

# Restrictii de integritate

- Ex. 

```
ALTER TABLE Dept  
ADD PRIMARY KEY (Deptno);  
ALTER TABLE Emp  
ADD FOREIGN KEY (Deptno) REFERENCES Dept(Deptno);
```
- Crearea indecsilor.
  - Orice cheie unica sau primara cere un index.
  - Cheile unice sau primare pot folosi indecsi unici sau neunici.
  - Cel mult o cheie unica sau primara poate folosi un index neunic.
  - Ordinea coloanelor in index si ordinea coloanelor din restrictie nu trebuie sa fie aceleasi.
  - Noi trebuie sa indexam cheile straine.
- NOT NULL
  - Implicit toate coloanele sunt de tip NULL.
  - Orice cimp din PRIMARY KEY trebuie sa fie cu NOT NULL.

# Restrictii de integritate

- ALTER TABLE Emp MODIFY ename NOT NULL;
- Cheie primara:
  - Fiecare tabela trebuie sa aiba o cheie primara, ce identifica fiecare linie din tabela.
  - Folosim o coloana ce contine un numar de secventa.
  - Alegem o coloana pentru care valorile ei sunt unice.
  - Datele unei coloane ce defineste cheia primara trebuie sa fie neschimbabile.
  - O cheie primara nu contine valori nule.
  - Alegem o coloana care este de mica dimensiune si numerica.
  - Evitam folosirea cheilor primare compuse.
- Cheie unica:
  - Putem defini cheie unica pentru orice coloana care nu permite chei duplicate(dar ea poate fi de tip NULL).

# Restriții de integritate

- - Nu putem avea valori identice în coloanele NOT NULL ale unei chei compuse UNIQUE
    - Restricția de tip cheie unică permite NULL.
  - Restriții de integritate referențială:
    - Fie două tabele T1 cu restricția PRIMARY sau UNIQUE pe coloana col1 și tabela T2 cu restricția FOREIGN pe câmpul col2. Câmpurile col1 și col2 – echivalente semantic.
    - Pentru fiecare valoare v a lui col2 din tabela T2 trebuie să existe o linie din T1 care are valoarea v în atributul col1.
- In cazul cheilor compuse, șirul de coloane din cheia primară sau unică trebuie să aibă același tip cu șirul de coloane din cheia FOREIGN.
- Numărul atributelor din cheia compusă  $\leq 32$ .
  - Cheile FOREIGN permit valori toate NULL, deci în acest caz nu cuplează cu o valoare a cheii unice sau primare.

# Restrictii de integritate

- Prin default(fara niciun NOT NULL sau CHECK) FOREIGN KEY specifica regula “fara cuplare” pentru chei straine compuse in modul ANSI/ISO.

- Regula “cuplare completa” pentru valori NULL in cheia compusa va cere ca toate componentele cheii sa fie NULL sau toate sa fie NOT NULL. Pentru aceasta vom folosi restrictia CHECK:

```
CHECK ((cimp1 IS NULL AND cimp2 IS NULL) OR  
        (cimp1 IS NOT NULL AND cimp2 IS NOT NULL))
```

- Relatii intre tabelele T1 si T2. T1-parinte, T2-copil:

1)Fara restrictii pe cheia FOREIGN:

Mai multe linii din tabela copil pot referi aceeasi linie din tabela parinte. Se permite valori NULL ale cheii FOREIGN.

Rezulta o functie 1-n intre T1 si T2 . Fiecare departament are mai multi functionari, pot exista functionari pentru care nu se cunoaste departamentul-are valoarea NULL in cheia FOREIGN.

# Restrictii de integritate

## 2) Restrictia NOT NULL pe cheia FOREIGN:

Fiecare linie din tabela copil trebuie sa refere o linie din tabela parinte. Un numar oarecare de linii din tabela copil pot referi aceeasi linie din tabela parinte.

## 3) Restrictia UNIQUE pe cheia straina:

Se permite NULL pe cheia straina, dar o singura linie din tabela copil poate referi o linie data a tabelii parinte.

```
Ex. CREATE TABLE Insurance(Memberno Number,...,  
    CONSTRAINT c3 PRIMARY KEY(Memberno));  
CREATE TABLE Emp1(Memberno NUMBER,..  
    CONSTRAINT c1 UNIQUE KEY(Memberno),  
    CONSTRAINT c2 FOREIGN KEY(Memberno)  
REFERENCES (Insurance.Memberno));
```

# Restrictii de integritate

4) Restrictia UNIQUE si NOT NULL pe cheia straina:

O singura linie din tabela copil refera o linie data din tabela parinte si fiecare linie din tabela copil trebuie sa refere o linie in tabela parinte.

