Baze de date, IESC – Îndrumar de Laborator

Capitolul 2 – Instrucţiuni SQL de bază

1) SQL – noţiuni introductive

Apariţia limbajului SQL este legată de lansarea proiectului System/R de către IBM în 1974 [FO]. În acelaşi an, Chamberlin şi Boyce au prezentat un limbaj de interogare a bazelor de date numit de ei SEQUEL (Structured English as QUery Language). Prin anii 1980 Chamberlain a schimbat denumirea limbajului SEQUEL în SQL (Structured Query Language).

Prima standardizare a SQL a fost SQL-86 (standard ANSI X3.135-1986), urmată în 1987 şi de ISO (ISO 9075-1987 Database Language SQL). Au urmat alte câteva standardizări, dar cel mai important şi cunoscut rămâne SQL-92 sau SQL2, adoptat de toţi producătorii de SGBD-uri ca limbaj relaţional de bază.

Printre numeroasele avantaje ale SQL-ului menţionăm [FO]: este un limbaj de nivel înalt, cu sintaxă apropiată de limba engleză, independent de producător, portabil, standardizat, este un limbaj relaţional complet, permite modificarea structurii bazei de date în timp ce există utilizatori conectaţi etc.

Cele mai importante comenzi SQL sunt:

a) *Comenzi pentru manipularea datelor (tip DML):*
   SELECT - extragerea de date din tabele
   INSERT - inserarea de linii din tabele
   DELETE - ştergerea de linii din tabele
   UPDATE - actualizarea valorii unor atribute

b) *Comenzi de definire şi descriere a datelor (tip DDL):*
   CREATE TABLE - crearea structurii unei tabele
   DROP TABLE - ştergerea de tabele
   ALTER TABLE - modificarea structurii unei tabele
   CREATE VIEW - crearea unei vederi (tabelă virtuală)
   DROP VIEW - ştergerea tabelei virtuale

c) *Comenzi pentru securitate şi controlul accesului la baza de date:*
   CREATE USER - crearea unui nou cont de utilizator în SGBD
   ALTER USER - modificarea proprietăţilor unui cont
   DROP USER - ştergerea unui cont
   GRANT - acordarea de drepturi
   REVOKE - revocarea unor drepturi

d) *Comenzi pentru controlul tranzacţiilor:*
   COMMIT - executarea tranzacţiei curente
   ROLLBACK - un fel de „undo” pentru tranzacţia curentă

2) Tipuri de date în Oracle

Noţiunea de tip de date este echivalentă cu cea întâlnită în limbajele de programare clasice (C, C++, Pascal, Java etc.), iar în cazul nostru va reprezenta tipul atributelor relaţiilor (tipul coloanelor din tabele), ale argumentelor funcţiilor şi procedurilor etc.

a) Tipuri de date alfanumerice

Există două tipuri principale: CHAR şi VARCHAR2.

Tipul CHAR poate stoca un şir de caractere de dimensiune fixă declarată, iar daca şirul atribuit este mai mic, spaţiul rămas se completează cu spaţii albe.

Tipul VARCHAR2 permite o utilizare mai eficientă a memoriei, ocupând doar atâţia octeţi cat este necesar (nu mai completează cu spaţii albe).

Exemplu de utilizare: dacă dorim să stocăm prenumele unei persoane pe maxim 25 de caractere putem declara coloana prenume ca VARCHAR2(25).

b) Tipuri de date numerice

Principalele tipuri numerice în Oracle sunt: NUMBER, BINARY_FLOAT şi BINARY_DOUBLE.

Tipul NUMBER este cel mai portabil şi este recomandat aproape în toate situaţiile.

Sintaxa tipului este: NUMBER(p,s)

unde: p este numărul total de cifre, iar s este numărul de zecimale.

Dacă dorim doar numere întregi, sintaxa este: NUMBER(p), sau echivalent: NUMBER(p,0).

Exemplu:

Numărul introdus	*Tipul specificat*	*Numărul stocat*
24351.58	NUMBER	24351.58
24351.58	NUMBER(7,2)	24351.58
24351.58	NUMBER(7,1)	24351.6
24351.58	NUMBER(7)	24352
24351.58	NUMBER(5,2)	depășește precizia
24351.58	NUMBER(4)	depășește precizia

Tipurile BINARY_FLOAT (32 biți) şi BINARY_DOUBLE (64 biți) nu sunt utilizate decât pentru numere reale, dacă dorim o stocare binară, care ocupă mai puţin spaţiu de memorie (binary_float ocupă 5 octeți, iar binary_double nouă).

c) Tipurile timp şi dată calendaristică

Oracle suportă următoarele tipuri de date din această categorie: DATE, TIMESTAMP, TIMESTAMP WITH TIME ZONE şi TIMESTAMP WITH LOCAL TIME ZONE

Tipul DATE conţine şapte valori numerice pentru: secol, an, lună, zi, oră, minut şi secundă. Anul este stocat cu patru cifre, de exemplu 2009 şi nu 09.

