Capitolul 5

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

Tipuri de organizare - cont

- In cazul tabelelor partitionate, impartirea liniilor in partitii se face dupa valoarea uneia sau mai multor coloane.
- O linie a tabelei 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 atributele (coloanele) dupa care se face partitionarea sa fie o submultime a cheii primare.

CREATE TABLE

- Pe langa elementele cunoscute din cursurile anterioare, cererea SQL CREATE TABLE poate avea si alte clauze, suplimentare, specificand parametrii de stocare pentru datele tabelei 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 acestul 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 incapea 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

- Combinatia PCTFREE - PCTUSED duce la directionarea noilor inregistrari fie in blocuri existente fie in blocuri noi (goale la acel moment)
- 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
- 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 decat 100.

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 tabelei – deci numarul maxim de 'tranzaction entries' care pot fi alocate unui bloc.
- Acestea se aloca dinamic de Oracle dupa depasirea INITRANS.
- Valoarea poate fi intre 1 si 255.
- Valoarea de default este in functie de dimensiunea blocului (255 in Oracle 8i)
- Este recomandat ca valoarea de default pentru MAXTRANS sa nu fie schimbata.
- MAXTRANS este parametru si in alte operatii 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 tabelei.
- Sintaxa clauzei STORAGE a fost prezentata in capitolul anterior (cel despre tablespace-uri).
- Aceasta clauza are implicatii in performantele obtinute in cazul tabelelor 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** – defineste dimensiunea primei extensii (minim 2 blocuri). Valoarea implicita este 5 blocuri ale BD.
- **NEXT** ***int*** **K | M** – da dimensiunea celei de-a doua extensii. Valoarea minima este de 1 bloc, valoarea implicita este de asemenea 5 blocuri.
- **MINEXTENTS** ***int*** - este numarul de extensii care sunt alocate cand segmentul este creat. Valoarea minima – si implicita – 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.

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    /
```

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 (parcurge) 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).

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;
```

PARALLEL - cont

- Cererea urmatoare specifica parcurgerea tabelei 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 tabelei.
- NOCACHE (valoare implicita) specifica faptul ca blocurile tabelei din buffer cache se supun algoritmului LRU standard atunci cand se executa o parcurgere completa a tabelei (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 tabelei 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 tabelei 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)

Clauze CREATE - CLUSTER

- CLUSTER arata ca tabela este parte a unui cluster.
- Coloanele din clauza sunt coloane ale tabelei care corespund cu coloanele clusterului.
- In general coloanele respective ale tabelei sunt parte a cheii primare (sau intreaga cheie primara).
- Trebuie specificata cate o coloana a tabelei 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 personnel
(department NUMBER(4))
SIZE 512
STORAGE (initial 100K next 50K);
```

- 2. Creare index pentru cheia cluster:

```
CREATE INDEX idx_personnel ON
  CLUSTER personnel;
```

Exemplu

- 3. Adaugare tabele la cluster

```
CREATE TABLE dept_10
CLUSTER personnel (department_id)
AS SELECT * FROM employees
WHERE department_id = 10;

CREATE TABLE dept_20
CLUSTER personnel (department_id)
AS SELECT * FROM employees
WHERE department_id = 20;
```

ENABLE / DISABLE

- ENABLE si DISABLE activeaza / inhiba o constrangere de integritate.
- Aceste constrangeri sunt dintre cele create in aceeasi comanda.
- In mod implicit, la creare, Oracle activeaza o constrangere de integritate

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 CREATE GLOBAL TEMPORARY TABLE
- Definitia tabelelor este vizibila tuturor sesiunilor active
- Datele din aceste tabele sunt vizibile doar sesiunii care le insereaza (doar una la un moment dat)
- Liniile din tabela se sterg la sfarsitul fiecarei tranzactii sau la sfarsitul sesiunii, dupa cum se specifica in clauza ON COMMIT:
  - ON COMMIT DELETE ROWS – liniile se sterg la sfarsitul fiecarei tranzactii
  - ON COMMIT PRESERVE ROWS – se sterg la sfarsit de sesiune.

Exemplu

