

# **UTILIZAREA BAZELOR DE DATE**

**CURS: FLORIN RADULESCU**

**Email: florin.radulescu@cs.pub.ro**

**SITE: CS.CURS.PUB.RO**

# NOTARE

- ◆ **60% IN CURSUL SEMESTRULUI:**
  - ◆ PREZENTA CURS **10%**
  - ◆ PREZENTA, ACTIVITATE SI TEST LABORATOR **35%**
  - ◆ LUCRARE LA MIJLOCUL SEMESTRULUI FARĂ DEGREVARĂ (SAPTAȚAMAÑA 8-9): **15%.** ACEASTA LUCRARE **NU SE POATE** REFACE IN SESIUNEA DE EXAMENE
- ◆ **40% VERIFICARE FINALA (EXAMEN)**

# REGULI DE TRECERE

- ◆ 50% din punctajul din timpul semestrului (30 pct. din 60)

ŞI

- ◆ 50% din punctajul de la examen (20 pct. din 40)

# Nelamuriri?

# DRAFT CONTINUT

- ◆ Administrarea bazelor de date relationale (Oracle)
- ◆ Notiuni de Data Mining

# BIBLIOGRAFIE

- ◆ Va fi indicata la fiecare capitol
- ◆ Pentru fiecare capitol, la bibliografie exista documente care aprofundeaza ceea ce s-a predat la curs si care **fac parte integranta** din materia pentru lucrarea de la mijlocul semestrului si din cea pentru examen.

# Incepem?

# Administrarea bazelor de date

## DEFINITII

Definitii care se pot gasi in Internet:

- ◆ A technical function that is responsible for
  - ◆ physical database design
  - ◆ security enforcement,
  - ◆ database performance,
  - ◆ backup and recovery.

([http://wps.prenhall.com/wps/media/objects/1374/1407508/Glossary\\_Terms.html](http://wps.prenhall.com/wps/media/objects/1374/1407508/Glossary_Terms.html))

# DEFINITII (2)

- ◆ An area of IT that
    - ◆ develops,
    - ◆ implements,
    - ◆ updates,
    - ◆ tests, and
    - ◆ repairs
- a company's server database.

(<http://www.nvcc.edu/home/lfeist/Class%203/Key%20Terms%20-%20Class%203.doc>)

# DEFINITII (3)

- ◆ Database Administration involves the overall design and management of the database. Administration tasks include
  - ◆ archiving,
  - ◆ consistency checks,
  - ◆ developing/maintaining indexing and retrieval functionality,
  - ◆ migration,
  - ◆ monitoring,
  - ◆ performance issues,
  - ◆ replication issues, and
  - ◆ database sizing/space management.

([http://it.toolbox.com/wiki/index.php/Database\\_Administration](http://it.toolbox.com/wiki/index.php/Database_Administration))

# JOB PROFILE

- ◆ A database administrator (DBA) is an IT professional responsible for the integrity, performance and security of databases in an organization.
- ◆ The role includes the development and design of database strategies, system monitoring and improving database performance and capacity, and planning for future expansion requirements.
- ◆ They may also plan, co-ordinate and implement security measures to safeguard the database

(sursa: wikipedia)

# JOB PROFILE - cont

## Duties:

- ◆ Installing and upgrading the database server and application tools
- ◆ Allocating system storage and planning future storage requirements for the database system
- ◆ Modifying the database structure, as necessary, from information given by application developers
- ◆ Enrolling users and maintaining system security
- ◆ Ensuring compliance with database vendor license agreement
- ◆ Controlling and monitoring user access to the database
- ◆ Monitoring and optimizing the performance of the database
- ◆ Planning for backup and recovery of database information
- ◆ Maintaining archived data
- ◆ Backing up and restoring databases
- ◆ Contacting database vendor for technical support
- ◆ Generating various reports by querying from database as per need

(sursa: wikipedia)

# CU CE INCEPEM?

## Administrare ORACLE

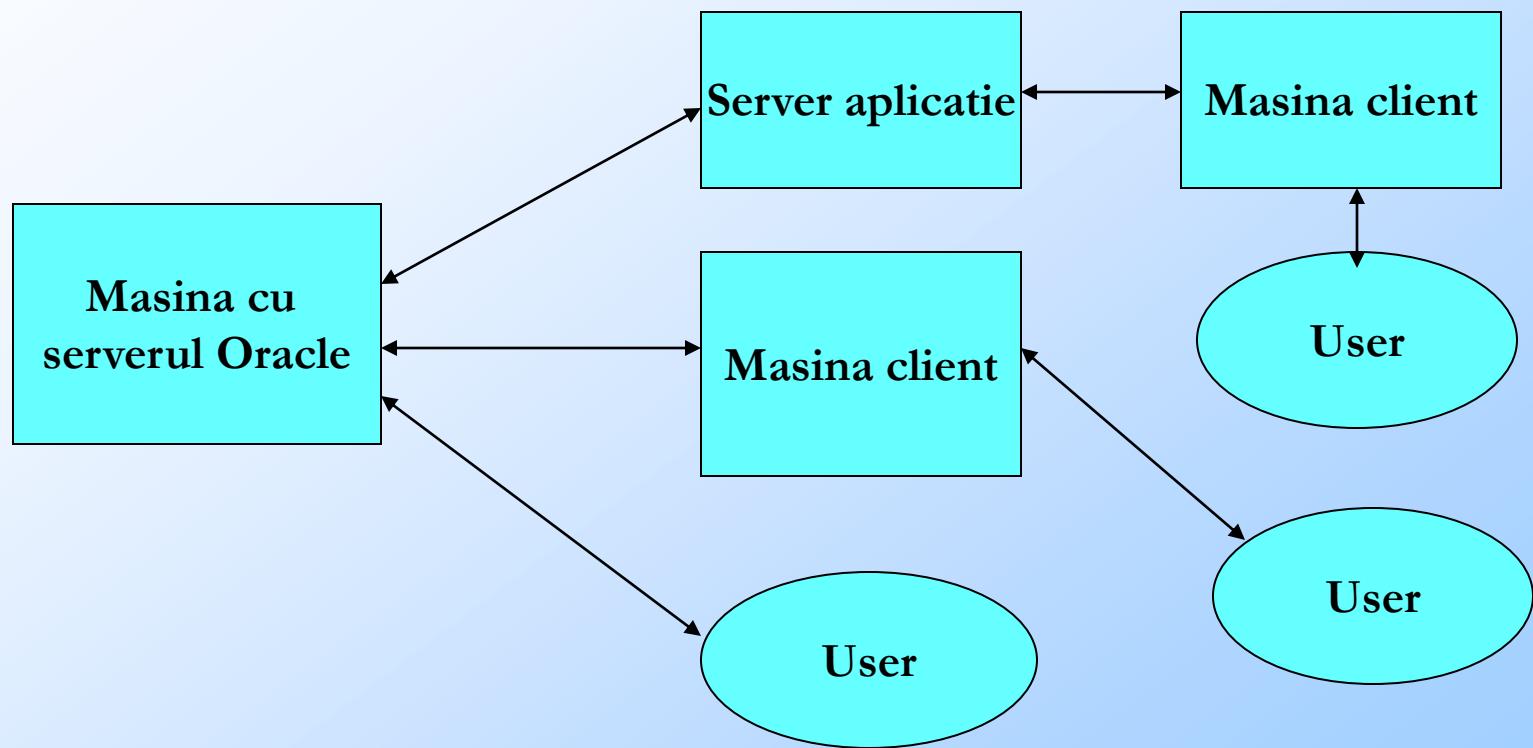
# Capitolul 1

## Arhitectura Oracle

# Serverul ORACLE

- ◆ Este un sistem de gestiune a bazelor de date relationale
- ◆ Userul poate lucra:
  - ◆ Cu un client pe aceeasi masina cu serverul (de exemplu un client SQL\*Plus ruland pe aceeasi masina cu serverul Oracle)
  - ◆ Clientul ruleaza pe o alta masina decat serverul (two-tiered)
  - ◆ Aplicatia userului acceseaza o alta aplicatie iar la randul ei aceasta e in comunicatie cu serverul (three-tiered)

# Serverul ORACLE



# USER vs UTILIZATOR

- ◆ Vom numi în cele ce urmează user un cont de utilizator Oracle (exemplu: user 'student' cu parola 'stud1234')
- ◆ Utilizator este o persoană care prin intermediul unui cont de user interacționează cu Oracle
- ◆ Utilizator poate fi numit într-un sens mai larg și un proces, o aplicație care folosește un cont user pentru a interacționa cu Oracle

# PROCESE

- ◆ Cand userul acceseaza o aplicatie care lucreaza cu Oracle sunt create 2 procese:
  - ◆ Un “Proces user” (client) pe masina pe care lucreaza utilizatorul. Acesta interactioneaza cu utilizatorul si asigura comunicatia cu cel de-al doilea proces
  - ◆ Un “Proces server”, pe masina unde este instalat serverul Oracle. Acesta comunica cu serverul Oracle in numele procesului user.

# SESIUNE

- ◆ O sesiune de lucru reprezinta o conexiune a unui user cu serverul Oracle.
- ◆ Un user poate avea mai multe sesiuni deschise simultan:
  - ◆ Pe aceeasi baza de date
  - ◆ Cu acelasi cont de user
  - ◆ Pe aceeasi masina sau pe masini diferite

# PROCESUL USER

- ◆ Este creat atunci cand un utilizator incepe o sesiune de lucru folosind fie un client care interactioneaza direct cu Oracle (SQL\*Plus) fie o unealta Oracle (Server Manager, Developer, etc)
- ◆ Ruleaza pe masina utilizatorului
- ◆ Asigura interfata - grafica uneori - pentru acesta (GUI, UPI – User Program Interface)
- ◆ Procesul se termina cand utilizatoruliese din programul folosit
- ◆ Trimit comenzi/cereri utilizatorului catre procesul server si afiseaza rezultatele (date, stare, eventualele mesaje de eroare)

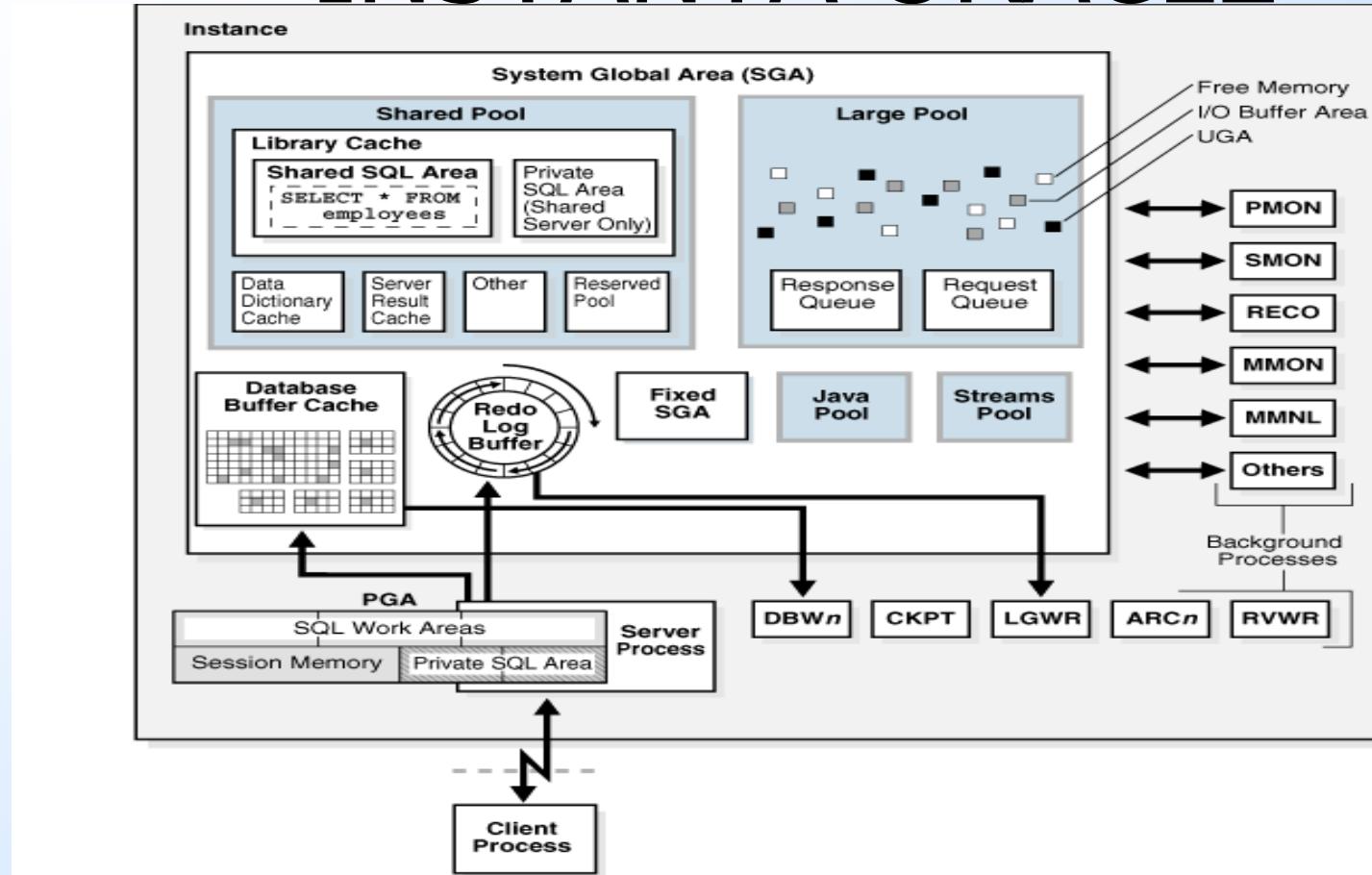
# PROCESUL SERVER

- ◆ Ruleaza pe masina unde este instalat serverul Oracle
- ◆ Deserveste un singur proces user sau mai multe procese user, in functie de configuratie: Dedicated Server sau Multithreaded server)
- ◆ Foloseste o zona de memorie (PGA – Program Global Area) care va fi descrisa in continuare.
- ◆ Este un intermediar intre procesul user si serverul Oracle, folosind OPI – Oracle Program Interface – pentru a interactiona cu acesta
- ◆ Procesul se incheie cand utilizatorul termina sesiunea.

# INSTANTA ORACLE

- ◆ Un server Oracle constă dintr-o instanță Oracle și o bază de date Oracle.
- ◆ Instanța Oracle este compusă dintr-o structură de date în memorie numita SGA – System Global Area – și procese rulate în background care sunt utilizate de Oracle pentru gestiunea bazei de date.
- ◆ O instanță Oracle deschide o singură bază de date.
- ◆ Este identificată prin ORACLE\_SID (variabilă la nivel de SO). SID = System ID

# INSTANTA ORACLE



Sursa:

[http://docs.oracle.com/cd/E11882\\_01/server.112/e25789/process.htm](http://docs.oracle.com/cd/E11882_01/server.112/e25789/process.htm)

# SGA – System Global Area

- ◆ Este alocata in memoria virtuala a sistemului pe care ruleaza serverul Oracle.
- ◆ Contine date si informatii de control
- ◆ Este de tip partajat - shared – mai multi useri pot folosi datele de aici pentru a se evita accesul repetat la disc
- ◆ Contine mai multe componente dintre care cele mai importante sunt:
  - ◆ Buffer Cache
  - ◆ Redo Log Buffer
  - ◆ Shared Pool

# SGA – System Global Area -cont

## ◆ Mai există de asemenea:

- ◆ Java Pool – utilizată pentru cod și date specifice Java de către Java Virtual Machine (JVM)
- ◆ Large Pool – optional, pentru date de mari dimensiuni.
- ◆ Streams Pool – folosită de produsul Oracle Streams

# Procese BACKGROUND

Sunt de trei tipuri:

A. Obligatorii - apar in toate configuratiile uzuale:

- ◆ **Database Writer (DBWn)** – Este responsabil cu scrierea blocurilor de date modificate/inserate din bufferele de memorie in fisierele de pe disc. In Oracle 10g pot fi maximum 20 de procese de acest fel
- ◆ **Log Writer (LGWR)** – Scrie datele din Redo Log Buffer pe disc. Scrierea se face secvential intr-un fisier de Redo Log.

# Procese BACKGROUND - cont

## A. Obligatorii - continuare

- ◆ **Checkpoint** (CKPT) – Acest proces scrie periodic pe disc toate blocurile de date (din buffere) care au fost modificate. O astfel de actiune este denumita *checkpoint*.
- ◆ Procesul anunta actiunea lui DBWR, actualizeaza fisierele de date si control ale bazei de date si inregistreaza momentul in care s-a facut checkpointul.

# Procese BACKGROUND - cont

## A. Obligatorii - continuare

- ◆ **System Monitor (SMON)** - Face verificarea consistentei datelor si initiaza recuperarea dupa incident atunci cand o instanta Oracle care a avut un incident reporneste.
- ◆ **Process Monitor (PMON)** - Dealoca resursele unui proces care are un incident. Folosit pentru procesele user care au incidente.

# Procese BACKGROUND - cont

## A.Obligatorii - continuare

### ◆ Manageability Monitor Processes :

Manageability monitor process (MMON) - efectueaza operatii legate de Automatic Workload Repository (AWR). Acesta contine date istorice despre performantele sistemului: statistici pentru sistem, sesiuni, cereri SQL individuale si servicii. E folosit pentru tuning.

Manageability monitor lite process (MMNL) - scrie statisticile din Active Session History (ASH - buffer in zona SGA) pe disc atunci cand acest buffer se umple.

# Procese BACKGROUND - cont

## A.Obligatorii - continuare

- ◆ **Recoverer Process (RECO)** - acesta este folosit in cazul bazelor de date distribuite pentru rezolvarea incidentelor aparute la tranzactiile distribuite.

# Procese BACKGROUND - cont

## B. Optionale

- ◆ **Archiver** (ARCn) –Copiaza fisierele Redo Log in arhiva de pe disc atunci cand acestea sunt pline sau cand acestea se schimba. Actiunea are loc in anumite conditii. Pot fi mai multe procese de acest fel.

# Procese BACKGROUND - cont

## B. Optionale - continuare

- ◆ **Job Queue Processes**: sunt folosite mai ales la lucrul pe loturi pentru executarea unui job la un moment dat sau repetat. Există:
  - ◆ Job queue coordinator process (CJQ0)
  - ◆ Job queue slave process (Jnnn) - pot fi mai multe
- ◆ **Flashback Data Archiver Process (FBDA)** - pastrează copii ale liniilor modificate în anumite tabele
- ◆ **Space management Coordinator Process (SMCO)** - pentru gestiunea spațiului

# Procese BACKGROUND - cont

## C. Procese 'slave'

- ◆ Sunt lansate de celelalte procese pentru a efectua diverse actiuni.

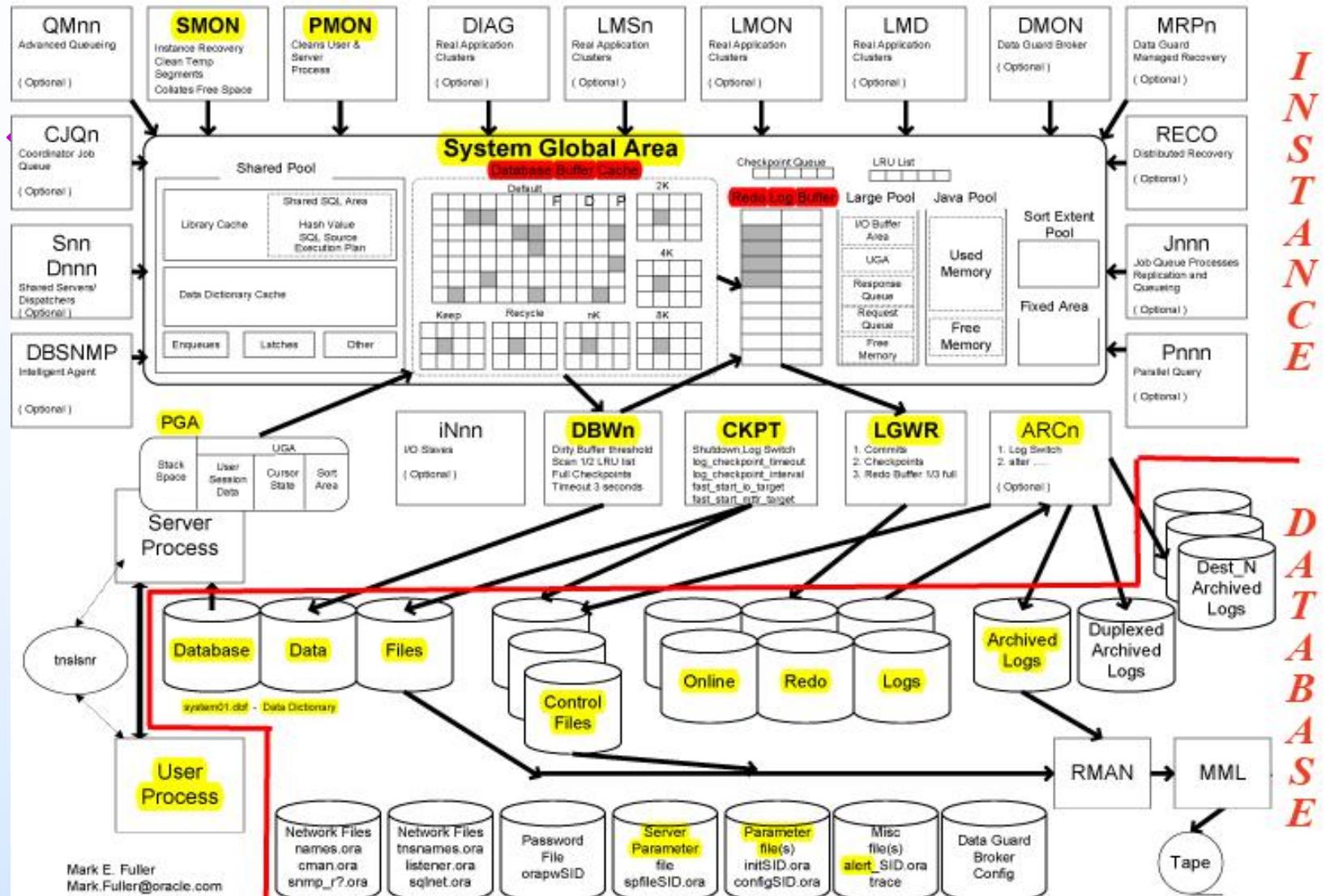
Nota: O lista completa a proceselor de background poate fi gasita de exemplu la adresa:

[http://docs.oracle.com/cd/E11882\\_01/server.112/e40402/bgprocesses.htm](http://docs.oracle.com/cd/E11882_01/server.112/e40402/bgprocesses.htm)

# Baza de date

- ◆ A doua componenta a serverului (pe langa instanta) este **Baza de date**
- ◆ Este identificata prin DB\_NAME (variabila SO)
- ◆ Oracle recomanda ca instanta si BD sa aiba acelasi nume (pot fi si diferite) pentru usurinta administrarii
- ◆ Este compusa din fisiere de mai multe tipuri

# ORACLE ARCHITECTURE



# Tipuri de fisiere

- ◆ **Data files** – fisiere de date. Fisierile de date contin:
  - ◆ Dictionarul de date al BD (mai stiti ce este acesta?),
  - ◆ Obiectele userului (mai stiti care sunt acestea?)
  - ◆ Contin si vechile valori ale datelor modificate de tranzactiile curente ('before image')
- ◆ O baza de date are cel putin un astfel de fisier.

# Tipuri de fisiere - cont

- ◆ **Redo Log Files** – Fisiere Redo Log.  
Contin modificarile facute in baza de date. Sunt necesare la reconstructia ei in caz de incident.
- ◆ Fiecare baza de date contine cel putin 2 astfel de fisiere scrise in paralel si care pot fi tinute pe dispozitive de stocare diferite, prevenind pierderi de date in cazul in care un disc este distrus.

# Tipuri de fisiere - cont

- ◆ **Control files** – fisiere cu date de control.  
Contin informatii necesare pastrarii integritatii bazei de date
- ◆ Fiecare baza de date are cel putin un astfel de fisier.

# Alte fisiere

- ◆ Pe langa fisierile bazei de date, Oracle mai utilizeaza si alte fisiere, ca de exemplu:
  - ◆ **Fisier de parametri**: contine parametri care characterizeaza instanta Oracle
  - ◆ **Fisier de parole**: utilizat pentru autentificarea userilor
  - ◆ **Fisiere Redo Log arhivate**: copii offline ale fisierelor Redo Log.

# Etape de procesare cerere

◆ Există mai multe etape parcuse de o cerere de la emiterea ei de către utilizator până la execuția completă.

Exemplu pentru o cerere SELECT:

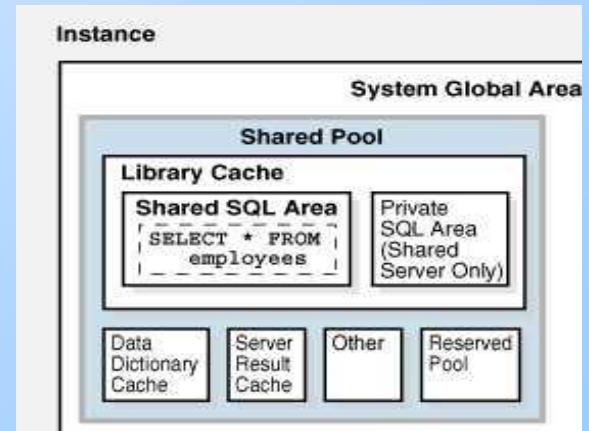
- ◆ Pasul 1: **Parse**: cererea primită de la user este analizată sintactic și semantic (compilată). Serverul întoarce o informație de stare (ok sau eroare). Shared Pool este folosită ca zona de memorie pentru aceasta operatie

# Etape de procesare cerere - cont

- ◆ Pasul 2: **Execute**. Cererea este executata si datele din tabela rezultat sunt pregatite pentru a fi returnate userului.
- ◆ Pasul 3: **Fetch**. Liniile returnate de cerere sunt trimise userului (procesului user) pentru a fi procesate acolo (afisare sau procesare). In functie de dimensiunea datelor returnate se pot executa una sau mai multe operatii de tip FETCH

# SHARED POOL

- ◆ Este parte a SGA. Este utilizata si in pasul 1 (parse) de executie a cererii.
- ◆ Dimensiunea sa e data de parametrul SHARED\_POOL\_SIZE (din fisierul de parametrii).
- ◆ Pentru Pasul 1 sunt utilize urmatoarele componente ale Shared Pool:
  - ◆ Library cache
  - ◆ Data dictionary cache



# Library cache

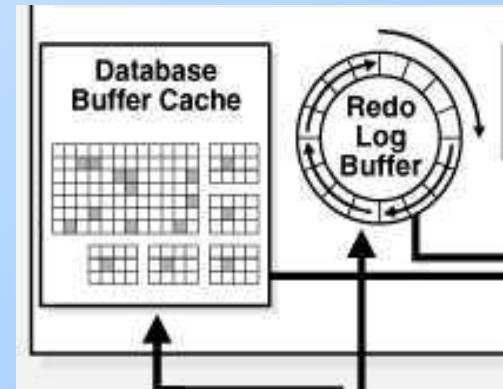
- ◆ Stocarea informatiei despre cele mai recente cereri SQL utilizeaza:
  - ◆ Textul cererii
  - ◆ Arboarele cererii rezultat in urma etapei Parse (analiza semantica). Aceasta este versiunea compilata a textului cererii
  - ◆ Planul de executie al cererii rezultat in urma optimizarii.
- ◆ Daca o cerere este re-executata pana sa apar alte cereri care sa afecteze planul de executie etapa Parse nu mai este necesara la re-executie (se foloseste rezultatul existent).

# Data Dictionary Cache

- ◆ Contine cele mai recent folosite informatii din dictionarul de date:
  - ◆ Descrieri table si coloane
  - ◆ Date cont user
  - ◆ Drepturi (privilegii)
  - ◆ Etc.
- ◆ In etapa Parse acest cache e folosit de procesul server pentru informatiile necesare compilarii cererii (numele folosite in cererea analizata). Daca nu exista sunt incarcate din fisierele de pe disc.

# Database Buffer Cache

- ◆ Parte a SGA. Tine cele mai recent utilizate blocuri de date.
- ◆ Cand o cerere este executata procesul server verifica aici existenta blocurilor necesare. Daca nu exista le citeste si le plaseaza aici.
- ◆ Dimensiunea sa este data de parametrul **DB\_BLOCK\_BUFFERS**
- ◆ Dimensiunea unui bloc e data de parametrul **DB\_BLOCK\_SIZE**.



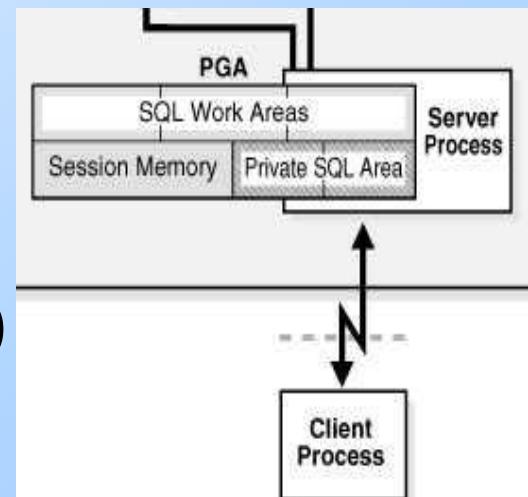
# Database Buffer Cache – cont.

- ◆ Oracle utilizeaza algoritmul LRU (ce este acesta?) pentru eliberare pozitii in buffer,
- ◆ Exceptie: operatiile de tip 'full table scan' (cand se parurge o intreaga tabela – deci este citita in memorie). In acest caz ultimele blocuri citite pot fi primele dealocate.
- ◆ Algoritmii utilizati sunt complecsi, cele de mai sus sunt o prezentare schematica a strategiilor de gestiune a bufferului.

(vezi de exemplu <http://www.adp-gmbh.ch/ora/concepts/cache.html>)

# Program Global Area (PGA)

- ◆ Zona de memorie folosita in mod exclusiv de un proces (de tip server sau background). Nu este comuna ca in cazul SGA. Ea contine:
  - ◆ Sort area – zona sortari, folosita la operatiile de ordonare (sortare).
  - ◆ Informatii sesiune, de exemplu drepturile userului acelei sesiuni.
  - ◆ Starea cursorilor folositi in sesiunea respectiva (daca exista)
  - ◆ Stiva, continand diverse variabile de sesiune.



# Exemplu: procesare UPDATE

- ◆ Sa luam exemplul unei cereri de actualizare:

```
UPDATE EMP  
SET SAL = SAL * 1.1  
WHERE EMPNO=1123
```

- ◆ Se executa intai Pasul PARSE
- ◆ La etapa EXECUTE (Pasul 2) se efectueaza operatiile:

# Exemplu: procesare UPDATE – cont

1. Procesul server citeste blocurile de date si de rollback – valori anterioare modificarilor necomise - din fisierele de date (daca nu sunt deja in Buffer Cache)
2. Copii ale blocurilor sunt puse in Buffer Cache (daca nu existau).
3. Procesul server blocheaza datele respective.
4. Procesul server inregistreaza in Redo Log Buffer schimbarile de facut in rollback si in date

# Exemplu: procesare UPDATE – cont

5. Procesul server plaseaza versiunile originale ale blocurilor (nemodificate) in Rollback si modifica blocurile de date. Ambele operatii se fac in Buffer Cache. Ambele blocuri (rollback si date) sunt marcate ca 'dirty blocks' (blocuri modificate), adica blocuri care nu sunt identice cu cele de pe disc.

# Segment de Rollback

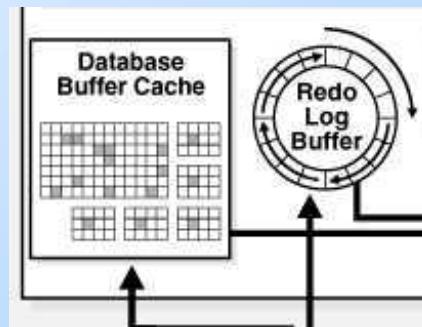
- ◆ Înainte de a se face schimbari, serverul salvează vechile valori de bloc într-un segment de rollback.
- ◆ Aceasta salvare permite anularea tranzacției (operatia ROLLBACK, opusa lui COMMIT), asigură ca alte tranzacții pot citi valorile anterioare începutului de tranzacție (read consistency – mai stiti ce era asta?) și ne permit de asemenea recuperarea după incident.

# Segment de Rollback - cont

- ◆ Segmentele de rollback sunt stocate in fisierele de date ale bazei de date si sunt aduse in buffer cache la cerere (cand este nevoie de ele).

# Redo Log Buffer

- ◆ Procesul server inregistreaza schimbarile facute de o instanta in Redo Log Buffer.
- ◆ Are o dimensiune data de parametrul **LOG\_BUFFER** (in bytes).
- ◆ Contine inregistrari Redo: blocul care a fost schimbat, pozitia schimbării, noua valoare.



# Redo Log Buffer - cont

- ◆ Contine toate schimbarile, atat in date cat si in blocuri de index, rollback, etc.
- ◆ Scrierea acestor inregistrari este sequentiala
- ◆ Contine inregistrari privind schimbarile facute de toate tranzactiile.
- ◆ Este folosit prin parcurgere circulara. Cand un bloc e refolosit, el este scris anterior in fisierele de pe disc.

# Database Writer

- ◆ Este un proces de tip background
- ◆ Scrie blocurile modificate - 'dirty blocks'  
- din Buffer Cache in fisierele de date,  
astfel incat sa existe suficiente blocuri  
libere care sa fie folosite de sistem
- ◆ A fost necesar pentru a degreva  
procesul server de aceasta operatie –  
imbunatatirea performantelor

# Database Writer - cont

## ◆ Scrierea se face cand:

- ◆ Numarul de 'dirty blocks' in buffer depaseste o anumita valoare
- ◆ Un proces care cauta locuri libere in buffer nu le gaseste
- ◆ Timeout (la fiecare N secunde)
- ◆ Evenimente care forteaza un checkpoint:  
exemplu: inchiderea bazei de date.

# Log Writer

- ◆ Este un proces de tip background
- ◆ Scrie intrari din Redo Log Buffer in fisierele de pe disc.
- ◆ Operatia se efectueaza cand:
  - ◆ Redo Log Buffer e aproape plin
  - ◆ Timeout (ca mai sus)
  - ◆ Inainte ca DBWR sa scrie blocurile modificate pe disc
  - ◆ Cand o tranzactie e comisa

# COMMIT

- ◆ Oracle utilizeaza un mecanism rapid de commit care garanteaza recuperarea schimbarilor comise in caz de incident.
- ◆ Oracle asigneaza un “commit System Change Number” (SCN - este de tip timestamp) fiecarei tranzactii care se comite. Aceste numere sunt crescatoare si unice la nivelul bazei de date

# COMMIT - Pasii

## ◆ La aparitia unui COMMIT:

1. Procesul server plaseaza o inregistrare de commit, impreuna cu SCN-ul acesteia in Redo Log Buffer
2. LGWR scrie o portiune contigua de inregistrari din buffer pana la cea care contine COMMIT-ul in fisierele Redo Log de pe disc. In felul acesta este garantata recuperarea dupa incident

# COMMIT - Pasii

3. Userul este informat ca s-a efectuat COMMIT-ul.
4. Procesul server inregistreaza ca tranzactia e completa si ca resursele blocate de ea pot fi eliberate.

Deci:

- ◆ Scrierea efectiva a datelor pe disc este efectuata independent de DBWR.
- ◆ Dimensiunea tranzactiei nu conteaza

# Sumar

1. Server Oracle = Instanta + Baza de date.
2. Baza de date = Ansamblu de fisiere de diverse tipuri: de date, de redo log, de control, de parametri, samd.
3. Instanta = Zona de memorie SGA + Procese de background
4. SGA contine: Buffer cache, Redo log buffer, Shared Pool si altele; e comună tuturor proceselor server.
5. Procese de background = 3 categorii fiecare continand mai multe tipuri, putând fi uneori mai multe procese de același tip.

# Sumar - continuare

6. Printre procesele de background avem:
  1. DBWR - Database writer: scrie date din Buffer cache pe disc; sunt blocuri de date si de rollback.
  2. LGWR – Log writer: scrie inregistrari din Redo log buffer pe disc; ele contin modificarile efectuate in date.
7. La momentul unui COMMIT nu se scriu efectiv datele pe disc (DBWR) ci doar inregistrarile de redo log (LGWR). Datele vor fi scrise ulterior de DBWR.

# Bibliografie generală

1. Colin McGregor - Oracle Database 2 Day DBA, 11g

[http://docs.oracle.com/cd/E11882\\_01/server.112/e10897.pdf](http://docs.oracle.com/cd/E11882_01/server.112/e10897.pdf)

2. Ulrike Schwinn, Vijayanandan Venkatachalam – Database administration

# Lecturi obligatorii

## 1. Colin McGregor - Oracle Database 2 Day DBA, 11g

Link: [http://docs.oracle.com/cd/E11882\\_01/server.112/e10897.pdf](http://docs.oracle.com/cd/E11882_01/server.112/e10897.pdf)

- ◆ Capitolul 5 (Managing the Oracle Instance)  
– integral
- ◆ Capitolul 6 (Managing Database Storage Structure) pag. 6-1 pana la 6-5 (fara Tablespaces)

# Sfârșitul primului capitol

# Capitolul 2

## Instanta si baza de date

# DBA

- ◆ Există doi utilizatori privilegiati care sunt creati inca de la instalarea Oracle (se cere doar parola pentru ei la instalare):
  1. SYS – proprietarul (owner) bazei de date precum si al tuturor tabelelor si vederilor din dictionarul bazei de date. Are rol de DBA
    - ◆ SYS are privilegiul SYSDBA - vom vedea ce e asta
    - ◆ Atentie: Nu creati/modificati niciodata obiecte in schema SYS (oare ce e o schema?)

## DBA - cont

2. SYSTEM – are de asemenea rol de DBA. Este proprietarul (owner) celorlalte tabele si vederi de sistem Oracle, altele decat cele din dictionarul de date (ex: cele folosite de uneltele Oracle)
  - ◆ Este bine sa nu creati obiecte in schema SYSTEM

# DBA vs SYSDBA

- ◆ DBA este un rol care contine majoritatea privilegiilor (drepturilor) de system – de tipul “root” din Unix
- ◆ SYSDBA este un privilegiu de sistem
- ◆ DBA nu contine totusi doua privilegii importante: SYSDBA si SYSOPER
- ◆ Acestea sunt privilegii importante care permit administratorului sa execute o serie de operatii de administrare.

# SYSDBA

Poate efectua operatiile:

- ◆ STARTUP si SHUTDOWN
- ◆ ALTER DATABASE: open, mount, back up, sau schimbarea setului de caractere
- ◆ CREATE DATABASE
- ◆ DROP DATABASE
- ◆ CREATE SPFILE
- ◆ ALTER DATABASE ARCHIVELOG
- ◆ ALTER DATABASE RECOVER
- ◆ Include privilegiul RESTRICTED SESSION

# SYSOPER

Poate efectua operatiile:

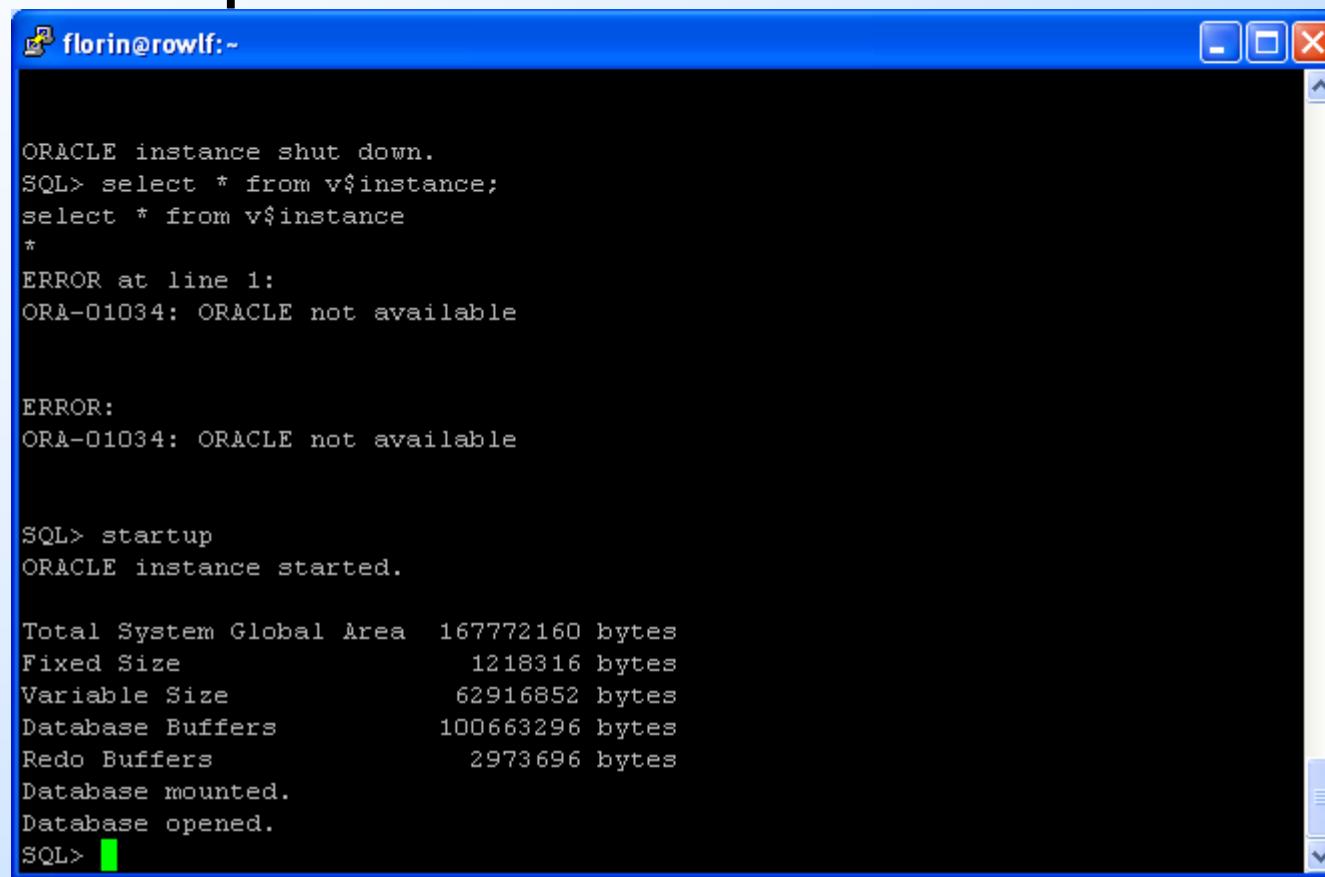
- ◆ STARTUP si SHUTDOWN
- ◆ ALTER DATABASE OPEN/MOUNT/BACKUP
- ◆ CREATE SPFILE
- ◆ ALTER DATABASE ARCHIVELOG
- ◆ ALTER DATABASE RECOVER (doar restaurare completa. Restaurarea incompleta - de tip UNTIL TIME | CHANGE| CANCEL| CONTROLFILE necesita privilegiul SYSDBA)
- ◆ Include privilegiul RESTRICTED SESSION

# Etapele pornirii unei BD

1. Pornirea (start) instantă
2. Montarea bazei de date (Mount)
3. Deschiderea bazei de date (Open)

# Etapele pornirii

## ◆ Exemplu:



The screenshot shows a terminal window with a blue title bar containing the text "florin@rowlf:~". The window content is a black terminal session. It starts with the message "ORACLE instance shut down.". Then, the user enters SQL\*Plus commands: "SQL> select \* from v\$instance;" followed by "select \* from v\$instance \*". Both commands result in an error: "ERROR at line 1: ORA-01034: ORACLE not available". Below this, there is another section labeled "ERROR:" with the same error message. Finally, the user runs the "startup" command, which outputs "ORACLE instance started." and provides memory statistics for the SGA and redo log buffers. The session ends with "Database mounted.", "Database opened.", and the prompt "SQL>".

```
florin@rowlf:~  
ORACLE instance shut down.  
SQL> select * from v$instance;  
select * from v$instance  
*  
ERROR at line 1:  
ORA-01034: ORACLE not available  
  
ERROR:  
ORA-01034: ORACLE not available  
  
SQL> startup  
ORACLE instance started.  
  
Total System Global Area 167772160 bytes  
Fixed Size 1218316 bytes  
Variable Size 62916852 bytes  
Database Buffers 100663296 bytes  
Redo Buffers 2973696 bytes  
Database mounted.  
Database opened.  
SQL>
```

# Etapele pornirii unei BD

- ◆ La pornirea instantei Oracle foloseste un fisier de parametri (init<SID>.ora) care este un fisier text.
- ◆ Dupa eventuale modificari, instanta trebuie oprita si repornita pentru a citi noile valori.

# Exemplu de continut

- ◆ db\_name=ORE
- ◆ db\_files = 80
- ◆ db\_block\_size = 8192
- ◆ db\_block\_buffers = 100
- ◆ shared\_pool\_size = 3500000
- ◆ log\_checkpoint\_interval = 10000
- ◆ log\_buffer = 32768
- ◆ log\_files = 10
- ◆ processes = 50
- ◆ max\_dump\_file\_size = 10240
- ◆ background\_dump\_dest = (/home/disk1/BDUMP)
- ◆ user\_dump\_dest = (/home/disk1/UDUMP)
- ◆ rollback\_segments = (r01, r02)
- ◆ control\_files = (ora\_control1, ora\_control2)
- ◆ compatible = 8.0.0

# Pornirea instantei

- ◆ Dupa momentul pornirii instantei (fara montarea si deschiderea bazei de date) se pot executa operatiile:
  - ◆ Crearea bazei de date
  - ◆ Recrearea fisierelor de control
- ◆ Pornirea instantei presupune:
  - ◆ Citirea fisierului de parametri init<SID>.ora
  - ◆ Alocarea SGA
  - ◆ Pornirea proceselor de background
  - ◆ Deschiderea fisierelor de tip TRACE si ALERT

# Montarea BD

- ◆ In momentul in care instanta este pornita si baza de date montata (dar nu deschisa) se pot executa operatii de mentenanta ca:
  - ◆ Redenumirea fisierelor bazei de date (Data files)
  - ◆ Activare/dezactivare arhivare fisiere Redo Log
  - ◆ Restaurarea bazei de date

# Montarea BD - cont

- ◆ Montarea bazei de date presupune:
  - ◆ Asocierea unei baze de date cu o instanta deja pornita
  - ◆ Localizarea si deschiderea fisierelor de control specificate in fisierul de parametri
  - ◆ Citirea fisierelor de control pentru cunoasterea numelui fisierelor de date si de Redo log (fara a verifica existenta lor fizica)

# Deschiderea BD

- ◆ Dupa deschiderea BD se poate opera normal cu baza de date. Userii se pot acum conecta si trimite cereri.
- ◆ Deschiderea presupune:
  - ◆ Deschiderea fisierelor de date
  - ◆ Deschiderea fisierelor Redo log.
  - ◆ Verificarea consistentei bazei de date.  
Daca este necesar, procesul SMON face o recuperare dupa incident.

# Deschiderea BD - cont

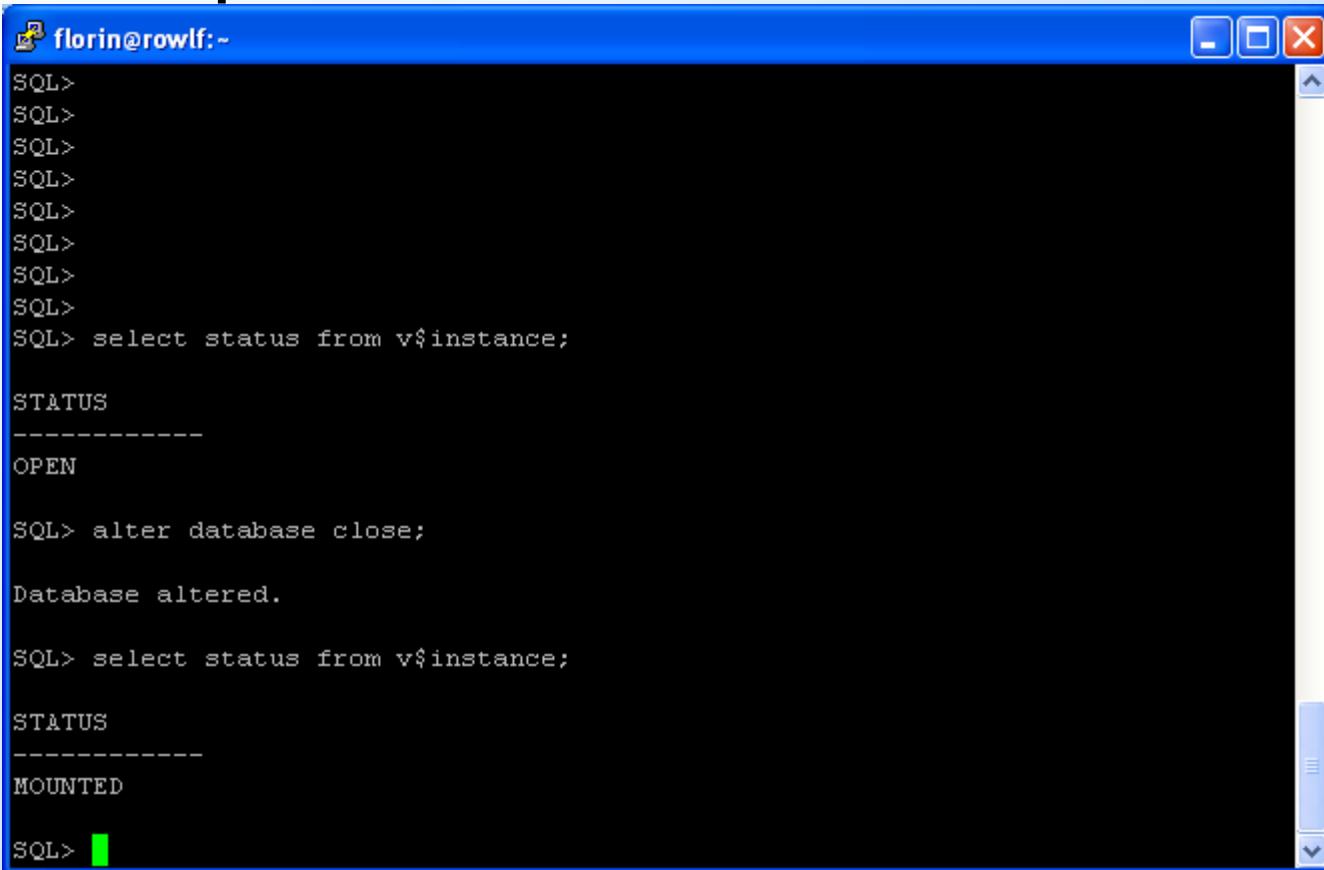
- ◆ Situațiile în care se face recuperarea după incident sunt aceleia în care instanta nu a reusit să efectueze toate operațiile (de exemplu în caz de crash de sistem).
- ◆ Recuperare presupune actualizarea fisierelor de date pe baza modificărilor din fisierele Redo log (care sunt actualizate la fiecare COMMIT, deci efectele tuturor tranzacțiilor încheiate cu succes sunt înregistrate aici).

# Etapele opririi BD

- ◆ Sunt cele de la pornire, in ordine inversa:
  - ◆ Inchidere BD
  - ◆ Demontare BD
  - ◆ Oprire instanta
- ◆ La inchiderea BD Oracle scrie pe disc blocurile modificate din Buffer cache si inregistrarile din Redo log buffer dupa care inchide fisierele de date si Redo log
- ◆ Fisierele de control sunt inchise la demontarea bazei de date.
- ◆ Dealocarea SGA si oprirea proceselor de background se fac la oprirea instantei.

# Etapele opririi BD - cont

- ◆ Exemplu:



```
florin@rowlf:~
```

```
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL> select status from v$instance;
```

```
STATUS
-----
OPEN
```

```
SQL> alter database close;
```

```
Database altered.
```

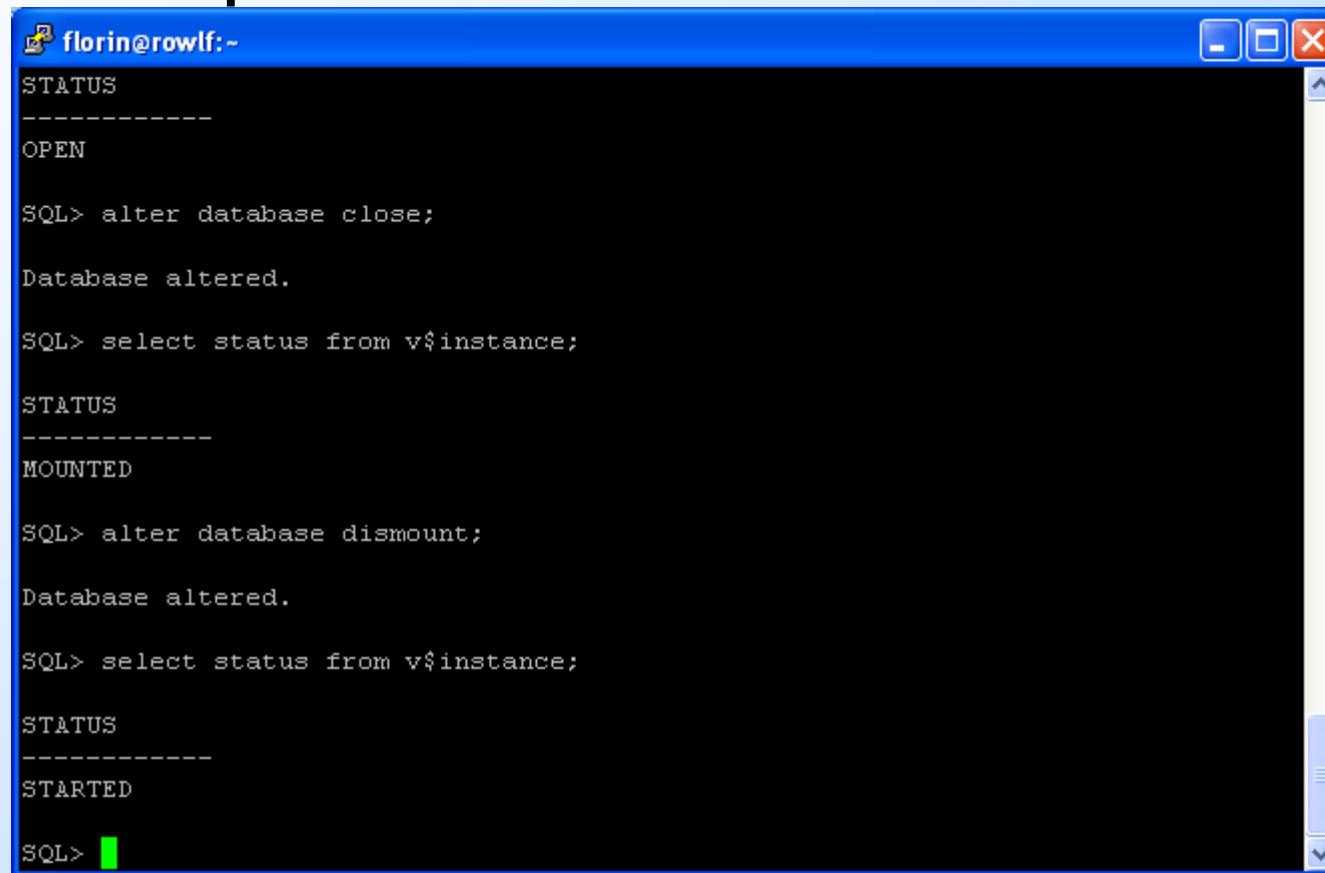
```
SQL> select status from v$instance;
```

```
STATUS
-----
MOUNTED
```

```
SQL> 
```

# Etapele opririi BD - cont

## ◆ Exemplu:



```
florin@rowlf:~
```

```
STATUS
-----
OPEN
```

```
SQL> alter database close;
```

```
Database altered.
```

```
SQL> select status from v$instance;
```

```
STATUS
-----
MOUNTED
```

```
SQL> alter database dismount;
```

```
Database altered.
```

```
SQL> select status from v$instance;
```

```
STATUS
-----
STARTED
```

```
SQL> [green square]
```

# Etapele opririi BD - cont

## ◆ Exemplu:

The screenshot shows a terminal window titled "florin@rowlf:~". It displays the following SQL\*Plus session:

```
Database altered.

SQL> select status from v$instance;

STATUS
-----
STARTED

SQL> shutdown
ORA-01507: database not mounted

ORACLE instance shut down.

SQL> select * from v$instance;
select * from v$instance
*
ERROR at line 1:
ORA-01034: ORACLE not available

ERROR:
ORA-01034: ORACLE not available

SQL>
```

# Tipuri de oprire

- ◆ Sunt 4 moduri de oprire. Oprirea normală este varianta implicită

	Normal	Tranzac tional	Imedi at	Abort
Permisione noi conexiuni	NU	NU	NU	NU
Asteapta pana se termina sesiunile curente	DA	NU	NU	NU
Asteapta pana se termina tranzactiile curente	DA	DA	NU	NU
Forteaza un checkpoint si inchide fisierele	DA	DA	DA	NU

# Oprire normală

- ◆ Nu sunt permise noi conexiuni
- ◆ Oracle asteapta ca toti userii deja conectati sa termine sesiunea de lucru (sa se deconecteze)
- ◆ Inchidere si demontare baza de date si oprire instanta
- ◆ Repornire normala (nu este nevoie de recuperare)

# Oprire tranzactionala

- ◆ Nu sunt permise noi conexiuni si nici noi tranzactii de la userii deja conectati
- ◆ La terminarea tranzactiei curente pentru orice user acesta e deconectat
- ◆ Se executa apoi pasii de la oprirea imediata
- ◆ Repornire normala (nu este nevoie de recuperare)

# Oprire imediata

- ◆ Nu sunt permise noi conexiuni
- ◆ Cererile SQL curente sunt operte din executie
- ◆ Oracle deconecteaza userii curenti
- ◆ Tranzactiile active sunt revocate (ROLLBACK)
- ◆ Inchidere si demontare baza de date si oprire instanta
- ◆ Repornire normala (nu este nevoie de recuperare)

# Orire tip ABORT

- ◆ Nu sunt permise noi conexiuni
- ◆ Cererile SQL curente sunt operte din executie
- ◆ Oracle deconecteaza userii curenti
- ◆ Tranzactiile active sunt revocate (ROLLBACK)
- ◆ Instanta este oprită fară inchiderea fisierelor
- ◆ La repornire este necesara recuperarea după incident a instantei (procesul SMON)

# Vederi dinamice privind performantele

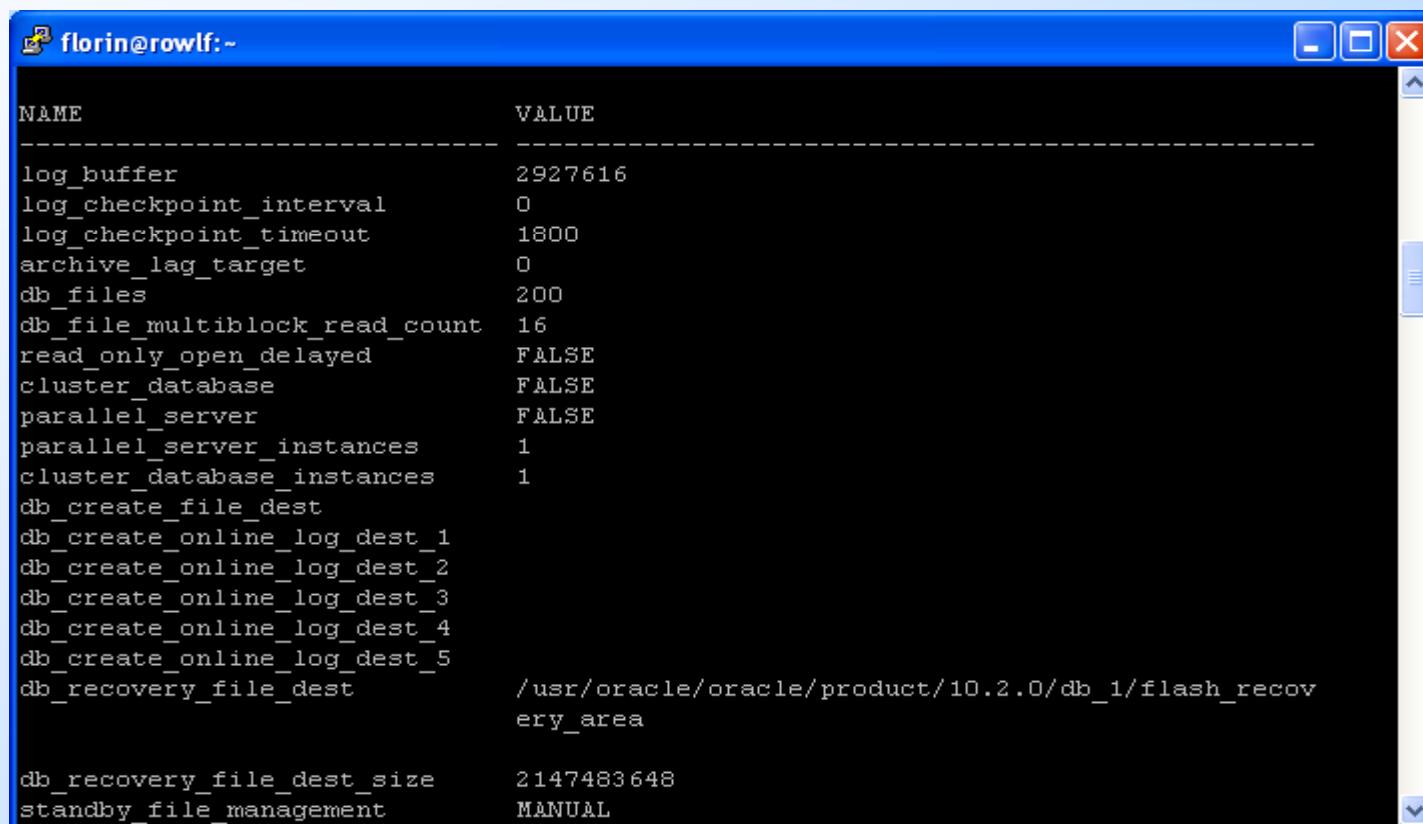
- ◆ Există tabele de sistem continând date legate de performanțe care sunt accesibile administratorului prin vederi
- ◆ Aceste vederi au un nume începând cu V\$
- ◆ Unele sunt accesibile după pornirea instantei (BD încă nemontată)
- ◆ Altele sunt accesibile doar după montarea BD
- ◆ Vezi de exemplu:  
<http://www.ss64.com/orav/>

# Exemple de vederi in cazul NOMOUNT

- ◆ V\$PARAMETER - informatii despre parametrii de initializare
- ◆ V\$SGA - informatii despre SGA
- ◆ V\$SESSION - informatii despre sesiunile curente
- ◆ V\$instance - starea instantei curente
- ◆ V\$OPTION - optiunile de instalare pentru serverul Oracle

# Vederi in caz NOMOUNT

- ◆ V\$PARAMETER - informatii despre parametrii de initializare

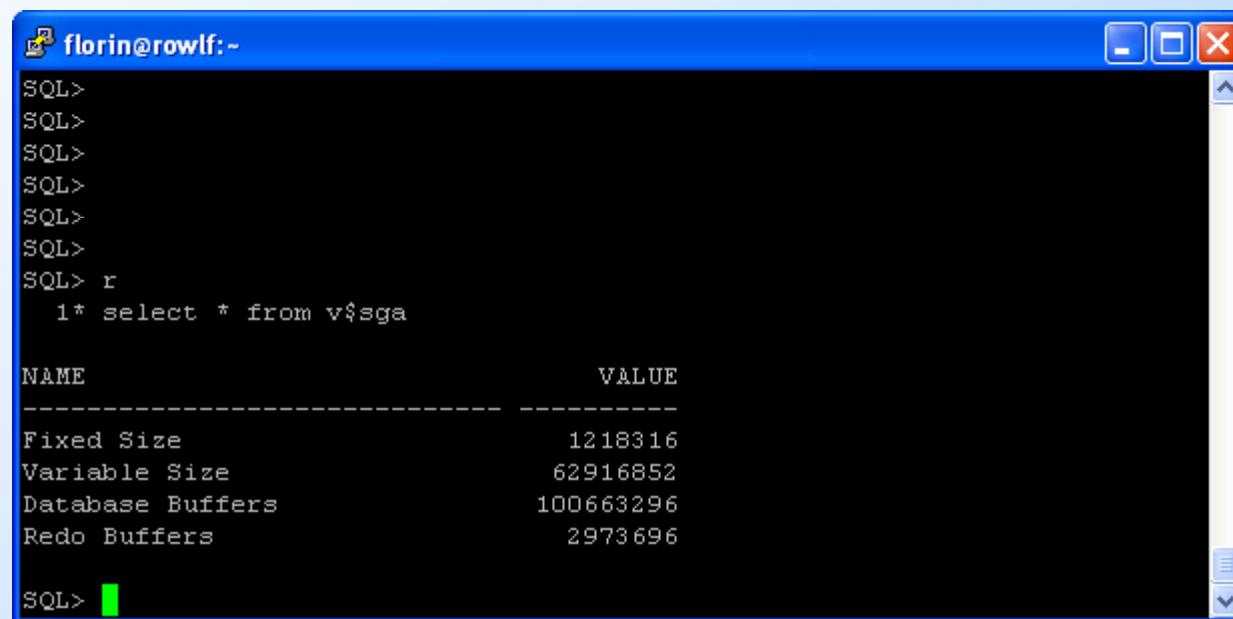


```
florin@rowlf:~$
```

NAME	VALUE
log_buffer	2927616
log_checkpoint_interval	0
log_checkpoint_timeout	1800
archive_lag_target	0
db_files	200
db_file_multiblock_read_count	16
read_only_open_delayed	FALSE
cluster_database	FALSE
parallel_server	FALSE
parallel_server_instances	1
cluster_database_instances	1
db_create_file_dest	
db_create_online_log_dest_1	
db_create_online_log_dest_2	
db_create_online_log_dest_3	
db_create_online_log_dest_4	
db_create_online_log_dest_5	
db_recovery_file_dest	/usr/oracle/oracle/product/10.2.0/db_1/flash_recovery_area
db_recovery_file_dest_size	2147483648
standby_file_management	MANUAL

# Vederi in caz NOMOUNT

- ◆ V\$SGA - informatii despre SGA

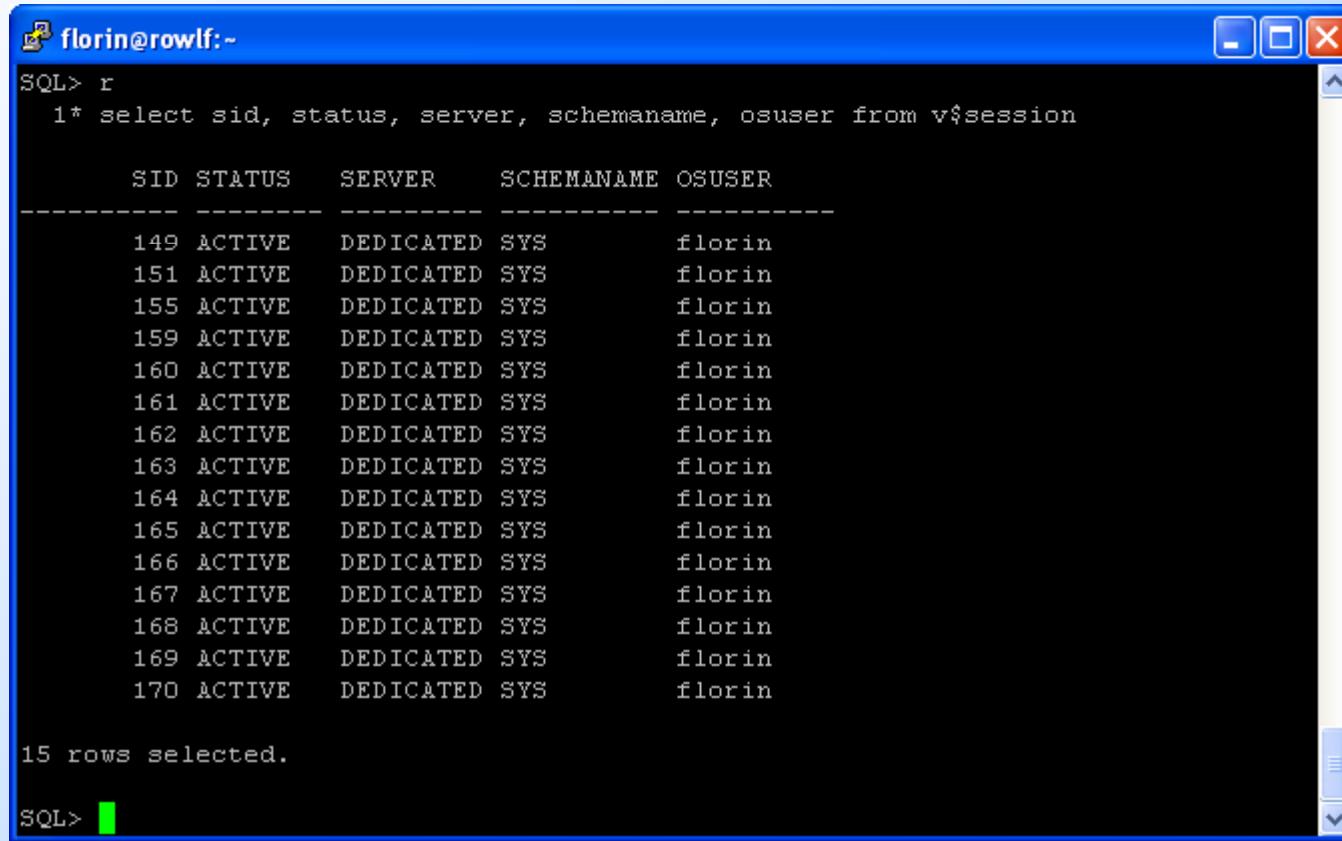


The screenshot shows a terminal window titled "florin@rowlf:~". The user has entered the command "select \* from v\$sga" and is viewing the results:

NAME	VALUE
Fixed Size	1218316
Variable Size	62916852
Database Buffers	100663296
Redo Buffers	2973696

# Vederi in caz NOMOUNT

- ◆ V\$SESSION - informatii despre sesiunile curente



The screenshot shows a Windows-style application window titled "florin@rowlf:-". Inside, an Oracle SQL\*Plus session is running. The command entered is:

```
SQL> r
  1* select sid, status, server, schemaname, osuser from v$session
```

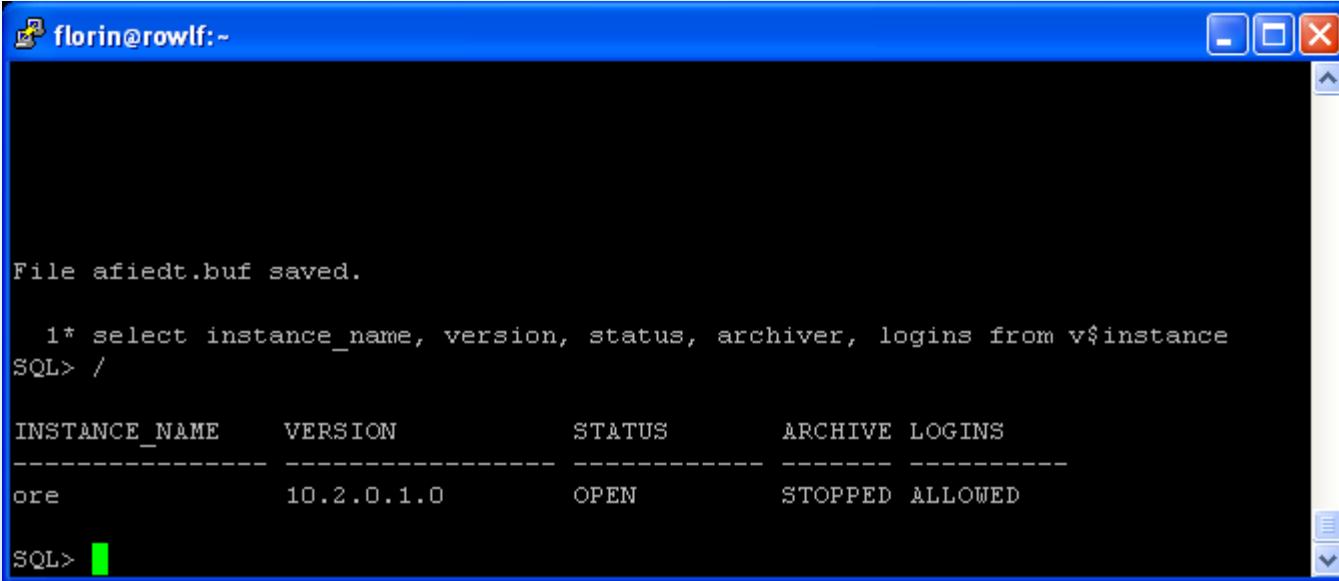
The output displays 15 rows of session details:

SID	STATUS	SERVER	SCHEMANAME	OSUSER
149	ACTIVE	DEDICATED	SYS	florin
151	ACTIVE	DEDICATED	SYS	florin
155	ACTIVE	DEDICATED	SYS	florin
159	ACTIVE	DEDICATED	SYS	florin
160	ACTIVE	DEDICATED	SYS	florin
161	ACTIVE	DEDICATED	SYS	florin
162	ACTIVE	DEDICATED	SYS	florin
163	ACTIVE	DEDICATED	SYS	florin
164	ACTIVE	DEDICATED	SYS	florin
165	ACTIVE	DEDICATED	SYS	florin
166	ACTIVE	DEDICATED	SYS	florin
167	ACTIVE	DEDICATED	SYS	florin
168	ACTIVE	DEDICATED	SYS	florin
169	ACTIVE	DEDICATED	SYS	florin
170	ACTIVE	DEDICATED	SYS	florin

At the bottom, the message "15 rows selected." is displayed, followed by the SQL prompt "SQL>".

# Vederi in caz NOMOUNT

- ◆ V\$INSTANCE - starea instantei curente



The screenshot shows a terminal window titled "florin@rowlf:~". The output of the SQL\*Plus command "select instance\_name, version, status, archiver, logins from v\$instance" is displayed. The output shows one row for the instance "ore".

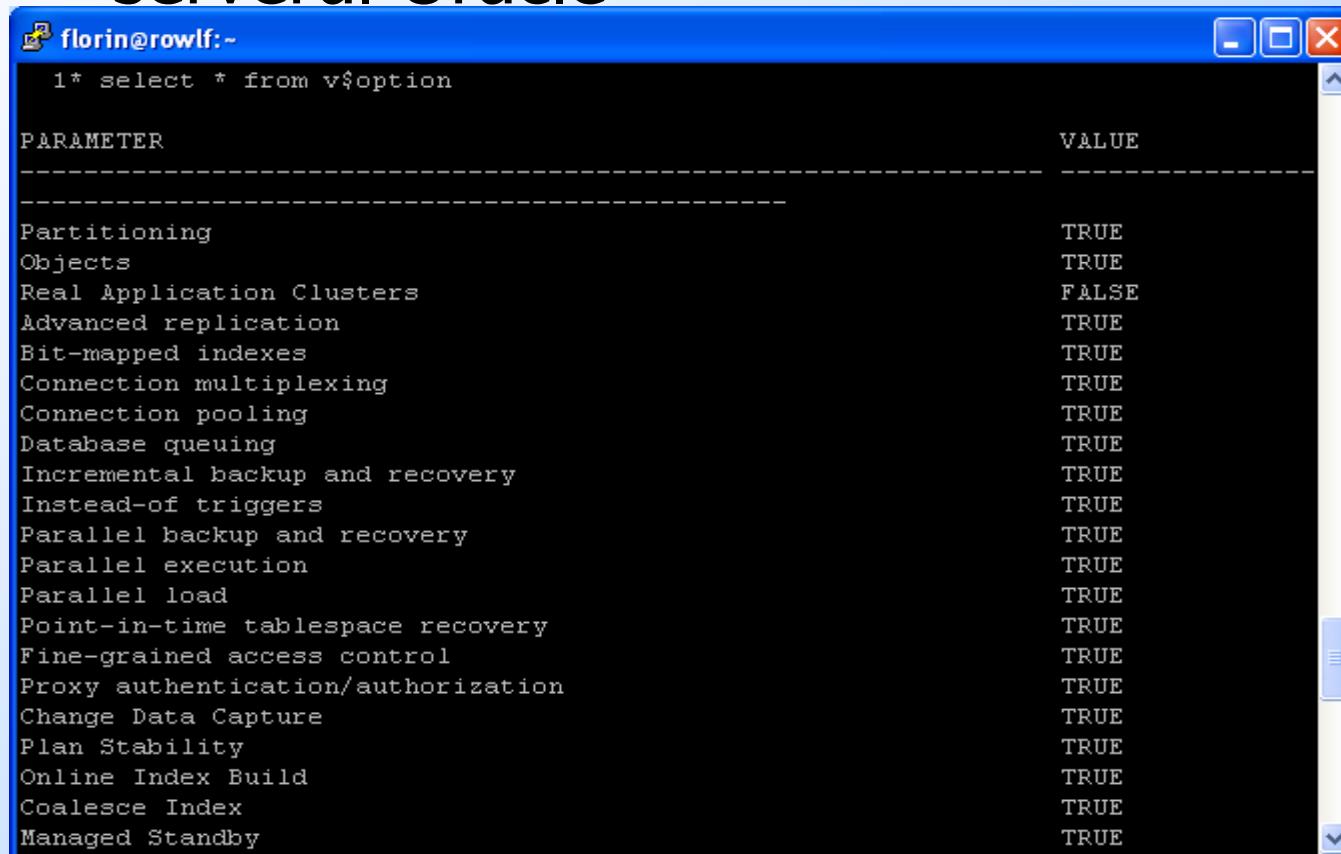
```
File afiedt.buf saved.

 1* select instance_name, version, status, archiver, logins from v$instance
SQL> /
 
 INSTANCE_NAME      VERSION          STATUS        ARCHIVE LOGINS
-----  -----
ore                10.2.0.1.0      OPEN          STOPPED ALLOWED

SQL> [green bar]
```

# Vederi in caz NOMOUNT

- ◆ V\$OPTION - optiunile de instalare pentru serverul Oracle



The screenshot shows a terminal window with a blue title bar containing the text "florin@rowlf:~". The main area of the window displays the results of a SQL query:

```
1* select * from v$option
```

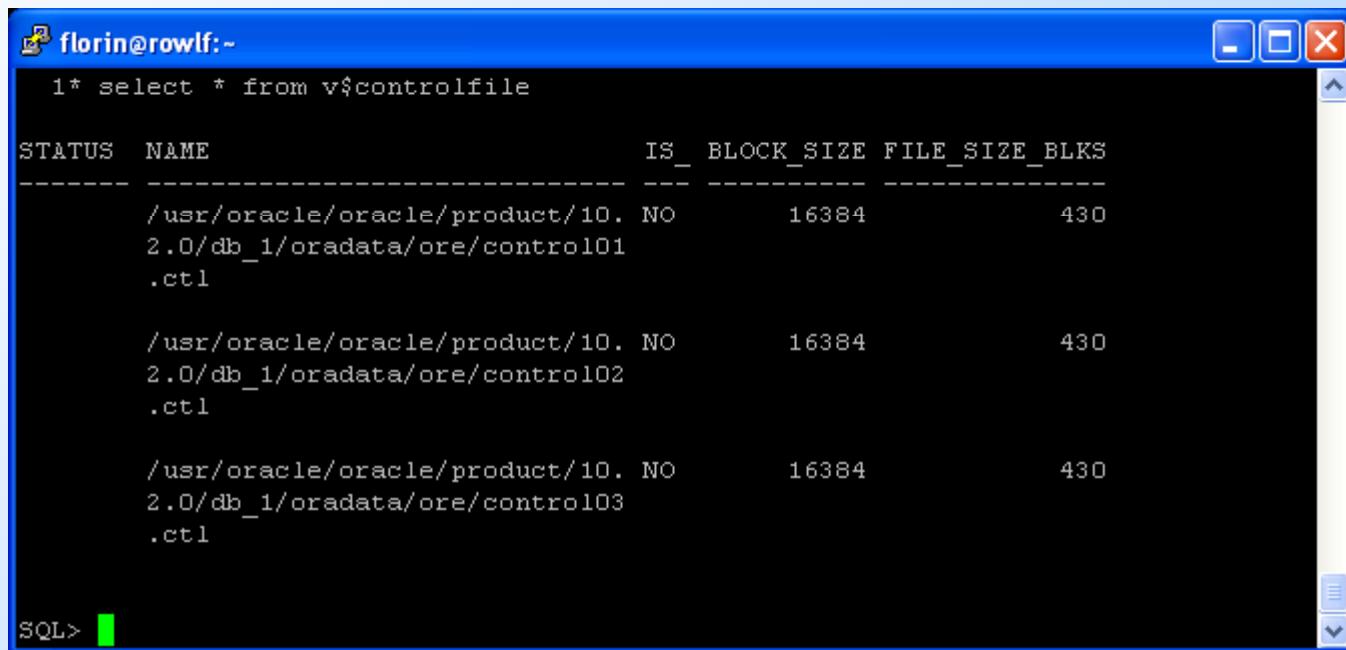
PARAMETER	VALUE
Partitioning	TRUE
Objects	TRUE
Real Application Clusters	FALSE
Advanced replication	TRUE
Bit-mapped indexes	TRUE
Connection multiplexing	TRUE
Connection pooling	TRUE
Database queuing	TRUE
Incremental backup and recovery	TRUE
Instead-of triggers	TRUE
Parallel backup and recovery	TRUE
Parallel execution	TRUE
Parallel load	TRUE
Point-in-time tablespace recovery	TRUE
Fine-grained access control	TRUE
Proxy authentication/authorization	TRUE
Change Data Capture	TRUE
Plan Stability	TRUE
Online Index Build	TRUE
Coalesce Index	TRUE
Managed Standby	TRUE

# Exemple de vederi in cazul MOUNT

- ◆ V\$CONTROLFILE - numele fisierelor de control
- ◆ V\$DATABASE - informatii despre baza de date
- ◆ V\$DATAFILE - informatii despre fisierele de date luate din fisierele de control
- ◆ V\$LOGFILE - informatii despre fisierele curente de tip Redo log

# Vederi in cazul MOUNT

- ◆ V\$CONTROLFILE - numele fisierelor de control



```
florin@rowlf:~
1* select * from v$controlfile

STATUS    NAME          IS_BLOCK_SIZE FILE_SIZE_BLKS
-----  -----
          /usr/oracle/oracle/product/10. NO        16384      430
          2.0/db_1/oradata/ore/control01
          .ctl

          /usr/oracle/oracle/product/10. NO        16384      430
          2.0/db_1/oradata/ore/control02
          .ctl

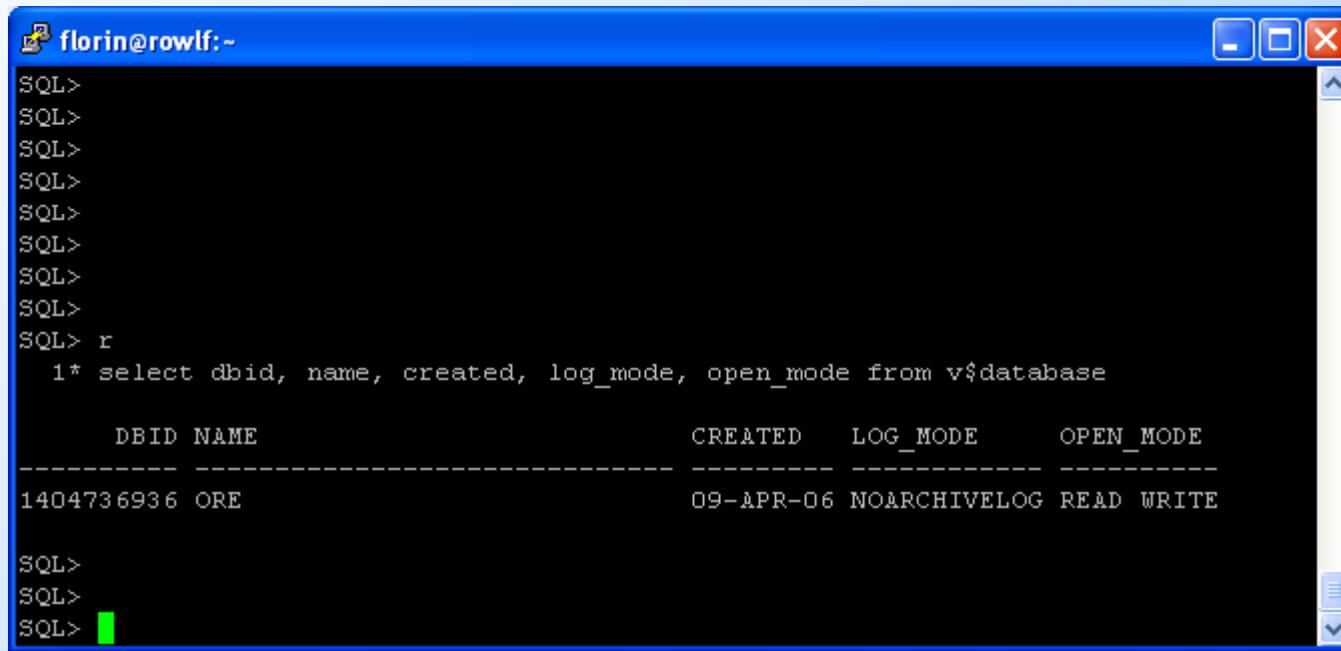
          /usr/oracle/oracle/product/10. NO        16384      430
          2.0/db_1/oradata/ore/control03
          .ctl

SQL> 
```

The screenshot shows a terminal window titled "florin@rowlf:~". It displays the output of a SQL\*Plus command: "select \* from v\$controlfile". The output lists three control files with their names, sizes, and block sizes. The names are "/usr/oracle/oracle/product/10.2.0/db\_1/oradata/ore/control01.ctl", "/usr/oracle/oracle/product/10.2.0/db\_1/oradata/ore/control02.ctl", and "/usr/oracle/oracle/product/10.2.0/db\_1/oradata/ore/control03.ctl". Each file has a size of 16384 bytes and 430 blocks. The "IS\_BLOCK\_SIZE" column shows "NO" for all three entries.