Se pot face operaţii aritmetice cu tipul DATE deoarece conţine valori numerice [ODoc].

Data este afișată după un format ales. Formatul prestabilit este: DD-MON-RR, care va afişa data sub forma: 01-JAN-09. Formatul RR are următoarea regulă [POP]: dacă anul transmis este între 00 şi 49, de exemplu 09 ca mai sus, şi anul curent are ultimele 2 cifre tot între 00 şi 49, atunci anul real stocat este 2009. Dacă anul transmis este între 50 şi 99, de exemplu dacă transmitem 89 şi anul curent este între 00 şi 49, atunci anul stocat real va fi 1989 (secolul trecut).

Ora în Oracle este stocată în format 24 de ore: HH:MI:SS.

Tipul de date TIMESTAMP este o extensie a tipului DATE, şi stochează şi fracţiuni de secundă.

Tipurile TIMESTAMP WITH TIME ZONE şi TIMESTAMP WITH LOCAL TIME ZONE ţin seama de regiunea geografică.

3) Crearea tabelelor, inserarea de linii, modificarea tabelelor

a) Crearea unei tabele – comanda CREATE TABLE

Sintaxa comenzii de creare a tabelelor este următoarea [FI]:

CREATE TABLE nume_tabel (

Col*1* domeniu*1* [constrângeri_coloană],
Col*2* domeniu*2* [constrângeri_coloană],
……
Col*n* domeniu*n* [constrângeri_coloană],
[constrângeri_tabel]);

Creaţi folosind Oracle SQLDeveoper următoarea tabelă:

AGENDA

ID	Nume	Prenume	Nr_tel

Instrucţiunea de creare a tabelei agenda este:

create table agenda(
id number(3),
nume varchar2(20),
prenume varchar2(20),
nr_tel char(14));

După executarea instrucţiunii de mai sus, va apărea tabela agenda vidă, fără nici o înregistrare.

Observaţie: orice comandă SQL se termină cu caracterul „punct şi virgulă” (;).

b) Inserarea de linii în tabela creată – comanda INSERT

Inserarea de linii într-o tabelă se face prin comanda INSERT cu sintaxa [FI]:

INSERT INTO nume_tabel (col*1*, col*2*, …, col*n*) VALUES (val*1*, val*2*, …, val*n*);

Dacă introducem valori în toate coloanele tabelei, atunci lista coloanelor poate lipsi, şi sintaxa se simplifică:

INSERT INTO nume_tabel VALUES (val*1*, val*2*, …, val*n*);

În acest caz ordinea valorilor trebuie, bineînţeles, să fie ordinea coloanelor din tabelă.

Exemplu – introducerea unei linii în tabela agenda:

insert into agenda **values** (1, 'Popescu' , 'Ion' , 0217011010);

Observaţii:

- Remarcați introducerea datelor de intrare (valorile din interiorul parantezelor). Valorile numerice se introduc ca atare; valorile de tip string (șiruri de caractere alfanumerice) se introduc obligatoriu între apostrofuri.
- Valoarea pentru ID (1 din exemplul de mai sus) va identifica în mod unic o persoană din tabela agenda; de aceea pentru liniile pe care le veți introduce ulterior, utilizați ID-uri diferite (de exemplu: 2, 3, 4 etc.).

Exerciţiu:

i) introduceţi între aproximativ 8-10 linii în tabela agenda, apoi afişaţi conţinutul întregii tabele cu comanda: **select * from** agenda;

ii) introduceţi încă 2-3 linii folosind prima sintaxă a comenzii insert (în care se specifică numele coloanelor). Observaţi că puteţi trece coloanele *în orice ordine* (respectând corespondenţa nume coloană – valoare), şi *puteţi omite nume de coloană*.