```
CREATE GLOBAL TEMPORARY TABLE
    admin_work_area
(startdate DATE,
 enddate DATE,
 class CHAR(20))
ON COMMIT DELETE ROWS;
```

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

Copierea unei tabele

- Se poate copia o tabela schimbandu-i la momentul copierii anumiti parametrii folosind CREATE TABLE .. AS SELECT
- 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.

Comanda ALTER TABLE

```
ALTER TABLE [schema.]table
  [optiuni ADD, MODIFY, etc - prezentate anul trecut]
  [PCTFREE integer] [PCTUSED integer]
  [INITRANS integer] [MAXTRANS integer]
  [STORAGE storage_clause]
  [DROP drop_clause] ...
  [ALLOCATE EXTENT [( [SIZE integer [K|M] ]
                     [DATAFILE 'filename']
                     [INSTANCE integer] )]
  [ PARALLEL { integer } |
    NOPARALLEL ]
  [ CACHE | NOCACHE ]
  [{ ENABLE | DISABLE } { enable_clause | TABLE LOCK }]
  [{ ENABLE | DISABLE } ALL TRIGGERS }]
```

ALTER TABLE - ALLOCATE

- ALLOCATE EXTENT aloca explicit o noua extensie pentru acea tabela.
- SIZE specifica dimensiunea extensiei in octeti (bytes). Se poate folosi K ori M pentru a specifica KB sau MB. In cazul absentei acestei clauze Oracle determina dimensiunea pe baza valorilor de STORAGE ale tabelei.
- DATAFILE specifica numele fisierului (aferent tablespace-ului in care se gaseste tabela) in care se va aloca noua extensie. In lipsa alegerea este facuta de Oracle.

ALTER TABLE - ALLOCATE

- INSTANCE face acea extensie disponibila pentru instanta specificata. Numarul instantei e dat de parametrul de initializare INSTANCE_NUMBER). In lipsa extensia va fi disponibila pentru toate instantele. Acest parametru este util in conjunctie cu Parallel Server.
- Alocarea explicita a unei extensii afecteaza dimensiunea urmatoarei extensii care va fi alocata pe baza parametrilor NEXT si PCTINCREASE (prezentati in capitolul anterior)

ALTER TABLE - cont

PARALLEL

NOPARALLEL

CACHE

NOCACHE

ENABLE

DISABLE

- Sunt folosite pentru a schimba setarile curente.
- Aceste clauze au fost prezentate la CREATE TABLE

ALTER TABLE - LOCK

- ENABLE TABLE LOCK - Activeaza posibilitatea obtinerii de blocari asupra tabelei de catre comenzi DDL.
- Aceste comenzi nu se pot executa daca blocarea nu este permisa.
- DISABLE TABLE LOCKS duce implicit la interzicerea operatiilor DDL asupra tabelei.

ALTER TABLE - TRIGGERS

ENABLE ALL TRIGGERS

DISABLE ALL TRIGERS

- Permit activarea / dezactivarea tuturor declansatorilor asociati unei tabele
- Pentru activarea / dezactivarea unui singur declansator se poate folosi comanda ALTER TRIGGER.

- Clauza DROP – specifica stergerea unei constrangeri de integritate.

Comanda ALTER TABLE - cont

- PCTFREE,
- PCTUSED,
- INITRANS,
- MAXTRANS,
