Oracle Tuning – SQL

Em Oracle Enterprise Edition 12.2.0.1 (RU 12.2.0.1.180417)
Em Oracle Enterprise Linux 7.5
Inclui CDB Architecture

Ricardo Portilho Proni
ricardo@nervinformatica.com.br

Esta obra está licenciada sob a licença
Creative Commons Atribuição-SemDerivados 3.0 Brasil.
Para ver uma cópia desta licença, visite
http://creativecommons.org/licenses/by-nd/3.0/br/.

Agenda

- Introdução
- CDB Architecture
- Lendas de SQL
- SQL
- Custo
- Seletividade e Cardinalidade
- Configuração do CBO
- Ferramentas Básicas
- SQL Engine
- Access Paths
- Modelagem
- Query Transformation
- DBMS_SQLTUNE
- Join Methods e Join Options
- Índices
- Estatísticas
- Otimizações
- Extração de Planos de Execução
- Análise de Planos de Execução
- Estabilidade de Plano de Execução
- SQLT

2

Conteúdo

Iremos abordar:

- SQL

Não iremos abordar:

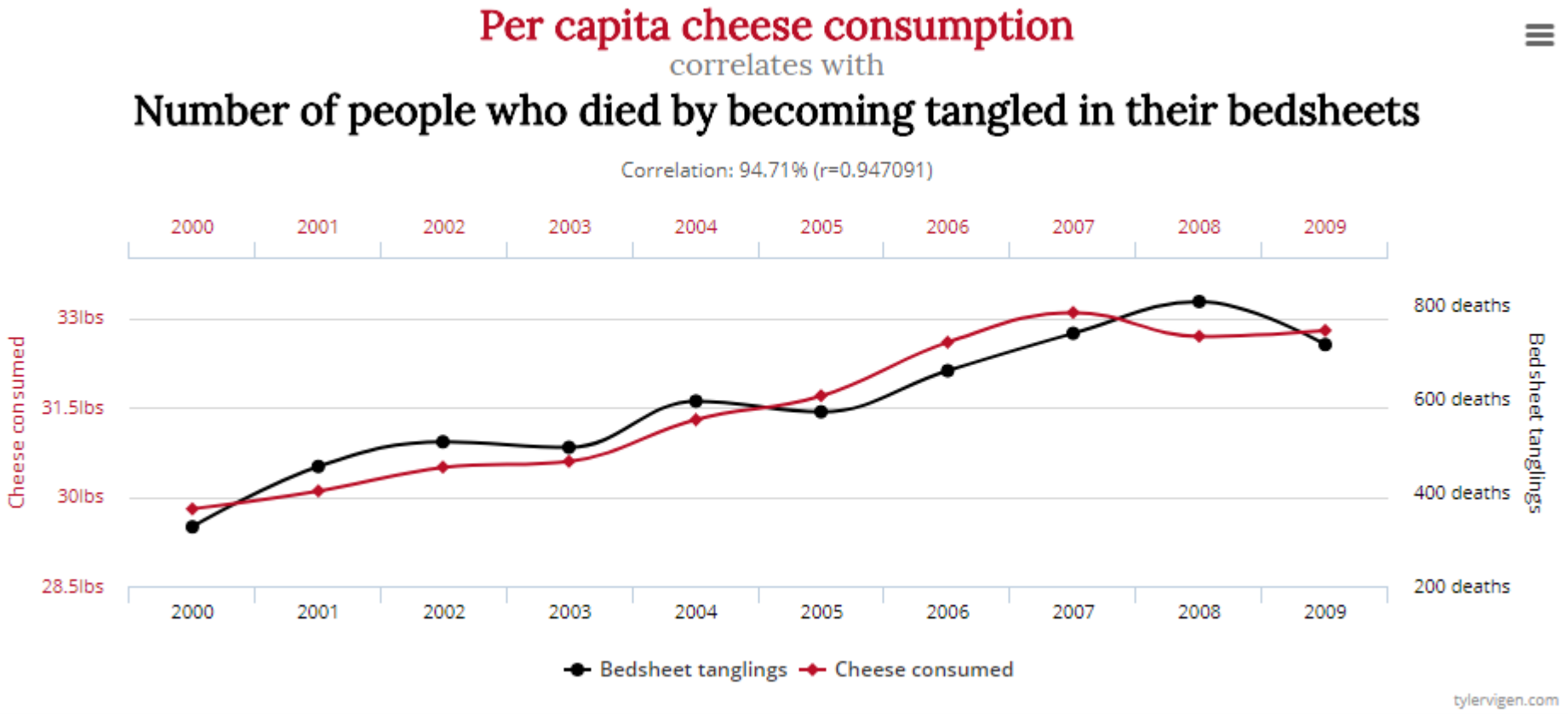
- PL/SQL
- Analytic / OLAP / Data Mining
- LOB / SecureFiles
- XML
- HTML DB / APEX
- Oracle Text
- Oracle Spatial
- Intermedia / Image
- JDBC / OCI
- ODBC / ADO / .NET