5) Reguli pentru restrictii FOREIGN KEY multiple.

- Cind o restrictie nu este satisfacuta se semnaleaza o eroare. Putem folosi SET CONSTRAINTS pentru realizarea verificarii restrictiei la sfirsitul tranzactiei. Operatia este numita “aminarea” verificarii restrictiei.

```
SET CONSTRAINTS{ ALL/r1,r2,..rh} {IMMEDIATE/DEFERRED};  
CREATE TABLE numet (... ,CONSTRAINT c1... DEFERRABLE,...
```

Asigurarea ca restrictiile au fost definite ca aminabile.(FOREIGN, UNIQUE, PRIMARY KEY).

# Restrictii de integritate

- CREATE TABLE dept ( deptno NUMBER PRIMARY KEY, dname VARCHAR2 (30) );
- CREATE TABLE emp ( empno NUMBER, ename VARCHAR2 (30), deptno NUMBER REFERENCES (dept),  
**CONSTRAINT epk PRIMARY KEY (empno) DEFERRABLE,**  
**CONSTRAINT efk FOREIGN KEY (deptno)**  
**REFERENCES (dept.deptno) DEFERRABLE);**  
INSERT INTO dept VALUES (10, 'Accounting');  
INSERT INTO dept VALUES (20, 'SALES');  
INSERT INTO emp VALUES (1, 'Corleone', 10);  
INSERT INTO emp VALUES (2, 'Costanza', 20);  
COMMIT;  
SET CONSTRAINT efk DEFERRED;

# Restrictii de integritate

```
UPDATE dept SET deptno = deptno + 10 WHERE deptno = 20;
```

```
SELECT * from emp ORDER BY deptno;
```

```
EMPNO ENAME DEPTNO
```

```
-----
```

```
1      Corleone    10
```

```
2      Costanza   20
```

```
UPDATE emp SET deptno = deptno + 10 WHERE deptno = 20;
```

```
SELECT * FROM emp ORDER BY deptno;
```

```
EMPNO ENAME DEPTNO
```

```
-----
```

```
1      Corleone    10
```

```
2      Costanza   30
```

```
COMMIT;
```

# Restrictii de integritate

- La crearea unei chei primare sau unice se verifica daca exista index, daca nu se creaza.
- Cind se foloseste un index pentru o restrictie si restrictia este eliminata, atunci se sterge si indexul. Pentru pastrarea indexului se da clauza `KEEP INDEX` in `DROP`..
- Cind avem referinta la o cheie primara sau unica, nu putem sterge restrictia de cheie primara sau unica sau indexul.
- Cheile `UNIQUE` si `PRIMARY` cu restrictii aminabile trebuie sa folosesca indecsi neunici.
- Pentru reutilizarea indecsilor existenti cind se creaza restrictii de cheie unica sau primara, putem include clauza `USING INDEX`:

```
CREATE TABLE b ( b1 INTEGER, b2 INTEGER, CONSTRAINT  
unique1 (b1, b2) USING INDEX (CREATE UNIQUE INDEX  
b_index on b(b1, b2), CONSTRAINT unique2 (b1, b2) USING  
INDEX b_index );
```

# Restrictii de integritate

- Restrictia de integritate CHECK:
    - Nu folosim CHECK atunci cind celelalte tipuri pot asigura verificarea necesara.
    - CHECK <expresie\_booleana>
    - Evaluarea expresiei pentru linia care se adauga sau se modifica.
    - Expresia nu poate contine subinterogari
    - Expresia nu poate contine functiile SYSDATE, UID, USER, USERNV.
    - Expresia nu poate contine pseudocoloanele LEVEL, PRIOR, ROWNUM.
    - Expresia nu poate contine o functie utilizator.
    - Restrictia este violata daca conditia este FALSE, altfel nu.
- Ex. CHECK(Sal>0 AND Comm>=0)
- O coloana poate avea multiple restrictii CHECK

# Restrictii de integritate

- O restrictie de tip NOT NULL este un exemplu de restrictie CHECK:  
CHECK(nume\_col IS NOT NULL)

