CAPITOLUL 12

Crearea si Definirea de structuri tabelare

In acest capitol, discutam aspectele logice ale tabelelor si coloanelor si commenzile necesare pentru a construi tabele cu Constrangeri de Integri- tate. Acesta este primul din doua capitole care acopera subsetul comenzilor SQL cunoscut ca Limbajul de Definire a Datelor(DDL).

Structuri de date ORACLE

In acest curs vom discuta aspectele logice de creare a tabelelor.Consi- deratiile fizice sunt acoperite in celelalte cursuri.Desi tabelele pot fi create fara a da informatii de asezare fizice, ar trebui notat ca un utili- zator ORACLE trebuie sa fi primit privilegiul de CREATE TABLE de catre admi- nistratorul bazei de date si sa aiba alocat ceva spatiu de tabela pentru a crea tabele.

In general, structurile de date ORACLE pot fi rezumate dupa cum urmeaza.

- Tabelele pot fi create oricand, chiar cu utilizatori folosind baza de date.
- Nu este necesar sa specificati dimensiunea nici unei tabele. Aceasta este definita ultima prin cat spatiu a fost alocat bazei de date ca un intreg. Oricum, este important sa estimati cat de mult spatiu va utiliza o tabela.
- Structurile pot fi modificate online.
- Tabelele pot capata automat mai mult spatiu daca dimensiunea i- nitiala este umpluta.

Limbajul de Definire a Datelor (LDD)

LDD este un subset al comenzilor SQL folosit pentru a crea, modifica sau muta structurile bazei de date ORACLE, si deasemenea sa inregistreze informatii in Dictionarul de Date (acesta este discutat mai tarziu).

Denumirea unei tabele

Numele pe care-l alegeti pentru o tabela trebuie sa urmeze regulile standard pentru numirea unui obiect al unei baze de date ORACLE.

1. Numele trebuie sa inceapa cu o litera, A-Z sau a-z.
2. Poate contine litere, numerale si caracterele speciale underscore (_). Caracterele $ si # sunt deasemenea legale, dar folosirea lor este descurajata.
3. Numele este acelasi indiferent daca sunt folosite litere mari sau mici, de exemplu, EMP, emp, si eMp sunt toate aceeasi tabela.
4. Poate fi de maxim 30 caractere in lungime.
5. Numele nu trebuie sa duplice numele altui obiect din contul dumneavoastra.
6. Numele nu trebuie sa fie un cuvant rezervat SQL.

```
NUME                          VALID ?
----                          -------

EMP85                          da

85EMP                          nu; nu incepe cu o litera

FIXED_ASSETS                   da

FIXED ASSETS                   nu; contine un blank

UP DATE                        nu; cuvant rezervat SQL
```

Ar trebui sa folositi nume de descriere pentru tabele si alte obiecte ale bazei de date. Folositi acelasi nume sa descrie aceeasi entitate in doua tabele diferite. De exemplu, coloana cu numarul departamentului este numita DEPTNO in ambele EMP si DEPT.

Crearea unei tabele

Creati o noua tabela folosind comanda CREATE TABLE. Una dintre cele mai simple forme a acestei comenzi este cand informatia de baza pentru fiecare coloana este definita impreuna cu tipul ei de data si dimensiunea.

```
Sintaxa: CREATE TABLE nume tabela
         (nume coloana tip(dimensiune),
          nume coloana tip(dimensiune),
          ...);

Exemplu: CREATE TABLE DEPT
         (DEPTNO NUMBER(2),
         DNAME   VARCHAR2(12),
         LOC     VARCHAR2(12));
```

Numele coloanelor intr-o tabela trebuie sa fie unice.

Tipurile coloanelor

Cand creati o tabela trebuie sa specificati fiecare tip de data al co- loanei. Tabela de mai jos arata cele mai importante tipuri de date.

Tipul de data poate fi urmat de unul sau mai multe numere in paranteze care dau informatii despre latimea coloanei. Latimea coloanei determina la- timea maxima pe care valorile in coloana pot s-o aiba. VARCHAR2 trebuie sa aiba o dimensiune, dar cele implicite sunt disponibile.

Tabela de mai jos arata tipurile de date principale in ORACLE7.

