si numarului de parametri care instanta a procedurii va fi executata. Numai subprogramele locale sau apartinand unui pachet pot fi *overload*. Subprogramele *stand-alone* nu pot fi *overload*.

Utilizarea unui pachet se realizeaza in functie de mediul *(SQL* sau *PL/SQL)* care solicita un obiect din pachetul respectiv.

1) În *PL/SQL* se face prin referirea:

   nume_pachet.nume_componenta [(lista_de_argumente)];

2) În *SQL*Plus* se face prin comanda:

   EXECUTE nume_pachet.nume_componenta [(lista_de_argumente)]

Exemplu:

Sa se creeze un pachet ce include o procedura prin care se verifica daca o combinatie specificata dintre atributele *cod_artist* si *stil* este o combinatie care exista in tabelul *opera*.

```
CREATE PACKAGE verif_pachet IS PROCEDURE verifica
(p_idartist IN opera.cod_artist%TYPE, p_stil    IN
  opera.stil%TYPE);
END verif_pachet;
/
CREATE OR REPLACE PACKAGE BODY verif_pachet IS i
  NUMBER := 0;
CURSOR opera_cu IS
SELECT cod_artist, stil FROM opera;
TYPE opera_table_tip IS TABLE OF opera_cu%ROWTYPE
  INDEX BY BINARY INTEGER; art_stil opera_table_tip;
PROCEDURE verifica
(p_idartist IN opera.cod_artist%TYPE, p_stil    IN
  opera.stil%TYPE);
IS
BEGIN
FOR k IN art_stil.FIRST..art_stil.LAST LOOP IF
  p_idartist = art_stil(k).cod_artist AND p_stil =
  art_stil(k).stil THEN RETURN;
END IF;
END LOOP;
RAISE_APPLICATION_ERROR (-20777,'nu este buna
combinatia');
END verifica;
BEGIN
FOR ope_in IN opera_cu LOOP art_stil(i) := ope_in; i
  := i+1;
END LOOP;
END verif_pachet;
/
```

Utilizarea in *PL/SQL* a unui obiect *(verifica)* din pachet se face prin:

```
verif_pachet.verifica (7935, 'impresionism');
```

Utilizarea in *SQL*Plus* a unui obiect *(verifica)* din pachet se face prin: **EXECUTE**

```
verif pachet.verifica (7935, 'impresionism')
```

Observaţii:

- Un declanşator nu poate apela o procedura sau o funcţie ce conţine comenzile *COMMIT, ROLLBACK, SAVEPOINT.* Prin urmare, pentru flexibilitatea apelului (de catre declansatori) subprogramelor continute in pachete, trebuie verificat ca nici una din procedurile sau functiile pachetului nu contin aceste comenzi.
- Procedurile si functiile continute intr-un pachet pot fi referite din fisiere *iSQL*Plus*, din subprograme stocate *PL/SQL*, din aplicatii client (de exemplu, *Oracle Forms* sau *Power Builder)*, din declansatori (baza de date), din programe aplicatie scrise in limbaje de generatia a 3-a.
- Într-un pachet nu pot fi referite variabile gazda.
- Într-un pachet, mai exact in corpul acestuia, sunt permise declaratii *forward.*
- Functiile unui pachet pot fi utilizate (cu restrictii) in comenzi *SQL*.

Daca un subprogram dintr-un pachet este apelat de un subprogram *stand¬alone* trebuie remarcat ca:

- daca corpul pachetului se schimba, dar specificatia pachetului nu se schimba, atunci subprogramul care refera o constructie a pachetului ramane valid;
- daca specificatia pachetului se schimba, atunci subprogramul care refera o constructie a pachetului, precum si corpul pachetului sunt invalidate.

Daca un subprogram *stand-alone* referit de un pachet se schimba, atunci intregul corp al pachetului este invalidat, dar specificatia pachetului ramane valida.

`Pachete predefinite`

PL/SQL contine pachete predefinite utilizabile pentru dezvoltare de aplicatii si care sunt deja compilate in baza de date. Aceste pachete adauga noi functionalitati limbajului, protocoale de comunicatie, acces la fisierele sistemului etc. Apelarea unor proceduri din aceste pachete solicita prefixarea numelui procedurii cu numele pachetului.