# EXEMPLE - cont

- ◆ V\$DATABASE - informatii despre baza de date



```
florin@rowlf:~
```

```
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL> r
  1* select dbid, name, created, log_mode, open_mode from v$database
```

DBID	NAME	CREATED	LOG_MODE	OPEN_MODE
1404736936	ORE	09-APR-06	NOARCHIVELOG	READ WRITE

```
SQL>
SQL>
```

# EXEMPLE - cont

- ◆ V\$DATAFILE - informatii despre fisierele de date luate din fisierele de control

The screenshot shows a terminal window titled "florin@rowlf:~". The command entered is:

```
SQL> select file#, name, status, enabled, blocks, block_size from v$datafile;
```

The output displays the following data:

FILE#	NAME	STATUS	ENABLED	BLOCKS	BLOCK_SIZE
1	/usr/oracle/oracle/product/10.2.0/db_1/oradata/ore/system01.dbf	SYSTEM	READ WRITE	62720	819
2	/usr/oracle/oracle/product/10.2.0/db_1/oradata/ore/undotbs01.dbf	ONLINE	READ WRITE	19840	819
3	/usr/oracle/oracle/product/10.2.0/db_1/oradata/ore/sysaux01.dbf	ONLINE	READ WRITE	44800	819
4	/usr/oracle/oracle/product/10.2.0/db_1/oradata/ore/users01.dbf	ONLINE	READ WRITE	2720	819
5	/usr/oracle/oracle/product/10.2.0/db_1/oradata/ore/example01.dbf	ONLINE	READ WRITE	12800	819

At the bottom of the window, there is a green vertical bar followed by the prompt:

```
SQL>
```

# EXEMPLE - cont

- ◆ V\$LOGFILE - informatii despre fisierele curente de tip Redo log



The screenshot shows a terminal window titled "florin@rowlf:~". The user has run the command "SQL> r" followed by "1\* select \* from v\$logfile". The output displays the current redo log configuration:

GROUP#	STATUS	TYPE	MEMBER	IS_
3	STALE	ONLINE	/usr/oracle/oracle/product/10.2.0/db_1/oradata/ore NO /redo03.log	
2		ONLINE	/usr/oracle/oracle/product/10.2.0/db_1/oradata/ore NO /redo02.log	
1		ONLINE	/usr/oracle/oracle/product/10.2.0/db_1/oradata/ore NO /redo01.log	

# Parametri dinamici

- ◆ Unii parametrii de initializare pot fi alterati dinamic (cand instanta este pornita)
- ◆ Sunt cei care sunt marcati ca modificabili in coloanele  
ISSES\_MODIFIABLE  
ISSYS\_MODIFIABLE  
din vedere V\$PARAMETER
- ◆ Comenzile ALTER SYSTEM [DEFERRED] sunt inregistrate in fisierul de alerte (ALERT file)

# Parametri dinamici - cont

Exemplu:

**ALTER SESSION SET nume\_parametru=valoare**

- modifica parametrul doar pentru sesiunea unde este executata comanda

**ALTER SYSTEM SET nume\_parametru=valoare  
[DEFERRED]**

- modifica global parametrul. Noua valoare este in uz pana la oprirea BD
- optiunea DEFERRED modifica parametrul pentru sesiunile care se deschid dupa executia comenzii (nu si pentru cele deschise)

# Sesiuni RESTRICTED

- ◆ Sunt folosite cand se efectueaza operatii de mentenanta asupra bazei de date.
- ◆ Cand baza de date e pornita in mod RESTRICTED doar userii cu privilegiul RESTRICTED SESSION pot sa se conecteze.
- ◆ La pornire se da STARTUP RESTRICT

# Sesiuni RESTRICTED - cont

- ◆ Daca baza de date este deja pornita se poate trece in mod RESTRICTED cu comanda:

```
ALTER SYSTEM {ENABLE | DISABLE }  
RESTRICTED SESSION
```

- ◆ ENABLE – se permit noi conexiuni doar de la userii cu privilegiul mentionat. Sesiunile existente nu sunt afectate.
- ◆ DISABLE – se permit conexiuni de la orice user

# Sesiuni RESTRICTED - cont

- ◆ Pentru a vedea modul curent putem lansa cererea:

```
select logins from v$instance;
```

- ◆ Obtinem ca rezultat o tabela (ca cea de mai jos)

**LOGINS**

---

**RESTRICTED**

# Inchiderea sesiunilor

- ◆ Dupa trecerea in modul RESTRICTED putem dori sa inchidem anumite sesiuni active.
- ◆ Aflarea datelor despre o sesiune:

```
SELECT SID, SERIAL# FROM V$SESSION  
WHERE USERNAME = 'SCOTT'
```

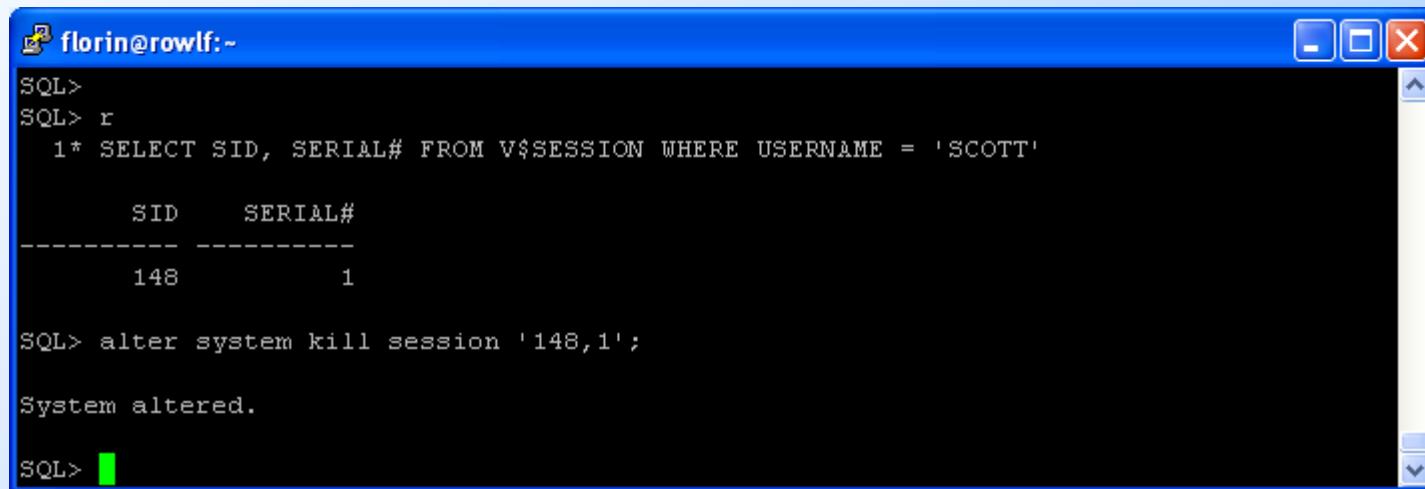
- ◆ Inchiderea unei sesiuni:

```
ALTER SYSTEM KILL SESSION '5,10'
```

Unde 5, 10 sunt numere returnate pentru SID si SERIAL#

# Inchiderea sesiunilor

- ◆ Efectul comenziilor:
- ◆ În fereastra administratorului bazei de date (SYS de exemplu):



```
florin@rowlf:~  
SQL>  
SQL> r  
 1* SELECT SID, SERIAL# FROM V$SESSION WHERE USERNAME = 'SCOTT'  
  
      SID      SERIAL#  
-----  
       148          1  
  
SQL> alter system kill session '148,1';  
  
System altered.  
  
SQL>
```

# Inchiderea sesiunilor

- ◆ Efectul comenziilor:
- ◆ În fereastra userului SCOTT:

```
florin@rowlf:-
SQL> select * from spec;

   CODS NUME      DOMENIU
   -----
        11 MATEMATICA STIINTE EXACTE
        21 GEOGRAFIE   UMANIST
        24 ISTORIE     UMANIST

SQL>
SQL>
SQL> select * from stud;
select * from stud
*
ERROR at line 1:
ORA-00028: your session has been killed

SQL>
```

# Inchiderea sesiunilor - cont

- ◆ Efectul comenzi (realizator: procesul PMON) este:
  - ◆ Se anuleaza tranzactia curenta din sesiune (rollback)
  - ◆ Se elibereaza toate resursele ocupate de acea sesiune inclusiv linii sau tabele blocate

# Fisiere TRACE

- ◆ Sunt scrise de procesele server si background
- ◆ Oracle inregistreaza in ele informatii despre erorile aparute
- ◆ Operatia se activeaza fie prin ALTER SESSION fie prin parametrul SQL\_TRACE

Exemplu:

```
ALTER SESSION SET SQL_TRACE=TRUE;
```

# Fisiere TRACE - cont

Caracteristicile fisierelor TRACE e data de parametrii:

- ◆ max\_dump\_file\_size – specificat in blocuri pe disc
- ◆ background\_dump\_dest – locatia fisierelor trace pentru procesele de background si a fisierelor ALERT
- ◆ user\_dump\_dest – locatia fisierelor TRACE create la cererea userului. Exemplu:

```
ALTER SESSION SET SQL_TRACE = TRUE
```

# FISIERE ALERT

- ◆ Sunt scrise de procesele server si background
- ◆ Oracle inregistreaza in ele cronologic mesajele si erorile
- ◆ Numele fisierului este de obicei ALERT\_<SID>.log sau <SID>alrt.log
- ◆ Contin toate erorile interne Oracle (cod -600) si erori privind coruperea datelor de pe disc (cod -1578) precum si informatii despre STARTUP, SHUTDOWN, ARCHIVE LOG, RECOVER

# CREAREA BD

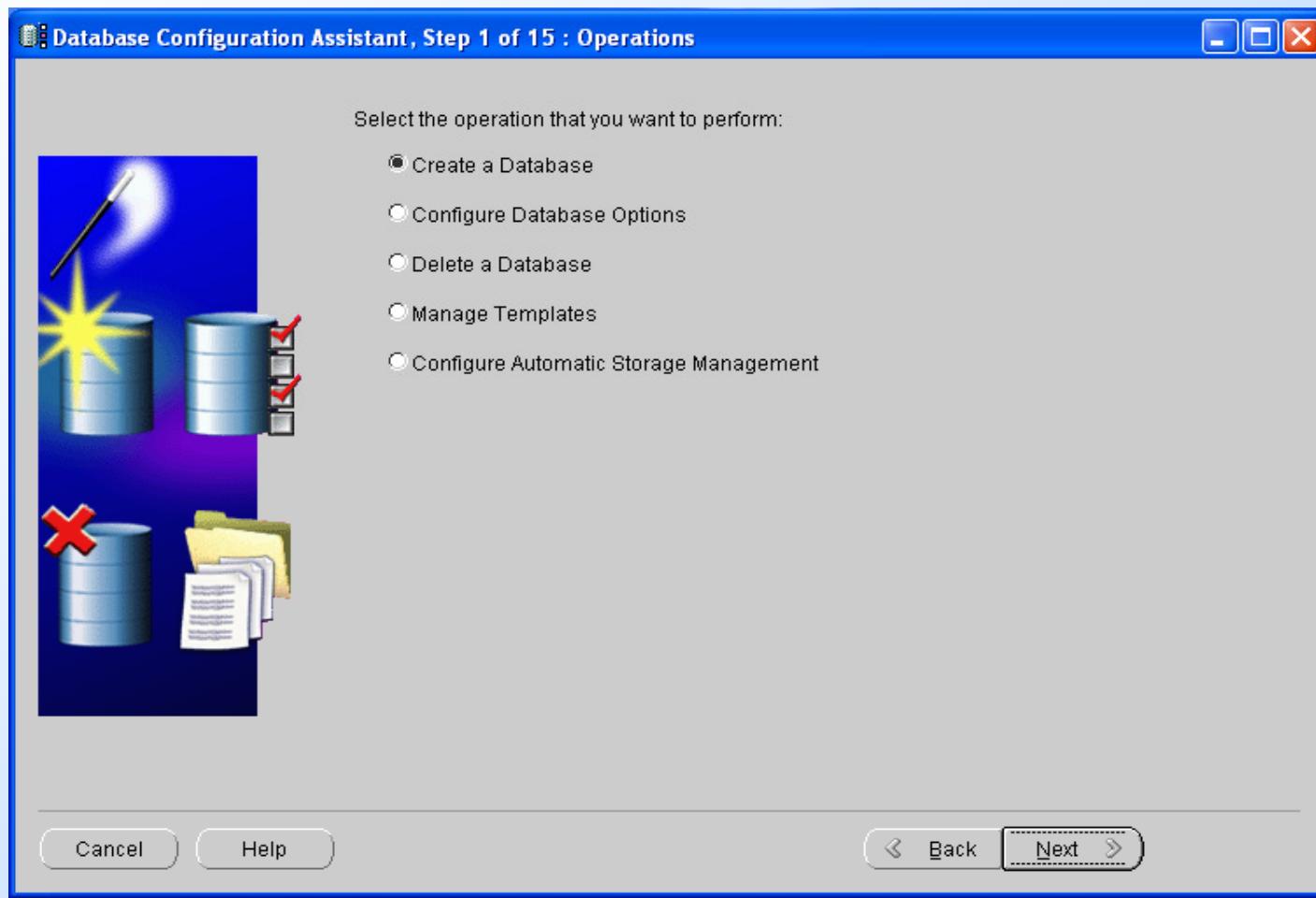
- ◆ In Oracle se poate crea o baza de date:
  - ◆ Folosind instrumentul DBCA - Database Configuration Assistant (asistent de creare a bazei de date)
  - ◆ Manual, prin comenzi SQL
- ◆ La instalarea Oracle de obicei se creaza o prima baza de date
- ◆ Se poate crea de asemenea o baza de date dupa instalare in cazuri ca:
  - ◆ S-a folosit *Oracle Universal Installer* (OUI) doar pentru instalare fara crearea unei baze de date
  - ◆ Crearea unei noi baze de date (si a unei noi instance) pe aceeasi masina
  - ◆ Crearea unei baze de date care sa fie o copie a uneia existente (clonare)

# Preliminarii

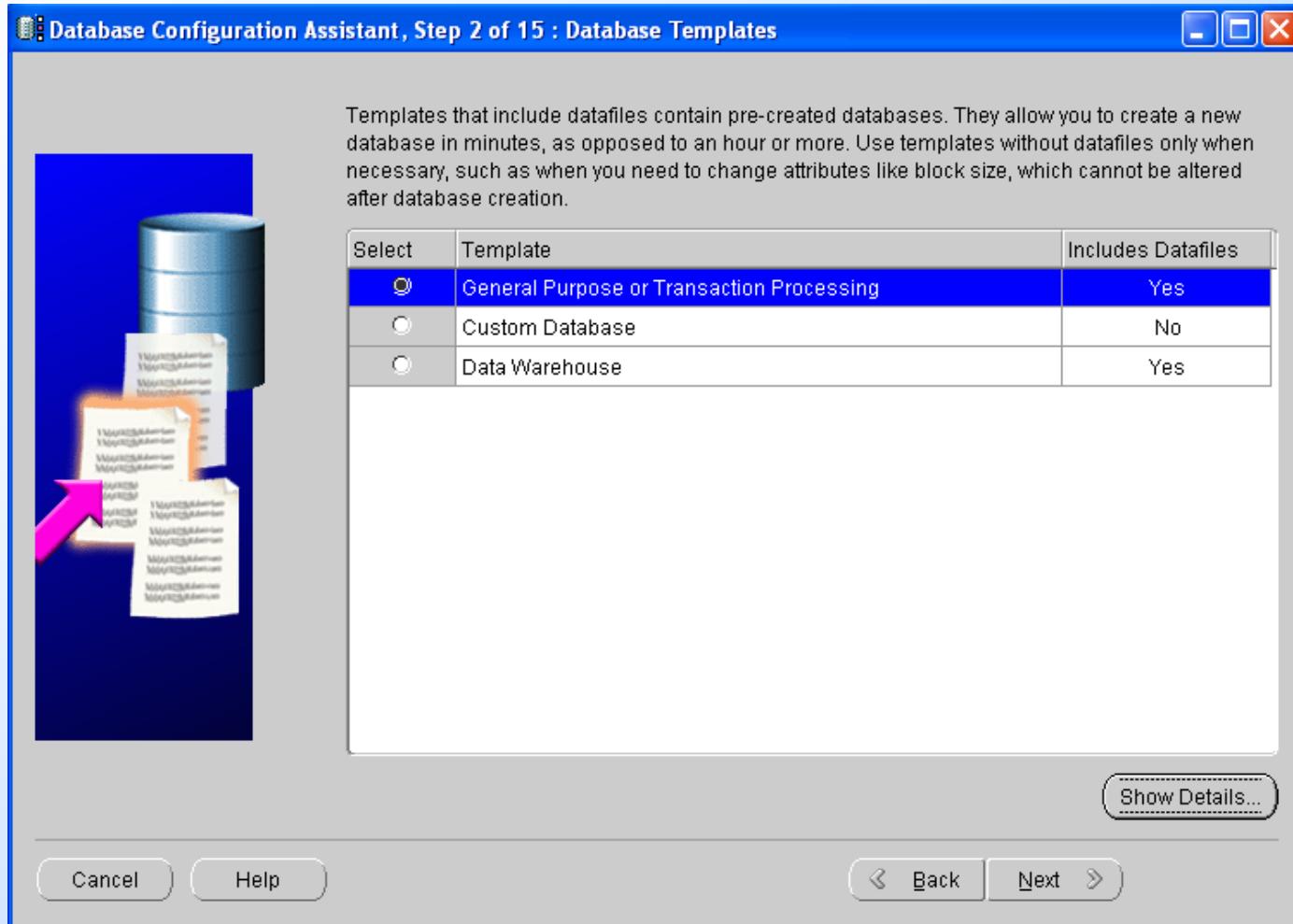
- ◆ Inainte de crearea unei baze de date trebuie sa ne asiguram ca:
- ◆ Oracle este instalat, deci exista inclusiv variabilele de mediu necesare si sunt stabilite directoarele care vor gazdui datele si aplicatiile
- ◆ Exista suficienta memorie interna pe masina in cauza pentru a putea lansa o instanta
- ◆ Exista suficient spatiu pe disc pentru crearea fisierelor necesare bazei de date
- ◆ Utilizatorul care efectueaza operatia are privilegiile necesare (este administrator de sistem de exemplu sau foloseste un fisier de parole pentru autentificare)

([http://docs.oracle.com/cd/E11882\\_01/server.112/e10897.pdf](http://docs.oracle.com/cd/E11882_01/server.112/e10897.pdf))

# DBCA: 1. Primul pas



# DBCA: 2. Tipul BD (DB template)

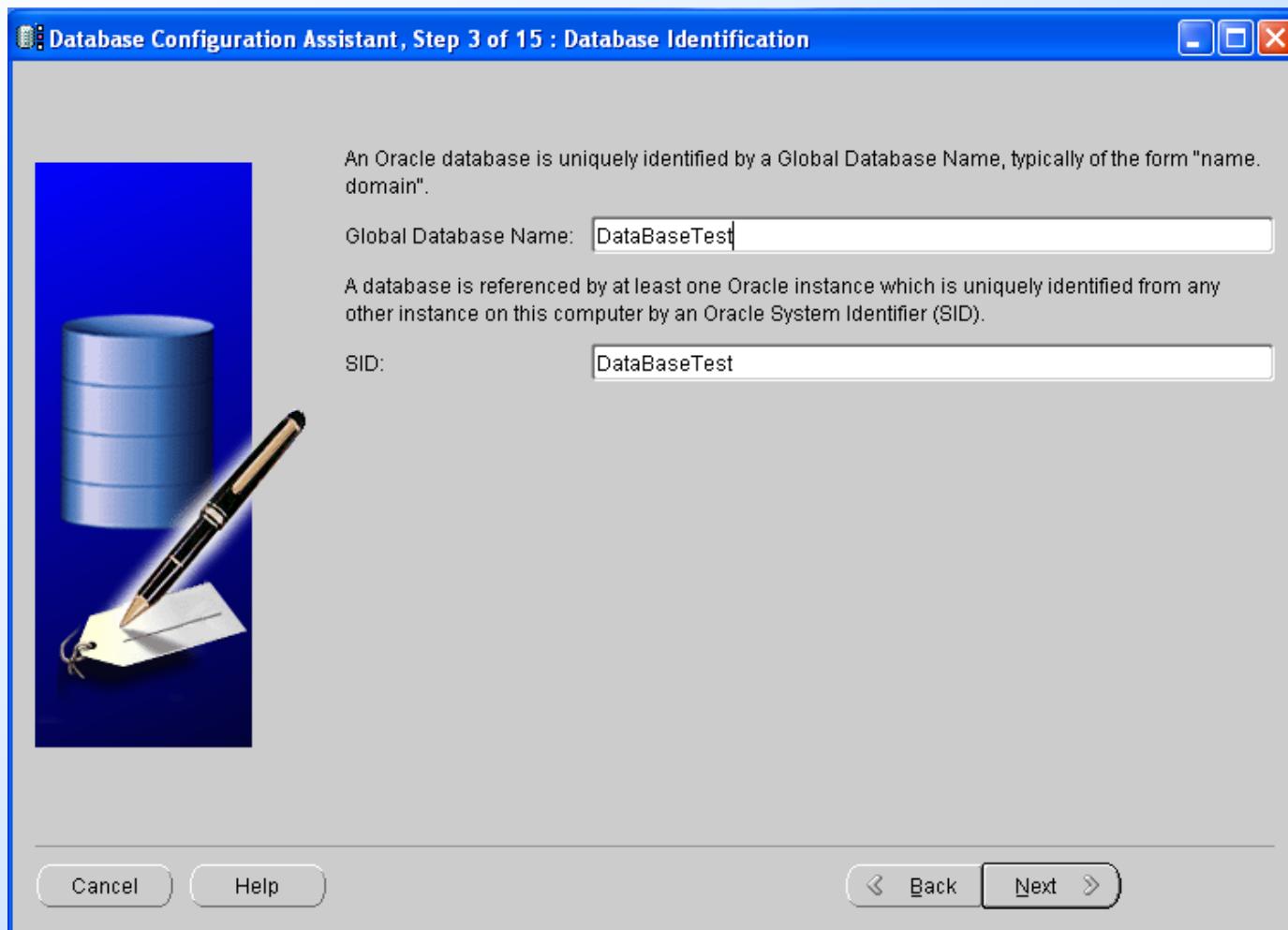


## DBCA: 2 - cont

Exista cateva sabloane predefinite de Oracle

- ◆ General Purpose or Transaction Processing
  - pentru baze de date folosite tranzactional (model ales in continuare)
- ◆ Data warehouse (pentru depozite de date)
- ◆ Se poate folosi optiunea Custom Database care implica insa o buna cunoastere a sistemului pentru configurare in acest caz. Timpul de creare pentru baza de date creste corespunzator

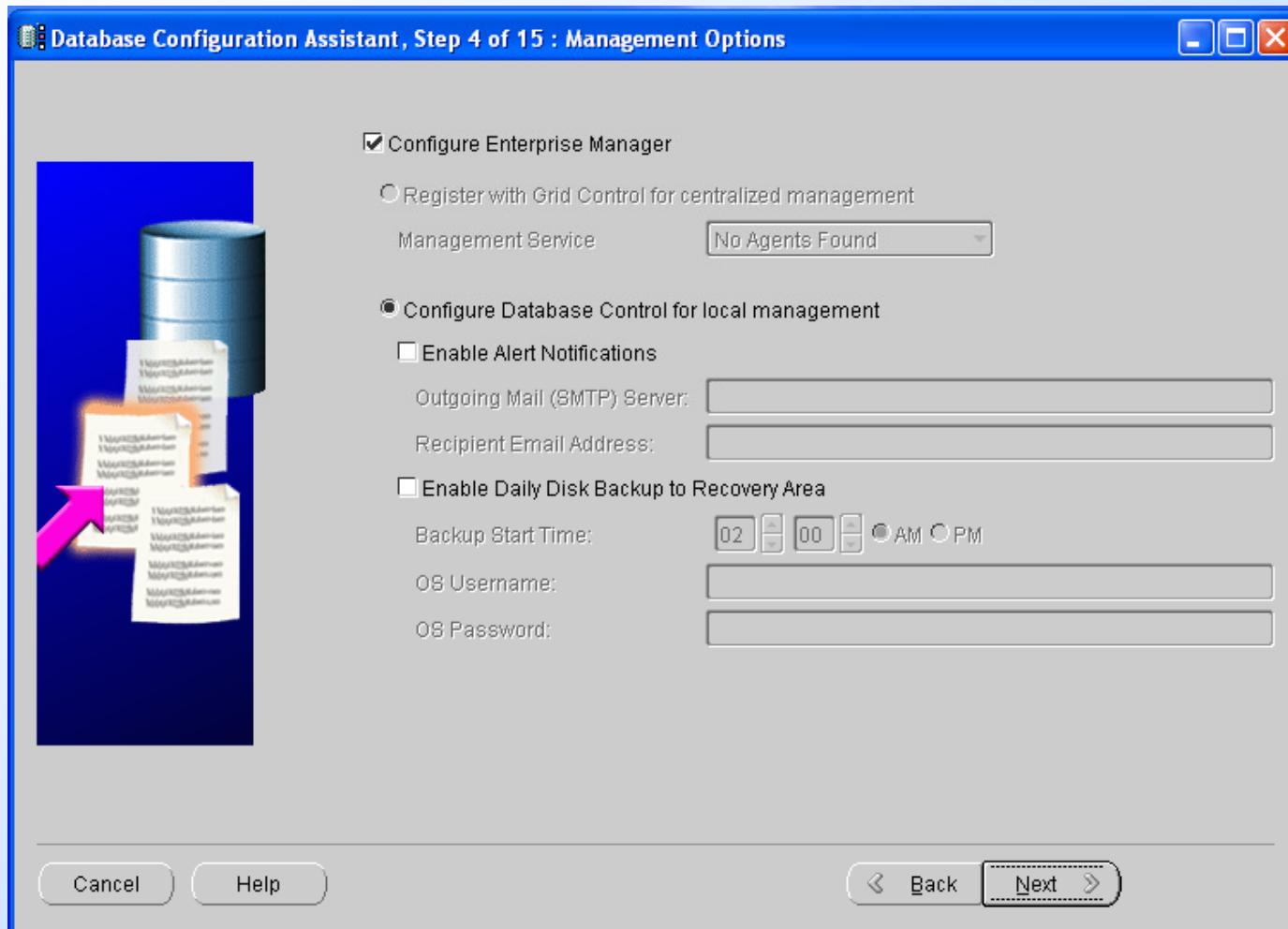
# DBCA: 3. Numele bazei



## DBCA: 3. - cont

- ◆ In campul Global Database Name se tasteaza numele bazei de date care se creeaza
- ◆ In campul SID se tasteaza identificatorul instantei pentru baza de date
- ◆ Asa cum am spus anterior este recomandat ca SID-ul sa fie acelasi cu numele bazei de date din motive de usurinta administrarii

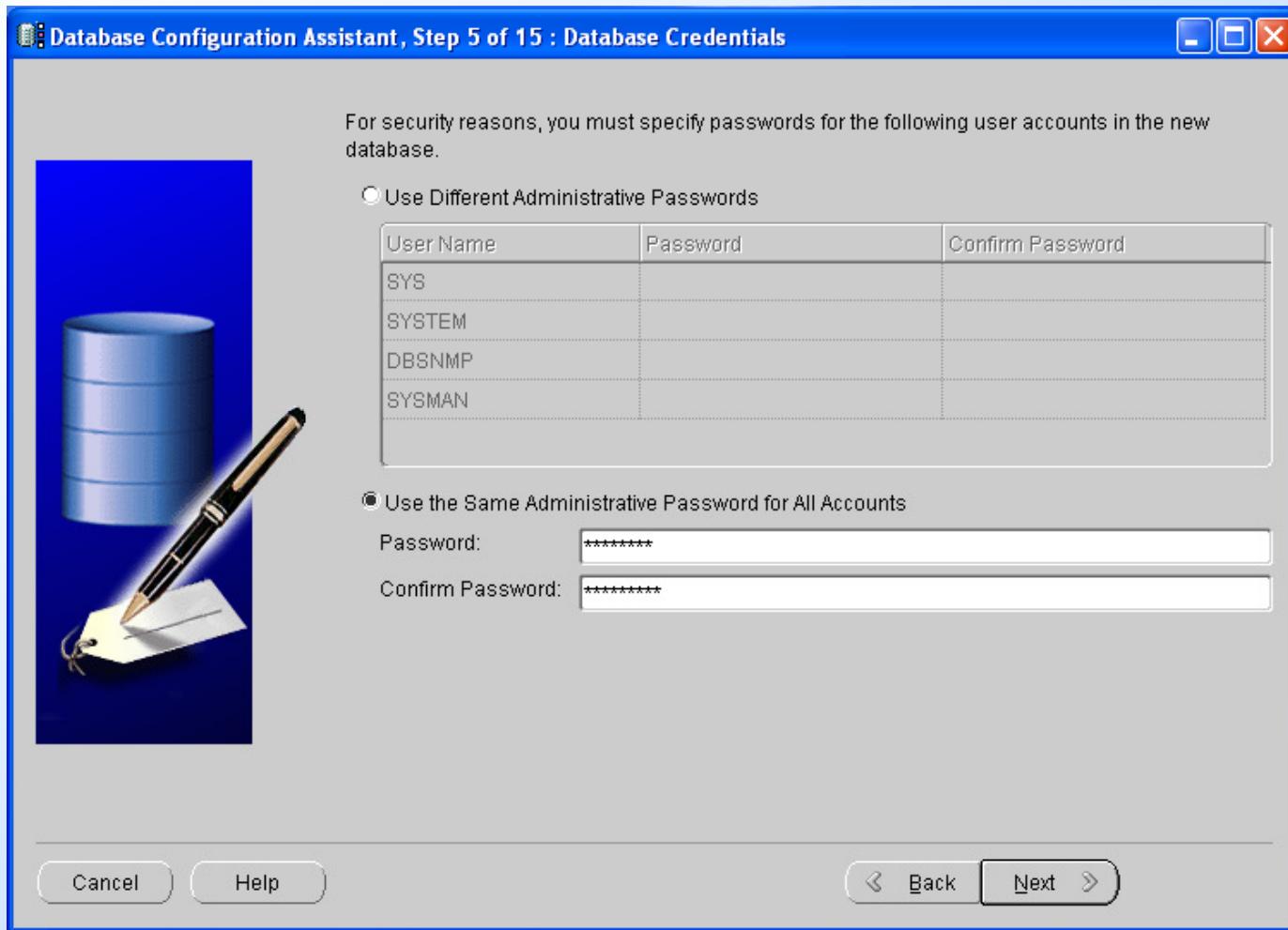
# DBCA: 4. Optiuni de gestiune



# DBCA: 4 - cont

- ◆ Se poate configura administrarea bazei de date cu ajutorul uneltei Oracle Enterprise Manager. Acesta contine posibilitatea gestionarii (web based) pentru fiecare baza de date precum si o gestiune centralizata a intregului mediu Oracle.
- ◆ Se poate apoi selecta:
  - ◆ fie gestiunea centralizata (daca Oracle Management Agent este instalat pe masina respectiva) cu optiunea *Register with Grid Control for centralized management*
  - ◆ Fie gestiunea locala – folosita in continuare – selectand *Configure Database Control for local management*.
  - ◆ In al doilea caz se poate opta pentru notificari prin email asupra diverselor probleme aparute si pentru o salvare zilnica a bazei de date

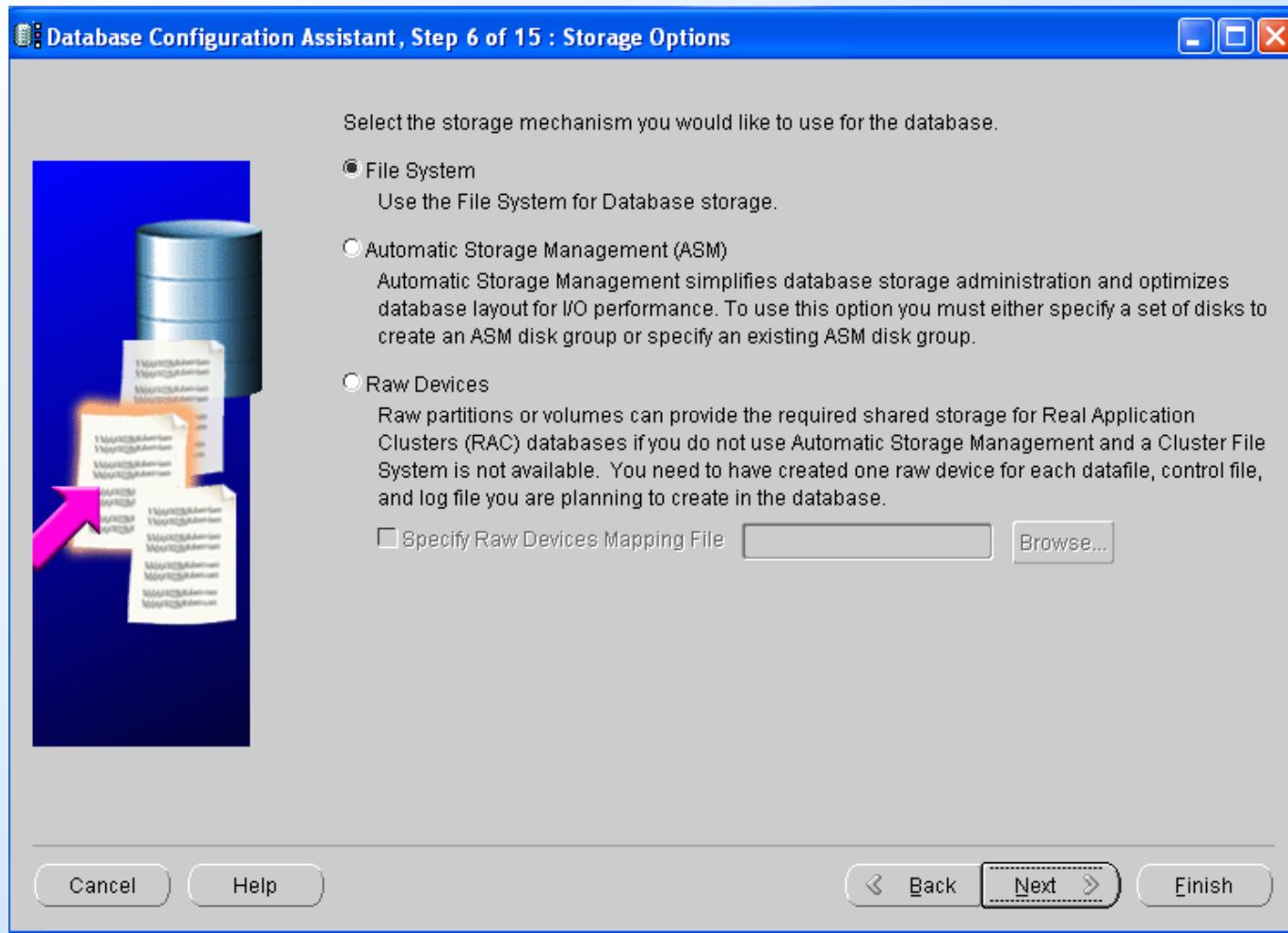
# DBCA: 5. Parole administrator



# DBCA: 5 - cont

- ◆ Se pot specifica fie parole diferite pentru conturile de administrare fie aceeasi parola pentru toate
- ◆ Conturile sunt cele din figura anterioara:
  - ◆ SYS
  - ◆ SYSTEM
  - ◆ DBSNMP – folosit de Oracle Management Agent, componenta a OEM (Oracle Enterprise Manager)
  - ◆ SYSMAN – folosit de asemenea de catre OEM

# DBCA: 6. Optiuni de stocare



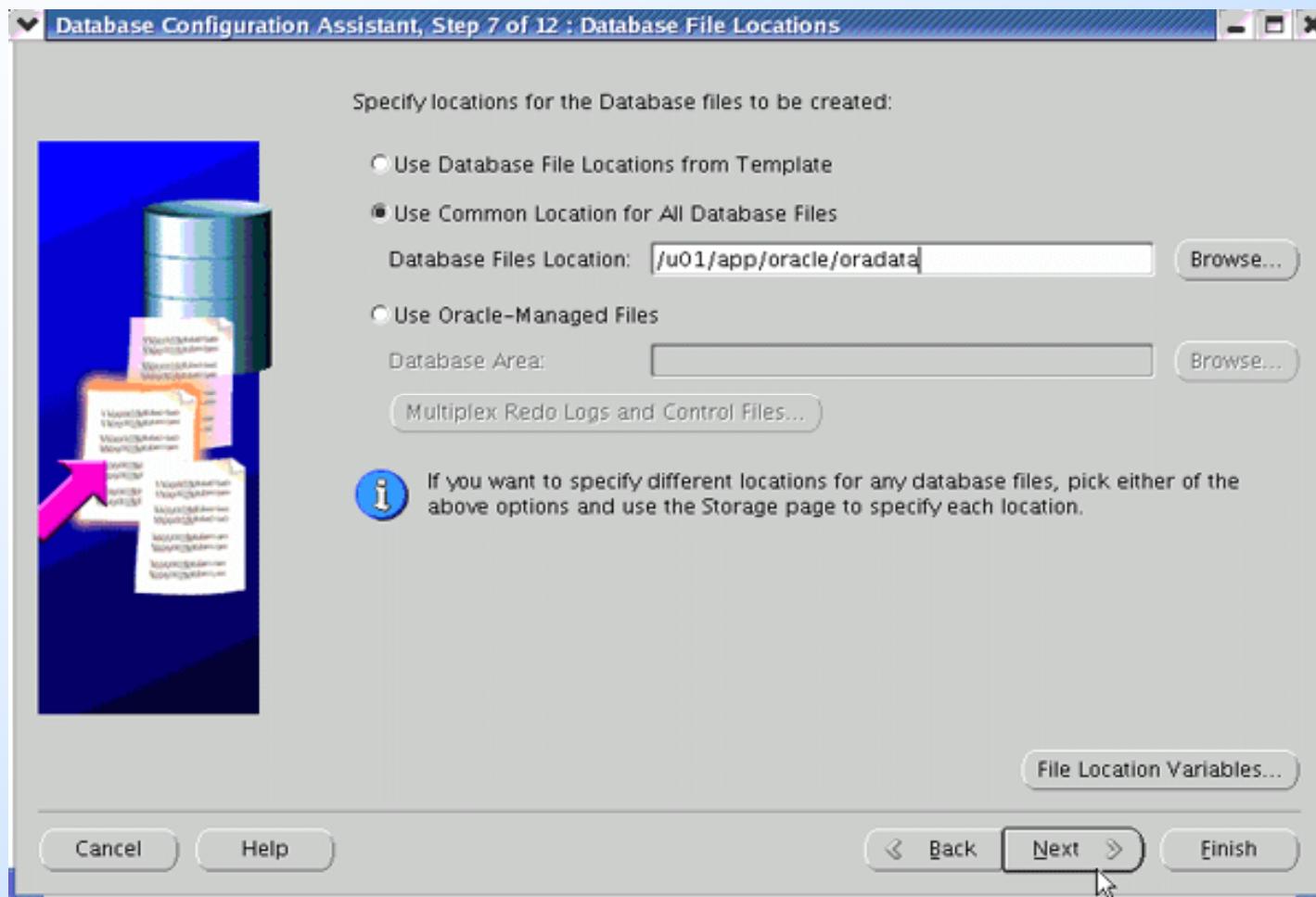
## DBCA: 6 – cont

- ◆ **File system** - fisierele care compun baza de date vor fi stocate in sistemul de fisiere al SO folosit de masina gazda – este optiunea folosita implicit
- ◆ **Automatic Storage Management** – folosita in sisteme cu un mare numar de discuri. Descrierea acestei optiuni se gaseste in anexa A din Oracle Database 2 Day DBA (v. bibliografia)

## DBCA: 6 – cont

- ◆ **Raw devices** – permite stocarea in zone din afara sistemului de operare. Pentru aceasta trebuie specificata o zona de stocare pe disc neformatata (in afara SO).
- ◆ Optiunea Raw devices se foloseste mai ales in RAC – Oracle Real Application Cluster.
- ◆ Zona respectiva trebuie anterior creata si libera de orice alta folosire, inclusiv de folosirea ei de catre o alta baza de date Oracle

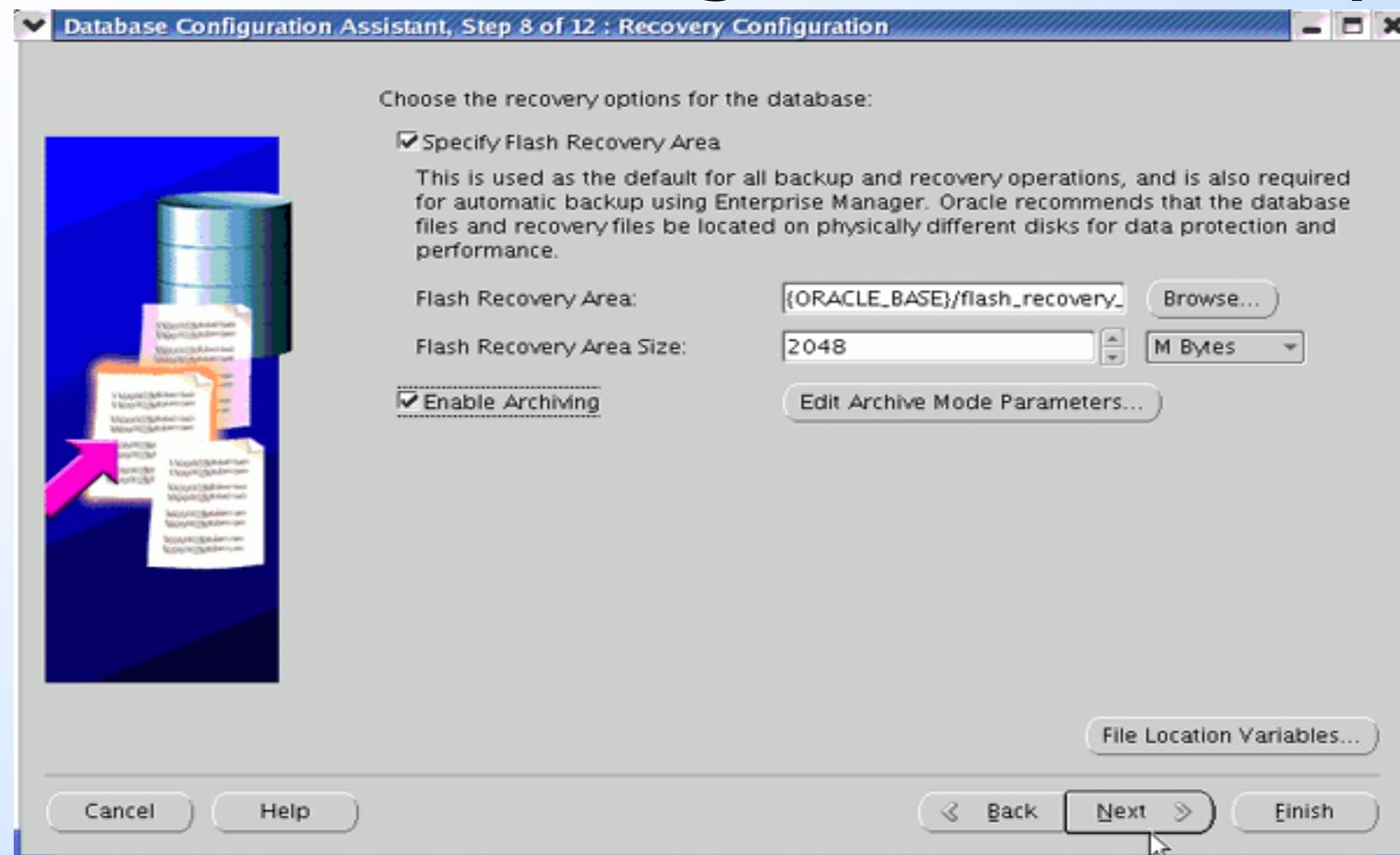
# DBCA: 7. Localizarea fisierelor



# DBCA: 7. - cont

- ◆ ***Se pot alege optiunile:***
  - ◆ ***Use Database File from Template :*** crearea se face in directoarele din sablon (vezi pasul 2)
  - ◆ ***Use Common Location for All Database Files :*** se specifica directorul unde vor fi create fisierele (ca in figura)
  - ◆ ***Use Oracle Managed Files :*** Se specifica o zona (numita database area) unde Oracle isi face singur gestiunea fisierelor. Nu mai trebuie specificate numele fisierelor, locatia lor, dimensiunile acestora.

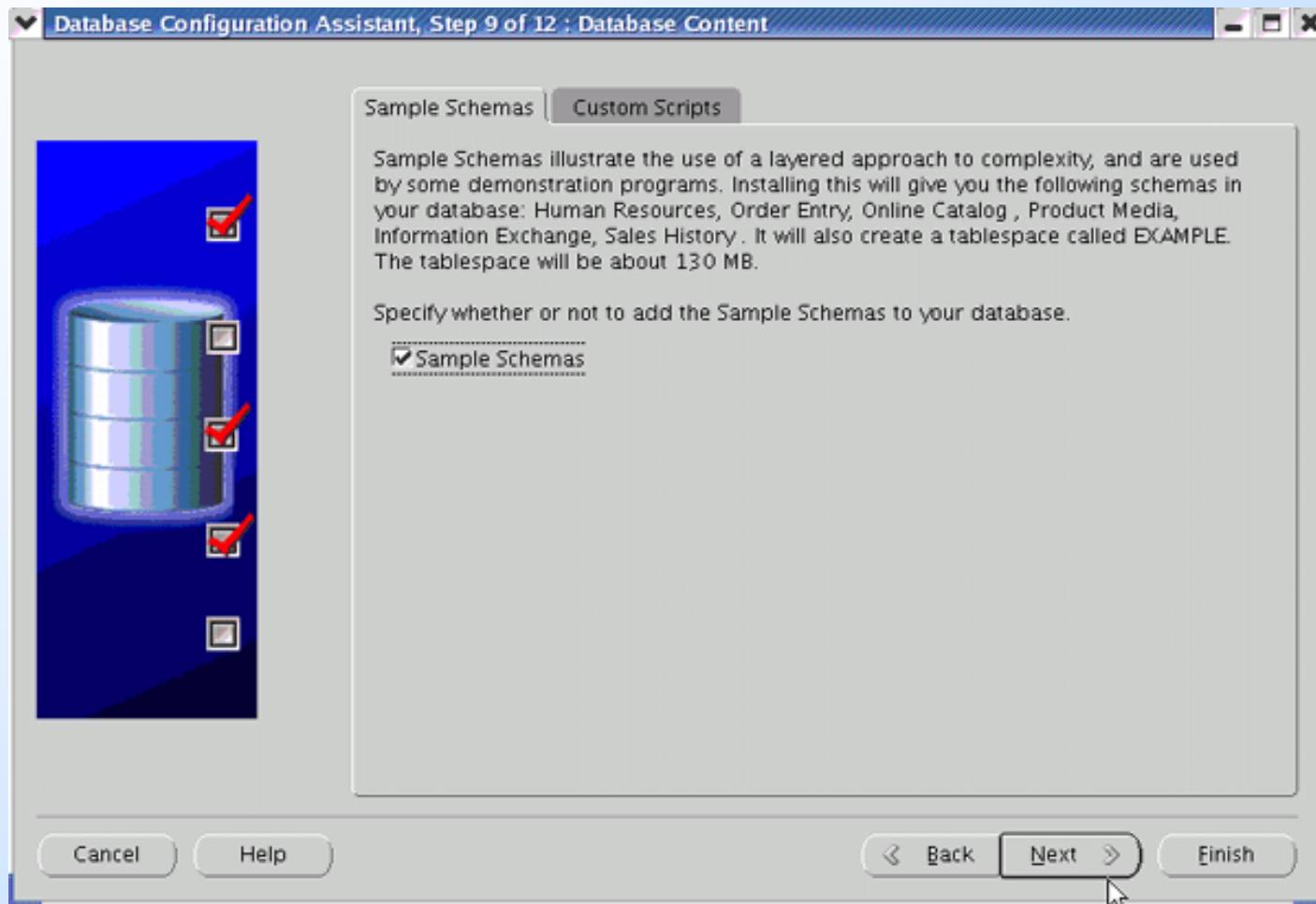
# DBCA: 8. Configurare recovery



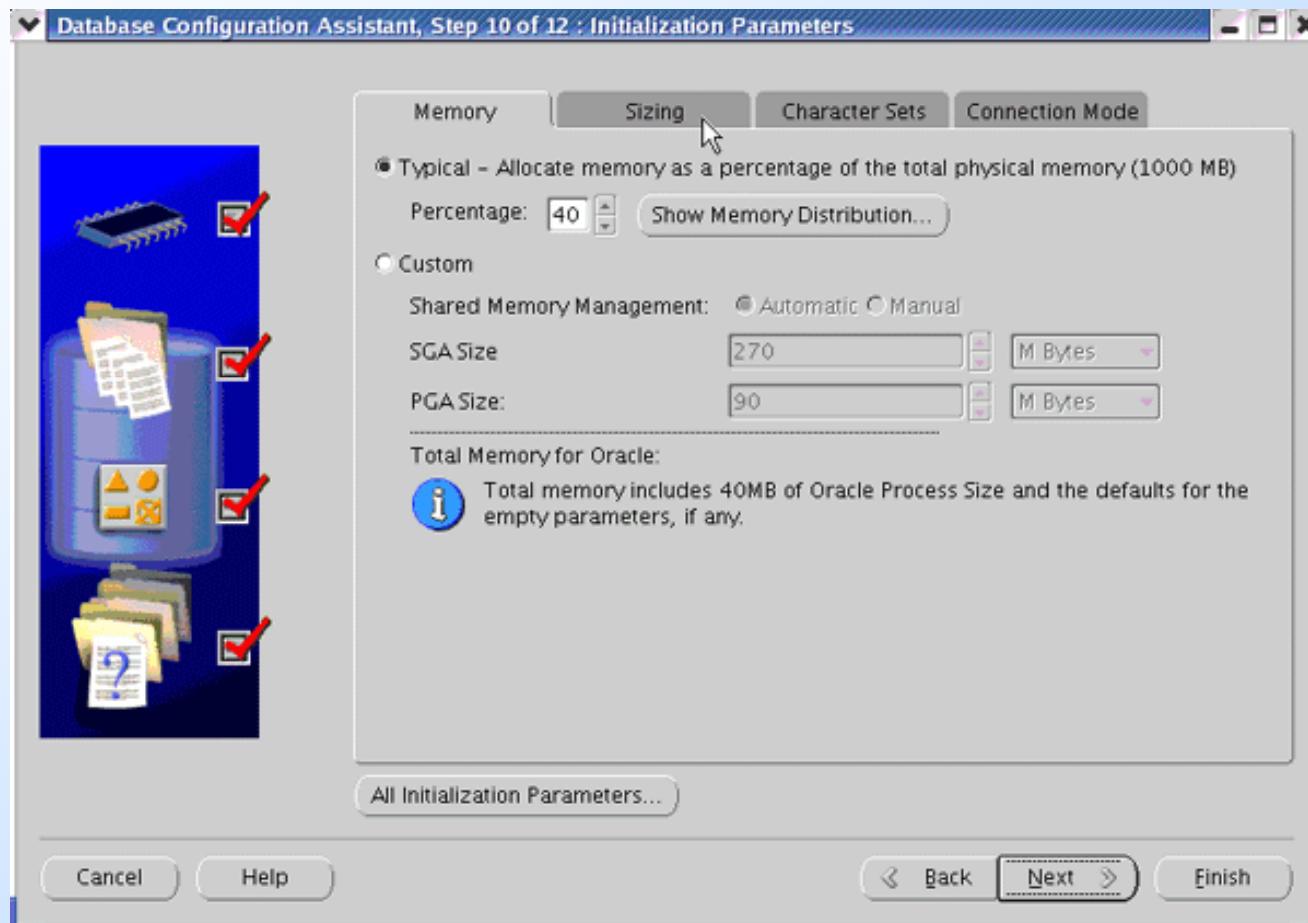
## DBCA: 8. - cont

- ◆ Aceste elemente de configuratie se folosesc in caz de incident de sistem pentru recuperarea datelor (data recovery)
- ◆ Se recomanda sa fie pe alt disc decat cel pe care se afla datele
- ◆ Se specifica Flash Recovery Area (zona de backup si recovery) si dimensiunea ei
- ◆ Se mai poate specifica si arhivarea fisierelor de tip Redo log

# DBCA: 9. BD de exemplu



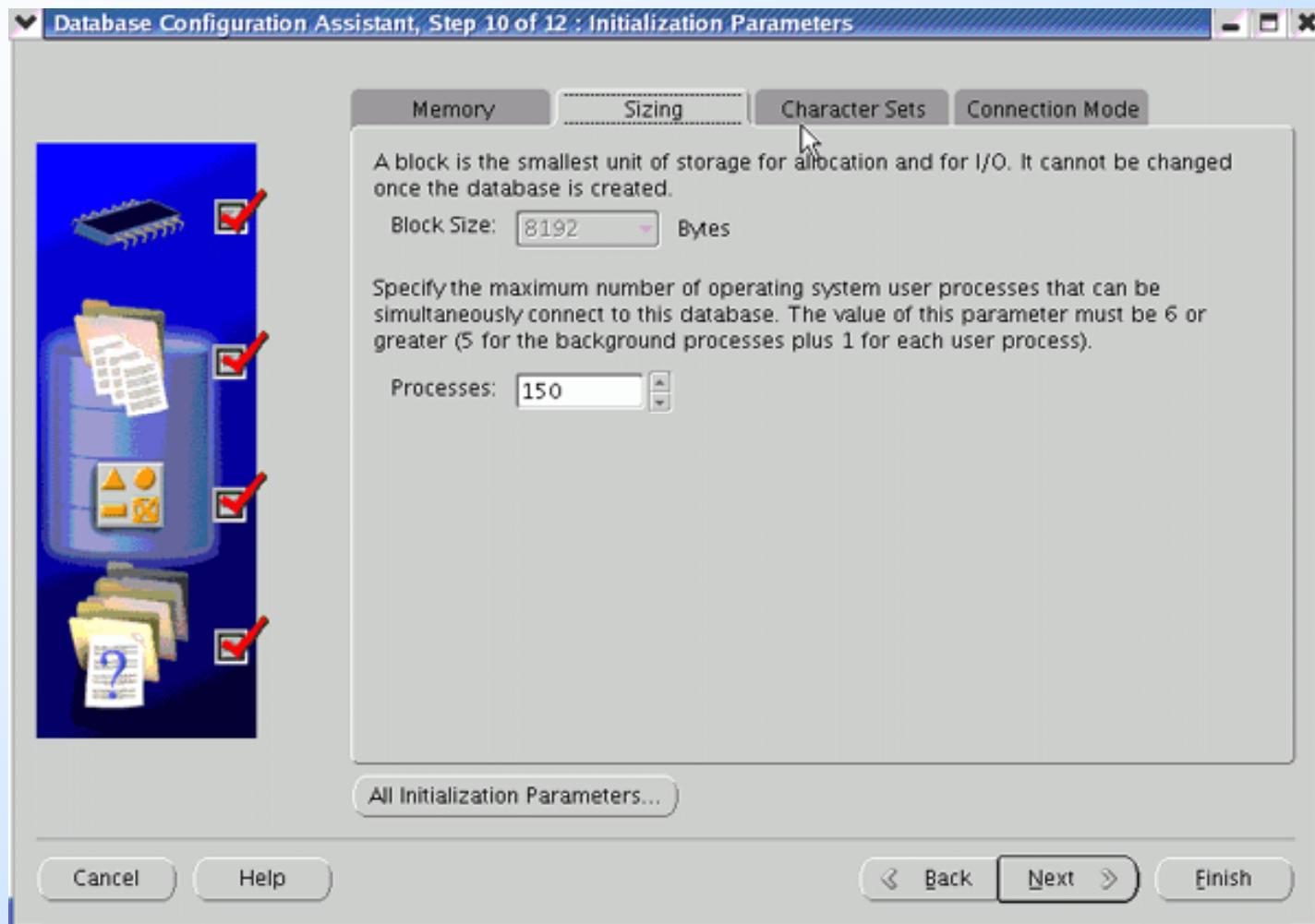
# DBCA: 10. Parametri de initializare



## DBCA: 10 - cont

- ◆ Se pot seta parametri privind:
- ◆ Tabul Memory (Memoria, cu optiunile Typical sau Custom).
  - ◆ In cazul Typical putem vedea ce s-a alocat cu "Show...".
  - ◆ In cazul Custom putem seta pe Automatic (se vad valorile alocate) sau Manual (putem seta noi aceste valori)

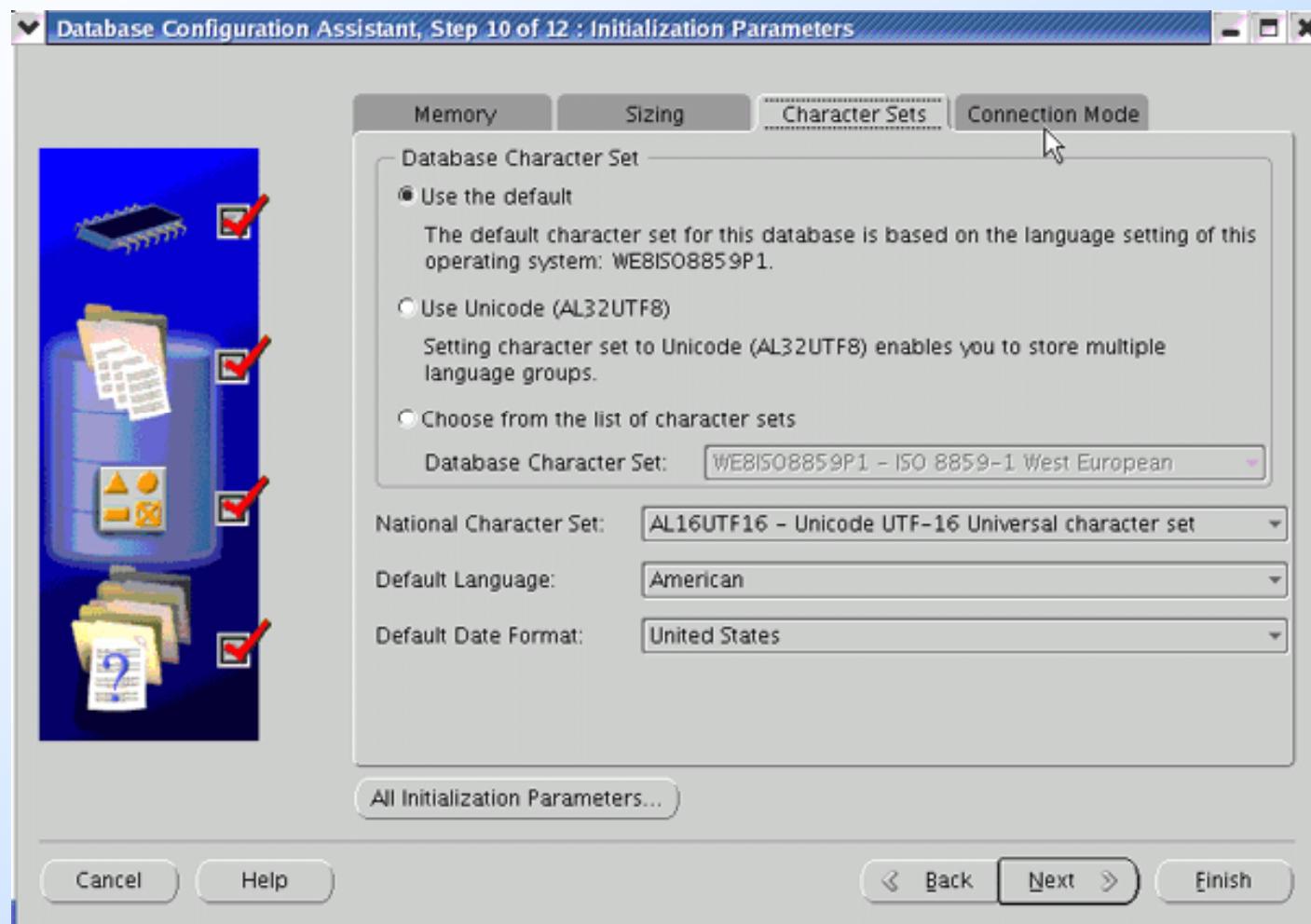
# DBCA: 10 - cont



## DBCA: 10 - cont

- ◆ Tabul Sizing. Se seteaza dimensiunea blocului si numarul maxim de procese user care se pot conecta simultan.
- ◆ Pentru dimensiunea blocului, in cazul in care se folosesc sabloane predefinite dimensiunea implicita e de 8KB
- ◆ Pentru procese, numarul implicit e de 150. Trebuie sa fie minim 6 pentru a include procesele de background.

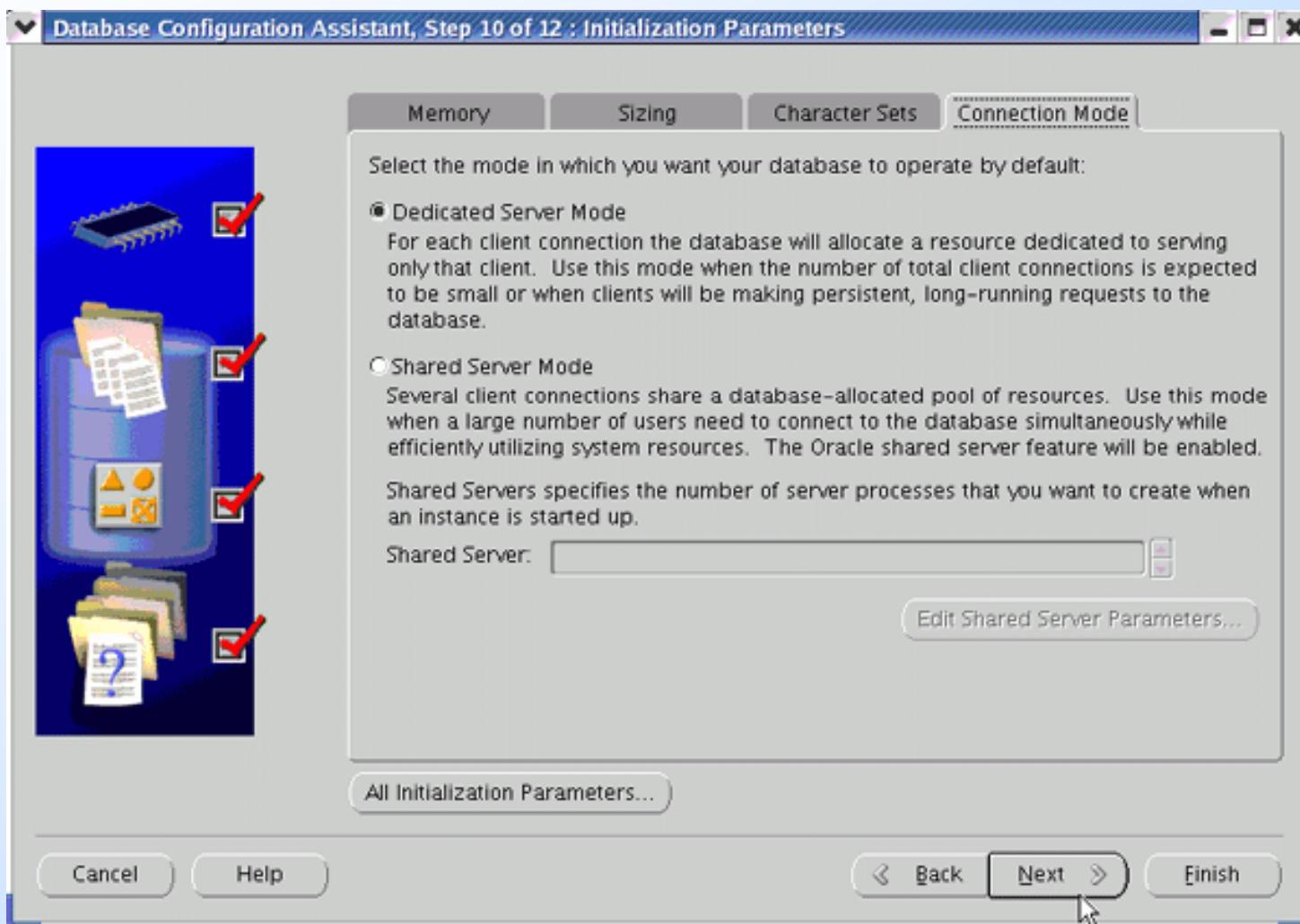
# DBCA: 10 - cont



# DBCA: 10 - cont

- ◆ Tabul Character Set specifica si setul de caractere utilizat pentru acea baza de date (VARCHAR2, CLOB). Se pot selecta:
  - ◆ Default – ia setul limbii implicite a SO pentru toti utilizatorii BD respective
  - ◆ Unicode (AL32UTF8) pentru a putea folosi mai multe seturi de caractere (pentru useri si aplicatiile lor)
  - ◆ Alegere din lista: ca prima optiune, dar se specifica setul prin alegere din lista
- ◆ Se mai pot specifica National Character Set (NVARCHAR2, NCLOB), Default Language si Default Date Format

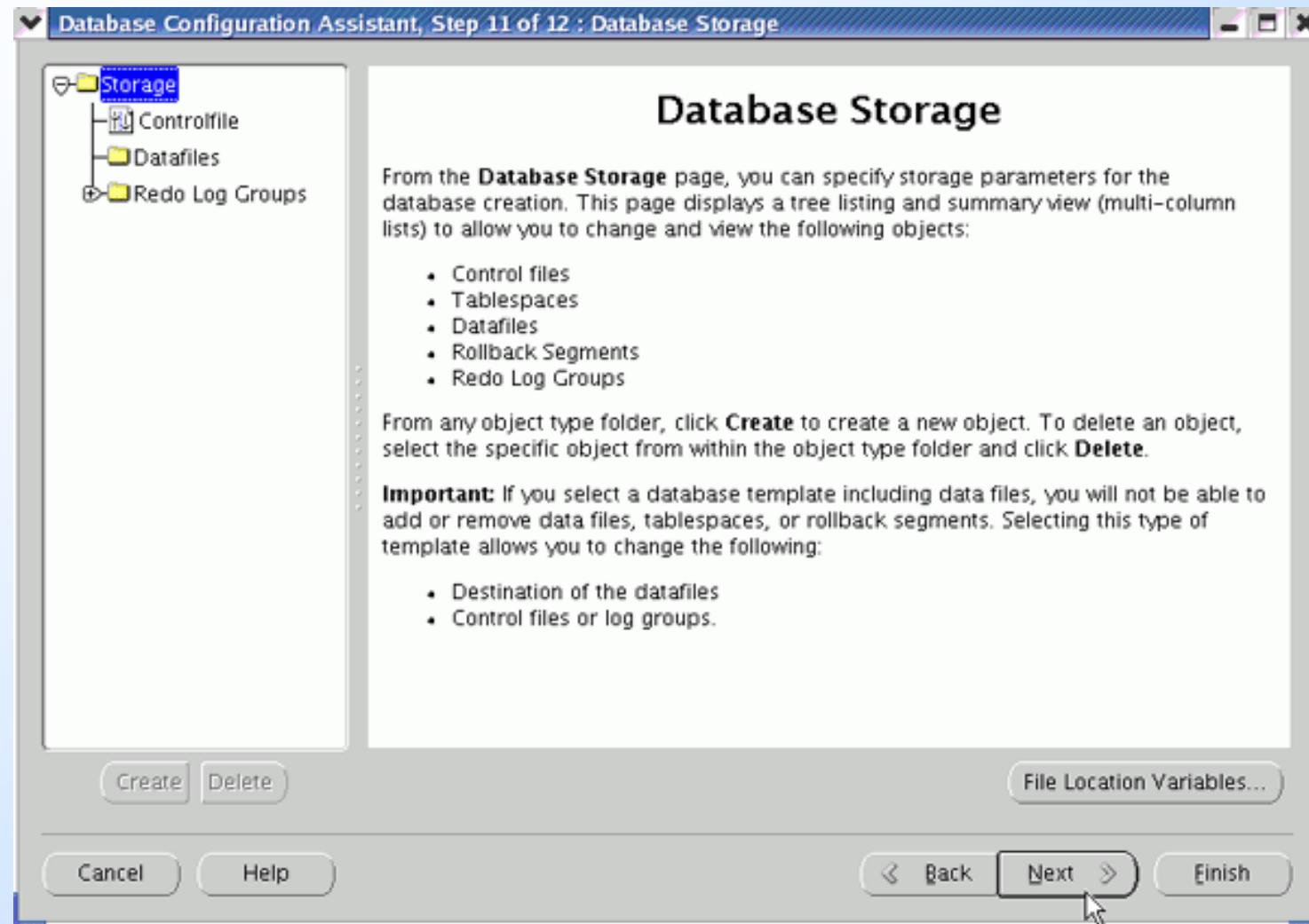
# DBCA: 10 - cont



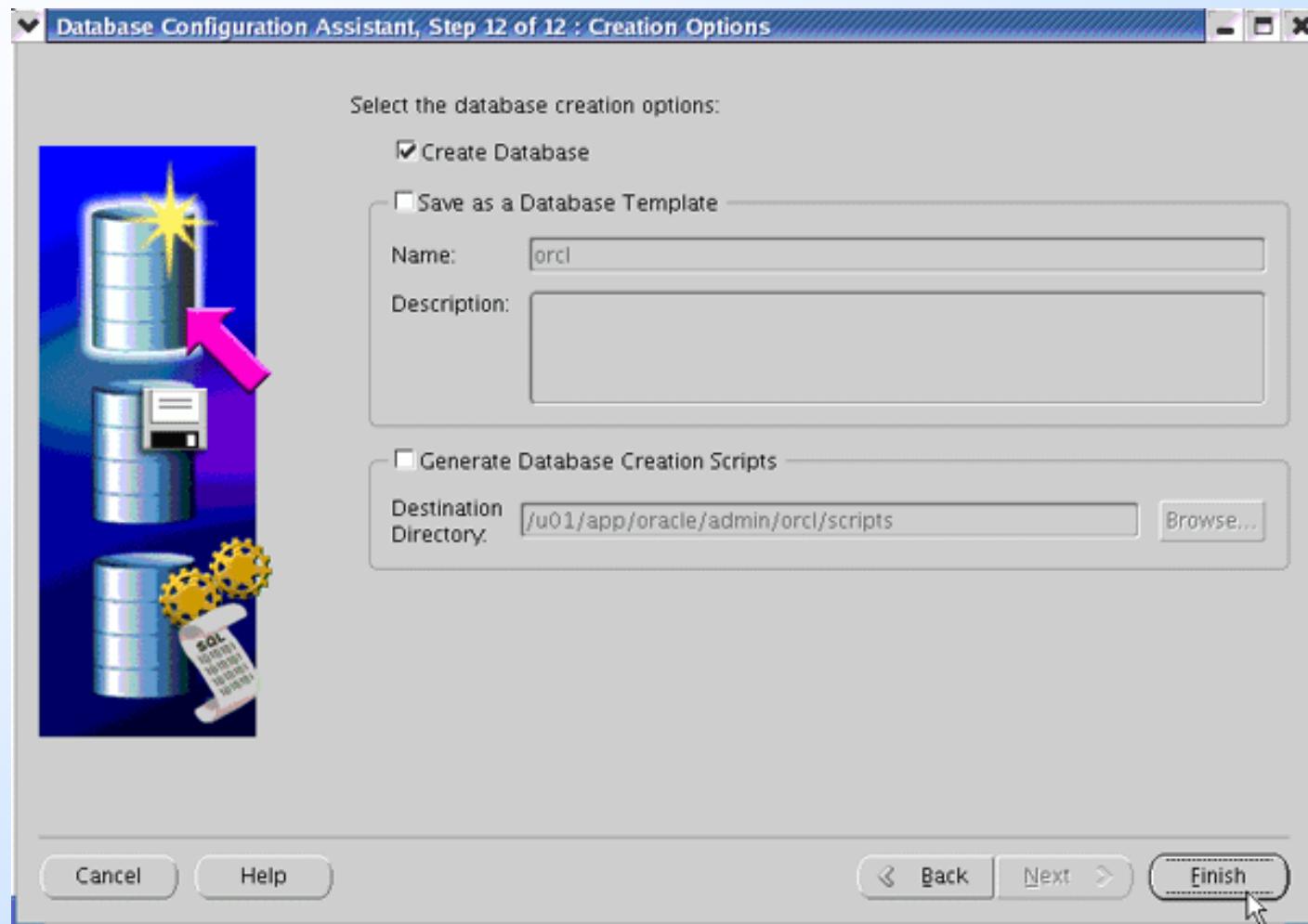
# DBCA: 10 - cont

- ◆ Tabul Connection Mode:
  - ◆ Dedicated Server – fiecare proces server este pentru un proces user. Se foloseste cand numarul de clienti nu e foarte mare sau cand clientii sunt conectati mult timp la baza de date (cereri care ruleaza mult timp)
  - ◆ Shared Server – mai multe procese client sunt deservite de acelasi proces server. Se foloseste cand memoria e limitata sau numarul de clienti este mare

# DBCA: 11. Parametri de stocare



# DBCA: 12. Ultima etapa



# DBCA: 12. - cont

- ◆ Sabloanele (a doua optiune) sunt fisiere XML care contin informatii pentru a crea o baza de date.
- ◆ Oracle pune la dispozitie niste sabloane pre-definite (cele de la pasul 2)
- ◆ Aceste sabloane se pot crea (cu DBCA):
  - ◆ Dintr-un alt sablon
  - ◆ Dintr-o baza de date existenta (doar structura acesteia e folosita, schemele user sunt ignotare)
  - ◆ Dintr-o baza de date existenta, folosindu-se si datele user existente)

# Crearea manuala a BD

- ◆ Pasii de urmat sunt urmatorii: (vezi documentul  
[http://download.oracle.com/docs/cd/B14117\\_01/server.101/b10739/create.htm](http://download.oracle.com/docs/cd/B14117_01/server.101/b10739/create.htm) )
- 1. Alegerea numelui instantei (SID)
- 2. Stabilirea metodei de autentificare a administratorului (OS sau fisier de parole)
- 3. Crearea fisierului de parametri (Initialization Parameter File)
- 4. Conectarea la instantă
- 5. Crearea fisierului de parametri server (Server Parameter File)
- 6. Pornirea instantei
- 7. Executia cererii CREATE DATABASE
- 8. Crearea de Tablespace aditionale
- 9. Rularea scripturilor de creare a vederilor din Dictionarul de date
- 10. Rularea scripturilor de instalare a optiunilor (Optional)
- 11. Salvarea bazei de date astfel create (back up)

# Exemplu pasul 7

```
CREATE DATABASE bazamea
USER SYS IDENTIFIED BY sys543
USER SYSTEM IDENTIFIED BY system555
LOGFILE GROUP 1 ('/u01/oracle/oradata/mynewdb/redo01.log') SIZE 100M,
          GROUP 2 ('/u01/oracle/oradata/mynewdb/redo02.log') SIZE 100M,
          GROUP 3 ('/u01/oracle/oradata/mynewdb/redo03.log') SIZE 100M
MAXLOGFILES 5
MAXLOGMEMBERS 5
MAXLOGHISTORY 1
MAXDATAFILES 100
MAXINSTANCES 1
CHARACTER SET US7ASCII
NATIONAL CHARACTER SET AL16UTF16
DATAFILE '/u01/oracle/oradata/mynewdb/system01.dbf' SIZE 325M REUSE
EXTENT MANAGEMENT LOCAL
SYSAUX DATAFILE '/u01/oracle/oradata/mynewdb/sysaux01.dbf' SIZE 325M REUSE
DEFAULT TABLESPACE tbs_1
DEFAULT TEMPORARY TABLESPACE tempts1
        TEMPFILE '/u01/oracle/oradata/mynewdb/temp01.dbf'
        SIZE 20M REUSE
UNDO TABLESPACE undotbs
        DATAFILE '/u01/oracle/oradata/mynewdb/undotbs01.dbf'
        SIZE 200M REUSE AUTOEXTEND ON MAXSIZE UNLIMITED;
```

# Exemplu pasul 7 - cont

Efectul este:

- ◆ Se creaza o baza de date cu numele ***bazamea*** (SID-ul creat la pasul 1 este acelasi)
- ◆ Sunt create fisierele de control specificate (ca nume) in fisierul de initializare (pasul 3) la CONTROL\_FILES
- ◆ Sunt setate parolele pentru userii privilegiati SYS si SYSTEM (sys543 respectiv system555). In cazul in care aceste clauze lipsesc se pun valorile implice ***change\_on\_install*** si ***manager***
- ◆ Noua baza de date va avea in cazul din exemplu 3 fisiere de tip Redo log, specificatia lor fiind in clauza LOGFILE
- ◆ MAXDATAFILES specifica numarul maxim de fisiere de date care pot fi deschise in baza de date

# Lecturi obligatorii

## 1. Din documentul: Colin McGregor - Oracle Database 2 Day DBA, 10g

Link: [http://docs.oracle.com/cd/B28359\\_01/server.111/b28301.pdf](http://docs.oracle.com/cd/B28359_01/server.111/b28301.pdf)

- ◆ Capitolul 2 (Installing Oracle and Building the Database)

## 2. Crearea manuala a unei baze de date, descrisa in pagina:

[http://download.oracle.com/docs/cd/B14117\\_01/server.101/b10739/create.htm#i1008760](http://download.oracle.com/docs/cd/B14117_01/server.101/b10739/create.htm#i1008760)

# Sfârșitul capitolului 2

# Capitolul 3

## Fisiere: Control, Redo Log

# Continut capitol

Ca structura fizica, baza de date contine fisiere de control, de date si de Redo log.

