

# Capitolul 5

## Gestiunea tabelelor

### - Partea 1. Crearea tabelelor -

# Tipuri de organizare

- Exista patru tipuri de organizare pentru tabelele unei baze de date:
  1. Tabele uzuale – heap-organized tables – este tipul de baza, uzual. O astfel de tabela reprezinta o multime neorganizata (heap) de linii. Acest tip de tabele este subiectul principal al capitolului de fata.
  2. Tabele partitionate – partitioned tables – in care liniile sunt impartite in mai multe grupuri, numite partitii, fiecare astfel de partitie (sau subpartitie) putand fi gestionata separat.

# Tipuri de organizare - cont

3. Tabele de tip cluster - 'clustered tables'. O astfel de tabela este parte a unui cluster. Un cluster contine mai multe tabele care au in comun blocuri de date - ele au in comun anumite coloane si, de asemenea, sunt folosite frecvent impreuna.
4. Tabele de tip index - 'index organized'. Spre deosebire de primele (heap organized), inregistrările (liniile) unei astfel de tabele sunt organizate sub forma unui arbore B, sortate dupa cheia primara.

# Tipuri de organizare - cont

5. Tabele externe - 'external tables'. O astfel de tabela nu se află în baza de date, ci în afara bazei de date, în fișiere externe, cum ar fi fișierele sistemului de operare sau fișierele de tip Hadoop Distributed File System (HDFS).
6. Tabele hibride partitionate - 'hybrid partitioned table'. O tabela hibrida partiționata este o tabela partiționata în care unele partiții se află în baza de date, iar altele în afara bazei de date, în fișiere externe, cum ar fi fișierele sistemului de operare sau fișierele Hadoop Distributed File System (HDFS). Au aparut in versiunea 19c.

# Tipuri de organizare - cont

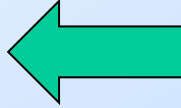
- ❑ In cazul tabelelor partitionate, impartirea liniilor in partitii se face dupa valoarea uneia sau mai multor coloane.
- ❑ O linie a tabeli poate sa apartina unei singure partitii
- ❑ Fiecare partitie are un nume si este stocata intr-un segment. Aceste segmente pot fi in tablespace-uri diferite
- ❑ Partitionarea se practica in cazul tabelelor de mari dimensiuni care sunt accesate concurent.
- ❑ Se pot partitiona si tabelele de tip 'index-organized' cu conditia ca attributele (coloanele) dupa care se face partitionarea sa fie o submultime a cheii primare.

# CREATE TABLE

- Pe langa elementele cunoscute din cursurile anterioare (BD an 3), cererea SQL **CREATE TABLE** poate avea si alte clauze, suplimentare, specificand parametrii de stocare pentru datele tabelului respective.

# Syntaxa CREATE TABLE (9i)

```
CREATE TABLE [schema.]table
  ( coloane_si_constrangeri ) - sintaxa clasica
  [ [PCTFREE integer] [PCTUSED integer]
    [INITRANS integer] [MAXTRANS integer]
    [TABLESPACE tablespace]
    [STORAGE storage_clause]
    [ PARALLEL [integer ] | NOPARALLEL ]
    [ CACHE | NOCACHE ]
    [ LOGGING | NOLOGGING ]
  |
  [CLUSTER cluster (column [, column]...)]
  ]
  [ ENABLE enable_clause | DISABLE disable_clause ] ...
  [AS subquery]
```



# Clauze CREATE – PCTFREE(1)

- ❑ PCTFREE specifica procentul de spatiu din fiecare bloc rezervat cresterii in lungime a inregistrarilor (liniilor) determinata de operatii de tip update.
- ❑ Valoarea trebuie sa fie intre 1 si 99.
- ❑ In cazul specificarii valorii 0 intregul bloc poate fi umplut prin inserare de noi linii.
- ❑ Valoarea implicita a acestui parametru este 10 (deci 10% spatiu disponibil pentru update, 90% spatiu disponibil pentru insert).