Dintre cele mai importante pachete predefinite se remarca:

- *DBMSOUTPUT* (permite afisarea de informatii);
- *DBMS DDL* (furnizeaza accesul la anumite comenzi *LDD* care pot fi folosite in programe *PL/SQL*);

- *UTL FILE* (permite citirea din fişierele sistemului de operare, respectiv scrierea in astfel de fişiere);
- *UTLHTTP* (foloseste *HTTP* pentru accesarea din *PL/SQL* a datelor de pe *Internet);*
- *UTL TCP* (permite aplicatiilor *PL/SQL* sa comunice cu *server-e* externe utilizand protocolul *TCP/IP);*
- *DBMSJOB* (permite planificarea programelor *PL/SQL* pentru executie si executia acestora);
- *DBMSSQL* (acceseaza baza de date folosind *SQL* dinamic);
- *DBMSPIPE* (permite operatii de comunicare intre doua sau mai multe procese conectate la aceeasi instanta *Oracle);*
- *DBMSLOCK* (permite folosirea exclusiva sau partajata a unei resurse),
- *DBMSSNAPSHOT* (permite exploatarea cliseelor);
- *DBMSUTILITY* (ofera utilitati *DBA,* analizeaza obiectele unei scheme particulare, verifica daca *server*-ul lucreaza in mod paralel etc.);
- *DBMSLOB* (realizeaza accesul la date de tip *LOB,* permitand compararea datelor *LOB*, adaugarea de date la un *LOB*, copierea datelor dintr-un *LOB* in altul, stergerea unor portiuni din date *LOB*, deschiderea, inchiderea si regasirea de informatii din date *BFILE* etc).

DBMSSTANDARD este un pachet predefinit fundamental prin care se declara tipurile, exceptiile, subprogramele care sunt utilizabile automat in programele *PL/SQL*. Continutul pachetului este vizibil tuturor aplicatiilor. Pentru referirea componentelor sale nu este necesara prefixarea cu numele pachetului. De exemplu, utilizatorul poate folosi ori de cate ori are nevoie in aplicatia sa functia *ABS (x),* apartinand pachetului *DBMS STANDARD,* care reprezinta valoarea absoluta a numarului x, fara a prefixa numele functiei cu numele pachetului.

Pachetul *DBMS_OUTPUT*

DBMSOUTPUT permite afisarea de informatii atunci cand se executa un program *PL/SQL* (trimite mesagele din orice bloc *PL/SQL* intr-un buffer in BD).

DBMS OUTPUT lucreaza cu un *buffer* (continut in *SGA)* in care poate fi scrisa informatie utilizand procedurile *PUT, PUTLINE* si *NEW LINE.* Aceasta informatie poate fi regasita folosind procedurile *GETLINE* si *GETLINES.* Procedura *DISABLE* dezactiveaza toate apelurile la pachetul *DBMS OUTPUT* (cu exceptia procedurii *ENABLE)* si curata *bufer-ul* de orice informatie.

Inserarea in *buffer* a unui sfarsit de linie se face prin procedura *NEW LINE.*

Procedura *PUT* depune (scrie) informaţie in *buffer*, informaţie care este de tipul *NUMBER*, *VARCHAR2* sau *DATE*. *PUTLINE* are acelaşi efect ca procedura *PUT*, dar insereaza si un sfarsit de linie. Procedurile *PUT* si *PUT LINE* sunt *overload*, astfel incat informatia poate fi scrisa in format nativ *(VARCHAR2*, *NUMBER* sau *DATE*).

Procedura *GETLINE* regaseste o singura linie de informatie (de dimensiune maxima 255) din *buffer* (dar sub forma de sir de caractere). Procedura *GETLINES* regaseste mai multe linii *(nrlinii)* din *buffer* si le depune intr-un tablou *(numetab) PL/SQL* avand tipul sir de caractere. Valorile sunt plasate in tabel incepand cu linia zero. Specificatia este urmatoarea:

```
TYPE string255_table IS TABLE OF VARCHAR2(255)
INDEX BY BINARY_INTEGER;
PROCEDURE GET_LINES
(nume_tab OUT string255_table, nr_linii IN OUT
INTEGER);
```

Parametrul *nr linii* este si parametru de tip *OUT,* deoarece numarul liniilor solicitate poate sa nu coincida cu numarul de linii din *buffer.* De exemplu, pot fi solicitate 10 linii, iar in *buffer* sunt doar 6 linii. Atunci doar primele 6 linii din tabel sunt definite.