Atenţie: datele introduse vor fi păstrate în baza de date doar dacă finalizaţi tranzacţia, adică după efectuarea unor operaţii executaţi comanda: COMMIT. Dacă aţi greşit ceva la un moment dat (nu doriţi ca modificările sa fie salvate în baza de date), puteţi renunţa la ele prin comanda ROLLBACK. Desigur, comanda ROLLBACK executată după COMMIT nu are nici un efect, modificările fiind deja salvate.

NOTĂ: Salvaţi secvenţa de instrucţiuni SQL de introducere a datelor in tabelă pentru a le folosi la punctul următor (Alegeţi din meniu comanda File->Save). Instrucţiunile vor fi salvate intr-un fişier text cu extensia .sql.

c) Ştergerea de linii dintr-o tabelă. Instrucţiunea DELETE

Ştergerea unei linii dintr-o tabelă se poate face conform următorului exemplu (testaţi direct):

```
DELETE FROM agenda  WHERE  ID = 4;
```

Va şterge linia care are valoarea ID egală cu 4, dacă există.

Afişaţi toate liniile tabelei pentru a vedea efectul comenzii de ştergere.

Efectul instrucţiunii: vor fi şterse toate liniile pentru care condiţia specificată în clauza where este adevărată. Exemplu: puteţi şterge liniile care conţin numele ‚Ionescu'.

Atenţie:

- Dacă uitaţi să folosiţi clauza where (condiţia), toate liniile tabelei vor fi şterse.
- Dacă ştergeţi accidental linii, puteţi anula modificările folosind comanda ROLLBACK.

d) Modificarea structurii tabelei – instrucţiunea ALTER TABLE

Setarea unei chei primare

O cheie primară a unei tabele este un atribut (coloană) sau grup de atribute care identifică în mod unic un tuplu (o linie) din tabelă. Conform modelului relaţional, o tabelă nu poate conţine tupluri (linii) identice, fiecare linie având o semnificaţie distinctă. Cheia primară este unul din mecanismele care asigură această unicitate a liniilor tabelei.

Setarea unei chei primare în tabela agenda poate fi făcută modificând tabela cu comanda ALTER TABLE:

```
alter table agenda
add constraint id primary key (id);
```

Atenţie: dacă tabela creată de dvs. nu respectă condiţia de unicitate a liniilor, atunci instrucţiunea de mai sus va returna o eroare, şi va trebui sa ştergeţi linia duplicat (a se vedea mai jos instrucţiunea de ştergere a liniilor dintr-o tabelă).

Alte două constrângeri frecvent utilizate sunt: NOT NULL şi DEFAULT.

Constrângerea NOT NULL specifică faptul că atributul (valorile din coloana) respectivă nu pot fi vide, iar constrângerea DEFAULT setează o valoare implicită pentru un atribut, dacă la introducere se omite valoarea sa.

În mod normal, setarea cheii primare, şi a celorlalte constrângeri (NOT NULL etc.) *trebuie făcută chiar la crearea tabelei*, după cum urmează:

```
create table agenda(
  id number(3) PRIMARY KEY,
  nume varchar2(20) NOT NULL,
  prenume varchar2(20),
  nr_tel number(10) NOT NULL);
```

Pentru a testa instrucţiunea de mai sus trebuie mai întâi să ştergeţi tabela (DROP TABLE agenda), sau să creaţi una nouă cu alt nume.

Dacă aţi rulat comanda DROP TABLE agenda, reintroduceţi datele în tabelă prin rularea scriptului salvat in fişierul cu extensia .sql la punctul anterior.

Incrementarea automată a cheii primare

În loc de a introduce manual valorile cheii primare (ID în cazul tabelei agenda), se poate automatiza introducerea prin setarea unei secvenţe de incrementare automată.

Comanda INSERT de mai sus va deveni:

create sequence PK_agenda;

insert into agenda **values** (PK_agenda.NEXTVAL, 'Popescu','Ion',0217011010);

NOTĂ: comanda **create sequence** trebuie rulată *o singură dată*, înainte de utilizarea în **insert** a codului pentru generarea valorii ID automat (PK_agenda.NEXTVAL). PK = primary key, iar NEXTVAL = următoarea valoare numerică (incrementare cu 1).

Valoarea de start a secvenței, precum și pasul de incrementare pot fi setate completând instrucțiunea de mai sus după cum urmează:

create sequence PK_agenda START WITH 5 INCREMENT BY 1;

Desigur, în loc de 5 și 1 puteți seta orice valoare.