Tip de date	Descriere
VARCHAR2(w)	Sir de caractere de lungime maxim w. Lungimea maxima este de 2000 caractere.
CHAR(w)	Sir de lungime fixa w. Lungimea implicita este 1. Lungimea maxima este 255.
NUMBER	Numere in virgula mobila cu precizie : 38 de cifre semnificative.
NUMBER(w)	Numere intregi de precizie w.
NUMBER(w,s)	Numere cu precizia w si scala s. Precizia reprezinta numarul maxim de cifre semnificative permise care nu pot depasi 38. Scala este numarul de pozitii zecimale inregistrate in dreapta punctului.

DATE	Valorile datei din 1 Ianuarie 4712 inainte de Hristos pana in 31 Decembrie 4712 dupa Hristos. Informatia de timp este deasemenea stocata.
LONG	Sir de caractere de lungime variabila de lungime 2 Gb, sau 2 la puterea 31 minus 1. Este permisa o singura coloana de tipul LONG pe tabela.
RAW si LONG, LONG RAW	Echivalent cu VARCHAR2 si respectiv dar folosit pentru a stoca date binare ca imagini grafice sau sunete digitizate.

NOTE:

- Tipul de date VARCHAR2 al Oracle7 este sinonim cu versiunea 6 a tipu- rilor de date 'VARCHAR' si 'CHAR'.
- ORACLE foloseste semantica prin comparatie non-padded daca o valoare VARCHAR2 este folosita intr-o comparatie(vezi Capitolul 2 pentru de- talii).
- Daca o valoare scrisa intr-o coloana depaseste scala coloanei, atunci va apare o rotunjire.

Tabela de mai jos arata exemple de specificatii de coloane:

NUMBER(4)	Poate contine toate numerele pana la 4 cifre.
NUMBER(8,3)	Poate contine pana la 8 cifre, dintre care 3

	pot fi in dreapta punctului zecimal.
VARCHAR2(1000)	Valorile pot contine pana la 1000 caractere.
CHAR(80)	Siruri de caractere de lungime fixa egala cu 80. Valorile mai scurte sunt inlocuite cu blank-uri.

Crearea tabelei EMP

Tabela noastra demonstrativa, EMP, ar fi putut fi creata de o comanda CREATE TABLE ca mai jos:

```
CREATE TABLE EMP
(EMPNO       NUMBER(4) NOT NULL,
ENAME        VARCHAR2(10),
JOB          VARCHAR2(10),
MGR          NUMBER(4),
HIREDATE     DATE,
SAL          NUMBER (7,2),
COMM         NUMBER (7,2),
DEPTNO       NUMBER (2) NOT NULL);
```

Comanda DESCRIBE a SQL*PLUS-ului poate fi folosita pentru a lista detal ale coloanelor create intr-o tabela:

```
DESCRIBE EMP
```

Constrangerea NOT NULL

In exemplul de mai sus, veti observa ca definitiile pentru coloanele EMPNO si DEPTNO sunt urmate de NOT NULL. Aceasta ne asigura ca valoril nule sunt permise

pentru aceste coloane, de cand aceste coloane servesc ca chei pentru operatii pe aceasta tabela. Coloanele fara constrangerea NOT NULL permit valori nule.

NOT NULL este una dintre constrangerile de integritate care pot fi defi- nite.

OPTIUNEA DEFAULT

Unei coloane ii poate fi data o valoare implicita prin optiunea DEFAULT. Aceasta previne aparitia de null-uri (sau erori, daca NOT NULL este specifi- cata) daca o linie este inserata fara o valoare din coloana.Valorile implici- te pot fi literali, o expresie, dar nu numele altei coloane. Functii ca SYSDATE si USER sunt valide.

De exemplu:

```
HIREDATE DATE DEFAULT SYSDATE,

SAL NUMBER (7,2) DEFAULT 0
```

Constrangeri de integritate

Oracle permite constrangerilor de integritate sa fie definite pentru ta- bele si coloanelor sa forteze reguli sigure, inauntrul unei tabele sau intre tabele.

Constrangerile sunt folosite:

- de serverul Oracle7 sa forteze reguli la nivelul tabelei oricand este inserata o linie, actualizata sau stearsa din acea tabela. Constrange- rea trebuie sa fie satisfacuta pentru ca operatiile sa reuseasca.
- pentru a preveni stergerea unei tabele daca sunt posesiuni din alte tabele.
- prin unelte sigure Oracle, ca Oracle Forms, pentru a furniza reguli pentru utilizarea intr-o aplicatie.

Constrangerile sunt clasate dupa cum urmeaza:

Constrangeri de tabela
Acestea pot referi una sau mai multe coloane si sunt definite SEPARAT de definitiile coloanelor din tabela.
Constrangeri de coloana
Acestea refera o singura coloana si sunt definite INAUNTRUL specificatiei pentru coloana posesoare.