Dezactivarea referirilor la pachet se poate realiza prin procedura *DISABLE*, iar activarea referirilor se face cu ajutorul procedurii *ENABLE.*

Exemplu:

Urmatorul exemplu plaseaza in *buffer* (apeland de trei ori procedura *PUT)* toate informatiile intr-o singura linie.

```
DBMS_OUTPUT.PUT(:opera.valoare||:opera.cod_artist);
DBMS_OUTPUT.PUT(:opera.cod_opera);
DBMS_OUTPUT.PUT(:opera.cod_galerie);
```

Daca aceste trei comenzi sunt urmate de comanda

```
DBMS_OUTPUT.NEW_LINE;
```

atunci informatia respectiva va fi gasita printr-un singur apel *GET LINE.* Altfel, nu se va vedea nici un efect al acestor comenzi deoarece *PUT* plaseaza informatia in *buffer,* dar nu adauga sfarsit de linie.

Cand este utilizat pachetul *DBMSOUTPUT* pot sa apara erorile *buffer overflow* si *line length overflow.* Tratarea acestor erori se face apeland procedura

RAISE APPLICA TION ERROR din pachetul standard *DBMS STANDARD.*

Pachetul *DBMS_SQL*

Pachetul *DBMSSQL* permite folosirea dinamica a comenzilor *SQL* in proceduri stocate sau in blocuri anonime si analiza gramaticala a comenzilor *LDD.* Aceste comenzi nu sunt incorporate in programul sursa, ci sunt depuse in siruri de caractere. O comanda *SQL* dinamica este o instructiune *SQL* care contine variabile ce se pot schimba in timpul executiei. De exemplu, pot fi utilizate instructiuni *SQL* dinamice pentru:

- a crea o procedura care opereaza asupra unui tabel al carui nume nu este cunoscut decat in momentul executiei;
- a scrie si executa o comanda LDD;
- a scrie si executa o comanda *GRANT, ALTER SESSION* etc.

În *PL/SQL* aceste comenzi nu pot fi executate static. Pachetul *DBMS SQL* permite, de exemplu, ca intr-o procedura stocata sa folosesti comanda *DROP TABLE*. Evident, folosirea acestui pachet pentru a executa comenzi *LDD* poate genera interblocari. De exemplu, pachetul este utilizat pentru a sterge o procedura care insa este utilizata.

SQL dinamic suporta toate tipurile de date *SQL*, dar nu suporta cele specifice *PL/SQL*. Unica exceptie o constituie faptul ca o inregistrare *PL/SQL* poate sa apara in clauza *INTO* a comenzii *EXECUTE IMMEDIATE*.

Orice comanda *SQL* trebuie sa treaca prin niste etape, cu observatia ca anumite etape pot fi evitate. Etapele presupun: analizarea gramaticala a comenzii, adica verificarea sintactica a comenzii, validarea acesteia, asigurarea ca toate referintele la obiecte sunt corecte si asigurarea ca exista privilegiile referitoare la acele obiecte *(parse*); obtinerea de valori pentru variabilele de legatura din comanda *SQL (binding variables);* executarea comenzii *(execute);* selectarea randurilor rezultatului si incarcarea acestor randuri *(fetch*).

Dintre subprogramele pachetului *DBMS SQL*, care permit implementarea etapelor amintite anterior se remarca:

- *OPEN CURSOR* (deschide un nou cursor, adica se stabileste o zona de memorie in care este procesata comanda *SQL*);
- *PARSE* (stabileste validitatea comenzii *SQL*, adica se verifica sintaxa instructiunii si se asociaza cursorului deschis);
- *BIND VARIABLE* (leaga valoarea data de variabila corespunzatoare din comanda *SQL* analizata)
- *EXECUTE* (executa comanda *SQL* si returneaza numarul de linii procesate);

- *FETCH ROWS* (regaseste o linie pentru un cursor specificat, iar pentru mai multe linii foloseste un *LOOP);*
- *CLOSECURSOR* (inchide cursorul specificat).