3

Isenção de responsabilidade

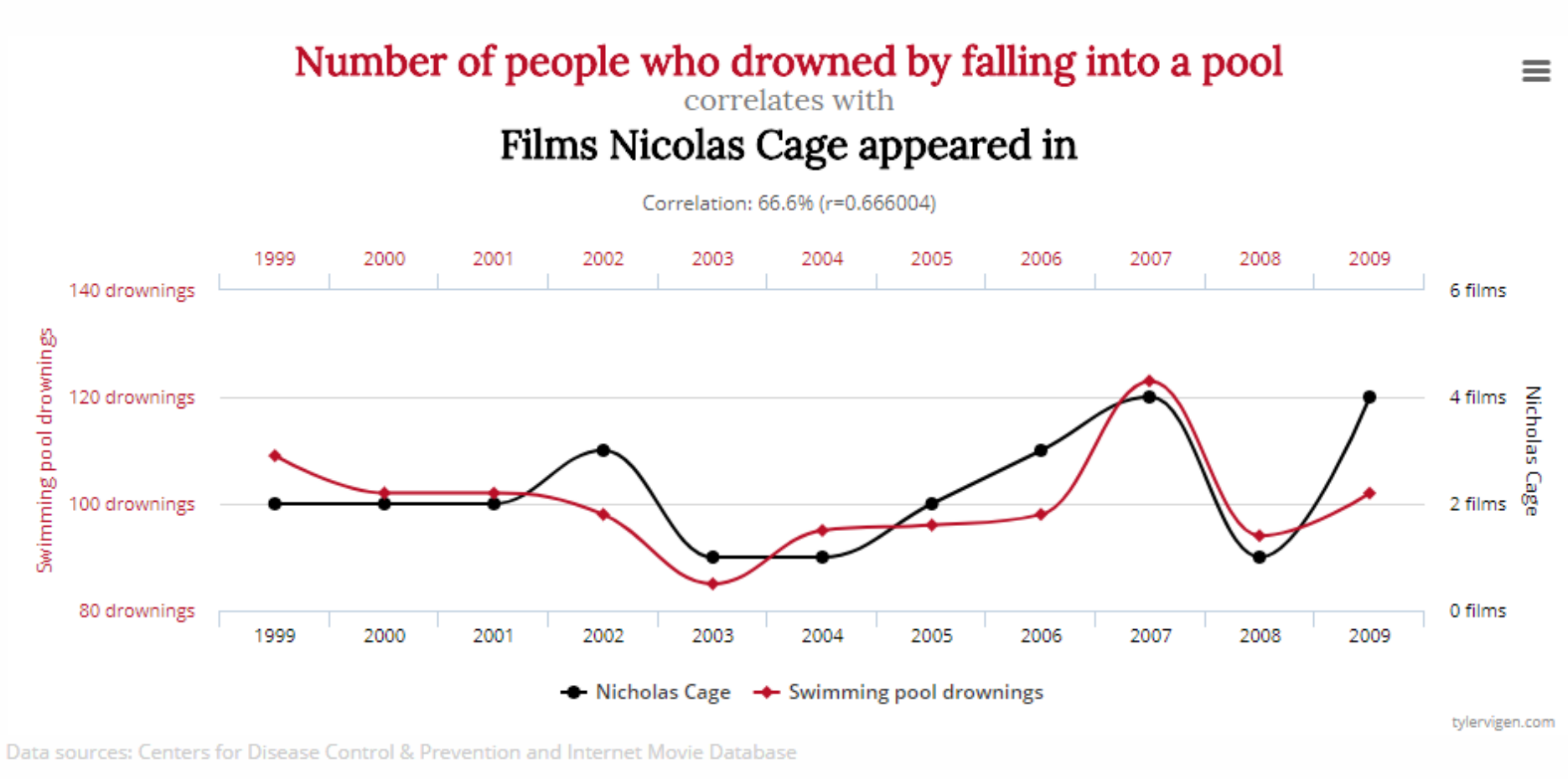
- Não acredite em tudo o que lê. Questione tudo.
- Por algo estar escrito, não significa que é verdade.
- O que é verdade aqui, pode não ser verdade lá.
- O que era verdade ontem, pode não ser verdade hoje.
- O que é verdade hoje, pode não ser verdade amanhã.
- Se os fatos não se adequam à teoria, modifique a teoria.
- Questione, e só acredite em fatos: teste.
- Também tente provar que você está errado.
- Implemente a solução no menor escopo possível.
- Quando você mudar algo, pode acontecer uma de três coisas.

Tendemos a encontrar soluções fáceis



http://www.tylervigen.com/spurious-correlations

Correlação != Causalidade



http://www.tylervigen.com/spurious-correlations

Comandos no Treinamento

Comando com o usuário root:

```
# ls -lh
```

Comando com um usuário administrador:

```
$ sudo ls -lh
```

Comando com um usuário normal:

```
$ ls -lh
```

Adicionar texto a um arquivo:

```
$ vi initORCL.ora
...
db_cache_size=500M
...
```

Comando no Oracle:

```
SQL> SELECT STATUS FROM V$INSTANCE;
```

Algo deve ser alterado de acordo com sua máquina:

Hostname: **nerv01**.localdomain

Quando algo dá errado propositalmente:

O que aconteceu?

rlwrap

```
# wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
# yum -y install epel-release-latest-7.noarch.rpm
# yum -y install rlwrap

$ rlwrap sqlplus / AS SYSDBA
$ rlwrap rman TARGET /
$ rlwrap lsnrctl
$ rlwrap dgmgrl SYS/Nerv2017@ORCL
```

glogin.sql

```
$ cat $ORACLE_HOME/sqlplus/admin/glogin.sql
…
SET PAGESIZE 1000
SET LINESIZE 220
SET TIMING ON
SET TIME ON
SET SQLPROMPT '&_user@&_connect_identifier> '
DEFINE _EDITOR=vi
…
```

```
09:08:44 SYS@ORCL> SELECT 1 FROM DUAL;

         1
----------
         1

Elapsed: 00:00:00.00
09:09:00 SYS@ORCL>
```

Logs

Oracle (11gR2, 12cR1, 12cR2)

```
$ tail -f $ORACLE_BASE/diag/rdbms/<database>/<SID>/trace/alert_<SID>.log
```

CDB Architecture

Deprecation of Non-CDB Architecture

Documentação 12cR1:

8.1.1 Deprecation of Non-CDB Architecture

The non-CDB architecture is deprecated in Oracle Database 12*c*, and may be desupported and unavailable in a release after Oracle Database 12*c* Release 2. Oracle recommends use of the CDB architecture.

Note:

There remain a small number of features that do not work with the CDB architecture (see README, section 2.2.1 "Features Restricted or Not Available for a Multitenant Container Database"). If you need these features, then continue to use the non-CDB architecture until your required feature works with the CDB architecture.

Documentação 12cR2:

Deprecation of Non-CDB Architecture

The non-CDB architecture was deprecated in Oracle Database 12*c*. It can be desupported and unavailable in a release after Oracle Database 12*c* Release 2.

Oracle recommends use of the CDB architecture.