Constrangerile pot fi adaugate unei tabele dupa crearea ei si deasemenea temporar dezactivate (vezi comanda ALTER TABLE in capitolul urmator).Toate detaliile despre constrangeri sunt stocate in Dictionarul de Date. Fiecarei constrangeri ii este repartizat un nume. Iti este mai usor sa suplimentezi una tu singur, astfel ca poate fi mai usor referita mao tarziu, dar daca nu, atunci un nume este generat automat pe forma:

```
SYS_Cn
```

unde n este un numar unic. Cuvantul cheie CONSTRAINT iti permite sa numesti o noua constrangere tu insuti.

Tipuri de constrangeri

Puteti defini urmatoarele tipuri de constrangeri:

- NULL/NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY(integritatea de referinta)
- CHECK

Constrangerea UNIQUE

Aceasta desemneaza o coloana sau o combinatie de coloane ca o cheie uni- ca.Doua linii in aceeasi tabela nu pot avea aceeasi valoare pentru aceasta cheie.NULL-urile sunt permise daca cheia unica este bazata pe o singura co- loana.

Sintaxa constrangerii de tabela :

```
[CONSTRAINT nume constrangere] UNIQUE (Coloana, Coloana, ...)
```

Sintaxa constrangerii de coloana :

```
[CONSTRAINT nume constrangere] UNIQUE
```

De exemplu, pentru a va asigura ca nu sunt 2 nume de departamente identice la o singura locatie:

```
CREATE TABLE DEPT
(DEPTNO NUMBER, DNAME VARCHAR2(9),
LOC VARCHAR2(10),
CONSTRAINT UNQ_DEPT_LOC UNIQUE(DNAME,LOC))
```

In exemplul de mai sus, constrangerea UNQ_DEPT_LOC este o constrangere de ta- bela. Notati ca virgula precede detaliile. O constrangere unica il face pe ORACLE sa creeze un singur index pentru a manui regula. Indecsii sunt discu- tati mai tarziu.

Constrangere de cheie primara

Ca si la cheile unice, o cheie primara forteaza unicitatea unei coloane sau combinatii de coloane implicate si un index unic este creat pentru a con- duce aceasta. Totusi poate fi o singura cheie primara pe o tabela, si aceasta este cunoscuta ca fiind cheia definitiva prin care liniile in tabela sunt i- dentificate individul. NULL-urile nu sunt permise in coloanele de chei prima- re.

Sintaxa constrangerii de tabela :

```
       [CONSTRAINT nume constrangere] PRIMARY KEY (Coloana, Coloana,
...)
```

Sintaxa constrangerii de coloana :

```
       [CONSTRAINT nume constrangere] PRIMARY KEY
```

Notati ca aceeasi combinatie de coloane nu poate fi folosita si pentru o cheie primara si pentru una unica. Urmatorul exempludefineste DEPTNO ca o cheie primara folosind o constrangere de coloana:

```
CREATE TABLE DEPT
(DEPTNO NUMBER(2) CONSTRAINT DEPT_PRIM PRIMARY KEY, ...)
```

Constrangere de cheie externa

Cheile externe furnizeaza reguli de integritate de referinta inauntrul unei tabele sau intre tabele. O cheie exeterna este folosita intr-o relatie cu fiecare cheie primara sau unica oriunde si poate fi folosita, de exemplu, pentru a preveni stergerea unui departament in DEPT daca angajatii exista cu acelasi numar de departament in EMP.

Sintaxa constrangerii de tabela :

```
       [CONSTRAINT nume constrangere] FOREIGN KEY (Coloana, Coloana,
...)
       REFERENCES tabela (Coloana, Coloana, ...)
```

Sintaxa constrangerii de coloana :

```
       [CONSTRAINT nume constrangere] REFERENCES tabela (Coloana)
```

Notati ca, cuvintele 'FOREIGN KEY' nu sunt folosite versiunea constrangerii de coloana a sintaxei.

```
                     ------------
Exemplul 1           |   DEPT   |  (parinte)
                     ------------
```

