Crearea si definirea de tabele

Un utilizator ORACLE trebuie sa fi primit privilegiul de CREATE TABLE si sa aiba alocat 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 niciunei tabele. 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 initiala este ocupata (migrarea randurilor).

Limbajul de Definire a Datelor (LDD)

LDD este un subset al comenzilor SQL folosit pentru a crea, modifica sau sterge structurile bazei de date ORACLE, si deasemenea sa inregistreze informatii in Dictionarul de Date.

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, cifre 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.
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

      UPDATE                     nu; cuvant rezervat SQL
```

Ar trebui sa folositi nume explicative 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 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 pentru fiecare coloana cate un tip de data.

Tipul de data poate fi urmat de latimea coloanei. Latimea coloanei determina latimea maxima pe care valorile in coloana pot s-o aiba.

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

Tip de date	Descriere
VARCHAR2(w)	Sir de caractere de lungime maxim w. Lungi-mea 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 precizia 38 de cifre semnificative.
NUMBER(w)	Numere intregi de precizie w.
NUMBER(w,s)	Numere cu precizia w si scala s. Precizia

	reprezinta numarul total de cifre, care nu poate 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.
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 respectiv LONG, LONG RAW	Echivalent cu VARCHAR2 si dar folosit pentru a stoca date binare ca imagini grafice sau sunete digitizate.

Daca o valoare scrisa intr-o coloana depaseste scala coloanei, atunci va aparea 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 umplute la dreapta cu blank-uri.

Crearea tabelei EMP

Tabela EMP este 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);
```

Constrangerea NOT NULL

In exemplul de mai sus, definitiile pentru coloanele EMPNO si DEPTNO sunt urmate de NOT NULL. Aceasta ne asigura ca valorile nule nu sunt permise pentru aceste coloane. Coloanele fara constrangerea NOT NULL permit valori nule.

NOT NULL este una dintre constrangerile de integritate care pot fi definite.

OPTIUNEA DEFAULT

Unei coloane ii poate fi alocata o valoare implicita prin optiunea DEFAULT. Aceasta previne aparitia de null-uri (sau erori, daca NOT NULL este specificata) daca o linie este inserata fara o valoare din coloana. Functii ca SYSDATE si USER sunt valide.

De exemplu:

```
HIREDATE DATE DEFAULT SYSDATE,
SAL NUMBER (7,2) DEFAULT 0
```

Constrangeri de integritate

Constrangerile sunt clasificate 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 in specificatia pentru coloana posesoare.

Constrangerile pot fi adaugate unei tabele dupa crearea ei si se pot dezactiva temporar (ALTER TABLE). Toate detaliile despre constrangeri sunt stocate in Dictionarul de Date. Fiecarei constrangeri ii este repartizat un nume. Cuvantul cheie CONSTRAINT permite denumirea unei constrangeri.

Tipuri de constrangeri

Putem 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 unica. 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 coloana.

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 tabela. Notati ca virgula precede detaliile.

Constrangere de cheie primara

Ca si la cheile unice, o cheie primara forteaza unicitatea unei coloane sau combinatii de coloane. Poate fi o singura cheie primara pe o tabela. 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 exemplu defineste 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. O cheie externa este folosita intr-o relatie cu o cheie primara sau unica: pentru a preveni stergerea unui departament in DEPT daca angajatii exista cu acelasi numar de departament in EMP.

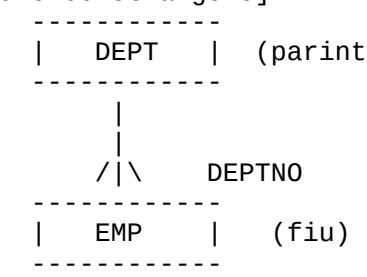
Sintaxa constrangerii de tabela:

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

Sintaxa constrangerii de coloana:

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

Exemplul 1



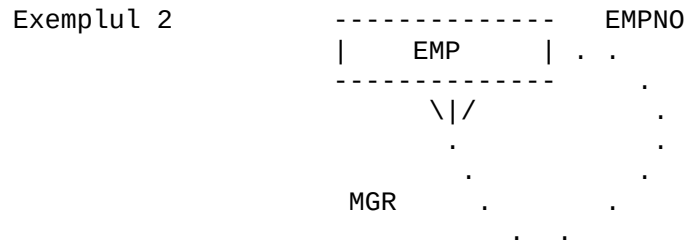
Pentru a stabili relatia dintre EMP si DEPT astfel incat EMP.DEPTNO este cheia externa, si fiecare angajat trebuie sa aiba un numar valid de departament care este cunoscut in DEPT:

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

Optiunea ON DELETE CASCADE

Ca rezultat al constrangerii de tabela de mai sus, 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 automat 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
```



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.

Sintaxa:

```
[CONSTRAINT nume constrangere] CHECK (conditie)
```

Alte optiuni ale constrangerilor

DISABLE constrangere	Adaugand DISABLE unei definitii de inseamna ca ORACLE nu o aplica.
Constrangerea poate construi posibila con-	fi citita de uneltele ORACLE pentru a reguli intr-o aplicatie si puteti face 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_PROBLEMS,
     . . . . . ) ;
```

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 declaratia SELECT inserata.
- Numai constrangerile NULL/NOT NULL sunt mostenite din tabela selectata.
- Daca toate coloanele in declaratia SELECT au nume bine definite (nu sunt expresii) specificatiile coloanei pot fi omise.
- Daca sunt date specificatiile coloanei, atunci numarul de coloane trebuie 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;
```

Exercitii - Crearea de tabele si de constrangeri

Creati o tabela numita PROJECTS, cu coloanele specificate ca mai jos.Pe langa aceasta , definiti PROJID CHEIE PRIMARA si asigurati-va ca datele P_END_DATE nu sunt mai recente decat datele P_START_DATE. PROJID P_DESC P_START_DATE P_END_DATE BUDGET_AMOUNT MAX_NO_STAFF

Creati o a doua tabela, ASSIGNMENTS. Definiti-i coloana PROJID ca o cheie externa care refera tabela PROJECTS. Coloana EMPNO a ta- belei dumneavoastra este o viitoare cheie externa a lui EMP. Aceste doua coloane nu ar trebui sa permita valori NULL (PROJID si EMPNO).

Folositi comanda DESCRIBE pentru a verifica definitiile coloanelor.

Solutii

Fiecare din constrangerile aratate mai jos ar fi putut fi definite de tabela sau constrangere de coloana.

```
      CREATE TABLE PROJECTS
      ( PROJID           NUMBER(4) CONSTRAINT PROJ_PRIM
PRIMARY KEY,
      P_DESC             VARCHAR2(20),
      P_START_DATE       DATE,
      P_END_DATE         DATE,
      BUDGET_AMOUNT      NUMBER(7,2),
      MAX_NO_STAFF       NUMBER(2),
      CONSTRAINT  P_DATE_RULE CHECK(P_START_DATE <=
P_END_DATE))


        CREATE TABLE ASSIGNMENTS
      (PROJID            NUMBER(4) NOT NULL REFERENCES
PROJECTS (PROJID),
      EMPNO              NUMBER(4) NOT NULL REFERENCES EMP
(EMPNO),
      A_START_DATE       DATE,
      A_END_DATE         DATE,
      BILL_RATE          NUMBER(4,2),
      ASSIGN_TYPE        VARCHAR2(2))
```