12cR2 CDB New Features

- Subset Standby
- Data Guard Broker PDB Migration or Failover
- Cross-Platform Import of a Pluggable Database into a Multitenant Container Database
- I/O Rate Limits for PDBs
- Heat Map and Automatic Data Optimization Support for CDBs
- PDB Character Set
- PDB Refresh
- Hints in CONTAINERS Query
- Parallel PDB Creation Clause
- PDB Archive Files (.pdb Files)
- Default Tablespace Clause
- Cloning a PDB
- Near Zero Downtime PDB Relocation
- Logical Standby Database to Support CDBs with PDBs with Different Character Sets
- LogMiner to Support CDBs with PDBs with Different Character Sets
- Support for PDBs with Different Character Sets, Time Zone File Versions, and Database Time Zones in a CDB
- Memory Resource Management
- Per-Process PGA Limits
- Performance Profiles and Mandatory PDB Profiles
- CDB-Level PDB Lockdown

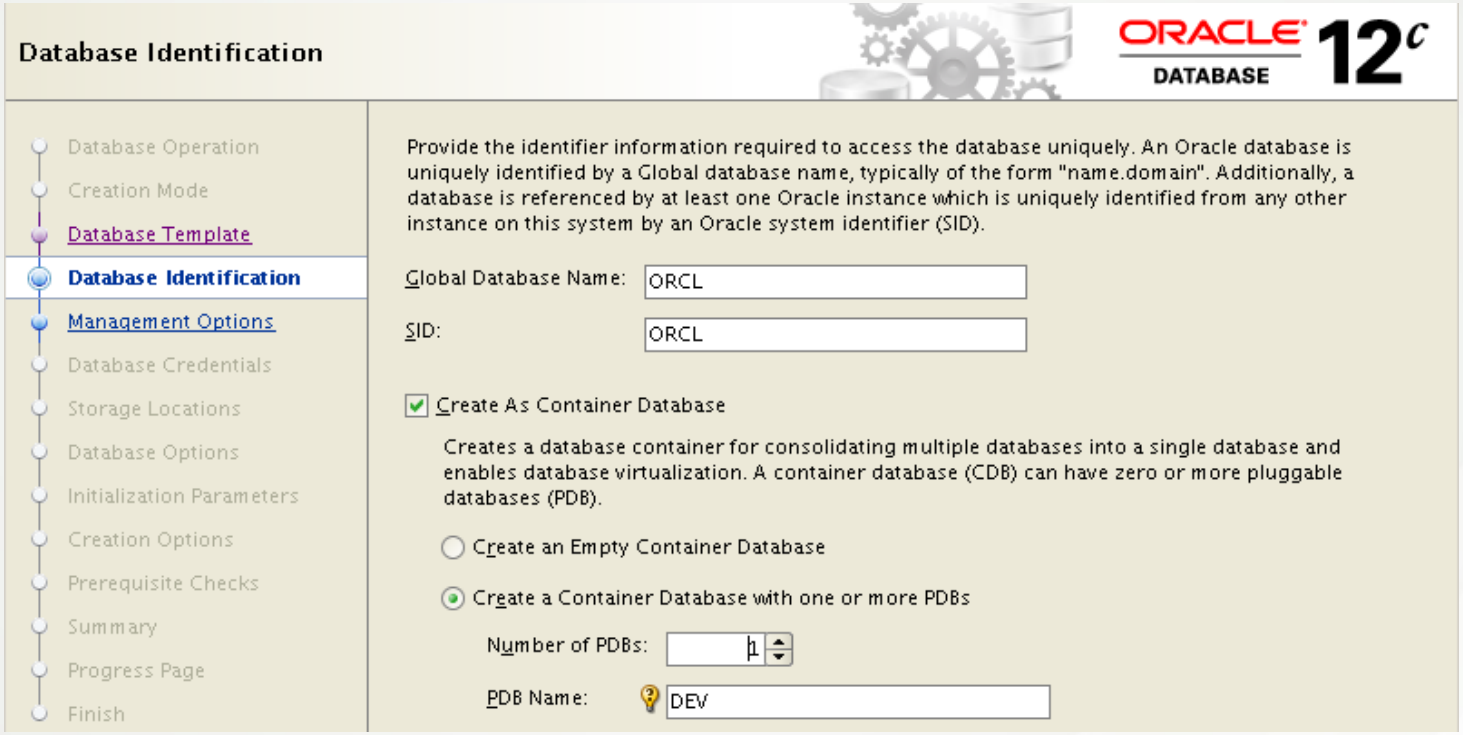
12cR2 CDB New Features

- Application Root
- Proxy PDB
- Forwarding Connections to A New Address Based on Service
- Service-Level ACLs for TCP Protocol
- Flashback Pluggable Database
- Upgrading a CDB With One or More PDBs in a Single Operation
- Support for Thousands of Pluggable Databases for Each Multitenant Container Database
- Pluggable Database Lockdown Profiles Enhancements
- Pluggable Database Operating System Credentials
- Oracle Enterprise Manager Database Express (EM Express) Resource Manager Support
- EM Express: Simplifying Configuration for Multitenant Through Single Port Access
- EM Express: Performance Hub Enhancements
- Automatic Workload Repository (AWR) Support for a Pluggable Database (PDB)
- Selective PDB Upgrades
- AWR_PDB_AUTOFLUSH_ENABLED Initialization Parameter
- ENABLE_AUTOMATIC_MAINTENANCE_PDB Initialization Parameter
- AUTOTASK_MAX_ACTIVE_PDBS Initialization Parameter
- Global and Shared Connection Pool for Oracle Cloud and Multitenant Oracle Databases
- Prioritized Rebalancing
- Oracle Database Vault Common Realms and Command Rules for Oracle Multitenant