Ca structura logica o baza de date se compune din:

Tablespace  $\supset$  Segment  $\supset$  Extent  $\supset$  Bloc (stocate in fisierele de date)

In acest capitol vom discuta despre:

- Fisierele de control
- Fisierele de Redo Log

In capitolul urmator vor fi detaliate:

- Tablespace (element in structura logica a fisierelor de date)
- Fisierele de date

# Fisier de control

- Este un fisier binar, folosit atat la pornirea bazei de date cat si in timpul operarii cu aceasta.
- Este deschis si citit la montarea bazei de date (vezi cap. precedent) pentru a localiza fisierele de date si fisierele de Redo log.
- Este actualizat permanent si trebuie sa fie disponibil pe intreaga perioada in care baza de date este montata si deschisa.

# Fisier de control - cont

- Contine informatii de consistenta necesare cand se face restaurarea dupa incident
- Daca vreunul dintre fisierele de control nu este disponibil functionarea bazei de date este afectata.

# Continut fisier de control

- Numele bazei de date
- Numele si localizarea fisierelor de date si de Redo log
- Numele pentru tablespace-uri
- Eticheta timp (timestamp) de la crearea bazei de date
- Numarul de secventa pentru fisierele Redo log
- Informatii despre punctele de checkpoint
- Etc.

# Important

- Montarea bazei de date (si ulterior deschiderea) se poate face doar daca fisierele de control sunt disponibile.
- In caz contrar, desi toate celelalte fisiere (date, Redo log) pot exista si pot fi consistente, baza de date nu poate trece de pasul MOUNT si sunt necesare scenarii de restaurare (din salvari ale fisierelor de control de exemplu – vom vedea tot azi cum).

## Important - cont

- Teoretic este posibil sa avem un singur fisier de control dar aceasta varianta este descurajata de Oracle.
- Varianta optima este de a avea mai multe copii exploataate in paralel (au acelasi continut - terminologia Oracle: multiplexate) chiar si in cazul in care masina gazda are un singur disc fizic
- In felul acesta este prevenita stergerea accidentală - exemple de acest fel sunt destule

# Parametrii

- În fisierul de parametri - init.ora - există parametrul CONTROL\_FILES care primește ca valoare o multime de nume de fisiere de control (care vor fi exploataate în paralel):

```
CONTROL_FILES =
  (/u01/oracle/prod/control01.ctl,
   /u02/oracle/prod/control02.ctl,
   /u03/oracle/prod/control03.ctl)
```

- Aceste fisiere sunt create automat de Oracle la crearea bazei de date

# Mutare / redenumire

- Pentru a redenumi/muta un fisier de control trebuie urmati urmatorii pasi:
  1. Oprire (Shutdown) baza de date.
  2. Copierea unui fisier de control existent in noua locatie / sub noul nume folosind comenziile SO
  3. Editarea fisierului de parametri de initializare si schimbarea corespunzatoare a parametrului CONTROL\_FILES.
  4. Repornirea bazei de date.

# Creare nou fisier de control

Crearea unui nou fisier de control poate fi necesara in cazurile:

1. Toate fisierele de control ale BD sunt distruse si nu exista salvari ale lor.
2. Se doreste schimbarea unor parametri permanenti ai BD (specificati la CREATE DATABASE) cum ar fi numele bazei de date sau valoarea parametrilor MAXLOGFILES, MAXLOGMEMBERS, MAXLOGHISTORY, MAXDATAFILES si MAXINSTANCES.

# Cererea CREATE CONTROLFILE

**CREATE CONTROLFILE**

**SET DATABASE stud**

**LOGFILE**

**GROUP 1 ('/dsk1/oracle/stud/redo01\_01.log', '/dsk1/oracle/stud/redo01\_02.log'),**

**GROUP 2 ('/dsk1/oracle/stud/redo02\_01.log', '/dsk1/oracle/stud/redo02\_02.log'),**

**GROUP 3 ('/dsk1/oracle/stud/redo03\_01.log', '/dsk1/oracle/stud/redo03\_02.log')**

**RESETLOGS**

**DATAFILE '/dsk1/oracle/stud/system01.dbf' SIZE 3M, '/dsk1/oracle/stud/rbs01.dbs' SIZE  
    5M, '/dsk1/oracle/stud/users01.dbs' SIZE 5M, '/dsk1/oracle/stud/temp01.dbs' SIZE 5M**

**MAXLOGFILES 50**

**MAXLOGMEMBERS 3**

**MAXLOGHISTORY 400**

**MAXDATAFILES 200**

**MAXINSTANCES 6**

**ARCHIVELOG;**

***Se presupune ca baza de date exista si anterior dar cu alt nume.***

# Etape creare nou FC

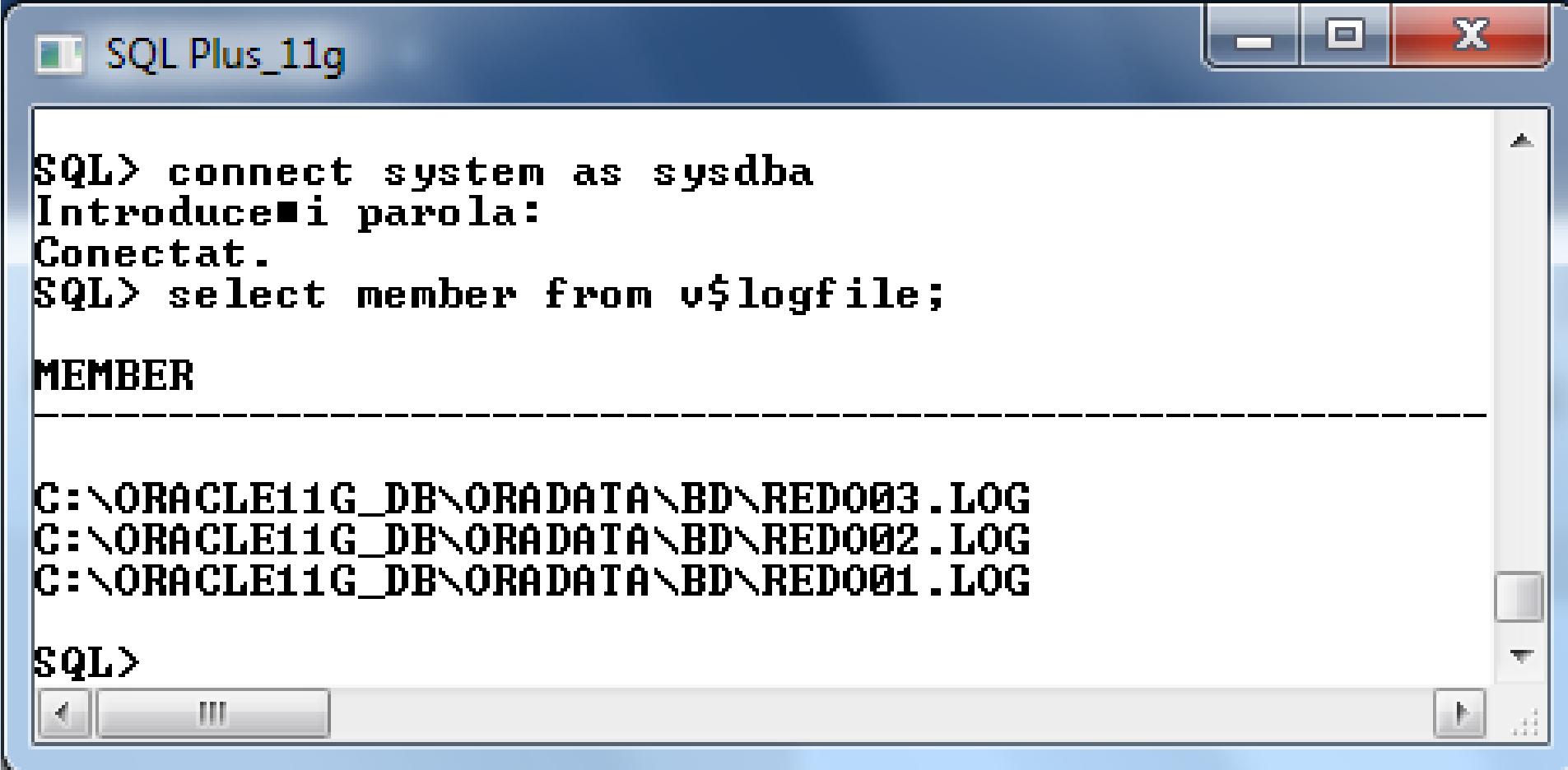
Pasul 1. Se face o lista cu TOATE fisierele de date si Redo Log ale bazei de date.

Cat timp BD este functionala aceste fisiere se pot obtine si ca rezultat al cererilor:

```
SELECT MEMBER FROM V$LOGFILE;  
SELECT NAME FROM V$DATAFILE;  
SELECT VALUE FROM V$PARAMETER  
  WHERE NAME = 'control_files';
```

Atentie: omiterea vreunui fisier atunci cand se va executa CREATE CONTROLFILE poate duce la pierderi iremediabile de date sau ale intregii baze de date (de exemplu daca nu se specifica fisierul in care se afla tablespace-ul pentru SYSTEM).

## SELECT MEMBER FROM V\$LOGFILE;



The screenshot shows a Windows window titled "SQL Plus\_11g". Inside, a SQL session is running:

```
SQL> connect system as sysdba
Introduceți parola:
Conectat.
SQL> select member from v$logfile;

MEMBER
-----
C:\ORACLE11G_DB\ORADATA\BD\REDO03.LOG
C:\ORACLE11G_DB\ORADATA\BD\REDO02.LOG
C:\ORACLE11G_DB\ORADATA\BD\REDO01.LOG

SQL>
```

# SELECT NAME FROM V\$DATAFILE;

The screenshot shows a Windows application window titled "SQL Plus\_11g". Inside, an SQL command is being run:

```
SQL> SELECT NAME FROM V$DATAFILE;
```

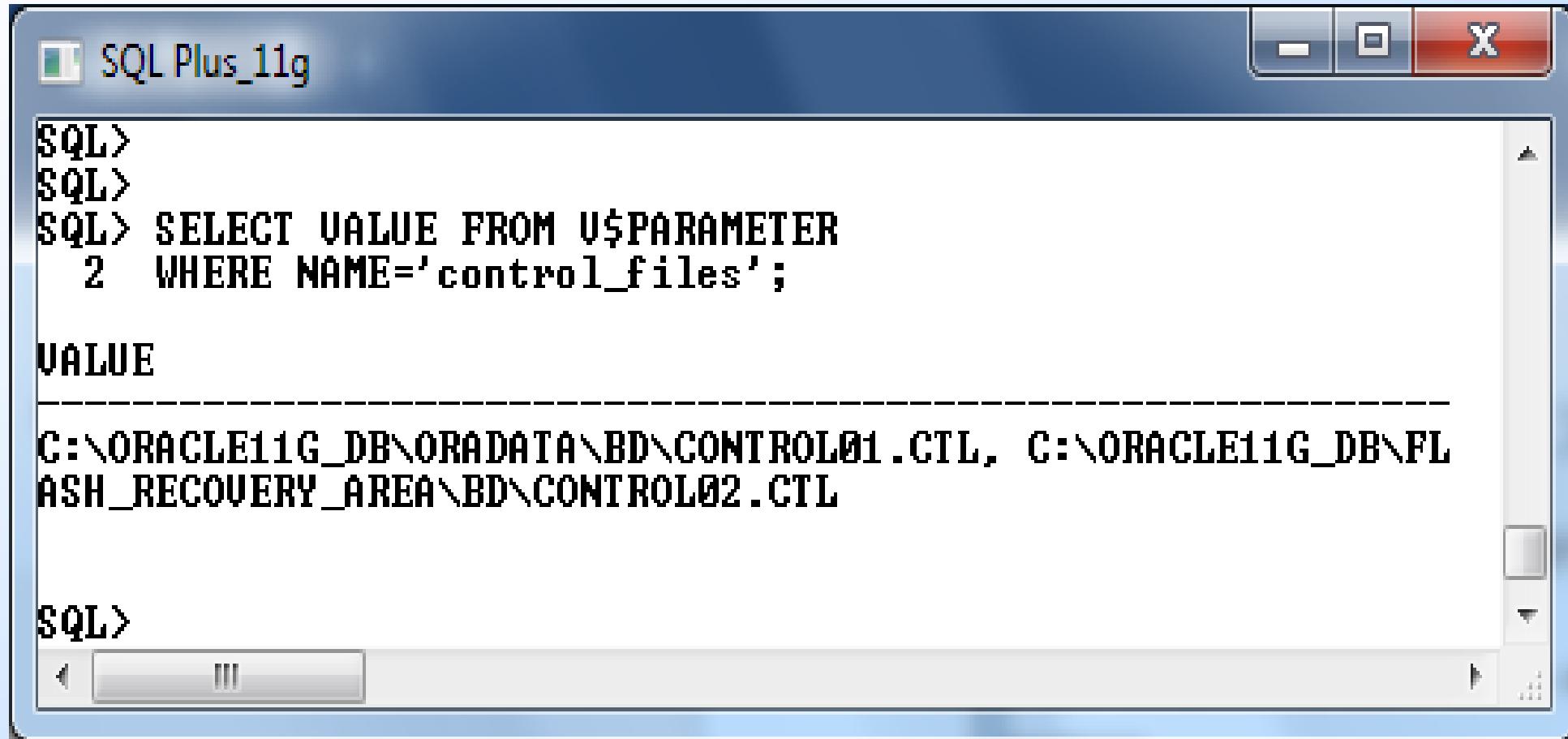
The output displays the names of various data files located on the C:\ORACLE11G\_DB\ORADATA\BD directory. The results are as follows:

NAME
C:\ORACLE11G_DB\ORADATA\BD\SYSTEM01.DBF
C:\ORACLE11G_DB\ORADATA\BD\SYSAUX01.DBF
C:\ORACLE11G_DB\ORADATA\BD\UNDOTBS01.DBF
C:\ORACLE11G_DB\ORADATA\BD\USERS01.DBF
C:\ORACLE11G_DB\ORADATA\BD\EXAMPLE01.DBF
C:\ORACLE11G_DB\ORADATA\BD\BD_DATA.DBF
C:\ORACLE11G_DB\PRODUCT\11.2.0\DBHOME_1\DATABASE\REPOS_GR_IDX
C:\ORACLE11G_DB\PRODUCT\11.2.0\DBHOME_1\DATABASE\REPOS_GR_TAB
C:\ORACLE11G_DB\PRODUCT\11.2.0\DBHOME_1\DATABASE\REPOS_DEP_IDX
C:\ORACLE11G_DB\PRODUCT\11.2.0\DBHOME_1\DATABASE\REPOS_DEP_TAB
C:\ORACLE11G_DB\PRODUCT\11.2.0\DBHOME_1\DATABASE\REPOS_DIA_IDX
C:\ORACLE11G_DB\PRODUCT\11.2.0\DBHOME_1\DATABASE\REPOS_DIA_TAB
C:\ORACLE11G_DB\PRODUCT\11.2.0\DBHOME_1\DATABASE\REPOS_LOB_DATA
C:\ORACLE11G_DB\PRODUCT\11.2.0\DBHOME_1\DATABASE\RAP_GR_IDX
C:\ORACLE11G_DB\PRODUCT\11.2.0\DBHOME_1\DATABASE\REPOS_SYS_META_IDX
C:\ORACLE11G_DB\PRODUCT\11.2.0\DBHOME_1\DATABASE\REPOS_SYS_META_TAB
C:\ORACLE11G_DB\PRODUCT\11.2.0\DBHOME_1\DATABASE\REPOS_TEMP_IDX
C:\ORACLE11G_DB\PRODUCT\11.2.0\DBHOME_1\DATABASE\REPOS_TEMP_TAB
C:\ORACLE11G_DB\PRODUCT\11.2.0\DBHOME_1\DATABASE\REPOS_VER_IDX
C:\ORACLE11G_DB\PRODUCT\11.2.0\DBHOME_1\DATABASE\REPOS_VER_TAB

20 înregistrări selectate.

```
SQL> _
```

## SELECT VALUE FROM V\$PARAMETER...



The screenshot shows a Windows application window titled "SQL Plus\_11g". Inside, the SQL command:

```
SQL>
SQL>
SQL> SELECT VALUE FROM V$PARAMETER
  2 WHERE NAME='control_files';
```

is executed. The output is displayed in a table format:

NAME	VALUE
control_files	C:\ORACLE11G_DB\ORADATA\BD\CONTROL01.CTL, C:\ORACLE11G_DB\FLASH_RECOVERY_AREA\BD\CONTROL02.CTL

At the bottom of the window, there is a toolbar with several icons.

# Etape creare nou FC - cont

Pasul 2. Oprirea bazei de date. De preferat ca aceasta oprire sa fie una normala (si nu IMMEDIATE sau ABORT)

Pasul 3. Salvarea tuturor fisierelor de date si de control (cele identificate la pasul 1)

Pasul 4. Se porneste o instanta dar nu se monteaza si nu se deschide baza de date (STARTUP NOMOUNT)

Pasul 5. Se creeaza noul fisier de control cu cererea SQL CREATE CONTROLFILE (exemplu in slide anterior). Se foloseste clauza RESETLOGS daca s-a redenumit baza sau s-au pierdut si fisiere de Redo log o data cu cele de control. Altfel se foloseste NORESETLOGS. Fisierul va fi create in locatia indicate de parametrul CONTROL\_FILES.

Pasul 6. Se salveaza undeva in siguranta fisierul creat (ex.: pe un CD)

# Etape creare nou FC - cont

Pasul 7. Se editeaza fisierul cu parametri de initializare pentru a contine numele specificate la pasul 5. Daca s-a redenumit si baza de date, se modifica si parametrul DB\_NAME

Pasul 8. Se recupereaza dupa incident – recovery - (daca este cazul) baza de date

Pasul 9 – si ultimul – se deschide baza de date cu:

**ALTER DATABASE OPEN;**

sau cu

**ALTER DATABASE OPEN RESETLOGS;**

in functie de optiunea RESETLOGS absenta sau prezenta la crearea fisierului de control (pasul 5)

# Salvare FC

- Se face cu:

**ALTER DATABASE BACKUP CONTROLFILE**

Exemplu:

1. In cazul in care se doreste salvarea unei copii (fisiere binare) a fisierelor de control putem da comanda:

**ALTER DATABASE BACKUP CONTROLFILE TO  
'/oracle/backup/control.bkp';**

2. In cazul in care se doreste producerea unei secvente de cereri SQL (text deci) care sa poata fi ulterior folosite pentru recrearea fisierului de control:

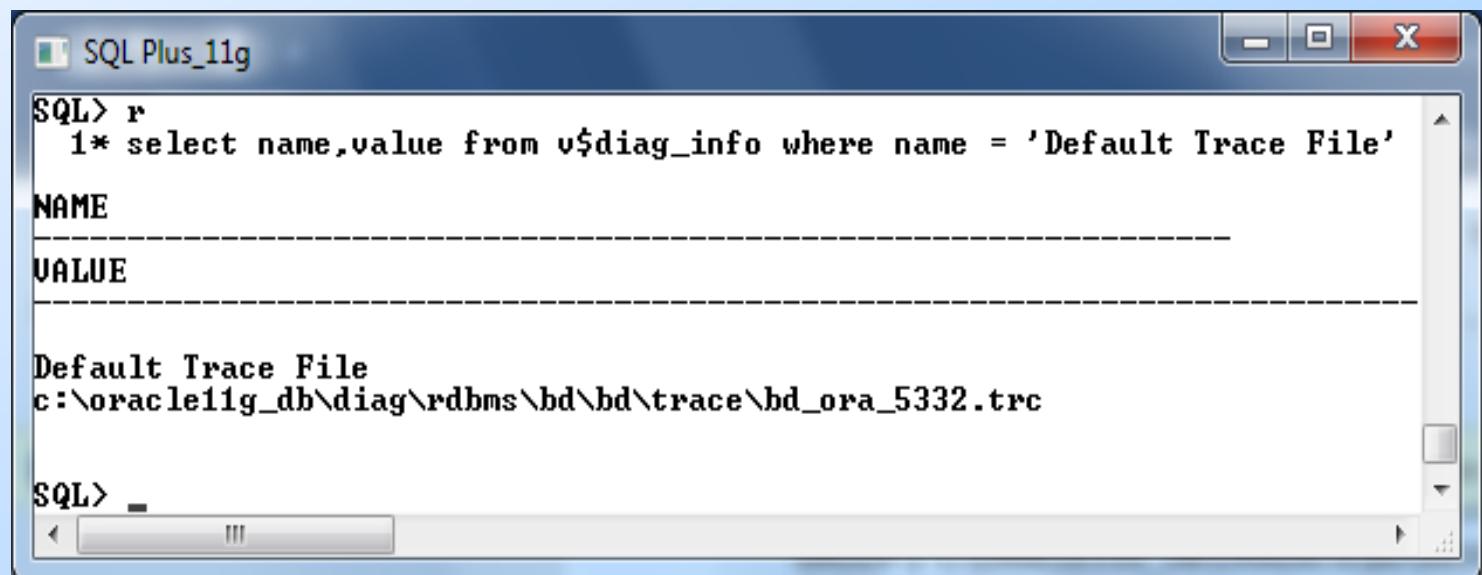
**ALTER DATABASE BACKUP CONTROLFILE TO TRACE;**

Efectul: in fisierul TRACE vor fi generate comenzile SQL respective

# Salvare FC

- Locatia fisierului TRACE folosit in sesiunea curenta se poate afla cu cererea:

```
SELECT VALUE  
FROM V$DIAG_INFO  
WHERE NAME='Default Trace File'
```



The screenshot shows an Oracle SQL Plus window titled "SQL Plus\_11g". The window contains the following text:

```
SQL> r  
 1* select name,value from v$diag_info where name = 'Default Trace File'  
  
NAME  
-----  
VALUE  
-----  
  
Default Trace File  
c:\oracle11g_db\diag\rdbms\bd\bd\trace\bd_ora_5332.trc  
  
SQL> _
```

# Salvare FC

In fisierul TRACE vom gasi:

```
CREATE CONTROLFILE REUSE DATABASE "BD" RESETLOGS NOARCHIVELOG
  MAXLOGFILES 16
  MAXLOGMEMBERS 3
  MAXDATAFILES 100
  MAXINSTANCES 8
  MAXLOGHISTORY 292
LOGFILE
  GROUP 1 'C:\ORACLE11G_DB\ORADATA\BD\REDO01.LOG' SIZE 50M BLOCKSIZE 512,
  GROUP 2 'C:\ORACLE11G_DB\ORADATA\BD\REDO02.LOG' SIZE 50M BLOCKSIZE 512,
  GROUP 3 'C:\ORACLE11G_DB\ORADATA\BD\REDO03.LOG' SIZE 50M BLOCKSIZE 512
-- STANDBY LOGFILE
DATAFILE
  'C:\ORACLE11G_DB\ORADATA\BD\SYSTEM01.DBF',
  'C:\ORACLE11G_DB\ORADATA\BD\SYSAUX01.DBF',
  'C:\ORACLE11G_DB\ORADATA\BD\UNDOTBS01.DBF',
  'C:\ORACLE11G_DB\ORADATA\BD\USERS01.DBF',
  'C:\ORACLE11G_DB\ORADATA\BD\EXAMPLE01.DBF',
  'C:\ORACLE11G_DB\ORADATA\BD\BD_DATA.DBF',
```

# Salvare FC

```
'C:\ORACLE11G_DB\PRODUCT\11.2.0\DBHOME_1\DATABASE\REPOS_GR_IDX',
'C:\ORACLE11G_DB\PRODUCT\11.2.0\DBHOME_1\DATABASE\REPOS_GR_TAB',
'C:\ORACLE11G_DB\PRODUCT\11.2.0\DBHOME_1\DATABASE\REPOS_DEP_IDX',
'C:\ORACLE11G_DB\PRODUCT\11.2.0\DBHOME_1\DATABASE\REPOS_DEP_TAB',
'C:\ORACLE11G_DB\PRODUCT\11.2.0\DBHOME_1\DATABASE\REPOS_DIA_IDX',
'C:\ORACLE11G_DB\PRODUCT\11.2.0\DBHOME_1\DATABASE\REPOS_DIA_TAB',
'C:\ORACLE11G_DB\PRODUCT\11.2.0\DBHOME_1\DATABASE\REPOS_LOB_DATA',
'C:\ORACLE11G_DB\PRODUCT\11.2.0\DBHOME_1\DATABASE\RAP_GR_IDX',
'C:\ORACLE11G_DB\PRODUCT\11.2.0\DBHOME_1\DATABASE\REPOS_SYS_META_IDX',
'C:\ORACLE11G_DB\PRODUCT\11.2.0\DBHOME_1\DATABASE\REPOS_SYS_META_TAB',
'C:\ORACLE11G_DB\PRODUCT\11.2.0\DBHOME_1\DATABASE\REPOS_TEMP_IDX',
'C:\ORACLE11G_DB\PRODUCT\11.2.0\DBHOME_1\DATABASE\REPOS_TEMP_TAB',
'C:\ORACLE11G_DB\PRODUCT\11.2.0\DBHOME_1\DATABASE\REPOS_VER_IDX',
'C:\ORACLE11G_DB\PRODUCT\11.2.0\DBHOME_1\DATABASE\REPOS_VER_TAB'
```

**CHARACTER SET EE8MSWIN1250**

;

# Restaurare din copie

Cazul 1. In cazul in care exista mai multe copii (multiplexate) si doar una din ele s-a distrus:

- Cu instanta **oprita** se copiaza fizic (comenzi OS) un fisier bun peste cel care a fost distrus:

```
cp /dsk3/oracle/stud/control03.ctl /dsk2/oracle/stud/control02.ctl
```

2. Se reporneste (STARTUP)

# Restaurare din copie - cont

Cazul 2: dispozitivul pe care se gaseste FC este avariat permanent

1. Cu instanta **oprita** se copiaza un fisier (bun) de control la o locatie accesibila:

**cp /dsk3/oracle/stud/control03.ctl /dsk14/oracle/stud/control02.ctl**

2. Se modifica in fisierul de parametri de initializare CONTROL\_FILES astfel incat sa se inlocuiasca locatia defecta cu cea noua (fisierul inaccesibil cu cel obtinut la pasul 1)
3. Se reporneste (STARTUP)

# Stergerea unui FC

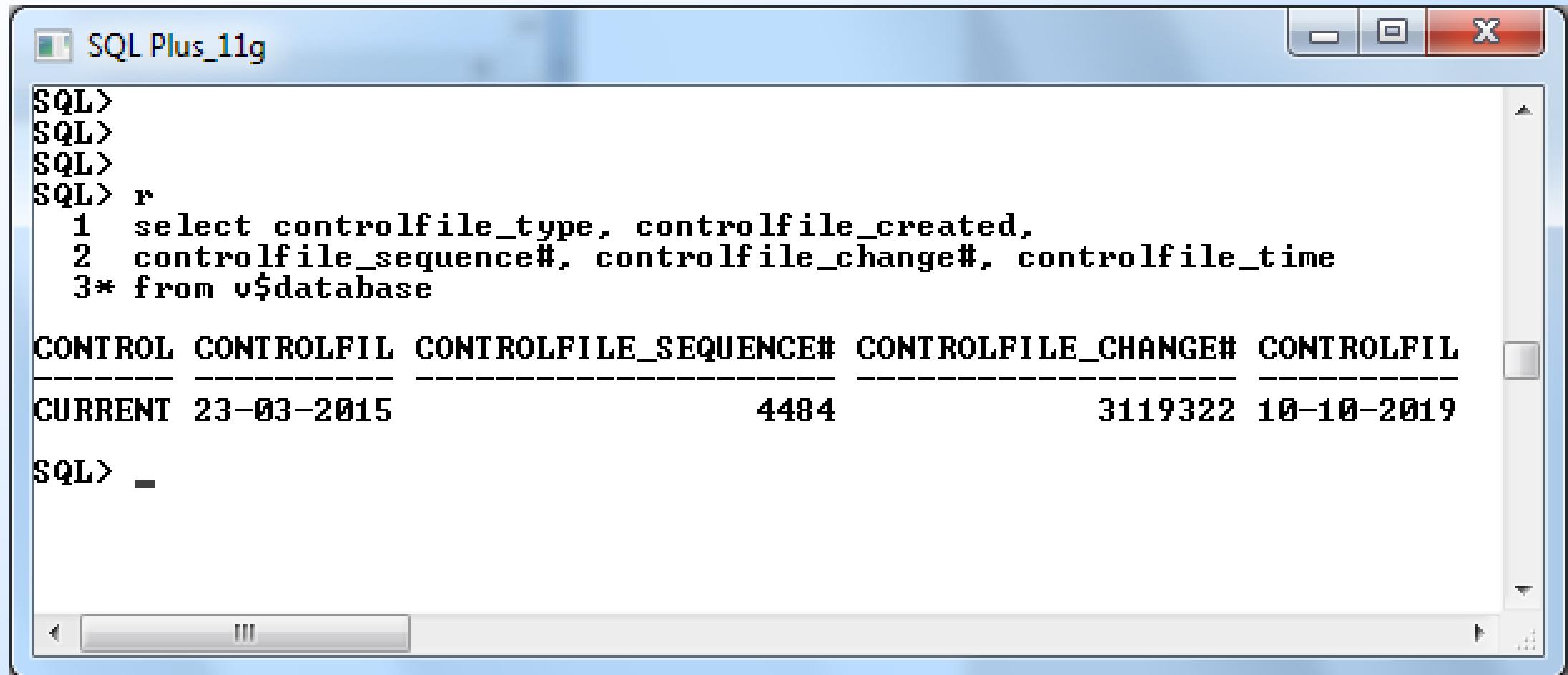
- Observatie: orice BD trebuie sa aiba permanent 2 FC.
- Se pot sterge fisiere cand sunt mai multe sau cand se creeaza noi fisiere care le inlocuiesc pe altele mai vechi (de exemplu ca in cazul anterior cand un dispozitiv existentiese din uz).
- Pasii sunt:
  1. Oprire baza
  2. Editare fisier de parametri pentru a elimina din CONTROL\_FILES fisierul care se sterge
  3. Repornire.

# Vederi care contin date despre FC

1. V\$DATABASE – contine date despre baza de date (luate din FC)
2. V\$CONTROLFILE, V\$PARAMETER – contin lista cu numele FC

Mai exista si alte vederi care returneaza date privind continutul FC.

# V\$DATABASE



The screenshot shows a Windows-style window titled "SQL Plus\_11g". Inside, a SQL command is run against the "v\$database" view:

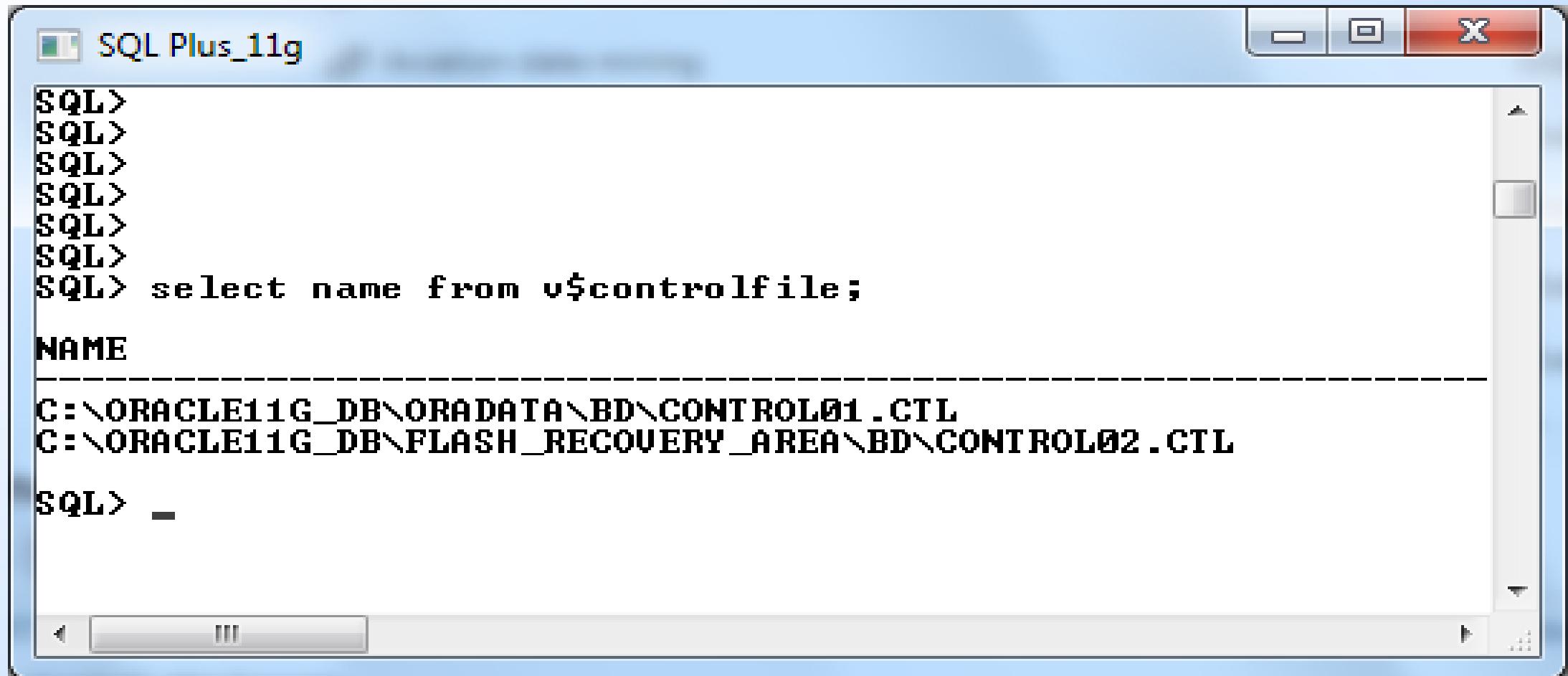
```
SQL>
SQL>
SQL>
SQL> r
  1 select controlfile_type, controlfile_created,
  2 controlfile_sequence#, controlfile_change#, controlfile_time
  3* from v$database
```

The output displays the current control file information:

CONTROL	CONTROLFIL	CONTROLFILE_SEQUENCE#	CONTROLFILE_CHANGE#	CONTROLFIL
CURRENT	23-03-2015	4484	3119322	10-10-2019

```
SQL> _
```

# V\$CONTROLFILE



The screenshot shows a window titled "SQL Plus\_11g". Inside, the SQL command "select name from v\$controlfile;" is run, followed by the output:

```
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL> select name from v$controlfile;

NAME
-----
C:\ORACLE11G_DB\ORADATA\BD\CONTROL01.CTL
C:\ORACLE11G_DB\FLASH_RECOVERY_AREA\BD\CONTROL02.CTL

SQL> -
```

# REDO LOG FILES

- ❑ Li se mai spune si ‘fisiere jurnal’ in cazul altor sisteme de gestiune.
- ❑ In ele se inregistreaza toate modificarile facute in Buffer Cache
- ❑ Se folosesc pentru recuperarea tranzactiilor comise ale caror date nu au fost inca scrise pe disc pana in momentul incidentului (deci se folosesc DOAR pentru recuperarea datelor – recovery).

# Grupuri si membri

- ❑ Fisierele de tip Redo Log folosite la un moment dat de sistem sunt impartite in grupuri, un grup putand contine mai multi membrii.
- ❑ Este de preferat ca membrii unui grup sa fie plasati pe dispozitive diferite pentru a evita pierderi in caz de incident de dispozitiv.
- ❑ Procesul care scrie aceste fisiere este LGWR (Log Writer, unul dintre procesele de background ale unei instante)

# Grupuri si membri – cont

- ❑ Toti membrii (=fisiere) unui grup au dimensiune identica si sunt scrisi in paralel de LGWR
- ❑ Toti membrii unui grup au acelasi numar de secventa (log sequence number). Acesta este un numar dat de Oracle atunci cand incepe sa scrie intr-un grup.
- ❑ Numarul de secventa curent (este unul singur!) este memorat si in fisierele de control si in antetul fisierelor de date.

# Cate grupuri sunt

- ❑ Pentru operarea normala a bazei de date Oracle are nevoie de minimul 2 grupuri de fisiere Redo Log.
- ❑ Pot fi maximul 255 de grupuri diferite.

Observatie: Termenul in engleza este 'online redo log file' pentru ca mai pot exista si fisiere de acest tip arhivate (vom vedea in continuare).

# PARAMETRI

- ❑ In CREATE DATABASE se folosesc urmatorii parametri:
  - ❑ MAXLOGFILES – numarul maxim de grupuri (asa cum am spus e  $\leq 255$ )
  - ❑ MAXLOGMEMBERS – numarul maxim de membrii per grup
- ❑ In fisierul de parametrii exista LOG\_FILES care specifica numarul de fisiere care sunt deschise la run-time (si care poate fi maxim egal cu produsul celor 2 valori maxime de mai sus).

# Utilizare

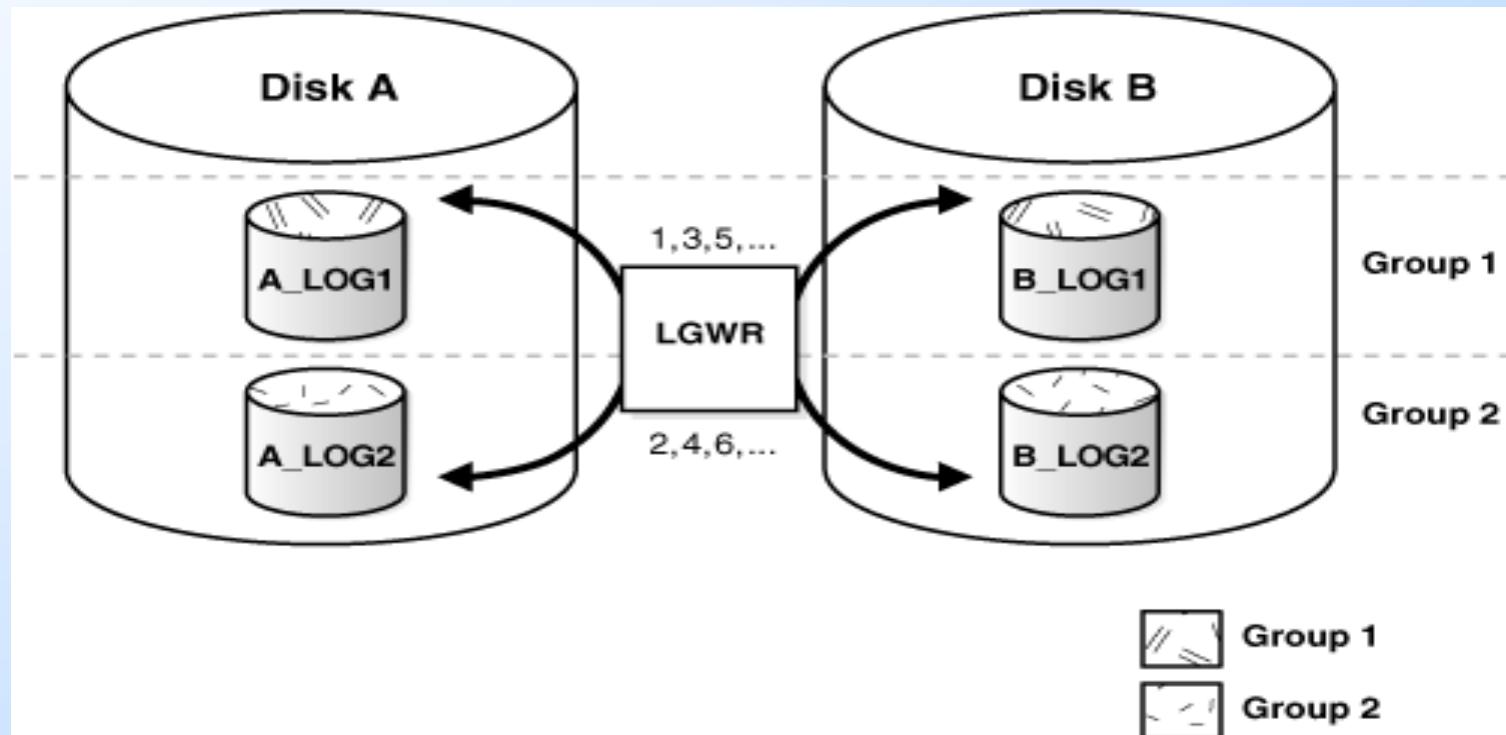
- ❑ Oracle inregistreaza secvential toate schimbarile facute in baza de date in Redo Log Buffer sub forma unor inregistrari de Redo.
- ❑ Bufferul este folosit circular (cand se ajunge la capat se reia cu inceputul).
- ❑ Din Buffer aceste inregistrari sunt scrise in fisierul (=grupul) curent de Redo de catre procesul LGWR in urmatoarele cazuri:

# Utilizare - cont

1. In momentul in care apare un COMMIT (s-a vorbit de asta la un capitol precedent)
2. In momentul in care Redo log buffer este plin intr-o anumita proportie
3. Cand apare un time-out al LGWR (la fiecare 3 secunde)
4. Inainte ca DBWR sa scrie blocurile de date modificate in fisierele de date

# Schimbare grup Redo

- Cand un fisier (=grup) se umple se trece la urmatorul fisier (sunt cel putin 2)



# Schimbare grup Redo - cont

- ❑ Aceasta schimbare se numeste in documentatia Oracle 'Log switch'
- ❑ DBA-ul poate forta o astfel de schimbare si in cazul in care fisierul curent nu e plin
- ❑ La fiecare log switch Oracle asociaza un nou numar de secventa noului fisier (=grup)
- ❑ Cand apare un log switch este de asemenea initiat si un checkpoint:

# Checkpoint

□ La aparitia unui checkpoint se executa urmatoarele operatii

- Toate blocurile de date modificate (dirty buffers) din memorie care sunt monitorizate de fisierul de Redo Log respectiv sunt scrise pe disc de catre DBWR
- Procesul de checkpoint (CKPT) actualizeaza antetele tuturor fisierelor de control si date pentru a reflecta schimbarea produsa.

# Checkpoint - cont

## ❑ Un checkpoint apare:

- La fiecare log switch
- La oprirea instantei in modurile normal, tranzactional si imediat
- In mod fortat prin setarea parametrilor:  
`LOG_CHECKPOINT_INTERVAL` si  
`LOG_CHECKPOINT_TIMEOUT`
- Cand e cerut de administratorul bazei de date

## ❑ Informatiile despre fiecare checkpoint sunt stocate in fisierul de alerte (dar doar daca parametrul `LOG_CHECKPOINTS_TO_ALERT` este setat pe TRUE)

# Clasificare

❑ Fisierele de Redo Log se pot clasifica in

1. CURRENT - Curent – cel in care se scrie la un moment dat
2. ACTIVE - Activ – s-a scris in el anterior dar modificarile cuprinse in el nu sunt inca scrise in fisierele de date si deci e necesar pentru recuperare instanta.
3. INACTIVE - Inactiv – s-a scris in el anterior si modificarile s-au inregistrat si in fisierele de date, deci nu mai e necesar pentru recuperare instanta
4. UNUSED – Nou – Nu s-a scris niciodata in el pana acum (probabil un fisier nou adaugat)

# Arhivare

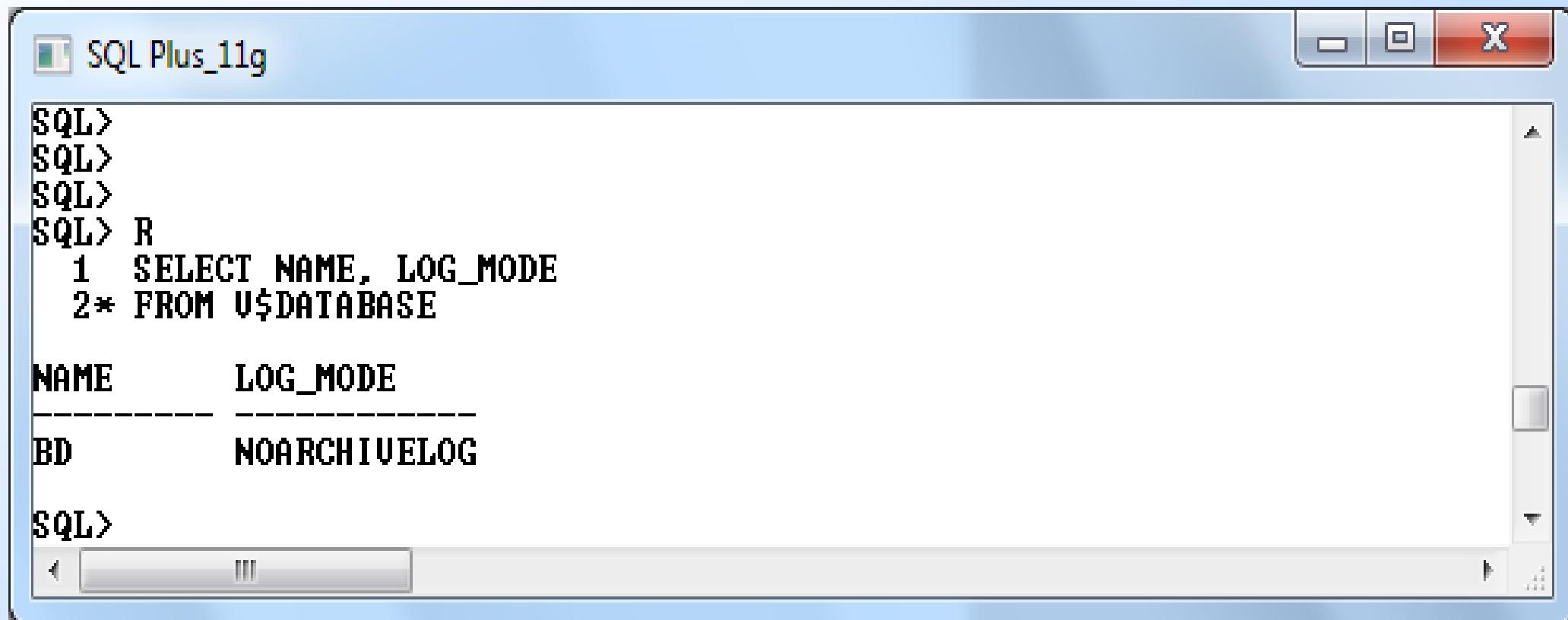
- Se poate dispune ca fisierele de tip Redo Log File sa fie arhivate de sistemul de gestiune
- Arhivarea acestor fisiere permite refacerea bazei de date de la 0 pornind de la o salvare la un moment dat si de la fisierele de tip Redo Log care au inregistrat modificarile facute in BD dupa salvarea respectiva.
- In cazul in care fisierele nu sunt arhivate este posibila in continuare recuperarea instantei dupa incident (pentru asta sunt necesare doar fisierele curente si cele active de Redo Log)

# Arhivare - cont

- ❑ O aceeasi baza de date poate fi la un moment dat intr-unul din cele 2 moduri:
  1. NOARCHIVELOG – in momentul in care ultimul fisier de Redo Log se umple se revine la primul care este rescris (scris-peste). Bineintele modificarile din acesta au fost scrise si in fisierele de date
  2. ARCHIVELOG – dupa un log switch procesele ARCH (de background) arhiveaza fisierele de log inactive.

# Vederi: v\$database

- Pentru a vizualiza modul in care este BD: vederea **v\$database**



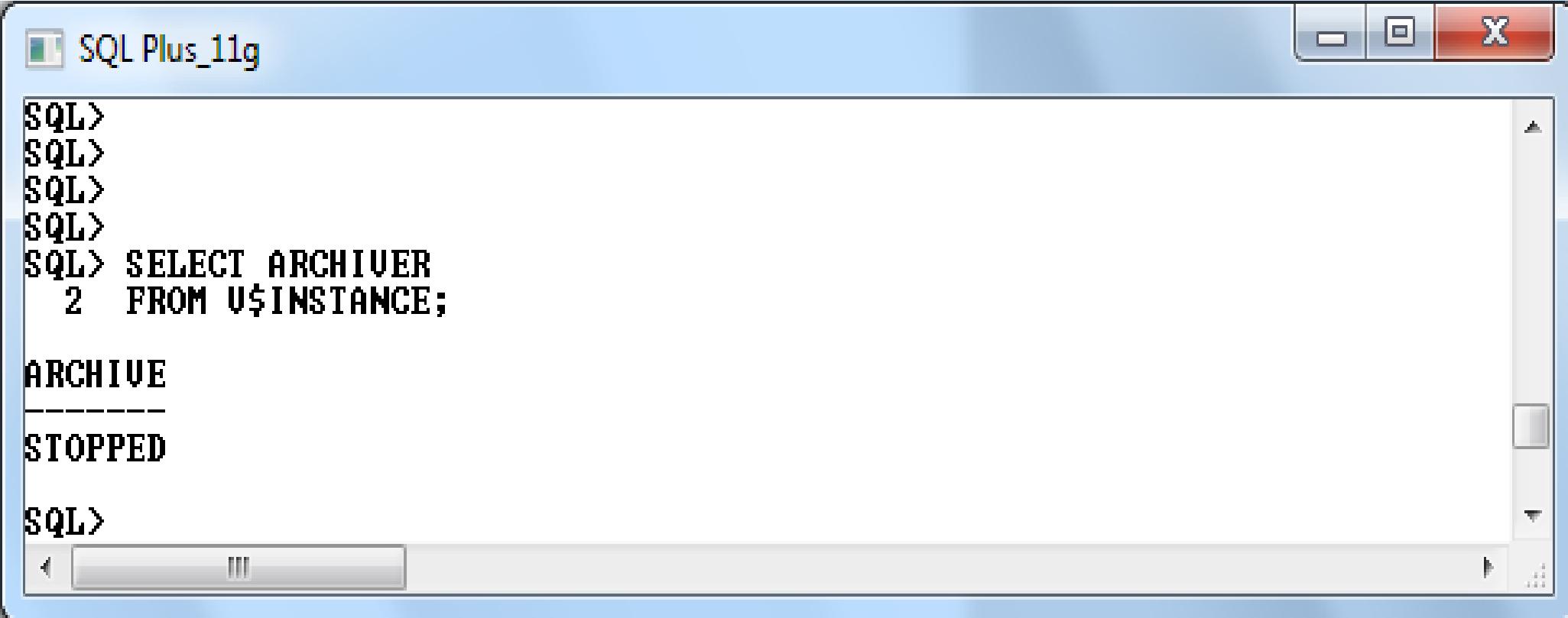
```
SQL>
SQL>
SQL>
SQL> R
  1  SELECT NAME, LOG_MODE
  2* FROM V$DATABASE

NAME        LOG_MODE
-----      -----
BD          NOARCHIVELOG

SQL>
```

# Vederi: v\$instance

- Pentru a vizualiza modul in care este BD: vederea **v\$instance**



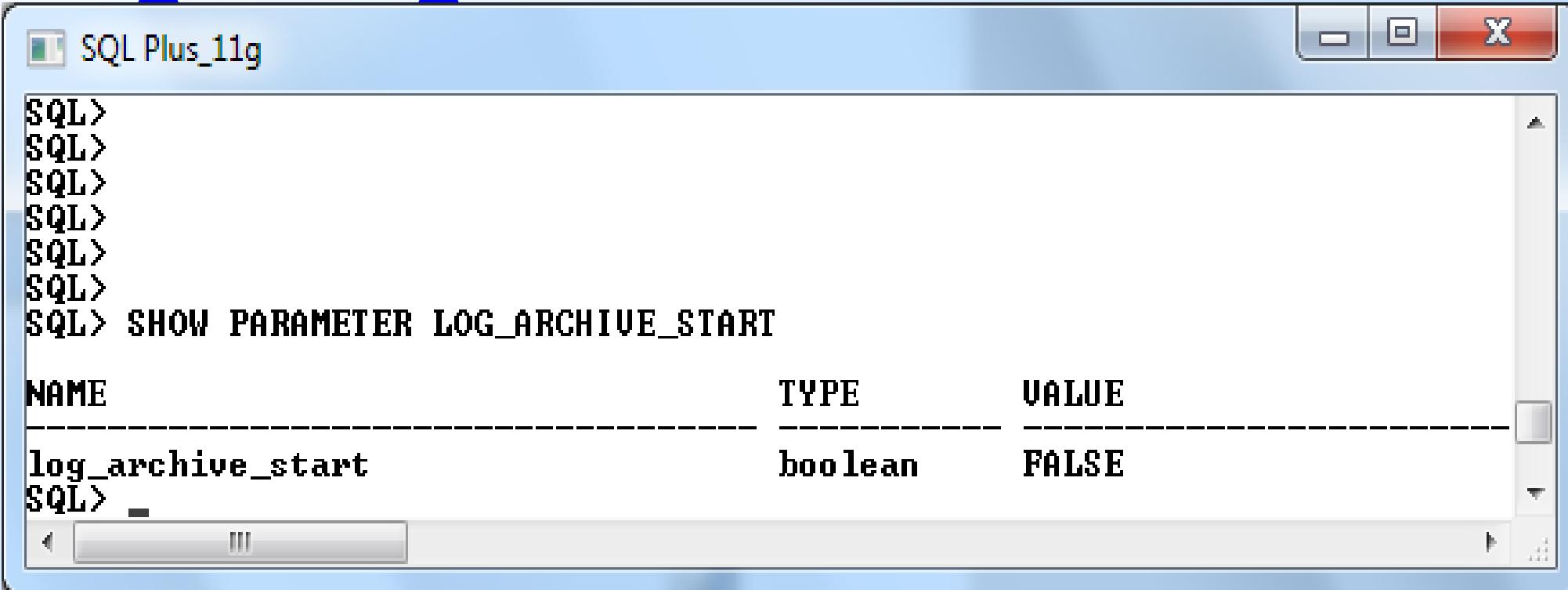
```
SQL>
SQL>
SQL>
SQL>
SQL> SELECT ARCHIVER
  2  FROM U$INSTANCE;

ARCHIVE
-----
STOPPED

SQL>
```

# Parametru

- Se poate si prin examinarea parametrului **log\_archive\_start** (comanda SQL\*Plus):

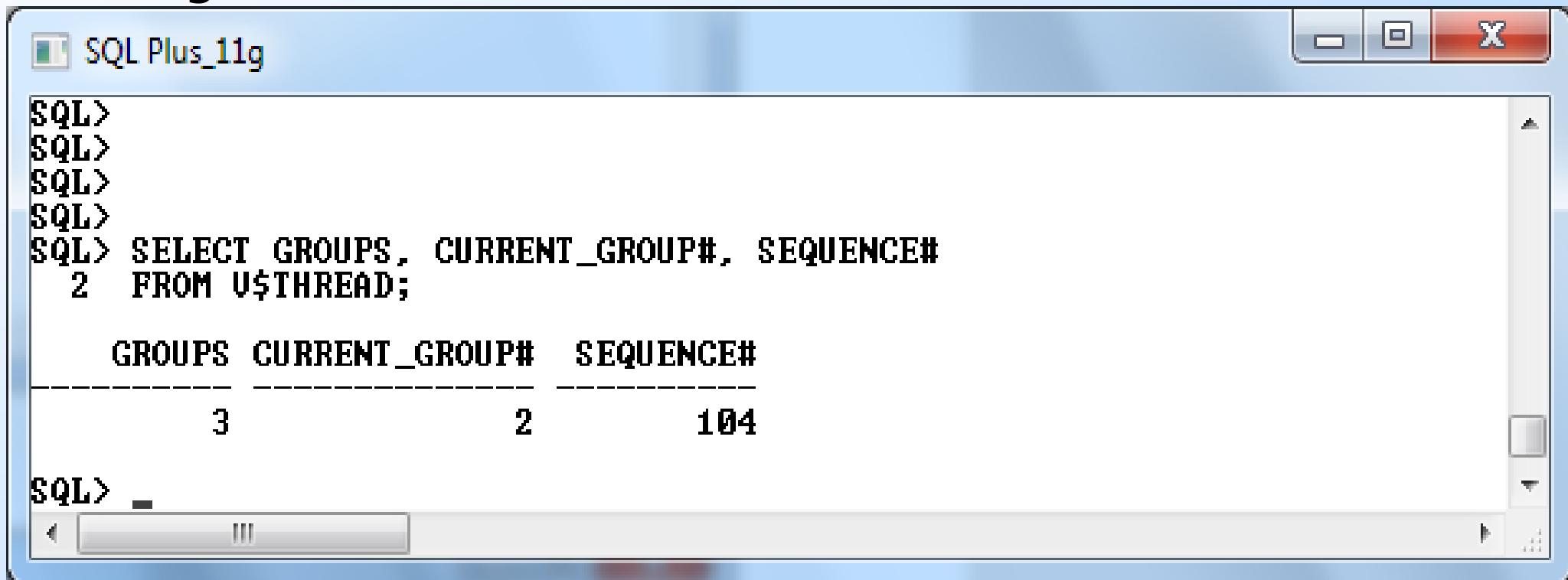


```
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL> SHOW PARAMETER LOG_ARCHIVE_START

NAME                           TYPE        VALUE
log_archive_start              boolean    FALSE
SQL> _
```

# Vederi: v\$thread

- Pentru a vizualiza care este fisierul (grupul) curent se poate interoga **v\$thread** :



The screenshot shows a Windows-style application window titled "SQL Plus\_11g". Inside, the SQL command line starts with "SQL>" followed by:

```
SQL>
SQL>
SQL>
SQL>
SQL> SELECT GROUPS, CURRENT_GROUP#, SEQUENCE#
  2  FROM V$THREAD;
```

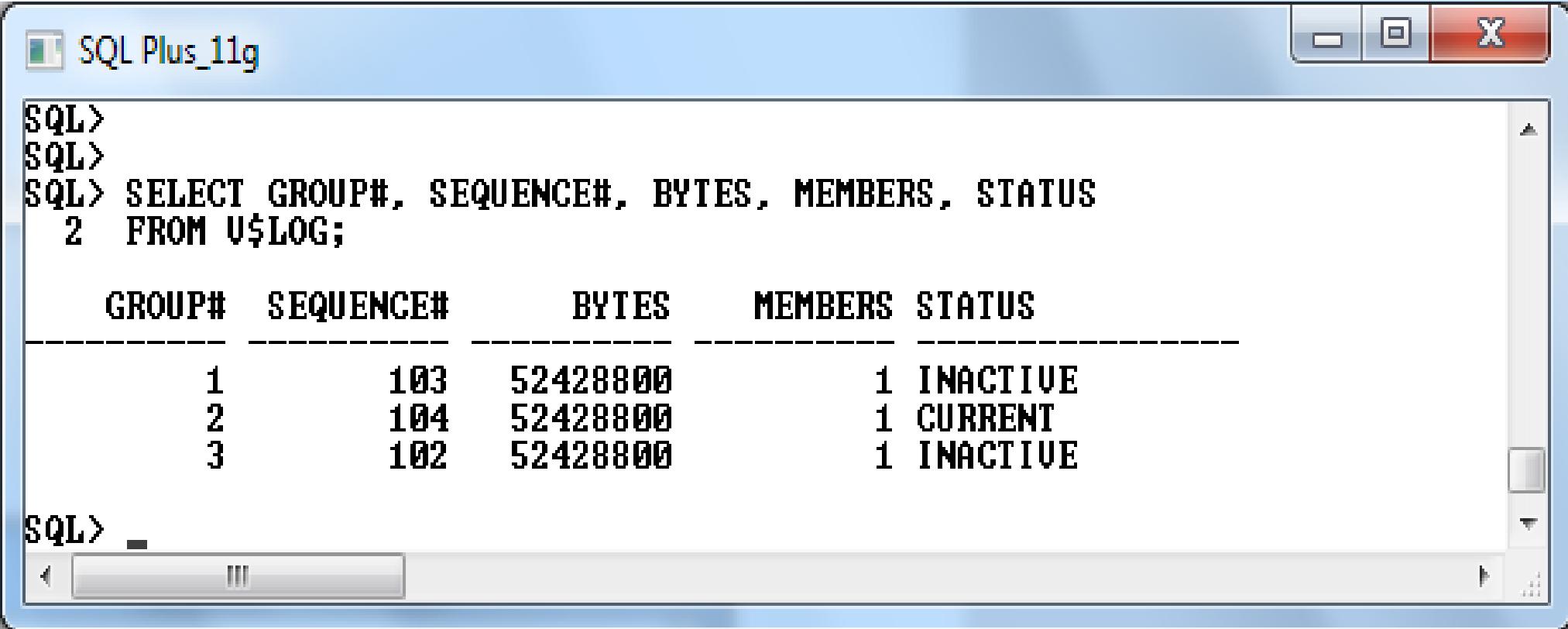
Below the command, the results are displayed in a tabular format:

GROUPS	CURRENT_GROUP#	SEQUENCE#
3	2	104

At the bottom of the window, there is another "SQL>" prompt.

# Vederi: v\$log

- Informatiile returnate sunt din fisierele de control:



The screenshot shows an Oracle SQL Plus window titled "SQL Plus\_11g". The window contains the following SQL command and its execution results:

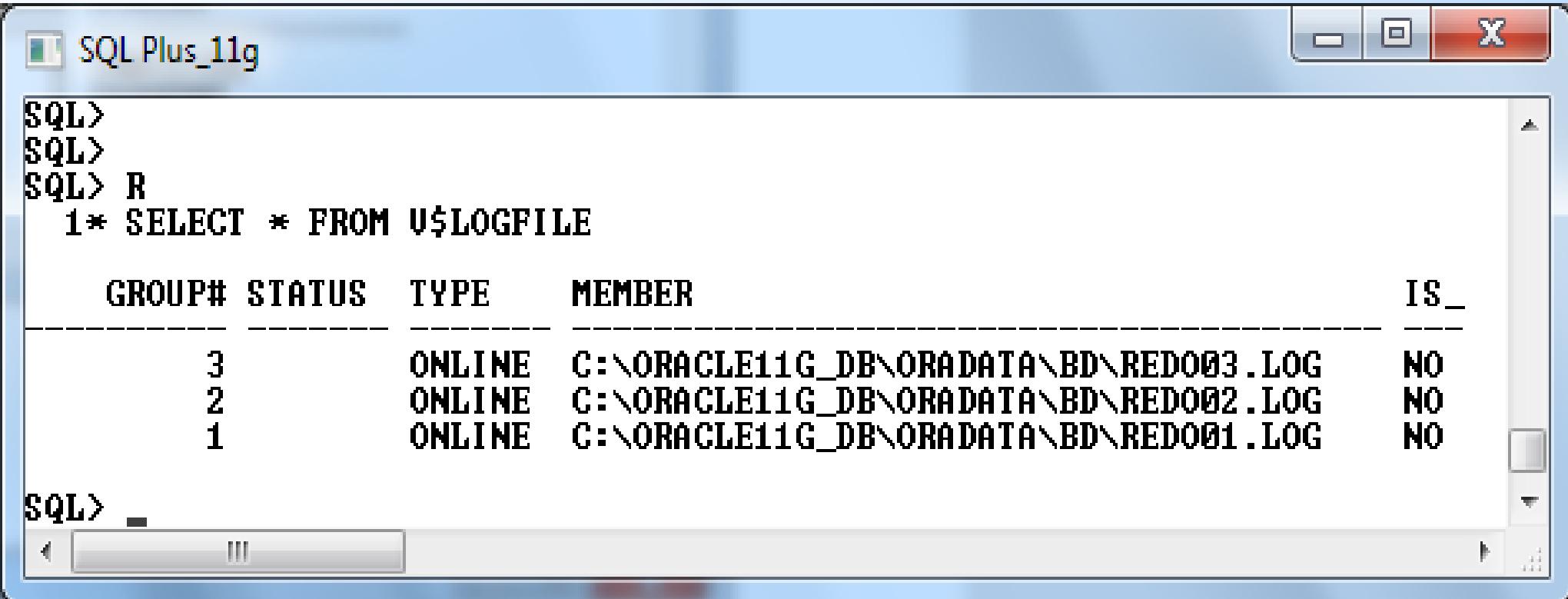
```
SQL>
SQL>
SQL> SELECT GROUP#, SEQUENCE#, BYTES, MEMBERS, STATUS
  2  FROM V$LOG;
```

GROUP#	SEQUENCE#	BYTES	MEMBERS	STATUS
1	103	52428800	1	INACTIVE
2	104	52428800	1	CURRENT
3	102	52428800	1	INACTIVE

SQL> \_

# Vederi: v\$logfile

- Coloana STATUS are valoarea NULL daca fisierul este utilizat si INVALID, STALE sau DELETED altfel.



The screenshot shows a window titled "SQL Plus\_11g". Inside, the SQL command "SELECT \* FROM V\$LOGFILE" is run, displaying the following results:

GROUP#	STATUS	TYPE	MEMBER	IS_
3	ONLINE		C:\ORACLE11G_DB\ORADATA\BD\REDO03.LOG	NO
2	ONLINE		C:\ORACLE11G_DB\ORADATA\BD\REDO02.LOG	NO
1	ONLINE		C:\ORACLE11G_DB\ORADATA\BD\REDO01.LOG	NO

# Vederi: v\$logfile

- ❑ Un fisier redo log devine INVALID daca sistemul nu il poate accesa.
- ❑ Un fisier redo log devine STALE daca sistemul suspecteaza ca nu este complet sau corect.
- ❑ Un fisier redo log STALE devine valid din nou cand grupul sau devine activ data viitoare.

# Fortarea unui log switch

□ Se face cu cererea:

**ALTER SYSTEM SWITCH LOGFILE ;**

Aceeași operatie se poate efectua și din consola de administrare

# Fortarea unui checkpoint

- ❑ Se face cu cererea:

**ALTER SYSTEM CHECKPOINT ;**

Precum si din consola de administrare

- ❑ De asemenea, in cazul in care baza de date foloseste fisiere de tip Redo Log mari, se poate comanda si efectuarea periodica a checkpointului prin parametrii :

**LOG\_CHECKPOINT\_INTERVAL** si

**LOG\_CHECKPOINT\_TIMEOUT**

# LOG\_CHECKPOINT\_INTERVAL

- ❑ Un checkpoint este initiat dupa ce LGWR a scris un numar de blocuri egal cu acest parametru (blocuri OS!)
- ❑ Cum orice log switch produce de asemenea checkpoint, daca parametrul este mai mare decat fisierul de Log checkpointul se va face doar la log switch
- ❑ Daca parametrul este 0 este ignorat (Oracle 10g). Aceasta este valoarea de default.

# **LOG\_CHECKPOINT\_TIMEOUT**

- ❑ Este o valoare specificata in secunde
- ❑ Se masoara de la inceputul precedentului checkpoint
- ❑ Valoarea 0 dezactiveaza declansarea de checkpoint-uri pe baza intervalelor de timp
- ❑ Valoarea de default este de 1800 (in Oracle 10g) deci 30 minute.
- ❑ Garanteaza ca nici un bloc modificat (dirty bloc) nu ramane in memorie mai mult de atatea secunde cat arata parametrul.

# Adaugare grup Redo Log

- Se face cu ALTER DATABASE:

```
ALTER DATABASE ADD LOGFILE GROUP 10
  ('/dsk01/oracle/dbs/log10a.rdo',
  '/dsk04/oracle/dbs/log10b.rdo') SIZE 500K;
```

- Specificarea numarului de grup este optionala (doar cand dorim sa le creem in alta ordine decat cea normala).
- Nu este bine sa creem grupuri ca 10, 20, 30, ... (pe sarite) pentru ca vom consuma inutil spatiu in fisierele de control

# Adaugare membrii

- Se face cu ALTER DATABASE. Exemplu:

```
ALTER DATABASE ADD LOGFILE MEMBER  
  '/dsk01/oracle/dbs/log2b.rdo'  
    TO GROUP 1,  
  '/dsk07/oracle/dbs/log2b.rdo'  
    TO GROUP 3;
```

- Se poate specifica grupul si prin membrii sai:

```
ALTER DATABASE ADD LOGFILE MEMBER  
  '/dsk06/oracle/dbs/log10c.rdo'  
    TO ('/dsk01/oracle/dbs/log10a.rdo',  
  '/dsk04/oracle/dbs/log10b.rdo');
```

# Adaugare membrii - cont

- Daca fisierul adaugat exista deja pe discul respectiv trebuie sa aiba dimensiunea necesara si se va specifica in plus clauza REUSE:

```
ALTER DATABASE ADD LOGFILE MEMBER  
' /dsk01/oracle/dbs/log2b.rdo ' REUSE  
TO GROUP 1 ;
```

# Redenumire / relocare

- ❑ In cazul acesta trebuie sa ne asiguram ca noul fisier (cu noul nume sau din noua locatie) exista.
- ❑ Oracle nu schimba decat informatii din fisierele de control fara sa redenumneasca sau sa creeze fizic fisiere din SO
- ❑ Etapele relocarii unui fisier de Redo Log sunt urmatoarele (valabile si la redenumire):

# Redenumire / relocate - cont

1. Oprirea bazei de date (SHUTDOWN)
2. Copierea fisierelor Redo Log in noua locatie. Se face cu comenzi OS.

Exemplu:

```
mv /diska/logs/log1a.rdo /diskc/logs/log1c.rdo  
mv /diska/logs/log2a.rdo /diskc/logs/log2c.rdo
```

3. Repornire in modul MOUNT:

```
CONNECT / as SYSDBA
```

```
STARTUP MOUNT
```

# Redenumire / relocate - cont

4. Redenumirea (in sistem) a fisierelor:

**ALTER DATABASE RENAME FILE**

```
'/diska/logs/log1a.rdo' , '/diska/logs/log2a.rdo' TO  
'/diskc/logs/log1c.rdo' , '/diskc/logs/log2c.rdo' ;
```

5. Deschiderea bazei:

**ALTER DATABASE OPEN**

# Stergere grup

- ❑ Trebuie sa ramana cel putin 2 grupuri (nu se poate sterge mai mult de atat)
- ❑ Un grup se poate sterge doar daca e INACTIVE.
- ❑ Daca se doreste stergerea grupului curent trebuie fortat un log switch.
- ❑ Grupul trebuie sa fie arhivat (daca arhivarea e pornita).
- ❑ Pentru a vedea starea grupurilor putem utiliza comanda:

# Stergere grup - cont

```
SELECT GROUP#, ARCHIVED, STATUS  
FROM V$LOG;
```

GROUP#	ARC	STATUS
1	YES	ACTIVE
2	NO	CURRENT
3	YES	INACTIVE
4	YES	INACTIVE

# Stergere grup - cont

- ❑ Stergerea efectiva se face cu ALTER DATABASE:

**ALTER DATABASE DROP LOGFILE GROUP 3;**

- ❑ Grupul se poate da nu numai ca numar ci si prin lista membrilor sai.
- ❑ Operatia de stergerea nu implica stergerea fisierelor de pe disc ci doar actualizarea informatiilor interne ale sistemului prin eliminarea grupului respectiv.
- ❑ Putem sa utilizam apoi comenzi SO pentru stergerea efectiva a fisierelor respective.

# Stergere membri

- ❑ Se poate sterge un membru doar daca nu este in grupul curent sau intr-un grup activ.
- ❑ Este bine ca grupul respectiv sa fie in acel moment arhivat
- ❑ Nu putem sterge ultimul membru valid al unui grup daca in felul acesta nu raman cel putin 2 grupuri continand membrii valizi.
- ❑ In cazul in care grupurile au cate doi membrii de exemplu, se poate sterge un membru din unul dintre ele dar este bine ca ulterior grupul sa fie completat (pentru siguranta in functionare)

# Stergere membri - cont

- ❑ Comanda este:

**ALTER DATABASE**

```
DROP LOGFILE MEMBER '/oracle/dbs/log3c.rdo';
```

- ❑ Si aici operatia nu implica stergerea fisierului de pe disc ci doar actualizarea informatiilor interne ale sistemului
- ❑ Dupa terminarea operatiei Oracle, putem sa utilizam comenzi SO pentru stergerea efectiva a fisierului respectiv.

# CLEAR LOGFILE

- ❑ In cazul in care un grup devine corupt el poate fi reinitializat cu comanda:

**ALTER DATABASE CLEAR LOGFILE GROUP 3;**

- ❑ Grupul se poate da ca numar sau ca lista de membri (intre paranteze).
- ❑ In cazul in care fisierul nu a fost arhivat trebuie mentionat in comanda:

**ALTER DATABASE CLEAR UNARCHIVED LOGFILE GROUP 3;**

- ❑ In acest caz insa recuperarea bazei din salvare+Loguri nu mai e posibila pentru salvarile care ar folosi si logul astfel initializat.

# Lecturi obligatorii

1. Oracle Database Administrator's Guide – Cap 5: Managing Control Files

[http://download.oracle.com/docs/cd/B14117\\_01/server.101/b10739/control.htm](http://download.oracle.com/docs/cd/B14117_01/server.101/b10739/control.htm)

2. Oracle Database Administrator's Guide – Cap 5: Managing the Redo Log

[http://download.oracle.com/docs/cd/B14117\\_01/server.101/b10739/onlineredo.htm](http://download.oracle.com/docs/cd/B14117_01/server.101/b10739/onlineredo.htm)

# Sfârșitul capitolului 3

# Capitolul 4

## Fisiere de date si Tablespace

# Continut capitol

Ca structura fizica, baza de date contine fisiere de control, de date si de Redo log.

Ca structura logica o baza de date se compune din:

Tablespace  $\supset$  Segment  $\supset$  Extensie (extent)  $\supset$   
Bloc (stocate in fisierele de date)

Din punct de vedere fizic avem:

Tablespace  $\supset$  Fisiere de date

In acest capitol vom discuta despre:

- ◆ Tablespace (element in structura logica a fisierelor de date)
- ◆ Fisierele de date

# Tablespace

- ◆ O baza de date contine (unul sau) mai multe subdiviziuni numite 'tablespace'.
- ◆ Un tablespace apartine unei singure baze de date.
- ◆ Un tablespace poate fi stocat in unul sau mai multe fisiere de date.
- ◆ Un fisier de date apartine unui singur tablespace.
- ◆ Cu unele exceptii (SYSTEM de ex.) un tablespace poate fi trecut intre starile
  - ◆ online ↔ offline si
  - ◆ read-write ↔ read-only

# Fisiere de date

- ◆ Aceste fisiere se creaza:
  - ◆ la crearea bazei de date (pentru tablespace-urile care sunt create atunci)
  - ◆ la crearea unui nou tablespace
  - ◆ La adaugarea unui nou fisier de date la un tablespace
- ◆ Dimensiunea fisierelor este specificata la creare.
- ◆ Unui fisier de date existent i se poate modifica dimensiunea ulterior.
- ◆ Unui fisier de date i se poate seta optiunea 'AUTOEXTEND' pentru a creste automat ca dimensiune cand este necesar.

# SEGMENTE

- ◆ Un tablespace contine segmente.
- ◆ Un segment contine un obiect (tabela, index, etc)
- ◆ Sunt de 4 tipuri generice (si 11 tipuri efective):
  - ◆ Segment de tip date (tabele si cluster)
  - ◆ Segment de tip index
  - ◆ Segment temporar
  - ◆ Segment de rollback
- ◆ Un segment se poate intinde pe mai multe fisiere de date care apartin aceluiasi tablespace.

# SEGMENTE

- ◆ Cele 11 tipuri de segmente sunt:
  1. table
  2. table partition
  3. index
  4. index partition
  5. cluster
  6. rollback
  7. deferred rollback
  8. temporary
  9. cache
  - 10.lobsegment
  - 11.lobindex

# SEGMENTE - cont

- ◆ Segmentele temporare sunt in general cele folosite pentru sortari.
- ◆ Urmatoarele cereri SQL au nevoie de segment temporar in cazul in care sortarile nu pot fi efectuate in memorie:
  - ◆ `create index`
  - ◆ `select ... order by`
  - ◆ `select distinct`
  - ◆ `select ... group by`
  - ◆ `select ... union`
  - ◆ `select ... intersect`
  - ◆ `select ... minus`
  - ◆ `analyze table`
  - ◆ joinuri care nu folosesc indecsi
  - ◆ anumite subcereri corelate

# SEGMENTE - cont

- ◆ Segmentele temporare se pot stoca in orice tablespace
- ◆ Exista insa posibilitatea de a crea un tablespace temporar (temporary tablespace)
- ◆ Segmentele temporare sunt eliberate dupa folosire de catre procesul de background SMON

# EXTENSII

- ◆ Un segment este format din una sau mai multe extensii (eng.: extent).
- ◆ O extensie e formata dintr-o succesiune ***contigua*** de blocuri de date (database blocks).
- ◆ O extensie se gaseste in intregime intr-un singur fisier de date dintre cele care formeaza tablespace-ul.
- ◆ Faptul ca este contigua este relevant pentru cresterea vitezei de exploatare a datelor (citire – scriere)

# EXTENSII

- ◆ Nota privind contiguitatea blocurilor unei extensii: acestea sunt blocuri logice, aparținând unui fișier de date (fișierul în care se găsește extensia).
- ◆ În documentația Oracle se precizează:
  - ◆ File system extents are not the same as Oracle Database extents.
  - ◆ File system extents are physical contiguous blocks of data written to a device as managed by the file system.
  - ◆ Oracle Database extents are logical structures managed by the database, such as tablespace extents.

([http://docs.oracle.com/cd/B28359\\_01/server.111/b28310/dfiles009.htm](http://docs.oracle.com/cd/B28359_01/server.111/b28310/dfiles009.htm))

# BLOC

- ◆ O extensie e formata din blocuri.
- ◆ Este vorba despre blocuri ale bazei de date (de dimensiune DB\_BLOCK\_SIZE)
- ◆ Un astfel de bloc poate fi format din unul sau mai multe blocuri fizice (de disc)
- ◆ Un bloc este cea mai mica unitate de intrare – iesire pentru SGBD.

# Revenim la TABLESPACE

- ◆ Avantajele folosirii mai multor tablespace-uri:
  - Se pot separa datele user de datele de sistem (prin stocarea in tablespace-uri diferite). In felul acesta se micsoreaza si traficul de date pe tablespace-urile de sistem.
  - Se pot separa datele unei aplicatii de ale altelui (prin stocarea in tablespace-uri diferite). In cazul in care un tablespace trece in starea offline – din diverse motive – doar o aplicatie va avea de suferit.
  - Se pot stoca pe discuri diferite, micsorand astfel traficul de date pentru fiecare disc in parte.

# Avantaje - cont

- Se poate optimiza utilizarea tablespace-urilor prin crearea de tablespace-uri dedicate:
  - Unele pentru aplicatii update-intensive
  - Altele pentru exploatare read-only
  - Altele pentru date temporare (segmente temporare)
- Se pot efectua operatii de salvare la nivel de tablespace deci se pot astfel salva doar datele aferente unor aplicatii importante care ruleaza in sistem.

# Tablespace-ul SYSTEM

- ◆ La crearea bazei de date se creaza automat tablespace-ul **SYSTEM** care contine printre altele dictionarul de date al sistemului si segmentul de rollback de sistem.
- ◆ Aceasta este primul tablespace creat si are caracteristici speciale:
  - ◆ Nu poate fi redenumit
  - ◆ Nu poate fi sters
  - ◆ Nu poate fi trecut in starea offline
  - ◆ Necesa privilegii sporite pentru operare

# Tablespace-ul SYSAUX

- ◆ La crearea bazei de date se creaza de asemenea si tablespace-ul SYSAUX care contine informatii despre schemele de date folosite de uneltele Oracle – astfel ele nu vor avea nevoie de un alt tablespace suplimentar.
- ◆ Aceasta are de asemenea caracteristici speciale:
  - ◆ Nu poate fi redenumit
  - ◆ Nu poate fi sters
  - ◆ Nu poate fi trecut in starea offline
  - ◆ Necesa privilegii sporite pentru operare

# Clasificare

- ◆ Un tablespace poate fi – din punct de vedere al datelor continute - de unul dintre urmatoarele tipuri:
  - ◆ Permanent – sunt tablespace-urile uzuale, inclusiv cele de sistem
  - ◆ Temporar – contin segmente temporare – am vorbit despre ele
  - ◆ De tip Undo – introduse incepand cu versiunea 9i – contin segmente de undo (rollback), necesare in cazul revocarii operatiilor de actualizare. Anterior versiunii 9i existau doar segmente de rollback – nu si tablespace.

# Alta clasificare: DMT si LMT

- ◆ Fiecare Tablespace este format dintr-o multime de extensii (continute in segmentele componente).
- ◆ Gestiunea acestora (care sunt libere si care sunt ocupate) se poate face in doua feluri: fie informatiile respective se stocheaza in dictionarul de date fie se memoreaza in tablespace
- ◆ Tablespace-urile pentru care gestiunea se face prin intermediul dictionarului de date (o solutie costisitoare ca timp) se numesc **DMT - dictionary managed tablespaces**

# DMT si LMT - cont

- ◆ Tablespace-urile pentru care gestiunea se face local, prin stocarea datelor privind starea extensiilor in interiorul tablespace-ului se numesc **LMT - locally managed tablespaces**
- ◆ Informatiile se tin in headerul acestuia, de fapt in headerul fiecarui fisier de date component
- ◆ In acest caz headerul contine un bitmap unde fiecare bit este un bloc sau un grup de blocuri. Bitul arata daca zona respectiva este ocupata sau nu.