# Clauze CREATE – PCTFREE(2)

- ❑ In cazul in care inregistrarile dintr-un bloc cresc in lungime si se depaseste spatiul alocat lor (inclusiv cel initial retinut pentru crestere prin PCTFREE)  
Oracle ia o linie din acel bloc si o muta in alt bloc, lasand in locul ei doar un pointer.
- ❑ Acest proces este numit si 'migrarea liniilor'
- ❑ In acest caz performantele scad, deoarece pentru citirea acelei linii sunt citite doua blocuri.

# Clauze CREATE – PCTFREE(3)

- ❑ In cazul in care o inregistrare este prea lunga pentru a incapa intr-un bloc aceasta inregistrare este sparta in mai multe bucati care sunt stocate in mai multe blocuri, impreuna cu pointerii necesari recuperarii intregii linii. Aceasta situatie se numeste 'row chaining'.
- ❑ Si in acest caz performantele scad, deoarece pentru citirea acelei linii sunt citite mai multe blocuri.
- ❑ Parametrul PCTFREE poate fi prezent in comenzile create/alter si pentru alte obiecte (ex. indecsi).

# Clauze CREATE - PCTUSED

- ❑ PCTUSED specifica procentajul minim de spatiu utilizat din fiecare bloc. Daca spatiul utilizat scade sub acea valoare blocul devine candidat pentru inserarea de noi inregistrari (linii).
- ❑ PCTUSED are valori intre 1 si 99.
- ❑ Valoarea de default este 40
- ❑ Combinatia PCTFREE - PCTUSED duce la directionarea noilor inregistrari fie in blocuri existente fie in blocuri noi (goale la acel moment)

# Clauze CREATE - PCTUSED

- ❑ Acest parametru poate fi prezent si in comenzile de creare pentru alte obiecte (ex. Indecsi)
- ❑ Suma dintre PCTFREE si PCTUSED trebuie sa fie mai mica sau egala cu 100.
- ❑ Cu cat diferenta intre 100 si aceasta suma este mai mica, cu atat este mai eficienta folosirea spatiului pe disc, insa pot sa scada performantele.

# Clauze CREATE - INITRANS

- ❑ INITRANS specifica numarul initial de 'transaction entries' alocate in fiecare bloc. Fiecare tranzactie care actualizeaza un bloc are nevoie de o astfel de intrare la nivelul blocului.
- ❑ Valoarea poate fi de la 1 la 255
- ❑ Valoarea implicita este 1 in cazul tabelelor (2 la indecsi).
- ❑ In general Oracle recomanda sa se pastreze valoarea implicita
- ❑ Acest parametru asigura un numar minim de tranzactii per bloc fara overheadul alocarii dinamice a unei intrari ("transaction entry").

# Clauze CREATE - MAXTRANS

- ❑ MAXTRANS specifica numarul maxim de tranzactii concurente care pot actualiza un bloc al tabelii – deci numarul maxim de 'transaction entries' care pot fi alocate unui bloc.
- ❑ Acestea se alocă dinamic de Oracle după depășirea INITRANS.
- ❑ Valoarea de default este 255. Citat din documentația pentru 19c:
  - "This parameter has been deprecated. Oracle now automatically allows up to 255 concurrent update transactions for any data block, depending on the available space in the block."
- ❑ MAXTRANS este parametru și în alte operații de creare obiecte ale bazei de date.

# TABLESPACE, STORAGE

- ❑ TABLESPACE specifica unde se va crea tabela respectiva.
- ❑ Daca optiunea lipseste, tabela se creaza in tablespace-ul implicit (default) al userului care detine schema in care se face crearea.
- ❑ STORAGE specifica modul in care extensiile vor fi alocate tabelului.
- ❑ Sintaxa clauzei STORAGE a fost prezentata in capitolul anterior (cel despre tablespace-uri).
- ❑ Aceasta clauza are implicatii in performantele obtinute in cazul tabelurilor de mari dimensiuni.