Exerciţiu:

i) introduceţi aprox. 7-10 linii în tabela agenda folosind comenzi insert cu auto-incrementare. Apoi, verificaţi conţinutul tabelei cu comanda: **select * from** agenda.
ii) Încercaţi să introduceţi câteva linii specificând doar numele / valoarea ID, nume şi prenume. De ce sistemul returnează o eroare?

Modificarea structurii tabelei

Adăugarea de coloane la o tabelă se face prin intermediul comenzii ALTER TABLE, de exemplu:

ALTER TABLE agenda ADD DataNast DATE;

Exerciţiu: adăugaţi şi coloanele „Adresa” şi „Oras” la tabela agenda, apoi afişaţi liniile tabelei

Observaţie: coloanele DataNast, Adresa si Oras vor fi vide (null).

Notă: Pentru ştergerea unei coloane din schema tabelei se foloseşte comanda DROP COLUMN, de exemplu: ALTER TABLE agenda DROP COLUMN Nume_coloana;

Actualizarea liniilor unei tabele

Actualizarea liniilor unei tabele se face utilizând comanda UPDATE.

Exemplu (testaţi direct):

```
UPDATE agenda
   SET DataNast = DATE'1980-08-20'  where ID = 1;
```

Observaţi modul de introducere a datelor calendaristice prin specificarea tipului DATE.

Atenţie: dacă uitaţi să folosiţi clauza where (condiţia), toate liniile tabelei vor fi modificate.

O funcție ajutătoare pentru introducerea datelor calendaristice este *TO_DATE*. Exemple:

- TO_DATE('05-OCT-07', 'DD-MON-YY') va returna data de 5 octombrie 2007
- TO_DATE('05-OCT-2007', 'DD-MON-YYYY') tot 5 octombrie 2007
- TO_DATE('05-10-2007', 'DD-MM-YYYY') idem
- TO_DATE('05-October-07', 'DD-MONTH-YY') idem
- TO_DATE('05-X-07', 'DD-RM-YY') idem

Astfel, exemplul de mai sus poate fi scris şi după cum urmează:

```
UPDATE agenda
   SET DataNast = TO_DATE('20-08-1980', 'DD-MM-YYYY') where ID = 1;
```

În Oracle cel mai simplu mod de a introduce date calendaristice este formatul 'DD-MON-YYYY', de exemplu comanda de mai sus devine:

```
UPDATE agenda
   SET DataNast = '20-Aug-1980' where ID = 1;
```

Exerciţiu: modificaţi şi celelalte linii ale tabelei, alocând valori pentru noile coloane (DataNast, Adresa si Oras). Se pot actualiza mai multe coloane printr-o singură comandă, separând perechile nume_colana = valoare prin virgule (,).

Apoi, inseraţi alte câteva (2-3) linii noi în tabela folosind comanda INSERT.

Modificarea tipului de date al unei coloane

Modificarea tipului unei coloane după crearea tabelei nu este permis decât cu câteva excepţii, cum ar fi mărirea dimensiunii atributelor tip şir de caractere. De exemplu, dacă dorim să stocăm nume mai lungi de 20 de caractere putem executa comanda:

```
ALTER TABLE agenda MODIFY Nume VARCHAR2(30);
```

Probleme propuse

1) Creaţi o tabelă ANGAJATI având următoarea schemă:

ANGAJATI(ID, Nume, Prenume, CNP, Salariu)

Setaţi atributul ID cu rol de Cheie Primară, şi atributele Nume, Prenume şi CNP cu NOT NULL.

2) Adăugaţi coloana Data_Nast la tabela ANGAJATI creată anterior

3) Creaţi secvenţa de incrementare automată a cheii primare, şi apoi introduceţi 7-10 linii în tabela (folosind codul de incrementare automată pentru cheia primară). Afişaţi apoi conţinutul tabelei.

4) Majoraţi salariul cu 100 lei pentru primul angajat, cu 200 pentru al doilea. Majoraţi salariul cu 10% pentru angajaţii $3 \div N$ din tabelă.

5) Setaţi o valoare DEFAULT pentru Salariu, cu valoarea 1000, apoi introduceţi câteva linii specificând doar valorile coloanelor: ID, Nume, Prenume şi CNP.

Notă: Sintaxa de adăugare a constrângerii DEFAULT este:

```
ALTER TABLE Angajati MODIFY Salariu DEFAULT 1000;
```

6) Modificați printr-o singură instrucțiune numele tuturor angajaților, astfel încât să fie scris cu majuscule. Puteți folosi funcția upper().

7) Ștergeți printr-o singură instrucțiune SQL toți angajații care au salariul multiplu de 5.