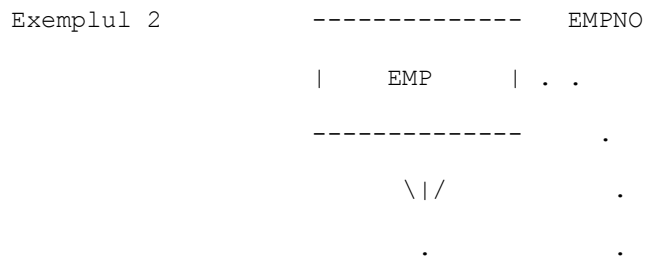
Pentru a stabili relatia dintre EMP si DEPT astfel incat EMP.DEPTNO este che- ia externa, si fiecare angajat trebuie sa aiba un numar valid de departament care este cunoscut in DEPT:

```
CONSTRAINT FK_DEPTNO FOREIGN KEY (DEPTNO)
REFERENCES DEPT(DEPTNO)
```

Optiunea ON DELETE CASCADE

Ca rezultat al constrangerii de tabela de mai sus, (care ar fi putut la fel de bine sa fie definita ca o constrangere de coloana), un departament in DEPT nu ar fi putut fi sters daca liniile exista in EMP cu aceeasi valoare DEPTNO. Alternativ, puteti cere ca angajatii corespunzatori sa fie stersi a- utomat daca departamentul parinte in DEPT este sters. Aceasta este realizata adaugand clauza ON DELETE CASCADE.

```
CONSTRAINT FK_DEPTNO FOREIGN KEY (DEPTNO)
REFERENCES DEPT(DEPTNO) ON DELETE CASCADE
```



. .

MGR . .

. .

Pentru a va asigura ca fiecarei linii de angajat in EMP ii este dat un numar

de manager (MGR) pentru un angajat existent valid:

```
CREATE TABLE EMP
(EMPNO NUMBER(4) PRIMARY KEY, ...
MGR NUMBER(4) CONSTRAINT EMP_MGR
REFERENCES EMP(EMPNO), ...
```

Constrangerea de verificare (CHECK)

Constrangerea CHECK defineste explicit o conditie pe care fiecare linie

trebuie sa o satisfaca(sau sa o faca necunoscuta datorita unui NULL). Condi-

tia poate folosi aceleasi constructii ca acelea intr-o restrictie de cerere,

cu urmatoarele exceptii:

- subcererile nu sunt permise
-
-
- referirile la pseudo-coloane ca SYSDATE nu sunt permise
-
-

Sintaxa:

[CONSTRAINT nume constrangere] CHECK (conditie)

Alte optiuni ale constrangerilor

DISABLE	Adaugand DISABLE unei definitii de constrangere
	inseamna ca ORACLE nu o forteaza. Constrangerea poate
	fi inca citita de uneltele ORACLE pentru a construi
	reguli intr-o aplicatie si puteti face posibila con-
	strangerea mai tarziu prin comanda ALTER TABLE.

CREATE TABLE EMP

(. . . . . ,

ENAME VARCHAR2(10) CONSTRAINT CHK_UPP_NAM CHECK(ENAME-UPPER(ENAME))

DISABLE,

. . . . .) ;

EXCEPTIONS Identifica o tabela existenta unde este plasata

INTO nume tabela informatia despre liniile care incalca constrangerea.

CREATE TABLE EMP

(. . . . . ,

ENAME VARCHAR2(10) CONSTRAINT CHK_UPP_NAM CHECK(ENAME-UPPER(ENAME))

EXCEPTIONS INTO CON_VIOLATE,

. . . . .) ;

In final, notati ca tabelele referite intr-o constrangere trebuie sa existe

in aceeasi baza de date. Daca ele apartin unui utilizator diferit atunci po-

sesorul trebuie specificat ca un prefix.

Iata un exemplu complet al constructiei tabelei EMP cu constrangeri :

```
CREATE TABLE EMP

(EMPNO NUMBER(4)     CONSTRAINT EMP_PRIM PRIMARY KEY,

ENAME VARCHAR2(10)   CONSTRAINT ENAME_CONS
                     CHECK(ENAME=UPPER(ENAME)),

JOB VARCHAR2(10),

MGR NUMBER(4)        CONSTRAINT EMP_MGR
                     REFERENCES EMP(EMPNO),

HIREDATE DATE DEFAULT SYSDATE,

SAL NUMBER(7,2)      CONSTRAINT SAL_CONS
                     NOT NULL,

COMM NUMBER(7,2),

DEPTNO NUMBER(2)     CONSTRAINT DEPTNO_CONS
                     NOT NULL,

CONSTRAINT           EMP_DEPT FOREIGN KEY (DEPTNO)
                     REFERENCES DEPT(DEPTNO))
```