# Reamintire:

Clauza Storage are optiuni ca:

- ☐ **INITIAL *int* K | M**
- ☐ **NEXT *int* K | M**
- ☐ **MINEXTENTS *int***
- ☐ **MAXEXTENTS *int***
- ☐ **MAXEXTENTS UNLIMITED**
- ☐ **PCTINCREASE *int***
- ☐ **FREELISTS *int***
- ☐ **FREELIST GROUPS *int***



# Reamintire:

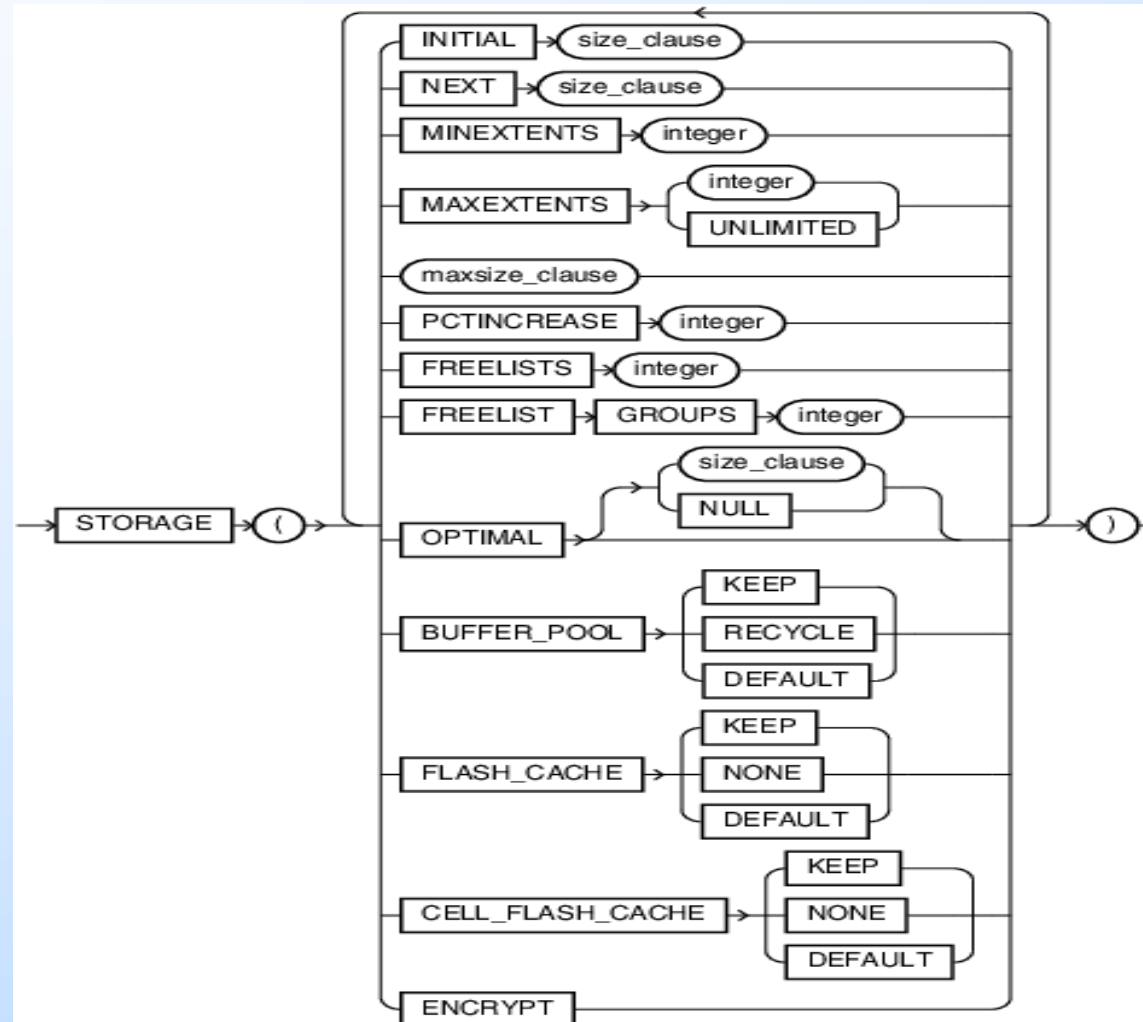
Unde:

- ❑ **INITIAL *int* K | M** – definește dimensiunea primei extensii (minim 2 blocuri). Valoarea implicită este 5 blocuri ale BD.
- ❑ **NEXT *int* K | M** – da dimensiunea celei de-a doua extensii. Valoarea minimă este de 1 bloc, valoarea implicită este de asemenea 5 blocuri.
- ❑ **MINEXTENTS *int*** - este numărul de extensii care sunt alocate când segmentul este creat. Valoarea minimă – și implicită – este 1.