- STORAGE

Schimba valorile si optiunile care exista la acel moment. Explicatia semnificatiei acestor parametri s-a facut la descrierea CREATE TABLE

Parametrii de stocare

- In cazul lui ALTER TABLE, semnificatia noilor valori ale acestor parametri este urmatoarea:
- NEXT – In momentul alocarii unei noi extensii Oracle va folosi noua valoare a lui NEXT dupa care cresterea se face pornind de la aceasta valoare si cea a lui PCTINCREASE (vezi formula din capitolul precedent pentru dimensiunea extensiei n)

Parametrii de stocare - cont

- PCTINCREASE – de asemenea schimbarea acestuia va afecta dimensiunea noilor extensii care se aloca.
- MINEXTENTS – poate fi schimbata cu orice valoare care e mai mica sau egala cu numarul de extensii pe care le are tabela la acel moment
- MAXEXTENTS – poate fi schimbata cu orice valoare mai mare sau egala cu numarul de extensii ale tabelei la acel moment

Parametri utilizare bloc

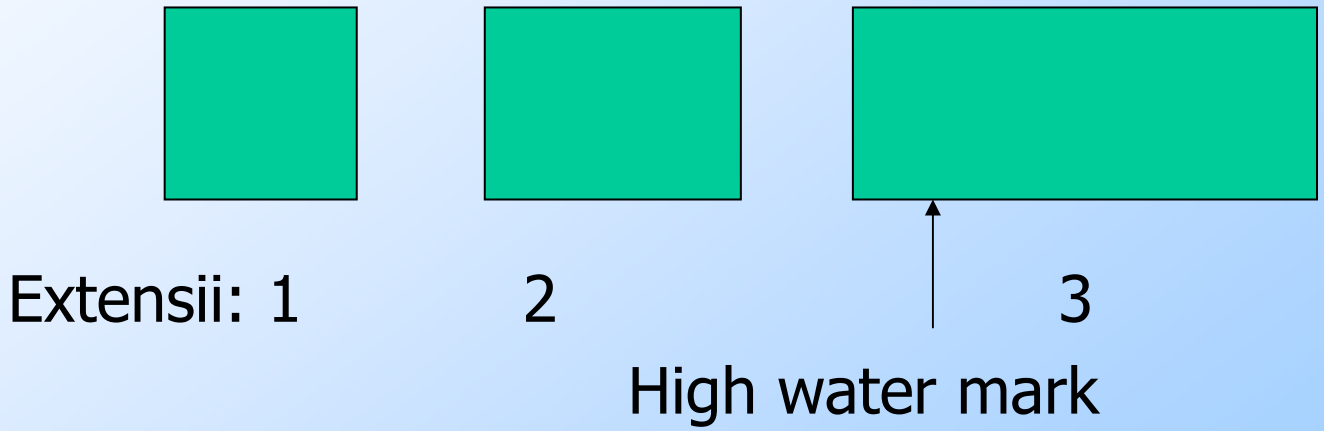
- PCTFREE – schimbarea sa afecteaza urmatoarele operatii de inserare. Blocurile ‘umplute’ dupa vechea valoare nu sunt afectate decat in momentul in care ele ajung in lista de blocuri cu spatiu liber – FREELIST, deci de blocuri in care se pot face operatii de inserare.
- Un bloc ajunge in lista de blocuri cu spatiu liber doar daca se efectueaza stergeri din el pana sub PCTUSED

Parametri utilizare bloc - cont

- PCTUSED – orice modificare a acestui parametru afecteaza toate blocurile din tabela. Daca o linie e actualizata sau stearsa blocul care o contine va fi testat daca poate fi pus in lista de blocuri cu spatiu liber.
- INITRANS – schimbarea acestui parametru va afecta doar blocurile noi
- MAXTRANS – schimbarea acestui parametru va afecta toate blocurile tabelei (pentru ca diferenta de la INITRANS pana la MAXTRANS sunt ‘transaction entries’ care se aloca dinamic.

High water mark

- Pentru orice segment (inclusiv deci pentru segmentele continand tabele) exista un marcaj al ultimului bloc care a fost vreodata utilizat.
- Acest marcaj se numeste ‘high water mark’ (HWM)



High water mark - cont

- Pe masura ce datele sunt inserate in tabela HWM este mutat spre blocuri superioare
- Acest marcaj NU este resetat in momentul in care sunt sterse linii din tabela (resetarea se face doar in cazul in care se executa TRUNCATE)
- Cand Oracle face o parcurgere completa a unei tabele (full table scan) atunci sunt citite toate blocurile pana la HWM, deci inclusiv blocuri golite ca urmare a stergerilor de inregistrari (linii).

High water mark - cont

- Daca se doreste insa 'defragmentarea' tabelei se poate executa ALTER TABLE MOVE care muta o tabela dintr-un tablespace in altul (cele doua nu sunt neaparat distincte):