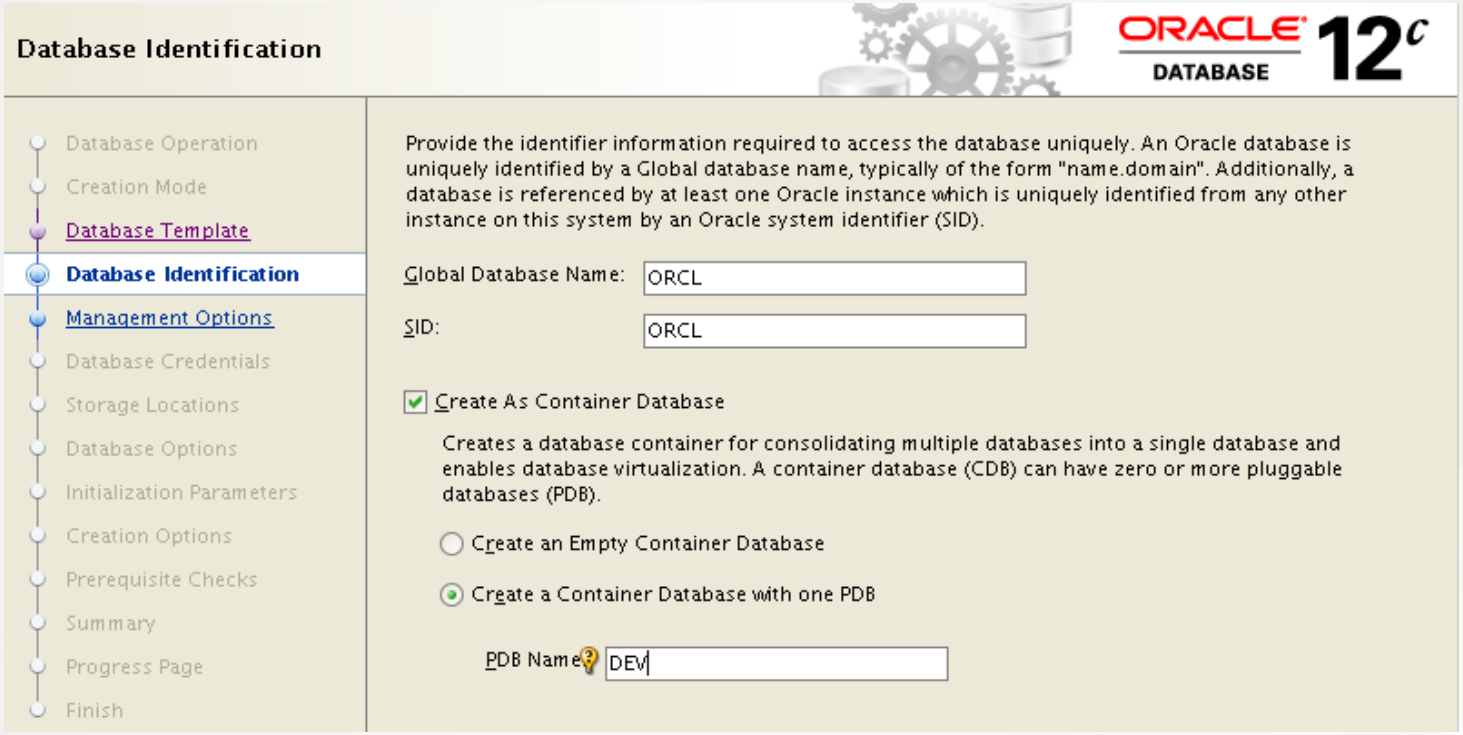
Option Multitenant

	Named User Plus	Software Update License & Support	Processor License	Software Update License & Support
Database Products				
Oracle Database				
Standard Edition 2	350	77.00	17,500	3,850.00
Enterprise Edition	950	209.00	47,500	10,450.00
Personal Edition	460	101.20	-	-
Mobile Server	-	-	23,000	5,060.00
NoSQL Database Enterprise Edition	200	44	10,000	2,200.00
Enterprise Edition Options:				
Multitenant	350	77.00	17,500	3,850.00
Real Application Clusters	460	101.20	23,000	5,060.00
Real Application Clusters One Node	200	44.00	10,000	2,200.00
Active Data Guard	230	50.60	11,500	2,530.00
Partitioning	230	50.60	11,500	2,530.00
Real Application Testing	230	50.60	11,500	2,530.00
Advanced Compression	230	50.60	11,500	2,530.00
Advanced Security	300	66.00	15,000	3,300.00
Label Security	230	50.60	11,500	2,530.00
Database Vault	230	50.60	11,500	2,530.00

Multitenant - EE



Singletenant - SE2



Esse banco utiliza CDB Arquitecture?

```
08:27:38 SYS@ORCL> SELECT CDB FROM V$DATABASE;

CDB
---
YES

Elapsed: 00:00:00.01
08:27:45 SYS@ORCL>
```

Que Containers / PDBs este banco tem?

```
09:26:03 SYS@ORCL> SELECT CON_ID, NAME FROM V$CONTAINERS;

    CON_ID NAME
---------- ------------------------------
         1 CDB$ROOT
         2 PDB$SEED
         3 PROD

Elapsed: 00:00:00.00
09:26:05 SYS@ORCL>
```

```
08:31:04 SYS@ORCL> SELECT CON_ID, NAME FROM V$PDBS;

    CON_ID NAME
---------- ------------------------------
         2 PDB$SEED
         3 PROD

Elapsed: 00:00:00.00
```

Em que Container estou conectado?

```
[oracle@nerv06 ~]$ rlwrap sqlplus / AS SYSDBA

SQL*Plus: Release 12.2.0.1.0 Production on Sun Jul 23 09:09:53 2017

Copyright (c) 1982, 2016, Oracle.  All rights reserved.


Connected to:
Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production

09:09:53 SYS@ORCL> SHOW CON_ID

CON_ID
------------------------------
1
09:09:58 SYS@ORCL> SHOW CON_NAME

CON_NAME
------------------------------
CDB$ROOT
09:10:00 SYS@ORCL>
```

Como conectar em um PDB? SET CONTAINER

```
Connected to:
Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production

09:12:25 SYS@ORCL> ALTER SESSION SET CONTAINER = PROD;

Session altered.

Elapsed: 00:00:00.06
09:12:30 SYS@ORCL> SHOW CON_ID

CON_ID
------------------------------
3
09:12:33 SYS@ORCL> SHOW CON_NAME

CON_NAME
------------------------------
PROD
09:12:37 SYS@ORCL>
```

Como conectar em um PDB?

```
$ lsnrctl status
...
Services Summary...
Service "5397376af7702ac5e0536a00a8c0505d" has 1 instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this service...
Service "ORCL" has 1 instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this service...
Service "ORCLXDB" has 1 instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this service...
Service "prod" has 1 instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this service...
The command completed successfully
$
```