# Reamintire:

- ❑ **MAXEXTENTS *int*** – determina numarul maxim de extensii pe care le poate avea un segment. Valoarea minima este 1 iar valoarea maxima depinde de dimensiunea blocului.
- ❑ **MAXEXTENTS UNLIMITED** – este echivalenta cu 2G extensii
- ❑ **PCTINCREASE *int*** – este procentul cu care creste dimensiunea extensiilor. Valoarea minima este 0, cea implicita 50.

# Sintaxa completa



# Exemplu:

```
create table tabela_mea (  
  2     nume      varchar2(30),  
  3     descriere  varchar2(4000) )  
  4     tablespace users  
  5     storage (  
  6       initial   1M  
  7       next      512K  
  8       pctincrease 0  
  9       minextents 2  
 10     maxextents unlimited )  
 11  /
```

Enter user-name: sys as sysdba

Enter password:

Connected to:

Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production

With the Partitioning, Oracle Label Security, OLAP and Data Mining Scoring Engine options

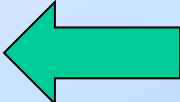
```
SQL> create table tabela_mea(  
  2  nume varchar2(20))  
  3  tablespace users  
  4  storage(  
  5  initial 1M  
  6  next 64K  
  7  pctincrease 0  
  8  minextents 2  
  9  maxextents unlimited) ;
```

Table created.

SQL>

# Syntaxa CREATE TABLE (9i)

```
CREATE TABLE [schema.]table
( coloane_si_constrangeri ) - sintaxa clasica
[ [PCTFREE integer] [PCTUSED integer]
  [INITRANS integer] [MAXTRANS integer]
  [TABLESPACE tablespace]
  [STORAGE storage_clause]
  [ PARALLEL [integer ] | NOPARALLEL ]
  [ CACHE | NOCACHE ]
  [ LOGGING | NOLOGGING ]
  |
  [CLUSTER cluster (column [, column]...)]
]
[ ENABLE enable_clause | DISABLE disable_clause ] ...
[AS subquery]
```



# Clauze CREATE: PARALLEL

- ❑ PARALLEL *int* specifica paralelizarea cererii de creare (de exemplu in cazul CREATE ... AS SELECT) sau gradul de paralelism pentru cereri DML - numarul de procese server care pot scana (parcure) in paralel tabela in cereri SELECT, INSERT, UPDATE, DELETE, MERGE.
- ❑ Se poate specifica: nimic (Oracle alege nr\_CPU x nr\_fire\_executie\_per\_CPU) sau un numar intreg.
- ❑ NOPARALLEL specifica faptul ca pe aceasta tabela cererile nu pot fi executate (in mod obisnuit) prin paralelizare – se pot folosi ‘hint’-uri pentru a forta executia paralela (ca in exemplul urmator).

# Clauze CREATE: PARALLEL - cont

- Cererea urmatoare specifica parcurgerea intregii tabele cu un grad de paralelism egal cu 5. Observam ca daca se defineste un alias de tabela hintul trebuie sa foloseasca acest alias:

```
SELECT /*+ FULL(s)
        PARALLEL(s, 5) */
ename
FROM emp s;
```



# Clauze CREATE: PARALLEL - cont

- ❑ Cererea urmatoare specifica parcurgerea tabelii fara paralelizare. De asemenea trebuie sa se foloseasca aliasul definit:

```
SELECT /*+ NOPARALLEL(s) */  
       ename  
FROM emp s;
```

# Clauze CREATE - CACHE

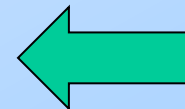
- ❑ CACHE se foloseste mai ales pentru tabele de mici dimensiuni si specifica faptul ca acea tabela va fi pastrata in buferele de memorie (deci nu va fi dealocata) prin plasarea blocurilor sale in zona celor mai recent utilizate chiar si atunci cand se executa o parcurgere completa a tablei.
- ❑ NOCACHE (valoare implicita) specifica faptul ca blocurile tablei din buffer cache se supun algoritmului LRU standard atunci cand se executa o parcurgere completa a tablei (full table scan), si deci se pun in zona celor mai putin recent utilizate.