# Exemplu

- ◆ Pentru a afla informatii despre tablespace-urile existente, se poate interoga vederea dba\_tablespaces din dictionarul de date al sistemului.
- ◆ Un exemplu de cerere este urmatorul:

```
SQL> select tablespace_name,  
      extent_management, allocation_type  
    from dba_tablespaces;
```

# Exemplu

TABLESPACE_NAME	EXTENT_MAN	ALLOCATIO
SYSTEM	LOCAL	SYSTEM
SYSAUX	LOCAL	SYSTEM
UNDOTBS1	LOCAL	SYSTEM
TEMP	LOCAL	UNIFORM
USERS	LOCAL	SYSTEM
EXAMPLE	LOCAL	SYSTEM
BD_DATA	LOCAL	SYSTEM
BD_TEMP	LOCAL	UNIFORM
.....		
TEMPORARY_INDEXES	LOCAL	UNIFORM
TEMPORARY_TABLES	LOCAL	UNIFORM
VERSION_INDEXES	LOCAL	UNIFORM
VERSION_TABLES	LOCAL	UNIFORM
REPOS_TEMP	LOCAL	UNIFORM

23 înregistrări selectate.

# Sintaxa CREATE TABLESPACE

CREATE [TEMPORARY / UNDO] TABLESPACE <tblspc\_name>

DATAFILE / TEMPFILE

'<datafile01\_name and Path where file to create>' SIZE <intM>[, '

'<datafile02\_name and Path where file to create>' SIZE <intM>[, '

'<datafile0N\_name and Path where file to create>' SIZE <intM>[,...]])

BLOCKSIZE <DB\_BLOCK\_SIZE parameter /2k/4k/8k/16k/32k >

AUTOEXTEND { [OFF/ON (NEXT <integer K/M > MAXSIZE<integer K/M >)  
/ UNLIMITED] }

LOGGING/NOLOGGING (implicit: logging)

FORCE LOGGING

ONLINE/OFFLINE (implicit: online)

SEGMENT SPACE MANAGEMENT { AUTO | MANUAL }

FLASHBACK ON | OFF

EXTENT MANAGEMENT { [DICTIONARY] /

[LOCAL (AUTOALLOCATE / UNIFORM <integer K/M >)] }

PERMANENT / TEMPORARY (implicit: permanent)

MINIMUM EXTENT integer

**<Clauza DEFAULT STORAGE>**

NOCACHE;

# Clauza DEFAULT STORAGE

```
DEFAULT STORAGE { [INITIAL <integer K/M >]
                  [NEXT <integer K/M >]
                  [PCTINCREASE <integer K/M >]
                  [MINEXTENTS <integer>]
                  [MAXEXTENTS <integer> / UNLIMITED]
                  [FREELISTS <integer>]
                  [FREELIST GROUPS <integer>]
                  [OPTIMAL <integer>/NULL]
                  [BUFFER_POOL < DEFAULT/KEEP/RECYCLE >] }
```

(pentru crearea unui tablespace vezi si:  
<http://www.orafaq.com/wiki/Tablespace>)

# CREATE TABLESPACE - cont

- ◆ EXTENT MANAGEMENT {DICTIONARY | LOCAL {AUTOALLOCATE | UNIFORM [SIZE int K | M]} }

Aceasta optiune arata daca acel tablespace va fi de tip DMT sau LMT

In cazul LMT se poate specifica suplimentar:

- ◆ optiunea AUTOALLOCATE (cand se interogheaza vederile din dictionarul de date se afiseaza in acest caz 'SYSTEM'):
  - ◆ tablespace-ul va contine extensii de dimensiuni diferite, gestiunea fiind facuta automat de catre sistem.

# CREATE TABLESPACE - cont

- ◆ optiunea AUTOALLOCATE – cont.
  - ◆ Aceasta optiune este buna atunci cand in acel tablespace vor fi stocate obiecte (segmente) de dimensiuni variabile, fiecare putand avea mai multe extensii.
  - ◆ Este un mod simplificat de gestiune (pt. ca e facuta de sistem) dar poate duce uneori la imobilizarea unor spatii pe disc.
  - ◆ Dimensiunea minima a unei extensii este de 64K. Daca blocul de date al BD este 16K sau mai mare atunci dimensiunea minima a unei extensii este de 1M.

# CREATE TABLESPACE - cont

- ◆ optiunea UNIFORM
  - ◆ Specifica faptul ca acel tablespace este gestionat folosindu-se extensii de dimensiune fixa.
  - ◆ Valoarea implicita a dimensiunii este 1M
  - ◆ Fiecare extensie trebuie sa aiba minim 5 blocuri (blocuri BD!). Deci:
    - Daca blocul este de 8192 octeti (8K) atunci dimensiunea minima pentru UNIFORM este de 40K.
    - Pentru 16384 octeti (16K) minimul pentru UNIFORM este 80K.

# CREATE TABLESPACE - cont

- ◆ optiunea UNIFORM - cont
  - ◆ UNIFORM nu este o optiune valida pentru tablespace-ul SYSTEM
  - ◆ Aceasta optiune permite o alocare mai precisa a spatiului astfel incat sa se minimizeze pierderile de spatiu pe disc.
  - ◆ Se foloseste atunci cand avem o estimare asupra spatiului ocupat de fiecare obiect din acel tablespace.

# Exemple

Exemple:

◆ Cazul AUTOALLOCATE:

```
CREATE TABLESPACE user
DATAFILE '/u02/oracle/data/user01.dbf'
      SIZE 50M
EXTENT MANAGEMENT LOCAL AUTOALLOCATE;
```

◆ Cazul UNIFORM:

```
CREATE TABLESPACE user
DATAFILE '/u02/oracle/data/user01.dbf'
      SIZE 50M
EXTENT MANAGEMENT LOCAL UNIFORM SIZE
      128K;
```

# CREATE TABLESPACE - cont

- ◆ MINIMUM EXTENT *int {K|M}* - arata dimensiunea minima a unei extensii (in KB sau MB dupa cum dupa numar urmeaza K sau M). O extensie este de acea dimensiune sau **multiplu** de acea dimensiune. Se foloseste pentru a impiedica o prea mare fragmentare a spatiului.
- ◆ BLOCKSIZE *int K* – se poate specifica o dimensiune nonstandard a blocului pentru acel tablespace. E legat si de alti parametri care trebuie setati.
- ◆ LOGGING | NOLOGGING – anumite operatii (cum ar fi crearea unui index sau incarcarea de date cu loaderul) nu sunt inregistrate in fisierele Redo Log in caz de NOLOGGING. Se aplica obiectelor din acel tablespace. Nu este recomandat!

# CREATE TABLESPACE - cont

- ◆ **FORCE LOGGING** – Se forteaza inregistrarea in Redo Log a modificarilor pe obiectele din acel tablespace chiar daca ele au fost create cu NOLOGGING.
- ◆ **ONLINE | OFFLINE** – in cazul OFFLINE acel tablespace nu este disponibil imediat dupa creare (trebuie ca ulterior sa fie adus in starea online)
- ◆ **PERMANENT | TEMPORARY** – tablespace permanent sau temporar.

# CREATE TABLESPACE - cont

- ◆ FLASHBACK ON | OFF

Se foloseste in conjunctie cu operatii de tip:

- ◆ ALTER DATABASE FLASHBACK ON si
- ◆ FLASHBACK DATABASE TO

pentru a readuce baza de date la o stare anterioara.

Un exemplu ilustrativ se gaseste la adresa:

<http://www.orafaq.com/node/1847>

# CREATE TABLESPACE - cont

Claузă Storage arată cum va stoca Oracle fiecare obiect în acel tablespace.

Optiunile sale sunt:

- ◆ INITIAL *int* K | M
- ◆ NEXT *int* K | M
- ◆ MINEXTENTS *int*
- ◆ MAXEXTENTS *int* | UNLIMITED
- ◆ PCTINCREASE *int*
- ◆ FREELISTS *int*
- ◆ FREELIST GROUPS *int*
- ◆ OPTIMAL *int* | NULL |
- ◆ BUFFER\_POOL {KEEP | RECYCLE | DEFAULT}

# CREATE TABLESPACE - cont

Detaliere:

- ◆ **INITIAL *intK | M*** – defineste dimensiunea primei extensii (minim 2 blocuri). Valoarea implicita este 5 blocuri ale BD.
- ◆ **NEXT *intK | 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.

# CREATE TABLESPACE - cont

## Clauza Storage - cont

- ◆ **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.

# CREATE TABLESPACE - cont

- ◆ Exista o formula care ne da dimensiunea extensiei cu numarul n:

$$\text{Size}(n) = \text{NEXT} * (1 + \text{PCTINCREASE}/100)^{(n-2)}$$

Deci daca NEXT = 200K iar PCTINCREASE este 50 atunci

- ◆ Size(2) = 200K,
- ◆ Size(3) = 300K,
- ◆ Size(4) = 450K, etc

# CREATE TABLESPACE - cont

- ◆ Optiunile FREELISTS int si FREELIST GROUPS int sunt legate de clauza:  
SEGMENT SPACE MANAGEMENT {MANUAL | AUTO}
- ◆ Aceasta clauza spune cum este gestionat spatiul liber dintr-un segment:
  - ◆ MANUAL
  - ◆ AUTO

# CREATE TABLESPACE - cont

- **MANUAL:** Sunt utilizate liste ale spatiului liber pentru gestiunea acestuia. Acestea sunt liste de blocuri care contin spatiu disponibil pentru noi operatii de INSERT.  
MANUAL este valoarea implicita pentru aceasta clauza. In acest caz FREELISTS este un parametru care specifica numarul de liste de blocuri care pot primi inregistrari. In aplicatii de tip paralel sau distribuit se folosesc grupuri de liste (cate unul pentru fiecare nod).
- **AUTO:** In acest caz sunt utilizate bitmapuri pentru spatiul liber din segmente. Acestea permit o gestiune automata a spatiului disponibil.  
Optiunea AUTO poate fi lenta insa in cazul in care se fac multe actualizari.

# Creare TABLESPACE - cont

- ◆ In cazul LMT nu este nevoie de a specifica in CREATE sau ALTER optiuni de stocare (gestiunea facandu-se automat).
- ◆ Deci nu vor aparea clauzele:
  - ◆ next
  - ◆ pctincrease
  - ◆ minextents
  - ◆ maxextents
  - ◆ default storage
- ◆ In cazul DMT insa aceste clauze pot sa apară atât la crearea unui tablespace cât și în comanda de modificare ALTER TABLESPACE

# CREATE TABLESPACE - cont

Daca nu exista clauza EXTENT MANAGEMENT atunci pentru determinarea tipului (DMT sau LMT) se folosesc informatiile de compatibilitate (parametru in fisierul init.ora) precum si clauzele MINIMUM EXTENT si DEFAULT clauza\_storage astfel:

1. If compatibil < 9.0.0 se creeaza un tablespace DMT
2. If compatibil >= 9.0.0 si **DEFAULT clauza\_storage** NU a fost specificata se creeaza un LMT cu AUTOALLOCATE.

# CREATE TABLESPACE - cont

3. If compatibil >= 9.0.0 si clauza ***DEFAULT clauza\_storage*** a fost specificata si
  - ◆ MINIMUM EXTENT a fost specificata atunci:
    - a. Daca MINIMUM EXTENT, INITIAL, si NEXT sunt egale intre ele iar PCTINCREASE = 0 atunci se creeaza un LMT cu UNIFORM avand dimensiune extensie = INITIAL
    - b. MINIMUM EXTENT, INITIAL si NEXT nu sunt egale SAU PCTINCREASE nu este 0 atunci se creeaza un LMT cu AUTOALLOCATE.
  - ◆ MINIMUM EXTENT nu a fost specificata atunci:
    - Daca INITIAL si NEXT sunt egale iar PCTINCREASE = 0 atunci LMT cu UNIFORM
    - Altfel LMT cu AUTOALLOCATE.

# Exemple

## ◆ Tablespace permanent:

```
create tablespace TS1
logging
datafile '/home/oracle/data/ts1.dbf'
size 16m
autoextend on next 32m maxsize 2048m
extent management local;
```

**Sau:**

```
create tablespace TSDATE
datafile '/home/oracle/data/date.dbf'
size 10M
autoextend on maxsize 200M
extent management local uniform size 64K;
```

# Exemplu

- ◆ Tablespace permanent cu mai multe fisiere de date:

```
create tablespace TS1
datafile '/home/oracle/data/ts1.dbf' size 16m autoextend on next 16m
        maxsize 2048m,
'/home/oracle/data/ts2.dbf' size 32m autoextend on next 32m maxsize
        2048m,
'/home/oracle/data/ts3.dbf' size 64m autoextend on next 64m maxsize
        2048m,
extent management local;
```

# Exemplu

- ◆ Tablespace temporar:

```
create temporary tablespace ttemp
tempfile '/home/oracle/temp/temp01.dbf'
size 16m
autoextend on next 16m maxsize 2048m
extent management local;
```

- ◆ Tablespace de tip UNDO:

```
create undo tablespace tsundo
datafile '/home/oracle/data/undo.dbf'
size 100M;
```

# ALTER TABLESPACE

- ◆ Sintaxa (partiala) este:

```
ALTER TABLESPACE tablespace
{ ADD DATAFILE { filespec
    [AUTOEXTEND [ OFF | ON [NEXT integer [K|M] ]
    [MAXSIZE {UNLIMITED | integer[K|M] } [, 
    filespec ...] }
| RENAME DATAFILE 'filename' [, 'filename'] ...
    TO 'filename' [, 'filename'] ...
| DEFAULT STORAGE storage_clause
| ONLINE
| OFFLINE [ {NORMAL | TEMPORARY | IMMEDIATE} ]
| READ ONLY
| READ WRITE
| {BEGIN | END} BACKUP
```

# Adaugare fisier

- ◆ Se adauga un nou fisier de date la un tablespace folosind cereri de tip ALTER TABLESPACE:

```
ALTER TABLESPACE user
ADD DATAFILE
  '/u02/oracle/data/user01.dbf' SIZE 50M
```

# Adaugare fisier - cont

- ◆ Se pot adauga mai multe fisiere cu aceeasi comanda:

```
ALTER TABLESPACE user
ADD DATAFILE '/u02/oracle/data/user01.dbf' SIZE
50M,
'/u02/oracle/data/user02.dbf' SIZE 50M,
'/u02/oracle/data/user03.dbf' SIZE 50M,
```

- ◆ Obs: Daca nu se specifica in comanda calea, Oracle creeaza fisierele in directorul default al serverului.

# AUTOEXTEND

- ◆ Clauza AUTOEXTEND permite / inhiba extinderea automata a fisierelor de date:
  - ◆ AUTOEXTEND OFF inhiba cresterea automata a acestora in dimensiune
  - ◆ In cazul AUTOEXTEND ON fisierele de date se extind automat la nevoie

# AUTOEXTEND - cont

- ◆ NEXT specifica dimensiunea minima a incrementului (in Kb sau Mb) in cazul in care sunt necesare noi extensii (extents) si spatiul disponibil din fisier nu este suficient pentru acestea.  
Valoarea implicita pentru NEXT este de 1 bloc al BD
- ◆ MAXSIZE specifica dimensiunea maxima a spatiului care se poate aloca pentru acel fisier de date (pana la ce dimensiune poate creste).
- ◆ UNLIMITED specifica faptul ca nu este setata o dimensiune maxima permisa (se poate extinde oricat, in limita spatiului existent).

# Exemplu

```
ALTER TABLESPACE user
ADD DATAFILE
  '/u02/oracle/data/user01.dbf' SIZE
  200M
AUTOEXTEND ON
NEXT 10M
MAXSIZE 500M
```

Clauza AUTOEXTEND poate fi prezenta in cererile:

- ◆ CREATE DATABASE
- ◆ ALTER DATABASE
- ◆ CREATE TABLESPACE
- ◆ ALTER TABLESPACE

# Specificare AUTOEXTEND pentru fisier existent:

- ◆ Se face folosind ALTER DATABASE:

```
ALTER DATABASE ora
DATAFILE '/u02/oracle/data/user01.dbf'
AUTOEXTEND ON
NEXT 10M
MAXSIZE 500M
```

# RESIZE

- ◆ Pentru schimbarea manuala a dimensiunii unui fisier (marire sau micsorare) se poate folosi ALTER DATABASE. In clauza DATAFILE pot fi prezente mai multe nume de fisiere (sunt toate afectate):

```
ALTER DATABASE ora  
DATAFILE '/u02/oracle/data/user01.dbf'   
RESIZE 500M
```

- ◆ Pentru cazul micsorarii dimensiunii, aceasta se poate face doar cu spatiul liber de la sfarsitul fisierului (daca exista!)

# ONLINE / OFFLINE

- ◆ Sintaxa clauzei:

**ONLINE | OFFLINE [{NORMAL | TEMPORARY | IMMEDIATE}]**

- ◆ Trecerea in modul ONLINE aduce un tablespace care nu era asa in mod online.
- ◆ OFFLINE este optiunea inversa, caz in care se inhiba accesul la acel tablespace si la segmentele care se afla in el.

# ONLINE / OFFLINE - cont

- ◆ Trecerea OFFLINE se poate face in trei feluri:
  1. NORMAL – se executa checkpoint pentru toate fisierile de date din acel tablespace (aceste fisiere trebuie sa fie toate online – si ele pot fi trecute offline!)
  - ◆ In cazul NORMAL, la revenirea online nu este necesar sa se execute operatii de recovery.
  - ◆ NORMAL este valoarea implicita (in caz in care la trecerea OFFLINE nu se specifica nici una din cele trei optiuni).
  - ◆ Daca baza de date este in modul NOARCHIVELOG, NORMAL este optiunea care trebuie aleasa (pentru ca in acest caz nu se poate face recuperarea).

# ONLINE / OFFLINE - cont

2. In cazul TEMPORARY se face checkpoint pentru toate fisierele de date care sunt online dar Oracle nu se asigura ca toate fisierele pot fi scrise.
  - ◆ Orice fisier care e in acel moment offline poate avea nevoie de recovery cand revenim online.
3. IMMEDIATE nu face checkpoint si nici nu verifica daca fisierele sunt disponibile sau nu. La revenirea online este nevoie de recovery.

# Exemple

**ALTER TABLESPACE user ONLINE**

**ALTER TABLESPACE user OFFLINE**

**ALTER TABLESPACE user OFFLINE TEMPORARY**

**ALTER TABLESPACE user OFFLINE IMMEDIATE**

# Read Only – Read Write

- ◆ READ ONLY specifica faptul ca nu sunt permise operatii de scriere in acel tablespace.
- ◆ Inainte de a trece un tablespace in acest mod trebuie sa fie indeplinite urmatoarele:
  - ◆ Acel tablespace trebuie sa fie online.
  - ◆ Nu trebuie sa existe tranzactii active in baza de date respectiva.
  - ◆ Acel tablespace nu trebuie sa contine segmente active de rollback.

# Read Only – Read Write

- ◆ Conditii de trecere R/O – cont:
  - ◆ Acel tablespace nu trebuie sa fie implicat in acel moment intr-o operatie de salvare (online backup).
  - ◆ Parametrul de initializare COMPATIBLE trebuie setat la versiunea 7.1.0 sau la una ulterioara acesteia.
- ◆ READ WRITE specifica faptul ca acel tablespace revine din starea de READ ONLY in starea READ WRITE in care poate fi si scris.
- ◆ In acest caz toate fisierele de date ale acelui tablespace trebuie sa fie online.

# Mutarea fisierelor de date

- ◆ Fisierele de date ale unui tablespace pot fi mutate astfel:
  1. Se trece acel tablespace offline.
  2. Cu comenzi SO copiem fisierele in noua locatie.
  3. Se executa ALTER TABLESPACE RENAME.
  4. Se readuce acel tablespace online.
  5. Se pot apoi sterge vechile fisiere de date cu comenzi SO.

# Mutarea fisierelor de date - cont

- ◆ Iata un exemplu de comanda:

```
ALTER TABLESPACE user
RENAME DATAFILE
  '/u02/oracle/data/user01.dbf' TO
  '/u15/oracle/data/user01.dbf'
```

- ◆ Oracle nu face efectiv vreo redenumire de fisiere ci doar inlocuieste in fisierele de control vechiul nume de fisier cu cel nou.

# Mutarea fisierelor – v2

- ◆ Exista si posibilitatea de a muta fisierele de date cu comanda ALTER DATABASE. Pentru aceasta:
  1. Se opreste baza de date.
  2. Se muta fisierele cu comenzi SO.
  3. Se monteaza baza.
  4. Se executa ALTER DATABASE RENAME FILE.
  5. Se deschide baza.

# Mutarea fisierelor de date - cont

- ◆ Iata un exemplu de comanda:

```
ALTER DATABASE ora
RENAME FILE
  '/u02/oracle/data/user01.dbf' TO
  '/u15/oracle/data/user01.dbf'
```

- ◆ La fel ca inainte, Oracle nu face efectiv vreo redenumire de fisiere ci doar inlocuieste in fisierele de control vechiul nume de fisier cu cel nou.
- ◆ In ambele cazuri se pot redenumi cu o singura comanda mai multe fisiere (RENAME FILE lista-old TO lista-new).

# Stergere TABLESPACE

- ◆ Se face cu DROP TABLESPACE.
- ◆ Sintaxa:

```
DROP TABLESPACE nume
[INCLUDING CONTENTS
 [CASCADE CONSTRAINTS ]]
]
```

# Stergere TABLESPACE

- ◆ INCLUDING CONTENTS specifica faptul ca se sterg inclusiv acele tablespace-uri care contin date (altfel acestea nu pot fi sterse).
- ◆ CASCADE CONSTRAINTS – sterge si constrangerile referentiale aferente obiectelor din acel tablespace.
- ◆ In cazul in care CASCADE CONSTRAINTS este omisa si exista astfel de constrangeri Oracle va returna o eroare si nu va efectua stergerea.

# VEDERI

- ◆ Există mai multe vederi care pot fi interogate pentru a obține informații despre tablespace-uri.
- ◆ Una dintre ele este DBA\_TABLESPACES. Iată un program de vizualizare:

```
set linesize 250
col "%INC" for 9999
col TABLESPACE_NAME for A21
col EXT_MAN for A6
col STATUS for A6
Col IN_EX for 9999
Col NX_EX for 9999
col minext for 9999
col blksz for 99999
select tablespace_name, logging, force_logging FLOG,
block_size blksz, status, contents, extent_management ext_man,
segment_space_management, allocation_type,
initial_extent/1024 in_ex, next_extent/1024
nx_ex, pct_increase "%INC", min_extents minext,
max_extents/1024 max_ext_db, min_extlen
from DBA_TABLESPACES
order by 1;
```

# DBA\_TABLESPACES

```
SQL*Plus 11g
```

```
SQL> SET PAGESIZE 300
SQL> /

```

TABLESPACE_NAME	LOGGING	FLO	BLKSZ	STATUS	CONTENTS	EXT_MA	SEGMENT	ALLOCATIO	IN_EX	NX_EX	%INC	MINEXT	MAX_EXT_DB	MIN_EXTLEN
BD_DATA	LOGGING	NO	8192	ONLINE	PERMANENT	LOCAL	AUTO	SYSTEM	64		1	2097152	65536	
BD_TEMP	NOLOGGING	NO	8192	ONLINE	TEMPORARY	LOCAL	MANUAL	UNIFORM	1024	1024	0	1	1048576	
CONSTANT_GROW_INDEXES	LOGGING	NO	8192	ONLINE	PERMANENT	LOCAL	AUTO	UNIFORM	104	104	0	1	2097152	
CONSTANT_GROW_TABLES	LOGGING	NO	8192	ONLINE	PERMANENT	LOCAL	AUTO	UNIFORM	104	104	0	1	2097152	
DEPENDENCY_INDEXES	LOGGING	NO	8192	ONLINE	PERMANENT	LOCAL	AUTO	UNIFORM	104	104	0	1	2097152	
DEPENDENCY_TABLES	LOGGING	NO	8192	ONLINE	PERMANENT	LOCAL	AUTO	UNIFORM	104	104	0	1	2097152	
DIAGRAM_INDEXES	LOGGING	NO	8192	ONLINE	PERMANENT	LOCAL	AUTO	UNIFORM	104	104	0	1	2097152	
DIAGRAM_TABLES	LOGGING	NO	8192	ONLINE	PERMANENT	LOCAL	AUTO	UNIFORM	104	104	0	1	2097152	
EXAMPLE	NOLOGGING	NO	8192	ONLINE	PERMANENT	LOCAL	AUTO	SYSTEM	64		1	2097152	65536	
LOB_DATA	LOGGING	NO	8192	ONLINE	PERMANENT	LOCAL	AUTO	UNIFORM	104	104	0	1	2097152	
RAPID_GROW_INDEXES	LOGGING	NO	8192	ONLINE	PERMANENT	LOCAL	AUTO	UNIFORM	104	104	0	1	2097152	
REPOS_TEMP	NOLOGGING	NO	8192	ONLINE	TEMPORARY	LOCAL	MANUAL	UNIFORM	1024	1024	0	1	1048576	
SYSAUX	LOGGING	NO	8192	ONLINE	PERMANENT	LOCAL	AUTO	SYSTEM	64		1	2097152	65536	
SYSTEM	LOGGING	NO	8192	ONLINE	PERMANENT	LOCAL	MANUAL	SYSTEM	64		1	2097152	65536	
SYSTEM_META_INDEXES	LOGGING	NO	8192	ONLINE	PERMANENT	LOCAL	AUTO	UNIFORM	104	104	0	1	2097152	
SYSTEM_META_TABLES	LOGGING	NO	8192	ONLINE	PERMANENT	LOCAL	AUTO	UNIFORM	104	104	0	1	2097152	
TEMP	NOLOGGING	NO	8192	ONLINE	TEMPORARY	LOCAL	MANUAL	UNIFORM	1024	1024	0	1	1048576	
TEMPORARY_INDEXES	LOGGING	NO	8192	ONLINE	PERMANENT	LOCAL	AUTO	UNIFORM	104	104	0	1	2097152	
TEMPORARY_TABLES	LOGGING	NO	8192	ONLINE	PERMANENT	LOCAL	AUTO	UNIFORM	104	104	0	1	2097152	
UNDOTBS1	LOGGING	NO	8192	ONLINE	UNDO	LOCAL	MANUAL	SYSTEM	64		1	2097152	65536	
USERS	LOGGING	NO	8192	ONLINE	PERMANENT	LOCAL	AUTO	SYSTEM	64		1	2097152	65536	
VERSION_INDEXES	LOGGING	NO	8192	ONLINE	PERMANENT	LOCAL	AUTO	UNIFORM	104	104	0	1	2097152	
VERSION_TABLES	LOGGING	NO	8192	ONLINE	PERMANENT	LOCAL	AUTO	UNIFORM	104	104	0	1	2097152	

```
23 înregistrări selectate.
```

```
SQL> _
```

# VEDERI - cont

- ◆ Coloane in DBA\_TABLESPACES:
  - ◆ Tablespace\_name – numele acelui tablespace.
  - ◆ Contents – daca el contine date permanente, de undo sau temporare.
  - ◆ Status – Daca este Online, Offline sau Read Only.
  - ◆ De asemenea sunt coloane pentru toti parametrii specificati la creare (pentru a putea vedea valoarea lor): BLOCK\_SIZE , INITIAL\_EXTENT , NEXT\_EXTENT , MIN\_EXENTS , MAX\_EXENTS, PCT\_INCREASE, etc.

# VEDERI - cont

- ◆ Vederea DBA\_DATA\_FILES contine date despre fisierele de date aferente fiecarui tablespace.
- ◆ Se pot folosi si vederile V\$DATAFILE si V\$TABLESPACE (legate prin coloana comuna TS# - id-ul de tablespace) pentru a obtine informatii despre fisierele de date ale fiecarui tablespace.

# Vederi - DBA\_DATA\_FILES

```
SQL> R
 1 SELECT FILE_NAME, TABLESPACE_NAME, STATUS, AUTOEXTENSIBLE,
 2 INCREMENT_BY, ONLINE_STATUS
 3* FROM DBA_DATA_FILES
```

FILE_NAME	TABLESPACE_NAME	STATUS	AUT	INCREMENT_BY	ONLINE_
C:\ORACLE11G_DB\ORADATA\BD\USERS01.DBF	USERS	AVAILA BLE	YES	160	ONLINE
C:\ORACLE11G_DB\ORADATA\BD\UNDOTBS01.DBF	UNDOTBS1	AVAILA BLE	YES	640	ONLINE
C:\ORACLE11G_DB\ORADATA\BD\SYSAUX01.DBF	SYSAUX	AVAILA BLE	YES	1280	ONLINE
C:\ORACLE11G_DB\ORADATA\BD\SYSTEM01.DBF	SYSTEM	AVAILA BLE	YES	1280	SYSTEM
C:\ORACLE11G_DB\ORADATA\BD\EXAMPLE01.DBF	EXAMPLE	AVAILA BLE	YES	80	ONLINE
C:\ORACLE11G_DB\ORADATA\BD\BD_DATA.DBF	BD_DATA	AVAILA BLE	NO	0	ONLINE
C:\ORACLE11G_DB\PRODUCT\11.2.0\DBHOME_1\DATABASE\REPOS_GR_IDX	CONSTANT_GROW_INDEXES	AVAILA BLE	NO	0	ONLINE
C:\ORACLE11G_DB\PRODUCT\11.2.0\DBHOME_1\DATABASE\REPOS_GR_TAB	CONSTANT_GROW_TABLES	AVAILA BLE	NO	0	ONLINE
C:\ORACLE11G_DB\PRODUCT\11.2.0\DBHOME_1\DEPENDENCY_INDEXES	DEPENDENCY_INDEXES	AVAILA BLE	NO	0	ONLINE

# Vederi - V\$DATFILE

# V\$TABLESPACE

The screenshot shows a window titled "SQL Plus\_11g" displaying the results of a SQL query. The query is:

```
SQL> R
1* SELECT * FROM V$TABLESPACE
```

The output is a table with the following columns:

TS#	NAME	INC	BIG	FLA	ENC
0	SYSTEM	YES	NO	YES	
1	SYSAUX	YES	NO	YES	
2	UNDOTBS1	YES	NO	YES	
4	USERS	YES	NO	YES	
3	TEMP	NO	NO	YES	
6	EXAMPLE	YES	NO	YES	
7	BD_DATA	YES	NO	YES	
8	BD_TEMP	NO	NO	YES	
9	CONSTANT_GROW_INDEXES	YES	NO	YES	
10	CONSTANT_GROW_TABLES	YES	NO	YES	
11	DEPENDENCY_INDEXES	YES	NO	YES	
12	DEPENDENCY_TABLES	YES	NO	YES	
13	DIAGRAM_INDEXES	YES	NO	YES	
14	DIAGRAM_TABLES	YES	NO	YES	
15	LOB_DATA	YES	NO	YES	
16	RAPID_GROW_INDEXES	YES	NO	YES	
18	SYSTEM_META_INDEXES	YES	NO	YES	
19	SYSTEM_META_TABLES	YES	NO	YES	
20	TEMPORARY_INDEXES	YES	NO	YES	
21	TEMPORARY_TABLES	YES	NO	YES	
22	VERSION_INDEXES	YES	NO	YES	
23	VERSION_TABLES	YES	NO	YES	
24	REPOS_TEMP	NO	NO	YES	

23 înregistrări selectate.

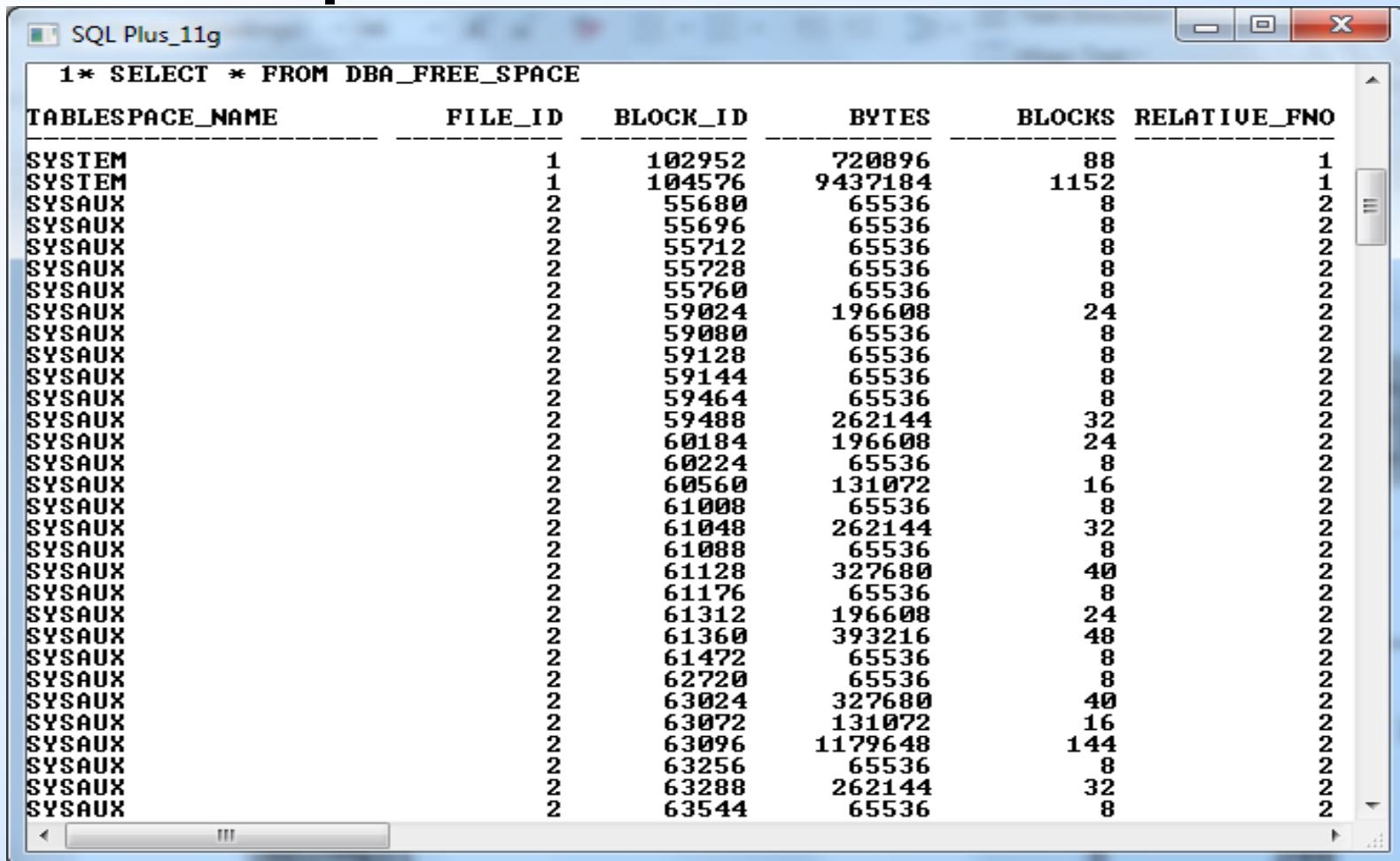
```
SQL>
```

# Alte vederi

Sursa: <http://psoug.org/reference tablespaces.html>

Tablespaces	ts\$ dba_tablespaces user tablespaces
Tablespace Quotas	tsq\$ dba_ts_quotas user ts quotas
Data Files	dba_data_files v_\$backup_datafile v_\$datafile v_\$datafile_copy v_\$datafile_header
Free Space	dba_free_space
Segments	dba_segments v_\$segment_statistics
Extents	dba_extents
Blocks	v_\$database_block_corruption
Groups	dba_tablespace_groups
SYSAUX Tablespace	v_\$sysaux_occupants
Temp Tablespace	dba_temp_files
Undo Tablespace	dba_rollback_segs dba_undo_extents v_\$rollstat v_\$undostat
Transportable Tablespaces	transport_set_violations
Dictionary Management	fet\$ uet\$

# Exemple: DBA\_FREE\_SPACE



The screenshot shows a SQL Plus window titled "SQL Plus\_11g" displaying the results of a query. The query is:

```
1* SELECT * FROM DBA_FREE_SPACE
```

The results are presented in a table with the following columns:

TABLESPACE_NAME	FILE_ID	BLOCK_ID	BYTES	BLOCKS	RELATIVE_FNO
SYSTEM	1	102952	720896	88	1
SYSTEM	1	104576	9437184	1152	1
SYSAUX	2	55680	65536	8	2
SYSAUX	2	55696	65536	8	2
SYSAUX	2	55712	65536	8	2
SYSAUX	2	55728	65536	8	2
SYSAUX	2	55760	65536	8	2
SYSAUX	2	59024	196608	24	2
SYSAUX	2	59080	65536	8	2
SYSAUX	2	59128	65536	8	2
SYSAUX	2	59144	65536	8	2
SYSAUX	2	59464	65536	8	2
SYSAUX	2	59488	262144	32	2
SYSAUX	2	60184	196608	24	2
SYSAUX	2	60224	65536	8	2
SYSAUX	2	60560	131072	16	2
SYSAUX	2	61008	65536	8	2
SYSAUX	2	61048	262144	32	2
SYSAUX	2	61088	65536	8	2
SYSAUX	2	61128	327680	40	2
SYSAUX	2	61176	65536	8	2
SYSAUX	2	61312	196608	24	2
SYSAUX	2	61360	393216	48	2
SYSAUX	2	61472	65536	8	2
SYSAUX	2	62720	65536	8	2
SYSAUX	2	63024	327680	40	2
SYSAUX	2	63072	131072	16	2
SYSAUX	2	63096	1179648	144	2
SYSAUX	2	63256	65536	8	2
SYSAUX	2	63288	262144	32	2
SYSAUX	2	63544	65536	8	2

# Exemple: V\$DATAFILE\_HEADER

V\$DATAFILE_HEADER										
FILE#	STATUS	ERR#	FORMAT REC	FUZ	CREATION_CHANGE#	CREATION_T	TABLESPACE_NAME	TS#	RFILE#	
RESETLOGS_CHANGE#	RESETLOGS_	CHECKPOINT_CHANGE#	CHECKPOINT_CHANGE#	CHECKPOINT_COUNT		BYTES	BLOCKS			
NAME		SPACE_HEADER						LAST DEALLOC CHA		
UNDO_OPT_CURRENT_CHANGE#										
1	ONLINE	10	NO	YES	7	30-03-2010	SYSTEM		0	
947455	23-03-2015	3102137	09-04-2019	637	866123776	942037	105728			
C:\ORACLE11G_DB\ORADATA\BD\SYSTEM01.DBF		4194306								
2	ONLINE	10	NO	YES	2160	30-03-2010	SYSAUX		1	
947455	23-03-2015	3102137	09-04-2019	637	587202560	947192	71680			
C:\ORACLE11G_DB\ORADATA\BD\SYSAUX01.DBF		8388610								
3	ONLINE	10	NO	YES	944668	30-03-2010	UNDOTBS1		2	
947455	23-03-2015	3102137	09-04-2019	225	256901120		31360			
C:\ORACLE11G_DB\ORADATA\BD\UNDOTBS01.DBF		12582914								
4	ONLINE	10	NO	YES	17981	30-03-2010	USERS		4	
947455	23-03-2015	3102137	09-04-2019	634	14417920		1760			
C:\ORACLE11G_DB\ORADATA\BD\USERS01.DBF		16777218								
5	ONLINE	10	NO	YES	976602	23-03-2015	EXAMPLE		6	
947455	23-03-2015	3102137	09-04-2019	221	104857600	967727	12800			
C:\ORACLE11G_DB\ORADATA\BD\EXAMPLE01.DBF		20971522								
6	ONLINE	10	NO	YES	1068220	24-03-2015	BD_DATA		7	
947455	23-03-2015	3102137	09-04-2019	208	52428800		6400			
C:\ORACLE11G_DB\ORADATA\BD\BD_DATA.DBF		25165826								
7	ONLINE	10	NO	YES	1078075	24-03-2015	CONSTANT_GROW_INDEXES		9	
947455	23-03-2015	3102137	09-04-2019	208	2097152		256			
C:\ORACLE11G_DB\PRODUCT\11.2.0\DBHOME_1\		29360130								

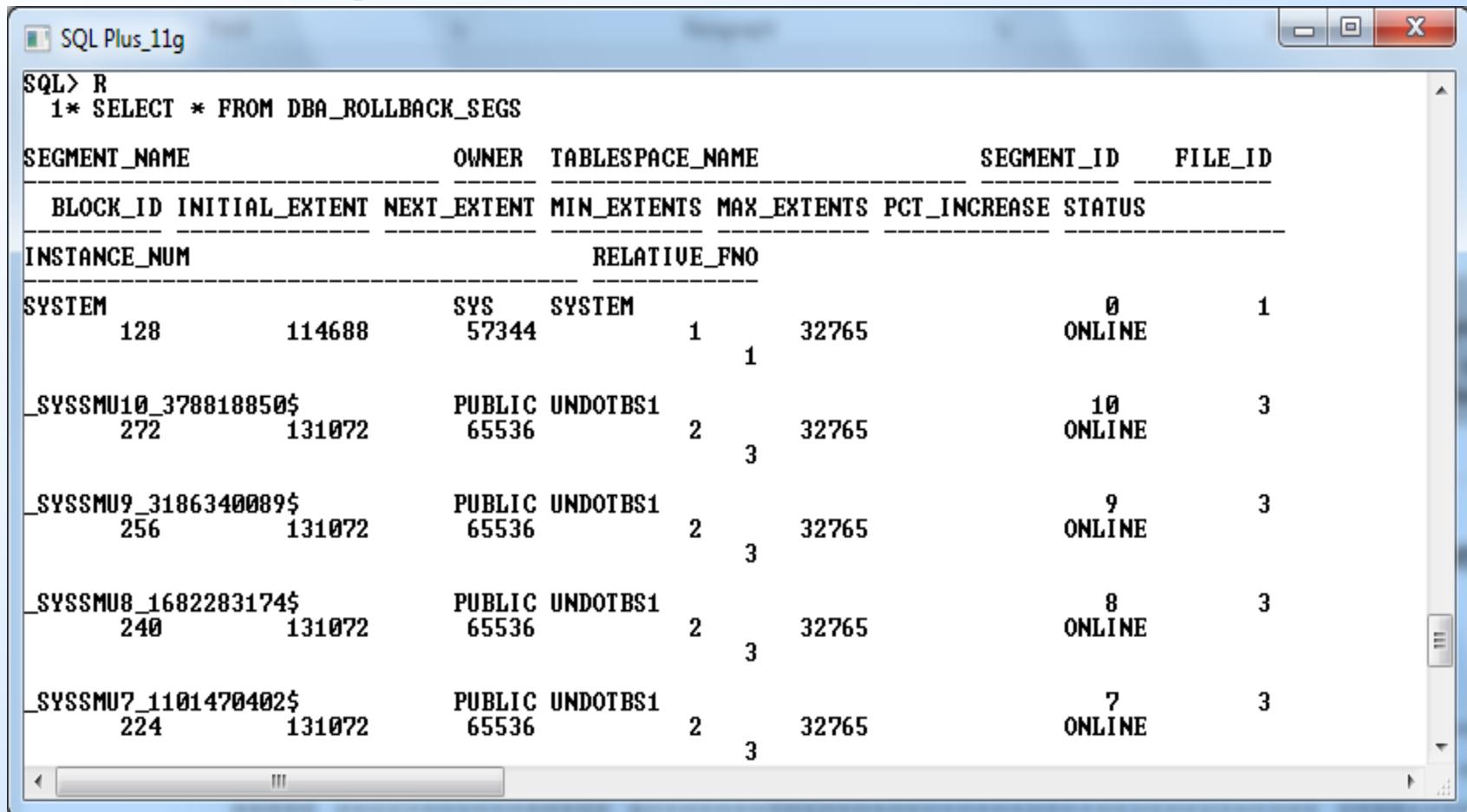
# Exemple: DBA\_TEMP\_FILES

```
SQL>
SQL>
SQL>
SQL> R
1* SELECT * FROM DBA_TEMP_FILES

FILE_NAME
-----
FILE_ID TABLESPACE_NAME          BYTES      BLOCKS STATUS RELATIVE_FNO AUT MAXBYTES
MAXBLOCKS INCREMENT_BY USER_BYTES USER_BLOCKS
C:\ORACLE11G_DB\ORADATA\BD\TEMP01.DBF
      1 TEMP            20971520        2560 ONLINE       1 YES 3,4360E+10
    4194302           80   19922944        2432
C:\ORACLE11G_DB\ORADATA\BD\BD_TEMP.DBF
      2 BD_TEMP         10485760        1280 ONLINE       1 NO   0
        0             0   9437184        1152
C:\ORACLE11G_DB\PRODUCT\11.2.0\DBHOME_1\DATABASE\REPOS_TEMP
      3 REPOS_TEMP      15745024        1922 ONLINE       1 NO   0
        0             0   14680064        1792

SQL> _
```

# Exemple: DBA\_ROLLBACK\_SEGS



SQL> R  
1\* SELECT \* FROM DBA\_ROLLBACK\_SEGS

SEGMENT_NAME	OWNER	TABLESPACE_NAME	SEGMENT_ID	FILE_ID		
BLOCK_ID	INITIAL_EXTENT	NEXT_EXTENT	MIN_EXTENTS	MAX_EXTENTS	PCT_INCREASE	STATUS
SYSTEM	SYS	SYSTEM	1	32765	0	ONLINE
128	114688	57344	1			1
_SYSSMU10_378818850\$	PUBLIC	UNDOTBS1	2	32765	10	ONLINE
272	131072	65536	3			3
_SYSSMU9_3186340089\$	PUBLIC	UNDOTBS1	2	32765	9	ONLINE
256	131072	65536	3			3
_SYSSMU8_1682283174\$	PUBLIC	UNDOTBS1	2	32765	8	ONLINE
240	131072	65536	3			3
_SYSSMU7_1101470402\$	PUBLIC	UNDOTBS1	2	32765	7	ONLINE
224	131072	65536	3			3

# Lecturi obligatorii

## 1. Locally vs. Dictionary Managed Tablespaces

<http://www.orafaq.com/node/3>

## 2. Oracle Database Administrator's Guide – Cap 8: Managing Tablespaces

[http://download.oracle.com/docs/cd/B14117\\_01/server.101/b10739/tspaces.htm](http://download.oracle.com/docs/cd/B14117_01/server.101/b10739/tspaces.htm)

## 3. Oracle Concepts - Tablespaces

<http://www.adp-gmbh.ch/ora/concepts tablespaces.html>

# Sfârșitul capitolului 4

# Capitolul 5

## Gestiunea tabelelor

# Tipuri de organizare

- Există patru tipuri de organizare pentru tabelele unei baze de date:
  1. Tabele uzuale – heap-organized tables – este tipul de bază, ușual. O astfel de tabelă reprezintă o mulțime neorganizată (heap) de linii. Acest tip de tabele este subiectul principal al capitolului de fata.
  2. Tabele partitionate – partitioned tables – în care liniile sunt împărțite în mai multe grupuri, numite partitii, fiecare astfel de partitie (sau subpartitie) putând fi gestionată 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

- În cazul tabelelor partitionate, împărțirea liniilor în partitii se face după valoarea uneia sau mai multor coloane.
- O linie a tabelui poate să aparțină unei singure partitii
- Fiecare partitie are un nume și este stocată într-un segment. Aceste segmente pot fi în tablespace-uri diferite
- Partitionarea se practică în cazul tabelelor de mari dimensiuni care sunt accesate concurențial.
- Se pot partitiona și tabelele de tip 'index-organized' cu condiția ca atributele (coloanele) după care se face partitionarea să 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 acestui parametru este 10 (deci 10% spatiu disponibil pentru update, 90% spatiu disponibil pentru insert).

# Clauze CREATE – PCTFREE(2)

- În cazul în care înregistrările dintr-un bloc cresc în lungime și se depășește spațiul alocat lor (inclusiv cel initial retinut pentru creștere prin PCTFREE) Oracle ia o linie din acel bloc și o mută în alt bloc, lăsând în locul ei doar un pointer.
- Acest proces este numit și ‘migrarea liniilor’
- În acest caz performanțele scad, deoarece pentru citirea acelei linii sunt citite două blocuri

# Clauze CREATE – PCTFREE(3)

- În cazul în care o înregistrare este prea lungă pentru a încapea într-un bloc aceasta înregistrare este spartă în mai multe bucăți care sunt stocate în mai multe blocuri, împreună cu pointerii necesari recuperării întregii linii. Această situație se numește ‘row chaining’.
- În acest caz performanțele scad, deoarece pentru citirea acelei linii sunt citite mai multe blocuri.
- Parametrul PCTFREE poate fi prezent în comenzi create/alter și 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

# Clauze CREATE - PCTUSED

- Acest parametru poate fi prezent si in comenzi de creare pentru alte obiecte (ex. Indeci)
- Suma dintre PCTFREE si PCTUSED trebuie sa fie mai mica sau egala cu 100.
- Cu cat diferență intre 100 și aceasta suma este mai mică, cu atât este mai eficientă folosirea spațiului pe disc, însă pot să scada performanțele.

# Clauze CREATE - INITTRANS

- INITTRANS 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 tableei – deci numarul maxim de ‘tranzaction entries’ care pot fi alocate unui bloc.
- Acestea se aloca dinamic de Oracle dupa depasirea INITTRANS.
- 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 schimbată.
- 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 tablei.
- 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_meia (
  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  /
```

# Rulare

```
florin@rowlf:~ - X ↑
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_meia(
  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>
```

# 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 specifică parcurgerea intregii tabele cu un grad de paralelism egal cu 5. Observam că dacă se definește un alias de tabelă hintul trebuie să folosească acest alias:

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

# PARALLEL - cont

- Cererea urmatoare specifică parcurgerea tabelei fără paralelizare. De asemenea trebuie să se folosească 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 parcursere completa a tableei.
- NOCACHE (valoare implicita) specifica faptul ca blocurile tableei din buffer cache se supun algoritmului LRU standard atunci cand se executa o parcursere completa a tableei (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 tableei 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 tableei 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.
- În general coloanele respective ale tabelei sunt parte a cheii primare (sau întreaga cheie primara).
- Trebuie specificată câte o coloană a tabelei pentru fiecare coloană a clusterului.
- Corespondența este pozitională (nu prin nume)
- Deoarece tabelele de tip cluster folosesc o altă alocare a spațiului NU se pot folosi în paralel clauzele PCTFREE, PCTUSED, INITTRANS, MAXTRANS, TABLESPACE în conjuncție 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 cluster:

```
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;
```

# Rulare

```
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>
```

# Rulare

```
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> 
```

# 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 tablelor este vizibila tuturor sesiunilor active
- Datele din aceste table 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 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 fiecarei tranzactii.

# Rulare

```
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

- Se poate copia o tabela schimbandu-i la momentul copierii anumiti parametrii folosind CREATE TABLE .. AS SELECT
- În acest caz se pot specifica noi nume ale coloanelor, noi parametrii de stocare fizica, etc.
- Atenție: nu sunt copiate în acest caz și constrangerile de integritate (cu excepția coloanelor definite cu NOT NULL, care vor fi la fel și în noua tabelă).

# Comanda ALTER TABLE

```
ALTER TABLE [schema.]table
[optiuni ADD, MODIFY, etc - prezentate anul trecut]
[PCTFREE integer] [PCTUSED integer]
[INITTRANS 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 tablei.
- 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 instancele. 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 tablei 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 tablei.

# ALTER TABLE - TRIGGERS

ENABLE ALL TRIGGERS

DISABLE ALL TRIGGERS

- 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 schimbată cu orice valoare care e mai mică sau egală cu numărul de extensii pe care le are tabela la acel moment
- MAXEXTENTS – poate fi schimbată cu orice valoare mai mare sau egală cu numărul de extensii ale tabelei la acel moment

# Parametri utilizare bloc

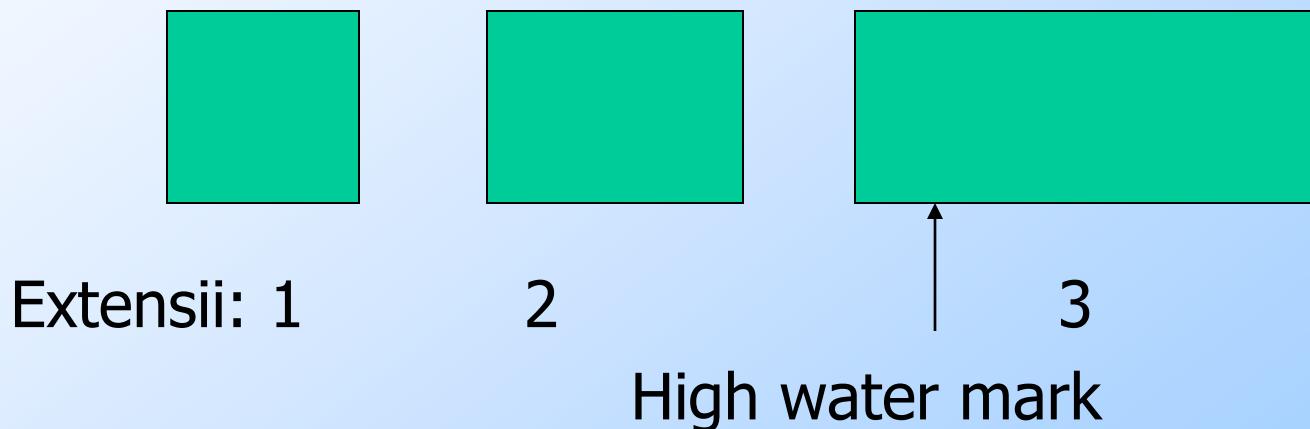
- PCTFREE – schimbarea sa afecteaza urmatoarele operatii de inserare. Blocurile ‘umplute’ dupa vechea valoare nu sunt afectate decat in momentul in care ele ajung in lista de blocuri cu spatiu liber – FREELIST, deci de blocuri in care se pot face operatii de inserare.
- Un bloc ajunge in lista de blocuri 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.
- INITTRANS – schimbarea acestui parametru va afecta doar blocurile noi
- MAXTRANS – schimbarea acestui parametru va afecta toate blocurile tablei (pentru ca diferența de la INITTRANS pana la MAXTRANS sunt ‘transaction entries’ care se aloca dinamic.

# High water mark

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



# De ce se numeste asa?



High water mark

Surse: [http://www.tripper.ro/view.php?ce=gal&gal\\_id=119&id=&limba\\_id=1](http://www.tripper.ro/view.php?ce=gal&gal_id=119&id=&limba_id=1) si <http://www.plaiurimioritice.ro/tag/lac-de-acumulare/>

# High water mark - cont

- Pe masura ce datele sunt inserate in tabela, marcajul 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 parcursere 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 tb1space2
```

- In acest caz se pastreaza definitiile tuturor constrangerilor de integritate si al indecsilor, dar acestia din urma trebuie refacuti (indecsi 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:
  - Permite 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 tablei 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 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 goneste 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)

# DROP STORAGE

- In cazul DROP STORAGE:
  - Sunt dealocate toate extensiile superioare lui MINEXTENTS
  - HWM e resetat
  - Valoarea lui NEXT\_EXTENT (dimensiunea urmatoarei extensii care va fi alocata la nevoie) este resetata la valoarea extensiei cu numarul cel mai mic care a fost dealocata
- In ambele cazuri (REUSE sau DROP), trunchierea afecteaza toti indecsii tablei 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 constraingerilor de integritate aferente (inclusiv cele referentiale)
- Daca nu se specifica CASCADE tabela nu se poate sterge daca exista constraingeri 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 tableei 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 tablei si se face si o verificare incrusisata intre continutul de date al tablei 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 tablei 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 parcursere completa a tablei
- 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';
```

The screenshot shows a terminal window titled "florin@rowlf:~/www". It contains the following SQL\*Plus session:

```
SQL>  
SQL>  
SQL>  
SQL>  
SQL>  
SQL> analyze table scott.emp compute statistics;  
Table analyzed.  
SQL> select blocks, empty_blocks, chain_cnt  
  2  from dba_tables  
  3  where owner='SCOTT'  
  4  and table_name='EMP';  
      BLOCKS   EMPTY_BLOCKS   CHAIN_CNT  
-----  -----  -----  
        5          3          0  
SQL>
```

- 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/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/b28310/tables.htm)  
[http://download.oracle.com/docs/cd/B28359\\_01/server.111/b28286/statements\\_7002.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>

# Sfârșitul capitolului 5

# Capitolul 6

## Gestiune utilizatori, profiluri, privilegii, roluri

# Setari pentru useri

- ❑ Mecanismul de autentificare
- ❑ Cota pe diverse tablespace-uri
- ❑ Tablespace implicit (default)
- ❑ Tablespace temporar
- ❑ Blocare cont
- ❑ Limitari de resurse (profiluri)
- ❑ Privilegii user
- ❑ Roluri

# Mecanismul de autentificare

- Autentificarea userului se poate face in mai multe feluri:
  1. De catre serverul de BD (**database authentication**) – pe baza unui username si a unei parole (cum lucrati de obicei la laborator).
  2. Prin sistemul de operare (**operating system authentication**) – Oracle foloseste informatiile despre user aflate in sistemul de operare si il autentifica, nemaifiind necesara introducerea unui username si a unei parole.
  3. Prin retea (**network authentication**) – folosind servicii de autentificare third-party, ca de exemplu: Distributed Computing Environment (DCE), Kerberos, public key infrastructure, the Remote Authentication Dial-In User Service (RADIUS), SSL, etc

# Cota pe diverse tablespace-uri

- ❑ La crearea unui nou user se poate specifica spatiul pe care acel user il poate ‘consuma’ din diversele tablespace-uri care există la acel moment în sistem.
- ❑ Nu se pot asocia cote pe tablespace-urile temporare
- ❑ Implicit userii nu au cote asociate cu nici un tablespace

# DBA\_TS\_QUOTAS

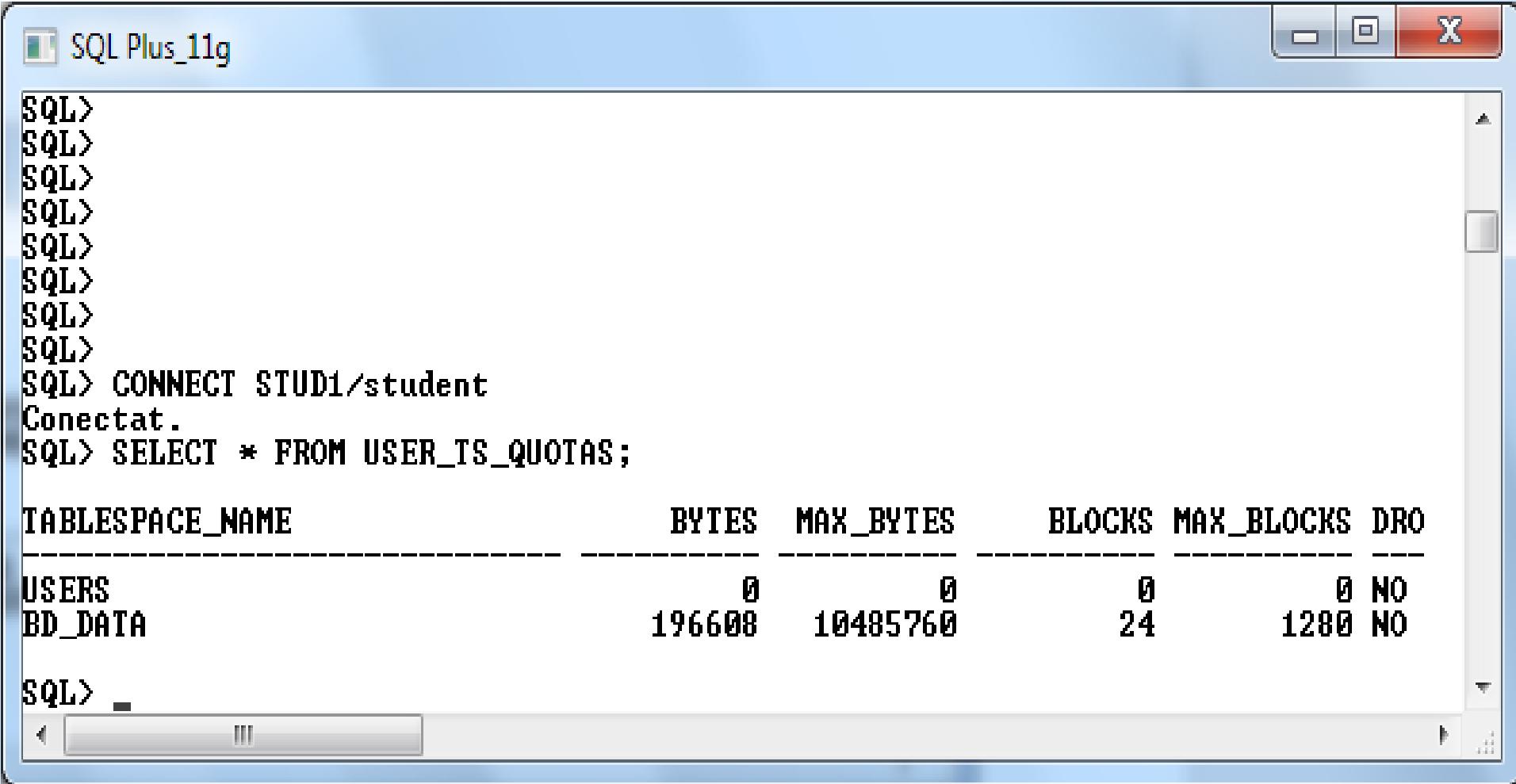
```
SQL>
SQL>
SQL>
SQL> select * from DBA_TS_QUOTAS;

TABLESPACE_NAME USERNAME    BYTES  MAX_BYTES   BLOCKS MAX_BLOCKS DRO
-----  -----  -----  -----  -----  -----  -----
BD_DATA      STUD1     196608  10485760      24    1280 NO
BD_DATA      UBD1     196608  10485760      24    1280 NO
BD_DATA      UBD2     196608  10485760      24    1280 NO
SYSAUX       APPQOSSYS    0      -1           0      -1 NO
SYSAUX       FLOWS_FILES    0      -1           0      -1 NO
SYSAUX       SYSMAN    95092736    -1          11608    -1 NO
SYSAUX       OLAPSYS   7667712     -1          936    -1 NO
BD_DATA      DMUSER     0      10485760      0    1280 NO
BD_DATA      STUD2     196608  10485760      24    1280 NO

9 înregistrări selectate.

SQL>
```

# USER\_TS\_QUOTAS



The screenshot shows a Windows-style window titled "SQL Plus\_11g". Inside, the SQL command-line interface is visible, displaying the following session:

```
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL> CONNECT STUD1/student
Conectat.
SQL> SELECT * FROM USER_TS_QUOTAS;
```

A table is displayed with the following data:

TABLESPACE_NAME	BYTES	MAX_BYTES	BLOCKS	MAX_BLOCKS	DRO
USERS	0	0	0	0	NO
BD_DATA	196608	10485760	24	1280	NO

At the bottom, there is a status bar with navigation icons.

# Cota - cont

- ❑ Asignarea unei cote pentru un user intr-un tablespace are urmatoarele efecte:
  - ❑ Userii care au privilegiul de a crea obiecte pot crea acele obiecte in tablespace-ul respectiv.
  - ❑ Oracle limiteaza spatiul pe care acele obiecte il pot ocupa in tablespace-ul specificat la cat spune cota alocata.
- ❑ Se poate inhiba pentru un user posibilitatea de creare de noi obiecte intr-un anumit tablespace prin setarea unei cote egale cu 0

# Cota - cont

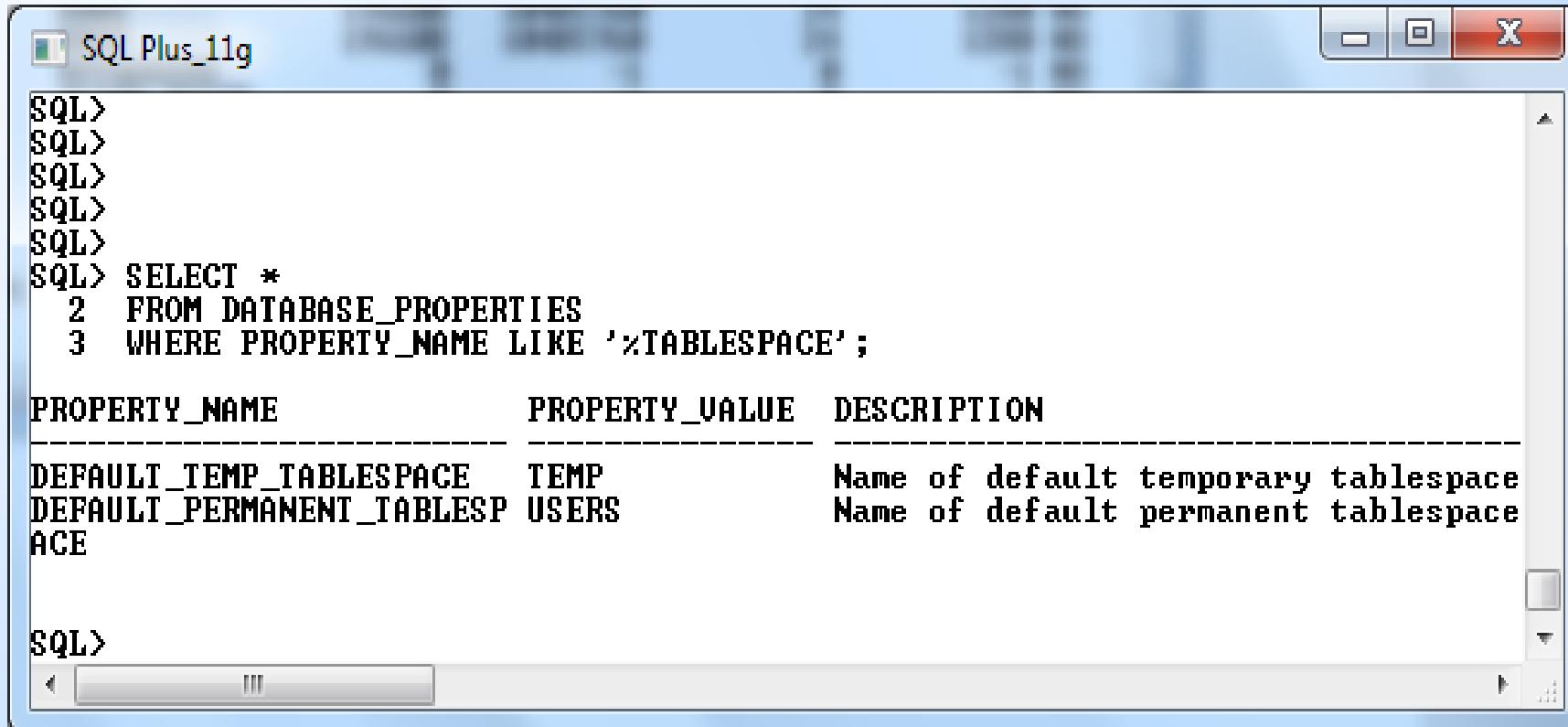
- ❑ Cand cota unui user este modificata la o valoare mai mica decat spatiul ocupat la acel moment de acel user in acel tablespace (inclusiv la setarea unei cote egala cu 0) obiectele existente nu se sterg dar:
  - ❑ Nu se mai pot crea noi obiecte
  - ❑ Obiectele existente nu mai pot creste in dimensiune (dar pot scadea)

# Tablespace implicit (default)

- ❑ Orice user are un tablespace implicit (default).
- ❑ Acest tablespace defineste locatia unde sunt create obiectele (segmentele) userului in absenta specificarii unui tablespace in momentul crearii acelui obiect.
- ❑ La crearea unui nou user se poate optional specifica tablespace-ul implicit al userului (cel permanent si cel temporar).
- ❑ Daca nu se specifica aceste informatii userul va mosteni valorile implicite ale bazei de date.

# Tablespace implicit (default)

- ❑ Aflarea tablespace-urilor implicite se poate face din DATABASE\_PROPERTIES:



```
SQL>
SQL>
SQL>
SQL>
SQL>
SQL> SELECT *
  2  FROM DATABASE_PROPERTIES
  3  WHERE PROPERTY_NAME LIKE '%TABLESPACE';

PROPERTY_NAME          PROPERTY_VALUE   DESCRIPTION
-----  -----
DEFAULT_TEMP_TABLESPACE    TEMP           Name of default temporary tablespace
DEFAULT_PERMANENT_TABLESPACE  USERS         Name of default permanent tablespace

SQL>
```

# Tablespace implicit (default)

- ❑ In Oracle valoarea de default este tablespace-ul SYSTEM, ceea ce nu este foarte bine in cazul in care userul creaza noi obiecte.
- ❑ Este bine sa se creeze un tablespace permanent si unul temporar iar userii uzuali sa le aiba pe acestea ca implicite.
- ❑ Userii de sistem (SYS, SYSTEM) trebuie insa sa ramana cu tablespace implicit SYSTEM.
- ❑ Tablespace-urile default ale unui user se pot schimba si dupa crearea userului, cu ALTER USER.

# Tablespace temporar

- ❑ In cazul in care sunt folosite segmente temporare (de exemplu sunt executate cereri care implica sortari de date voluminoase), acestea sunt stocate:
  - In tablespace-ul implicit (default) daca nu s-a specificat un tablespace temporar la crearea userului
  - In tablespace-ul temporar daca acesta a fost specificat
- ❑ Si acest tablespace se poate specifica si ulterior, prin ALTER USER

# Aflarea valorilor implicite

- Pentru a afla valorile implicite ale unui user existent se poate interoga vederea DBA\_USERS:

```
SQL> select USERNAME, DEFAULT_TABLESPACE,  
      TEMPORARY_TABLESPACE  
    from DBA_USERS  
   where USERNAME='stud1';
```

USERNAME	DEFAULT_TABLESPACE	TEMPORARY_TABLESPACE
stud1	USERS	TEMP

```
SQL>
```

# Blocare cont

- ❑ Un cont poate fi configurat sa se blocheze dupa un anumit numar de incercari de intrare fara succes.
- ❑ Contul se poate debloca dupa un anumit interval de timp, specificat, sau de catre administratorul bazei de date.
- ❑ De asemenea, parola de la creare se poate seta ca expirata, fortand astfel schimbarea parolei (de catre user sau de catre administratorul bazei de date) inainte de a putea intra in sistem.

# Obiectele unui user

- ❑ Ele formează 'schema' aceluia user
- ❑ Pot fi:
  - ❑ Tabele (cu declanșatori și constrângeri asociate)
  - ❑ Indecsi
  - ❑ Vederi
  - ❑ Secvențe
  - ❑ Subprograme stocate
  - ❑ Sinonime
  - ❑ Tipuri definite de user
  - ❑ Legături (database links – prin ele se pot accesa obiecte din alte baze de date)

# Crearea unui nou user

- ❑ La crearea unui nou user se stabilesc mai intai urmatoarele:
  - ❑ Numele, parola si metoda de autentificare pentru acel user
  - ❑ Tablespace-urile care pot fi utilizate de catre acesta
  - ❑ Cota alocata userului pentru fiecare tablespace
  - ❑ Tablespace-ul implicit si cel temporar
- ❑ Se emite comanda CREATE USER care foloseste informatiile de mai sus
- ❑ Se adauga apoi privilegii si roluri pentru user.

# Sintaxa

```
CREATE USER username
IDENTIFIED {BY password
             | EXTERNALLY
             | GLOBALLY AS 'external_name'}
[ DEFAULT TABLESPACE tablespace ]
[ TEMPORARY TABLESPACE tablespace ]
[ QUOTA int {K | M} ON tablespace ]
[ QUOTA UNLIMITED ON tablespace ]
[ PROFILE {profile_name | DEFAULT } ]
[ PASSWORD EXPIRE ]
[ ACCOUNT {LOCK|UNLOCK} ]
```

# Detalii

```
IDENTIFIED { BY password
              | EXTERNALLY
              | GLOBALLY AS 'external_name' }
```

- ❑ Aceasta clauza spune modul de autentificare pentru acest user:
- ❑ BY password arata ca este un user local care trebuie sa specifice username si parola la login,
- ❑ EXTERNALLY indica un user extern, autentificat fie prin sistemul de operare fie prin servicii third party
- ❑ GLOBALY arata ca este un user global, autentificat prin 'directory services'

# Detalii

[ **DEFAULT TABLESPACE** *tablespace* ]

- ❑ Aceasta clauza specifica tablespace-ul default (implicit)

[ **TEMPORARY TABLESPACE** *tablespace* ]

- ❑ Aceasta clauza specifica tablespace-ul pentru segmente temporare

[ **QUOTA** *int {K | M}* **ON** *tablespace* ]

- ❑ Aceasta clauza specifica valoarea cotei pe un anumit tablespace in bytes / KB / MB.

[ **QUOTA UNLIMITED** **ON** *tablespace* ]

- ❑ Aceasta clauza specifica faptul ca nu este fixata o limita superioara pentru cota pe acel tablespace (bineinteles segmentele userului nu pot depasi spatiul existent acolo)

# Detalii

[ PROFILE { *profile\_name* | DEFAULT } ]

- ❑ Specifica profilul asociat cu acel user, acesta aratand limitarile privind resursele pe care le poate consuma userul. Daca nu se specifica, va fi asociat un profil implicit.

[ PASSWORD EXPIRE ]

- ❑ Specifica faptul ca parola este 'pre-expirata', deci DBA sau userul trebuie sa o schimbe inainte de a se putea intra in acel cont

[ ACCOUNT {LOCK|UNLOCK} ]

- ❑ Specifica faptul ca acel cont este blocat (LOCK), deci necesita deblocare inainte de a fi utilizat. Implicit contul este deblocat (UNLOCK) si se poate lucra.

# Exemplu

## ❑ User autentificat prin parola:

```
CREATE USER mihai341C5  
IDENTIFIED BY ec004  
DEFAULT TABLESPACE users  
QUOTA 100M ON test  
QUOTA 500K ON users  
TEMPORARY TABLESPACE temp  
PROFILE clerk;
```

## ❑ Se adauga si niste privilegii:

```
GRANT create session TO mihai341C5;
```

# Restrictii pentru parola

- In cazul in care autentificarea se face prin parola, aceasta trebuie sa verifice restrictiile de nume Oracle:
  1. Maximum 30 de caractere (pana in versiunea 11g este case-insensitive.)
  2. Incepe cu o litera
  3. Contine litere, cifre sau caracterele speciale:

# \_ \$

Motivatia acestor restrictii tine de sintaxa comenzii de creare a unui user cu specificarea parolei sau a modificarii parolei sale - aceasta nu se pune intre apostrofi deci trebuie sa respecte regulile pe care le respecta si numele de obiecte.

Exemplu: CREATE USER U1 IDENTIFIED BY PAROLA\_MEA

# Restrictii pentru parola

- Incepand cu versiunea 11g se pot seta parole 'case sensitive':

Exemplu: creare user test2 cu parola Test2

CONN / AS SYSDBA

```
ALTER SYSTEM SET SEC_CASE_SENSITIVE_LOGON = TRUE;
```

```
CREATE USER TEST2 IDENTIFIED BY Test2;
```

```
GRANT CONNECT TO test2
```

-- autentificare

```
SQL> CONNECT TEST2/Test2
```

Connected.

```
SQL> CONNECT TEST2/test2
```

ERROR:

```
ORA-01017: invalid username/password; logon denied
```

# Restrictii pentru parola

In cazul in care se seteaza SEC\_CASE\_SENSITIVE\_LOGON pe FALSE literele mari vor fi la fel cu cele mari.

Exemplu: pentru userul anterior:

```
CONN / AS SYSDBA
```

```
ALTER SYSTEM SET SEC_CASE_SENSITIVE_LOGON = FALSE;
```

```
SQL> CONN TEST2/Test2
```

```
Connected.
```

```
SQL> CONN TEST2/test2
```

```
Connected.
```

```
SQL>
```

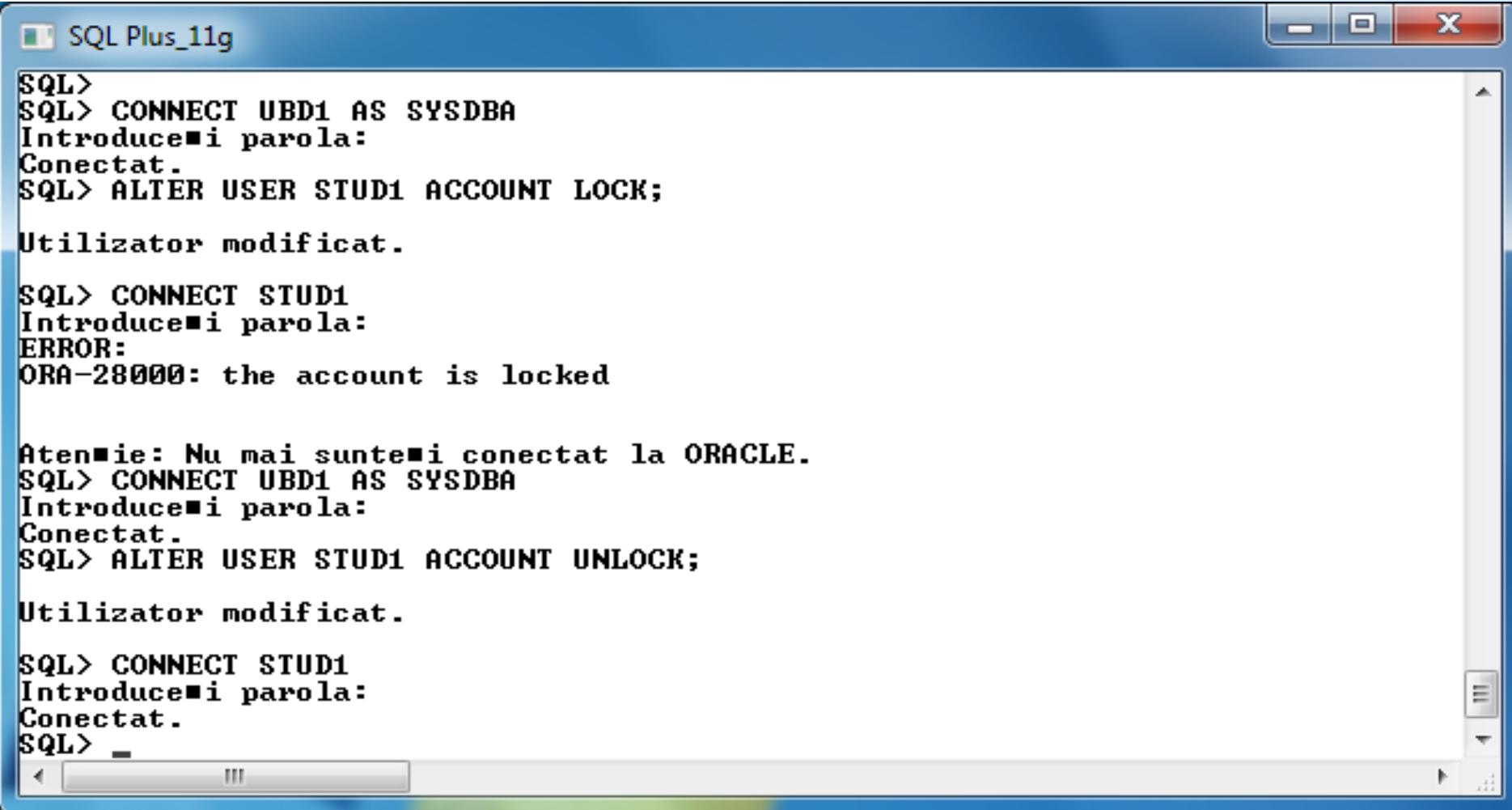
# Modificare date user

- Datele privind autentificarea userului:

```
ALTER USER username
IDENTIFIED {BY password
             | EXTERNALLY
             | GLOBALLY AS 'external_name'}
[ PASSWORD EXPIRE ]
[ ACCOUNT {LOCK|UNLOCK} ]
```

- In momentul blocarii unui cont (LOCK), daca userul e logat la acel moment nu va fi afectat. Modificarile date de comanda de mai sus sunt valabile **incepand cu urmatoarea sesiune** de lucru.

# Modificare date user



The screenshot shows a window titled "SQL Plus\_11g" containing Oracle SQL commands. The session starts with connecting as SYSDBA to UBD1, then locking the STUD1 user account. It then attempts to connect as STUD1, which fails due to the lock. A warning message in Romanian is displayed. Finally, the account is unlocked, and the user connects successfully.

```
SQL>
SQL> CONNECT UBD1 AS SYSDBA
Introducești parola:
Conectat.
SQL> ALTER USER STUD1 ACCOUNT LOCK;

Utilizator modificat.

SQL> CONNECT STUD1
Introducești parola:
ERROR:
ORA-28000: the account is locked

Atenție: Nu mai suntemi conectat la ORACLE.
SQL> CONNECT UBD1 AS SYSDBA
Introducești parola:
Conectat.
SQL> ALTER USER STUD1 ACCOUNT UNLOCK;

Utilizator modificat.