Como conectar em um PDB? tnsnames.ora

```
$ cat $ORACLE_HOME/network/admin/tnsnames.ora
...
ORCL =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = nerv06.localdomain)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = ORCL)
    )
  )

PROD =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = nerv06.localdomain)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = PROD)
    )
  )

...
$
```

Como conectar em um PDB? tnsnames.ora

```
[oracle@nerv06 ~]$ sqlplus SYSTEM/Nerv2017@PROD

SQL*Plus: Release 12.2.0.1.0 Production on Sun Jul 23 09:17:54 2017

Copyright (c) 1982, 2016, Oracle.  All rights reserved.

Last Successful login time: Mon Jul 10 2017 10:34:33 -03:00

Connected to:
Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production

09:17:54 SYSTEM@PROD> SHOW CON_NAME

CON_NAME
------------------------------
PROD
09:18:00 SYSTEM@PROD>
```

Como conectar em um PDB? Easy Connect

```
[oracle@nerv06 ~]$ rlwrap sqlplus SYSTEM/Nerv2017@localhost:1521/PROD

SQL*Plus: Release 12.2.0.1.0 Production on Sun Jul 23 09:22:48 2017

Copyright (c) 1982, 2016, Oracle.  All rights reserved.

Last Successful login time: Sun Jul 23 2017 09:20:43 -03:00

Connected to:
Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production

09:22:48 SYSTEM@localhost:1521/PROD> SHOW CON_NAME

CON_NAME
------------------------------
PROD
09:22:51 SYSTEM@localhost:1521/PROD>
```

E agora, qual o nome da Instância? E do Banco?

```
09:28:20 SYS@ORCL> SHOW CON_NAME

CON_NAME
------------------------------
CDB$ROOT
09:28:24 SYS@ORCL> SELECT INSTANCE_NAME FROM V$INSTANCE;

INSTANCE_NAME
----------------
ORCL

Elapsed: 00:00:00.00
09:28:28 SYS@ORCL> SELECT NAME FROM V$DATABASE;

NAME
---------------------------------------
ORCL

Elapsed: 00:00:00.00
09:28:31 SYS@ORCL>
```

E agora, qual o nome da Instância? E do Banco?

```
09:30:48 SYS@ORCL> SHOW CON_NAME

CON_NAME
------------------------------
PROD
09:30:50 SYS@ORCL> SELECT INSTANCE_NAME FROM V$INSTANCE;

INSTANCE_NAME
----------------
ORCL

Elapsed: 00:00:00.06
09:30:56 SYS@ORCL> SELECT NAME FROM V$DATABASE;

NAME
---------------------------------------
ORCL

Elapsed: 00:00:00.00
09:30:58 SYS@ORCL>
```

CDB e PDBs: STARTUP

```
ORACLE instance started.

Total System Global Area 2147483648 bytes
Fixed Size                  8794848 bytes
Variable Size             603983136 bytes
Database Buffers         1526726656 bytes
Redo Buffers                7979008 bytes
Database mounted.
Database opened.
10:04:30 SYS@ORCL> ALTER SESSION SET CONTAINER = PROD;

Session altered.

Elapsed: 00:00:00.00
10:04:42 SYS@ORCL> SELECT COUNT(*) FROM DBA_TABLES;
SELECT COUNT(*) FROM DBA_TABLES
                     *
ERROR at line 1:
ORA-01219: database or pluggable database not open: queries allowed on fixed table
or views only


Elapsed: 00:00:00.02
10:04:51 SYS@ORCL>
```

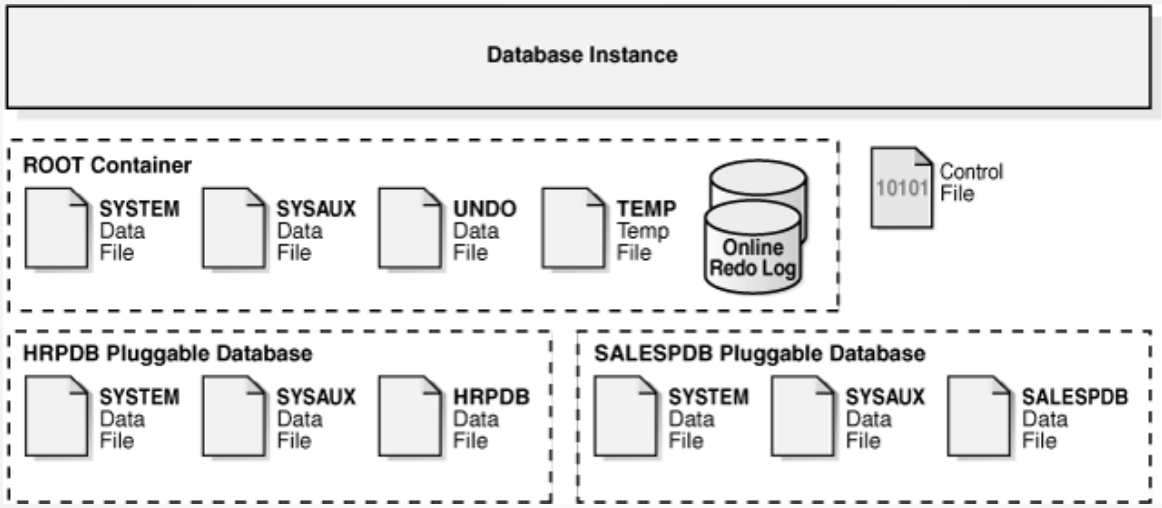
CDB e PDBs: STARTUP

```
10:09:03 SYS@ORCL> ALTER PLUGGABLE DATABASE PROD OPEN READ WRITE;

Pluggable database altered.

Elapsed: 00:00:02.01
10:09:07 SYS@ORCL> ALTER PLUGGABLE DATABASE PROD SAVE STATE;

Pluggable database altered.

Elapsed: 00:00:00.08
10:09:13 SYS@ORCL> ALTER SESSION SET CONTAINER = PROD;

Session altered.

Elapsed: 00:00:00.03
10:09:18 SYS@ORCL> SELECT COUNT(*) FROM DBA_TABLES;

  COUNT(*)
----------
      1768

Elapsed: 00:00:00.22
10:09:21 SYS@ORCL>
```