# Clauze CREATE - LOGGING

- ❑ LOGGING arata ca atat operatia de creare a tabelii cat si operatiile care vor fi facuta apoi asupra acesteia vor fi inregistrate in fisierele Redo Log.
- ❑ NOLOGGING specifica faptul ca operatia de creare a tabelii precum si unele operatii de incarcare cu date (nu insa si operatiile obisnuite de insert) nu vor fi inregistrate in fisierele Redo Log.
- ❑ In lipsa acestor optiuni se folosesc parametrii de la crearea tablespace-ului in care este gazduita tabela (si acolo aveam aceste doua optiuni)

# Syntaxa CREATE TABLE (9i)

```
CREATE TABLE [schema.]table
( coloane_si_constrangeri ) - sintaxa clasica
[ [PCTFREE integer] [PCTUSED integer]
  [INITRANS integer] [MAXTRANS integer]
  [TABLESPACE tablespace]
  [STORAGE storage_clause]
  [ PARALLEL [integer ] |
    NOPARALLEL ]
  [ CACHE | NOCACHE ]
  [ LOGGING | NOLOGGING ]
  |
  [CLUSTER cluster (column [, column]...)]
]
[ ENABLE enable_clause | DISABLE disable_clause ] ...
[AS subquery]
```



# Clauze CREATE - CLUSTER

- ❑ CLUSTER arata ca tabela este parte a unui cluster.
- ❑ Coloanele din clauza sunt coloane ale tablei care corespund cu coloanele clusterului.
- ❑ In general coloanele respective ale tablei sunt parte a cheii primare (sau intreaga cheie primara).

# Clauze CREATE - CLUSTER

- ❑ Trebuie specificata cate o coloana a tabelului pentru fiecare coloana a clusterului.
- ❑ Corespondenta este pozitionala (nu prin nume)
- ❑ Deoarece tabelele de tip cluster folosesc o alta alocare a spatiului NU se pot folosi in paralel clauzele PCTFREE, PCTUSED, INITRANS, MAXTRANS, TABLESPACE in conjunctie cu clauza CLUSTER

# Exemplu

- ❑ 1. Creare cluster:

```
CREATE CLUSTER pers
```

```
(dept NUMBER(2))
```

```
SIZE 512
```

```
STORAGE (initial 100K next 50K);
```

- ❑ 2. Creare index pentru cheia clusterului:

```
CREATE INDEX idx_pers ON CLUSTER pers;
```

# Exemplu

□ 3. Adaugare tabele la cluster:

```
CREATE TABLE dept_10 CLUSTER pers (deptno)
AS SELECT * FROM scott.emp
WHERE deptno = 10;
```

```
CREATE TABLE dept_20 CLUSTER pers (deptno)
AS SELECT * FROM scott.emp
WHERE deptno = 20;
```



```
florin@rowlf:~  
SQL>  
SQL>  
SQL>  
SQL>  
SQL> CREATE CLUSTER pers (dept NUMBER(2)) SIZE 512 STORAGE (initial 100K next 50K)  
;  
  
Cluster created.  
  
SQL> CREATE INDEX idx_pers ON CLUSTER pers;  
  
Index created.  
  
SQL> CREATE TABLE dept_10 CLUSTER pers (deptno) AS SELECT * FROM scott.emp WHERE d  
eptno = 10;  
  
Table created.  
  
SQL> CREATE TABLE dept_20 CLUSTER pers (deptno) AS SELECT * FROM scott.emp WHERE d  
eptno = 20;  
  
Table created.  
  
SQL>
```

florin@rowlf:~

SQL>

SQL>

SQL>

SQL> select \*  
2 from pers  
3 -- eroare, e un cluster  
4 ;  
from pers  
  \*

ERROR at line 2:  
ORA-00942: table or view does not exist

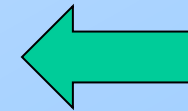
SQL> select \* from dept\_10;

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7839	KING	PRESIDENT		17-NOV-81	5000		10
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