Sa se construiasca o procedura care foloseste *SQL* dinamic pentru a sterge liniile unui tabel specificat *(num tab).* Subprogramul furnizeaza ca rezultat numarul liniilor sterse *(nrlin).*

```
CREATE OR REPLACE PROCEDURE sterge_linii (num_tab IN
   VARCHAR2, nr_lin OUT NUMBER)
AS
nume_cursor INTEGER;
BEGIN
nume_cursor := DBMS_SQL.OPEN_CURSOR;
DBMS_SQL.PARSE (nume_cursor, 'DELETE FROM' ||
num_tab, DBMS_SQL.V7); nr_lin := DBMS_SQL.EXECUTE
   (nume_cursor); DBMS_SQL.CLOSE_CURSOR (nume_cursor);
END;
```

Argumentul *DBMSSQL.V7* reprezinta modul (versiunea 7) in care *Oracle* trateaza comenzile *SQL.* Stergerea efectiva a liniilor tabelului *opera* se realizeaza:

```
VARIABLE linii_sterse NUMBER
EXECUTE sterge_linii ('opera', :linii_sterse)
PRINT linii_sterse
```

Pentru a executa o instructiune *SQL* dinamic poate fi utilizata si comanda *EXECUTE IMMEDIATE.* Comanda contine o clauza optionala *INTO* care este utilizabila pentru interogari ce returneaza o singura linie. Pentru o cerere care returneaza mai multe linii trebuie folosite comenzile *OPEN FOR*, *FETCH*, *CLOSE.*

```
CREATE OR REPLACE PROCEDURE sterge_linii (num_tab IN
   VARCHAR2, nr_lin OUT NUMBER)
IS
BEGIN
EXECUTE IMMEDIATE 'DELETE FROM' || num_tab; nr_lin :=
   SQL%ROWCOUNT;
END;
```

Procedura se poate apela printr-o secventa identica cu cea prezentata anterior.

Pachetul *DBMS_DDL*

Pachetul *DBMSDDL* furnizeaza accesul la anumite comenzi *LDD* care pot fi folosite in subprograme *PL/SQL* stocate. În felul acesta, pot fi accesate (in *PL/SQL*) comenzile *ALTER* sau *ANALYZE*.

Pachetul include procedura *ALTER_COMPILE* care permite recompilarea programului modificat (procedura, funcţie, declanşator, pachet, corp pachet).

ALTER_COMPILE *(tipobiect, numeschema, nume obiect);*

Procedura este echivalenta cu instructiunea *SQL:*

ALTER PROCEDURE | FUNCTION | PACKAGE [schema.] nume COMPILE [BODY]

Cu ajutorul procedurii *ANALYZEOBJECT* poate fi analizat un obiect de tip *table*, *cluster* sau *index.* Procedura furnizeaza statistici referitoare la obiectele amintite. De exemplu, se pot obtine numarul liniilor unui tabel, numarul de blocuri ale unui tabel, lungimea medie a unei linii, numarul valorilor distincte ale unei col., numarul elementelor *null* dintr-o coloana, distributia datelor (histograma) etc.

ANALYZE_OBJECT (tip_obiect, nume_schema, nume_obiect, metoda, numar_linii_estimate, procent, optiune_metoda, nume_partitie);

Metodele care pot fi utilizate sunt *COMPUTE, ESTIMATE* sau *DELETE*. Prin aceste metode se cuantifica distributia datelor si caracteristicile de stocare. *DELETE* determina stergerea statisticilor (depuse in *DD)* referitoare la obiectul analizat. *COMPUTE* calculeaza statistici referitoare la un obiect analizat si le depune in *DD,* iar *ESTIMATE* estimeaza statistici. Statisticile calculate sau estimate sunt utilizate pentru optimizarea planului de executie a comenzilor *SQL* care acceseaza obiectele analizate.

Procedura este echivalenta cu instructiunea:

ANALYZE TABLE | CLUSTER | INDEX [nume_schema] nume_obiect [metoda] STATISTICS [SAMPLE n] [ROWS | PERCENT]]

Daca *nume schema* este *null,* atunci se presupune ca este vorba de schema curenta. Daca *tip obiect* este diferit de *table, index* sau *cluster,* se declanseaza eroarea *ORA - 20001*. Parametrul *procent* reprezinta procentajul liniilor de estimat si este ignorat daca este specificat numarul liniilor de estimat *(numarliniiestimate).* Implicit, ultimele patru argumente ale procedurii iau valoarea *null*.