Cind o cheie compusa permite toate valorile null sau toate valorile nenule, atunci folosim CHECK.

```
CREATE TABLE Dept_tab ( Deptno NUMBER(3) CONSTRAINT  
Dept_pkey PRIMARY KEY, Dname VARCHAR2(15), Loc  
VARCHAR2(15), CONSTRAINT Dname_ukey UNIQUE (Dname,  
Loc), CONSTRAINT Loc_check1 CHECK (loc IN ('NEW YORK',  
'BOSTON', 'CHICAGO')));
```

```
CREATE TABLE Emp_tab ( Empno NUMBER(5) CONSTRAINT  
Emp_pkey PRIMARY KEY, Ename VARCHAR2(15) NOT NULL,  
Job VARCHAR2(10), Mgr NUMBER(5) CONSTRAINT Mgr_fkey  
REFERENCES Emp_tab, Hiredate DATE, Sal NUMBER(7,2),  
Comm NUMBER(5,2), Deptno NUMBER(3) NOT NULL  
CONSTRAINT dept_fkey REFERENCES Dept_tab ON DELETE  
CASCADE);
```

# Restrictii de integritate

```
CREATE UNIQUE INDEX I_dept ON Dept_tab(deptno);  
ALTER TABLE Dept_tab ADD CONSTRAINT Dept_pkey  
PRIMARY KEY (deptno);
```

```
ALTER TABLE Emp_tab ADD CONSTRAINT Dept_fkey FOREIGN  
KEY (Deptno) REFERENCES Dept_tab;
```

```
ALTER TABLE Emp_tab MODIFY (Ename VARCHAR2(15) NOT  
NULL);
```

- Nu se poate crea o restrictie pe o tabela care contine linii ce violeaza restrictia.
- Userul ce creaza o restrictie trebuie sa aiba privilegiul de a crea tabele, sau sa modifice tabele.

```
CREATE TABLE Emp_tab ( FOREIGN KEY (Deptno)  
REFERENCES Dept_tab);
```

```
CREATE TABLE Emp_tab ( FOREIGN KEY (Deptno)  
REFERENCES Dept_tab ON DELETE CASCADE);
```

```
CREATE TABLE Emp_tab ( FOREIGN KEY (Deptno)  
REFERENCES Dept_tab ON DELETE SET NULL);
```

# Restrictii de integritate

Vizualizarea restrictiilor din dictionarul de date:

- ALL\_CONSTRAINTS
- ALL\_CONS\_COLUMNS
- USER\_CONSTRAINTS
- USER\_CONS\_COLUMNS
- DBA\_CONSTRAINTS
- DBA\_CONS\_COLUMNS

Operatiile de ENABLE sau DISABLE pentru restrictii:

- La definirea unei restrictii(CREATE TABLE sau ALTER TABLE restrictia este automat ENABLE.

```
CREATE TABLE Emp_tab ( Empno NUMBER(5) PRIMARY KEY  
DISABLE);
```

```
ALTER TABLE Emp_tab ADD PRIMARY KEY (Empno) DISABLE;
```

# Restrictii de integritate

- ALTER TABLE Dept\_tab ENABLE CONSTRAINT Dname\_ukey;  
ALTER TABLE Dept\_tab ENABLE PRIMARY KEY ENABLE  
UNIQUE (Dname) ENABLE UNIQUE (Loc);
- ALTER TABLE Dept\_tab DISABLE CONSTRAINT Dname\_ukey;  
ALTER TABLE Dept\_tab DISABLE PRIMARY KEY DISABLE  
UNIQUE (Dname) DISABLE UNIQUE (Loc);
- ALTER TABLE Dept\_tab DROP UNIQUE (Dname);
- ALTER TABLE Dept\_tab DROP UNIQUE (Loc);
- ALTER TABLE Emp\_tab DROP PRIMARY KEY, DROP  
CONSTRAINT Dept\_fkey;
- DROP TABLE Emp\_tab CASCADE CONSTRAINTS;

## Tabele nested include

- CREATE TYPE phone AS OBJECT (telephone NUMBER)  
/  
CREATE TYPE phone\_list AS TABLE OF phone  
/  
CREATE TYPE my\_customer AS OBJECT ( cust\_name  
VARCHAR2(25), phones phone\_list)  
/  
CREATE TYPE customer\_list AS TABLE OF my\_customer  
/  
CREATE TABLE business\_contacts ( company\_name  
VARCHAR2(25), company\_reps customer\_list) NESTED TABLE  
company\_reps STORE AS outer\_ntab (NESTED TABLE phones  
STORE AS inner\_ntab);