SQL> CONNECT STUD1
Introducești parola:
Conectat.
SQL> _
```

# Modificare date user - cont

- Datele privind tablespace si cote:

```
ALTER USER username
```

```
[ DEFAULT TABLESPACE tablespace ]
```

```
[ TEMPORARY TABLESPACE tablespace ]
```

```
[ QUOTA int {K | M} ON tablespace ]
```

```
[ QUOTA UNLIMITED ON tablespace ]
```

- La trecerea pe 0 a cotei nu se mai pot crea obiecte si cele existente nu mai pot creste. Exemplu:

```
ALTER USER mihai341c5
```

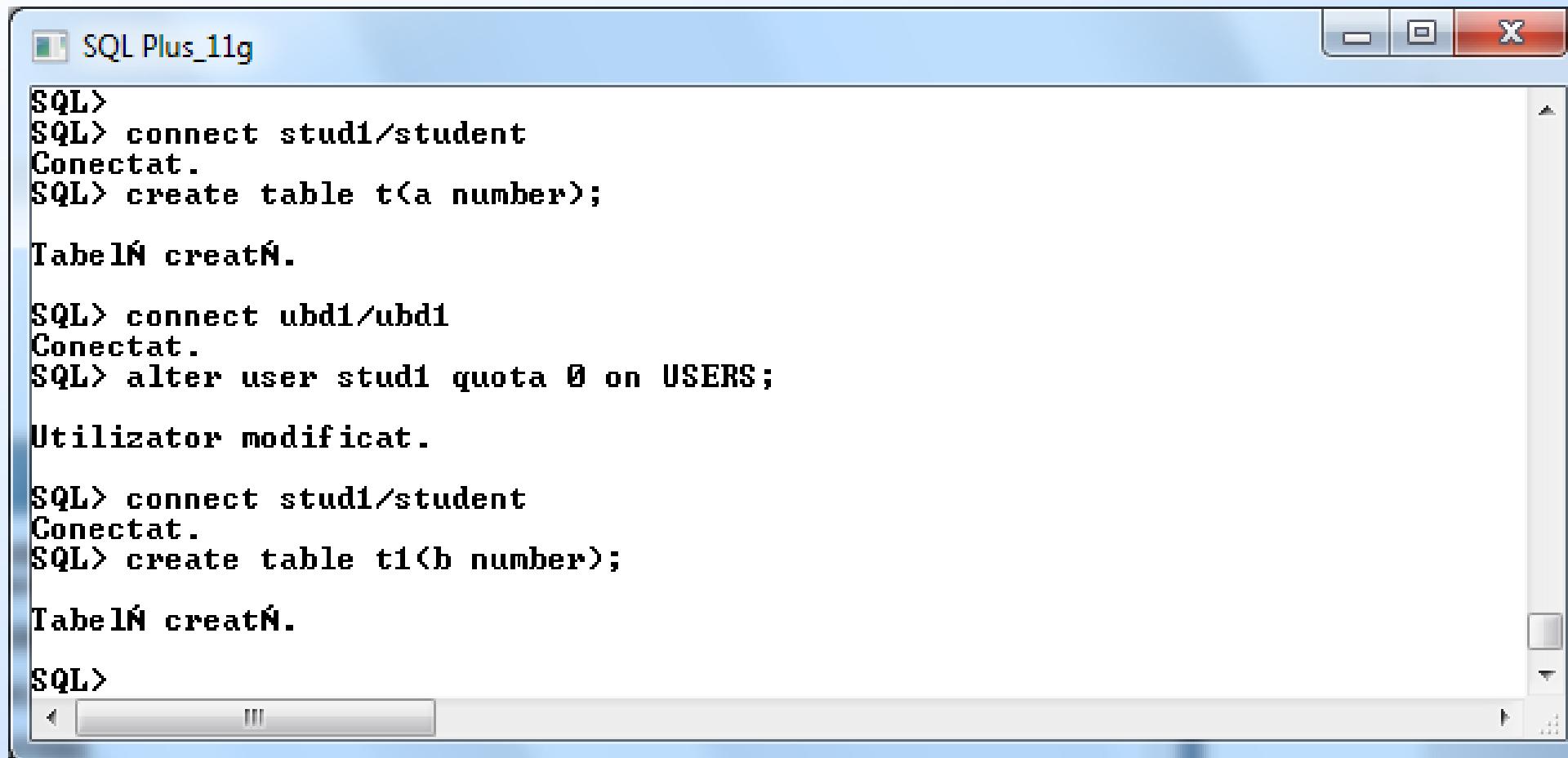
```
QUOTA 0 ON users;
```

# Modificare date user - cont

□ Observatie: trecerea pe 0 a cotei nu are efect daca userul are asignat rolul (colectia de privilegii) **RESOURCE** deoarece aceasta implica o cota nelimitata.

# Modificare date user - cont

- In exemplul de mai jos STUD1 are asignat rolul RESOURCE:



The screenshot shows a Windows application window titled "SQL Plus\_11g". Inside, the SQL command-line interface is used to perform several database operations:

```
SQL>
SQL> connect stud1/student
Conectat.
SQL> create table t(a number);

Tabelul creat.

SQL> connect ubd1/ubd1
Conectat.
SQL> alter user stud1 quota 0 on USERS;

Utilizator modificat.

SQL> connect stud1/student
Conectat.
SQL> create table t1(b number);

Tabelul creat.

SQL>
```

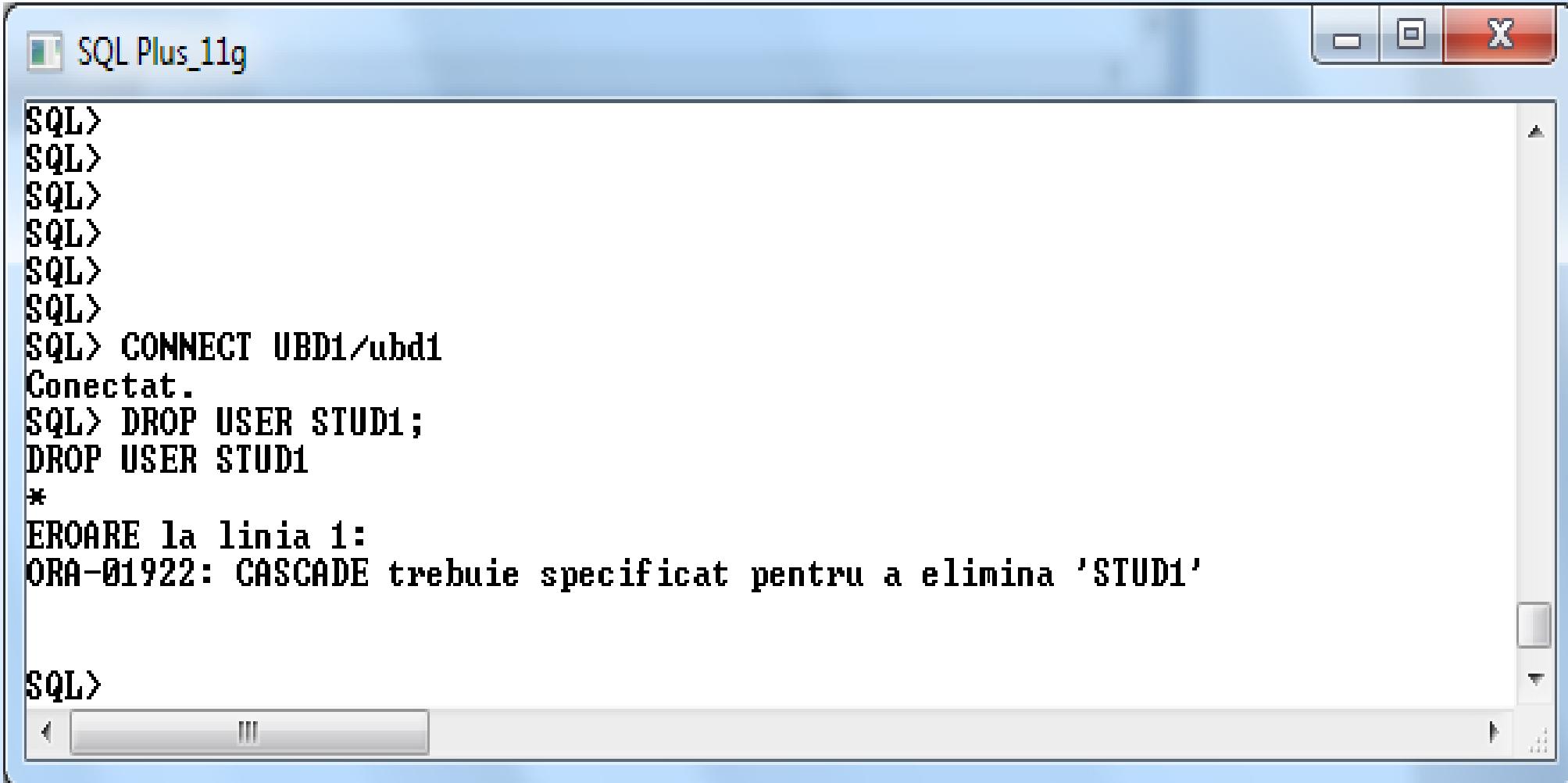
# Stergere user

- ❑ Stergerea unui user se face cu comanda **DROP USER**:

**DROP USER nume [CASCADE]**

- ❑ Optiunea CASCADE sterge intai toate obiectele din schema userului respectiv (altfel se obtine un mesaj de eroare).
- ❑ Fara CASCADE se pot sterge useri care nu detin nici un obiect in schema proprie.

# Stergere user – cont.



The screenshot shows a window titled "SQL Plus\_11g". Inside, the SQL command line shows several attempts to delete a user named "STUD1". The first four lines are blank SQL prompts. The fifth line shows a connection attempt: "SQL> CONNECT UBD1/ubd1". The response is "Conectat.". The sixth line is "SQL> DROP USER STUD1;". The seventh line is "DROP USER STUD1". The eighth line is an asterisk (\*) indicating an error. The ninth line is "EROARE la linia 1:". The tenth line is "ORA-01922: CASCADE trebuie specificat pentru a elimina 'STUD1'" (ORA-01922: CASCADE must be specified to delete 'STUD1'). The bottom line is another blank SQL prompt: "SQL>".

```
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL> CONNECT UBD1/ubd1
Conectat.
SQL> DROP USER STUD1;
DROP USER STUD1
*
EROARE la linia 1:
ORA-01922: CASCADE trebuie specificat pentru a elimina 'STUD1'

SQL>
```

# Vederi care se pot utiliza

View	Description
DBA_USERS	Describes all users of the database.
ALL_USERS	Lists users visible to the current user, but does not describe them.
USER_USERS	Describes only the current user.
DBA_TS_QUOTAS , USER_TS_QUOTAS	Describes tablespace quotas for users.
USER_PASSWORD_LIMITS	Describes the password profile parameters that are assigned to the user (vezi partea despre profiluri din curs).
USER_RESOURCE_LIMITS	Displays the resource limits for the current user (vezi partea despre profiluri din curs).
DBA_PROFILES	Displays all profiles and their limits.
RESOURCE_COST	Lists the cost for each resource.
V\$SESSION	Lists session information for each current session. Includes user name.
V\$SESSTAT	Lists user session statistics.
V\$STATNAME	Displays decoded statistic names for the statistics shown in the V\$SESSTAT view.
PROXY_USERS	Describes users who can assume the identity of other users.

# Exemplu

```
SELECT TABLESPACE_NAME, BLOCKS, MAX_BLOCKS, BYTES,  
      MAX_BYTES  
FROM DBA_TS_QUOTAS  
WHERE USERNAME = 'SCOTT';
```

- Se obtine un rezultat care contine date despre cota userului:

TABLESPACE_NAME	BLOCKS	MAX_BLOCKS	BYTES	MAX_BYTES
-----	-----	-----	-----	-----
DATE	10	-1	20480	-1

- Valoarea -1 reprezinta cota nelimitata. Restul valorilor reprezinta spatiul ocupat la acel moment.

# Alt exemplu

```
SELECT USERNAME, ACCOUNT_STATUS,  
       TEMPORARY_TABLESPACE  
  FROM DBA_USERS
```

- Se obtine o lista cu starea fiecarui cont (si alte date):

USERNAME	ACCOUNT_STATUS	TEMPORARY_TABLESPACE
SYS	OPEN	TEMP
SYSTEM	OPEN	TEMP
DBSNMP	OPEN	TEMP
SCOTT	OPEN	TEMP

# PROFIL

- ❑ Profilurile sunt o modalitate prin care se pot limita resursele care pot fi utilizate de un utilizator.
- ❑ Un profil se creaza cu CREATE PROFILE si se asigneaza userului la creare sau ulterior prin comanda ALTER USER.
- ❑ Exista un profil DEFAULT care se asociaza implicit la userii pentru care la creare nu s-a specificat un profil.

# Resurse ale sistemului

Pentru ca aceste limitari de sistem sa fie active trebuie ca parametrul de initializare RESOURCE\_LIMIT sa fie setat pe True – se poate modifica folosind ALTER SYSTEM

- Numarul maxim de sesiuni concurente pentru user ([SESSIONS\\_PER\\_USER](#))
- Timp CPU per sesiune ([CPU\\_PER\\_SESSION](#)) – masurat in sutimi de secunda.
- Timp CPU per operatie ([CPU\\_PER\\_CALL](#)) – masurat in sutimi de secunda. O operatie este un ciclu parse, execute, fetch.

# Resurse ale sistemului

- Timpul maxim de conectare masurat in minute (CONNECT\_TIME).  
Sesiunile userului sunt inchise de Oracle dupa expirarea acestui timp.
- Timp maxim de asteptare (IDLE\_TIME) – masurat in minute - sesiunile vor fi inchise de Oracle dupa expirarea perioadei specificate daca in sesiunea respectiva nu s-a facut nimic ('idle'). Atentie: cererile a caror executie este lunga nu intra in aceasta categorie!

# Resurse ale sistemului

- Numar maxim de blocuri citite per sesiune. Este vorba aici de numarul de blocuri citite de pe disc **sau** din memorie. Acest parametru este gandit pentru a limita cererile care fac citiri intensive ([LOGICAL\\_READS\\_PER\\_SESSION](#)).
- Numarul maxim de blocuri citite per operatie (call) ([LOGICAL\\_READS\\_PER\\_CALL](#)).
- Dimensiunea maxima de memorie ocupata in shared pool – parte a SGA - de o sesiune de lucru – in bytes ([PRIVATE\\_SGA](#)).

# Resurse legate de parola

- Numarul maxim de incercari eronate de login  
**(FAILED\_LOGIN\_ATTEMPTS)**
- Timpul maxim (in zile) cat parola este valida  
**(PASSWORD\_LIFE\_TIME)**
- Numarul minim de parole diferite utilizate pana cand o parola poate fi reutilizata **(PASSWORD\_REUSE\_MAX)**
- Numarul minim de zile dupa care o parola poate fi utilizata  
**(PASSWORD\_REUSE\_TIME)**

# Resurse legate de parola

Mai exista si:

- PASSWORD\_LOCK\_TIME** : Cate zile se blocheaza contul dupa incercari repetate de login esuate
- PASSWORD\_GRACE\_TIME** : Cate zile sunt disponibile pentru a schimba o parola dupa expirarea acesteia
- PASSWORD\_VERIFY\_FUNCTION**: bloc (program) PL/SQL utilizat pentru verificarea parolei
- SEC\_CASE\_SENSITIVE\_LOGON** : literele mari si cele mici sunt considerate identice sau nu intr-o parola.

# Alte informatii

- ❑ Lista de mai sus nu este exhaustiva. Un tabel cu limitarile care se pot folosi in Oracle 11.1 se gaseste in pagina:

<http://www.psoug.org/reference/profiles.html>

- ❑ Am dat numele parametrilor pentru ca fiecare in parte se poate modifica ulterior prin comenzi ALTER PROFILE.

# Limitari

- ❑ Daca este atinsa o limita la nivel de sesiune atunci:
  - ❑ Fie se afiseaza un mesaj de eroare (de exemplu cand seincearca deschiderea unei noi sesiuni si se depaseste sessions\_per\_user)
  - ❑ Fie Oracle deconecteaza userul (sesiunea), de exemplu cand s-a atins durata ei maxima.

# Limitari

- ❑ Daca este atinsa o limita la nivel de operatie (call) atunci:
  - ❑ Procesarea cererii curente este oprita
  - ❑ Cererea curenta este revocata (rollback)
  - ❑ Efectul cererilor anterioare persista
  - ❑ Userul ramane conectat.

# Creare si utilizare profil

- ❑ In continuare vom discuta despre cum se creaza si se modifica un profil si despre cum este asignat un profil la un user.
- ❑ Sintaxa cererii SQL de creare a unui nou profil este urmatoarea:

# Creare profil

```
CREATE PROFILE profile
LIMIT
[SESSIONS_PER_USER {integer | UNLIMITED | DEFAULT}]
[CPU_PER_SESSION {integer | UNLIMITED | DEFAULT}]
[CPU_PER_CALL {integer | UNLIMITED | DEFAULT}]
[CONNECT_TIME {integer | UNLIMITED | DEFAULT}]
[IDLE_TIME {integer | UNLIMITED | DEFAULT}]
[LOGICAL_READS_PER_SESSION {integer | UNLIMITED | DEFAULT}]
[LOGICAL_READS_PER_CALL {integer | UNLIMITED | DEFAULT}]
[COMPOSITE_LIMIT {integer | UNLIMITED | DEFAULT}]
[PRIVATE_SGA {integer [K|M] | UNLIMITED | DEFAULT}]
```

- Nu sunt listate mai sus toate optiunile.
- Pentru a executa aceasta operatie trebuie privilegiul CREATE PROFILE.

# Creare profil – cont.

- ❑ UNLIMITED: arata faptul ca pentru acel profil resursa respectiva poate fi folosita in cota nelimitata
- ❑ DEFAULT: arata faptul ca resursa respectiva poate fi folosita limitat, valoarea fiind aceeasi cu a profilului DEFAULT
- ❑ COMPOSIT\_LIMIT limiteaza costul total al resurselor pentru o sesiune in unitati de servire. Oracle calculeaza acest cost ca o suma ponderata intre:
  - ❑ CPU\_PER\_SESSION
  - ❑ CONNECT\_TIME
  - ❑ LOGICAL\_READS\_PER\_SESSION
  - ❑ PRIVATE\_SGA

# COMPOSIT\_LIMIT

- ❑ Exemplu de setare a costului resurselor:

```
ALTER RESOURCE COST
```

```
CPU_PER_SESSION 100  
CONNECT_TIME 1  
LOGICAL_READS_PER_SESSION 0  
PRIVATE_SGA 0;
```

- ❑ Cele 2 elemente vor determina costul:

```
COST = (100 * CPU_PER_SESSION) + (1 * CONNECT_TIME)
```

- ❑ La depasirea costului stabilit de COMPOSIT\_LIMIT sesiunea este incheiata de Oracle.
- ❑ Pentru parametri nesetati se iau valorile din DEFAULT.

# Exemplu

```
CREATE PROFILE EXEMPLU LIMIT  
SESSIONS_PER_USER 1  
IDLE_TIME 1  
FAILED_LOGIN_ATTEMPTS 3;
```

❑ Pentru a vizualiza restrictiile aferente profilului creat cu cererea de mai sus se poate executa cererea:

```
SELECT RESOURCE_NAME, LIMIT  
FROM DBA_PROFILES  
WHERE PROFILE = 'EXEMPLU';
```

# Exemplu - cont

□ Rezultatul va contine lista urmatoare:

COMPOSITE_LIMIT	DEFAULT
SESSIONS_PER_USER	1
CPU_PER_SESSION	DEFAULT
CPU_PER_CALL	DEFAULT
LOGICAL_READS_PER_SESSION	DEFAULT
LOGICAL_READS_PER_CALL	DEFAULT
IDLE_TIME	1
CONNECT_TIME	DEFAULT
PRIVATE_SGA	DEFAULT
FAILED_LOGIN_ATTEMPTS	3
PASSWORD_LIFE_TIME	DEFAULT
PASSWORD_REUSE_TIME	DEFAULT
PASSWORD_REUSE_MAX	DEFAULT
PASSWORD_VERIFY_FUNCTION	DEFAULT
PASSWORD_LOCK_TIME	DEFAULT
PASSWORD_GRACE_TIME	DEFAULT

# Asignarea unui profil

Asignarea unui profil la un user se poate face:

- ❑ La crearea userului (CREATE USER) există clauza PROFILE care specifică un profil asociat aceluiași user (în lipsă se ia profilul DEFAULT).
- ❑ Ulterior se poate schimba profilul cu ALTER USER:

**ALTER USER scott**

**PROFILE exemplu;**

# Exemplu de limitare

```
SQL>
SQL> CONNECT UBD1/ubd1
Conectat.
SQL> CREATE PROFILE STUD1 LIMIT FAILED_LOGIN_ATTEMPTS 2;
Profil creat.

SQL> ALTER USER STUD1 PROFILE STUD1;
Utilizator modificat.

SQL> CONNECT STUD1/primaincercare
ERROR:
ORA-01017: invalid username/password; logon denied

Atenție: Nu mai suntemi conectat la ORACLE.
SQL> CONNECT STUD1/adouaincercare
ERROR:
ORA-01017: invalid username/password; logon denied

SQL> CONNECT STUD1/atreiaincercare
ERROR:
ORA-28000: the account is locked

SQL>
```

# Profilul DEFAULT

```
SQL>
SQL> CONNECT UBD1/ubd1
Conectat.
SQL> SELECT * FROM DBA_PROFILES
  2 WHERE PROFILE='DEFAULT';

PROFILE RESOURCE_NAME          RESOURCE LIMIT
-----  -----
DEFAULT COMPOSITE_LIMIT        KERNEL    UNLIMITED
DEFAULT SESSIONS_PER_USER      KERNEL    UNLIMITED
DEFAULT CPU_PER_SESSION        KERNEL    UNLIMITED
DEFAULT CPU_PER_CALL          KERNEL    UNLIMITED
DEFAULT LOGICAL_READS_PER_SESSION KERNEL    UNLIMITED
DEFAULT LOGICAL_READS_PER_CALL  KERNEL    UNLIMITED
DEFAULT IDLE_TIME              KERNEL    UNLIMITED
DEFAULT CONNECT_TIME           KERNEL    UNLIMITED
DEFAULT PRIVATE_SGA             KERNEL    UNLIMITED
DEFAULT FAILED_LOGIN_ATTEMPTS  PASSWORD  10
DEFAULT PASSWORD_LIFE_TIME     PASSWORD  180
DEFAULT PASSWORD_REUSE_TIME    PASSWORD UNLIMITED
DEFAULT PASSWORD_REUSE_MAX     PASSWORD UNLIMITED
DEFAULT PASSWORD_VERIFY_FUNCTION PASSWORD NULL
DEFAULT PASSWORD_LOCK_TIME     PASSWORD 1
DEFAULT PASSWORD_GRACE_TIME    PASSWORD ??

16 înregistrări selectate.

SQL>
```

# Profilul STUD1

```
SQL Plus_11g

Utilizator modificat.

SQL> SELECT * FROM DBA_PROFILES
  2 WHERE PROFILE='STUD1';

PROFILE RESOURCE_NAME          RESOURCE LIMIT
----- -----
STUD1  COMPOSITE_LIMIT        KERNEL   DEFAULT
STUD1  SESSIONS_PER_USER      KERNEL   DEFAULT
STUD1  CPU_PER_SESSION        KERNEL   DEFAULT
STUD1  CPU_PER_CALL          KERNEL   DEFAULT
STUD1  LOGICAL_READS_PER_SESSION KERNEL   DEFAULT
STUD1  LOGICAL_READS_PER_CALL  KERNEL   DEFAULT
STUD1  IDLE_TIME              KERNEL   DEFAULT
STUD1  CONNECT_TIME           KERNEL   DEFAULT
STUD1  PRIVATE_SGA            KERNEL   DEFAULT
STUD1  FAILED_LOGIN_ATTEMPTS  PASSWORD 2
STUD1  PASSWORD_LIFE_TIME     PASSWORD DEFAULT
STUD1  PASSWORD_REUSE_TIME    PASSWORD DEFAULT
STUD1  PASSWORD_REUSE_MAX     PASSWORD DEFAULT
STUD1  PASSWORD_VERIFY_FUNCTION PASSWORD DEFAULT
STUD1  PASSWORD_LOCK_TIME    PASSWORD DEFAULT
STUD1  PASSWORD_GRACE_TIME   PASSWORD DEFAULT

16 înregistrări selectate.

SQL>
```

# **RESOURCE\_LIMIT**

- ❑ Aşa cum s-a specificat, parametrul de initializare RESOURCE\_LIMIT trebuie să fie TRUE
- ❑ Pentru a vedea care este valoarea curentă a acestui parametru se poate folosi în SQL\*Plus comanda SHOW PARAMETER:

```
SQL> SHOW PARAMETER RESOURCE_LIMIT
RESOURCE_LIMIT BOOLEAN FALSE
```

# RESOURCE\_LIMIT - cont

- ❑ Putem schimba valoarea curenta cu ALTER SYSTEM:

**ALTER SYSTEM**

**SET RESOURCE\_LIMIT=TRUE**

- ❑ Efectul acestei comenzi dureaza pana la o noua schimbare a valorii cu ALTER SYSTEM sau pana cand se opreste baza de date (la repornire va citi din nou valoarea din fisierul de parametri)

# Modificare profil

- Modificarea unui profil se poate face cu ALTER PROFILE (similară cu CREATE):

```
ALTER PROFILE profile
LIMIT
[SESSIONS_PER_USER {integer | UNLIMITED | DEFAULT}]
[CPU_PER_SESSION {integer | UNLIMITED | DEFAULT}]
[CPU_PER_CALL {integer | UNLIMITED | DEFAULT}]
[CONNECT_TIME {integer | UNLIMITED | DEFAULT}]
[IDLE_TIME {integer | UNLIMITED | DEFAULT}]
[LOGICAL_READS_PER_SESSION {integer | UNLIMITED | DEFAULT}]
[LOGICAL_READS_PER_CALL {integer | UNLIMITED | DEFAULT}]
[COMPOSITE_LIMIT {integer | UNLIMITED | DEFAULT}]
[PRIVATE_SGA {integer [K|M] | UNLIMITED | DEFAULT}]
```

# Modificare profil - cont

- ❑ Modificarile de profil nu afecteaza sesiunile curente ci doar pe cele deschise dupa modificare
- ❑ Pentru executia comenzii trebuie privilegiul **ALTER PROFILE**.

# Stergerea unui profil

□ Se face cu DROP PROFILE:

DROP PROFILE nume\_profil [CASCADE]

- Profilul DEFAULT nu se poate sterge
- Stergerea unui profil nu afecteaza sesiunile curente
- Optiunea CASCADE revoca acest profil de la userii care il au
- Userii care au profilul sters vor trece automat pe profilul DEFAULT
- Pentru executia operatiei trebuie privilegiul DROP PROFILE

# Vizualizari

- Pentru a vedea informatii despre stare cont, blocare, data expirarii parolei si alte limitari ale acesteia se poate folosi vederea DBA\_USERS sau vederea DBA\_PROFILES:

```
SELECT USERNAME, PASSWORD, ACCOUNT_STATUS, EXPIRY_DATE  
FROM DBA_USERS;
```

- Se va obtine un rezultat despre fiecare user, de tipul:

USERNAME	PASSWORD	ACCOUNT_ST	EXPIRY_DA
-----	-----	-----	-----
SYS	D4C5123456B4DC8	OPEN	19-DEC-20

# Vizualizari - cont

- ❑ Pentru limitari asupra parolei, in cererile asupra lui DBA\_PROFILES se poate folosi in clauza WHERE conditia

WHERE RESOURCE\_TYPE='PASSWORD'

- ❑ Exemplu:

```
SELECT PROFILE, RESOURCE_NAME, LIMIT  
FROM DBA_PROFILES  
WHERE RESOURCE_TYPE='PASSWORD'
```

# PRIVILEGII

- ❑ Privilegiile sunt o alta denumire pentru drepturile de acces pe care userii le au asupra sistemului de gestiune si asupra bazei de date (incluzand obiectele din aceasta).
- ❑ Privilegiile se pot acorda si revoca individual sau sub forma rolurilor.

# PRIVILEGII

❑ Există două tipuri de privilegii:

1. Privilegii sistem: ele permit userilor să execute operații pe baza de date: creare, stergere, modificare pe tabele, vederi, segmente de rollback și proceduri
2. Privilegii obiect: permit userilor să efectueze anumite operații pe un obiect: tabela, secvența, vedere, procedura, funcție sau pachet

# Privilegii sistem

❑ Există un număr mare de privilegii sistem.

([http://www.psoug.org/reference/system\\_privs.html](http://www.psoug.org/reference/system_privs.html))

❑ Privilegiile de sistem pot fi clasificate în:

- Privilegii pentru operații care afectează întreg sistemul, ca de exemplu CREATE SESSION, CREATE TABLESPACE
- Privilegii care afectează obiectele din orice schema, de ex. CREATE ANY TABLE
- Privilegii care afectează doar obiectele din schema proprie, de ex. CREATE TABLE

# Privilegii sistem – cont.

- ❑ Particula ANY arata ca userul are acel privilegiu in orice schema.
- ❑ Pentru a adauga privilegii la un user se foloseste GRANT.
- ❑ Pentru a inlatura privilegii de la un user se foloseste REVOKE.

# Exemplu de privilegii - CREATE

- Create Any Index
- Create Any Indextype
- Create Any Materialized View
- Create Any Measure Folder
- Create Any Operator
- Create Any Outline
- Create Any Procedure
- Create Any Rule
- Create Any Rule Set
- Create Any Sequence
- Create Any SQL Profile
- Create Any Synonym
- Create Any Table
- Create Any Trigger
- Create Any Type
- Create Any View
- Cele mai multe dintre acestea au si varianta fara ANY.

# Observatii

- ❑ Nu exista privilegiul CREATE INDEX. El este inclus in CREATE TABLE
- ❑ Privilegiile CREATE TABLE, CREATE PROCEDURE si CREATE CLUSTER includ si dreptul de a sterge aceste obiecte
- ❑ Privilegiul UNLIMITED TABLESPACE nu poate fi asignat unui rol ci doar userilor particulari.
- ❑ Pentru trunchierea unei tabele este necesar privilegiul DROP ANY TABLE

# ROLURI

- ❑ Rolurile sunt colectii de privilegii (ca niste 'coșuri' de privilegii) care pot fi asignate si revocate impreuna.
- ❑ Un rol poate fi creat, i se pot asocia privilegii (umplem coșul) dupa care el se poate asigna cu GRANT unui user sau unui alt rol.
- ❑ Exista o serie de roluri predefinite (CONNECT, RESOURCE, DBA, etc).

# ROLURI - Oracle 9

❑ Rolul CONNECT contine privilegiile:

- ❑ ALTER SESSION,
- ❑ CREATE CLUSTER,
- ❑ CREATE DATABASE LINK,
- ❑ CREATE SEQUENCE,
- ❑ CREATE SESSION,
- ❑ CREATE SYNONYM,
- ❑ CREATE TABLE,
- ❑ CREATE VIEW

# ROLURI - Oracle 9

❑ Rolul RESOURCE contine privilegiile:

- ❑ CREATE CLUSTER,
- ❑ CREATE INDEXTYPE,
- ❑ CREATE OPERATOR,
- ❑ CREATE PROCEDURE,
- ❑ CREATE SEQUENCE,
- ❑ CREATE TABLE,
- ❑ CREATE TRIGGER,
- ❑ CREATE TYPE

# ROLURI - Oracle 10

❑ Nota: In documentatia Oracle 10 se mentioneaza:

"Customers should discontinue using the CONNECT and RESOURCE roles, as they will be deprecated in future Oracle Database releases. The CONNECT role presently retains only the CREATE SESSION privilege."

(sursa: [http://docs.oracle.com/cd/B19306\\_01/network.102/b14266/authoriz.htm](http://docs.oracle.com/cd/B19306_01/network.102/b14266/authoriz.htm))

# ROLURI - Oracle 12c

- ❑ Legat de aceste roluri, in Oracle 12c:
  - ❑ CONNECT: Contine doar CREATE SESSION si SET CONTAINER – ca privilegii de sistem
  - ❑ RESOURCE: Contine toate privilegiile mentionate anterior dar se specifica faptul ca in versiunile ulterioare de Oracle ar putea sa nu mai fie prezent.
- ❑ Dezvoltatorii sunt sfatuiti sa-si creeze propriile roluri, adaptate nevoilor lor de securitate.

# Asignarea privilegiilor: GRANT

- Sintaxa pentru comanda GRANT este:

```
GRANT { priv_sistem | rol },  
       { priv_sistem | rol }...  
TO { user | rol | PUBLIC },  
   { user | rol | PUBLIC }...
```

[WITH ADMIN OPTION]

- Se asigneaza lista de privilegii si/sau roluri unui user sau unui rol
- In cazul in care se specifica PUBLIC privilegiile respective sunt asignate tuturor userilor.

# Asignarea privilegiilor: GRANT

- ❑ In cazul specificarii WITH ADMIN OPTION cel care primeste privilegiul il poate si el asigna mai departe, inclusiv cu WITH ADMIN OPTION.
- ❑ Userii care au privilegiul GRANT ANY ROLE pot sa asigneze orice rol in sistem.
- ❑ Cel care primeste un privilegiu cu WITH ADMIN OPTION il poate de asemenea revoca de la orice user sau rol din sistem (nu numai de la cei carora el l-a asignat).
- ❑ In general privilegiile SYSDBA si SYSOPER (s-a discutat despre ele anterior) trebuie asignate doar userilor de tip administrator pentru ca ele dau acces la orice operatie in baza de date (SYSDBA).

# Vizualizare privilegii

- Există vederile DBA\_SYS\_PRIVS și SESSION\_PRIVS care pot fi interogate pentru a vedea privilegiile asociate fiecarui user (DBA\_SYS\_PRIVS) sau sesiunii curente (SESSION\_PRIVS).
- Privilegiile afisate provin atât din privilegii asignate individual cat și din privilegii asociate cu rolurile asignate userilor.

# Revocarea privilegiilor: REVOKE

- Sintaxa este:

```
REVOKE { priv_sistem | rol },  
        { priv_sistem | rol }...  
FROM { user | rol | PUBLIC },  
      { user | rol | PUBLIC }...
```

- Revoca acele privilegii care au fost asignate cu GRANT.
- Revocarea unor privilegii poate face ca anumite proceduri sau vederi care aveau nevoie de acel privilegiu sa devina invalide.
- Revocarea unui privilegiu de la un user nu afecteaza userii carora acesta le-a transmis privilegiul – deci REVOKE nu are efect in cascada.

# Privilegii obiect

## □ Sintaxa:

```
GRANT {object_priv | ALL [PRIVILEGES] }  
[ (column [, column] ...) ]  
[, {object_priv | ALL [PRIVILEGES] }  
[ (column [, column] ...) ] ] ...  
ON [schema.]object  
TO {user | role | PUBLIC}  
[, {user | role | PUBLIC} ] ...  
[WITH GRANT OPTION]
```

# Privilegii obiect

Pe post de object\_priv poate fi:

- ALTER
- DELETE
- EXECUTE
- INDEX
- INSERT
- REFERENCES
- SELECT
- UPDATE

# Revocare privilegi obiect

```
REVOKE {object_priv | ALL [PRIVILEGES] }  
[ (column [, column] ...) ]  
[, {object_priv | ALL [PRIVILEGES] }  
[ (column [, column] ...) ] ] ...  
ON [schema.]object  
FROM {user | role | PUBLIC}  
[, {user | role | PUBLIC}] ...  
[CASCADE CONSTRAINTS]
```

- ❑ Ultima optiune elimina constrangerile referentiale afectate.

# Exemple

## ❑ Asignare:

```
GRANT SELECT, INSERT, UPDATE, DELETE ON emp TO  
stud1;
```

```
GRANT ALL ON emp TO stud1;
```

```
GRANT SELECT ON emp TO public;
```

```
GRANT UPDATE(punctaj) ON stud TO scott
```

## ❑ Revocare:

```
REVOKE DELETE ON emp FROM stud1;
```

```
REVOKE ALL ON emp FROM stud1;
```

```
REVOKE ALL ON emp FROM public;
```

# Listare privilegii si roluri

```
SELECT LPAD(' ', 2*level) || granted_role
      "USER PRIVS"
  FROM (
    SELECT NULL grantee,   username granted_role
      FROM dba_users
     WHERE username LIKE UPPER('%&uname%')
UNION
    SELECT grantee, granted_role
      FROM dba_role_privs
UNION
    SELECT grantee, privilege
      FROM dba_sys_privs)
START WITH grantee IS NULL
CONNECT BY grantee = prior granted_role;
```

SQL Plus\_11g

```
SQL> r
 1 SELECT LPAD(' ', 2*level) || granted_role "USER PRIUS"
 2 FROM (
 3   SELECT NULL grantee, username granted_role
 4   FROM dba_users
 5   WHERE username LIKE UPPER('&uname%')
 6     UNION
 7   SELECT grantee, granted_role
 8   FROM dba_role_privs
 9     UNION
10   SELECT grantee, privilege
11   FROM dba_sys_privs)
12* START WITH grantee IS NULL
12* CONNECT BY grantee = prior granted_role
Introduceți valoarea pentru uname: STUD1
vechi 4:      WHERE username LIKE UPPER('&uname%')
nou  4:      WHERE username LIKE UPPER('STUD1%')
```

### USER PRIUS

---

```
STUD1
CONNECT
  CREATE SESSION
CREATE VIEW
RESOURCE
  CREATE CLUSTER
  CREATE INDEXTYPE
  CREATE OPERATOR
  CREATE PROCEDURE
  CREATE SEQUENCE
  CREATE TABLE
  CREATE TRIGGER
  CREATE TYPE
UNLIMITED TABLESPACE
```

14 înregistrări selectate.

SQL>

# Lecturi obligatorii

1. Oracle Database Security Guide – Cap 10: Administering User Privileges, Roles, and Profiles  
[http://download.oracle.com/docs/cd/B14117\\_01/network.101/b10773/admusers.htm](http://download.oracle.com/docs/cd/B14117_01/network.101/b10773/admusers.htm)

# Sfârșitul capitolului 6

# Prelucrarea datelor cu tehnici de Data Mining

# Cuprins

---



- Ce este data mining
- Etapele procesului de data mining
- Metode si subdomenii in data mining
- Preprocesarea datelor
- Sumar

# Definitie ([Liu 11])

---



- Data mining este cunoscut si sub numele KDD - Knowledge Discovery in Databases (descoperirea de cunostinte in date).
- In mod obisnuit este definit ca procesul de descoperire a unor sabloane/modele (eng. patterns) / cunostinte utile in diverse surse de date: baze de date, colectii de texte, de imagini sau de pagini web, etc.
- Sabloanele trebuie sa fie valide, utile si intelligibile.

# Definitie ([Ullman 09, 10])

---



- Descoperirea de sabloane utile, uneori neasteptate, in date.
- Descoperirea de "modele" pentru date prin tehnici de tipul:
  - Modelare statistica
  - Invatare automata (Machine learning)
  - Abordari computationale in modelare
  - Sumarizare
  - Extragerea proprietatilor (Feature Extraction)

# Definitie ([Wikipedia])

---



- Data mining (etapa de analiza a procesului KDD - knowledge discovery in databases) este procesul de descoperire de noi sabloane in seturi mari de date prin metode aflate la intersectia dintre intelectuala artificiala, invatarea automata, statistica si baze de date.
- Obiectivul procesului de data mining este acela de a extrage cunostinte dintr-un set de date intr-o forma inteligibila pentru utilizatorii umani.
- Procesul implica baze de date si gestiunea informatiei, preprocesarea datelor, probleme de modelare si inferenta, metrici pentru performanta rezultatelor, probleme de complexitate, post procesare si vizualizare.

# Definitie ([Kimball, Ross 02])

---



- O categorie de cereri execute adesea pe datele atomice pentru a gasi sabloane/modele neasteptate in date.
- Cele mai cunoscute subdomenii ale data mining sunt clasificarea, gruparea (clustering), estimarea, predictia si gasirea evenimentelor care apar impreuna.
- Exista multe tipuri de unelte pentru data mining. Cele mai cunoscute sunt arborii de decizie, retelele neurale, unelte pentru vizualizarea seturilor de date, algoritmi genetici, logica fuzzy si unelte folosite in statistica obisnuita.
- In cazul aplicatiilor concrete pentru mediul economic data mining foloseste de obicei date aflate intr-un depozit de date (**data warehouse**).

# Concluzii

---



- Procesul de data mining **converteste datele in cunostinte valoroase** care pot fi folosite pentru suportul deciziei.
- Domeniul Data mining consta intr-o **colectie de metodologii** de analiza a datelor, tehnici si algoritmi pentru descoperirea de noi sabloane.
- Data mining se foloseste cu precadere pentru **seturi mari de date**.
- Procesul de data mining este **automat** (nu necesita interventie umana).
- Data mining si Knowledge Discovery in Databases (**KDD**) sunt considerate de multi autori ca fiind acelasi lucru dar exista autori pentru care data mining este doar etapa de analiza si de extragere a sabloanelor a procesului de descoperire a cunostintelor in date (**KDD**), etapa care se desfasoara dupa curatarea si transformarea datelor si inainte de vizualizarea si evaluarea rezultatelor.

# Exemple de succes

---



In cursurile profesorului J. Ullman de la Stanford sunt enumerate cateva exemple de succes ale folosirii tehniciilor de data mining (preluare din [Ullman 03]):

- **Arbore de decizie** construiti din istoria imprumuturilor bancare pentru a genera algoritmi de accordare a imprumuturilor.
- **Sabioane** ale comportamentului calatorilor pentru a optimiza vanzarea cu pret redus a locurilor la avion sau a camerelor de hotel.
- **"Scutece si bere"** S-a observat ca cei care cumpara scutece sunt mai inclinati decat ceilalți să cumpere bere. Datorită acestui rezultat, cele două produse se pot plasa într-un supermarket unul în apropierea celuilalt, astfel încât mulți cumpăratori vor circula între cele două raioane. Plasarea cipsurilor pe traseu a dus la creșterea vanzarilor pentru toate cele 3 produse.

# Exemple de succes

---



- **Skycat si Sloan Sky Survey:** gruparea obiectelor ceresti dupa nivelele radiatiei electromagnetice intr-un numar de benzi de frecventa a permis identificarea galaxiilor, stelelor apropiate si a altor categorii de obiecte celeste.
- Compararea **genotipului** persoanelor avand sau neavand o anumita problema de sanatate a permis descoperirea unor gene care sunt asociate cu respectiva problema (de exemplu diabet). In acest fel se pot lua masuri de preventie inainte de declansarea bolii in cazul persoanelor care prezinta respectivele caracteristici.

# Ce NU este Data Mining:

---



- Cautarea unei persoane in baza de date a unei organizatii.
- Calcularea de valori minime, maxime, medii, sume sau numararea valorilor aflate in tabelele unei baze de date (operatii pur statistice).
- Folosirea unui motor de cautare pentru a gasi referintele asociate numelui tau.

# Date, informatii, cunostinte

---



- Să zicem că ai un magazin și înregistrezi toate detaliile clientilor tăi în baza de date a acestuia. Știi numele clientilor și produsele pe care le cumpără în fiecare zi.
- De exemplu, Alex, Jessica și Paul vizitează magazinul tau în fiecare duminică și cumpără lumânări. Stochezi aceste informații în aplicația informatică a magazinului. Acestea sunt **date**. Ori de câte ori doresti să afli cine sunt clientii care cumpără lumânări, poți interoga baza de date și primesti răspunsul. Aceasta este o **informație**. Dacă vrei să știi câte lumânări sunt vândute în fiecare zi din săptămână în magazinul tău, poți să interoghezi din nou baza de date și vei primi răspunsul - aceasta este o altă **informație**.
- Să presupunem că există și alți 1000 de clienti care, de asemenea, cumpără lumânări de la tine în fiecare duminică (majoritatea - cu un anumit procent de variație) și toți aceștia sunt creștini prin religie. Deci, poți concluziona că Alex, Jessica și Paul trebuie să fie și ei creștini.

Sursa: <http://www.dwbiconcepts.com/data-warehousing/11-data-mining/97-data-mining-for-beginners.html>

# Date, informații, cunoștințe

---



- ❑ Acum, religia lui Alex, a lui Jessica și a lui Pavel nu o ai ca **date** în datele stocate. Acest lucru nu poate fi regăsit din baza de date ca **informație**. Dar ai obținut indirect această informație. Aceasta este o "**cunoștință**" pe care ai descoperit-o. și această descoperire a fost făcută printr-un proces numit "**Data Mining**".
- ❑ **Există șanse să te înseli în legătură cu Alex, Jessica și Paul.** Dar există o mulțime de șanse să ai dreptate. De aceea este foarte important să "**evaluăm**" rezultatul procesului de **Data Mining**.
- ❑ Am dat acest exemplu pentru că am vrut să fac o distincție clară între cunoștințe și informații în contextul domeniului Data Mining. Este important de înțeles. . . de ce **regăsirea informațiilor** de oriunde s-ar afla în baza de date **nu este același lucru cu Data Mining**. Indiferent cât de complex este procesul de regăsire a informației, indiferent cât de adânc este localizată informația, asta nu este Data Mining.

Sursa: <http://www.dwbiconcepts.com/data-warehousing/11-data-mining/97-data-mining-for-beginners.html>

# Software pentru DM

---



In lucrarea ([Mikut, Reischl 11]) programele software pentru data mining sunt clasificate in 9 categorii:

- Suite pentru data mining** (Data mining suites -DMS) sunt pachete dedicate aplicatiilor de DM. Includ numeroase metode / algoritmi.  
Exemple:
  - Comerciale: IBM SPSS Modeler, SAS Enterprise Miner, DataEngine, GhostMiner, Knowledge Studio, NAG Data Mining Components, STATISTICA
  - Open source: RapidMiner
- Pachete de Business intelligence** (Business intelligence packages - BIs) includ functionalitati de baza pentru DM - de exemplu metode statistice pentru aplicatii de business. Exemple:
  - Comerciale: IBM Cognos 8 BI, Oracle DataMining, SAPNetweaver Business Warehouse, Teradata Database, DB2 Data Warehouse from IBM
  - Open source: Pentaho

# Software pentru DM

---



- **Pachete matematice** (Mathematical packages - MATs) contin o multime mare si extensibila de algoritmi precum si rutine de vizualizare. Exemple:
  - Comerciale: MATLAB, R-PLUS
  - Open source: R, Kepler
- **Pachete integratoare** (Integration packages - INTs) sunt colectii extensibile de algoritmi de tip open-source. Exemple:
  - Stand-alone software (KNIME, versiunea GUI pentru WEKA, KEEL, TANAGRA)
  - Extensiile pentru uneltele de tip MAT descrise mai sus.
- **Extensiile** (EXT) sunt accesorii (add-ons) pentru alte unelte ca Excel, Matlab, R, cu o functionalitate limitata dar utila in diverse cazuri. Exemple:
  - Retele neuronale pentru Excel (Forecaster XL, XLMiner)
  - MATLAB (Matlab Neural Networks Toolbox).

# Software pentru DM

---



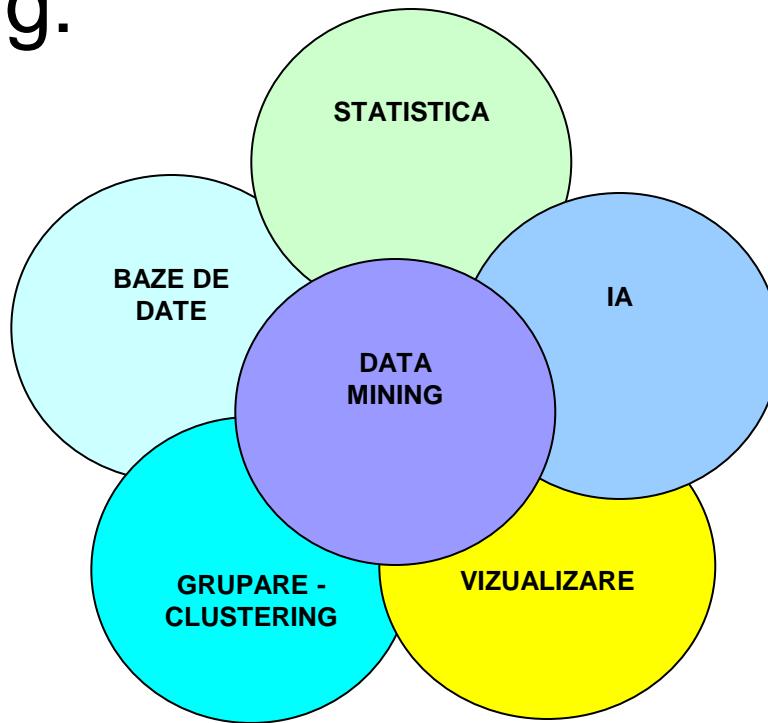
- **Biblioteci de DM** (Data mining libraries - LIBs) contin functii care pot fi folosite in alte unelte software. Exemple: Neurofusion for C++, WEKA, MLC++, JAVA Data Mining Package, LibSVM
- **Specialitati** (Specialties - SPECs) sunt similare cu DMS dar implementeaza doar o categorie/familie de metode. Exemple: CART, Bayesia Lab, C5.0, WizRule, Rule Discovery System, MagnumOpus, JavaNNS, Neuroshell, NeuralWorks Predict, RapAnalyst.
- **Cercetare** (Research - RES) sunt primele implementari ale unor noi algoritmi, avand de obicei un suport grafic restrans si fara foarte multe facilitati in utilizare. Sunt de obicei open source. WEKA si RapidMiner au debutat in aceasta categorie.
- **Solutii** (Solutions - SOLs) descriu unelte care sunt adaptate la un domeniu ingust de aplicatii. Exemple: pentru text mining: GATE; pentru prelucrare de imagini: ITK, ImageJ; cercetarea farmaceutica: Molegro Data Modeler

# Comunitati implicate

---



Cele mai importante comunitati implicate in data mining:



# Cuprins

---



- Ce este data mining
- Etapele procesului de data mining
- Metode si subdomenii in data mining
- Preprocesarea datelor
- Sumar

# Etapele procesului de DM (1)

---



1. **Colectarea datelor**: Datele sunt culese din baze de date existente sau prin parcurgerea paginilor web (Web crawling).
2. **Preprocesarea datelor**, presupune mai multe activitati desfasurate in scopul pregatirii datelor pentru aplicarea algoritmilor specifici

# Preprocesarea datelor

---



- Curatarea datelor (Data cleaning):
  - inlocuirea sau inlaturarea valorilor lipsa,
  - netezirea datelor care contin zgomot,
  - identificarea si eventual inlaturarea punctelor izolate (eng.: **outliers** - valori izolate, departate de toate celelalte),
  - inlaturarea inconsistentelor.

Nota: in multe documente termenul '**outliers**' a fost tradus in limba romana prin '**valori aberante**' - prin similitudine cu traducerea in limba franceza.

# Preprocesarea datelor

---



- ❑ **Integrarea datelor** (Data integration):
  - ❑ integrarea intr-un corp unic a datelor provenite din mai multe surse,
  - ❑ Conversii de tip si de structura,
  - ❑ eliminarea duplicatelor,
  - ❑ tratarea inconsistentelor datorate integrarii.

# Preprocesarea datelor

---



- Transformarea datelor (Data transformation):
  - normalizarea (sau standardizarea) datelor,
  - summarizari,
  - generalizari,
  - constructia de noi atribute, etc.

# Preprocesarea datelor

---



- **Reducerea datelor** (Data reduction): nu toate atributele existente sunt necesare pentru un anumit proces de data mining.
- Prin reducere se opresc doar acele atribut care sunt necesare diminuand astfel volumul de date prelucrate (si implicit timpul de rulare al algoritmului).

# Preprocesarea datelor

---



- **Discretizare:** Anumiti algoritmi lucreaza doar pe date discrete. De aceea pentru atributele avand valori continue trebuie efectuata discretizarea constand in inlocuirea acestor valori cu altele dintr-o multime de valori discrete.
  - De exemplu varsta se poate inlocui cu un atribut avand doar trei valori: Tanar, Adult si Batran.
-

# Etapele procesului de DM (2)

---



3. **Extragerea sabloanelor si descoperirea de cunostinte.** Aceasta este etapa in care sunt efectiv utilizati algoritmii de DM pentru obtinerea rezultatului. Unii autori reduc domeniul Data Mining la aceasta etapa, intregul proces fiind numit KDD.
4. **Vizualizarea:** deoarece data mining extrage proprietati/informatii ascunse din date este necesara o vizualizare a rezultatelor pentru o mai buna inteleghere a lor si pentru a le evalua. Vizualizarea este uneori necesara si pentru datele de intrare.

# Etapele procesului de DM (3)

---



3. **Evaluarea rezultatului:** nu tot ceiese dintr-un proces de data mining este valoros.
- Unele rezultate sunt adevaruri pur statistice care se puteau deduce si fara aplicarea algoritmilor iar altele nu prezinta interes pentru utilizator.
  - De aceea este necesara evaluarea rezultatelor de catre experti pentru a separa cunostintele valoroase de celelalte lucruri obtinute in urma rularii algoritmului.

# Principiul lui Bonferroni

---



O 'cunostinta' descoperita in urma procesului de data mining poate fi un **adevar pur statistic**. Exemplu (din [Ullman 03]):

- In 1950 David Rhine, un parapsiholog de la universitatea Duke a testat studentii pentru a afla daca au sau nu perceptie extrasenzoriala (ESP).
- Pentru asta le-a cerut sa ghiceasca culoarea a 10 carti de joc successive - rosu sau negru. Rezultatul a fost ca 1/1000 din participanti au ghicit toate cele 10 carti - in consecinta el i-a declarat ca avand ESP.

# Principiul lui Bonferroni

---



- Re-testarea efectuata doar asupra acestora nu a mai avut aceleasi rezultate, el considerand ca in momentul in care au aflat ca au ESP au pierdut aceasta capacitate.
- David Rhine nu a realizat ca statistica spune ca probabilitatea de a ghici 10 carti succesive este de  $1/2^{10} = 1/1024$  deoarece probabilitatea de a ghici o carte este  $1/2$  (rosu sau negru)!

# Principiul lui Bonferroni

---



- Astfel de rezultate pot fi regasite in iesirea unui algoritm de data mining si trebuie recunoscute ca adevaruri statistice si nu ca rezultate reale ale procesului de data mining.
- Astfel de rezultate sunt obiectul **principiului lui Bonferroni**. Acesta poate fi sintetizat astfel:

*Daca sunt prea multe concluzii, unele vor fi adevаратe din motive pur statistice.*

Data Mining, noun: "**Torturing data until it confesses ... and if you torture it enough, it will confess to anything**" - Jeff Jonas, IBM ([http://www.cs.ccsu.edu/~markov/ccsu\\_courses/DataMining-1.html](http://www.cs.ccsu.edu/~markov/ccsu_courses/DataMining-1.html))

# Cuprins

---



- Ce este data mining
- Etapele procesului de data mining
- Metode si subdomenii in data mining
- Preprocesarea datelor
- Sumar

# 2 tipuri de metode

---



- **Metode predictive:** Aceste metode utilizeaza anumite variabile pentru a prezice valoarea altor variabile. Un exemplu din aceasta categorie este clasificarea: bazat pe date cunoscute si deja etichetate (clasificate), algoritmii de clasificare creaza modele care pot fi folosite pentru clasificarea unor date noi, necunoscute.
- **Metode descriptive:** Algoritmii din aceasta categorie gasesc sabloane / modele care descriu structura interna a unui set de date. De exemplu algoritmii de grupare (clustering) gasesc grupuri de obiecte similare in setul de date (grupuri numite clustere) si de asemenea detecteaza posibile obiecte izolate, departate de oricare dintre clustere - asa numitii 'outliers'.

# Algoritmi

---



## ❑ Algoritmi de predictie:

- ❑ Clasificare
- ❑ Regresie
- ❑ Detectia deviatiei

## ❑ Algoritmi de descriere:

- ❑ Grupare - Clustering
  - ❑ Gasirea de reguli de asociere
  - ❑ Descoperirea de sabloane secentiale
-

# Clasificare



## Intrare:

- Un set avand un numar k de clase  $C = \{c_1, c_2, \dots, c_k\}$
- Un set de  $n$  articole etichetate  $D = \{(d_1, c_{i1}), (d_2, c_{i2}), \dots, (d_n, c_{in})\}$ . Articolele sunt  $d_1, \dots, d_n$ , fiecare articol  $d_j$  fiind etichetat cu clasa  $c_{ij} \in C$ . D este numit si setul (multimea) de **antrenare (training set)**.
- Pentru calibrarea unor algoritmi este necesar si un set (multime) **de validare (validation set)**. Acesta contine de asemenea articole etichetate, neincluse in setul de antrenare

## Iesire:

- Un model sau metoda de a clasifica noi articole (clasificator). Setul continand noile articole se numeste set **de test (test set)**

# Exemplu

---

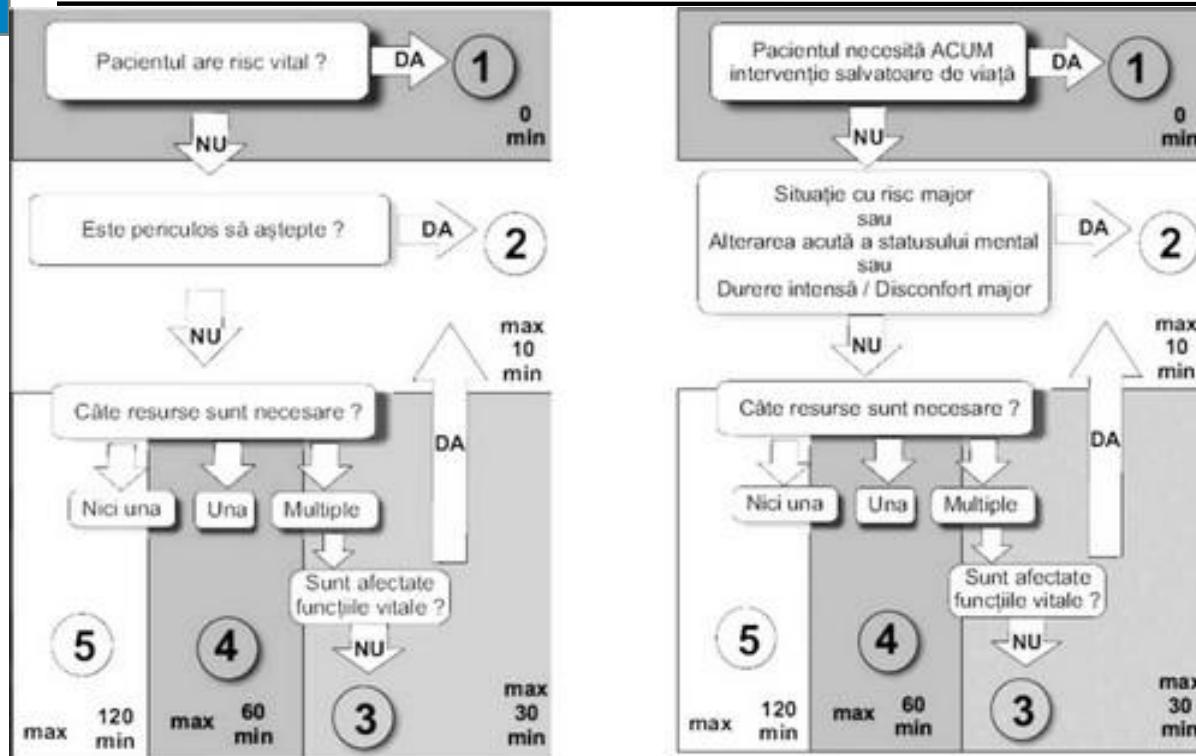


- Sa consideram ca articolele sunt pacienti ai unui serviciu de urgente medicale.
- Exista 5 clase,  $C_0$ ,  $C_{10}$ ,  $C_{30}$ ,  $C_{60}$  si  $C_{120}$ , unde eticheta  $C_k$  inseamna ca pacientul poate fi lasat sa astepte maxim  $k$  minute fara ca starea sa se inrautateasca.
- Vom reprezenta datele setului de antrenare sub forma tabelara.
- Iesirea unui algoritm de creare a unui clasificator poate fi de exemplu un arbore de decizie sau un set ordonat de reguli.
- Modelul obtinut poate fi utilizat apoi pentru a clasifica noi pacienti asignandu-le o eticheta din multimea celor 5 de mai sus.

# Multimea de antrenare UPU

Name (or ID)	Vital risk?	Danger if waits?	0 resource needed	1 resource needed	>1 resource needed	>1 resource needed and vital functions affected	Waiting time (class label)
John	Yes	Yes	No	Yes	No	No	C0
Maria	No	Yes	No	No	Yes	No	C10
Nadia	Yes	Yes	Yes	No	No	No	C0
Omar	No	No	No	No	Yes	Yes	C30
Kiril	No	No	No	Yes	No	Yes	C60
Denis	No	No	No	No	Yes	No	C10
Jean	No	No	Yes	Yes	No	No	C120
Patricia	Yes	Yes	No	No	Yes	Yes	C60

# Rezultat: arbori de decizie



Pentru un nou pacient:

Felix	Yes	Yes	No	No	Yes	?????
-------	-----	-----	----	----	-----	-------

arborele de decizie produce clasa C0

# Regresia (1)

---



- Regresia provine din statistica.
- Inseamna: **prezicerea (predictia)** valorii unei anumite variabile continue pe baza valorilor altor variabile, considerand un model de dependenta liniara sau neliniara ([Tan, Steinbach, Kumar 06]).
- Utilizata in **predictie si prognoza**; este folosita si in domeniul invatarii automate.
- Analizele bazate pe regresie sunt folosite pentru intelegerarea relatiilor dintre variabile dependente si independente.

Intrebare: care e deosebirea intre predictie si prognoza?

# Regresia (2)

---



Există multe tipuri de regresie, de exemplu:

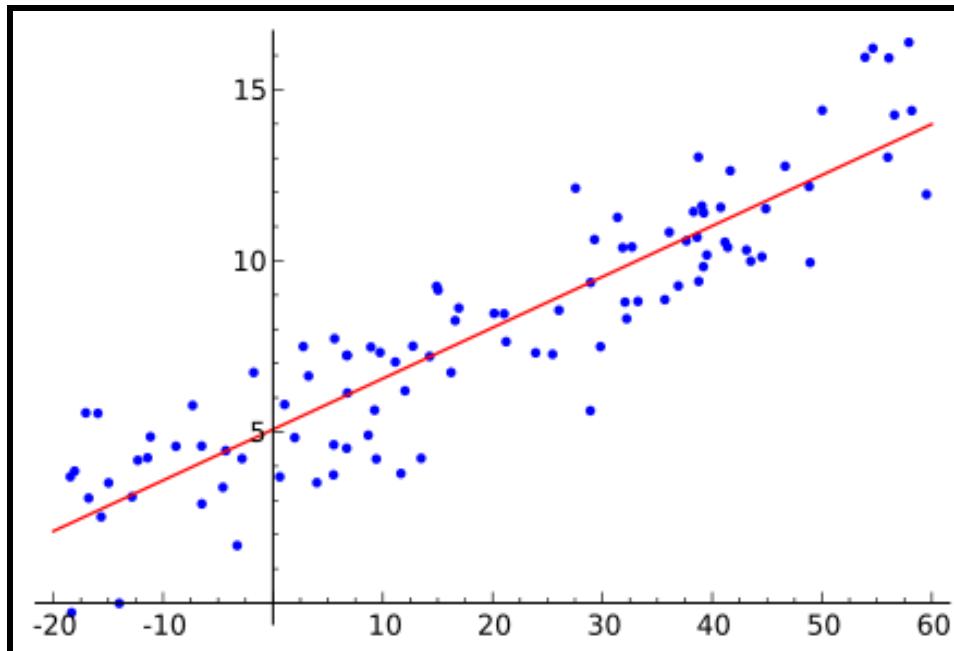
- Regresie liniara
- Regresie liniara simplă
- Regresie logistica
- Regresie neliniara
- Regresie nonparametrică
- Regresie robustă

# Exemplu



## Exemplu de regresie liniara

(sursa: [http://en.wikipedia.org/wiki/File:Linear\\_regression.svg](http://en.wikipedia.org/wiki/File:Linear_regression.svg))



# Detectia deviatiei

---



- Detectia deviatiei (sau a anomalilor) presupune descoperirea de deviatii semnificative de la comportamentul normal. **Punctele izolate (outliers)** sunt o categorie semnificativa de astfel de date anormale.
- Detectia deviatiei poate fi utilizata in multe situatii:
  - In faza de rulare a algoritmului de data mining, datele anormale putand avea un efect puternic asupra algoritmului
  - Audit: astfel de informatii pot arata existenta unor probleme sau practici gresite
  - Detectia fraudelor: cererile frauduloase contin deseori informatii inconsistente
  - Detectia intruziunilor intr-o retea de calculatoare poate fi facuta si pe baza unor date anormale
  - Curatarea datelor (data cleaning): astfel de informatii anormale pot reprezenta date eronate care trebuie corectate

# Metode de detectie a deviatiei

---



- Tehnici bazate pe distante (ex. : k-nearest neighbor).
  - Folosirea "One Class Support Vector Machines" (Ca un SVM clasic dar toate exemplele de antrenare sunt din aceeasi clasa si doar originea reprezinta cea de-a doua clasa).
  - Metode predictive (arbori de decizie, retele neuronale).
  - Detectia punctelor izolate folosind clustering.
  - Inregistrari care nu respecta regulile de asociere
  - Analize Hotspot (clustere avand o valoare ridicata a anumitor parametri; de exemplu politia foloseste astfel de analize pentru analiza zonelor unde criminalitatea are valori ridicate)
-

# Algoritmi

---



## ❑ Algoritmi de predictie:

- ❑ Clasificare
- ❑ Regresie
- ❑ Detectia deviatiei

## ❑ Algoritmi de descriere:

- ❑ Grupare - Clustering
  - ❑ Gasirea de reguli de asociere
  - ❑ Descoperirea de sabloane secentiale
-

# Clustering

---



## Intrare:

- Un set de obiecte  $D = \{d_1, d_2, \dots, d_n\}$  (numite uzual puncte). Obiectele **nu sunt etichetate** si nu exista definit vreun set de clase.
- O **functie de distanta** (masura a disimilaritatii) care poate fi utilizata pentru a calcula distanta dintre oricare doua puncte. O distanta mica inseamna "aproape" iar una mare "departe".
- Unii algoritmi necesita introducerea unei valori pentru **numarul de clustere** de obtinut.

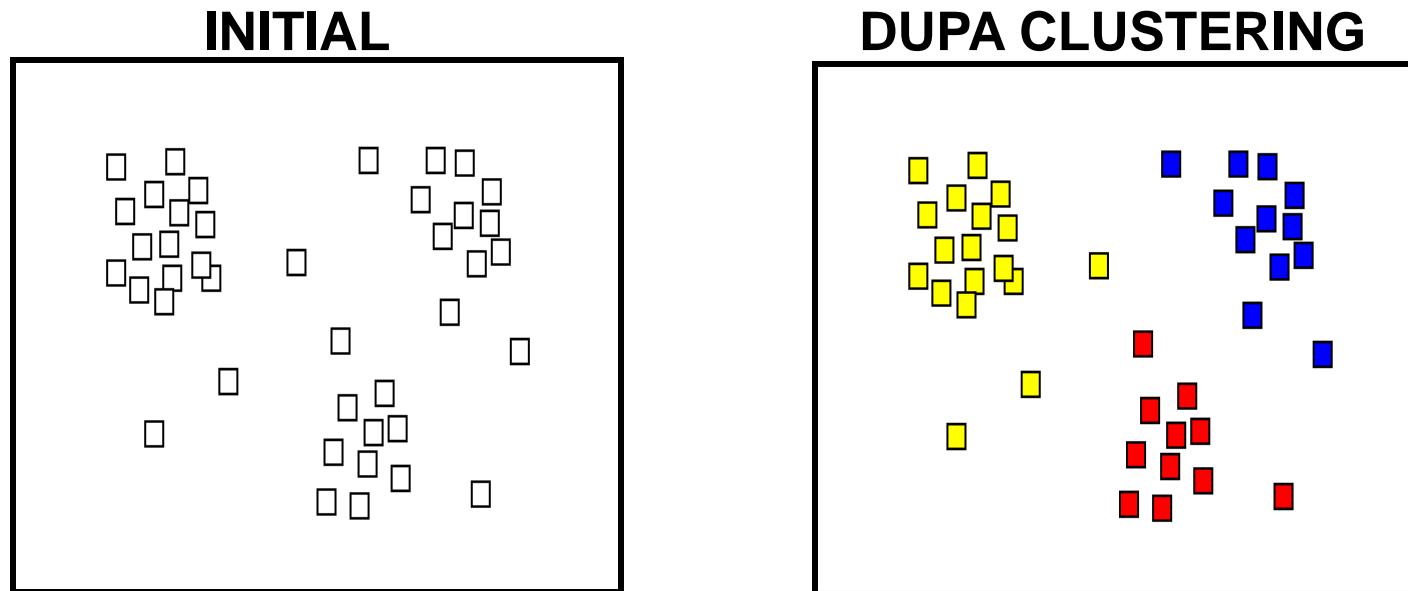
## Iesire:

- Un set de grupuri de obiecte/puncte, numite clustere unde punctele din acelasi cluster sunt 'apropiate' iar cele din clustere diferite 'departate' unele de altele, considerand functia de distanta existenta.

# Exemplu



- Daca avem un set de 2 puncte intr-un spatiu 2D, sa se gaseasca grupurile/clusterele naturale formate de ele



Sursa: <http://en.wikipedia.org/wiki/File:Cluster-1.svg>, <http://en.wikipedia.org/wiki/File:Cluster-2.svg>

# Reguli de asociere

---



## Intrare:

- Un set de **m** articole  $I = \{i_1, i_2, \dots, i_m\}$ .
- Un set de **n** tranzactii  $T = \{t_1, t_2, \dots, t_n\}$ , fiecare tranzactie continand un subset al lui  $I$ . Deci daca  $t_k \in T$  atunci  $t_k = \{i_{k1}, i_{k2}, \dots, i_{kj}\}$  unde  $j$  depinde de  $k$  (tranzactiile au lungimi diferite).
- Un prag **s** numit prag de suport, dat fie in procente fie in valoare absoluta. Daca o multime de articole (itemset)  $X \in I$  este inclusa in **w** tranzactii atunci **w** este suportul lui  $X$ . Daca **w >= s** atunci  $X$  este numita **multime frecventa de articole**.
- Un al doilea prag **c** pentru increderea regulilor obtinute.

## Iesire:

- Multimile frecvente de articole** din  $T$ , avand suportul  $>= s$
- Multimea de reguli** derive din  $T$  avand suportul  $>= s$  si increderea  $>= c$

# Reguli de asociere

---



- O **regula** este o constructie de tipul  $X \rightarrow Y$  unde  $X$  si  $Y$  sunt multimi de articole (itemsets).
- **Suportul** unei reguli  $X \rightarrow Y$  este numarul de aparitii ale  $X \cup Y$  in  $T$  (egal cu suportul acestei reuniuni ca itemset):  
$$\text{support}(X \rightarrow Y) = \text{support}(X \cup Y)$$
- **Increderea** unei reguli este proportia tranzactiilor continandu-l pe  $Y$  in multimea tranzactiilor continandu-l pe  $X$ :  
$$\text{confidence}(X \rightarrow Y) = \text{support}(X \cup Y) / \text{support}(X)$$
- Acceptam o regula ca valida doar daca suportul si increderea ei sunt mai mari sau egale cu pragurile date.

# Exemplu



- Sa consideram urmatorul set de tranzactii:

Transaction ID	Items
1	Bread, Milk, Butter, Orange Juice, Onion, Beer
2	Bread, Milk, Butter, Onion, Garlic, Beer, Orange Juice, Shirt, Pen, Ink, Baby diapers
3	Milk, Butter, Onion, Garlic, Beer
4	Orange Juice, Shirt, Shoes, Bread, Milk
5	Butter, Onion, Garlic, Beer, Orange Juice

- Daca  $s = 60\%$  atunci  $\{Bread, Milk, Orange\ Juice\}$  sau  $\{Onion, Garlick, Beer\}$  sunt multimi frecvente de articole.
- De asemenea, daca  $s = 60\%$ ,  $c=70\%$  atunci regula  $\{\underline{Onion}, \underline{Beer}\} \rightarrow \{Garlic\}$  e valida avand suport de  $60\%$  si incredere de  $75\%$ .

# Descoperire sabloane secentiale

---



## Intrare:

- O multime de **m** articole  $I = \{i_1, i_2, \dots, i_m\}$ . O **seventă** este o lista ordonata de multimi de articole din  $I$ .
- O multime de secente **S** (numita si 'sequence database').
- O **functie booleana** care poate testa daca o secentă  $S_1$  este inclusa (este o subsecentă) in secentă  $S_2$ . In acest caz  $S_2$  este numita o supersecentă a lui  $S_1$ .
- Un **prag s** (procent sau valoare absoluta) necesar pentru a gasi secente frecvente.

## Iesire:

- Multimea de secente frecvente**, i.e. multinea de secente incluse in cel putin **s** secente din  $S$ .
- Uneori se poate obtine si **un set de reguli**, fiecare regula fiind de forma  $S_1 \rightarrow S_2$  unde  $S_1$  si  $S_2$  sunt secente.

# Exemplu

---



- Într-o librerie putem gasi sevante ca:  
**{Book\_on\_C, Book\_on\_C++, Book\_on\_Perl}**
  
- Din aceasta sevanta se poate deriva o regula de tipul:  
după cumpărarea unor carti de C și C++, clientul  
cumpără carti de Perl:  
**Book\_on\_C, Book\_on\_C++ → Book\_on\_Perl**

# Cuprins

---



- Ce este data mining
- Etapele procesului de data mining
- Metode si subdomenii in data mining
- Preprocesarea datelor**
- Sumar

# Preprocesarea datelor

---



- Tipuri de date
- Masurarea datelor
- Curatarea datelor
- Integrarea datelor
- Transformarea datelor
- Reducerea datelor
- Discretizarea datelor

# Tipuri de date

---



- ❑ Categorice vs. Numerice
- ❑ Scale de masurare
  - ❑ Nominala
  - ❑ Ordinala
  - ❑ Interval
  - ❑ Proportionala (eng.: ratio scale)

# Date categorice si numerice

---



- **Datele categorice** constau în nume reprezentând categorii. Exemple: culoare (cu categoriile roșu, verde, albastru și alb) sau gen (masculin, feminin)
- Valorile de acest tip nu sunt de obicei ordonate (nu există o relație de ordine între ele), operațiile posibile fiind testul de egalitate sau de apartenență (incluziune) la o mulțime.

# Date categorice si numerice

---



- **Datele numerice** constau în numere aparținând unei mulțimi continue sau discrete de valori numerice.
- Valorile sunt ordonate, astfel încât se poate testa ordinea ( $<$ ,  $\leq$ ,  $>$ ,  $\geq$ ).
- Uneori este necesară conversia datelor categorice în date numerice asignând o valoare numerică (sau cod) pentru fiecare eticheta (categorie).

# Scale de masurare

---



- Stanley Smith Stevens, directorul laboratorului de psihohigie-metria de la Universitatea Harvard a afirmat într-un articol publicat în 1946 în revista *Science* că toate masuratorile din știință utilizează patru tipuri diferite de scale de masurare:
  - Nominala
  - Ordinala
  - Interval
  - Proportionala (ratio scale)

# Nominala

---



- Valorile apartinand scalei nominale sunt caracterizate prin etichete (nume).
- Valorile sunt neordonate si au importanta (pondere) egala.
- Nu putem calcula valoarea medie sau mediana a unui astfel de set.
- Putem totusi calcula valoarea modala – valoarea cea mai frecventa.
- Datele nominale sunt de tip categoric si uneori este nevoie sa fie convertite in valori numerice.

# Ordinala

---



- Valorile de acest tip sunt ordonate dar diferenta / distanta intre doua valori nu poate fi calculata.
  - Putem specifica doar ordinea / pozitia in sirul ordonat pentru aceste valori.
  - Exemplu: lista gradelor militare, lista gradelor didactice, lista in ordinea sosirii a concurentilor la o cursa sportiva (doar lista numelor, fara timpii de sosire), etc.
  - Pentru astfel de valori putem calcula valoarea modală și mediana (valoarea mijlocie) dar nu media.
  - Valorile sunt în esență categorice dar este mai ușor să le asignăm valori numerice în caz de conversie.
-

# Interval

---



- Aceste valori sunt numerice.
  - In acest caz diferența între două valori are semnificativă.
  - Exemplu: temperatura în grade Celsius: diferența între 10 și 20 de grade este aceeași ca cea între 40 și 50 de grade (aceeași cantitate de energie în cazul încălzirii unui corp).
  - Valoarea zero (0) nu înseamnă 'nimic' ci este o valoare arbitrară aleasă. De aceea sunt permise și valori negative.
  - Putem calcula media, mediana, valoarea modală, deviația standard și putem utiliza regresia pentru a prognoza noi valori.
-

# Proportionala (ratio)

---



- Aceste valori sunt numerice, ca si in cazul Interval, dar zero (0) inseamna 'Nimic'.
- Valorile negative nu au sens.
- Raportul dintre doua valori are semnificatie.
- Exemplu: greutatea in kg. Un corp de 10 kg este de doua ori mai greu decat unul de 5 kg.
- Alte exemple: temperatura in grade Kelvin, lungimea in metri, varsta in ani, etc.
- Toate operatiile matematice sunt permise.

# Date binare

---



- Uneori un atribut poate avea doar 2 valori: 0 sau 1, masculin sau feminin, da sau nu, etc. În acest caz datele se numesc **binare**. Avem două cazuri:
  1. Binare simetrice: cele două valori au importanță/pondere egală (ca în cazul genului).
  2. Binare asimetrice: una dintre valori are o importanță mai mare decât cealaltă. De exemplu în cazul unei analize pentru depistarea HIV valoarea Pozitiv are o greutate mai mare decât Negativ.

# Date binare

---



- Atributele binare pot fi tratate uneori ca fiind de tip interval sau proportional dar in cea mai mare parte a cazurilor ele vor fi considerate nominale (cele binare simetrice) sau ordonale (cele binare asimetrice).
- Există o serie de functii de distanță (disimilaritate) care pot fi utilizate in acest caz.

# Preprocesarea datelor

---



- Tipuri de date
- Masurarea datelor
- Curatarea datelor
- Integrarea datelor
- Transformarea datelor
- Reducerea datelor
- Discretizarea datelor

# Masurarea datelor

---



- Masurarea tendinte/valorii centrale:
  - Medie
  - Mediana
  - Valoare modala
  - Mijlocul intervalului
- Masurarea dispersiei:
  - Interval de valori
  - Procentul k
  - IQR
  - Sumarul de 5 numere
  - Deviatia standard si varianta

# Tendinta centrala - Medie

---



- Fie  $n$  valori ale unui atribut:  $x_1, x_2, \dots, x_n$ .
- **Medie: Media aritmetica** sau valoarea medie este:  
$$\mu = (x_1 + x_2 + \dots + x_n) / n$$
- Daca valorile lui  $x$  au ponderile  $w_1, \dots, w_n$ , atunci **media aritmetica ponderata** este:  
$$\mu = (w_1 x_1 + w_2 x_2 + \dots + w_n x_n) / (w_1 + w_2 + \dots + w_n)$$
- In unele cazuri se practica eliminarea celor mai mici si celor mai mari valori (de exemplu cele mai mici 1% si cele mai mari 1%).

# Tendinta centrala - Mediana

---



- **Mediana:** Valoarea mediana a unui set de  $n$  valori ordonate este **valoarea din mijlocul secventei de valori.**
- Exemplu: Mediana pentru  $\{1, 3, 5, 7, 1001, 2002, 9999\}$  is 7.
- Daca  $n$  este par (si datele sunt numerice), mediana este media valorilor din mijlocul secventei:
  - mediana pentru  $\{1, 3, 5, 7, 1001, 2002\}$  este 6 (media valorilor 5 si 7).

# Tendinta centrala – Valoare modala

---



- **Valoarea modala:** Cea mai frecventa valoare din multimea respectiva.
- Un set de date poate avea mai multe valori modale. Pentru 1, 2 si 3 setul este denumit unimodal, bimodal sau trimodal.
- Cand fiecare valoare apare o singura data setul nu are valoare modala
- Pentru un set unimodal, valoarea modala masoara tendinta centrala a acestuia. In acest caz exista relatia empirica:

$$\text{medie} - v_{\text{mod}} = 3 \times (\text{medie} - \text{mediana})$$

---

# Tendinta centrala – Mijlocul intervalului

---



- **Mijlocul intervalului:** este media aritmetica a celei mai mari si celei mai mici valori.
- Exemplu: pentru {1, 3, 5, 7, 1001, 2002, 9999} valoarea de mijloc a intervalului este 5000 (media lui 1 si 9999).

# Dispersia valorilor (1)

---



- **Intervalul de valori.** Este diferența între cea mai mare și cea mai mică valoare.
- Exemplu: for {1, 3, 5, 7, 1001, 2002, 9999} valoarea este  $9999 - 1 = 9998$ .
- **Procentul k.** Aceasta este o valoare  $x_j$  apartinând setului de date și care are proprietatea că un procent de  $k$  valori din set sunt mai mici sau egale cu ea.
- Exemplu: Mediana este procentul 50.
- Cele mai utilizate procente sunt 25 și 75, numite și **cuartile** (eng. Quartiles). Notație: Q1 pentru 25% și Q3 pentru 75%.

# Dispersia valorilor (2)

---



- **Intervalul intre quartile** (eng. Interquartile range sau IQR) este diferența intre Q3 și Q1:

$$\text{IQR} = \text{Q3} - \text{Q1}$$

- Potențialele puncte izolate (outliers) sunt valori la distanța de cel puțin  $1.5 \times \text{IQR}$  sub Q1 sau peste Q3.
- **Sumarul de 5 numere**. Uneori mediana și quartilele nu sunt suficiente pentru a reprezenta dispersia valorilor. În acest caz adaugăm și valoarea minima și maxima din setul de date:
- (Min, Q1, Median, Q3, Max) este ceea ce se numește sumarul de 5 numere (eng. five-number summary).

# Dispersia valorilor (3)

---



## □ Exemple:

Pentru {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}

Interval de valori = 10; Mijloc interval = 6;

Q1 = 3; Q2 = 6; Q3 = 9; IQR = 9 - 3 = 6

Pentru {1, 3, 3, 4, 5, 6, 6, 7, 8, 8}

Interval de valori = 7; Mijloc interval = 5.5;

Q1 = 3; Q2 = 5.5 [=  $(5+6)/2$ ]; Q3 = 7; IQR = 7 - 3 = 4

Pentru {1, 3, 5, 7, 8, 10, 11, 13}

Interval de valori = 12; Mijloc interval = 7;

Q1 = 4; Q2 = 7.5; Q3 = 10.5; IQR = 10.5 - 4 = 6.6

# Dispersia valorilor (4)

---



- **Deviatia standard** (abaterea standard). Pentru n valori (observati) aceasta este:

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \mu)^2}, \text{ where } \mu = \frac{1}{N} \sum_{i=1}^N x_i.$$

- Patratul deviatiei standard se numeste **varianță** (dispersie)
- Deviatia standard masoara raspandirea valorilor in jurul valorii medii.
- Valoarea 0 este obtinuta doar daca toate valorile sunt egale.

# Preprocesarea datelor

---



- Tipuri de date
- Masurarea datelor
- Curatarea datelor
- Integrarea datelor
- Transformarea datelor
- Reducerea datelor
- Discretizarea datelor

# Curatarea datelor

---



- Obiectivele principale ale acestei etape sunt:
  - Înlocuirea (sau eliminarea) valorilor lipsă
  - Netezirea zgomotului
  - Eliminarea sau doar identificarea punctelor izolate (eng.: outliers - valori aberante)

# Valori lipsa

---



- Anumite atribute pot contine valori lipsa / valori nule (NULL).
- Pot sa apară din diverse motive:
  - Probleme hardware, software sau erori ale operatorului de introducere date
  - Date care nu au fost colectate într-o anumita perioadă (nu au fost considerate importante)
  - Valori eliminate pentru că erau inconsistente cu restul
- Există două soluții în acest caz:
  1. Ignorarea liniilor din tabela de date de intrare care contin valori lipsa. Asta se poate face doar în cazul în care numărul lor este mic și eliminarea nu induce erori în calculele ulterioare.

# Valori lipsa

---



2. Inlocuirea valorilor lipsa cu alte valori. Posibilitati:

- Completarea manuala a valorilor. Solutia nu este viabila in cazul seturilor de date mari
- Inlocuirea cu o valoare nenua de tip 'Necunoscut' sau 'Indisponibil'.
- Inlocuirea cu o valoare care masoara tendinta centrala a acelui atribut (medie, mediana, valoare modala).
- Inlocuirea cu o valoare care masoara tendinta centrala dar doar a unui subset (exemplile apartinand aceleiasi clase pentru seturi etichetate).
- Cea mai probabila valoare, calculata cumva (arbori de decizie, etc.)

# Netezire zgomot

---



- Zgomotul poate fi definit ca o eroare aleatoare sau variatie pentru o valoare masurata ([Han, Kamber 06])
- Pentru eliminarea zgomotului se pot folosi diverse tehnici de netezire ca:
  - Regresia (prezentata mai devreme)
  - Binning = Partitionarea in transe

# Binning

---



- Aceasta tehnica poate fi folosita pentru netezirea unui set ordonat de valori. Este bazata pe valorile din vecinatate.
- Sunt 2 pasi:
  1. Partitionarea setului ordonat in mai multe transe
  2. Netezirea fiecarei transe de date pe baza mediei, medianei sau capitelor intervalului.

# Exemplu

---



- Fie urmatorul set ordonat de valori: 1, 2, 4, 6, 9, 12, 16, 17, 18, 23, 34, 56, 78, 79, 81:

Transele initiale	Se utilizeaza media	Se utilizeaza mediana	Se utilizeaza capetele intervalului
1, 2, 4, 6, 9	4, 4, 4, 4, 4	4, 4, 4, 4, 4	1, 1, 1, 9, 9
12, 16, 17, 18, 23	17, 17, 17, 17, 17	17, 17, 17, 17, 17	12, 12, 12, 23, 23
34, 56, 78, 79, 81	66, 66, 66, 66, 66	78, 78, 78, 78, 78	34, 34, 81, 81, 81

# Rezultat

---



Rezultatul netezirii este:

- Initial: 1, 2, 4, 6, 9, 12, 16, 17, 18, 23, 34, 56, 78, 79, 81
- Cu medie: 4, 4, 4, 4, 4, 17, 17, 17, 17, 17, 66, 66, 66, 66
- Cu mediana: 4, 4, 4, 4, 4, 17, 17, 17, 17, 17, 78, 78, 78, 78
- Cu capetele intervalului: 1, 1, 1, 9, 9, 12, 12, 12, 23, 23, 34, 34, 81, 81, 81

# Puncte izolate

---



- Punctele izolate / valorile aberante (outliers) sunt valori numerice de atribut aflate la distanță mare de restul datelor.
- Uneori sunt valori corecte: salariul unui CEO poate fi mult mai mare decât al restului angajatilor.
- De cele mai multe ori însă sunt date eronate / zgromot.
- Trebuie identificate și eliminate sau înlocuite, deoarece multi algoritmi sunt sensibili la astfel de puncte – de exemplu k-means da clustere eronate în prezența lor.

# Identificare puncte izolate

---



- **Folosind IQR:** am mai spus ca potențiale puncte izolate (outliers) sunt valorile aflate la o distanță de cel puțin  $1.5 * \text{IQR}$  sub Q1 sau peste Q3.
- **Folosind deviația standard:** valorile care sunt la distanță mai mare de  $2\sigma$  de valoarea medie sunt potențiale valori izolate.
- **Folosind clustering:** După gasirea clusterelor, punctele aflate în afara oricărui cluster (sau foarte departate de orice centru de cluster) sunt potențiale puncte izolate.

# Preprocesarea datelor

---



- Tipuri de date
- Masurarea datelor
- Curatarea datelor
- Integrarea datelor
- Transformarea datelor
- Reducerea datelor
- Discretizarea datelor

# Integrarea datelor

---



- Dupa ce datele provenind din sursele de date individuale au fost curatare ca mai inainte, are loc integrarea acestora intr-un unic ansamblu de date.
- Activitatile din aceasta categorie includ:
  1. **Integrarea schemelor.** Problemele sunt de tip sinonime (acelasi lucru apare in doua surse cu nume diferite) si omonime (cu acelasi nume se gasesc date de tip diferit in diverse surse).
  2. **Duplicate si redundanta.** In surse diferite se regasesc aceleiasi informatii. Duplicatele si redundanta trebuie eliminate in aceasta faza.

# Integrarea datelor

---



## 3. Inconsistente.

- Acestea sunt valori aflate in conflict in acelasi set de date.
- De exemplu Data nasterii='1.1.1980' si Varsta=30 pentru o aceeasi persoana reprezinta o inconsistenta.
- Pentru detectarea acestora sunt necesare informatii suplimentare despre date.

# Preprocesarea datelor

---



- Tipuri de date
- Masurarea datelor
- Curatarea datelor
- Integrarea datelor
- Transformarea datelor
- Reducerea datelor
- Discretizarea datelor

# Transformarea datelor

---



- ❑ Aceasta etapa cuprinde procese de transformare si summarizare a datelor ca:
  - ❑ Normalizare
  - ❑ Constructie de noi atribute
  - ❑ Sumarizare folosind functiile statistice

# Normalizare

---



- Toate atributele numerice sunt scalate pentru a avea valori intr-un acelasi interval specificat:
  - De la 0 la 1,
  - De la -1 la 1 sau, mai general:
  - $|v| \leq r$  unde  $r$  este o valoare specificata.
- Normalizarea este necesara cand anumite atribut sunt mai importante decat altele doar pentru ca plaja lor de valori este mai intinsa.
- Exemplu: Distanța euclidiană intre A(0.5, 101) și B(0.01, 2111) este  $\approx 2010$ , determinată aproape exclusiv pe baza celei de-a doua coordonate.

# Normalizare



Putem folosi pentru normalizare:

**Normalizarea min-max:**

$$v_{\text{new}} = (v - v_{\min}) / (v_{\max} - v_{\min})$$

**Pentru valori pozitive** putem folosi formula:

$$v_{\text{new}} = v / v_{\max}$$

**Normalizarea de tip z-score** ( $\sigma$  este deviatia standard):

$$v_{\text{new}} = (v - v_{\text{mean}}) / \sigma$$

**Scalarea zecimala:**

$$v_{\text{new}} = v / 10^n$$

Unde  $n$  e cel mai mic intreg pentru care toate numerele devin in modul mai mic decat o valoare data  $r$  (de exemplu  $r=1$ ).

# Constructie noi atribute

---



- Este o tehnica prin care se adauga noi atribute (in engleza se numeste **feature construction**).
- Exemplu: daca setul de date are un atribut Culoare cu valorile {Rosu, Verde, Albastru} putem construi 3 noi atribute numite Rosu, Verde, Albastru continand doar valori binare (0 sau 1) cu 1 singur 1 pe atributul (coloana) corespunzatoare culorii curente.
- Alt exemplu: se poate utiliza un arbore de decizie sau set de reguli pentru a construi noi atribute din cele existente – ca de exemplu clasa.

# Sumarizare

---



- Se folosesc functii statistice (min, max, sum, avg, count, etc) pentru a adauga date sumare datelor originale.
- Exemplu: se adauga suma vanzarilor pe zi sau luna sau an, numarul mediu sau total de clienti sau tranzactii, etc.
- Aceste date vor fi folosite pentru cresterea vitezei de procesare a unor cereri in procesul de data mining.
- Rezultatul este un cub de date si fiecare data sumara este atasata unui nivel de granularitate.

# Preprocesarea datelor

---



- Tipuri de date
- Masurarea datelor
- Curatarea datelor
- Integrarea datelor
- Transformarea datelor
- Reducerea datelor
- Discretizarea datelor

# Reducerea datelor

---



- Nu toate informatiile colectate sunt necesare pentru anumite analize asupra datelor.
- Reducerea inseamna extragerea unei parti din datele colectate si preprocesate, si anume doar a acelei parti care contine informatiile necesare procesului de analiza respectiv.
- Se realizeaza de exemplu prin selectarea doar a anumitor atribute (coloane) sau exemple (linii) din matricea de date supusa analizei.

# Discretizare

---



- Anumiti algoritmi sunt conceputi pentru analiza datelor discrete.
- Pentru a ii folosi, este necesar ca valorile continue sa fie inlocuite cu unele discrete.
- De exemplu Varsta care are valori numerice va fi inlocuita cu un atribut categoric avand doar valorile Copil, Adult, Batran.
- Exista diverse metode de a efectua operatia de discretizare, folosind impartirea in transe (binning), histograme, intervale bazate pe entropie, clustering, ierarhii de concepte, etc.

# Sumar

---



In acest curs am prezentat:

- O lista de definitii alternative pentru Data Mining si cateva exemple pentru a intelege ce este Data Mining si ce nu este Data Mining.
- O discutie despre comunitatile implicate in Data Mining si despre faptul ca Data Mining este de fapt o reuniune heterogena de subdomenii.
- Pasii procesului de Data Mining process de la colectarea datelor aflate in locatii diferite (depozite de date, arhive sau sisteme operationale) pana la pasul final de evaluare
- O scurta descriere a subdomeniilor principale ale Data Mining cu exemple pentru fiecare dintre ele.
- O descriere cuprinzatoare a procesului de preprocesare a datelor.

Lectia viitoare: Multimi frecvente de articole si reguli de asociere

# Bibliografie si referinte

---



- ➡ [Liu 11] Bing Liu, 2011. Web Data Mining, Exploring Hyperlinks, Contents, and Usage Data, Second Edition, Springer, 1-13.
- ➡ [Tan, Steinbach, Kumar 06] Pang-Ning Tan, Michael Steinbach, Vipin Kumar, 2006. Introduction to Data Mining, Adisson-Wesley, 1-16.
- ➡ [Kimbal, Ross 02] Ralph Kimball, Margy Ross, 2002. The Data Warehouse Toolkit, Second Edition, John Wiley and Sons, 1-16, 396
- ➡ [Mikut, Reischl 11] Ralf Mikut and Markus Reischl, Data mining tools, 2011, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, Volume 1, Issue 5,  
<http://onlinelibrary.wiley.com/doi/10.1002/widm.24/pdf>
- ➡ [Ullman] Jeffrey Ullman, Data Mining Lecture Notes, 2003-2009, web page: <http://infolab.stanford.edu/~ullman/mining/mining.html>
- ➡ What is the difference between data mining, statistics, machine learning and AI? <http://stats.stackexchange.com/questions/5026/what-is-the-difference-between-data-mining-statistics-machine-learning-and-ai>

# Bibliografie si referinte

---



- ▶ Akash Mitra, Data Mining - a simple guide for beginners, 2012, <http://www.dwbiconcepts.com/data-warehousing/11-data-mining/97-data-mining-for-beginners.html>
- ▶ [Wikipedia] Wikipedia, the free encyclopedia, [en.wikipedia.org](http://en.wikipedia.org)
- ▶ [Han, Kamber 06] Jiawei Han, Micheline Kamber, Data Mining: Concepts and Techniques, Second Edition, Morgan Kaufmann Publishers, 2006, 47-101
- ▶ [Stevens 46] Stevens, S.S, On the Theory of Scales of Measurement. Science June 1946, 103 (2684): 677–680.
- ▶ [Wikipedia] Wikipedia, the free encyclopedia, [en.wikipedia.org](http://en.wikipedia.org)
- ▶ [Liu 11] Bing Liu, 2011. CS 583 Data mining and text mining course notes, <http://www.cs.uic.edu/~liub/teach/cs583-fall-11/cs583.html>

# Capitolul 8

Data mining:  
Reguli de asociere si multimi  
frecvente de articole  
(frequent itemsets)

# Problema

- Problema cosului de produse presupune ca avem un mare numar de articole, e.g. “paine”, “lapte”.
- Cumpăratorii pun în cosul lor de produse anumite submultimi de articole iar noi vom afla ce articole sunt cumpărate împreună chiar dacă nu și de cine.

# Problema

- Vanzatorii utilizeaza aceste informatii pentru asezarea articolelor in magazin, in felul acesta putand sa controleze modul in care anumite clase de cumparatori parcurg raioanele magazinului.

# Alte utilizari

- Cos = documente; articole = cuvinte.
- Cuvintele aparand frecvent impreuna in documente pot reprezenta fraze sau concepte legate intre ele.
- Poate fi utilizat pentru colectarea de informatii (intelligence).

# Alte utilizari

- Cos = propozitii, articole = documente.
- Doua documente continand aceleasi propozitii pot reprezenta un plagiat sau “mirror sites” pe web.

# Obiective pentru aceasta clasa de probleme

- Gasirea de:
  - Reguli de asociere
  - Cauzalitate
  - Multimi frecvente de articole

# Reguli de asociere

- *Regulile de asociere:* Daca X si Y sunt multimi de articole, o regula este o propozitie de forma  $X \rightarrow Y$  insemnand ca daca gasim toate articolele din X intr-un cos atunci sunt mari sanse sa gasim in acel cos si articolele din Y.
- Probabilitatea de a-l gasi pe Y pentru a accepta aceasta regula este numita *increderea* acelei reguli.
- In mod normal vom cauta doar reguli care au o incredere peste un anumit prag.

# Reguli de asociere

- Putem de asemenea cere ca increderea sa fie semnificativ mai mare decat in cazul in care articolele ar fi plasate aleator in cos.
- De exemplu putem gasi o regula ca  $\{lapte, unt\} \rightarrow paine$  doar pentru ca foarte multa lume cumpara paine.
- Totusi exemplul bere/scutece arata ca regula  $\{scutece\} \rightarrow \{bere\}$  este verificata cu o incredere semnificativ mai mare decat a submultimii de cosuri continand bere.

# Scutece și bere

- În 1992, Thomas Blischok, managerul unui grup de consultanță pentru vânzarea cu amănuntul de la Teradata, și personalul său au pregătit o analiză a 1,2 milioane de coșuri de produse de la aproximativ 25 de magazine Osco Drug.
- Au fost rulate interogări ale bazei de date pentru a identifica afinitățile.
- Analiza „a descoperit că între orele 17:00 și 19:00 consumatorii cumpărău bere și scutece”.
- Managerii Osco NU au exploatat asocierea dintre bere și scutece mutând produsele mai aproape unul de celălalt pe rafturi.
- Acest studiu pentru suportul deciziei a fost realizat folosind instrumente de interogare pentru a găsi associații.
- Povestea adevărată este foarte fadă în comparație cu legenda.

**Sursa: <http://www.dssresources.com/newsletters/66.php>**

# Cauzalitate

- *Cauzalitate.* Ideal, am vrea sa stim daca intr-o regula de asociere prezenta elementelor X efectiv “cauzeaza” (determina) cumpararea lui Y.
- “Cauzalitatea” este insa un concept echivoc.
- Exemplu:

# Cauzalitate

- Daca scadem pretul scutecelor si crestem pretul berii putem ademenii cumparatorii de scutece care au inclinatia de a cumpara bere din magazin, acoperind astfel pierderile din vanzarea scutecelor.
- Aceasta strategie este valabila deoarece “scutece determina bere”.

# Cauzalitate

- Actiunea inversa, micsorarea pretului la bere si marirea pretului scutelor nu va determina cumparatorii de bere sa cumpere scutece in numar mare si vom pierde bani deoarece regula “bere determina scutece” nu este adevarata.

# Multimi frecvente

- *Multimi frecvente de articole (frequent itemsets)*: in multe situatii (dar nu in toate) ne intereseaza doar regulile de asociere si cauzalitatea in ceea ce priveste multimi de articole care apar frecvent in cosuri.
- De exemplu, nu putem conduce o buna strategie de marketing care implica produse pe care oricum nu le cumpara nimeni.

# Multimi frecvente

- Astfel, cautarile in date pornesc de la premiza ca ne intereseaza doar multimile de articole cu un larg suport;
- Larg suport = ele apar impreuna in multe cosuri de produse.
- Gasim apoi reguli de asociere sau cauzalitate implicand doar articolele cu larg suport (i.e.  $\{X, Y\}$  trebuie sa apară în cel puțin un anumit procent din cosuri, numit *prag de suport*)

# Cadru de cautare

- Utilizam termenul *multimi frecvente de articole (frequent itemsets)* pentru “o multime de articole S care apare în cel puțin a ‘s’-a parte din cosuri”, unde s este o constantă aleasă, de obicei 0.01 sau 1%.
- Vom presupune că avem o cantitate de date care nu încape în memoria centrală a calculatorului.

# Cadru de cautare

- Stocare (pentru exemplele urmatoare):
  - Fie sunt stocate intr-o baza de date relationale (BDR), de exemplu o relatie (tabela) *Cosuri(IdCos, articol)*
  - Fie ca un fisier text cu linii de forma *(ICos, articol1, articol2, ..., articol-n)*.

# Cadru de cautare

- Cand evaluam timpul de rulare al algoritmilor parametrul optimizat este *numarul de treceri prin date*.
- Deoarece costul principal este dat adesea de timpul necesar citirii datelor de pe disc, numarul de citiri pentru fiecare data este adesea cea mai buna masura a timpului de rulare al algoritmului.

# Principiul a-priori

- Există un principiu cheie, numit *monotonicitate* (eng. monotonicity) sau *principiul a-priori* care ne ajuta să gasim multimele frecvente de articole:  
**“Dacă o multime de articole S este frecventă (i.e., apare cel puțin în a ‘s’-a parte a cosurilor), atunci orice submultime a lui S este de asemenea frecventă.”**

# Abordari

- Pentru a gasi multimile frecvente de articole avem doua abordari:
  1. Nivel cu nivel (aplicand principiul a-priori)
  2. Toate multimile de articole - de orice dimensiune - in cateva treceri (2-3 treceri)

# Nivel cu nivel

- Procedam nivel cu nivel, gasind intai articolele frecvente (multimi de dimensiune 1), apoi perechile frecvente, tripletele frecvente, etc.
- In multe cazuri partea cea mai grea este gasirea perechilor frecvente; continuarea pe nivelele superioare necesita mai putin timp decat gasirea acestora.

# Nivel cu nivel

- Algoritmii de acest tip utilizeaza o trecere per nivel.
- Există însă și abordări care nu sunt de tip nivel cu nivel:

# Toate multimile

- Gasim toate *multimile frecvente de articole maxime* (i.e., multimile S a.i. nici o multime care include strict pe S nu este frecventa) intr-o singura trecere sau in cateva treceri.
- Tehnicile de acest tip includ fie esantionarea datelor fie citirea lor in transe
- De obicei sunt suficiente 2 treceri prin date (o trecere pentru esantionare si inca una pentru verificarea finala)

# Algoritmul a-priori

Acest algoritm procedeaza nivel cu nivel.

1. Dandu-se pragul de suport  $s$ , in prima trecere gasim articolele care apar in cel putin a ' $s$ '-a parte a cosurilor. Aceasta multime este notata  $L_1$ , multimea articolelor frecvente.

# Algoritmul a-priori

2. Perechile de articole din  $L_1$  devin multimea  $C_2$  a *perechilor candidate* pentru a doua trecere. Speram ca dimensiunea lui  $C_2$  nu este atat de mare pentru ca altfel nu este suficient spatiu in memoria centrala pentru un contor numeric intreg al aparitiei fiecarei perechi. Perechile din  $C_2$  al caror contor ajunge sau depaseste pe  $s$  formeaza multimea  $L_2$  a perechilor frecvente.

# Algoritmul a-priori

3. Tripletele candidate  $C_3$  sunt multimile  $\{A, B, C\}$  pentru care  $\{A, B\}$ ,  $\{A, C\}$  si  $\{B, C\}$  sunt in  $L_2$ . In a treia trecere sunt numarate aparitiile tripletelor din  $C_3$ ; cele al caror contor este cel putin  $s$  formeaza multimea  $L_3$  a tripletelor frecvente.

# Algoritmul a-priori

4. Se poate merge oricat de departe se doreste (sau pana multimile devin vide).  $L_i$  contine multimile frecvente de articole de dimensiune  $i$ ,  $C_{i+1}$  este multimea candidatelor de dimensiune  $i+1$  a.i. fiecare submultime a lor de dimensiune  $i$  este inclusa in  $L_i$ .

# Algoritmul a-priori

- Oprirea algoritmului se face:
  1. In cazul in care nici una dintre multimile din  $C_{i+1}$  nu are un contor de aparitii mai mare decat pragul de suport.

Sau

2. Nu putem forma nici un element al multimii  $C_{i+1}$  din elementele lui  $L_i$ .

De exemplu daca  $L_2 = \{ (1, 2), (1, 3) \}$  nu se poate gasi nici un triplet  $(a, b, c)$  cu toate submultimile de 2 elemente in  $L_2$

# Efect a-priori

- Exemplu de cerere pentru gasirea de perechi frecvente:

```
SELECT b1.articol, b2.articol,  
       COUNT(*)  
  
FROM Cosuri b1, Cosuri b2  
  
WHERE b1.IdCos = b2.IdCos AND  
      b1.articol < b2.articol  
  
GROUP BY b1.articol, b2.articol  
  
HAVING count(*) >= s;
```

# Efect a-priori

- Sa consideram cererea SQL din slide-ul anterior cu ipotezele:
  - Utilizeaza tabela *Cosuri(IdCos, articol)*
  - avand  $10^8$  tupluri
  - care contin date despre  $10^7$  cosuri
  - de cate 10 articole fiecare.
  - Presupunem existenta a 100.000 articole diferite (tipic pentru o retea ca Wal-Mart de exemplu).

# Efect a-priori

- 's' este pragul de suport (nu in procente ci in valoare absoluta) iar al doilea termen al clauzei WHERE elimina perechile formate din acelasi produs si aparitia de doua ori a aceleiasi perechi.
- In joinul  $Cosuri \bowtie Cosuri$  fiecare cos contribuie cu  $C_{10}^2 = 45$  de perechi astfel incat joinul are  $4,5 \times 10^8$  tupluri (multe!).

# Efect a-priori

- A-priori “impinge clauza HAVING în jos pe arborele expresiei”, determinându-ne în primul rand să înlocuim *Cosuri* cu rezultatul ‘cererii’:

```
SELECT *  
FROM Cosuri  
GROUP BY articol  
HAVING COUNT(*) >= s ;
```

# Efect a-priori

- Cererea corecta care returneaza doar liniile continand articolele frecvente din cosuri este:

```
SELECT * FROM COSURI  
WHERE articol IN  
(SELECT articol // articole  
FROM Cosuri // frecvente  
GROUP BY articol  
HAVING COUNT(*) >= s) ;
```

# Efect a-priori

- Daca  $s = 0,01$  atunci cel mult 1000 de grupuri de articole pot trece de clauza HAVING.
- Motivul: sunt  $10^8$  linii continand articole in relatia Cosuri iar fiecare articol are nevoie de  $0,01 \times 10^7 = 10^5$  dintre acestea pentru a aparea in 1% din cosuri.
- Rezulta o diminuare a tabelei Cosuri si implicit a volumului de date pentru calculul joinului cu ea insasi.

# Dar ...

- Desi 99% dintre articole sunt eliminate de algoritmul a-priori nu trebuie sa concluzionam ca relatia *Cosuri* care rezulta are doar  $10^6$  tupluri.
- In fapt, *toate* tuplurile pot fi pentru produse cu larg suport.
- Totusi, in situatiile reale, micsorarea relatiei *Cosuri* este substantiala si dimensiunea joinului scade cu patratul acestei micsorari.

# Imbunatatiri pentru a-priori

De doua tipuri:

1. Micsorarea dimensiunii multimilor candidat  $C_i$  pentru  $i \geq 2$ .
- Aceasta optiune este importanta chiar si pentru gasirea perechilor frecvente deoarece numarul de elemente trebuie sa fie suficient de mic pentru ca un contor de aparitii pentru fiecare sa fie tinut in memoria centrala.
2. Contopirea incercarilor de gasire a  $L_1, L_2, L_3, \dots$  in doar una sau doua treceri in loc de o trecere per nivel.

# PCY

- Park, Chen si Yu au propus, utilizand o tabela de dispersie, sa determine la prima trecere (cand este calculat  $L_1$ ) ca multe perechi nu sunt frecvente – deci nu le mai numaram.
- Profita de faptul ca memoria centrala este uzual *mult* mai mare decat numarul de articole.
- In timpul celor doua faze pentru gasirea lui  $L_2$  memoria centrala este ocupata ca in figura urmatoare.
- Presupunem ca datele sunt stocate intr-un fisier cu inregistrari constand dintr-un identificator IdCos si o lista cu articolele sale.

# PCY

Trecerea 1

Trecerea 2

Contori de  
articole

Tabela de  
dispersie

Articole  
frecvente

Bitmap

Contori  
perechi  
candidat

# Trecerea 1

- Se numara aparitiile fiecarui articol
- Pentru fiecare cos constand din articolele  $\{i_1, \dots, i_k\}$ , se aplica functia de dispersie fiecarei perechi asociind-o unei intrari a tablelei de dispersie si se incrementeaza contorul acesteia cu 1.
- La sfarsitul trecerii, se determina  $L_1$ , articolele cu contorul cel putin  $s$ .

# Trecerea 1

- De asemenea la sfarsitul trecerii se determina acele intrari din tabela de dispersie care au contorul cel putin egal cu  $s$  (intrari frecvente).
- Punct cheie: o pereche  $(i, j)$  nu poate fi frecventa decat daca este dispersata intr-o intrare frecventa astfel incat perechile care sunt dispersate in alte intrari NU POT fi candidate in  $C_2$ .

# Trecerea 1

- Se inlocuieste tabela de dispersie cu un bitmap avand un bit per intrare: 1 daca intrarea a fost frecventa, 0 altfel.
- Acest bitmap va fi folosit la trecerea 2 prin date.

# Trecerea 2

- Memoria centrală conține o listă cu toate articolele frecvente, i.e.  $L_1$ .
- Tot memoria centrală conține un bitmap reprezentând rezultatele dispersiei din prima trecere.
- Punct cheie: intrările utilizează 16 sau 32 de biti pentru un contor dar sunt compțimate la un singur bit.

Contori de  
articole  
  
Tabela de  
dispersie

Articole  
frecvente  
  
Bitmap  
  
Contori  
perechi  
candidat

# Trecerea 2

- Astfel, chiar daca tabela de dispersie ocupa aproape intreaga memorie centrala la prima trecere, bitmapul nu ocupa mai mult de 1/16 din memoria centrala la trecerea 2.
- In final, memoria centrala contine de asemenea o tabela cu toate perechile candidat si contorii asociati lor.

# Trecerea 2

- O pereche  $(i, j)$  poate fi candidata in  $C_2$  doar daca *toate* conditiile urmatoare sunt adevarate:
  1.  $i$  este in  $L_1$ .
  2.  $j$  este in  $L_1$ .
  3.  $(i, j)$  este dispersata intr-o intrare frecventa (se utilizeaza bitmapul)
- Ultima conditie differentiaza PCY de a-priori clasic si reduce necesarul de memorie in trecerea 2.

Contori de  
articole  
  
Tabela de  
dispersie

Articole  
frecvente  
  
Bitmap  
  
Contori  
perechi  
candidat

# Trecerea 2

- În timpul trecerii 2 luam în considerare fiecare cos și fiecare pereche de articole din el, efectuând testul de mai sus.
- Dacă o pereche indeplinește toate cele trei condiții, se incrementează contorul acesteia din memorie sau se crează unul dacă acesta nu există încă.
- În final perechile numarate care au un contor de cel puțin **s** formează L2.

# Q & A

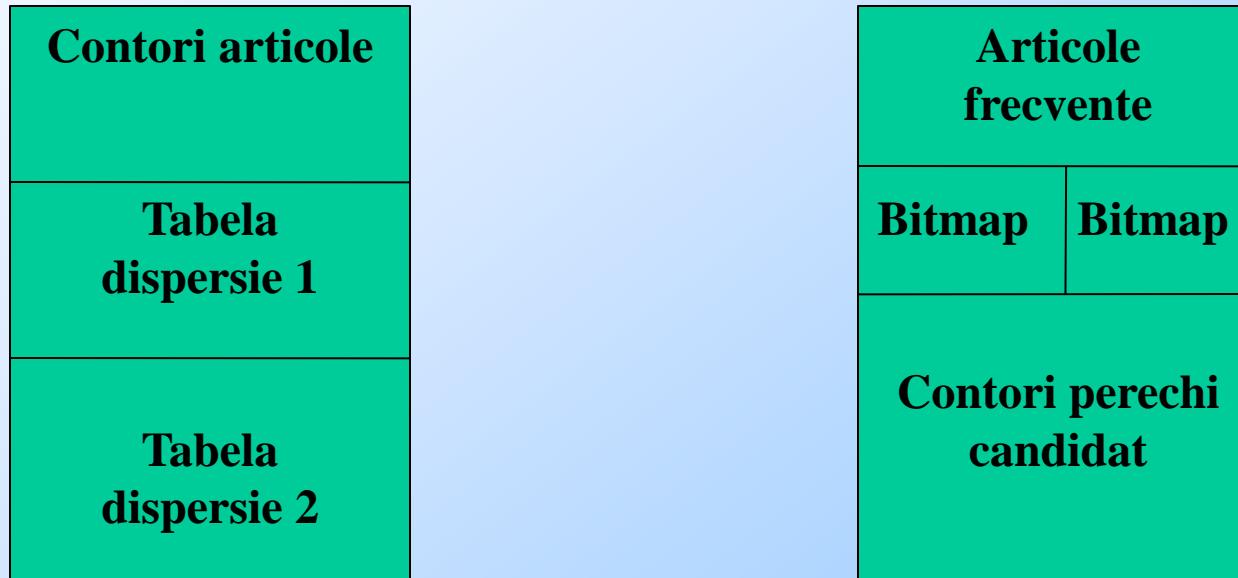
- Q: Cand este mai performant PCY decat a-priori?
- A: Cand sunt prea multe perechi de articole din  $L_1$  pentru a incapea intr-o tabela de perechi candidate si de contori asociati in memoria centrala iar numarul de intrari frecvente din algoritmul PCY este suficient de mic pentru a reduce dimensiunea lui  $C_2$  suficient pentru a incapea in memoria centrala (chiar si fara 1/16 din ea consumata de bitmap).

# Q & A

- Q: Cand o mare parte a intrarilor nu vor fi frecvente in PCY?
- A: Cand sunt putine perechi frecvente iar cea mai mare parte a perechilor sunt atat de putin frecvente incat chiar daca sumam contorii tuturor perechilor care sunt dispersate intr-o intrare data nu sunt mari sanse sa se obtina o valoare egala sau mai mare ca  $s$ .

# Tabele de dispersie multiple

- Varianta a PCY. Se folosesc mai multe functii de dispersie.
- Se imparte memoria intre doua sau mai multe tabele de dispersie, ca in figura urmatoare:



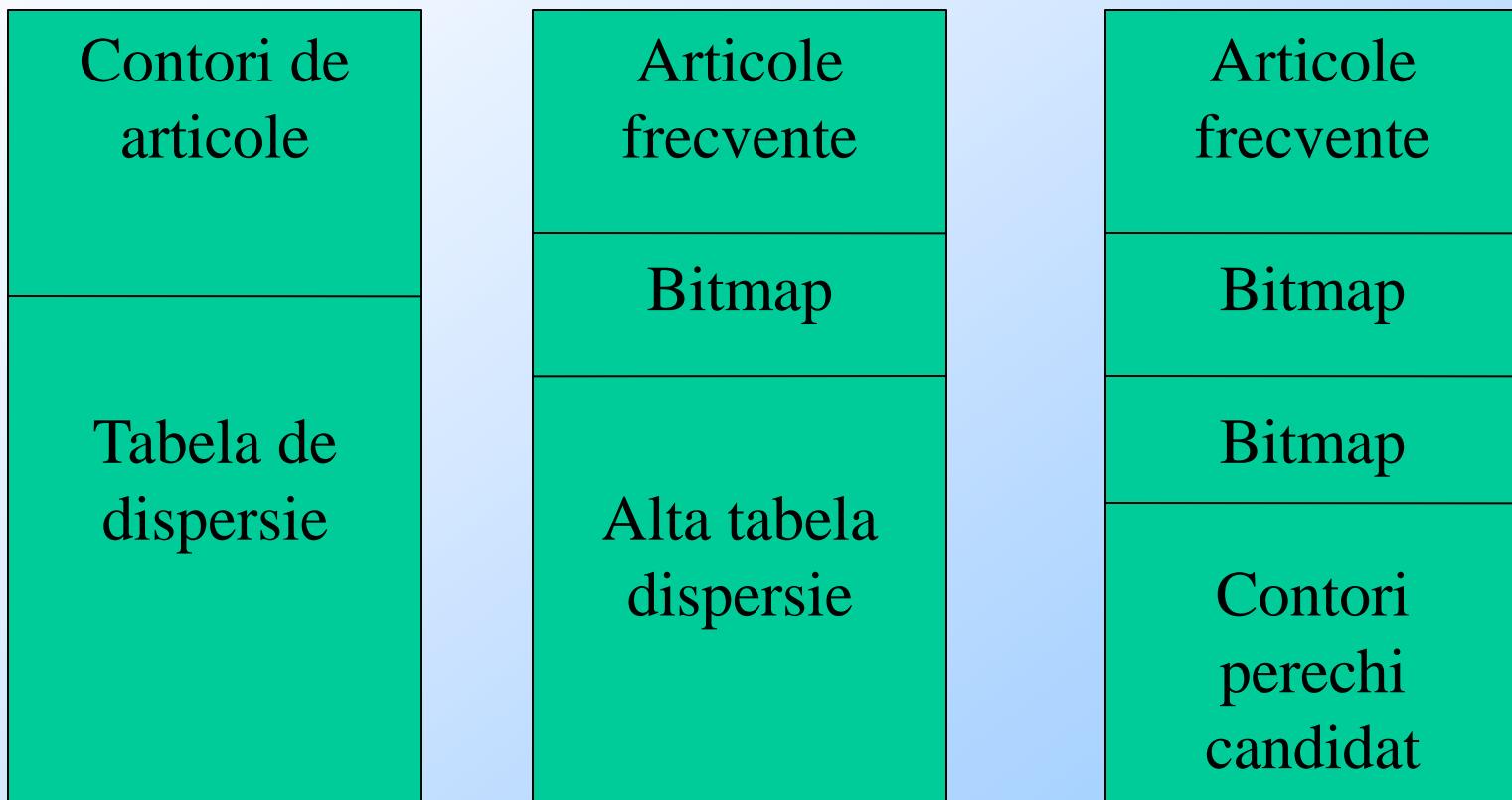
# Tabele de dispersie multiple

- La trecerea 2 se retine in memorie cate un bitmap pentru fiecare dintre acestea; de notat ca spatiul necesar pentru aceste bitmap este exact acelasi cu cel necesar in bitmapul unic din PCY deoarece numarul total de intrari reprezentate este acelasi.
- Pentru a fi candidata la  $C_2$  o pereche:
  1. Consta din articole din  $L_1$ , si
  2. Este dispersata intr-o intrare frecventa in *fiecare* tabela de dispersie.

# Tabele de dispersie iterate

- Tabele de dispersie iterate *Multistage*:
- Ca la PCY dar in locul verificarii candidatelor in trecerea 2 se creaza o alta tabela de dispersie (alta functie de dispersie) si sunt dispersate doar acele perechi care indeplinesc conditiile pentru PCY; i.e., ambele articole sunt din  $L_1$  si sunt dispersate intr-un intrare frecventa in prima trecere.

# Tabele de dispersie iterate



# Tabele de dispersie iterate

- Intr-a treia trecere, pastram cate un bitmap pentru fiecare tabela de dispersie si tratam o pereche ca fiind candidata (in  $C_2$ ) doar daca:
  - Ambele articole sunt in  $L_1$ .
  - Perechea a fost dispersata intr-o intrare frecventa in prima trecere.
  - Perechea a fost dispersata de asemenea intr-o intrare frecventa la trecerea 2.

# Q & A

- Q: Cand sunt utile tabelele de dispersie multiple?
- A: Cand cele mai multe dintre intrari de la prima trecere a PCY au contori cu mult sub pragul **s**. Atunci putem dubla contorii intrarilor si totusi cele mai multe vor fi sub prag.

# Q & A

- Q: Cand sunt utile tabelele de dispersie iterate?
- A: Cand numarul intrarilor frecvente din prima trecere este mare (e.g. 50%) - dar nu toate. Atunci, a doua dispersie cu unele dintre perechile ignorante poate reduce semnificativ numarul de intrari.

# Toate multimile in 2 treceri

- Metodele de mai sus sunt cele mai bune cand dorim perechile frecvente, cazul cel mai comun.
- Daca dorim toate multimile frecvente maximale de articole, inclusiv multim mari, sunt necesari prea multi pasi.
- Exista mai multe abordari pentru obtinerea tuturor multimilor frecvente de articole in doua treceri sau mai putin.

# Abordarea simplă

- *Abordarea simplă*: Se ia un esantion de date de dimensiunea memoriei centrale.
- Se ruleaza un algoritm nivel cu nivel in memoria centrala (deci nu sunt costuri de I/O) si
- Se spera ca esantionul ne va conduce la adevaratele multimi frecvente.

# Abordarea simplă

- De notat ca pragul  $s$  trebuie scalat prin micsorare; e.g. daca esantionul este 1% din date, se utilizeaza  $s/100$  ca prag de suport (in valoare absoluta).
- Se poate face o trecere completa prin date pentru a verifica daca multimile frecvente de articole ale esantionului sunt cu adevarat frecvente,

# Abordarea simplă

- Vor fi pierdute insa multimile de articole care sunt frecvente in ansamblul datelor dar nu in esantion.
- Pentru a minimiza falsele negative se poate scadea putin pragul pentru esantion gasindu-se mai multe candidate pentru trecerea prin ansamblul datelor.
- Risc: prea multe candidate pentru a incapea in memoria centrala.

# SON95

- *SON95* (Savasere, Omiecinski and Navathe in VLDB 1995; referit de Toivonen).
- Se citesc submultimi (transe) ale datelor în memoria centrală aplicându-se abordarea simplă pentru descoperirea multimilor candidat.
- Fiecare cos este parte a uneia dintre aceste submultimi.

# SON95

- In a doua trecere o multime este candidata daca a fost identificata ca si candidata in una sau mai multe submultimi ale datelor.
- Punct cheie: O multime nu poate fi frecventa in ansamblul datelor daca nu este frecventa in cel putin o submultime a acestora (oare de ce?).

# Toivonen

- Se ia un esantion care incape in memoria centrala.
- Se ruleaza abordarea simpla pe aceste date dar cu un prag micsorat astfel incat sa fie improbabila pierderea vreunei adevarate multimi frecvente de articole (e.g. daca esantionul este de 1% din date se foloseste s/125 ca prag de suport).

# Toivonen

- Se adauga candidatelor din esantion *marginea negativa*: acele multimi de articole  $S$  astfel incat  $S$  nu este identificata ca frecventa in esantion dar *orice* submultime stricta maxima a lui  $S$  este identificata astfel.
- De exemplu, daca  $ABCD$  nu este frecventa in esantion dar  $ABC$ ,  $ABD$ ,  $ACD$  si  $BCD$  sunt frecvente in esantion, atunci  $ABCD$  este in marginea negativa.

# Toivonen

- Se face o trecere prin date, numarand toate multimile frecvente de articole si marginea negativa.
- Daca nici o multime din marginea negative nu este frecventa in ansamblul datelor, atunci multimile frecvente de articole sunt exact acele candidate care sunt deasupra pragului.

# Bibliografie

- J.D.Ullman - CS345 --- Lecture Notes, primele doua capitole (Overview of Data Mining, Association-Rules, A-Priori Algorithm)

<http://infolab.stanford.edu/~ullman/cs345-notes.html>

# Sfârșitul capitolului 8

# Capitolul 9

## Data mining – date corelate

# Reprezentarea datelor

- Vom continua să considerăm modelul de date “coșuri de produse” și vom vizualiza datele ca o matrice booleană unde:
  - linii=coșuri și
  - coloane=articole.

# Asertiuni

1. Matricea este foarte rară; aproape peste tot 0.
  2. Numărul de coloane (articole) este suficient de mic pentru a putea stoca în memoria centrală ceva per coloană dar suficient de mare astfel încât nu putem stoca ceva per pereche de coloane în memoria centrală (aceeași asertiu-
- ne pe care am făcut-o până acum privind regulile de asociere).

# Asertiuni

3. Numărul de linii este atât de mare încât nu putem stoca întreaga matrice în memorie chiar profitând de faptul ca e rară și comprimând-o (din nou aceeași aserțiune ca întotdeauna).
4. Nu suntem interesați de perechile sau multimile de coloane cu larg suport; în schimb dorim perechile de coloane puternic corelate (similar).

# Aplicații

- Aplicațiile de marketing sunt interesate doar de produsele de larg consum (nu merită să încercam promovarea obiectelor pe care oricum nu le cumpără mai nimeni).
- Există însă un număr de aplicații care se potrivesc cu modelul de mai sus, de interes fiind în special problema ***perechilor*** de coloane / articole inclusiv de consum restrâns dar puternic corelate:

# Aplicatii

1. Liniile și coloanele sunt pagini de web:

$(r, c) = 1$  înseamna că pagina corespunzatoare liniei  $r$  conține o legătură către pagina coloanei  $c$ .

- Coloanele similare pot fi pagini despre același domeniu.
- Obs: Coloane similare = au 1 în cam aceleasi pozitii.

# Aplicatii

2. Liniile și coloanele sunt pagini de web:

$(r, c) = 1$  înseamna că pagina corespunzatoare coloanei  $c$  conține legături către pagina liniei  $r$ .

- Acum, coloane similare pot reprezenta copii multiple (mirror) ale unei pagini.

# Aplicatii

3. Linii = pagini web sau documente; coloane = cuvinte.
- Coloane similare reprezintă cuvinte care apar aproape mereu împreuna, e.g. "fraze".

# Aplicatii

4. liniile sunt propoziții iar coloanele sunt pagini web.
- Coloane similare pot indica copii multiple ale unei unei pagini sau plagiat.

# Similaritate

- ❑ Am vorbit despre similaritatea coloanelor fără să dam o masură cantitativă a acesteia.
- ❑ **Definīție:** Să ne gândim la o coloană ca la multimea liniilor pentru care coloana conține 1.  
Atunci ***similaritatea*** a două coloane C1 și C2 este  
$$\text{Sim}(C1, C2) = |C1 \cap C2| / |C1 \cup C2|.$$

# Similaritate

Unde:

- $|x|$  reprezinta cardinalul multimii  $x$ , deci:
- $|C_1 \cap C_2|$  reprezinta numarul de linii in care ambele coloane au valoarea 1
- $|C_1 \cup C_2|$  reprezinta numarul de linii in care macar una dintre coloane are valoarea 1.

# Exemplu

0 1

1 0

1 1 = 2/5 = 40% similare

0 0

1 1

0 1

# Problema

- ❑ Deoarece matricea contine foarte multe linii si coloane, testarea similaritatii este laborioasa (matricea nu incape in memoria centrala).
- ❑ Ar fi preferabil ca fiecare coloana sa fie reprezentata de o cantitate mult mai mica de informatie avand proprietatea ca testul de similaritate efectuat pe aceasta sa fie relevant pentru similaritatea coloanelor.

# Signatura

- Ideia principală: Se mapează ("dispersează") fiecare coloană  $C$  într-o cantitate mică de date numita ***signatura lui C***, (*notatie  $Sig(C)$* ) astfel încât:
  - $Sig(C)$  este suficient de mică pentru ca signaturile tuturor coloanelor să încapă în memoria centrală și să se poată efectua testul de similaritate.

# Signatura

- ❑ Cand un mod de calcul pentru signaturi este bun?
- Cand coloanele  $C1$  și  $C2$  sunt puternic similare dacă și numai dacă  $Sig(C1)$  și  $Sig(C2)$  sunt puternic similare (dar de notat că este nevoie să definim “similaritatea” pentru signaturi).

# Exemplu (prost)

- ❑ O idee - care însă nu funcționează:
  - Se iau aleator 100 de linii și sirul de 100 de biti ai coloanelor pentru acele linii este signatura fiecărei coloane.
- ❑ De ce nu functioneaza?

# Exemplu (prost)

□ Motivul pentru care ideea nu functioneaza este că matricea este presupusă ca fiind **foarte rară** deci multe coloane vor avea signaturi identice formate doar din 0 chiar dacă ele nu sunt deloc similare.

# Conventie utila

□ Dându-se două coloane  $C1$  și  $C2$ , ne vom referi la liniile lor ca fiind de patru tipuri –  $a, b, c, d$  – în funcție de biții lor pe aceste coloane, după cum urmează:

Tip	C1	C2
a	1	1
b	1	0
c	0	1
d	0	0

# Conventie utila

- De asemenea vom utiliza  $a$  pentru “numărul de linii de tip  $a'$ ”, și.a.m.d.
- De notat că  $\text{Sim}(C1, C2) = a/(a+b+c)$ .
- Dar cum cele mai multe linii sunt de tip  $d$ , într-o selecție de, să spunem, 100 de linii alese aleator toate vor fi de tip  $d$ , deci similaritatea coloanelor doar pentru aceste 100 de linii nici nu este definită.

Ce vom prezenta in continuare:

- a. *Dispersia de tip Min* si *dispersia de tip k-min***
  - metode de calcul signaturi
- b. *Dispersia sensitiva la localizare (DSL)*** – o metoda prin care se minimizeaza numarul de perechi de signaturi care se testeaza pentru similaritate

# Dispersia de tip Min (Minhashing)

- ❑ Este o metoda de calcul signaturi
- ❑ Signatura unei coloane va fi un sir de numere intregi.
- ❑ Să ne imaginăm liniile permute într-o ordine aleatoare. “Dispersăm” fiecare coloana  $C$  în  $h(C)$ , numărul primei linii în care coloana  $C$  are un 1.
- ❑ În felul acesta obținem primul intreg al signaturii

# Dispersia de tip Min (Minhashing)

- Probabilitatea ca  $h(C1) = h(C2)$  este  $a/(a+b+c)$  deoarece valoarea de dispersie este aceeasi dacă prima linie cu un 1 în vreuna din coloane este de tip  $a$  și este diferită dacă prima astfel de linie este de tip  $b$  sau  $c$ .
- De notat că această probabilitate este aceeași cu  $Sim(C1, C2)$ .

# Dispersia de tip Min (Minhashing)

- Dacă repetăm experimentul cu o noua permutare a liniilor de un număr mare de ori, să zicem 100, obținem o signură constând din 100 de numere de linii pentru fiecare coloană.
- “Similaritatea” acestor liste (fracțiunea a pozițiilor în care ele sunt egale) va fi foarte apropiată de similaritatea coloanelor.

# Dispersia de tip Min (Minhashing)

- ❑ Observație importantă: nu trebuie să permutam fizic liniile, ceea ce ar duce la multe treceri prin întreaga cantitate de date.
- ❑ În schimb citim liniile într-o ordine oarecare și dispersăm fiecare linie (numărul acesteia) utilizând (să zicem) 100 de funcții de dispersie diferite.

# Dispersia de tip Min (Minhashing)

- ❑ Pentru fiecare coloană memorăm cea mai mică valoare a funcției de dispersie a unei linii în care acea coloană are un 1, independent pentru fiecare dintre cele 100 de funcții de dispersie.
- ❑ După parcurgerea tuturor liniilor vom avea pentru fiecare coloană primele linii în care coloana are 1 dacă liniile ar fi fost permute în ordinea data de fiecare dintre cele 100 de functii de dispersie.

# Exemplu

Functii

1	4	3
3	2	4
7	1	7
6	3	6
2	6	1
5	7	2
4	5	5

h3 h2 h1

Tabela

1	0	1	0
1	0	0	1
0	1	0	1
0	1	0	1
0	1	0	1
1	0	1	0
1	0	1	0

Signaturi

2	1	2	1
2	1	4	1
1	2	1	2

# Exemplu

## Similaritati

	1-3	2-4	1-2	3-4
Col-Col	0,75	0,75	0	0
Sig-Sig	0,67	1,00	0	0

## Signaturi

2	1	2	1
2	1	4	1
1	2	1	2

Deci signaturi similare => Coloane similare.

Aici diferențele sunt mai mari datorita dimensiunii tableei si signaturii (prea mica)

# Dispersia senzitiva la localizare

- ❑(eng: Locality-Sensitive Hashing - LSH)
- ❑Problema: avem signaturile fiecarei coloane în memoria centrală iar signaturi similare înseamnă cu mare probabilitate coloane similare,
- ❑Pot fi totuși atât de multe coloane încât a face ceva care este proporțional cu pătratul numărului de coloane, chiar și în memoria centrală, este prohibitiv.

# Dispersia senzitivă la localizare

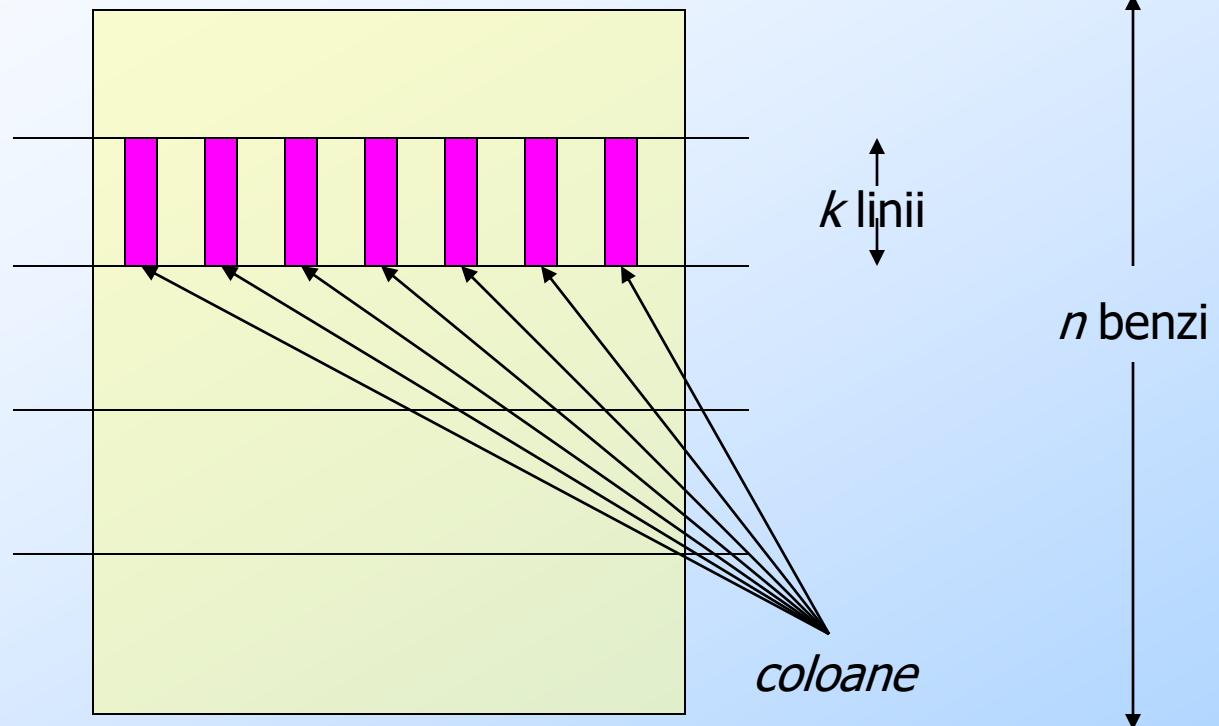
- Dispersia senzitivă la localizare (DSL, LSH în engleză) este o tehnică destinată a fi utilizată în memoria centrală pentru a aproxima mulțimea de perechi de coloane similare cu o complexitate mult mai mică decât cea pătratică.
- Scopul: în timp proporțional cu numărul de coloane să se eliminate cea mai mare parte a perechilor de coloane din mulțimea posibilelor perechi similare.

<https://towardsdatascience.com/understanding-locality-sensitive-hashing-49f6d1f6134>

# Etape

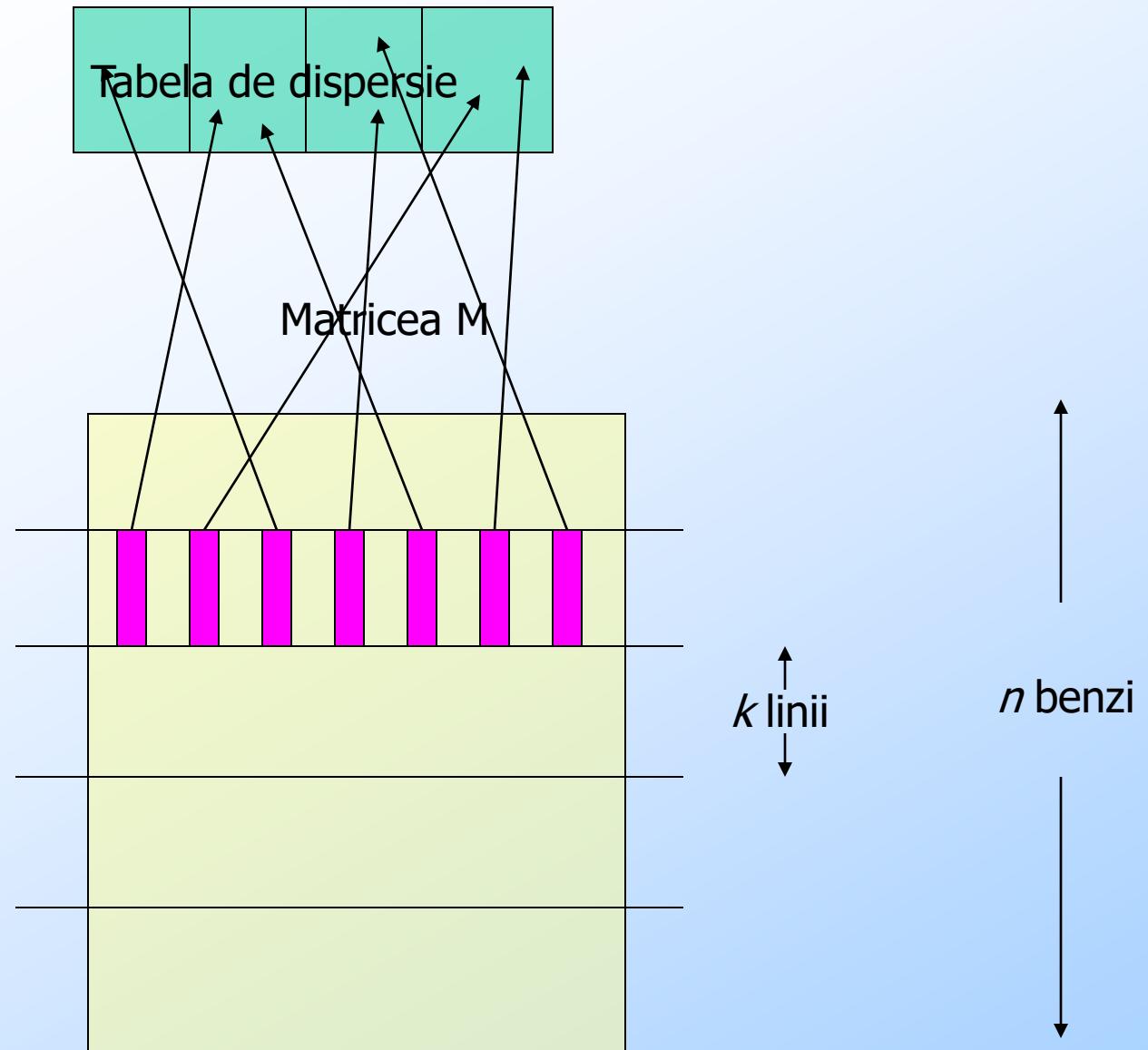
- Este deci o metoda de a micsora numarul de perechi de signaturi care se compara pentru similaritate.
- Etapele sunt:
  1. Considerăm signatura ca fiind o coloană de întregi.
  2. Partiționăm liniile signaturilor în *benzi*, să spunem  $n$  benzi de câte  $k$  linii fiecare.

Matricea M



# Etape

3. Dispersăm coloanele din fiecare banda în intrări ale unei tabele de dispersie. O pereche de coloane este o pereche-candidat dacă ambele sunt dispersate în aceeași intrare în vreo bandă.
4. Dupa identificarea candidatelor se verifică fiecare pereche candidat ( $C_i, C_j$ ) examinând pentru similaritate  $Sig(C_i)$  și  $Sig(C_j)$ .



# Exemplu

- Pentru a vedea efectul DSL să considerăm o matrice M cu 100.000 de coloane și signaturi constând din 100 de întregi fiecare.
- Signaturile ocupă 40Mb de memorie, nu atât de mult la standardele actuale.
- Să presupunem că vrem perechile care sunt 80% similare.
- Vom examina signaturile în loc de coloane, deci în mod real vom identifica coloanele ale caror *signaturi* sunt 80% similare – deci nu chiar același lucru.

# 80%

- Dacă două coloane sunt 80% similare atunci probabilitatea ca ele să fie identice în una dintre benzile de 5 întregi este  $(0,8)^5 = 0,328$ .
- Probabilitatea ca ele să nu fie identice în *nici una* dintre cele 20 de benzi este  $(1-0,328)^{20} = 0,00035$ .
- Astfel toate mai puțin aproximativ 1/3000 dintre perechile cu signaturi 80% similare vor fi identificate ca și candidate.

40%

- Acum, să presupunem că doua coloane sunt doar 40% similare.
- Atunci probabilitatea ca ele să fie identice într-o bandă este  $(0,4)^5 = 0,01$
- Probabilitatea ca ele să fie identice în cel puțin una dintre cele 20 de benzi nu este mai mare ca  $0,2 (<= 20 * 0,01)$
- Astfel, putem ignora cel putin  $4/5$  dintre perechi care nu vor deveni candidate dacă 40% este similaritatea tipică a coloanelor.

# Concluzie

❑ În fapt, cele mai multe perechi vor fi cu mult mai puțin decât 40% similare astfel încât realmente eliminăm o parte importantă a perechilor de coloane care nu sunt similare.

# Dispersia k-Min

- ❑ Dispersia Min ne cere să dispersăm fiecare număr de linie de  $k$  ori dacă vrem signaturi de  $k$  întregi.
- ❑ În loc de asta, în cazul *dispersiei k-min* dispersăm fiecare linie o singură dată și, pentru fiecare coloană luăm ca signatură numerele primelor  $k$  linii în care acea coloană are un 1.

# Dispersia k-Min

- Pentru a vedea de ce similaritatea acestor signaturi este aproape aceeași cu similaritatea coloanelor din care derivă să examinam figura:

Sig1	Sig2
1	0
1	1
0	1
.. .	.. .
0	1
1	0
1	1

# Dispersia k-Min

- În figura sunt signaturile  $Sig1$  și  $Sig2$  pentru coloanele  $C1$  și respectiv  $C2$ .
- S-a presupus ca liniile au fost permute în ordinea valorilor funcției de dispersie.
- Liniile de tip  $d$  (în care nici o coloana nu are 1) sunt omise.
- Astfel, vedem doar liniile de tip  $a$ ,  $b$  și  $c$  și indicăm că o linie este în signatura printr-un 1.

# Dispersia k-Min

□ Să presupunem că  $c \geq b$  astfel încât situația tipică (pentru  $k = 100$ ) este cea din figura: cele 100 de linii pentru prima coloană includ unele linii care nu sunt printre cele 100 ale celei de-a doua coloane.

# Dispersia k-Min

- ❑ Atunci o estimare a similarității lui *Sig1* și *Sig2* poate fi calculată astfel:

$$| \text{Sig1} \cap \text{Sig2} | = 100 * a / (a+c)$$

- ❑ Justificare: În medie, proporția (data de probabilitate) primelor 100 de linii din C2 care sunt și în C1 este  $a/(a+c)$ .

# Dispersia k-Min

□ De asemenea:

$$| \text{Sig1} \cup \text{Sig2} | = 100 + 100*c / (a + c)$$

□ Motivul este că toate cele 100 de linii din *Sig1* sunt în reuniune.

□ În plus, liniile din *Sig2* care nu sunt în *Sig1* sunt de asemenea în reuniune, iar acestea sunt în medie în număr de  $100c/(a+c)$ .

# Dispersia k-Min

- Astfel, similaritatea lui  $Sig_1$  cu  $Sig_2$  este:

$$|\ Sig_1 \cap Sig_2 | / |\ Sig_1 \cup Sig_2 | = \\ a / (a + 2c)$$

- Observăm că dacă  $c$  este apropiat ca valoare de  $b$  atunci similaritatea signurilor este apropiată de similaritatea coloanelor.
- În fapt, dacă două coloane sunt foarte similare, atunci  $b$  și  $c$  sunt ambele mici comparate cu  $a$  și similaritățile signurilor și coloanelor *trebuie* să fie apropiate.

# DSL Hamming

- ❑ În cazul în care coloanele nu sunt rare ci au aprox. 50% de 1 nu avem nevoie de dispersie Min;
- ❑ O colecție aleatoare de linii servește în acest caz ca signatură.
- ❑ *DSL Hamming* construiește o serie de matrici, fiecare având jumătate din numărul de linii ale precedentei, aplicând operatorul SAU (OR) la câte două linii succesive din matricea precedentă

# DSL Hamming

- ❑ Nu există mai mult de  $\log_2 n$  matrici, unde  $n$  este numărul de linii. Numărul total de linii în toate matricile este  $2n$  și pot fi calculate toate într-o singură trecere prin matricea originală, stocându-le pe cele mari pe disc.

# DSL Hamming

- ❑ În fiecare matrice, se aleg ca perechi candidat acele coloane care:
  - ❑ Au o densitate de 1 să zicem intre 20% și 80%
  - ❑ E posibil să fie similare bazat pe testul DSL

# DSL Hamming

□ Observam ca plaja de densitate 20% - 80% ne garantează că două coloane care sunt cel puțin 50% similare vor fi considerate împreună în cel puțin o matrice, în afara cazului nefericit în care densitatea lor relativă se schimbă din cauza operației OR care combină doi de 1 într-unul singur.

# DSL Hamming

- ❑ O a doua trecere prin datele originale confirmă care dintre candidate sunt întradevăr similare.
- ❑ Aceasta metodă exploatează o idee care poate fi folositoare și în alte parti: coloanele similare au un număr similar de 1, deci nu are rost compararea coloanelor al căror număr de 1 este foarte diferit

# Bibliografie

- J.D.Ullman - CS345 --- Lecture Notes, Capitolul 3  
(Overview of Data Mining, Association-Rules, A-Priori Algorithm)

<http://infolab.stanford.edu/~ullman/cs345-notes.html>

# Sfârșitul capitolului 9

# Capitolul 10

# Algoritmi de clasificare

# Cuprins

- ❑ Ce este învățarea supervizată
- ❑ Evaluare clasificatori
- ❑ Arbori de decizie: ID3 și C4.5
- ❑ Clasificatori bayesieni (Naïve Bayes)
- ❑ Mașini cu vectori suport (Support vector machines)
- ❑ K-nearest neighbors
- ❑ Metode asambliste: Bagging, Boosting, Random Forest

# Obiective

- ❑ Învățarea supervizată este un subdomeniu foarte studiat din Data Mining
- ❑ În acest caz se construiesc noi modele din experiențe trecute (date)

# Definiții

- Învățarea supervizată include:
  - Clasificare: rezultatele sunt valori discrete (obiectiv: identificarea apartenenței la un grup / clasă).
  - Regresie: rezultatele sunt valori continue sau ordonate (obiectiv: estimarea sau prezicerea unui răspuns).
- În acest capitol ne concentrăm pe clasificare

# Clasificarea

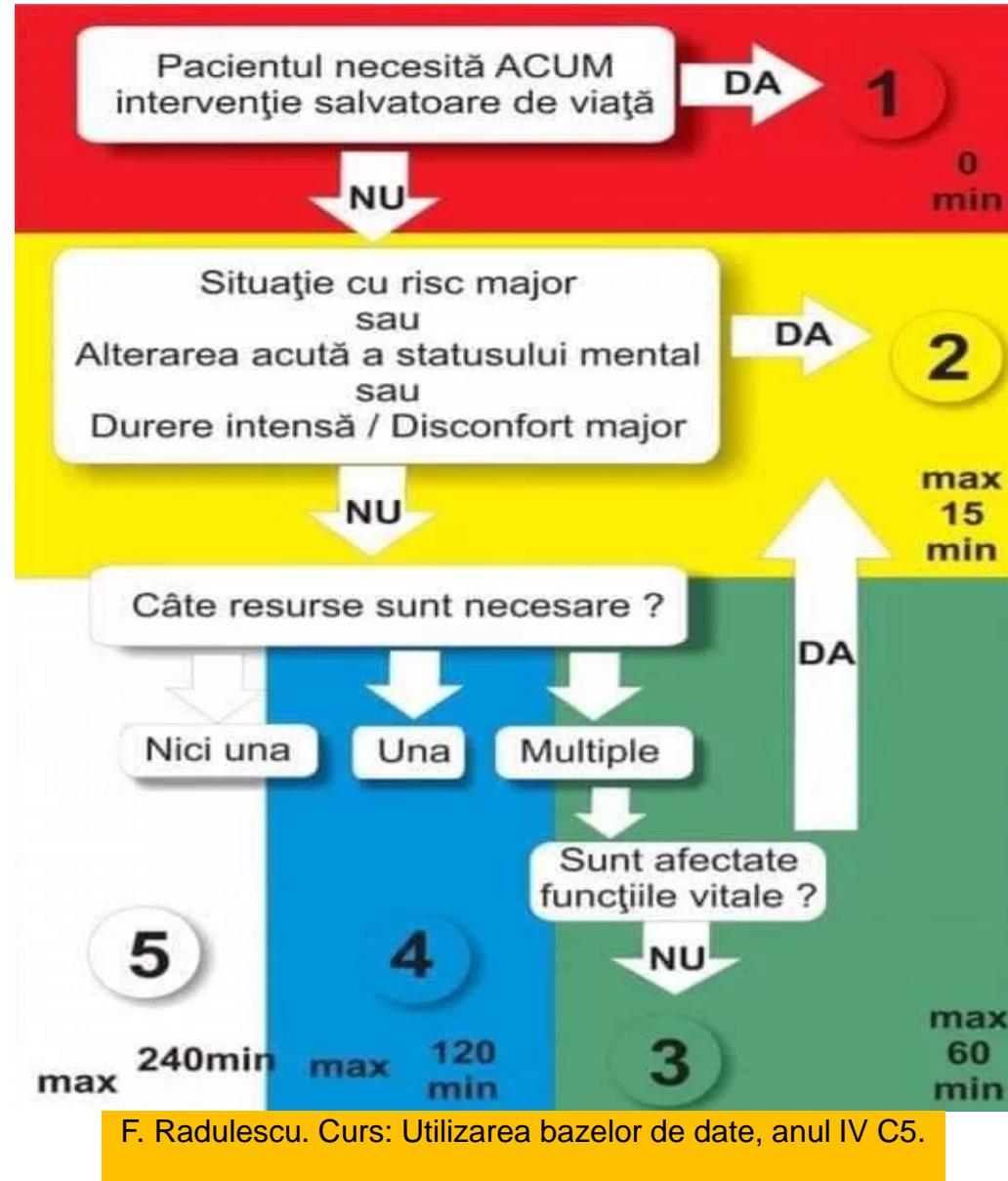
## Intrare:

- Un set de clase  $C = \{c_1, c_2, \dots, c_k\}$  în număr de  $k$ .
- Un set de  $n$  articole etichetate  $D = \{(d_1, c_{i1}), (d_2, c_{i2}), \dots, (d_n, c_{in})\}$ . Articolele sunt  $d_1, \dots, d_n$ , fiecare articol  $d_j$  fiind etichetat cu o clasa  $c_j$  din  $C$ .  $D$  se numește **multime/set de antrenare**.
- Pentru calibrarea/evaluarea unor algoritmi, este necesara o mulțime de validare. Aceasta conține de asemenea elemente etichetate care nu sunt incluse în setul de antrenare.

## Ieșire:

- Un model sau o metodă pentru clasificarea articolelor noi. Mulțimea de articole noi care vor fi clasificate folosind modelul / metoda obținută se numește **multime/set de test**.

# Exemplu. Model: arbore de decizie



# Formatul datelor de intrare

- ❑ În majoritatea cazurilor, setul de antrenament (precum și setul de validare și setul de testare) pot fi reprezentate ca un tabel având o coloană pentru fiecare atribut al lui X, iar ultima coloană conține eticheta clasei.
- ❑ Un exemplu foarte folosit este Play-tenis în care sunt folosite condițiile meteorologice pentru a decide dacă jucătorii pot sau nu începe un nou joc.
- ❑ Acest set de date va fi utilizat și în acest curs.

# Setul Play tennis

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

# Arborei de decizie

- Arborei de decizie: un exemplu este arborele de decizie UPU aşa cum este descris în exemplul precedent.
- Într-un arbore de decizie nodurile interne conțin decizii bazate pe valorile atributelor exemplelor (attribute ale argumentului  $x_i$ ) și fiecare frunză  $y_i$  este o clasă din C.
- ID3 și C4.5 sunt doi algoritmi cunoscuți pentru construirea arborilor de decizie și sunt prezentati în acest curs.

# Clasificatori bayesieni

- Clasificare bayesiană (Naïve Bayes): pentru fiecare exemplu  $x_i$ , metoda calculează probabilitatea pentru fiecare clasă  $y_j$  din  $C$ .
- Clasificarea se face alegând cea mai probabilă clasă pentru fiecare exemplu.
- Cuvântul naiv este folosit deoarece se fac unele presupuneri simplificatoare.

# Mașini cu vectori suport

- Mașini cu vectori suport: această metodă este utilizată pentru clasificarea binară.
- Exemplele sunt clasificate în doar două clase: fie pozitive, fie negative.
- Este o metodă foarte eficientă și poate fi folosită recursiv pentru a construi un clasificator cu mai mult de două clase.

# KNN

- Cei mai apropiati K vecini (K-nearest neighbors): o metoda foarte simplă, dar puternică pentru clasificarea exemplelor bazate pe etichetele vecinilor.

# Metode asambliste

- Metode asambliste: Random Forest, Bagging și Boosting.
- În aceste cazuri, se construiesc mai mulți clasificatori iar clasificarea finală se face prin agregarea rezultatelor acestora.
- De exemplu, un clasificator de tip Random Forest constă din mai mulți arbori de decizie, iar clasa de ieșire este valoarea modala (cea mai frecventă) a claselor returnate de arborii individuali.

# Cuprins

- ❑ Ce este învățarea supervizată
- ❑ Evaluare clasificatori
- ❑ Arbori de decizie: ID3 și C4.5
- ❑ Clasificatori bayesieni (Naïve Bayes)
- ❑ Mașini cu vectori suport (Support vector machines)
- ❑ K-nearest neighbors
- ❑ Metode asambliste: Bagging, Boosting, Random Forest

# Acuratețea și rata de eroare

- ❑ Pentru estimarea eficienței unui clasificator pot fi utilizate mai multe măsuri:
- ❑ Acuratețea (sau acuratețea predictivă) este proporția exemplelor de test corect clasificate de model (sau metodă):

*Numărul de exemple de test corect clasificate*

$$\text{Acuratețea} = \frac{\text{Numărul de exemple de test corect clasificate}}{\text{Numărul total de exemple de test}}$$

- ❑ Rata de eroare este proporția exemplelor de test clasificate incorect:

*Rata de eroare = 1 - Acuratețea*

# Alte măsuri

- ◻ În unele cazuri exemplele sunt clasificate în doar două clase (pozitive și negative). Atunci pot fi definite și alte măsuri.
- ◻ Să considerăm matricea de confuzie care conține numărul de exemple clasificate corect și incorect (pentru exemplele pozitive, precum și pentru exemplele negative):

	<b>Classified as Positive</b>	<b>Classified as Negative</b>
<b>Actual positive</b>	<b>TP = True Positive</b>	<b>FN = False Negative</b>
<b>Actual negative</b>	<b>FP = False Positive</b>	<b>TN = True Negative</b>

# Alte măsuri

Unde:

- ❑ TP = numărul de clasificări corecte pentru exemple pozitive.
  - ❑ TN = numărul de clasificări corecte pentru exemple negative.
  - ❑ FP = numărul de clasificări incorecte pentru exemple negative.
  - ❑ FN = numărul de clasificări incorecte pentru exemple pozitive.
- ❑ **Precizia** este proporția exemplelor pozitive clasificate corect din mulțimea exemplelor clasificate ca pozitive:
- $$\text{Precizie} = \text{TP} / (\text{TP} + \text{FP})$$

# Alte măsuri

- **Senzitivitatea** (Recall, sensitivity) este proporția de exemple pozitive corect clasificate în mulțimea exemplelor pozitive (sau rata de recunoaștere a exemplelor pozitive):

$$\text{Senzitivitatea} = \text{TP} / (\text{TP} + \text{FN})$$

- **Specificitatea** (Specificity) este rata de recunoaștere a exemplelor negative:

$$\text{Specificitatea} = \text{TN} / (\text{TN} + \text{FP})$$

## Alte măsuri

- Formula acurateței poate fi rescrisă astfel:

$$\text{Accuracy} = \frac{\text{Recall} * \text{Pos}}{\text{Pos} + \text{Neg}} + \frac{\text{Specificity} * \text{Neg}}{\text{Pos} + \text{Neg}}$$

unde Pos și Neg sunt numărul total de exemple pozitive și negative.

# Alte măsuri

- Precizia și senzitivitatea sunt utilizate de obicei împreună: pentru unele multimi de test folosind doar una din ele poate duce la o judecată incorectă.
- Exemplu: Un set conține 100 de exemple pozitive și 100 de exemple negative iar clasificatorul produce următorul rezultat:

	<b>Clasificate Pozitive</b>	<b>Clasificate negative</b>
<b>Pozitive</b>	30	70
<b>Negative</b>	0	100

# alte măsuri

- Atunci precizia  $p = 100\%$  dar senzitivitatea  $r = 30\%$ . Putem combina precizia și senzitivitatea obținând scorul  $F_1$  ( **$F_1$ -score**) care este media lor armonica:

$$F1\text{-score} = \frac{\frac{2}{p+r}}{\frac{1}{p} + \frac{1}{r}} = \frac{2pr}{p+r}$$

- Pentru exemplul anterior  $F_1$ -score = 46%; în general  $F_1$ -score este mai aproape de valoarea mai mică a celor două.

# Metode de evaluare

- Metodele de evaluare utilizează un set de date **D** cu exemple etichetate.
- Acest set este divizat în mai multe subseturi și aceste subseturi devin seturi de antrenare / test / validare.
- Notă: Evaluarea se referă la metoda / algoritmul de construire a clasificatorului.

# Metoda Holdout

- ❑ În acest caz, setul de date D se împarte în două: un set de antrenare și un set de test.
- ❑ Setul de test se mai numește **multimea reținută - holdout set** (de aici numele metodei).
- ❑ Clasificatorul obținut folosind setul de antrenare este utilizat pentru clasificarea exemplelor din setul de test.
- ❑ Deoarece aceste exemple sunt etichetate se pot calcula acuratețea, precizia, senzitivitatea și alte măsuri și pe baza lor este evaluat clasificatorul.

# Validarea încrucișată

Există mai multe versiuni de validare încrucișată:

1. Validare încrucișată folosind k subseturi - ***k-fold cross validation.***

Setul de date D este împărțit în k subseturi disjuncte cu aceeași dimensiune.

Pentru fiecare subset se construiește un clasificator folosind acel subset ca set de test și reuniunea subseturilor rămase ca set de antrenare.

În acest fel se obțin k valori pentru acuratețe (una pentru fiecare clasificator). Media acestor valori este acuratețea finală. Valoarea obișnuită pentru k este 10.

# Validarea încrucișată

- 2.** Validare încrucișată folosind 2 subseturi - ***2-fold cross validation***. Pentru  $k = 2$ , metoda de mai sus are avantajul de a folosi seturi mari atât pentru antrenament, cât și pentru testare.

# Validarea încrucișată

## 3. Validare încrucișată stratificată - ***Stratified cross validation***.

Este o variație de validare încrucișată având k subseturi.

Fiecare subset are aceeași distribuție a etichetelor.

- De exemplu, pentru exemple pozitive și negative, fiecare subset conține aproximativ aceeași proporție de exemple pozitive și aceeași proporție de exemple negative.

# Validarea încrucișată

4. Validarea încrucișată având subseturi de un element - ***Leave one out cross validation***. Când D conține doar un număr mic de exemple, o validare încrucișată specială poate fi folosită: fiecare exemplu devine set de test și toate celelalte exemple sunt setul de antrenare.
- Acuratețea pentru fiecare clasificator este fie 100%, fie 0%. Media tuturor acestor valori este acuratețea finală.

# Metoda bootstrap

- ❑ Metoda **bootstrap**: face parte din metodele cu eşantionare şi constă în obţinerea setului de antrenare din setul de date original prin eşantionare cu înlocuire (eşantionul nu este înlăturat din set).
- ❑ Cazurile care nu sunt alese în setul de antrenament sunt utilizate ca set de testare.
- ❑ De exemplu, dacă  $D$  are 1000 de exemple etichetate, alegând la întâmplare un exemplu de 1000 de ori obţinem setul de antrenare. În acesta unele exemple sunt prezente de mai multe ori.

# Metoda bootstrap

- ❑ Statistic, 63,2% din exemplele din D sunt alese pentru setul de antrenare iar 36,8% nu. Aceste 36,8% devin setul de test.
- ❑ După construirea unui clasificator și rularea acestuia pe setul de test este determinată acuratețea și astfel poate fi evaluată metoda de construire a clasificatorului în funcție de valoarea obținută.
- ❑ Mai multe despre această metodă și alte tehnici de evaluare pot fi găsite în [Sanderson 08].

# De ce 63,2%?

- Din Data Mining: Concepts and Techniques\*: să presupunem că avem un set de date cu **d** exemple. Fiecare exemplu are o probabilitate de **1/d** de a fi selectat, deci probabilitatea de a nu fi ales este **(1-1/d)**.
- Cum alegem de **d** ori, probabilitatea ca un exemplu să nu fie ales în tot acest timp este **(1-1/d)<sup>d</sup>**.
- Dacă **d** este mare, probabilitatea se apropiе de **e<sup>-1</sup> = 0.368**. Astfel, 36,8% din exemple nu vor fi selectate pentru setul de antrenare și vor ajunge în setul de test iar restul de 63,2% vor forma setul de antrenare.

\* Jiawei Han, Micheline Kamber, Data Mining: Concepts and Techniques, Second Edition, 2006, Morgan Kaufman, pagina 365.

# Cuprins

- ❑ Ce este învățarea supervizată
- ❑ Evaluare clasificatori
- ❑ Arbore de decizie: ID3 și C4.5
- ❑ Clasificatori bayesieni (Naïve Bayes)
- ❑ Mașini cu vectori suport (Support vector machines)
- ❑ K-nearest neighbors
- ❑ Metode asambliste: Bagging, Boosting, Random Forest

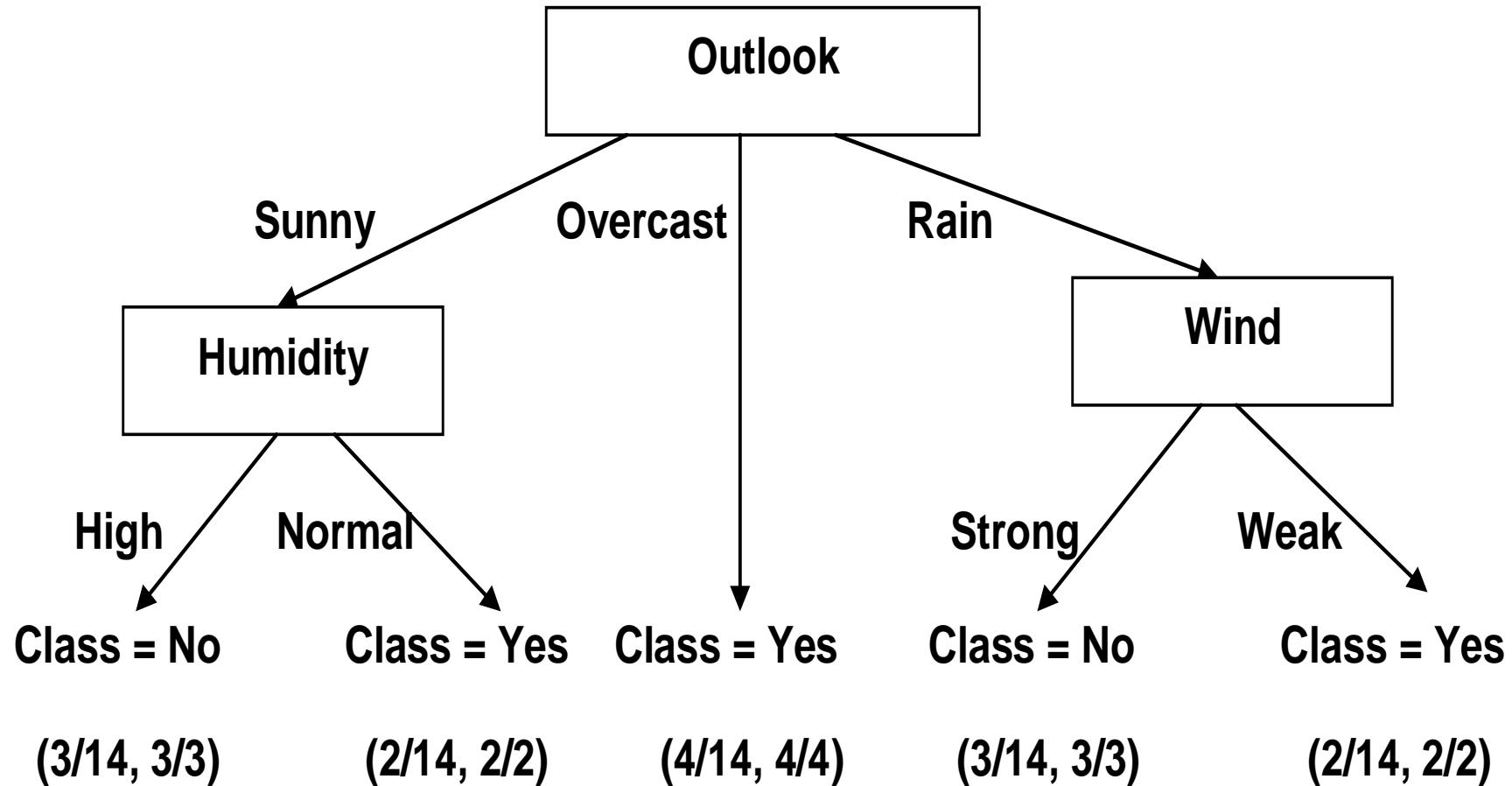
# Ce este un arbore de decizie?

- ❑ Un mod comun de a reprezenta un model sau un algoritm de clasificare este un arbore de decizie.
- ❑ Având un set de antrenare  $D$  și un set  $A$  de atrbute ale exemplelor, fiecare exemplu etichetat din  $D$  are forma:  $(a_1 = v_1, a_2 = v_2, \dots, a_n = v_n)$ .
- ❑ Pe baza acestor atrbute se poate construi un arbore de decizie astfel:

# Ce este un arbore de decizie?

1. Nodurile interne sunt attribute (nicio cale nu conține de două ori același atribut).
2. Ramurile se referă la valorile discrete (una sau mai multe) sau intervale pentru aceste attribute. Uneori pot fi utilizate condiții mai complexe pentru ramificare.
3. Frunzele sunt etichetate cu clase. Pentru fiecare frunză se poate calcula un suport și o încredere: suportul este proporția de exemple care se potrivesc cu calea de la rădăcină la frunza respectivă, iar încrederea este acuratețea clasificării pentru exemplele care se potrivesc cu acea cale. La trecerea de la arbori de decizie la reguli, fiecare regulă are același suport și aceeași încredere ca frunza din care provine.
4. Orice exemplu se potrivește cu o singură cale a arborelui (deci o singură frunză = clasă).

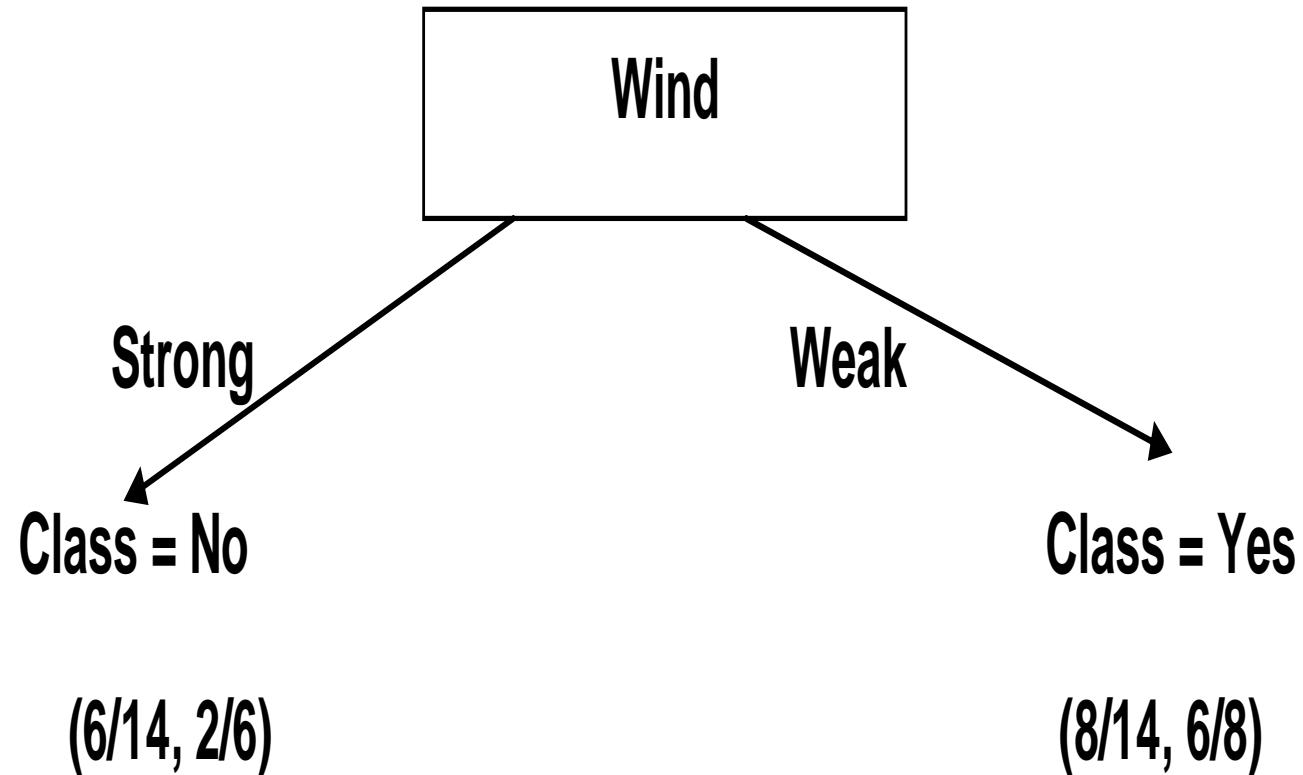
# Exemplu



# Exemplu

- Numerele de pe ultima linie sunt suportul și încrederea asociate fiecărei frunze.
- Pentru același set de date pot fi creați mai mulți arbori de decizie.
- De exemplu un alt arbore de decizie pentru același set de date se află în figura următoare (cu valori mai mici ale încredерii decât arboarele anterior):

# Decision trees



# ID3

- ❑ ID3 (Iterativ Dichotomiser 3) este un algoritm pentru construirea arborilor de decizie introdus de Ross Quinlan în 1986 (vezi [Quinlan 86]).
- ❑ Algoritmul construiește arborele de decizie într-o manieră top-down alegând la fiecare nod atributul „cel mai bun” pentru ramificare:
- ❑ Mai întâi este ales un atribut rădăcină, construind o ramură separată pentru fiecare valoare diferită a atributului.

# ID3

- ❑ Setul de antrenare este de asemenea împărțit, fiecare ramură moștenind exemplele care se potrivesc cu valorile atributelor ramurii respective.
- ❑ Procesul se repetă pentru fiecare descendant până când toate exemplele au aceeași clasă (în acest caz nodul devine o frunză etichetată cu acea clasă) sau toate attributele au fost utilizate (nodul devine, de asemenea, o frunză etichetată cu valoarea modală - clasa majoritară).
- ❑ Un atribut nu poate fi ales de două ori pe aceeași cale; din momentul în care a fost ales pentru un nod, nu va mai fi niciodată testat pentru descendenții acelui nod.

# Cel mai bun atribut

- ❑ Esența ID3 este modul în care este descoperit atributul „cel mai bun”. Algoritmul folosește teoria informației încercând să crească puritatea seturilor de date de la nodul tată la descendenți.
- ❑ Să luăm în considerare un set de date:

$$D = \{e_1, e_2, \dots, e_m\}$$

cu exemple etichetate cu clase din

$$C = \{c_1, c_2, \dots, c_n\}.$$

Atributele exemplelor sunt  $A_1, A_2, \dots, A_p$ .

# Entropia

- Atunci Entropia lui D poate fi calculată astfel:

$$\text{entropy}(D) = - \sum_{i=1}^n \Pr(c_i) \log_2 \Pr(c_i)$$

- Dacă atributul  $A_k$  având  $r$  valori distincte este considerat pentru ramificare, acesta va parta D în subseturile disjuncte  $D_1, D_2, \dots, D_r$ .

# Entropia

- ❑ Entropia combinată a acestor subseturi, calculată ca medie ponderată a acestor entropii este:

$$\text{entropy}(D, A_k) = \sum_{i=1}^r \frac{\text{count}(D_i)}{\text{count}(D)} * \text{entropy}(D_i)$$

- ❑ Toate probabilitățile implicate în ecuațiile de mai sus sunt determinate prin numărare!

# Câştigul informațional

- Deoarece puritatea seturilor de date crește, entropia ( $D$ ) este mai mare decât entropia ( $D, A_k$ ). Diferența dintre ele se numește câştig informațional:

$$Gain(D, A_k) = \text{entropy}(D) - \text{entropy}(D, A_k)$$

- Cel mai bun atribut este cel având câştigul informațional maxim.

# Exemplu

- ❑ Pentru setul de date Play tenis există patru atribute posibile pentru rădăcina arborelui decizional: Aspect, Temperatură, Umiditate și Vânt.
- ❑ Entropia întregului set de date și valorile ponderate pentru împărțirea utilizării celor patru atribute sunt:

$$entropy(D) = -\frac{9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \log_2 \frac{5}{14} = 0.94$$

# Pentru fiecare atribut

$$\text{entropy}(D, \text{Humidity}) = \frac{7}{14} \left( -\frac{3}{7} \log_2 \frac{3}{7} - \frac{4}{7} \log_2 \frac{4}{7} \right) + \frac{7}{14} \left( -\frac{6}{7} \log_2 \frac{6}{7} - \frac{6}{7} \log_2 \frac{6}{7} \right) = 0.79$$

□ La fel:

$$\text{entropy}(D, \text{Wind}) = 0.89$$

$$\text{entropy}(D, \text{Temperature}) = 0.91$$

$$\text{entropy}(D, \text{Outlook}) = 0.69$$

# Cel mai bun atribut: Outlook

- Următorul tabel conține valorile pentru entropie și câștig.
- Cel mai bun atribut pentru nodul rădăcină este Aspect (Outlook), cu câștigul maxim de 0,25:

Attribute	entropy	gain
Humidity	0.79	0.15
Wind	0.89	0.05
Temperature	0.91	0.03
Outlook	0.69	0.25

# Discuție ID3

1. Deoarece este un algoritm greedy ID3 produce un optim local.
2. Atributele cu multe valori distincte duc la un câștig mai mare.  
Pentru rezolvarea acestei probleme câștigul poate fi înlocuit cu raportul de câștig (**gain-ratio**):

$$gain\text{-}ratio(D, A_k) = \frac{gain(D, A_k)}{entropy(D, A_k)}$$

# Discuție ID3

3. Uneori (când doar câteva exemple sunt asociate cu frunzele) arborele se mulează pe datele de instruire (**overfitting**) și nu funcționează bine la alte exemple de test.
- ❑ Pentru a evita overfittingul arborele poate fi simplificat prin reducere (tăiere - pruning):
    - **Pre-pruning** : creșterea este oprită înainte de sfârșitul normal. Frunzele nu sunt 100% pure și sunt etichetate cu clasa majoritară (valoarea modală).
    - **Post-pruning**: după rularea algoritmului, unii subarbore sunt înlocuiți cu frunze. De asemenea, în acest caz, etichetele sunt valori modale pentru exemplele din setul de antrenare care se potrivesc. Tehnica post-pruning este mai bună, deoarece în pre-pruning este greu de estimat când se trebuie să ne oprim.

# Discuție ID3

4. Unele atrbute (de exemplu A) pot fi continue. Valorile pentru A pot fi împărțite în două intervale:

$$A \leq t \text{ și } A > t.$$

- Valoarea lui  $t$  poate fi gasită după cum urmează:
  - Se sortează exemple după A
  - Se alege ca valoare candidat media a două valori consecutive pentru care clasa se schimbă.
  - Pentru fiecare valoare candidat de la pasul anterior se calculează câștigul dacă partitōnarea se face folosind acea valoare. Candidatul cu câștig maxim este ales pentru partitōnare.

# Discuție ID3

- ❑ În acest fel, atributul continuu este înlocuit cu unul discret (două valori, una pentru fiecare interval).
- ❑ Acest atribut concurează cu attributele rămase pentru „cel mai bun” atribut.
- ❑ Procesul se repetă pentru fiecare nod, deoarece valoarea partitionării se poate schimba de la un nod la altul.

# Discuție ID3

5. Costul atributelor: unele attribute sunt mai scumpe decât altele (măsurate nu numai în bani).
- Este mai bine ca attributele cu costuri mai mici să fie mai aproape de rădăcină decât alte attribute.
  - De exemplu, pentru o unitate de urgență, este mai bine să testăm pulsul și temperatura mai întâi și numai atunci când este necesar să efectuăm o biopsie.
  - Acest lucru se poate face prin ponderarea câștigului cu costul:

$$\text{weighted-gain}(D, A_k) = \frac{\text{gain}(D, A_k)}{\text{cost}(A_k)}$$

## C4.5

- ❑ C4.5 este versiunea îmbunătățită a ID3 și a fost dezvoltată de același Ross Quinlan. Câteva caracteristici:
  - Sunt permise atrbute numerice (continue)
  - Tratează cazul valorilor lipsă
  - Utilizează post-pruning pentru a face față la date cu zgomot.
- ❑ Cele mai importante îmbunătățiri de la ID3 sunt:
  1. Atributele sunt alese pe baza raportului de câștig (gain ratio) și nu pur și simplu a câștigului informațional.

## C4.5

2. Folosește post-pruning pentru a reduce dimensiunea arborelui. Tăierea se face numai dacă reduce eroarea estimată. Există două metode de tăiere:
  - Înlocuire sub-arbore: Un sub-arbore este înlocuit cu o frunză, dar fiecare sub-arbore este considerat numai după toți sub-arborii săi. Aceasta este o abordare de jos în sus.
  - Ridicare sub-arbore: Un nod este ridicat și înlocuiește un nod superior. Dar în acest caz, unele exemple trebuie reasignate. Această metodă este considerată mai puțin importantă și mai lentă decât prima.

# Cuprins

- ❑ Ce este învățarea supervizată
- ❑ Evaluare clasificatori
- ❑ Arbori de decizie: ID3 și C4.5
- ❑ Clasificatori bayesieni (Naïve Bayes)
- ❑ Mașini cu vectori suport (Support vector machines)
- ❑ K-nearest neighbors
- ❑ Metode asambliste: Bagging, Boosting, Random Forest

# Naïve Bayes: Generalități

- ❑ Această abordare este una probabilistă.
- ❑ Algoritmii bazați pe teorema lui Bayes calculează pentru fiecare exemplu de test nu o singură clasă, ci probabilitatea pentru fiecare clasă din  $C$  (setul de clase).
- ❑ Dacă setul de date are  $k$  attribute,  $A_1, A_2, \dots, A_k$ , obiectivul este să calculăm pentru fiecare clasă  $c \in C = \{c_1, c_2, \dots, c_n\}$  probabilitatea ca exemplul de test  $(a_1, a_2, \dots, a_k)$  să aparțină clasei  $c$ :  
$$\Pr(\text{Class} = c | A_1 = a_1, \dots, A_k = a_k)$$
- ❑ Dacă este necesară clasificarea, clasa cu cea mai mare probabilitate poate fi atribuită aceluia exemplu.

# Teorema lui Bayes

- Thomas Bayes (1701 - 1761) a fost un cleric presbiterian și un matematician englez.
- Teorema numită după el poate fi exprimată astfel:

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

unde  $P(A | B)$  înseamnă probabilitatea lui A fiind dat B.

# Exemplu

- Studenții din EC004 sunt 70% de la C5 și 30% optionali.
- 20% dintre studenți sunt plasați în primele 3 rânduri, dar pentru optionali acest procent este de 40%.
- Când decanul intră în sala și se așează undeva în primele 3 rânduri, lângă un student, care este probabilitatea ca acel student să fie din categoria optionali?

# Soluție

$$\Pr(\text{Opt}) = 0.3$$

$$\Pr(3 \text{ rânduri} | \text{Opt}) = 0.4$$

$$\Pr(3 \text{ rânduri}) = 0.2$$

□ Deci:

$$(\Pr(3 \text{ rânduri} | \text{Opt}) * \Pr(\text{Opt}))$$

$$\Pr(\text{Opt} | 3 \text{ rânduri}) = \frac{(\Pr(3 \text{ rânduri} | \text{Opt}) * \Pr(\text{Opt}))}{\Pr(3 \text{ rânduri})} =$$

$$= 0.4 * 0.3 / 0.2 = 0.6 \text{ sau } 60\%$$

# Construirea modelului

- Obiectivul este de a calcula:

$$\Pr(\text{Class} = c \mid A_1 = a_1, \dots, A_k = a_k).$$

- Aplicând teorema lui Bayes:

$$\Pr(C = c_j \mid A_1 = a_1, \dots, A_k = a_k) =$$

$$\frac{\Pr(A_1 = a_1, \dots, A_k = a_k \mid C = c_j) * \Pr(f)(C = c_j)}{\Pr(A_1 = a_1, \dots, A_k = a_k)} =$$

$$\frac{\Pr(A_1 = a_1, \dots, A_k = a_k \mid C = c_j) * \Pr(f)(C = c_j)}{\sum_{x=1}^n \Pr(A_1 = a_1, \dots, A_k = a_k \mid C = c_x) \Pr(f)(C = c_x)}$$

# Construirea modelului

- Facem următoarea presupunere: „toate atributele sunt condițional independente dându-se clasa  $C = c_j$ ” atunci:

$$\Pr(A_1 = a_1, \dots, A_k = a_k | C = c_j) = \prod_{i=1}^k \Pr(A_i = a_i | C = c_j)$$

- Din cauza acestei presupunere metoda este numită „naivă”.
- Presupunerea nu este valabilă în toate situațiile,.
- Practica arată însă că rezultatele obținute folosind această presupunere simplificatoare sunt suficient de bune în majoritatea cazurilor.

# Construirea modelului

- În final, înlocuind expresia anterioară în formulă obținem:

$$\Pr(C = c_j | A_1 = a_1, \dots, A_k = a_k) =$$

$$\frac{\Pr(C = c_j) * \prod_{i=1}^k \Pr(A_i = a_i | C = c_j)}{\sum_{x=1}^n \Pr(C = c_x) \prod_{i=1}^k \Pr(A_i = a_i | C = c_x)}$$

- Toate probabilitățile din expresia de mai sus pot fi obținute prin numărare!

# Construirea modelului

- Când este necesară numai clasificarea, numitorul expresiei de mai sus poate fi ignorat (este același pentru toate clasele  $c_j$ ), iar clasa de etichetare se obține prin maximizarea numărătorului:

$$c = \operatorname{argmax}_{cj} \Pr(c_j) * \prod_{i=1}^k \Pr(A_i = a_i | c = c_j)$$

# Exemplu

- Fie versiunea simplificată a tabelului PlayTennis:

Outlook	Wind	Play Tennis
Overcast	Weak	Yes
Overcast	Strong	Yes
Overcast	Absent	No
Sunny	Weak	Yes
Sunny	Strong	No
Rain	Strong	No
Rain	Weak	No
Rain	Absent	Yes

# Exemplu

$$\Pr(\text{Yes}) = 4/8$$

$$\Pr(\text{Overcast} \mid C = \text{Yes}) = 2/4$$

$$\Pr(\text{Overcast} \mid C = \text{No}) = 1/4$$

$$\Pr(\text{Sunny} \mid C = \text{Yes}) = 1/4$$

$$\Pr(\text{Sunny} \mid C = \text{No}) = 1/4$$

$$\Pr(\text{Rain} \mid C = \text{Yes}) = 1/4$$

$$\Pr(\text{Rain} \mid C = \text{No}) = 2/4$$

$$\Pr(\text{No}) = 4/8$$

$$\Pr(\text{Weak} \mid C = \text{Yes}) = 2/4$$

$$\Pr(\text{Weak} \mid C = \text{No}) = 1/4$$

$$\Pr(\text{Strong} \mid C = \text{Yes}) = 1/4$$

$$\Pr(\text{Strong} \mid C = \text{No}) = 2/4$$

$$\Pr(\text{Absent} \mid C = \text{Yes}) = 1/4$$

$$\Pr(\text{Absent} \mid C = \text{No}) = 1/4$$

Calculam pentru exemplul de test:

Sunny	Absent	???
-------	--------	-----

# Exemplu

For C = Yes

$$\Pr(\text{Yes}) * \Pr(\text{Sunny} | \text{Yes}) * \Pr(\text{Absent} | \text{Yes}) = \frac{4}{8} * \frac{1}{4} * \frac{1}{4} = \frac{1}{32}$$

For C = No

$$\Pr(\text{No}) * \Pr(\text{Sunny} | \text{No}) * \Pr(\text{Absent} | \text{No}) = \frac{4}{8} * \frac{1}{4} * \frac{1}{4} = \frac{1}{32}$$

- Rezultatul este că ambele probabilități au aceeași valoare și clasa poate fi oricare dintre ele.

# Caz special: împărțire la 0

- ❑ Uneori, o clasă nu apare cu o anumită valoare de atribut.
- ❑ În acest caz, un termen  $\Pr(A_i = a_i | C = c_j)$  este zero, deci expresia de mai sus pentru probabilitățile fiecărei clase este evaluată la 0/0.
- ❑ Pentru a evita această situație, expresia:

$$\Pr(A_i = a_i | C = c_j) = \frac{a}{b}$$

trebuie modificată.

( $a$  = numărul de exemple de antrenare cu  $A_i = a_i$  și  $C = c_j$  și  $b$  = numărul de exemple de antrenare cu  $C = c_j$ )

# Caz special: împărțire la 0

❑ Expresia modificată este:

$$\Pr(A_i = a_i | C = c_j) = \frac{a + s}{b + s * r}$$

unde:

- $s = 1 /$  Număr de exemple din setul de antrenare
- $r =$  Numărul valorilor distincte pentru  $A_i$

❑ În acest caz toți termenii expresiei sunt mai mari decât zero.

# Caz special: valori non-categorice sau absente

Valori non-categorice sau valori absente:

- ❑ Toate atributele care nu sunt categorice trebuie discretizate (înlocuite cu unele categorice).
- ❑ De asemenea, dacă unele atrbute au valori lipsă, aceste valori sunt ignore.

# Cuprins

- ❑ Ce este Învățarea supervizată
- ❑ Evaluare clasificatori
- ❑ Arbori de decizie: ID3 și C4.5
- ❑ Clasificatori bayesieni (Naïve Bayes)
- ❑ Mașini cu vectori suport (Support vector machines)
- ❑ K-nearest neighbors
- ❑ Metode asambliste: Bagging, Boosting, Random Forest

# SVM: Generalități

- Este prezentată doar ideea generală a metodei de clasificare Support Vector Machines (SVM).
- SVM-urile sunt descrise în detaliu în multe documentații și cărți, de exemplu [Liu 11] sau [Han, Kamber 06].
- Metoda a fost descoperită în Uniunea Sovietică în anii '70 de Vladimir Vapnik și a fost dezvoltată în SUA după ce Vapnik s-a alăturat AT&T Bell Labs la începutul anilor '90 (a se vedea [Cortes, Vapnik 95]).

# SVM: Model

- Considerăm o mulțime de antrenare:

$$D = \{(X_1, y_1), (X_2, y_2), \dots, (X_k, y_k)\}$$

unde:

- $X_i = (x_1, x_2, \dots, x_n)$  este un vector din  $R^n$  (toate componentele lui  $x_i$  sunt numere reale)
- $y_i$  este eticheta de clasă, cu  $y_i \in \{-1, +1\}$ . Dacă  $X_i$  este etichetat cu  $+1$  aparține clasei pozitive, altfel aparține clasei negative ( $-1$ ).

# SVM: Model

□ Un posibil clasificator este o funcție liniară:

$$f(X) = \langle w \cdot X \rangle + b$$

Astfel încât:

$$y_i = \begin{cases} 1 & \text{if } \langle w \cdot X \rangle + b \geq 0 \\ -1 & \text{if } \langle w \cdot X \rangle + b < 0 \end{cases}$$

unde:

- **w** este un vector de ponderi (weight vector),
- $\langle w \cdot X \rangle$  este produsul scalar intre vectorii **w** și **X**,
- **b** este un număr real, și
- **w** și **b** pot fi scalări, după cum se va vedea.

# SVM: Model

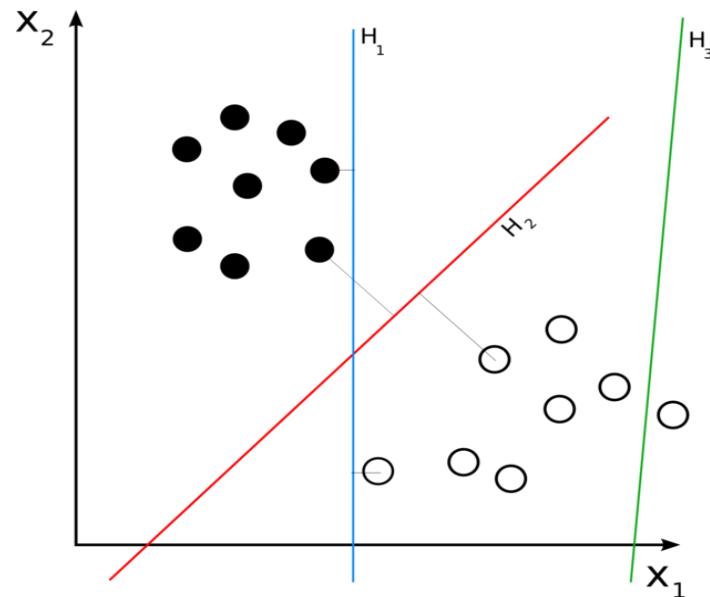
- ❑ Semnificația lui  $f$  este că hiperplanul

$$\langle w \cdot X \rangle + b = 0$$

separă punctele setului de antrenare  $D$  în două:

- o jumătate din spațiu conține valorile pozitive și
  - cealaltă jumătate valorile negative din  $D$  (precum hiperplanurile  $H_1$  și  $H_2$  din figura următoare).
- ❑ Toate exemplele de test pot fi acum clasificate folosind  $f$ : valoarea lui  $f$  dă eticheta pentru exemplu.

# Figura 1

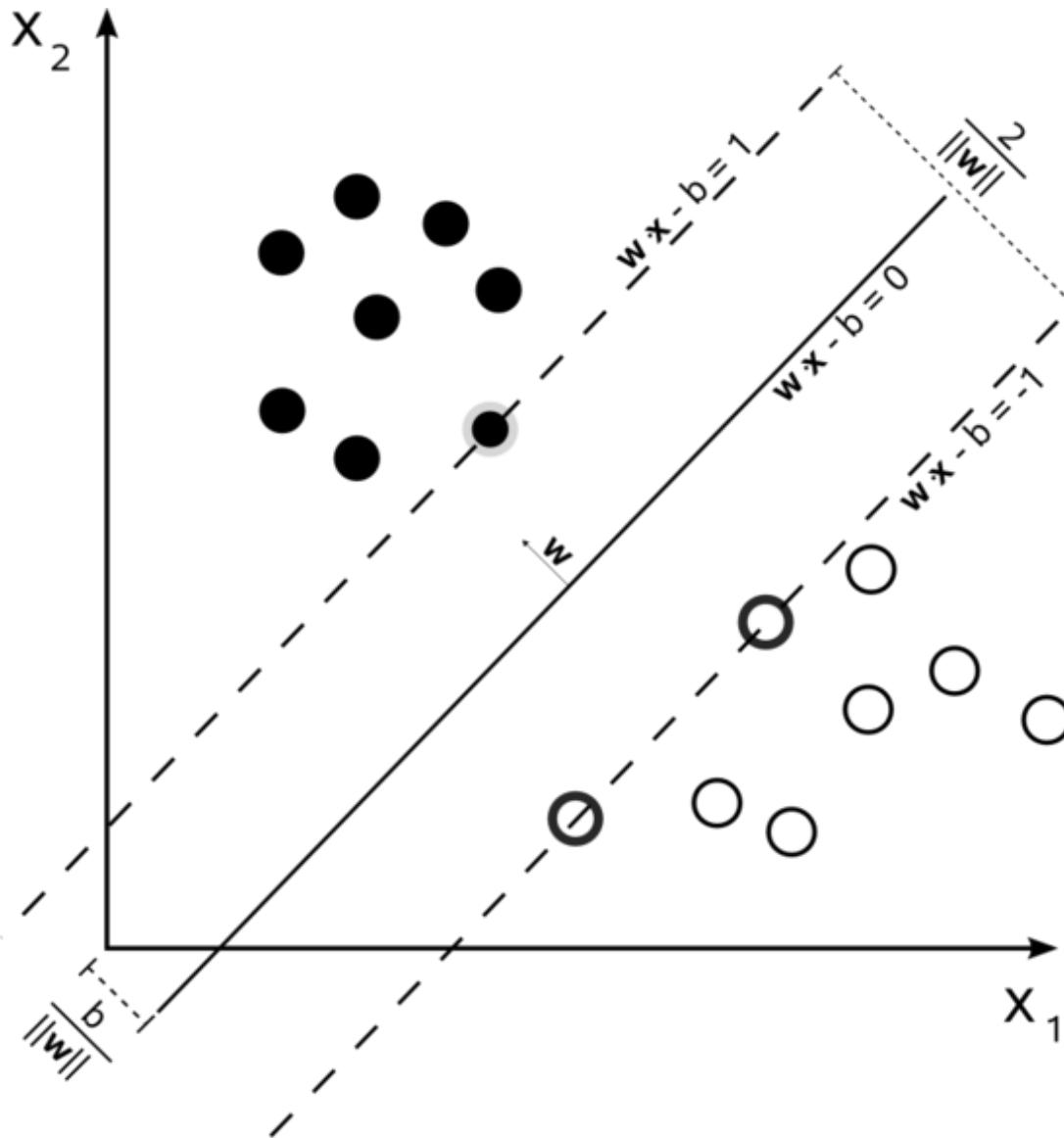


□ Source: Wikipedia

# Cel mai bun hiperplan

- SVM încearcă să găsească „cel mai bun” hiperplan de acea formă.
- Teoria arată că cel mai bun plan este cel care maximizează aşa-numita “margine” (distanța ortogonală minimă între un punct pozitiv și negativ din setul de antrenament - a se vedea figura următoare pentru un exemplu).

# Figura 2



Sursa: Wikipedia

# Separarea non-liniară

- În multe situații nu există un hiperplan care separă exemplele pozitive de cele negative.
- În astfel de cazuri este posibilă maparea punctelor din setul de antrenare (exemplele din D) într-un alt spațiu dimensional superior.
- Aici punctele pot fi liniar separabile.
- Funcția de mapare primește exemple (vectori) din spațiul de intrare  $X$  și le mapează în aşa-numitul spațiu characteristic (feature space)  $F$ :

$$\phi: X \rightarrow F$$

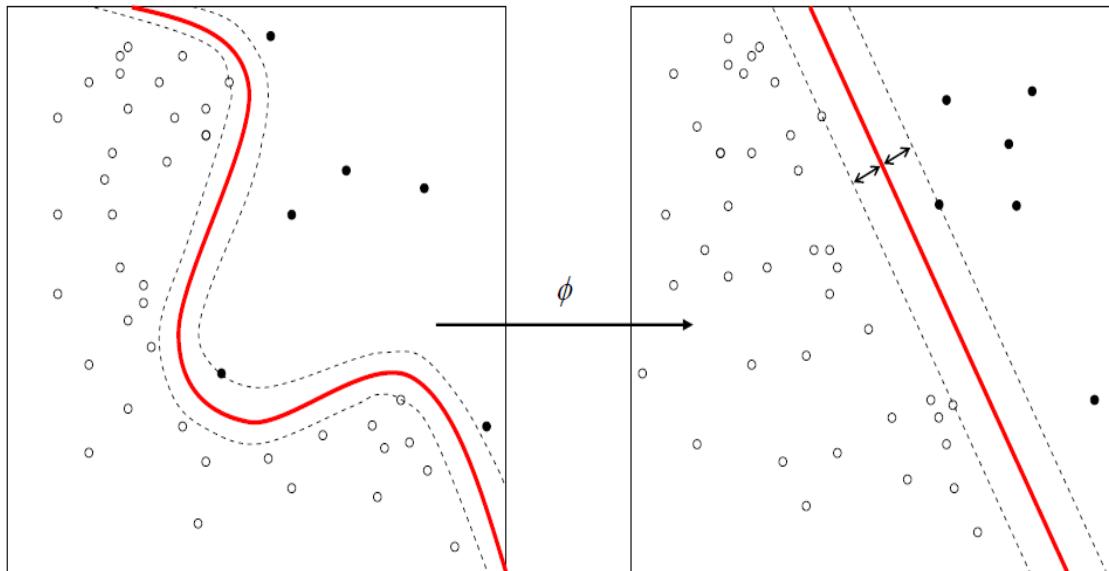
# Separarea non-liniară

- ❑ Fiecare punct  $X$  din  $D$  este mapat în  $\phi(X)$ . După maparea tuturor punctelor avem alt set de antrenare conținând vectori din  $F$  și nu din  $X$ , cu  $\dim(F) \geq n = \dim(X)$ :

$$D = \{(\phi(X_1), y_1), (\phi(X_2), y_2), \dots, (\phi(X_k), y_k)\}$$

- ❑ Pentru un  $\phi$  corespunzător aceste puncte sunt liniar separabile.

# Figura 3



□ Sursa: Wikipedia

# Funcții kernel

- Dar cum putem găsi această funcție de mapare?
- În soluționarea problemei de optimizare pentru găsirea hiperplanului de separare liniară în noul spațiu  $F$  toți termenii sunt doar de forma:

$$\phi(X_i) \cdot \phi(X_j).$$

- Înlocuind acest produs scalar cu o funcție atât în  $X_i$  cât și în  $X_j$ , nevoia de a găsi  $\phi$  dispără. O astfel de funcție se numește **funcție kernel**:

$$K(X_i, X_j) = \phi(X_i) \cdot \phi(X_j)$$

- Pentru a găsi hiperplanul de separare în  $F$  trebuie doar să înlocuim toate produsele scalare cu funcția kernel aleasă și apoi să continuăm cu problema de optimizare ca în cazul separabil.

# Functii kernel

Unele dintre cele mai utilizate functii kernel sunt:

❑ Kernel liniar:

$$K(X, Y) = \langle X \cdot Y \rangle + b$$

❑ Kernel polinomial:

$$K(X, Y) = (a * \langle X \cdot Y \rangle + b)^p$$

❑ Kernel sigmoid:

$$K(X, Y) = \tanh(a * \langle X \cdot Y \rangle + b)$$

# Discuție SVM

- ❑ SVM se folosește când atributele au valori reale
  - Când în setul de antrenare există atributе categorice, este necesară o conversie la valori reale.
- ❑ Când sunt necesare mai mult de două clase, SVM poate fi utilizat recursiv.
  - Prima utilizare separă o clasă; a doua utilizare separă clasa a doua și aşa mai departe. Pentru N clase sunt necesare rulaje N-1.
- ❑ SVM sunt o metodă foarte bună în clasificarea datelor hiper-dimensionale.

# Cuprins

- ❑ Ce este învățarea supervizată
- ❑ Evaluare clasificatori
- ❑ Arbori de decizie: ID3 și C4.5
- ❑ Clasificatori bayesieni (Naïve Bayes)
- ❑ Mașini cu vectori suport (Support vector machines)
- ❑ K-nearest neighbors
- ❑ Metode asambliste: Bagging, Boosting, Random Forest

# kNN

- ❑ Metoda “Cei mai apropiati k vecinii” - K-nearest neighbors (kNN) nu produce un model, ci este o metodă simplă pentru determinarea clasei unui exemplu bazat pe etichetele vecinilor săi aparținând setului de antrenare.
- ❑ Pentru rularea algoritmului este necesară o funcție de distanță pentru calcularea distanței de la exemplul de test până la exemplele din setul de antrenare.

# kNN

□ O funcție  $f(x, y)$  poate fi utilizată ca funcție de distanță dacă sunt îndeplinite patru condiții:

1.  $f(x, y) \geq 0$
2.  $f(x, x) = 0$
3.  $f(x, y) = f(y, x)$
4.  $f(x, y) \leq f(x, z) + f(z, y)$ .

# Algoritm

## Intrare:

- ❑ Un set de date  $D$  care conține exemple etichetate (setul de antrenare)
- ❑ O funcție de distanță  $f$  pentru măsurarea distanței între două exemple
- ❑ Un întreg  $k$  - parametru care spune câți vecini sunt luați în considerare
- ❑ Un exemplu de test  $t$

## Ieșire:

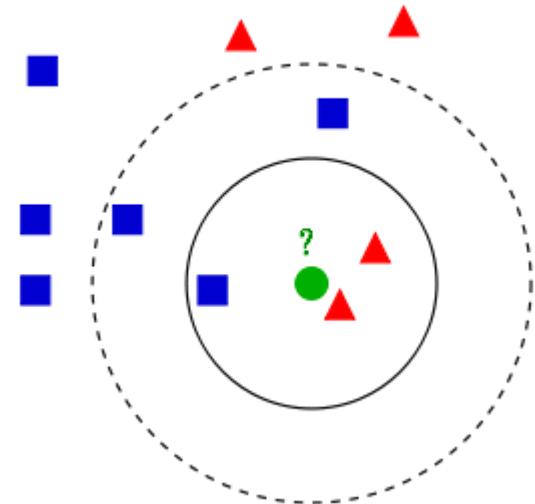
- ❑ Clasa lui  $t$

## Metodă:

- ❑ Se foloseste  $f$  pentru a calcula distanța dintre  $t$  și fiecare punct din  $D$
- ❑ Se selecteză cei mai apropiati  $k$  vecini
- ❑ Se alocă pentru  $t$  clasa majoritară din setul de cei mai apropiati  $k$  vecini.

# Exemplu

- ❑  $K = 3 \rightarrow$  Roșu
- ❑  $K = 5 \rightarrow$  Albastru



- ❑ kNN este foarte sensibil la valoarea parametrului  $k$ .
- ❑ Cea mai bună valoare pentru  $k$  poate fi găsită, de exemplu, prin validare încrucișată.

# Cuprins

- ❑ Ce este Învățarea supervizată
- ❑ Evaluare clasificatori
- ❑ Arbori de decizie: ID3 și C4.5
- ❑ Clasificatori bayesieni (Naïve Bayes)
- ❑ Mașini cu vectori suport (Support vector machines)
- ❑ K-nearest neighbors
- ❑ Metode asambliste: Bagging, Boosting, Random Forest

# Metode asambliste

- Metodele asambliste combină mai mulți clasificatori “slabi” (weak) pentru a obține unul mai bun (mai “puternic” - strong).
- Clasificatorii combinați sunt similari (utilizează aceeași metodă de învățare), dar seturile de antrenare și ponderile exemplelor din ele sunt diferite.

# Bagging

- Numele Bagging vine de la **Bootstrap Aggregating**.
- Așa cum am văzut mai devreme metoda bootstrap face parte din metodele de eșantionare și constă în obținerea unui set de antrenare din datele etichetate inițiale prin eșantionare cu înlocuire.

# Exemplu

Setul original	a	b	c	d	e	f
Set antrenare 1	a	b	b	c	e	f
Set antrenare 2	b	b	c	c	d	e
Set antrenare 3	a	b	c	c	d	f

# Bagging

În ce constă tehnica Bagging:

- ❑ Pornind de la setul de date original, construim **n** seturi de date de antrenare prin eșantionare cu înlocuire (eșantioane bootstrap)
- ❑ Pentru fiecare set de date de antrenare, construim un clasificator folosind același algoritm de învățare (se obțin **n** clasificatori slabii).
- ❑ Clasificatorul final se obține prin combinarea rezultatelor clasificatorilor slabii (prin vot, de exemplu).
- ❑ Baggingul ajută la îmbunătățirea preciziei pentru algoritmi instabili de învățare: arbori de decizie, rețele neuronale.
- ❑ Nu ajută în cazul kNN sau Naïve Bayes.

# Boosting

- ❑ Tehnica Boosting constă în construirea unei secvențe de clasificatori slabî și adăugarea lor în structura clasificatorului final puternic.
- ❑ Clasificatorii slabî sunt ponderați în funcție de acuratețe.
- ❑ De asemenea, datele sunt reevaluate după ce construim fiecare clasificator slab, astfel încât exemplele clasificate incorrect câștigă în greutate.
- ❑ Rezultatul este că următorii clasificatori slabî din secvență se concentrează mai mult pe exemplele pe care le-au clasificat incorrect clasificatorii slabî precedenți.

# Random forest

- ❑ Tehnica Random forest este un clasificator asamblist format dintr-un set de arbori de decizie. Clasificatorul final produce valoarea modală a claselor de ieșire de fiecare arbore.
- ❑ Algoritmul este următorul:
  1. Alege  $T$  - numărul de arbori de construit.
  2. Alegeti  $m$  - numărul de variabile utilizate pentru ramificare la fiecare nod.  $m \ll M$ , unde  $M$  este numărul de variabile de intrare.

# Random forest

1. Construim  $T$  arbori. Pentru fiecare:

- Construim un set de antrenare prin eșantionare cu înlocuire.  
Din el se crește un arbore de decizie.
  - La fiecare nod, selectăm  $m$  variabile la întâmplare (atribute) și le folosim pentru a găsi cea mai bună ramificare.
  - Crestem arborele la maxim. Nu există tăieri (pruning).
4. Se prezic rezultatele aggregând predicțiilor arborilor (de ex. vot majoritar pentru clasificare, medie pentru regresie).

# Referințe

- ▶ [Liu 11] Bing Liu, 2011. Web Data Mining, Exploring Hyperlinks, Contents, and Usage Data, Second Edition, Springer, chapter 3.
- ▶ [Han, Kamber 06] Jiawei Han, Micheline Kamber, Data Mining: Concepts and Techniques, Second Edition, Morgan Kaufmann Publishers, 2006
- ▶ [Cortes, Vapnik 95] Cortes, Corinna; and Vapnik, Vladimir N.; "Support-Vector Networks", Machine Learning, 20, 1995. <http://www.springerlink.com/content/k238jx04hm87j80g/>
- ▶ [Wikipedia] Wikipedia, the free encyclopedia, [en.wikipedia.org](http://en.wikipedia.org)
- ▶ [Sanderson 08] Robert Sanderson, Data mining course notes, Dept. of Computer Science, University of Liverpool 2008, Classification: Evaluation  
<http://www.csc.liv.ac.uk/~azaroth/courses/current/comp527/lectures/comp527-13.pdf>
- ▶ [Quinlan 86] Quinlan, J. R. 1986. Induction of Decision Trees. *Mach. Learn.* 1, 1 (Mar. 1986), 81-106, <http://www.cs.nyu.edu/~roweis/csc2515-2006/readings/quinlan.pdf>

# Capitolul 11

## Data mining – clustering

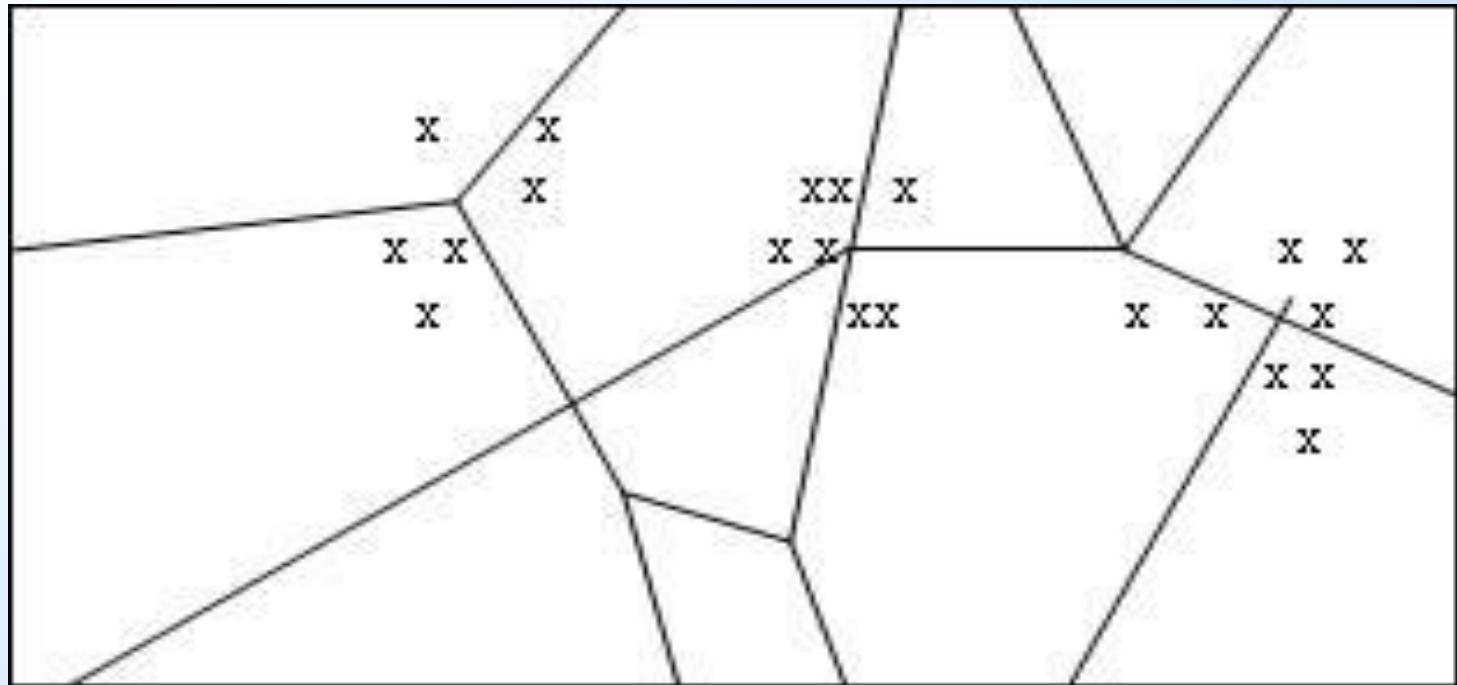
# Problema

- Dându-se puncte într-un spațiu oarecare – deseori un spațiu cu foarte multe dimensiuni – grupează punctele într-un numar mic de *clustere*, fiecare cluster constând din puncte care sunt "apropiate" într-un anume sens.

# Exemple de aplicatie

1. Cu mulți ani în urmă, în timpul unei izbucniri a holerei în Londra, un medic a marcat localizarea cazurilor pe o hartă, obținând un desen care arăta ca în figura urmatoare:

# Exemplu de aplicatie



# Exemple de aplicatie

- Vizualizate corespunzător, datele au indicat că aparițiile cazurilor se grupează în jurul unor intersecții, unde existau puțuri infestate, arătând nu numai cauza holerei ci indicând și ce e de făcut pentru rezolvarea problemei.
- Din păcate nu toate problemele de data mining sunt atât de simple, deseori deoarece clusterele sunt în atât de multe dimensiuni încât vizualizarea este foarte dificilă.

# Exemple de aplicatie

2. *Skycat* a grupat în clustere  $2 \times 10^9$  obiecte cerești în stele, galaxii, quasari, etc.
  - Fiecare obiect era un punct într-un spațiu cu 7 dimensiuni, unde fiecare dimensiune reprezenta nivelul radiației într-o bandă a spectrului.
  - Proiectul Sloan Sky Survey este o încercare mult mai ambicioasă de a cataloga și grupa întregul univers vizibil.

# Exemple de aplicatie

3. Documentele pot fi percepute ca puncte într-un spațiu multi-dimensional în care fiecare dimensiune corespunde unui cuvânt posibil.