Argumentul *optiunemetoda* poate avea forma:

[FOR TABLE] [FOR ALL INDEXES] [FOR ALL [INDEXED] COLUMNS] [SIZE n]

Pentru metoda *ESTIMATE* trebuie sa fie prezenta una dintre aceste optiuni.

Exemplu:

Utilizand pachetul *DBMS DDL* si metoda *COMPUTE,* sa se creeze o procedura care analizeaza un obiect furnizat ca parametru acestei proceduri.

```
CREATE OR REPLACE PROCEDURE analiza (p_obiect_tip IN
   VARCHAR2, p_obiect_nume IN VARCHAR2);
IS
BEGIN
DBMS_DDL.ANALYZE_OBJECT(p_obiect_tip, USER,
                        UPPER(p_obiect_nume), 'COMPUTE');
END;
/
```

Procedura se testeaza (relativ la tabelul *opera)* in felul urmator:

```
EXECUTE analiza ('TABLE', 'opera')
SELECT LAST_ANALYZED
FROM USER_TABLES
WHERE TABLE_NAME = 'opera';
```

Pachetul *DBMS_JOB*

Pachetul *DBMSJOB* este utilizat pentru planificarea programelor *PL/SQL* in vederea executiei. Cu ajutorul acestui pachet se pot executa programe *PL/SQL* la momente determinate de timp, se pot sterge sau suspenda programe din lista de planificari pentru executie, se pot rula programe de intretinere in timpul perioadelor de utilizare scazuta etc.

Dintre subprogramele acestui pachet se remarca:

- *SUBMIT* - adauga un nou *job* in coada de asteptare a job-urilor;
- *REMOVE* - sterge un *job* specificat din coada de asteptare a job-urilor;
- *RUN* - executa imediat un *job* specificat;
- *BROKEN* - dezactiveaza executia unui *job* care este marcat ca *broken* (implicit, orice *job* este *not broken*, iar un *job* marcat *broken* nu se executa);
- *CHANGE* - modifica argumentele *WHAT, NEXTDATE, INTERVAL;*
- *WHAT* - furnizeaza descrierea unui *job* specificat;
- *NEXTDATE* - da momentul urmatoarei executii a unui *job;*
- *INTERVAL* - furnizeaza intervalul intre diferite executii ale unui *job*.

Fiecare dintre subprogramele pachetului are argumente specifice. De exemplu, procedura *DBMSJOB.SUBMIT* are ca argumente:

- *JOB* - de tip *OUT*, un identificator pentru *job (BINARYINTEGER);*

- *WHAT* - de tip *IN*, codul *PL/SQL* care va fi executat ca un *job (VARCHAR2);*
- *NEXT DATE* - de tip *IN*, data urmatoarei execuţii a job-ului (implicit este *SYSDATE);*
- *INTERVAL* - de tip *IN*, functie care furnizeaza intervalul dintre executiile job-ului *(VARCHAR2*, implicit este *null);*
- *NO PARSE* - de tip *IN*, variabila logica care indica daca job-ul trebuie analizat gramatical *(BOOLEAN*, implicit este *FALSE).*

Daca unul dintre parametri *WHAT, INTERVAL* sau *NEXT DATE* are valoarea *null*, atunci este folosita ultima valoare asignata acestora.

Exemplu:

Sa se utilizeze pachetul *DBMS JOB* pentru a plasa pentru executie in coada de asteptare a job-urilor, procedura *verifica* din pachetul *verif_pachet.*

```
VARIABLE num_job NUMBER BEGIN
DBMS_JOB.SUBMIT(
job => :num_job,
what    =>
'verif_pachet.verifica(8973,''impresionism'');'
next_date => TRUNC(SYSDATE+1), interval =>
'TRUNC(SYSDATE+1)');
COMMIT;
END;
/
PRINT num_job
```