# Renunmirea restrictiilor de integritate

- prompt Enter table name to find its primary key:  
accept table\_name  
select constraint\_name from user\_constraints where table\_name =  
upper('&table\_name.') and constraint\_type = 'P';  
prompt Enter new name for its primary key:  
accept new\_constraint  
set serveroutput on  
declare  
-- USER\_CONSTRAINTS.CONSTRAINT\_NAME is declared as  
-- VARCHAR2(30).  
-- Using %TYPE here protects us if the length changes in a future  
-- release.  
constraint\_name user\_constraints.constraint\_name%type;

# Renunirea restrictiilor de integritate

BEGIN

```
select constraint_name into constraint_name from user_constraints  
where table_name = upper('&table_name.') and constraint_type =  
'P';
```

```
dbms_output.put_line('The primary key for ' || upper('&table_name.') ||  
' is: ' || constraint_name);
```

```
execute immediate
```

```
'alter table &table_name. rename constraint ' || constraint_name ||  
' to &new_constraint.';
```

```
END;
```

```
/
```

# Restrictii de integritate

- Daca lista de coloane nu apare in cheia straina, atunci se presupune referinta la cheia primara.
- ```
CREATE TABLE Dept_tab ( Deptno NUMBER(3) PRIMARY KEY,  
Dname VARCHAR2(15), Loc VARCHAR2(15), CONSTRAINT  
Dname_ukey UNIQUE (Dname, Loc), CONSTRAINT  
LOC_CHECK1 CHECK (Loc IN ('NEW YORK', 'BOSTON',  
'CHICAGO')))  
/
```
- ```
CREATE TABLE Emp_tab ( Empno NUMBER(5) PRIMARY KEY,  
Ename VARCHAR2(15) NOT NULL, Job VARCHAR2(10), Mgr  
NUMBER(5) CONSTRAINT Mgr_fkey REFERENCES Emp_tab  
ON DELETE CASCADE, Hiredate DATE, Sal NUMBER(7,2),  
Comm NUMBER(5,2), Deptno NUMBER(3) NOT NULL,  
CONSTRAINT Dept_fkey REFERENCES Dept_tab)  
/
```

# Restrictii de integritate

- SELECT Constraint\_name, Constraint\_type, Table\_name, R\_constraint\_name FROM User\_constraints;

```
CONSTRAINT_NAME C TABLE_NAME R_CONSTRAINT_NAME
```

```

----- -
SYS_C00275      P DEPT_TAB
DNAME_UKEY      U DEPT_TAB
LOC_CHECK1      C DEPT_TAB
SYS_C00278      C EMP_TAB
SYS_C00279      C EMP_TAB
SYS_C00280      P EMP_TAB
MGR_FKEY        R EMP_TAB      SYS_C00280
DEPT_FKEY       R EMP_TAB      SYS_C00275

```

---

```
PRIMARY KEY - P
```

```
FOREIGN -R
```

```
UNIQUE -U
```

```
CHECK, NOT NULL -C
```

# Restrictii de integritate

- **SELECT Constraint\_name, Search\_condition FROM User\_constraints WHERE (Table\_name = 'DEPT\_TAB' OR Table\_name = 'EMP\_TAB') AND Constraint\_type = 'C';**
- **CONSTRAINT\_NAME SEARCH\_CONDITION**

-----

<b>LOC_CHECK1</b>	<b>loc IN ('NEW YORK', 'BOSTON', 'CHICAGO')</b>
<b>SYS_C00278</b>	<b>ENAME IS NOT NULL</b>
<b>SYS_C00279</b>	<b>DEPTNO IS NOT NULL</b>

**SELECT Constraint\_name, Table\_name, Column\_name FROM User\_cons\_columns;**

- **CONSTRAINT\_NAME TABLE\_NAME COLUMN\_NAME**

-----

<b>DEPT_FKEY</b>	<b>EMP_TAB</b>	<b>DEPTNO</b>
<b>DNAME_UKEY</b>	<b>DEPT_TAB</b>	<b>DNAME</b>
<b>DNAME_UKEY</b>	<b>DEPT_TAB</b>	<b>LOC</b>
<b>LOC_CHECK1</b>	<b>DEPT_TAB</b>	<b>LOC</b>
<b>MGR_FKEY</b>	<b>EMP_TAB</b>	<b>MGR</b>