Alert Log

```
2017-07-23T11:39:06.237614-03:00
ALTER SYSTEM SET workarea_size_policy='AUTO' SCOPE=BOTH;
2017-07-23T11:39:18.790181-03:00
PROD(3):ALTER SYSTEM SET workarea_size_policy='MANUAL' SCOPE=BOTH PDB='PROD';
2017-07-23T11:39:23.868266-03:00
DEV(5):ALTER SYSTEM SET workarea_size_policy='MANUAL' SCOPE=BOTH PDB='DEV';
```

DATAFILEs


Database Instance
ROOT Container
SYSTEM Data File
SYSAUX Data File
UNDO Data File
TEMP Temp File
Online Redo Log
10101
Control File
HRPDB Pluggable Database
SYSTEM Data File
SYSAUX Data File
HRPDB Data File
SALESPDB Pluggable Database
SYSTEM Data File
SYSAUX Data File
SALESPDB Data File

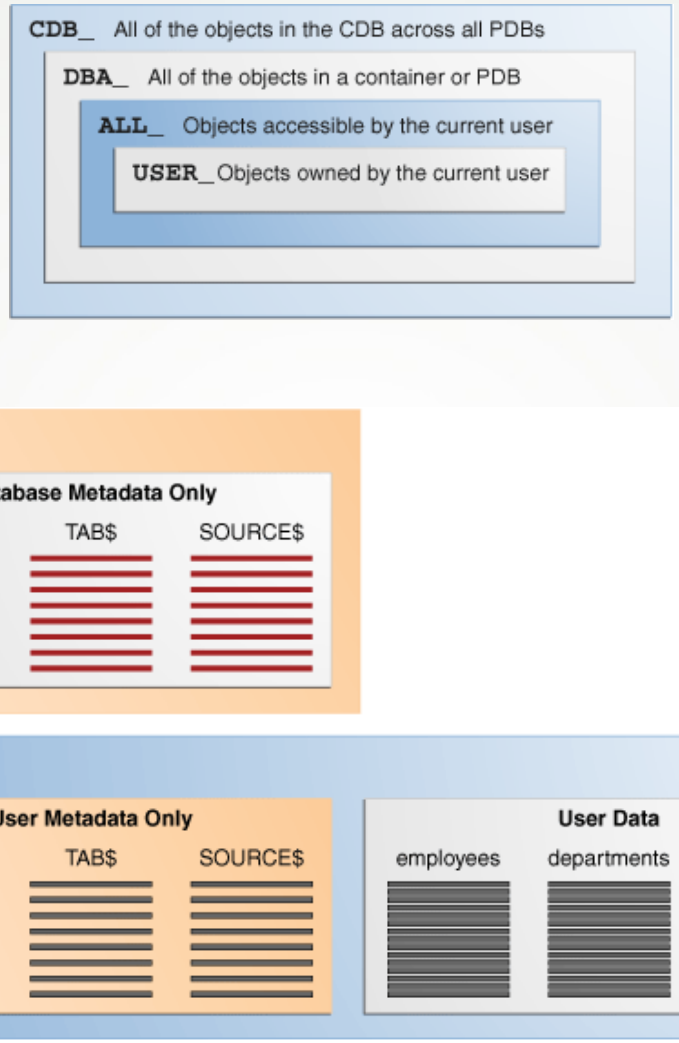
DATAFILEs

```
09:32:59 SYS@ORCL> SHOW CON_NAME

CON_NAME
------------------------------
CDB$ROOT
09:33:04 SYS@ORCL> SELECT FILE_NAME FROM DBA_DATA_FILES;

FILE_NAME
----------------------------------------------------
/u01/app/oracle/oradata/ORCL/system01.dbf
/u01/app/oracle/oradata/ORCL/sysaux01.dbf
/u01/app/oracle/oradata/ORCL/undotbs01.dbf
/u01/app/oracle/oradata/ORCL/users01.dbf

Elapsed: 00:00:00.01
09:33:06 SYS@ORCL>
```

DATAFILEs

```
09:34:17 SYS@ORCL> SHOW CON_NAME

CON_NAME
------------------------------
PROD
09:34:19 SYS@ORCL> SELECT FILE_NAME FROM DBA_DATA_FILES;

FILE_NAME
--------------------------------------------------
/u01/app/oracle/oradata/ORCL/PROD/undotbs01.dbf
/u01/app/oracle/oradata/ORCL/PROD/sysaux01.dbf
/u01/app/oracle/oradata/ORCL/PROD/users01.dbf
/u01/app/oracle/oradata/ORCL/PROD/system01.dbf
/u01/app/oracle/oradata/ORCL/PROD/soe01.dbf
/u01/app/oracle/oradata/ORCL/PROD/shsb01.dbf
/u01/app/oracle/oradata/ORCL/PROD/shsbp01.dbf

7 rows selected.

Elapsed: 00:00:00.00
09:34:20 SYS@ORCL>
```

Dicionário de Dados



VIEWs

```
09:36:19 SYS@ORCL> SHOW CON_NAME

CON_NAME
------------------------------
CDB$ROOT
09:36:23 SYS@ORCL> SELECT COUNT(*) FROM DBA_TABLES;

  COUNT(*)
----------
      1724

Elapsed: 00:00:00.24
09:36:31 SYS@ORCL> SELECT COUNT(*) FROM CDB_TABLES;

  COUNT(*)
----------
      3492

Elapsed: 00:00:00.52
09:36:35 SYS@ORCL>
```

VIEWs

```
09:37:29 SYS@ORCL> ALTER SESSION SET CONTAINER = PROD;

Session altered.

Elapsed: 00:00:00.01
09:37:33 SYS@ORCL> SELECT COUNT(*) FROM DBA_TABLES;

  COUNT(*)
----------
      1768

Elapsed: 00:00:00.18
09:37:36 SYS@ORCL> SELECT COUNT(*) FROM CDB_TABLES;

  COUNT(*)
----------
      1768

Elapsed: 00:00:01.50
09:37:40 SYS@ORCL>
```