Vizualizarea *DBA JOBS* din dictionarul datelor furnizeaza informatii referitoare la starea *job*-urilor din coada de asteptare, iar vizualizarea *DBA JOBS RUNNING* contine informatii despre job-urile care sunt in curs de executie. Vizualizarile pot fi consultate doar de utilizatorii care au privilegiul *SYS.DBAJOBS.*

Exemplu:

```
SELECT JOB, LOG_USER, NEXT_DATE, BROKEN, WHAT
FROM DBA JOBS;
```

Pachetul *UTL_FILE*

Pachetul *UTLFILE* permite programului *PL/SQL* citirea din fişierele sistemului de operare, respectiv scrierea in aceste fişiere. El este utilizat pentru exploatarea fisierelor text.

Folosind componentele acestui pachet (functiile *FOPEN* si *ISOPEN;* procedurile *GETLINE, PUT, PUTLINE, PUTF, NEW LINE, FCLOSE, FCLOSEALL, FFLUSH)* se pot deschide fisiere, obtine text din fisiere, scrie text in fisiere, inchide fisiere.

Pachetul proceseaza fisierele intr-o maniera clasica:

- verifica daca fisierul este deschis (functia *ISOPEN);*
- daca fisierul nu este deschis, il deschide si returneaza un *handler* de fisier (de tip *UTLFILE.FILETYPE)* care va fi utilizat in urmatoarele operatii *I/O* (functia *FOPEN);*
- proceseaza fisierul (citire/scriere din/in fisier);
- inchide fisierul (procedura *FCLOSE* sau *FCLOSEALL).*

Functia *IS OPEN* verifica daca un fisier este deschis. Are antetul:

```
FUNCTION    IS_OPEN (handler_fisier IN FILE_TYPE)
  RETURN BOOLEAN;
```

Functia *FOPEN* deschide un fisier si returneaza un handler care va fi utilizat in urmatoarele operatii I/O. Parametrul *openmode* este un string care specifica modul cum a fost deschis fisierul.

```
FUNCTION    FOPEN    (locatia    IN    VARCHAR2,
                nume_fisier    IN    VARCHAR2,
                open~mode    IN    VARCHAR2)
  RETURN UTL_FILE.FILE_TYPE;
```

Prin procedura *GET LINE,* pachetul *UTL FILE* citeste o linie de text din fisierul deschis pentru citire si o plaseaza intr-un *buffer* de tip sir de caractere, iar prin procedurile *PUT* si *PUT LINE* scrie un text din *buffer* in fisierul deschis pentru scriere sau adaugare.

Utilizarea componentelor acestui pachet pentru procesarea fisierelor sistemului de operare poate declansa exceptii, dintre care remarcam:

- *INVALID PATH* - numele sau locatia fisierului sunt invalide;
- *INVALID MODE* - parametrul *OPEN MODE* (prin care se specifica daca fisierul este deschis pentru citire, scriere, adaugare) este invalid;
- *INVALID FILEHANDLE* - *handler-ul* de fisier obtinut in urma deschiderii este invalid;

- *INVALIDOPERATION*- operaţie invalida asupra fişierului;
- *READERROR* - o eroare a sistemului de operare a aparut in timpul operaţiei de citire;
- *WRITE ERROR* - o eroare a sistemului de operare a aparut in timpul operatiei de scriere;
- *INTERNALERROR* - o eroare nespecificata a aparut in *PL/SQL*.

Exceptiile trebuie prefixate cu numele pachetului. Evident, pot aparea si erorile *NO DATA FOUND* si *VALUEERROR*.

Pachetele *DBMS_PIPE si DBMS_ALERT*

Pachetul *DBMSPIPE* permite operatii de comunicare intre doua sau mai multe sesiuni conectate la aceeasi baza de date. De exemplu, pachetul poate fi utilizat pentru comunicarea dintre o procedura stocata si un program *Pro*C*. Comunicarea se face prin conducte *(pipe)*. O conducta este o zona de memorie utilizata de un proces pentru a transmite informatie altui proces. Informatia trimisa prin conducta este depusa intr-un *buffer* din *SGA*. Toate informatiile din conducta sunt pierdute atunci cand instanta este inchisa.

Conductele sunt asincrone, ele operand independent de tranzactii. Daca un anumit mesaj a fost transmis, nu exista nici o posibilitate de oprire a acestuia, chiar daca sesiunea care a trimis mesajul este derulata inapoi *(rollback)*.