SQL>

# Syntaxa CREATE TABLE (9i)

```
CREATE TABLE [schema.]table
( coloane_si_constrangeri ) - sintaxa clasica
[ [PCTFREE integer] [PCTUSED integer]
  [INITRANS integer] [MAXTRANS integer]
  [TABLESPACE tablespace]
  [STORAGE storage_clause]
  [ PARALLEL [integer ] |
    NOPARALLEL ]
  [ CACHE | NOCACHE ]
  [ LOGGING | NOLOGGING ]
  |
  [CLUSTER cluster (column [, column]...)]
]
[ ENABLE enable_clause | DISABLE disable_clause ] ...
[AS subquery]
```

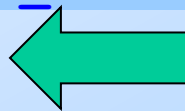


# ENABLE / DISABLE

- ❑ ENABLE si DISABLE activeaza / inhiba o constrangere de integritate.
- ❑ Aceste constrangeri sunt dintre cele create in aceeaasi comanda.
- ❑ In mod implicit, la creare, constrangerile de integritate sunt active (ENABLE)

# Syntaxa CREATE TABLE (9i)

```
CREATE TABLE [schema.]table
( coloane_si_constrangeri ) - sintaxa clasica
[ [PCTFREE integer] [PCTUSED integer]
  [INITTRANS integer] [MAXTRANS integer]
  [TABLESPACE tablespace]
  [STORAGE storage_clause]
  [ PARALLEL [integer ] |
    NOPARALLEL ]
  [ CACHE | NOCACHE ]
  [ LOGGING | NOLOGGING ]
  |
  [CLUSTER cluster (column [, column]...)]
]
[ ENABLE enable_clause | DISABLE disable_clause ] ...
[AS subquery]
```



# Clauze CREATE - AS

- ❑ AS specifica faptul ca noua tabela va fi populata cu liniile rezultate din cererea select prezenta in clauza AS.
- ❑ Crearea unei tabele ca rezultat al unei cereri SELECT a fost studiata in semestrele trecute.

# CREATE .. TEMPORARY

- ❑ Se pot crea tabele temporare cu cereri de tipul: `CREATE GLOBAL TEMPORARY TABLE`
- ❑ Definitia acestor tabele este vizibila tuturor sesiunilor active
- ❑ Datele din aceste tabele sunt vizibile doar sesiunii care le insereaza (doar una la un moment dat)

# CREATE .. TEMPORARY

- Liniile din tabela se sterg la sfarsitul fiecărei tranzactii sau la sfarsitul sesiunii, dupa cum se specifica in clauza ON COMMIT:
- ON COMMIT DELETE ROWS – liniile se sterg la sfarsitul fiecărei tranzactii
- ON COMMIT PRESERVE ROWS – se sterg la sfarsit de sesiune.



# Exemplu

```
CREATE GLOBAL TEMPORARY TABLE  
test_temp (nume VARCHAR2(20)) ON  
COMMIT DELETE ROWS;
```

- ❑ Se va crea in acest caz o tabela temporara in care liniile se sterg la sfarsitul fiecărei tranzactii.

```
florin@rowlf:~
SQL> CREATE GLOBAL TEMPORARY TABLE test_temp (nume VARCHAR2(20)) ON COMMIT DE
LETE ROWS;

Table created.

SQL> INSERT INTO test_temp VALUES('ION');

1 row created.

SQL> SELECT * FROM test_temp;

NUME
-----
ION

SQL> COMMIT;

Commit complete.

SQL> SELECT * FROM test_temp;

no rows selected

SQL>
```

# Copierea unei tabele

- ❑ Folosind `CREATE TABLE .. AS SELECT` se poate copia o tabela schimbându-i, la momentul copierii, anumiti parametrii
- ❑ In acest caz se pot specifica noi nume ale coloanelor, noi parametrii de stocare fizica, etc.
- ❑ Atentie: nu sunt copiate in acest caz si constrangerile de integritate (cu exceptia coloanelor definite cu `NOT NULL`, care vor fi la fel si in noua tabela.

# Lecturi obligatorii

1. Oracle Database Administrator's Guide (v19c) – Cap 20: Managing Tables.

<https://docs.oracle.com/en/database/oracle/oracle-database/19/admin/>

2. Sintaxa cereri Oracle de la adresa:

<http://www4.utc.fr/~nf17/DOCS/complement/sqlplus-ref/>

3. On shrinking table sizes:

<http://www.adp-gmbh.ch/blog/2005/july/20.html>

# Sfârșit creare tabele