Parâmetros

```
10:19:26 SYS@ORCL> SELECT COUNT(*) FROM V$PARAMETER;

  COUNT(*)
----------
       417

Elapsed: 00:00:00.03
10:19:33 SYS@ORCL> SELECT COUNT(NAME) FROM V$PARAMETER
10:19:39   2  WHERE ISPDB_MODIFIABLE = 'TRUE';

COUNT(NAME)
-----------
        160

Elapsed: 00:00:00.03
10:19:48 SYS@ORCL>
```

Parâmetros

```
10:22:17 SYS@ORCL> SHOW CON_NAME

CON_NAME
------------------------------
CDB$ROOT
10:22:21 SYS@ORCL> SHOW PARAMETER WORKAREA_SIZE_POLICY

NAME                                 TYPE        VALUE
------------------------------------ ----------- -------
workarea_size_policy                 string      AUTO
10:22:22 SYS@ORCL> ALTER SESSION SET CONTAINER = PROD;

Session altered.

Elapsed: 00:00:00.00
10:22:27 SYS@ORCL> SHOW PARAMETER WORKAREA_SIZE_POLICY

NAME                                 TYPE        VALUE
------------------------------------ ----------- -------
workarea_size_policy                 string      MANUAL
10:22:29 SYS@ORCL>
```

TABLESPACEs

```
10:00:48 SYS@ORCL> SHOW CON_NAME

CON_NAME
------------------------------
CDB$ROOT
10:00:55 SYS@ORCL> CREATE TABLESPACE T DATAFILE '/u01/users01-CDB.dbf' SIZE 100M;

Tablespace created.

Elapsed: 00:00:02.86
10:01:03 SYS@ORCL> ALTER SESSION SET CONTAINER = PROD;

Session altered.

Elapsed: 00:00:00.00
10:01:07 SYS@ORCL> CREATE TABLESPACE T DATAFILE '/u01/users01-PROD.dbf' SIZE 100M;

Tablespace created.

Elapsed: 00:00:02.49
10:01:16 SYS@ORCL>
```

USERs

```
10:24:21 SYS@ORCL> SHOW CON_NAME

CON_NAME
------------------------------
CDB$ROOT
10:24:27 SYS@ORCL> CREATE USER RICARDO IDENTIFIED BY Nerv2017;
CREATE USER RICARDO IDENTIFIED BY Nerv2017
            *
ERROR at line 1:
ORA-65096: invalid common user or role name


Elapsed: 00:00:00.02
10:24:39 SYS@ORCL> CREATE USER C##RICARDO IDENTIFIED BY Nerv2017;

User created.

Elapsed: 00:00:00.49
10:24:45 SYS@ORCL>
```

USERs

```
10:34:02 SYS@ORCL> SHOW CON_NAME

CON_NAME
------------------------------
CDB$ROOT
10:34:05 SYS@ORCL> GRANT CREATE SESSION TO C##RICARDO;

Grant succeeded.

Elapsed: 00:00:00.03
10:34:07 SYS@ORCL> CONN C##RICARDO/Nerv2017@PROD
ERROR:
ORA-01045: user C##RICARDO lacks CREATE SESSION privilege; logon denied


Warning: You are no longer connected to ORACLE.
10:34:11 SYS@ORCL>
```

USERs

```
10:35:20 SYS@ORCL> SHOW CON_NAME

CON_NAME
------------------------------
PROD
10:35:23 SYS@ORCL> GRANT CREATE SESSION TO C##RICARDO;

Grant succeeded.

Elapsed: 00:00:00.12
10:35:30 SYS@ORCL> CONN C##RICARDO/Nerv2017@PROD
Connected.
10:35:34 C##RICARDO@PROD>
```