  - Poziția documentului într-o dimensiune este dată de numărul de ori în care cuvântul apare în document (sau doar 1 dacă apare, 0 dacă nu).
  - Clusterele de documente în acest spațiu corespund deseori cu grupuri de documente din același domeniu.

# Distanta

- Pentru a discuta dacă o mulțime de puncte sunt suficient de apropiate pentru a fi considerate un cluster avem nevoie de o *măsură a distanței*  $D(x, y)$  care spune cât de depărtate sunt punctele  $x$  și  $y$ .
- Nu orice funcție poate fi utilizată ca funcție de măsurarea distanței.

# Distanta

- Axiomele uzuale pentru o măsură a distanței  $D$  sunt urmatoarele:
  1.  $D(x, y) \geq 0$
  2.  $D(x, x) = 0$ . Un punct este la distanță 0 de el însuși.
  3.  $D(x, y) = D(y, x)$ . Distanța e simetrică.
  4.  $D(x, y) \leq D(x, z) + D(z, y)$ .  
*Inegalitatea triunghiului.*

# Distanța

- Deseori punctele pot fi percepute ca existând într-un spațiu euclidian k-dimensional și distanța între orice două puncte:
  - $x = [x_1, x_2, \dots, x_k]$  și
  - $y = [y_1, y_2, \dots, y_k]$

este dată într-una din manierele uzuale:

- Distanța comună ("norma L<sub>2</sub>")
- Distanța *Manhattan* ("norma L<sub>1</sub>)
- Maximul pe o dimensiune ("norma L<sub>∞</sub>)

# Distanta comuna

- Distanta comuna (sau norma L2) este data de formula cunoscuta:

$$\sqrt{\sum_{i=1}^k (x_i - y_i)^2}$$

# Distanta Manhattan

- Distanta Manhattan (sau norma L1) este data de formula urmatoare:

$$\sum_{i=1}^k |x_i - y_i|$$

- Ea poate fi folosita de exemplu si pentru calculul distantei intre doua puncte pe o placeta cu circuite imprimate multistrat.

# Maximul pe o dimensiune

- Este data de formula:

$$\max_{i=1}^k |x_i - y_i|$$

- Aceasta functie verifica toate cele 4 conditii pentru a fi functie de distanta.
- Poate fi folosita de exemplu pentru spatii euclidiene hiperdimensionale (numar de dimensiuni foarte mare)

# Alte distante

- Unde nu există un spațiu euclidian în care să plasăm punctele gruparea devine mult mai dificilă.
- Iată un exemplu în care are sens: o măsură a distanței în lipsa unui spațiu euclidian

# Alte distante

- Sirurile de caractere, cum sunt sevențele ADN, pot fi similare chiar și dacă există unele inserări și ștergeri precum și modificări ale unor caractere.
- De exemplu, *abcde* și *bcdxye* sunt destul de similare chiar dacă nu au nici o poziție comună și nu au nici chiar aceeași lungime.

# Alte distante

- Astfel, în loc să construim un spațiu euclidian cu câte o dimensiune pentru fiecare poziție, putem defini funcția distanță:

$$D(x, y) = |x| + |y| - 2|\text{LCS}(x, y)|$$

unde LCS este cea mai lungă subsecvență comună lui  $x$  și  $y$ .

- În exemplul nostru  $\text{LCS}(abcde, bcdxye)$  este  $bcd$  de lungime 4, deci  $D(abcde, bcdxye) = 5 + 6 - 2 \times 4 = 3$ ; i.e. sirurile sunt destul de apropiate.

# Alte distante

- Aceasta functie de distanta arata cate caractere trebuie sterse sau adaugate unuia dintre siruri pentru a obtine celalalt sir.
- Intr-adevar, pentru a obtine de exemplu pe *abcde* din *bcdxye* trebuie sa:
  1. Adaugam un a in fata
  2. Stergem pe x
  3. Stergem pe y

# Hiperdimensionalitatea

- O consecință mai puțin intuitivă a lucrului în spații hiperdimensionale este că aproape toate perechile de puncte sunt la o depărtare aproape egală cu media distanțelor între puncte.

Exemplu:

- Să presupunem că aruncăm aleator puncte într-un cub  $k$ -dimensional.
- Pentru  $k=2$ , ne aşteptăm ca punctele să fie răspândite în plan cu unele foarte apropiate între ele și alte perechi aproape la distanța maxim posibilă.

# Hiperdimensionalitatea

- Cu toate acestea, să presupunem  $k$  foarte mare, să zicem 100.000. Indiferent de normă folosită,  $L_2$ ,  $L_1$  sau  $L_\infty$ , știm că:

$$D(x, y) \geq \max_i |x_i - y_i|$$

pentru  $x = [x_1, x_2, \dots]$  și  $y = [y_1, y_2, \dots]$ .

- Pentru  $k$  foarte mare, e foarte posibil să existe o dimensiune  $i$  astfel încât  $x_i$  și  $y_i$  sunt diferite aproape de maximul posibil, chiar dacă  $x$  și  $y$  sunt foarte apropiate în alte dimensiuni.
- Astfel  $D(x, y)$  va fi foarte apropiată de 1. □

# Hiperdimensionalitatea

- O alta consecință interesantă a hiper-dimensionalității este că toți vectorii,

$$x = [x_1, x_2, \dots] \text{ și } y = [y_1, y_2, \dots]$$

sunt aproape ortogonali.

- Motivul este că dacă proiectăm  $x$  și  $y$  pe oricare dintre cele  $C_k^2$  plane formate de două dintre cele  $k$  axe va exista unul în care proiecțiile vectorilor sunt aproape ortogonale (probabilitatea sa existe creste cu numarul de dimensiuni)

# Abordari clustering

- La nivel înalt, putem împărți algoritmii de grupare în două mari clase:
  1. Abordarea tip centroid: 'ghicim' centroizii sau punctele centrale pentru fiecare cluster și asignăm punctele la clusterul având cel mai apropiat centroid.

# Abordari clustering

## 2. Abordarea ierarhică:

- Începem prin a considera că fiecare punct formează un cluster.
- Comasăm repetat clusterele apropiate prin folosirea unei măsuri pentru apropierea a două clustere (e.g. distanța dintre centroizii lor), sau pentru cât de bun va fi clusterul rezultat (e.g. distanța medie de la punctele din cluster la noul centroid rezultat).

# Algoritmul k-means

- Acest algoritm este un algoritm popular care *ține datele în memoria centrală*
- Pe acest algoritm se bazează alti algoritmi de clustering (ex.: BFR)
- $k$ -means alege aleator  $k$  centroizi de cluster dintre punctele care se grupeaza și asignează celelalte puncte la acestia alegând centroidul cel mai apropiat de punctul respectiv.

# Observatie

- Dupa asignarea tuturor punctelor se recalculeaza centroizii clusterelor ca centrul de greutate al punctelor din fiecare cluster.
- El nu este de obicei unul din punctele care formeaza clusterul ci un punct intre ele in spatiul euclidian respectiv.
- Procesul de asignare puncte – recalculare centroizi se repeta pana se indeplineste o conditie de oprire.

# Observatie

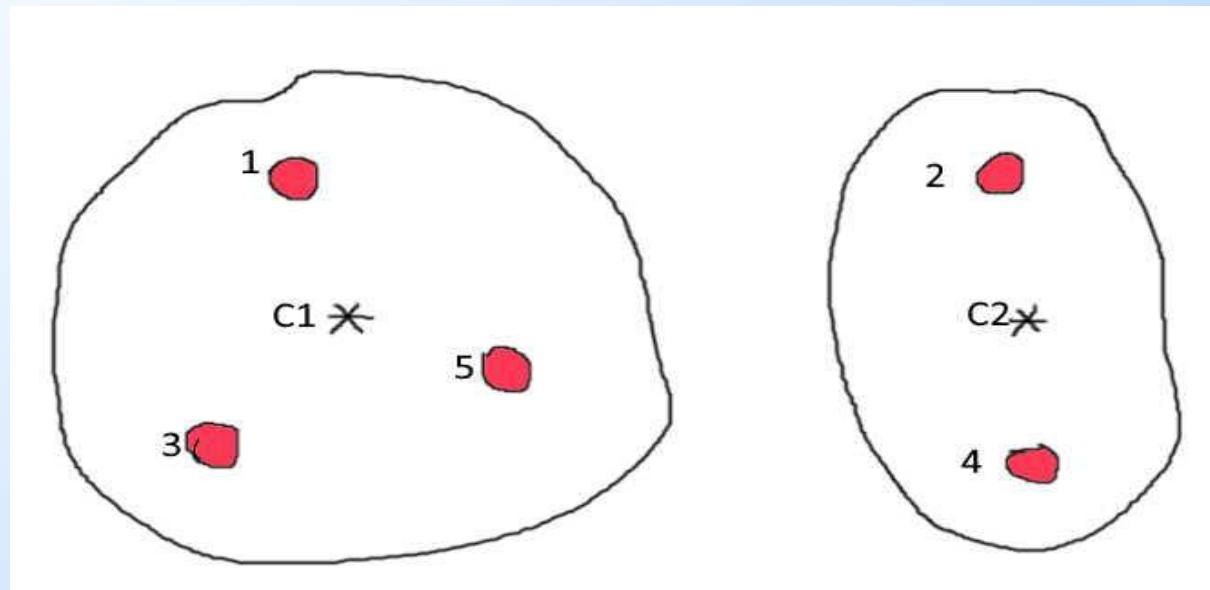
- Conceptul de centroid nu este valabil decat in cazul clusteringului in spatii euclidiene.
- Pentru cazurile in care nu avem un spatiu euclidian (punctele nu au coordonate numerice) exista o serie de variante ale algoritmului precum k-medoids sau k-modes.

# Exemplu

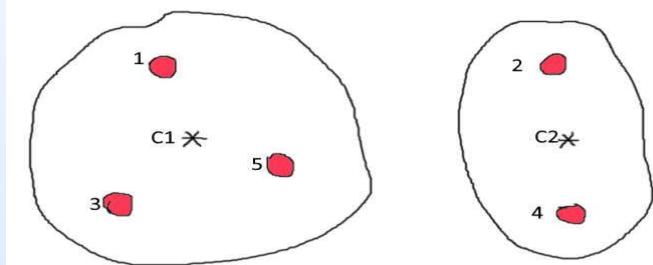
- Un exemplu foarte simplu cu cinci puncte în două dimensiuni.
- Considerăm pentru  $k$  valoarea 2 ( $k$  este un parametru de intrare în algoritm).
- Presupunem că alegem punctele 1, 2 ca centroizi initiali.
- Așignam punctele 3, 4 și 5 la cel mai apropiat centroid.

# Exemplu

□ Rezultatul este urmatorul:



# Exemplu



- Când considerăm punctul 3, presupunem ca acesta este mai apropiat de 1, deci 3 se adaugă clusterului conținând 1.
- Presupunem că atunci când asignăm 4 găsim că 4 este mai aproape de 2 decât de 1, deci 4 se alătură lui 2 în clusterul acestuia.
- În final, 5 este mai aproape de 1 decât de 2, deci el se adaugă la clusterul {1, 3}.
- Recalculam acum centroizii și gasim noiii centroizi C1 și C2.

# Aplicare k-means

- Repetam acum operatia si asignam toate cele 5 puncte la cei mai apropiati centroizi (dintre C1 si C2).
- Obtinem aceleasi clustere si aleiasi centroizi.
- Este evident ca daca vom continua nu obtinem altceva. Procesul se incheie.
- Am obtinut clusterele {1, 3, 5} si {2, 4}.

# Oprire k-means

□ Criteriile de oprire pot fi:

1. Clusterele nu se schimbă de la o iterație la alta (ca în exemplul anterior).
2. Modificările clusterelor sunt sub un prag fixat (de exemplu, nu mai mult de  $p$  puncte schimbă clusterul între două iterării successive).
3. Mișcarea centroizilor este sub un prag dat (de exemplu, suma distanțelor dintre pozitiile vechi și cele noi pentru centroizi nu depășește între două iterării successive un prag fixat).

# Oprire k-means

- Criteriile de oprire - continuare:
  4. Scăderea SSD (suma pătratelor distanțelor) este sub un prag dat.
- SSD măsoară cat de compacte sunt clusterele (ca ansamblu).
- Este suma pătratelor distanțelor de la fiecare punct la centroidul său:

$$SSD = \sum_{i=1}^k \sum_{p \in \text{Cluster}_i} \text{Dist}(p, \text{Centroid}(\text{Cluster}_i))^2$$

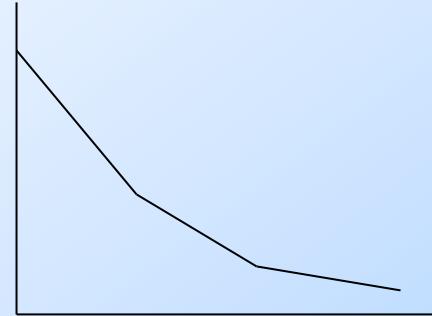
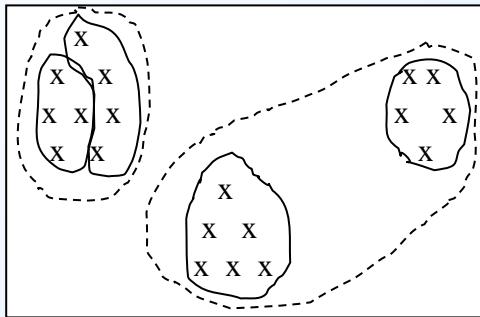
# Aplicare k-means

- Dacă nu suntem siguri de valoarea lui  $k$  putem încerca valori diferite pentru  $k$ .
- Procesul se opreste când găsim cel mai mic  $k$  astfel încât mărirea lui  $k$  nu micșorează prea mult distanța medie a punctelor față de centroidul lor.
- Exemplul urmator ilustrează acest lucru.

# Alt exemplu

- Să considerăm datele din figura urmatoare.
- În mod clar  $k=3$  este numărul corect de clustere dar să presupunem ca întâi încercăm  $k=1$ .
- În acest caz toate punctele sunt într-un singur cluster și distanța medie la centroid va fi mare.

# Alt exemplu - cont



- Presupunem că apoi încercăm  $k=2$ .
- Unul dintre cele trei clustere va fi un cluster iar celealte două vor fi forțate să creeze împreuna un singur cluster, asa cum arată linia punctată.
- Distanța medie a punctelor la centroid de va micșora astfel considerabil.

# Alt exemplu - cont

- Daca luăm  $k=3$  atunci fiecare dintre clusterele vizibile va forma un cluster iar distanța medie de la puncte la centroizi se va micșora din nou, aşa cum arată graficul din figura.
- Totuși, dacă mărim  $k$  la 4 unul dintre adevăratele clustere va fi partaționat artificial în două clustere apropiate, aşa cum arata liniile continui.

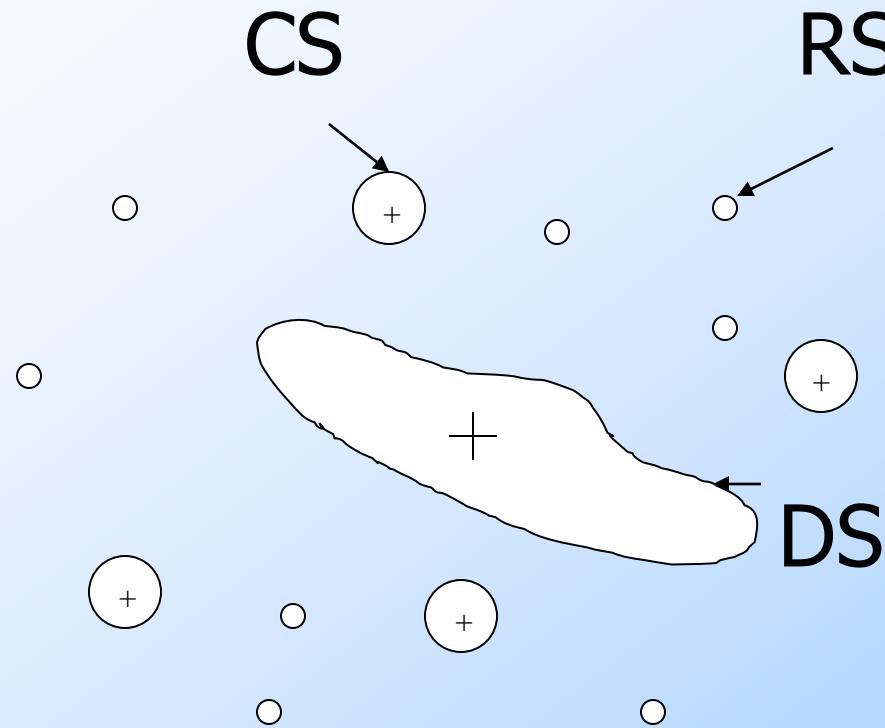
# Alt exemplu - cont

- Distanța medie la centroid va scădea puțin dar nu mult.
- Acest eșec de a merge mai departe ne arată că valoarea  $k=3$  este corectă chiar dacă datele sunt în atât de multe dimensiuni încât nu putem vizualiza clusterele.
- În acest fel aflam valoarea corecta a lui  $k$  – numarul de clustere

# Algoritmul BFR

- Bazat pe  $k$ -means, acest algoritm citește datele o singură dată în tranșe egale cu memoria centrală disponibilă la fiecare pas.
- Algoritmul lucrează cel mai bine dacă clusterele sunt normal distribuite în jurul unui punct central, eventual cu o deviație standard diferită în fiecare dimensiune.

# Reprezentare clustere în BFR



Cei care au creat acest algoritm și-au reprezentat clusterele ca pe niște galaxii.

# Reprezentare clustere în BFR

Un cluster constă din:

1. Un nucleu central numit **Discard set - DS**.
  2. Mulțimea acestor puncte este considerată ca aparținând în mod sigur clusterului.
- Toate punctele din această mulțime sunt înlocuite de niște statistici simple, descrise în continuare.
  - Notă: deși numite puncte « de aruncat » acestea au de fapt un efect semnificativ pe parcursul execuției algoritmului de vreme ce determină colectiv unde este centroidul și care este deviația standard a clusterului în fiecare dimensiune.

# Reprezentare clustere în BFR

2. Galaxii înconjurătoare, numite colectiv  
**Compression set** – CS (*Mulțimea comprimată*).
  - Fiecare subcluster din CS constă într-un grup de puncte care sunt suficient de apropiate unele de altele încât pot fi înlocuite cu statisticile lor, la fel ca și DS.
  - Totuși, ele sunt suficient de departe de orice centroid de cluster încât nu suntem încă siguri de care cluster aparțin.

# Reprezentare clustere în BFR

- Stele individuale care nu sunt parte a nici unei galaxii sau subgalaxii,  
*Mulțimea reținută (Retained set – RS)*.
- Aceste puncte nici nu pot fi asignate vreunui cluster și nici grupate în vreun subcluster al CS.
- Ele sunt stocate în memoria centrală ca puncte individuale împreună cu statisticile pentru DS și CS.

# Reprezentare comprimata

- Statisticile utilizate pentru a reprezenta fiecare cluster din DS și fiecare subcluster din CS sunt:
  1. Contorul numărului de puncte  $N$ .
  2. Vectorul sumelor coordonatelor punctelor în fiecare dimensiune. Vectorul este notat cu  $SUM$  iar componenta pentru dimensiunea  $i$  cu  $SUM_i$ ;
  3. Vectorul sumelor pătratelor coordonatelor punctelor în fiecare dimensiune notat cu  $SUMSQ$ . Componenta pentru dimensiunea  $i$  cu  $SUMSQ_i$ ;

# Reprezentare - cont

- De notat că aceste trei tipuri de informații, totalizând în cazul în care avem  $k$  dimensiuni  $2k+1$  numere sunt suficiente pentru a calcula statistici importante pentru un cluster sau subcluster.
- Este mai convenabil de menținut pe măsură ce punctele sunt adăugate la cluster decât, să spunem, media și varianța în fiecare dimensiune.

# Reprezentare - cont

- Cordonata  $\mu_i$  a centroidului clusterului în dimensiunea  $i$  este  $SUM_i/N$
- Varianța (dispersia) în dimensiunea  $i$  este:

$$\frac{SUMSQ_i}{N} - \left( \frac{SUM_i}{N} \right)^2$$

- iar deviația standard  $\sigma$  este rădăcina pătrată a acesteia.

# Procesare

- La prima încărcare cu date a memoriei centrale, BFR selectează  $k$  centroizi de clustere utilizând un algoritm oarecare lucrând în memoria centrală, e.g. se ia un eşantion al datelor, se optimizează exact clusterele şi se aleg centroizii lor ca centroizi iniţiali.
- O memorie centrală de puncte este procesată la fel în toate încărcările cu date urmatoare după cum urmează:
  1. Se determină care puncte sunt suficient de apropiate de un centroid curent astfel încât pot fi luate în DS iar statisticile lor ( $N$ ,  $SUM$ ,  $SUMSQ$ ) combinate cu statisticile anterioare ale clusterului.

# Procesare

2. În memoria centrală se încearcă gruparea punctelor care nu au fost încă plasate în DS, inclusiv puncte ale RS din pașii precedenți.
  - Dacă găsim un cluster de puncte a căror varianță este sub un prag ales, atunci vom privi aceste puncte ca un subcluster, le înlocuim cu statisticile lor și le considerăm parte a CS.
  - Toate celelalte puncte vor fi plasate în RS.

# Procesare

- Luăm în considerare unirea unui subcluster nou apărut cu un subcluster anterior din CS.
- Testul pentru a vedea dacă este de dorit să facem asta este ca multimea combinată de puncte să aibă o varianță sub un anumit prag.
- De notat că statisticile ținute pentru subclusterele din CS sunt suficiente pentru a calcula varianța mulțimii combine.

# Procesare

- Dacă este ultimul pas, i.e. nu mai sunt date, atunci putem asigna subclusterele din CS și punctele din RS la cel mai apropiat cluster de ele chiar dacă ele vor fi destul de departe de orice centroid de cluster.
- În felul acesta obținem clusterele finale produse de algoritm

# Scalarea multidimensională

- În multe cazuri nu avem un spațiu euclidian ci doar o multime de puncte și distanța între oricare două dintre acestea.
- Exemplu: un graf în care cunoaștem lungimea fiecarui arc. Din aceste lungimi putem afla distanța între oricare două noduri ca fiind lungimea drumului minim între ele

# Scalarea multidimensională

- Se poate demonstra că având  $N$  puncte și distanțele între oricare 2 dintre ele putem crea un spațiu cu  $N-1$  dimensiuni
- În acest spațiu punctele sunt plasate plasate exact (distanța calculată din coordonatele după plasare este aceeași cu distanța de la care s-a plecat pentru orice pereche de puncte)

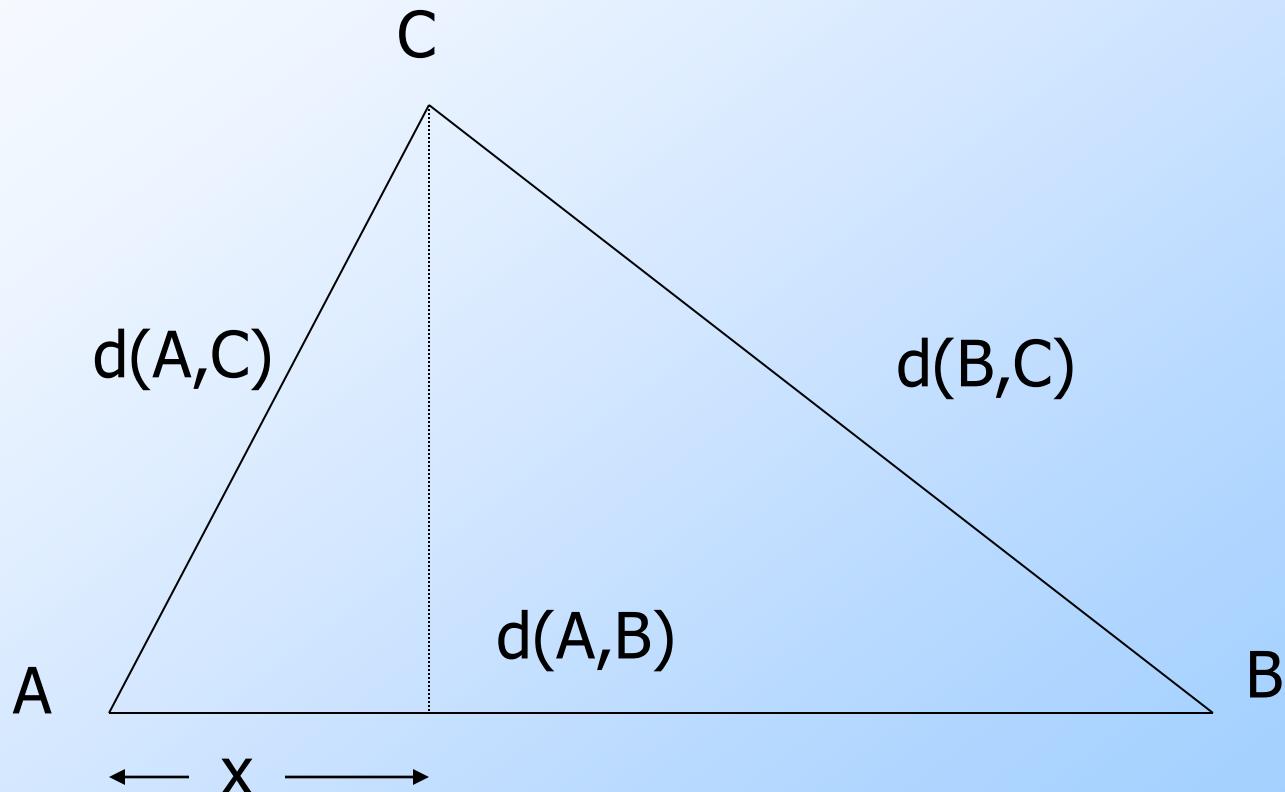
# Scalarea multidimensională

- Problema este că în cazul unui număr mare de puncte rezulta un număr mare de dimensiuni (spatiu hiperdimensional)
- Ideal ar fi să plasam cât mai exact cele  $N$  puncte într-un spatiu având  $K$  dimensiuni unde  $K \ll N$ .
- Acest proces se numește scalare multidimensională.
- Plasarea celor  $N$  puncte nu este 100% exactă (distanțele calculate din coordonatele rezultate nu sunt total exacte cu cele de la care s-a pornit)

# Scalarea multidimensională

- Formula de baza folosita este cea prin care avand doua puncte putem afla distantele proiectiei unui al treilea punct pe segmentul format de primele doua puncte.
- Formula este obtinuta din teorema lui Pitagora generalizata (teorema cosinusului)

# Proiectia lui C pe AB



$$x = (d^2(A,C) + d^2(A,B) - d^2(B,C))/(2d(A,B))$$

# Fastmap

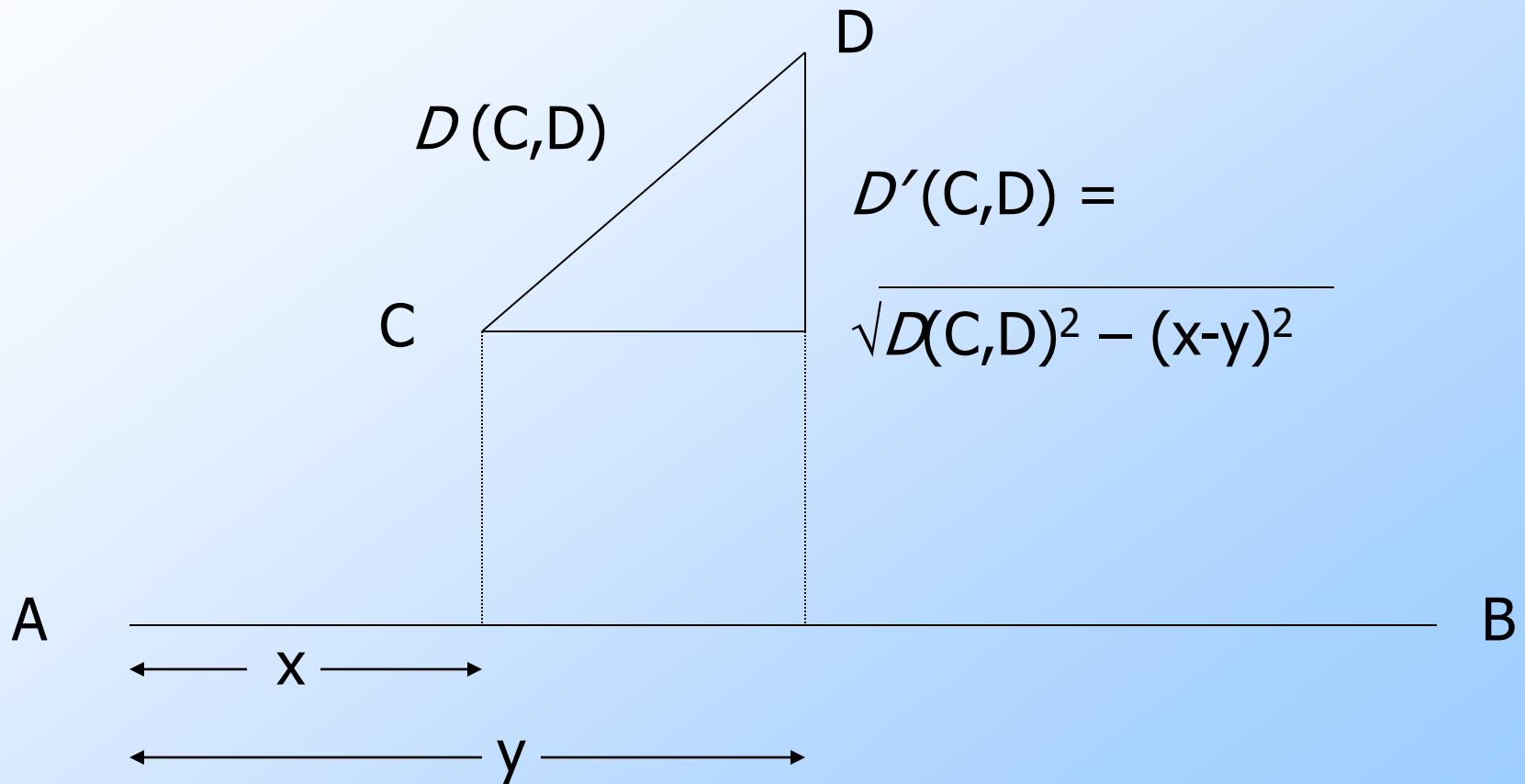
- Algoritmul Fastmap este unul dintre algoritmii de scalare multidimensională.
- Acesta este un algoritm prin care se calculează succesiv coordonatele punctelor, cate o coordonată (o dimensiune) la fiecare pas.
- Pasul algoritmului este urmatorul:

# Fastmap

1. Se aleg doua puncte aflate la distanta cat mai mare, **a** si **b**. Acestea devin o axa de coordonate (cu originea in **a**).
2. Pentru orice punct **c** din cele N se calculeaza coordonata pe aceasta axa conform formulei anterioare:

$$x = (D^2(a, c) + D^2(a, b) - D^2(b, c)) / (2 * D(a, b))$$

# Fastmap



# Fastmap

3. Pentru urmatoarele axe se vor folosi nu distantele initiale dintre puncte ci distante reportate in modul urmator:

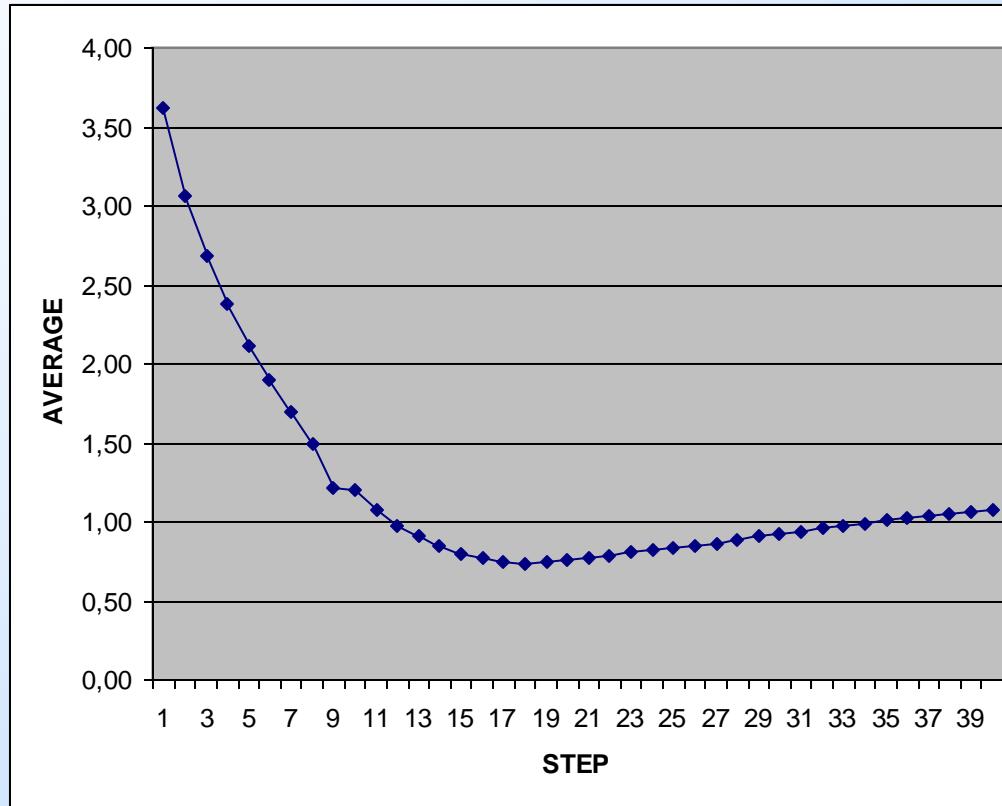
$$D'^2 = D^2 - (x - y)^2$$

4. Procesul se sfarseste dupa calculul numarului dorit de coordonate sau cand nu mai pot fi alese noi axe de coordonate.

# Probleme in cazuri reale

- In cazurile reale (matricea nu este euclidiană) se poate întâmpla ca patratul lui  $D'$  calculat cu formula anterioară să dea un număr negativ.
- În astfel de cazuri pentru a putea continua se poate lua  $D' = 0$ .
- Aceasta alegere duce însă la erori care se propagă.
- Iată un exemplu de rulare pentru algoritm în cazul în care sunt considerate 2000 de noduri.

# Probleme in cazuri reale



# Probleme in cazuri reale

- Figura arata media diferenței între Dreal și Dcalculat unde:
  - Dreal este distanța între puncte de la care s-a pornit (cunoscută prin ipoteza problemei)
  - Dcalculat este distanța dintre puncte calculată pe baza coordonatelor obținute până la pasul respectiv.
  - Se observă că există un minim după 18 pași (spatiu optim are deci pentru acest exemplu 18 dimensiuni,  $k=18$ )

# Probleme in cazuri reale

- In mod normal graficul ar trebui sa tinda asimptotic catre 0.
- Faptul ca nu se intampla asa e datorat erorii induse de considerarea lui  $D' = 0$  in cazul in care patratul este negativ.
- Erorile acumulate fac ca dupa al 18-lea pas graficul sa inceapa sa creasca.

# Bibliografie

- J.D.Ullman - CS345 --- Lecture Notes,  
Clustering I, II

<http://infolab.stanford.edu/~ullman/cs345-notes.html>

# Sfârșitul capitolului 10

# Capitolul 12

## Data mining – cautare pe web

# Căutarea pe web

## □ Puncte importante :

- 1. Rangul paginii*, pentru descoperirea celor mai "importante" pagini de web, utilizat de Google.
- 2. Indecși și autorități*, o evaluare mai detaliată a importanței paginilor web utilizând o variantă a calculului de valori proprii utilizată pentru rangul paginii.

# Rangul paginii

- Intuitiv rezolvăm problema definiției "importanței" recursiv : o pagina este importantă dacă pagini importante conțin legături către ea.
- Creăm o matrice stochastică a Internetului astfel :
  - Fiecare pagină /corespunde liniei /și coloanei /a matricii.
  - Dacă pagina  $j$  are  $n$  succesiuni (legături), atunci elementul  $i, j$  al matricii este  $1/n$  dacă pagina  $i$  este unul dintre acești succesiuni ai paginii  $j$  și 0 altfel.

# Rangul paginii

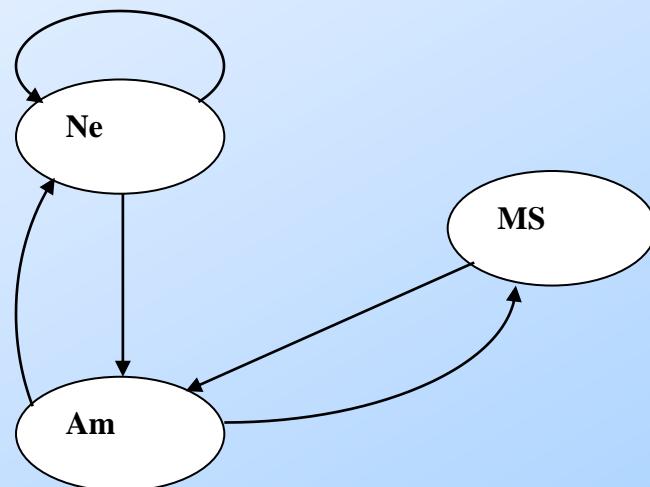
- ❑ Intuiția care stă în spatele acestei matrici este :
- ❑ Să ne imaginăm că inițial fiecare pagină are o unitate de importanță.
- ❑ La fiecare pas fiecare pagină își împarte importanța între succesorii săi și primește noi fracțiuni de importanță de la predecesorii săi.

# Rangul paginii

- Dupa mai multe iteratii, importanța fiecărei pagini atinge o limită care este componenta corespunzătoare ei din vectorul principal de valori proprii al matricii.
- Această importanță este de asemenea probabilitatea ca un navigator pe web, pornind de la o pagină aleatoare și urmând legături aleator alese din fiecare pagină, să ajungă la pagina în discuție după o lungă serie de legături.

# Exemplul 1

□ In 1839 Internetul consta din doar trei pagini : Netscape, Amazon si Microsoft. Legăturile între aceste trei pagini erau ca în figura urmatoare:



# Exemplul 1

- ❑ Fie  $[n, m, a]$  vectorul importanței pentru cele trei pagini : Netscape, Microsoft respectiv Amazon.
- ❑ Atunci ecuația care descrie valorile asymptotice ale acestor trei variabile este :

$$\begin{bmatrix} n \\ m \\ a \end{bmatrix} = \begin{bmatrix} 1/2 & 0 & 1/2 \\ 0 & 0 & 1/2 \\ 1/2 & 1 & 0 \end{bmatrix} \begin{bmatrix} n \\ m \\ a \end{bmatrix}$$

# Exemplul 1

- ❑ Prima coloană a matricii reflectă faptul că Netscape își divide importanța între el însuși și Amazon. A doua coloană că Microsoft dă toată importanța sa către Amazon.
- ❑ Putem rezolva ecuații ca aceasta începând cu asertiunea că  $n = m = 1$  și aplicând repetat matricea la estimarea curentă a acestor valori.

# Exemplul 1

□ Primele patru iterații dă următoarele estimări :

$n$	=	1	1	$5/4$	$9/8$	$5/4$
$m$	=	1	$1/2$	$3/4$	$1/2$	$11/16$
$a$	=	1	$3/2$	1	$11/8$	$17/16$

□ La limită, soluția este  $n = a = 6/5$  ;  $m = 3/5$ .

# Observatii

- De notat că nu putem să obținem niciodată valorile absolute ale lui  $n$ ,  $m$  și  $a$  ci ***doar raportul lor***, de vreme ce aserțiunea inițială că fiecare a pornit de la 1 a fost arbitrară.
- Deoarece matricea este stochastică (suma pe fiecare coloană este 1), procesul de ***relaxare*** de mai sus converge către ***vectorul principal de valori proprii*** al matricii.

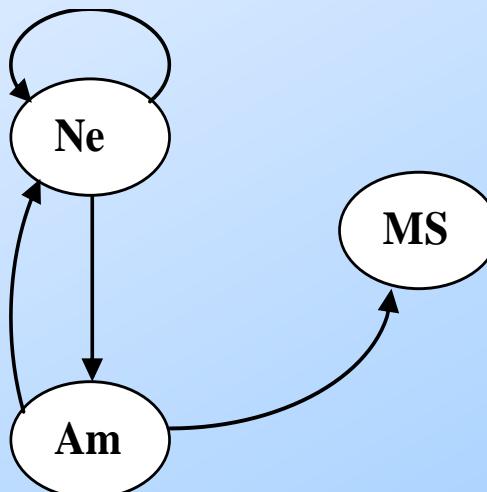
# Probleme cu grafuri reale

## □ 2 tipuri de probleme:

1. *Dead end*: o pagină care nu are succesiuni nu are către cine să-și trimită importanța. În final totă importanța "se va scurge" din Internet
2. *Capcane* : un grup de una sau mai multe pagini care nu au legături către pagini din afara grupului vor acumula la final totă importanța din Internet.

## Exemplul 2: Dead end

□ Să presupunem că Microsoft încearcă să profite că este un monopol înlăturând toate legăturile din situl său. Noul Internet este ca în figura urmatoare:



## Exemplul 2

□ Ecuatia matriciala este in acest caz:

$$\begin{bmatrix} n \\ m \\ a \end{bmatrix} = \begin{bmatrix} 1/2 & 0 & 1/2 \\ 0 & 0 & 1/2 \\ 1/2 & 0 & 0 \end{bmatrix} * \begin{bmatrix} n \\ m \\ a \end{bmatrix}$$

□ Se observa ca suma pe coloane nu mai este intotdeauna 1 (coloane cu suma nula)

# Exemplul 2

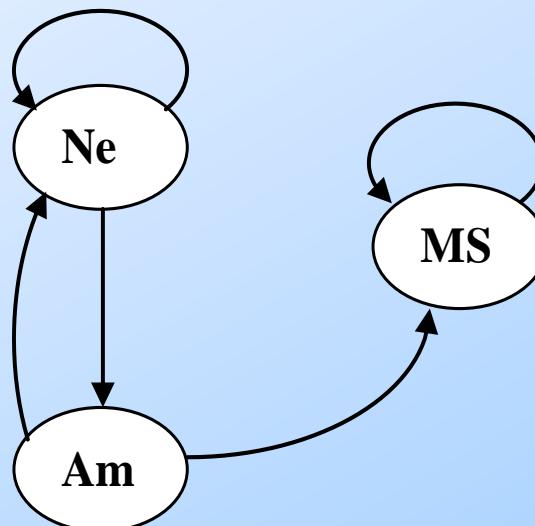
□ Primii patru pași ai soluției iterative sunt :

n	=	1	1	$3/4$	$5/8$	$1/2$
m	=	1	$1/2$	$1/4$	$1/4$	$3/16$
a	=	1	$1/2$	$1/2$	$3/8$	$5/16$

□ In acest caz, fiecare dintre  $n$ ,  $m$  și  $a$  tinde către 0; i.e. toată importanța se scurge afară.

# Exemplul 3

□ Microsoft decide să nu folosească decăt legături către el însuși de acum încolo. Acum, Microsoft a devenit o capcană. Noul Internet este în figura urmatoare:



# Exemplul 3

□ Ecuatia matriciala este in acest caz:

$$\begin{bmatrix} n \\ m \\ a \end{bmatrix} = \begin{bmatrix} 1/2 & 0 & 1/2 \\ 0 & 1 & 1/2 \\ 1/2 & 0 & 0 \end{bmatrix} * \begin{bmatrix} n \\ m \\ a \end{bmatrix}$$

□ Suma pe coloane este 1 dar apar valori de 1 pe diagonala principala a matricii.

# Exemplul 3

□ Primii pasi ai algoritmului produc valorile:

$n$	=	1	1	$3/4$	$5/8$	$1/2$
$m$	=	1	$3/2$	$7/4$	2	$35/16$
$a$	=	1	$1/2$	$1/2$	$3/8$	$5/16$

□ Se observă acumularea pe linia  $m$ .

# Prevenire dead end și capcane

- ❑ În loc de a aplica matricea direct, "taxăm" fiecare pagină cu o fracțiune din importanța sa curentă și distribuim importanța taxată în mod egal tuturor paginilor.
- ❑ Dacă folosim o taxă de 20% ecuația din exemplul 3 devine cea de pe transparentul urmator.

# Prevenire dead end și capcane

□ Ecuatia cu taxare:

$$\begin{bmatrix} n \\ m \\ a \end{bmatrix} = 0.8 \begin{bmatrix} 1/2 & 0 & 1/2 \\ 0 & 1 & 1/2 \\ 1/2 & 0 & 0 \end{bmatrix} \begin{bmatrix} n \\ m \\ a \end{bmatrix} + \begin{bmatrix} 0.2 \\ 0.2 \\ 0.2 \end{bmatrix}$$

□ Soluția acestei ecuații este  $n = 7/11$ ;  $m = 21/11$ ;  $a = 5/11$ .

□ De notat că suma celor trei valori nu este 3 dar obținem o distribuție mult mai rezonabilă a importanței decât în Exemplul 3.

# Spam

- "Spamming" este în acest context încercarea multor situri web de a părea că sunt despre un subiect care atrage vizitatorii fără ca într-adevăr să fie despre acel subiect.
- Google, ca și alte motoare de căutare, încearcă să potrivească cuvintele din cererile de căutare cu cuvinte din pagini web.
- Cu toate acestea, Google, spre deosebire de alte motoare de căutare, tinde să creadă ceea ce spun alții în textul legăturilor despre o pagină web facând mai greu pentru aceasta să pară ca fiind despre ceva ce nu este.

# Spam

- ❑ Utilizarea rangului paginii pentru a măsura importanța în locul unei măsuri mult mai naive ca "numărul de legături către acea pagină" protejează de asemenea impotriva spamului.
- ❑ Masura naivă poate fi înșelată de un spammer care creează 1000 de pagini care se referă între ele în timp ce rangul paginii recunoaște că nici una dintre acestea nu au importanță reală.

# Indecși și autorități

- ❑ Intuitiv, definim "index" și "autoritate" într-un mod mutual recursiv: un index conține legături către multe autorități iar o autoritate este referită de mulți indecși.
- ❑ Autoritățile pot fi pagini care oferă informații despre un subiect.
- ❑ Indecșii sunt pagini care nu furnizează informații ci spun unde se găsesc informații.

# Indecsi si autorități

- Utilizează o formalizare matricială similară cu cea de la rangul paginii dar fără restricția stochastică. Numărăm fiecare legătură ca 1, indiferent de câți succesi sau predecesori are o pagină.
- Aplicarea repetată a matricii duce la divergență, dar putem introduce un factor de scalare pentru a ține valorile calculate pentru gradul de "autoritate" sau de "indexare" pentru fiecare pagina între limite finite.

# Indecsi si autoritati

- Definim matricea  $A$  ale cărei linii și coloane corespund paginilor web având elementul  $A_{ij} = 1$  dacă pagina  $i$  referă pagina  $j$  și 0 altfel.
- De notat că  $A^T$ , transpusa lui  $A$ , arată ca matricea utilizată pentru calculul rangului paginilor dar  $A^T$  are 1 acolo unde matricea pentru rang are fracții.

# Indecși și autorități

- Fie  $a$  și  $h$  doi vectori iar componenta lor  $i$  corespunde gradului de autoritate respectiv indexare a paginii  $i$ . Fie  $\lambda$  și  $\mu$  factorii de scalare corespunzători care vor fi calculați mai târziu. Atunci putem afirma că:
  - $h = \lambda A a$ . Adică gradul de indexare al fiecărei pagini este suma gradelor de autoritate ale tuturor paginilor referite, scalată cu  $\lambda$ .
  - $a = \mu A^T h$ . Adică gradul de autoritate al fiecărei pagini este suma gradelor de indexare ale tuturor paginilor care o referă, scalată cu  $\mu$ .

# Indecși și autorități

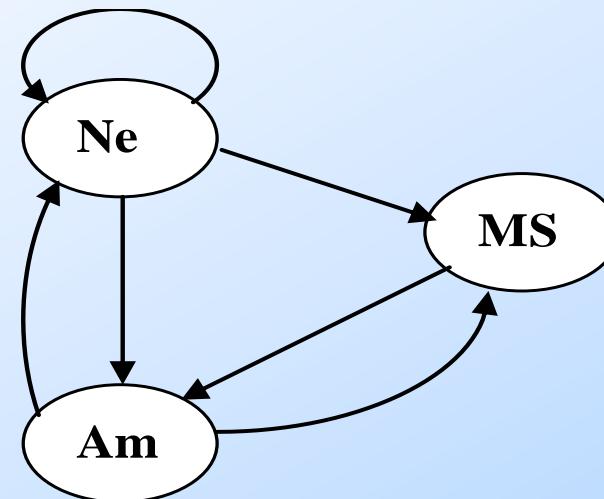
- Din ecuațiile (1) și (2) putem deduce folosind substituția, două ecuații care leagă vectorii  $a$  și  $h$  doar de ei însăși:

$$\begin{aligned} a &= \lambda \mu A^T A a \\ h &= \lambda \mu A A^T h \end{aligned}$$

- Ca urmare, putem calcula  $\mathbf{h}$  și  $\mathbf{a}$  prin relaxare, obținând vectorul principal de valori proprii al matricilor  $AA^T$  și respectiv  $A^TA$

# Exemplul 4

□ Fie graful urmator:



# Exemplul 4

□ Matricile sunt:

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \quad A^T = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \quad AA^T = \begin{bmatrix} 3 & 1 & 2 \\ 0 & 0 & 1 \\ 2 & 0 & 2 \end{bmatrix} \quad A^TA = \begin{bmatrix} 2 & 2 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 2 \end{bmatrix}$$

□ Dacă utilizăm  $\lambda = \mu = 1$  și considerăm că vectorii  $h = [h_n, h_m, h_a]$  și  $a = [a_n, a_m, a_a]$  sunt inițial fiecare  $[1, 1, 1]$ , primele trei iterații ale ecuațiilor pentru  $a$  și  $h$  sunt:

# Exemplul 4

□ Seccesiunea de valori pentru a este:

$a_n$	=	1	5	24	114
$a_m$	=	1	5	24	114
$a_a$	=	1	4	18	84

□ Iar pentru h:

$h_n$	=	1	6	28	132
$h_m$	=	1	2	8	36
$h_a$	=	1	4	20	96

## Exemplul 4

□ Vectorul  $a$ , **scalat corespunzător**, va converge către un vector în care:

- $a_n = a_m$  și
- fiecare dintre aceste numere este mai mare ca  $a_a$  în raportul  $1 + \sqrt{3} / 2$  sau aproximativ 1.36

# Extragerea de cunoștințe din web

## ❑ Puncte importante:

- 1. *Numărarea dinamică a mulțimilor de articole:***  
Căutarea de mulțimi *interesante* de articole într-un spațiu mult prea mare pentru a se putea lua în considerare fiecare pereche de articole.
- 2. "Cărți și autori":** Intrigantul experiment al lui Sergey Brin de extragere de date relationale din web.

# Numarare dinamica

- Problema este de a găsi multimi de cuvinte care apar "neobișnuit de des" împreună pe web, e.g. "New" și "York" sau {Ducesa, de, York}.
- "Neobișnuit de des" poate fi definit în diverse moduri pentru a încorpora ideea că numărul de documente web conținând multimea de cuvinte este mult mai mare decât cel așteptat în cazul în care cuvintele ar fi fost alese la întamplare, fiecare cu probabilitatea sa de apariție într-un document.

# Numarare dinamica

- Un mod adekvat este *entropia per cuvânt din mulțime*. Formal, **interesul unei mulțimi de cuvinte S este:**

$$\frac{\log_2\left(\frac{prob(S)}{\prod_{w \text{ in } S} prob(w)}\right)}{|S|}$$

# Numarare dinamica

- ❑ De notat că împărțim la dimensiunea lui S pentru a evita "efectul Bonferroni", în care sunt atât de multe multimi de o dimensiune dată încât unele, din motive probabilistice, par a fi corelate.
- ❑ Exemplu: Daca  $a$ ,  $b$  și  $c$  (cuvinte) apar fiecare în 1% din toate documentele și  $S=\{a, b, c\}$  apar în 0.1% din documente, interesul lui  $S$  este  $(\log_2(0.001/(0.01 \times 0.01 \times 0.01))/3 = \log_2(1000)/3$  adică aproximativ 3.3.

# Numarare dinamica

- Problema tehnică: interesul nu este monoton sau "Închis în jos" în modul de la produse cu larg suport.
- Astă înseamnă că putem avea o mulțime  $S$  cu o valoare mare a interesului și totuși unele sau chiar toate submulțimile sale stricte să nu fie interesante.
- Prin contrast, dacă  $S$  are suport larg, atunci toate submulțimile sale au cel puțin același suport.
- Observatie: Cu mai mult de  $10^8$  cuvinte diferite apărând pe web nu este posibil nici măcar să considerăm toate perechile de cuvinte.

# DICE

- DICE (dynamic itemset counting engine) vizitează repetat paginile web într-un mod de tip "round-robin" ("zborul prigorului/prigoriei").



# DICE

- ❑ De fiecare dată numără aparițiile anumitor multimi de cuvinte și ale fiecărui cuvânt din aceste multimi.
- ❑ Numarul de multimi numărate este suficient de mic încât contorii lor încap în memoria centrală.

# DICE

- ❑ Din când în când, să spunem la fiecare 5000 de pagini, DICE își reconsideră mulțimile pentru care numără. Înlătură acele mulțimi care au cel mai mic interes și le înlocuiește cu alte mulțimi.
- ❑ Alegerea noilor mulțimi se bazează pe proprietatea numita ***heavy edge*** care este o observație justificată experimental că acele cuvinte care apar în mulțimi cu interes ridicat au probabilitatea mai mare să apară în alte mulțimi cu interes ridicat.

# DICE

- ❑ Astfel, când selectează noi mulțimi pentru a începe numărarea, DICE este direcționat în favoarea cuvintelor care apar deja în mulțimi cu interes ridicat.
- ❑ Totuși, el nu se bazează exclusiv pe aceste cuvinte altfel nu ar putea niciodată să găsească mulțimi cu interes ridicat compuse din multele cuvinte pe care nu le-a considerat niciodată.
- ❑ Unele (dar nu toate) din construcțiile pe care le utilizează DICE pentru crearea noilor mulțimi sunt:

# DICE: noi multimi

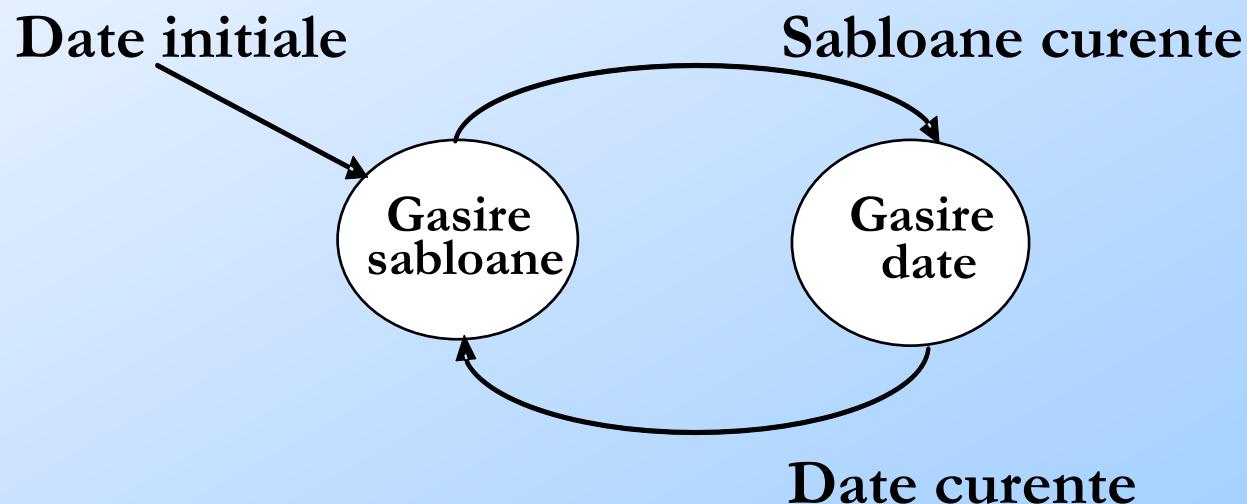
1. Două cuvinte aleatorii. Aceasta este singura regulă independentă de aserțiunea "heavy edge" și ajuta noi cuvinte să ajungă în mulțime.
2. Un cuvânt dintr-o mulțime interesantă și un cuvânt aleator.
3. Două cuvinte din două mulțimi interesante diferite.

# DICE: noi multimi

4. Reuniunea a două multimi interesante a căror intersecție are dimensiunea 2 sau mai mult.
  5.  $\{a, b, c\}$  dacă toate multimile  $\{a, b\}$ ,  $\{a, c\}$  și  $\{b, c\}$  sunt găsite ca fiind interesante.
- Bineînțeles, în general sunt mult prea multe opțiuni de a aplica cele de mai sus în toate modurile posibile astfel încât se utilizează o selecție aleatoare a opțiunilor dând o anumită sansă fiecăreia dintre ele.

# Carti si autori

- ❑ Ideea principală este de a căuta pe web fapte de un anumit tip, de genul celor care ar putea forma o relație de genul *Cărți(titlu, autor)*. Procesarea este sugerată de figura urmatoare:



# Cum lucreaza

1. Se pornește de la un eşantion al tuplurilor (liniilor) care se doresc găsite.
  - În exemplul discutat în lucrarea lui Brin au fost folosite cinci exemple de perechi cu titluri de cărți și autorii acestora.

# Cum lucreaza

2. Pe baza exemplelor cunoscute, se caută pagini unde apar aceste date pe web.
  - Dacă se găsește un şablon care identifică un număr de tupluri cunoscute și este suficient de specific încât e puțin probabil să identifice prea mult, atunci se acceptă acest şablon.

# Cum lucreaza

3. Fiind dată o mulțime de şabloane acceptate, se caută date care satisfac aceste şabloane și se adaugă la mulțimea datelor cunoscute.
4. Se repetă pașii (2) și (3) de un număr de ori. În exemplul citat au fost utilizate patru ciclări care au dus la 15,000 de tupluri; aprox. 95% au fost perechi adevărate titlu-autor.

# Ce este un sablon

Are 5 componente:

- 1. Ordinea;* i.e. daca titlul apare în text înaintea autorului sau vice-versa. Într-un caz general, în care tuplele au mai mult de 2 componente, ordinea va fi dată de permutarea componentelor.
- 2. Prefixul adresei web (URL).*
- 3. Prefixul/ textului, care apare înaintea primului dintre titlu și autor*

# Ce este un sablon

- Are 5 componente:
  4. *Mijlocul*: text care apare între cele două elemente de date.
  5. *Sufixul*/textului care urmează după al doilea dintre cele două elemente de date. Atât prefixul cât și sufixul au fost limitate la 10 caractere.

# Exemplu

□ Un şablon posibil poate consta din următoarele:

**1. Ordinea:** titlul și apoi autorul.

**2. Prefixul URL:** www.stanford.edu/class

**3. Prefixul,** mijlocul și sufixul de forma următoare:

<LI><I>**titlu**</I> de **autor**<P>

Aici **prefixul** este <LI><I>, **mijlocul** este </I> de (inclusiv spațiul de după "de") și **sufixul** este <P>.

Titlul este orice apare între prefix și mijloc; autorul este orice apare între mijloc și sufix.

# Tuning sablon

- Definim ***specificitatea*** unui şablon ca fiind produsul lungimilor prefixului, mijlocului, sufixului și prefixului URL.
- În mare, specificitatea măsoară cât de posibil este să găsim date care corespund şablonului; cu cât specificitatea este mai mare, cu atât ne aşteptăm la mai puţine apariţii ale acestuia în date.

# Tuning sablon

- Şablonul trebuie să îndeplinească două condiții pentru a fi acceptat:
  1. Trebuie să fie cel puțin 2 elemente de date cunoscute care apar conform aceluia şablon.
  2. Produsul specificității şablonului cu numărul de apariții de date conform acestuia trebuie să depășească un anumit prag  $T$  (nespecificat în articolul original).

# Pasii executiei

1. Găsirea aparițiilor pornind de la datele cunoscute
2. Construcția şabloanelor din aparițiile de date
3. Găsirea aparițiilor pornind de la şabloane

# Aparitie

- O apariție a unui tuplu (existent în tabela) constă în:
  1. Un anumit titlu și autor.
  2. Adresa Internet completa (URL) și nu doar prefixul ca în cazul şablonului.
  3. Ordinea, prefixul, mijlocul și sufixul şablonului după care au apărut titlul și autorul respectiv.

# Constructia sabloanelor

1. Se grupează aparițiile de date după ordinea și mijlocul lor. De exemplu, un grup din acest "group-by" poate corespunde ordinii "titlu-apoi-autor" și mijlocului "</I> de ".
2. Pentru fiecare grup se găsește cel mai lung prefix, sufix și prefix URL comun.
3. Dacă testul de specificitate pentru acest şablon este îndeplinit, se acceptă şablonul.

# Constructia sabloanelor

4. Dacă testul de specificitate *nu* este îndeplinit, se încearcă spargerea grupului în două prin extinderea lungimii prefixului URL cu un caracter și apoi se repetă pasul (2). Dacă este imposibil să spargem grupul (pentru că există doar un URL) atunci am eșuat în a produce un nou şablon din acel grup.
- **Exemplu:** Să presupunem că grupul conține trei URL-uri:
- [www.stanford.edu/class/cs345/index.html](http://www.stanford.edu/class/cs345/index.html)
  - [www.stanford.edu/class/cs145/index.html](http://www.stanford.edu/class/cs145/index.html)
  - [www.stanford.edu/class/cs340/readings.html](http://www.stanford.edu/class/cs340/readings.html)

# Constructia sabloanelor

- Prefixul comun este [www.stanford.edu/class/cs](http://www.stanford.edu/class/cs)
- Dacă trebuie să spargem grupul, atunci următorul caracter, 3 sau 1, sparge grupul în două, cu acele apariții ale datelor din prima pagină (pot fi multe astfel de apariții) mergând într-un prim grup și aparițiile din celelalte două pagini în celălalt:
  - [www.stanford.edu/class/cs345/index.html](http://www.stanford.edu/class/cs345/index.html)
  - [www.stanford.edu/class/cs340/readings.html](http://www.stanford.edu/class/cs340/readings.html)

Și

- [www.stanford.edu/class/cs145/index.html](http://www.stanford.edu/class/cs145/index.html)

# Gasire aparitii din sabloane

1. Se găsesc toate URL-urile care se potrivesc cu prefixul URL al cel puțin unui şablon.
2. Pentru fiecare astfel de pagină se parcurge textul folosind o expresie regulată construită din prefixul, mijlocul și sufixul şablonului.
3. Se extrage din fiecare potrivire titlul și autorul, după ordinea specificată în şablon.

# Bibliografie

- J.D.Ullman - CS345 --- Lecture Notes:  
PageRank, Hubs-and-Authorities , Web Mining  
<http://infolab.stanford.edu/~ullman/cs345-notes.html>

# Sfârșitul capitolului 12

# Depozite de date și Modelarea dimensională

# Câteva definiții

- Wikipedia:
  - Un **depozit de date** este locul de stocare al datelor electronice ale unei organizații. Depozitele de date sunt concepute pentru a facilita raportarea și analiza (din cartea lui Inmon spun ei - [1]).
  - Un depozit de date găzduiește o formă standardizată, consistentă, curățată și integrată a datelor provenite din diverse sisteme operaționale utilizate în acea organizație, structurată pentru a aborda în mod specific cerințele de raportare și de analiză.

# Câteva definiții

- R. Kimball (vezi [2, 3]):

Un **depozit de date** este o copie a datelor tranzacționale structurate special pentru interogare și analiză.

- Conform acestei definiții:

- Forma datelor stocate (SGBDR, fișier) nu este legată de definiția unui depozit de date.
- Depozitarea datelor nu este legată exclusiv de „factorii de decizie” sau utilizată doar în procesul de luare a deciziilor.

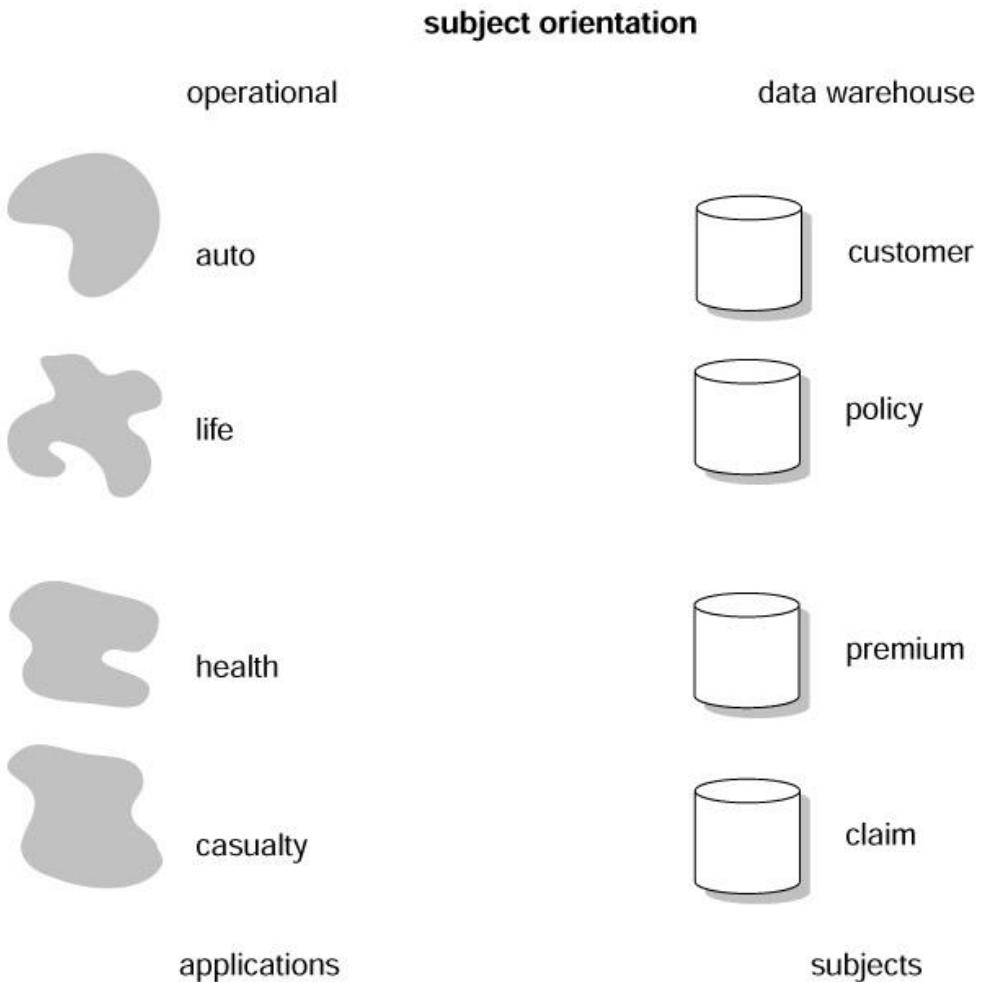
# Câteva definiții

- Inmon (vezi [1, 3]):

Un **depozit de date** este o colecție de date pentru asistarea deciziilor factorilor de conducere care este:

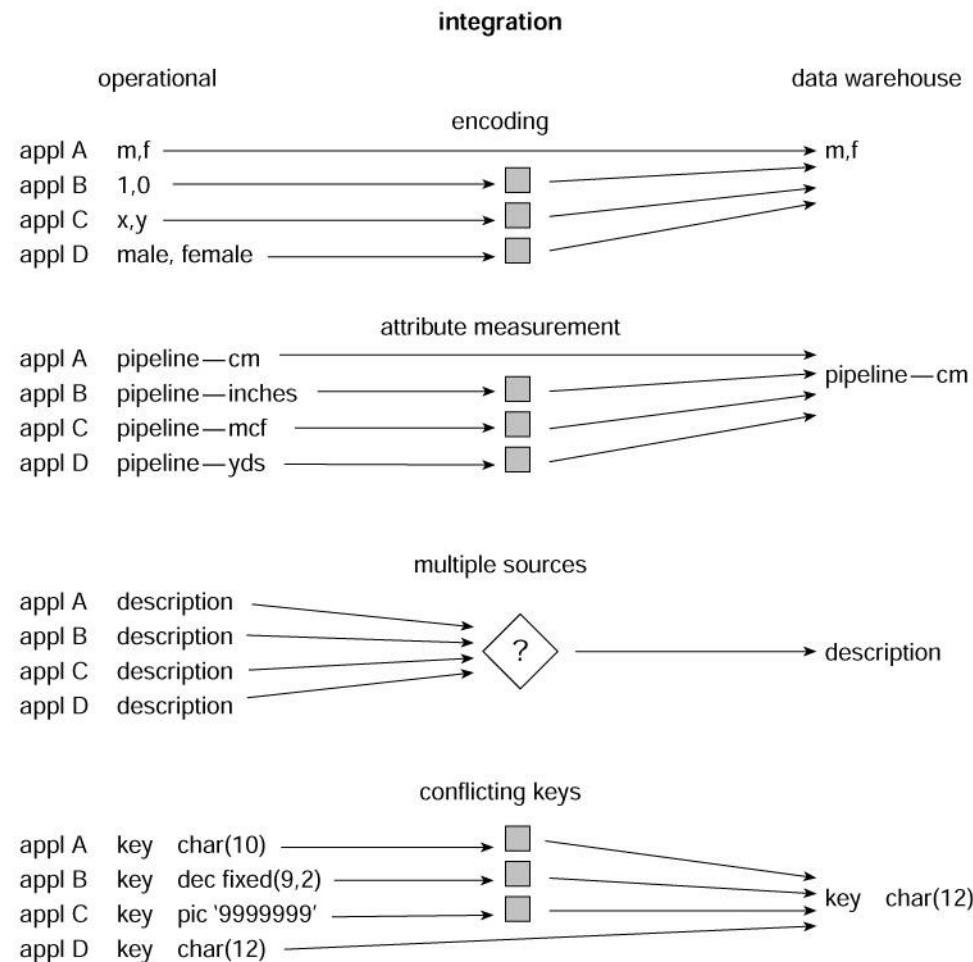
- orientată pe subiecte,
- integrată,
- nevolatilă,
- istorice (time variant)

# Colecție orientată pe subiecte



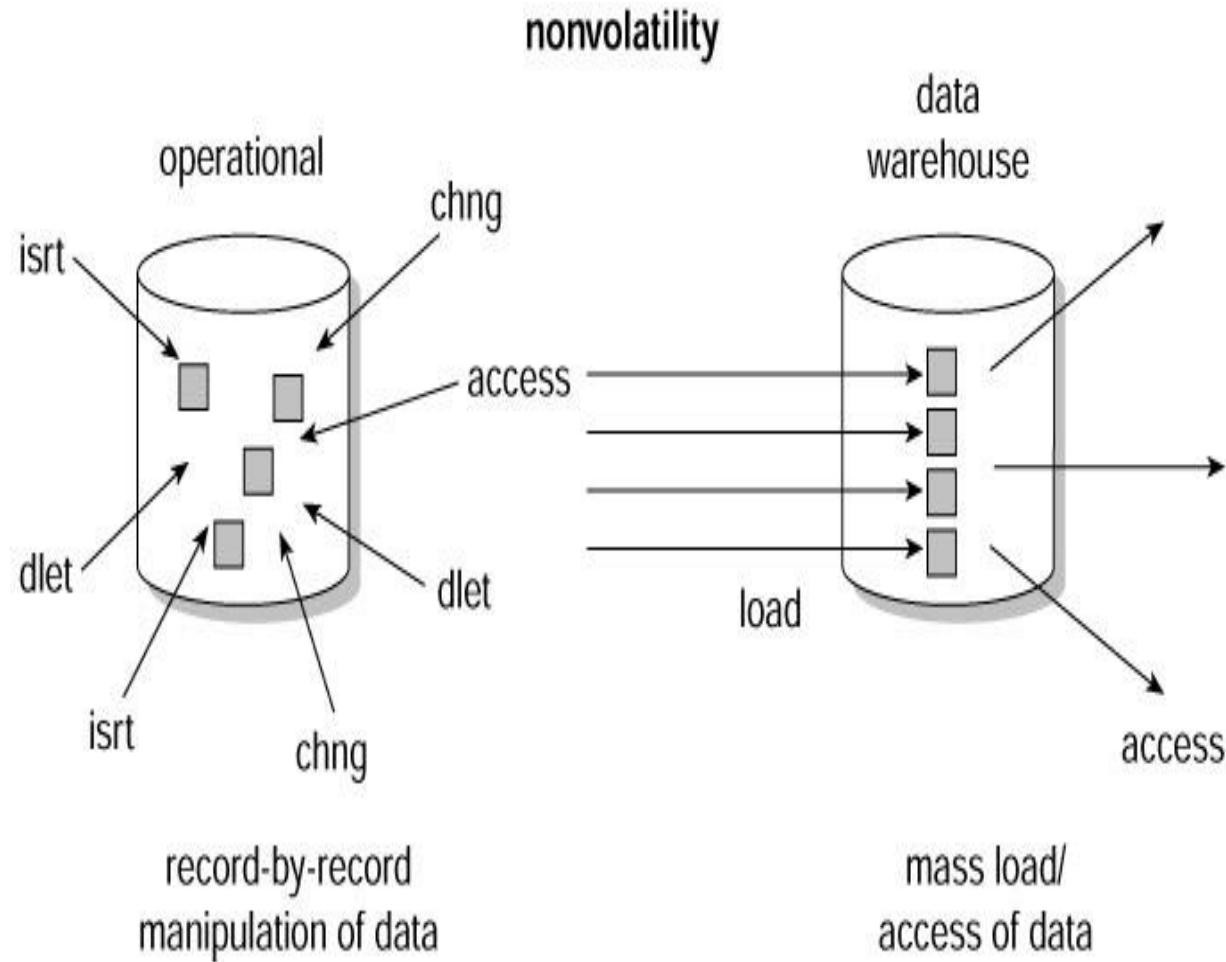
- ❑ Sistemele operaționale clasice sunt organizate în jurul aplicațiilor companiei. De exemplu, pentru o companie de asigurări, aplicațiile pot fi asigurări auto, de sănătate, de viață sau de accidente.
- ❑ Principalele subiecte în acest cay sunt clientul, polița, prima și cererea de despăgubire.
- ❑ Pentru o firmă de producție principalele domenii pot fi produsul, comanda, furnizorul, factura de materiale și materiile prime.
- ❑ Pentru un retailer principalele subiecte pot fi produsul, pachetul, vânzarea, furnizorul și altele. Fiecare tip de companie are propriul său set unic de subiecte

# Colecție integrată



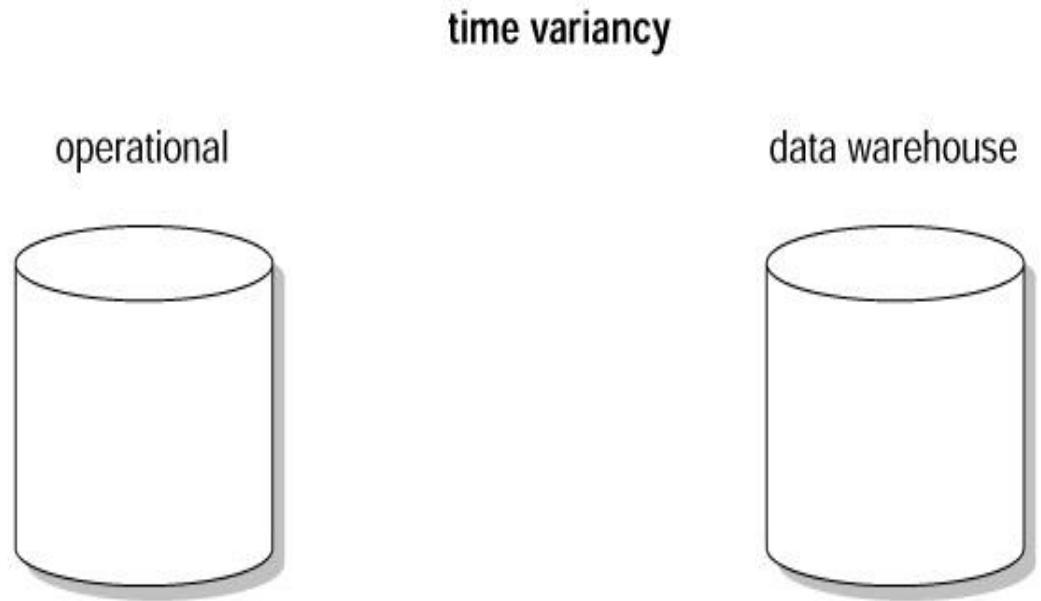
- ❑ Dintre toate aspectele unui depozit de date, integrarea este cea mai importantă.
- ❑ Datele sunt introduse din mai multe surse disparate în depozitul de date.
- ❑ Pe măsură ce datele sunt furnizate, acestea sunt convertite, reformatate, regrupate, rezumate și aşa mai departe.
- ❑ Rezultatul este că datele odată stocate în depozitul de date se comportă ca un întreg.

# Colecție nevolatilă



- ❑ În sistemele operaționale actualizarea datelor este o operație obișnuită dar un depozit de date prezintă un set de caracteristici diferite.
- ❑ Datele din depozitului de date sunt încărcate (de obicei în masă) și accesate, dar nu sunt actualizate (în sensul obișnuit).
- ❑ În schimb datele din depozitul de date sunt încărcate ca un snapshot (vedere statică).
- ❑ Ulterior când apar modificări se încarcă un nou snapshot.
- ❑ În acest fel toată istoria datelor se păstrează în depozitul de date.

# Date istorice (time variant)

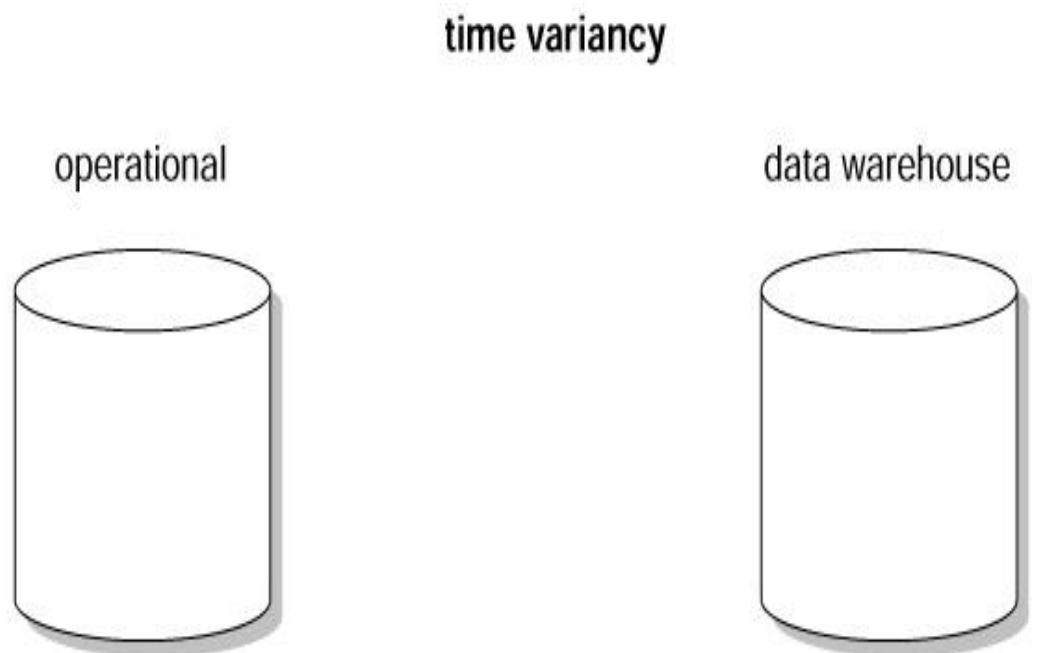


- time horizon—current to 60–90 days
- update of records
- key structure may/may not contain an element of time

- time horizon—5–10 years
- sophisticated snapshots of data
- key structure contains an element of time

- ❑ Istoricitatea datelor presupune că fiecare unitate de date din depozitul de date a fost corecta la un moment dat.
- ❑ În unele cazuri, o înregistrare este stampilată cu o etichetă de timp (timestamp). În alte cazuri înregistrarea are asociată o dată a tranzacției.
- ❑ Există în toate cazurile o anumită formă de marcare a timpului pentru a arăta momentul în care înregistrarea a fost corectă.

# Date istorice (time variant)



- time horizon—current to 60–90 days
- update of records
- key structure may/may not contain an element of time

- time horizon—5–10 years
- sophisticated snapshots of data
- key structure contains an element of time

- Un orizont de timp de 60 până la 90 de zile este normal pentru sistemele operaționale; un orizont de timp între 5 și 10 ani este normal pentru un depozit de date.
- Ca urmare a acestei diferențe de orizonturi de timp, depozitul de date conține mult mai mult istoric decât orice alt mediu.

# Date normalize vs. Modelare dimensională

- ❑ Există două abordări principale pentru stocarea datelor într-un depozit de date:
  - ❑ 1. Abordarea **normalizată** (Inmon - [1])
  - ❑ 2. Abordarea **dimensională** (Kimball - [2])
- ❑ Aceste abordări nu se exclud reciproc și există și alte abordări. Abordările dimensionale pot implica normalizarea datelor într-un anumit grad.
- ❑ Cursul de azi se bazează pe abordarea dimensională și cartea lui Kimball & Ross - vezi [2].

# Date normalize vs. Modelare dimensională

- În abordarea normalizată, datele din depozitul de date sunt stocate urmând, într-o anumită măsură, regulile cunoscute privind normalizarea din domeniul bazelor de date.
- Tabelele sunt grupate pe subiecte care reflectă categoriile generale de date (de exemplu date despre clienți, produse, finanțe etc.).
- Principalul avantaj al acestei abordări este că este simplu să adaugi noi informații în baza de date.

# Date normalize vs. Modelare dimensională

- Un dezavantaj al acestei abordări este faptul că, din cauza numărului mare de tabele implicate, poate fi dificil pentru utilizatori:
  1. Să reunească prin join date din diferite surse pentru a obține informații semnificative și
  2. Să accesese informația fără o înțelegere precisă a surselor de date și a structurii depozitului de date.

# Date normalize vs. Modelare dimensională

- În abordare **dimensională**, datele tranzacționale sunt împărțite în **fapte** sau **măsuri** (date numerice ale tranzacțiilor) și dimensiuni (informații de referință care conferă contextul faptelor).
- De exemplu, o tranzacție de vânzare poate fi defalcată în fapte precum numărul de produse comandate și prețul plătit pentru produse și în dimensiuni precum data comenzi, numele clientului, codul produsului, locația expedierii, locația de facturare și agentul de vânzări responsabil de primirea comenzi.

# Date normalize vs. Modelare dimensională

- Un **avantaj** esențial al unei abordări dimensionale este că depozitul de date este mai ușor de înțeles și de folosit. De asemenea regăsirea datelor în depozitul de date poate să funcționeze foarte repede.
- Principalele **dezavantaje** ale abordării dimensionale sunt:
  1. Pentru a menține integritatea tabelelor de fapte și dimensiuni încărcarea depozitului de date cu date din diferite sisteme operaționale este mai complicată și
  2. Este mai dificil de modificat structura depozitului de date dacă organizația care adoptă abordarea dimensională schimbă modul în care își desfășoară activitatea.

# Modelare dimensională

- Este o modalitate de structurare pentru un depozit de date
- Bazat pe:
  - Tabele de fapte (sau măsuri)
  - Tabele de dimensiuni

# Tabelă de fapte (măsuri)

- Reprezintă un proces de business
- Conține măsurătorile sau valorile sau faptele proceselor de business
  - De exemplu „cantitate” și „preț total” în cazul unei companii de retail
  - Majoritatea atributelor (coloanelor) sunt aditive (vânzări în această lună), unele sunt semi-aditive (sold la data de), altele nu sunt aditive (preț unitar)
- Nivelul de detaliu se numește „granularitatea” tabelului – ce reprezintă o linie din tabela de fapte respectivă.
- Conține chei străine pentru fiecare tabelă de dimensiuni cu care se relaționează.

# Tabelele de dimensiuni

- Reprezintă cine, ce, unde, când și cum pentru o măsură
- Reprezintă entități din lumea reală și nu procese de business
- Dau contextul unei măsuri (subiect)
- De exemplu pentru tabelul de fapte Vânzări, caracteristicile măsurii „cantitate” pot fi o locație (unde), timp (când), produs vândut (ce).
- Atributele dimensiunilor sunt coloanele din respectivul tabel.

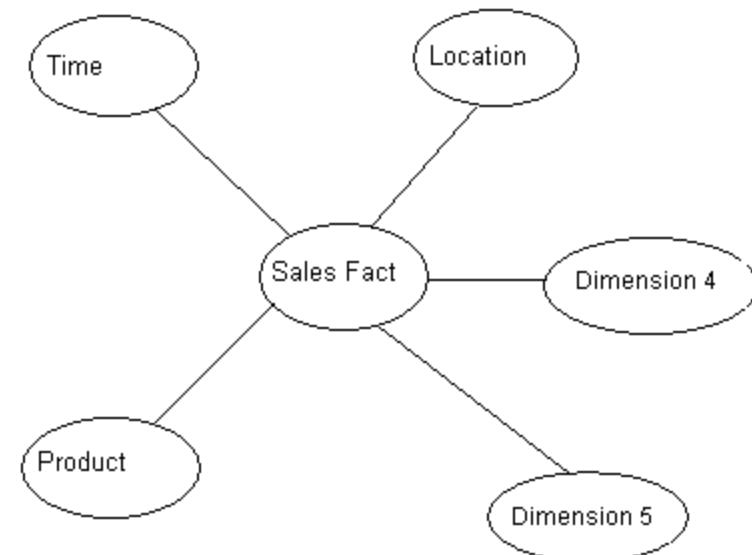
# Tabelele de dimensiuni

- În dimensiunea Locație, atributele pot fi codul locației, județul, țara, codul poștal.
- În general, atributele dimensiunilor sunt utilizate în etichetele rapoartelor și în condiții cum ar fi Tara = 'USA'.
- Înainte de a projecța depozitul de date trebuie decis ce conține acesta. Să zicem că se dorește un depozit de date care să conțină vânzările pe diverse locații ale magazinelor, pe timp (dată) și pe produse. Atunci dimensiunile vor fi: Locație, Dată și Produs.

# Scheme stea

-- Regăsire masuri:

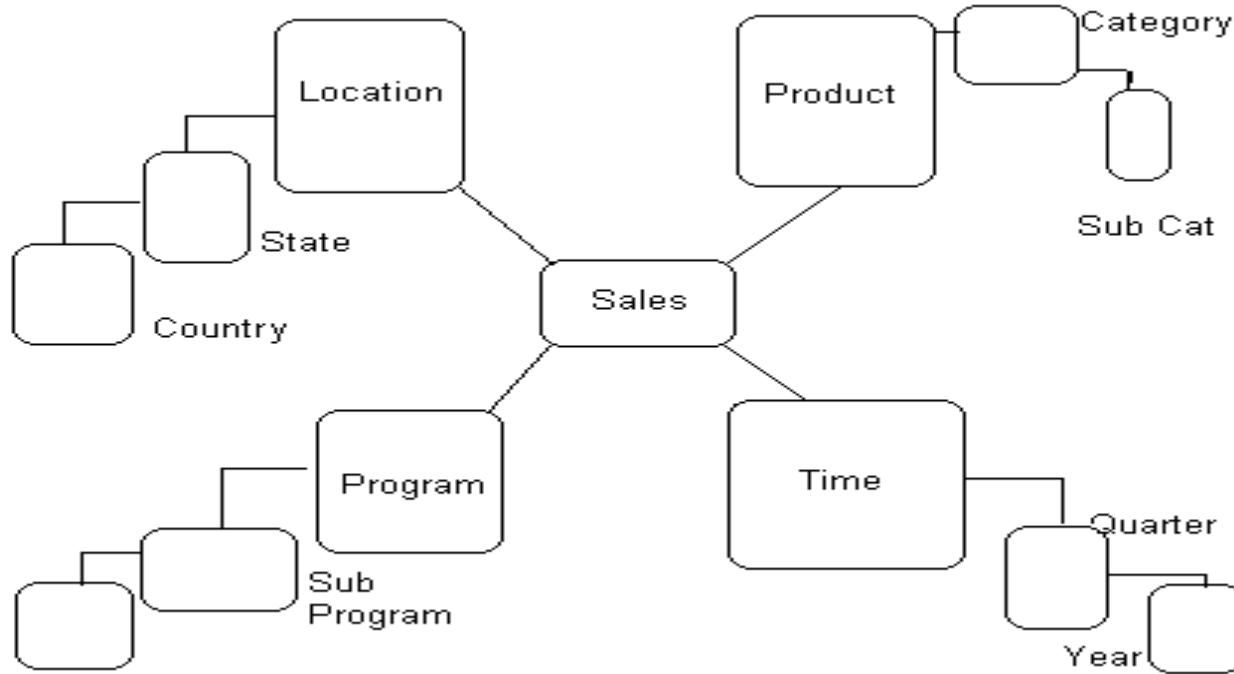
```
SELECT P.Name, SUM(F.Sales)...
-- JOIN între fapte și dimensiuni
FROM Sales F, Time T, Product P,
     Location L
WHERE F.TM_Dim_Id = T.Dim_Id
AND   F.PR_Dim_Id = P.Dim_Id
AND   F.LOC_Dim_Id = L.Dim_Id
-- Constrângeri dimensiuni
AND   T.Month='Jan' AND T.Year='2003'
AND   L.Country_Name='USA'
-- Grupare pentru agregare
GROUP BY P.Category
```



Avantaje:

- ușor de înțeles
- performanță mai bună
- extensibil

# Schema Fulg de nea (snowflake)



Daca nu denormalizăm  
dimensiunile

Regulă generală:  
Nu folosiți!

# Cei patru pași ai modelării dimensionale

Obiectiv: proiectarea unei baze de date dimensionale, luând în considerare patru pași în această ordine:

1. Selectarea procesului de business de modelat.
2. Declararea gradului de detaliere (declare the grain).
3. Stabilirea dimensiunilor pentru tabelele de fapte.
4. Identificarea atributelor tabelelor de fapte.

(Ralph Kimball, Margy Ross - The Data Warehouse Toolkit, Second Edition, Wiley & Sons, 2002, pp.29-65)

# 1. Selectare proces de business

- ❑ Un proces este o activitate naturală de business desfășurată într-o organizație
- ❑ De obicei este susținut de un sistem de colectare a datelor.
- ❑ Exemple de procese de afaceri includ:
  - ❑ Aprovizionare,
  - ❑ Comenzi,
  - ❑ Livrări,
  - ❑ Facturare,
  - ❑ Inventar (stocuri),
  - ❑ Contabilitate

## 2. Declararea gradului de detaliere

- Declararea gradului de detaliere înseamnă specificarea exactă a ceea ce reprezintă o linie din tabela de fapte.
- Aceasta oferă răspunsul la întrebarea „Cum descrieți o singură linie din tabelul de fapte?”

## 2. Declararea gradului de detaliere

Exemple - ce este o linie din tabela de fapte:

- O linie de pe bonul de casă al unui client, generat de POS
- O linie de pe factura primită de la o firmă
- Un permis de îmbarcare pentru a urca într-un zbor
- Valoarea zilnică a stocului pentru fiecare produs dintr-un depozit
- Soldul la sfârșit de lună pentru un cont bancar

## 2. Declararea gradului de detaliere

- O alegere necorespunzătoare a gradului de detaliere va compromite implementarea depozitului de date.
- Declarația gradului de detaliere este un pas critic care nu poate fi făcut ușor.
- Dacă în etapele 3 sau 4 vedem că declarația gradului de detaliere este greșită, trebuie să revenim la pasul 2, să redescoperim corect gradul de detaliere și să revedem pașii 3 și 4 din nou.

### 3. Stabilirea dimensiunilor

- Dacă este clar gradul de detaliere dimensiunile pot fi identificate destul de ușor.
- Ele reprezintă toate descrierile posibile formate din valori singulare ale unei linii din tabela de fapte.

### 3. Stabilirea dimensiunilor

- Exemple de dimensiuni comune:
  - Data,
  - Produs,
  - Client,
  - Tip tranzacție,
  - Stare tranzacție.

## 4. Identificare attribute tabela de fapte

- Faptele sunt determinate răspunzând la întrebarea „Ce măsurăm?”
- Toate faptele/masurile candidate din proiect trebuie să fie adevărate pentru gradul de detaliere stabilit la pasul 2.
- Faptele/masurile care aparțin în mod clar unui alt grad de detaliere trebuie puse într-un tabel de fapte separat.
- Faptele/masurile tipice sunt numerice și aditive: ex. cantitatea comandată sau valoarea totală.

# Studiu de caz: firmă de retail

- Să luăm în considerare un lanț de magazine alimente cu 100 de magazine distribuite pe o zonă de cinci state (USA).
- Fiecare magazin are o gamă completă de departamente, inclusiv alimente, alimente congelate, lactate, carne, produse, panificație, produse florale și auxiliare pentru sănătate / frumusețe.
- Fiecare magazin are aproximativ 60.000 de produse individuale pe rafturile sale.

# Studiu de caz: firmă de retail

- Produsele individuale se numesc **stock keeping units** (SKU).
- Aproximativ 55.000 dintre SKU-uri provin de la producători externi și au coduri de bare înscrise pe pachetul de produse. Aceste coduri de bare se numesc coduri universale de produse (CUP).
- CUP-urile sunt în același nivel cu SKU-urile individuale.
- Fiecare variație de împachetare a unui produs are un CUP separat și, prin urmare, este un SKU separat.

# Studiu de caz: firmă de retail

- Restul de 5.000 de SKU-uri provin din departamente cu produse vrac precum carne, brânzeturi, panificație, flori, etc.
- Deși aceste produse nu au CUP-uri recunoscute la nivel național, lanțul alimentar le atribuie coduri SKU.
- Deoarece lanțul nostru alimentar este automatizat, etichetele generate se lipesc pe multe articolele din aceste departamente.

# Studiu de caz: firmă de retail

- Deși codurile de bare nu sunt CUP-uri, sunt cu siguranță coduri SKU.
- Datele sunt colectate în mai multe locuri din magazin. Unele dintre cele mai utile date sunt colectate la casele de marcat pe măsură ce clienții cumpără produse.
- Sistemul POS se află la ieșirea magazinului alimentar unde este înregistrat conținutul coșului de produse.
- Uşa din spate, în care furnizorii fac livrări, este un alt punct interesant de colectare a datelor.

# Studiu de caz: firmă de retail

- Atât managementul magazinului, cât și departamentul de marketing de la sediul central petrec o mare cantitate de timp cu stabilirea prețurilor și a promoțiilor.
- Promoțiile într-un magazin alimentar includ reduceri temporare de preț, anunțuri în zare, modul de plasare în magazin (inclusiv plasarea pe cap de rând) și cupoane.

# Studiu de caz: firmă de retail

- Cea mai directă și mai eficientă metodă de a crește volumul de vânzări este scădea dramatică a prețului.
  - O reducere de 50 de centi a prețului șervețelelor de hârtie, în special atunci când este cuplată cu un anunț sau o plasare preferențială poate determina creșterea vânzării șervețelelor de hârtie cu un factor de 10.
  - Din păcate, o reducere atât de mare a prețurilor, de obicei, nu este sustenabilă, deoarece șervețele probabil se vând în pierdere.
- 
- Pe baza acestui exemplu vom începe să proiectăm modelul dimensional al datelor.

# 1. Selectare proces de business

- ❑ Primul pas în proiectare este de a decide ce proces (e) de afaceri vor fi modelate, combinând o înțelegere a cerințelor de afaceri cu o înțelegere a datelor disponibile.
- ❑ Primul model dimensional construit ar trebui să fie cel cu cel mai mare impact - ar trebui să răspundă la cele mai presante întrebări de afaceri și să fie ușor accesibil pentru extragerea datelor

# 1. Selectare proces de business

- În studiul de caz de vânzare cu amănuntul, managementul dorește să înțeleagă mai bine procesul de cumpărare al clienților înregistrat de sistemul POS.
- Astfel, procesul de business pe care îl vom modela este cel de vânzări prin POS.
- Aceste date ne vor permite să analizăm ce produse se vând în ce magazine în ce zile în ce condiții promoționale.

## 2. Declararea gradului de detaliere

- De preferat, ar trebui să dezvoltăm modele dimensionale pentru cele mai atomice informații captate de un proces de business .
- Datele atomice sunt cele mai detaliate informații colectate; aceste date nu pot fi divizate în continuare.
- Datele atomice sunt înalt dimensionale - deci este o potrivire perfectă pentru abordarea dimensională.
- Modelul cu date mai puțin detaliate este vulnerabil la solicitările neașteptate ale utilizatorilor de a explora detaliile.

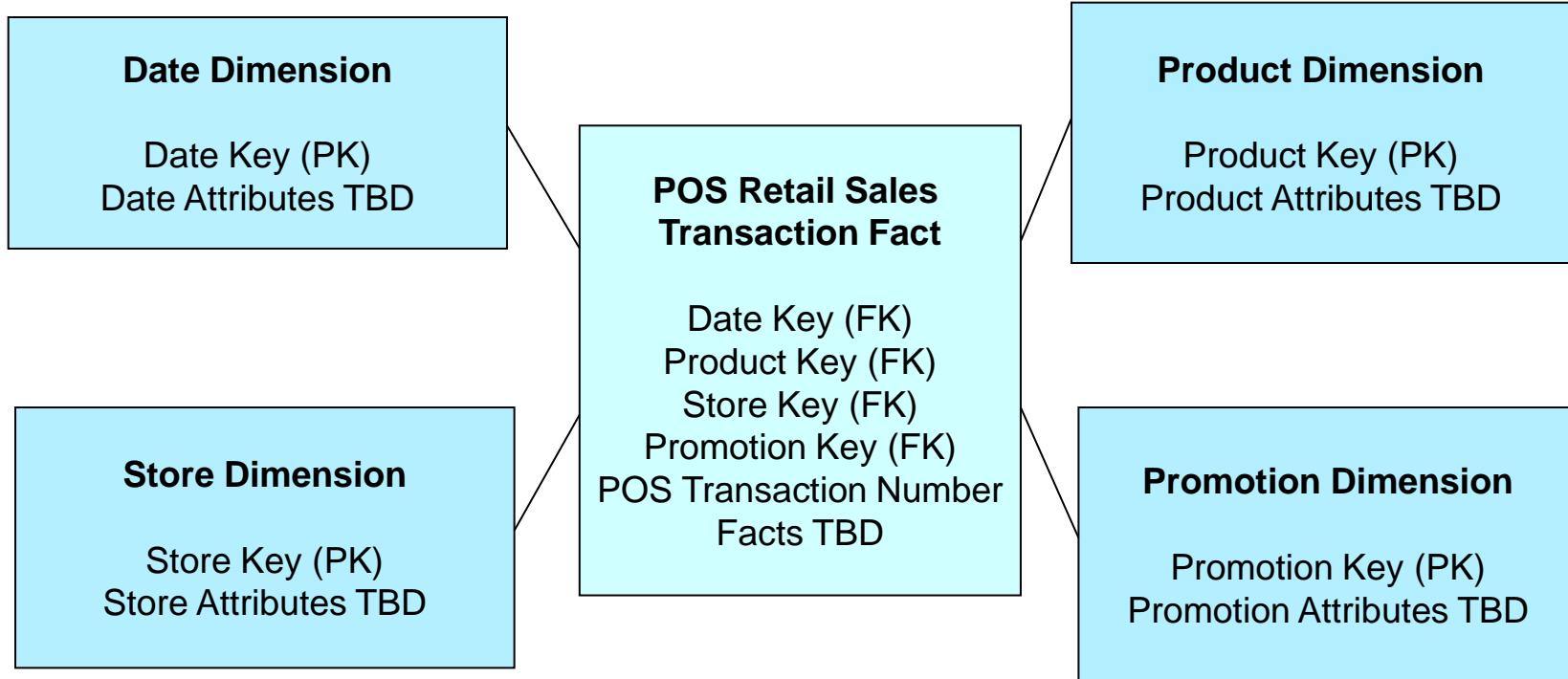
## 2. Declararea gradului de detaliere

- Datele sumare (aggregate: sume, medii, numărări, etc) joacă un rol important ca instrument de ajustare a performanței, dar nu constituie un substitut pentru a oferi utilizatorilor accesul la cele mai mici detalii.
- Un depozit de date necesită aproape întotdeauna date exprimate la cel mai înalt grad de detaliere posibil pe fiecare dimensiune.
- În studiul nostru de caz, cele mai granulare date sunt cele de pe o linie a bonului de casă.

### 3. Stabilirea dimensiunilor

- În studiul nostru de caz identificăm 3 dimensiuni principale: data, produsul și magazinul.
- În plus, putem adăuga noi dimensiuni, ca promoția sub care se vinde produsul.
- În studiul nostru de caz, decidem următoarele dimensiuni descriptive: data, produsul, magazinul și promoția.
- În plus, vom include numărul bonului de casă ca dimensiune specială (descrișă mai târziu)

# 3. Stabilirea dimensiunilor

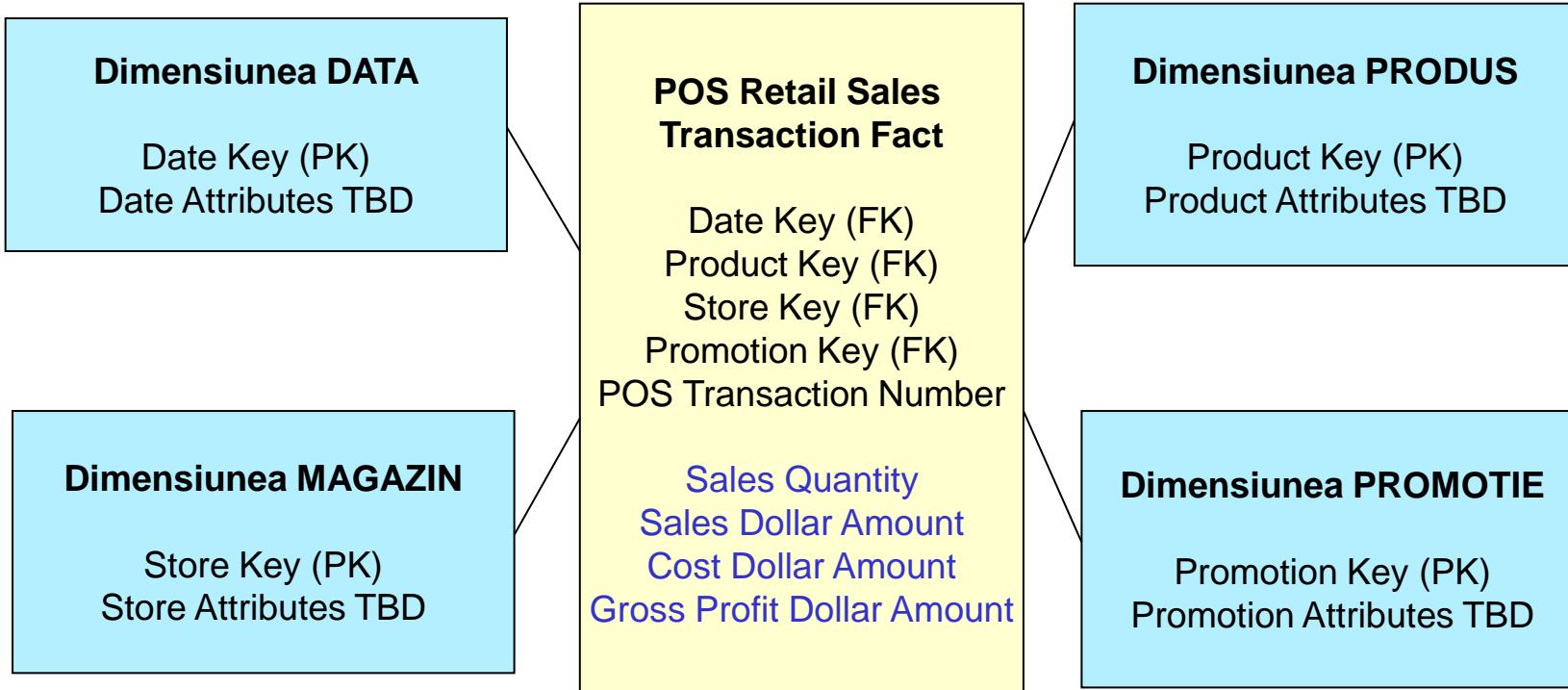


- ❑ TBD – To be determined

## 4. Identificare attribute tabela de fapte

- Faptele trebuie să existe pentru gradul de detaliere ales.  
(în acest caz, linieindividuală din tranzacția POS).
- Faptele colectate de sistemul POS includ **cantitatea**,  
**prețul unitar și valoarea totală** (= cantitatea \* prețul  
unitar).
- Unele sisteme POS oferă un **cost standard** pentru produs,  
și anume cel cu care a fost livrat de furnizor.

# 4. Identificare atribute tabela de fapte



- ❑ TBD – To be determined

## 4. Identificare attribute tabela de fapte

- Profitul brut se obține prin scăderea costului produsului din valoarea totală.
- Toate aceste fapte (cantitatea, suma totală, costul total și profitul brut) sunt aditive pe toate dimensiunile.
- Procente și raporturile, cum ar fi marja brută, sunt nonaditive ( $\text{marja brută} = \text{profitul brut} / \text{valoarea totală}$ )
- Numărătorul și numitorul ar trebui stocate în tabelul de fapte.
- Raportul poate fi calculat la nevoie.
- Prețul unitar este, de asemenea, un fapt nonaditiv (este un raport: valoare totală/cantitate)

# Atribute dimensiuni : Data

- Dimensiunea **dată** este singura dimensiune aproape garantată a fi găsită în fiecare schemă stea deoarece practic aceasta este o serie de timp.
- Dimensiunea **timp** (al zilei) este diferită de cea a datei.

# Atribute dimensiuni: Data

Date Key (PK)	Calendar Quarter
Date	Calendar Year-Quarter
Full Date Description	Calendar Half Year
Day of Week	Calendar Year
Day Number in Epoch	Fiscal Week
Week Number in Epoch	Fiscal Week Number in Year
Month Number in Epoch	Fiscal Month
Day Number in Calendar Month	Fiscal Month Number in Year
Day Number in Calendar Year	Fiscal Year-Month
Day Number in Fiscal Month	Fiscal Quarter
Day Number in Fiscal Year	Fiscal Year-Quarter
Last Day in Week Indicator	Fiscal Half Year
Last Day in Month Indicator	Fiscal Year
Calendar Week Ending Date	Holiday Indicator
Calendar Week Number in Year	Weekday Indicator
Calendar Month Name	Selling Season
Calendar Month Number in Year	Major Event
Calendar Year-Month (YYYY-MM)	SQL Date Stamp

# Atribute dimensiuni: Data

- Indicatorul de zi liberă (holiday indicator ) are doar două valori: vacanță sau non-vacanță.
- Indicatorul zi de lucru (Weekday indicator ) are ca valori Zi a Săptămânii sau a Weekend-ului.
- Coloana sezonului de vânzare este setată la numele sezonului de vânzare cu amănuntul, dacă este cazul. Exemple în Statele Unite ar putea include Crăciun, Ziua Recunoștinței, Paște, Ziua Îndrăgostitilor, 4 iulie sau Niciunul.
- Coloana evenimente majore (major event ) este similară cu coloana sezonului și poate fi folosită pentru a marca evenimente speciale din afara, cum ar fi Super Bowl Sunday sau Labor Strike.

# Atribute dimensiuni: Data.

- Depozitele de date au întotdeauna nevoie de o dimensiune Dată explicită.
- Există multe atrbute ale datei calendaristice care nu sunt calculate de funcții SQL, inclusiv perioadele fiscale, anotimpurile, sărbătorile și weekendurile.
- În loc să încercăm să calculăm aceste informații calendaristice non-standard într-o interogare, ar trebui să le căutăm într-un tabel ('dimensiunea Dată').
- Data și ora sunt aproape complet independente.
- Dacă am combina cele două dimensiuni, dimensiunea ar crește semnificativ.

# Atribute dimensiuni: Data

KeyDate	Date	Full Date Description	Day of Week	Calendar Month	Calendar Year	Fiscal Year-Month	Holiday Indicator	Weekday Indicator
1	01.01.2002	January 1, 2002	Tuesday	January	2002	F2002-01	Holiday	Weekday
2	01.02.2002	January 2, 2002	Wednesday	January	2002	F2002-01	Non-Holiday	Weekday
3	01.03.2002	January 3, 2002	Thursday	January	2002	F2002-01	Non-Holiday	Weekday
4	01.04.2002	January 4, 2002	Friday	January	2002	F2002-01	Non-Holiday	Weekday
5	01.05.2002	January 5, 2002	Saturday	January	2002	F2002-01	Non-Holiday	Weekend
6	01.06.2002	January 6, 2002	Sunday	January	2002	F2002-01	Non-Holiday	Weekend
7	01.07.2002	January 7, 2002	Monday	January	2002	F2002-01	Non-Holiday	Weekday
8	01.08.2002	January 8, 2002	Tuesday	January	2002	F2002-01	Non-Holiday	Weekday

# Atribute dimensiuni: Produs

- Dimensiunea Produs descrie fiecare SKU din magazinul alimentar.
- Un magazin obișnuit din lanț poate stoca 60.000 de SKU-uri, dar atunci când avem în vedere diferite scheme de merchandising din lanț și produse istorice care nu mai sunt disponibile, dimensiunea produsului ar avea cel puțin 150.000 de linii sau mai mult.

# Atribute dimensiuni: Produs

- Product Key (PK)
- Product Description
- SKU Number (Natural Key)
- Brand Description
- Category Description
- Department Description
- Package Type Description
- Package Size
- Fat Content
- Diet Type
- Weight
- Weight Units of Measure
- Storage Type
- Shelf Life Type
- Shelf Width
- Shelf Height
- Shelf Depth

# Atribute dimensiuni: Produs

Product Key	Product Description	Brand Description	Category Description	Department Description	Fat Content
1	Baked Well Light Sourdough Fresh Bread	Baked Well	Bread	Bakery	Reduced Fat
2	Fluffy Sliced Whole Wheat	Fluffy	Bread	Bakery	Regular Fat
3	Fluffy Light Sliced Whole Wheat	Fluffy	Bread	Bakery	Reduced Fat
4	Fat Free Mini Cinnamon Rolls	Light	Sweeten Bread	Bakery	Non-Fat
5	Diet Lovers Vanilla 2 Gallon	Coldpack	Frozen Dessert	Frozen Foods	Non-Fat
6	Light and Creamy Butter Pecan 1 Pint	Freshlike	Frozen Dessert	Frozen Foods	Reduced Fat
7	Chocolate Lovers 1/2 Gallon	Frigid	Frozen Dessert	Frozen Foods	Regular Fat
8	Strawberry Ice Creamy 1 Pint	Icy	Frozen Dessert	Frozen Foods	Regular Fat
9	Icy Ice Cream Sandwiches	Icy	Frozen Dessert	Frozen Foods	Regular Fat

# Atribute dimensiuni: Produs

- Un tabel rezonabil de tip **Produs** are 50 sau mai multe atribute descriptive.
- Fiecare atribut este o sursă bogată pentru construirea rapoartelor.

# Atribute dimensiuni: Magazin

- Dimensiunea **Magazin** descrie fiecare magazin din lanțul nostru.
- Dimensiunea **Magazin** este dimensiunea geografică principală în studiul nostru de caz.
- Fiecare magazin poate fi gândit ca o locație. Din această cauză, putem asocia un magazin cu orice atribut geografic, cum ar fi codul poștal, județul și statul din Statele Unite.
- De obicei, magazinele se plasează în districte și regiuni.

# Atribute dimensiuni: Magazin

- De obicei, magazinele se plasează în districte și regiuni.
- Aceste două ierarhii diferite sunt ambele reprezentate cu ușurință în dimensiunea magazinului, deoarece atât ierarhiile geografice cât și cele regionale ale magazinului sunt bine definite pentru o singura linie din dimensiunea Magazin.
- Nu este neobișnuit să reprezentăm mai multe ierarhii într-un tabel cu dimensiuni. În mod ideal, numele și valorile atributului ar trebui să fie unice pentru ierarhiile multiple.

# Atribute dimensiuni: Magazin

- Store Name
- Store Number (Natural Key)
- Store Street Address
- Store City
- Store County
- Store State
- Store Zip Code
- Store Manager
- Store District
- Store Region
- Floor Plan Type
- Photo Processing Type
- Financial Service Type
- Selling Square Footage
- Total Square Footage
- First Open Date
- Last Remodel Date

# Atribute dimensiuni: Promoție

- Dimensiunea **Promoție** descrie condițiile de promovare în care a fost vândut un produs.
- Condițiile de promovare includ reduceri temporare de preț, plasare preferențiată, anunțuri în ziare și cupoane.
- Această dimensiune este adesea numită dimensiune cauzală - causal dimension (spre deosebire de o dimensiune obișnuită - casual dimension), deoarece descrie factorii gândiți să provoace o schimbare a cifrelor de vânzări.

# Atribute dimensiuni: Promoție

- Promotion Key (PK)
- Promotion Name
- Price Reduction Type
- Promotion Media Type
- Ad Type
- Display Type
- Coupon Type
- Ad Media Name
- Display Provider
- Promotion Cost
- Promotion Begin Date
- Promotion End Date .....

# Număr tranzacție: Dimensiuni degenerate

- Numerele de control operațional, cum ar fi numerele de ordine, numerele facturii și numerele de facturare, de obicei, dau naștere la dimensiuni goale și sunt reprezentate ca dimensiuni degenerate (adică avem chei străine în tabela de fapte fără tabele de dimensiune corespunzătoare).

# Bibliografie

1. W.H. Inmon - Building The Data Warehouse. Third Edition, Wiley & Sons, 2002
2. Ralph Kimball, Margy Ross - The Data Warehouse Toolkit, Second Edition, Wiley & Sons, 2002
3. Dimitra Vista, CS 680 Course notes
4. Wikipedia – Pages on Data Warehouse, etc.
5. <http://searchsqlserver.techtarget.com/>
6. Vincent Rainardi, Building a Data Warehouse with Examples in SQL Server, Springer, 2008