Pachetul *DBMS PIPE* este utilizat pentru a trimite mesaje in conducta *(DBMSPIPE.SENDMESSAGE)*, mesaje ce constau din date de tip *VARCHAR2*, *NUMBER*, *DATE*, *RAW* sau *ROWID*. Tipurile obiect definite de utilizator si colectiile nu sunt acceptate de acest pachet.

De asemenea, pachetul poate realiza primirea de mesaje din conducta in *buffer-ul* local *(DBMS_PIPE.RECEIVE_MESSAGE)*, accesarea urmatorului articol din *buffer (DBMSPIPE.UNPACKMESSAGE)*, crearea unei noi conducte *(DBMS PIPE. CREATEPIPE)* etc.

DBMSALERT este similar pachetului *DBMS PIPE*, fiind utilizat tot pentru comunicarea dintre sesiuni conectate la aceeasi baza de date. Exista totusi cateva deosebiri esentiale.

- *DBMS ALERT* asigura o comunicare sincrona.
- Un mesaj trimis prin *DBMS PIPE* este primit de un singur destinatar (cititor) chiar daca exista mai multi pe conducta, pe cand cel trimis prin *DBMS ALERT* poate fi primit de mai multi cititori simultan.
- Daca doua mesaje sunt trimise printr-o conducta (inainte ca ele sa fie citite), ambele vor fi primite de destinatar prin *DBMS PIPE*. În cazul pachetului *DBMS ALERT,* doar cel de al 2-lea mesaj va fi primit.

Pachete predefinite furnizate de Oracle

Oracle furnizează o varietate de pachete predefinite care simplifica administrarea bazei de date si ofera noi functionalitati legate de noile caracteristici ale sistemului. Dintre pachetele introduse se remarca:

- *DBMS REDEFINITION* - permite reorganizarea *online* a tabelelor;
- *DBMS LIBCACHE* - permite extragerea de comenzi *SQL* si *PL/SQL* dintr-o instanta distanta intr-una una locala (vor fi compilate local, dar nu executate);
- *DBMSLOGMNRCDCPUBLISH* - realizeaza captarea schimbarilor din tabelele bazei de date (identifica datele adaugate, modificate sau sterse si editeaza aceste informatii intr-o forma utilizabila in aplicatii);
- *DBMSLOGMNRCDCSUBSCRIBE* - face posibila vizualizarea si interogarea schimbarilor din datele care au fost captate cu pachetul *DBMSLOGMNRCDCPUBLISH*;
- *DBMS METADATA* - furnizeaza informatii despre obiectele bazei de date;
- *DBMSRESUMABLE* - permite setarea limitelor de spatiu si timp pentru o operatie specificata, operatia fiind suspendata daca sunt depasite aceste limite;
- *DBMSXMLQUERY, DBMSXMLSAVE, DBMS XMLGEN* - permit prelucrarea si conversia datelor *XML (XMLGEN* converteste rezultatul unei cereri *SQL* in format *XML, XMLQUERY* este similara lui *XMLGEN,* doar ca este scrisa in C, iar *XMLSA VE* face conversia din format *XML* in date ale bazei);
- *UTLINADDR* - returneaza numele unei gazde locale sau distante a carei adresa *IP* este cunoscuta si reciproc, returneaza adresa *IP* a unei gazde careia i se cunoaste numele (de exemplu, *www.oracle.com*);
- *DBMSAQELM* - furnizeaza proceduri si functii pentru gestionarea configuratiei cozilor de mesaje asincrone prin *e-mail* si *HTTP*;
- *DBMS FGA* - asigura intretinerea unor functii de securitate;
- *DBMS FLASHBACK* - permite trecerea la o versiune a bazei de date corespunzatoare unei unitati de timp specificate sau unui *SCN (system change number)* dat, in felul acesta putand fi recuperate linii sterse sau mesaje *e-mail* distruse;
- *DBMS TRANSFORM* - furnizeaza subprograme ce permit transformarea unui obiect (expresie *SQL* sau functie *PL/SQL)* de un anumit tip (sursa) intr-un obiect avand un tip (destinatie) specificat;