```
ALTER TABLE emp MOVE tblspace2
```

- In acest caz se pastreaza definitiile tuturor constrangerilor de integritate si al indecsilor, dar acestia din urma trebuiesc refacuti (indecsii sunt bazati pe ROWID iar in procesul de compactare acesta se schimba).

Exemplu

- Creare tabela si umplere cu date:

```
create table table_size_test (
 a char(100), b number )
storage (initial 65K next 65K pctincrease 0)
tablespace ts_01;
begin
for i in 1 .. 10000 loop
  insert into table_size_test values
   (dbms_random.string('X', 100),i);
end loop;
end;
/
commit;
```

Exemplu - cont

- Crearea unui index:

```
create index ix_table_size_test on table_size_test(a)
storage (initial 65K next 65K pctincrease 0)
tablespace ts_02;
```

- Vizualizare spatiu utilizat:

```
select substr(segment_name,1,20) segment,
       bytes / 1024 "Size [KB]"
from user_segments
where segment_name in ('TABLE_SIZE_TEST',
     'IX_TABLE_SIZE_TEST');

SEGMENT              Size [KB]
-------------------- ----------
TABLE_SIZE_TEST            1280
IX_TABLE_SIZE_TEST         1280
```

Exemplu - cont

- Stergere din tabela

```
delete from table_size_test where mod(b,2)=0;

commit;
```

- Vizualizare spatiu (rezultat)

```
SEGMENT                Size [KB]
---------------------- ----------
TABLE_SIZE_TEST            1280
IX_TABLE_SIZE_TEST         1280
```

- Alter table move

```
alter table table_size_test move;
```

- Vizualizare spatiu (rezultat)

```
SEGMENT                Size [KB]
---------------------- ----------
TABLE_SIZE_TEST             640
IX_TABLE_SIZE_TEST         1280
```

Exemplu - cont

- Indexul insa a devenit UNUSABLE:

```
select status from user_indexes
where index_name = 'IX_TABLE_SIZE_TEST';
STATUS
--------
UNUSABLE
```

- Il refacem

```
alter index ix_table_size_test rebuild;
```

- Date despre indexul refacut (acum a devenit valid):

```
select status, bytes/1024
from user_indexes
join user_segments on index_name = segment_name
where index_name = 'IX_TABLE_SIZE_TEST';
STATUS   BYTES/1024
-------- ----------
VALID           704
```

High water mark - cont

- Incepand cu Oracle 10 se mai poate face ajustarea HWM in cazul segmentelor care utilizeaza ASSM – Automatic Segment Space Management
- In acest caz putem face ajustarea astfel:
  - Permitem schimbarea ROWID-ului liniilor:

```
ALTER TABLE emp ENABLE ROW MOVEMENT;
```

  - Dam comanda de shrink:

```
ALTER TABLE emp SHRINK SPACE;
```

- Efectul comenzii de shrink este: muta liniile compactandu-le si muta HWM. Pentru asta e nevoie de o blocare a tabelei dar pentru o perioada scurta de timp.

High water mark - cont

- Variante ale comenzii:

1. Muta linii si HWM intr-o tabela:

```
ALTER TABLE emp SHRINK SPACE;
```

2. Muta linii si HWM intr-o tabela si compacteaza si obiectele dependente:

```
ALTER TABLE emp SHRINK SPACE CASCADE;
```

3. Muta linii doar liniile fara sa mute HWM:

```
ALTER TABLE emp SHRINK SPACE COMPACT;
```

High water mark - cont

Restrictii pentru SHRINK:

- Doar in tablespace-uri cu ASSM
- Nu se pot compacta (lista e mai lunga):
  - Segmente UNDO
  - Segmente temporare
  - Tabele de tip cluster
  - Tabele cu o coloana de tip LONG
  - Indecsi de tip LOB
- Se poate utiliza pachetul de sistem DBMS_SPACE pentru a vedea informatii despre spatiul utilizat.

DEALOCARE SPATIU LIBER

- Spatiul liber ocupat de un segment (cel de dupa HWM) poate fi dealocat folosind:

```
ALTER TABLE [schema.]tabela
DEALLOCATE UNUSED [KEEP int [K | M] ]
```

- In cazul folosirii KEEP se pastreaza o parte a acestui spatiu liber (dimensiunea e data in bytes, KB sau MB).
- Spatiul astfel dealocat poate fi folosit de alte segmente.

DEALOCARE SPATIU - cont

- In cazul in care HWM este intr-o extensie cu numar mai mic decat MINEXTENTS se dealoca toate extensiile de dupa MINEXTENTS.
- Pentru a dealoca tot spatiul disponibil (pana la HWM) inclusiv in cazul in care HWM e sub MINEXTENTS se foloseste KEEP 0.

Trunchiere

- Comanda de trunchiere goleste o tabela si reseteaza HWM.
- Spatiul ocupat de tabela este dealocat (vezi slide urmator) in afara cazului cand se specifica explicit REUSE STORAGE
- Sintaxa comenzii este:

```
TRUNCATE TABLE [schema.]tabela
[{ DROP | REUSE } STORAGE]
```

- Comanda TRUNCATE e o comanda DDL deci este comisa automat si nu se poate face rollback (nu poate fi anulata ca in cazul unui DELETE)

TRUNCATE - DROP

- In cazul DROP:
  - Sunt dealocate toate extensiile superioare lui MINEXTENTS
  - HWM e resetat
  - Valoarea lui NEXT_EXTENT este resetata la valoarea extensiei cu numarul cel mai mic care a fost dealocata
- In ambele cazuri (REUSE sau DROP), trunchierea afecteaza toti indecsii tabelei respective.

DROP TABLE

- Stergerea unei tabele se face cu DROP TABLE
- Sintaxa este:

```
DROP [schema.]tabela
[CASCADE CONSTRAINTS]
```

- Efectul este stergerea tabelei si a tuturor constrangerilor de integritate aferente (inclusiv cele referentiale)
- Daca nu se specifica CASCADE tabela nu se poate sterge daca exista constrangeri referentiale care o refera.

Validare structura

- Se face cu comanda ANALYZE TABLE
- Aceasta colecteaza statistici despre tabela si le stocheaza in dictionarul de date
- Printre alte optiuni sunt si cele de:
  - Validare a structurii unei tabele
  - Identificarea liniilor care au migrat sau sunt inlantuite
- In cazul validarii structurii, toate blocurile tabelei sunt verificate din punct de vedere al integritatii

VALIDATE STRUCTURE - cont

- Sintaxa este

  ANALYZE TABLE [schema.]tabela

  VALIDATE STRUCTURE [CASCADE]

- In cazul folosirii optiunii CASCADE este validata si structura tuturor indecsilor asociati tabelei si se face si o verificare incrucisata intre continutul de date al tabelei si al indecsilor respectivi.

Migrare si inlantuire

- ANALYZE TABLE poate fi folosita si pentru detectarea liniilor care au migrat sau a celor inlantuite (din cauza lui PCTUSED sau pt. ca sunt prea voluminoase).
- Pentru aceasta intai se calculeaza sau se estimeaza statisticile asupra tabelei respective.
- Statisticile estimate se fac pe baza unui esantion de 1064 linii (valoare implicita).

Migrare si inlantuire - cont

- Sintaxa comenzii in acest caz este:

```
ANALYZE TABLE [schema.]tabela
{ COMPUTE STATISTICS
| ESTIMATE STATISTICS
    [SAMPLE integer { ROWS | PERCENT }]
```

- COMPUTE va genera statistici pornind de la o parcurgere completa a tabelei
- La ESTIMATE se poate specifica (in linii sau in procente) dimensiunea esantionului

Migrare si inlantuire - cont

- Dupa generarea statisticilor, in vederea de dictionar DBA_TABLES exista in coloana CHAIN_CNT numarul de linii care sunt migrate sau inlantuite.
- In cazul in care un numar mare de linii sunt in aceasta situatie trebuie ca tabela sa fie reorganizata pentru a remedia aceasta situatie (de exemplu prin recrearea tabelei folosind CREATE ... AS SELECT ... ORDER BY)

VEDERI

- Pe langa DBA_TABLES se mai pot folosi si DBA_OBJECTS si DBA_SEGMENTS.
- Toate cele 3 tabele pot fi unite (join) dupa conditia compusa:

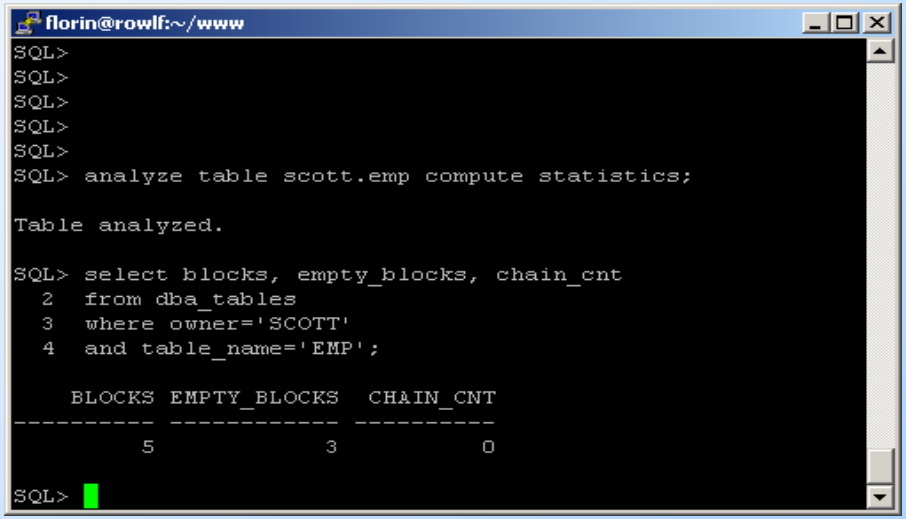
DBA_TABLES.OWNER =
DBA_OBJECTS.OWNER=
DBA_SEGMENTS.OWNER

AND

DBA_TABLES.TABLE_NAME =
DBA_OBJECTS.OBJECT_NAME=
DBA_SEGMENTS.SEGMENT_NAME

Exemplu

```
SELECT BLOCKS,
EMPTY_BLOCKS, CHAIN_CNT
FROM DBA_TABLES
WHERE OWNER = 'SCOTT'
AND TABLE_NAME = 'EMP';
```



- Obtinem in acest caz un rezultat continand:
  - Prima coloana contine HWM (ce este acela?)
  - A doua numarul de blocuri de dupa HWM
  - A treia numarul de linii (inregistrari) migrate sau inlantuite

DBA_EXTENTS

- Aceasta vedere poate fi folosita pentru a afla numarul de extensii si alte informatii despre ele.
- Printre coloanele vederii sunt:
  - OWNER
  - SEGMENT_NAME
  - EXTENT_ID
  - FILE_ID
  - BLOCK_ID
  - BLOCKS
- Fiecare linie reprezinta o extensie si in BLOCKS este numarul de blocuri ale acesteia.

Lecturi obligatorii

1. Oracle Database Administrator's Guide – Cap 14: Managing Tables (versiunile 10g si 11g):

   http://download.oracle.com/docs/cd/B14117_01/server.101/b10739/tables.htm
   http://download.oracle.com/docs/cd/B28359_01/server.111/b28310/tables.htm
   http://download.oracle.com/docs/cd/B28359_01/server.111/b28286/statements_7002.htm

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

Sfârşitul capitolului 5