Crearea unei tabele cu linii din alta tabela

Exista o a doua forma a declaratiei CREATE TABLE in care tabela este

creata cu linii potrivite, derivate din alta tabela:

```
CREATE TABLE DEPT
[(nume-coloana ,. . . .)]
AS SELECT declaratie
```

- Tabela va fi creata cu coloane specificate si linii recuperate din de-
- claratia SELECT inserata.
- Numai constrangerile NULL/NOT NULL sunt mostenite din tabela selecta-
- ta.
- Daca toata coloanele in declaratia SELECT au nume bine definite (nu
- sunt expresii s.a.m.d.) specificatiile coloanei pot fi omise.
- Daca sunt date specificatiile coloanei, atunci numarul de coloane tre-
- buie sa fie egal cu numarul de articole in lista SELECT.

Pentru a crea o tabela DEPT30 care tine numerele angajatilor, nume, job-uri

si salariile angajatilor din departamentul 30, introduceti:

```
CREATE TABLE DEPT30
AS
SELECT       EMPNO,ENAME,JOB,SAL
FROM         EMP
WHERE        DEPTNO = 30;
```

Table created.

Pentru a vedea descrierea lui DEPT30, introduceti:

```
DESC DEPT30
```

Pentru a crea o tabela tinand numele angajatului, salariul si detalii de

grad, introduceti:

```
CREATE TABLE EMP_SALS
(NAME,SALARY,GRADE)
```

```
AS
SELECT      ENAME, SAL, GRADE
FROM        EMP, SALGRADE
WHERE       EMP.SAL BETWEEN LOSAL AND HISAL;
```

Table created.

```
DESC EMP_SALS;
```

Pentru a afisa continutul tabelei EMP SALS, introduceti:

```
SELECT      *
FROM        EMP_SALS;
```

,/pre>

CAPITOLUL 12 Exercitii - Crearea de tabele si de constrangeri

Atelier

1. Creati o tabela numita PROJECTS, cu coloanele specificate ca mai jos.Pe
2.
3. langa aceasta , definiti PROJID ca, coloana de CHEIE PRIMARA , si asigura-
4.
5. ti-va ca datele P_END_DATE nu sunt mai recente decat datele P_START_DATE.
6.
7.
8.
9.
10.
11.
12. Creati o a doua tabela, ASSIGNMENTS, ca cea de mai jos. Definiti-i coloana
13.
14. PROJID ca o cheie externa care refera tabela PROJECTS. Coloana EMPNO a ta-
15.
16. belei dumneavoastra este o viitoare cheie externa a lui EMP. Aceste doua
17.
18. coloane nu ar trebui sa permita valori NULL (PROJID si EMPNO).
19.
20.
21.
22.
23.
24. Sunt alte constrangeri pe care ar trebui sa le angajam in aceste tabele.
25.
26. Acestea vor fi adaugate intr-un viitor atelier.
27.
28.
29.
30.
31. Folositi comanda DESCRIBE pentru a verifica definitiile coloanelor. (Ne
32.
33. vom uita la constrangeri mai tarziu).
34.
35.
36.
37.

CAPITOLUL 12 -Solutii

Fiecare din constrangerile aratate mai jos ar fi putut fi definite ca orica-

re constrangere de tabela sau constrangere de coloana.

```
1.
2.
3.
4.          CREATE TABLE PROJECTS
5.
6.          ( PROJID            NUMBER(4) CONSTRAINT PROJ_PRIM
   PRIMARY KEY,
7.
8.          P_DESC              VARCHAR2(20),
9.
10.         P_START_DATE        DATE,
11.
12.         P_END_DATE          DATE,
13.
14.         BUDGET_AMOUNT       NUMBER(7,2),
15.
16.         MAX_NO_STAFF        NUMBER(2),
17.
18.         CONSTRAINT  P_DATE_RULE CHECK(P_START_DATE <=
   P_END_DATE))
19.
20.
21.
22.
23.           CREATE TABLE ASSIGNMENTS
24.
25.         (PROJID             NUMBER(4) NOT NULL REFERENCES
   PROJECTS (PROJID),
26.
27.         EMPNO               NUMBER(4) NOT NULL REFERENCES EMP
   (EMPNO),
28.
29.         A_START_DATE        DATE,
30.
```

```
31.         A_END_DATE          DATE,
32.
33.         BILL_RATE           NUMBER(4,2),
34.
35.         ASSIGN_TYPE         VARCHAR2(2))
36.
```