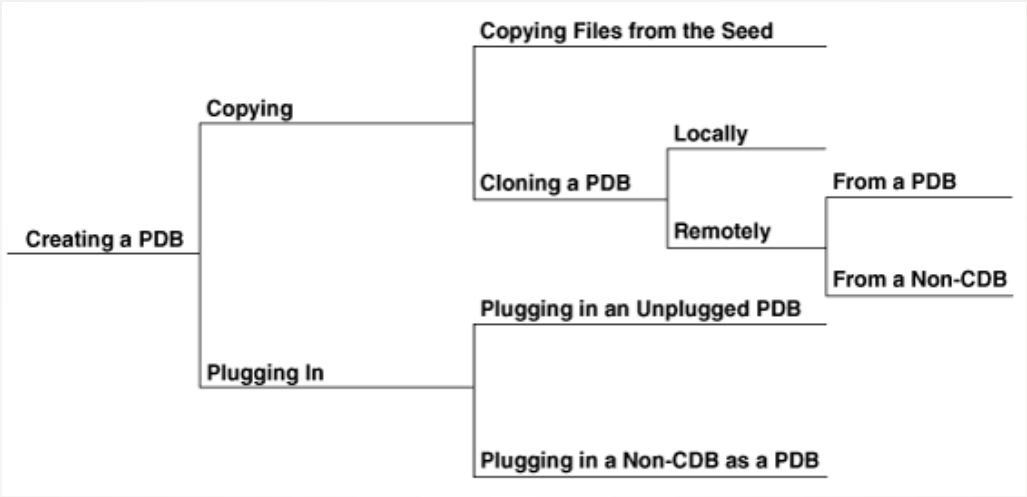
USERs

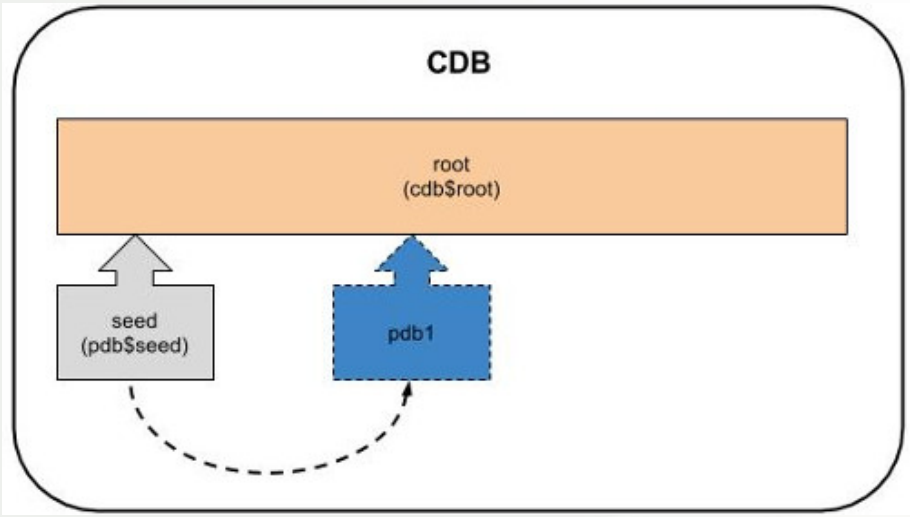
```
11:43:38 SYS@ORCL> ALTER SESSION SET CONTAINER = PROD;

Session altered.

Elapsed: 00:00:00.00
11:43:45 SYS@ORCL> CREATE USER TESTE IDENTIFIED BY Nerv2017;

User created.

Elapsed: 00:00:00.18
11:44:01 SYS@ORCL> ALTER SESSION SET CONTAINER = DEV;

Session altered.

Elapsed: 00:00:00.00
11:44:04 SYS@ORCL> CREATE USER TESTE IDENTIFIED BY Nerv2017;

User created.

Elapsed: 00:00:00.19
11:44:06 SYS@ORCL>
```

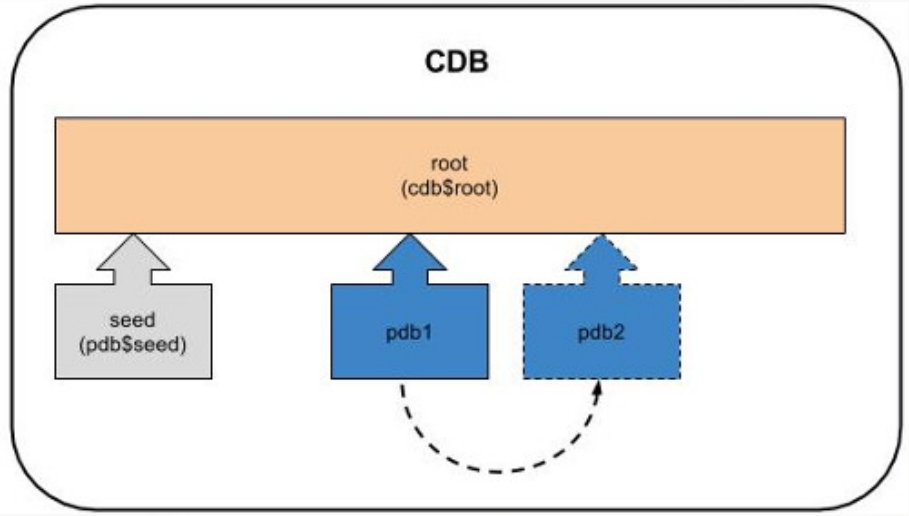
SELECT em outros Containers

```
08:05:23 C##RICARDO@ORCL> SELECT C1 FROM T;

        C1
----------
         1

Elapsed: 00:00:00.01
08:05:25 C##RICARDO@ORCL> SELECT C1 FROM CONTAINERS(T) WHERE CON_ID = 3;

        C1
----------
         1

Elapsed: 00:00:00.01
08:05:30 C##RICARDO@ORCL>
```

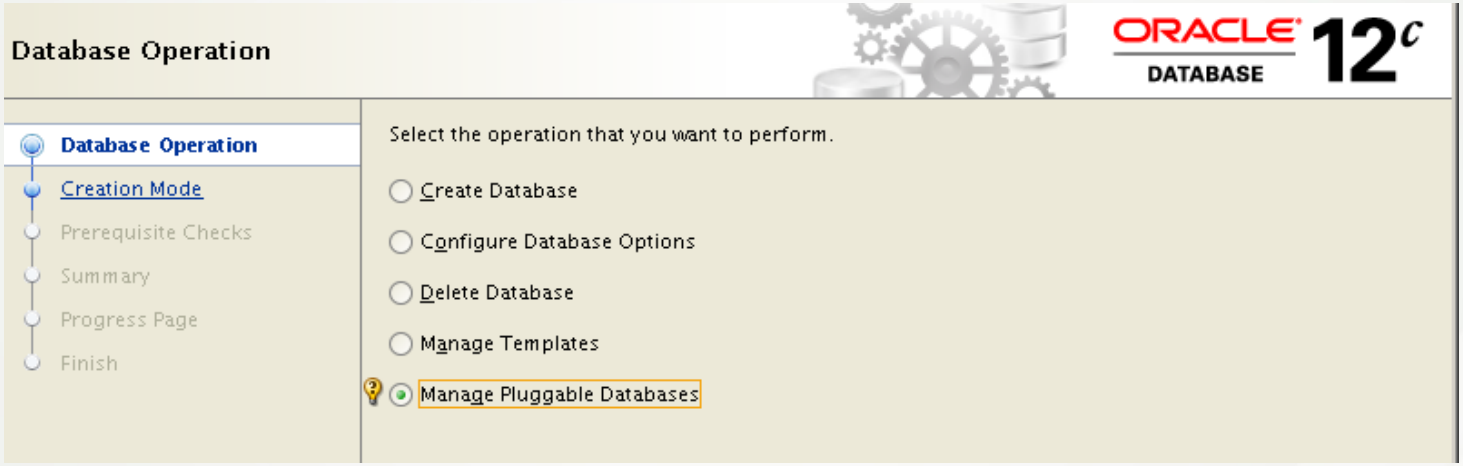
Criando (Clonando) um PDB







Criando (Clonando) um PDB



Criando (Clonando) um PDB

Criando (Clonando) um PDB

```
11:27:54 SYS@ORCL> CREATE PLUGGABLE DATABASE DEV
11:28:04   2  ADMIN USER RICARDO IDENTIFIED BY Nerv2017
11:28:13   3  FILE_NAME_CONVERT=(
11:28:19   4  '/u01/app/oracle/oradata/ORCL/pdbseed/',
11:28:25   5  '/u01/app/oracle/oradata/ORCL/DEV/');

Pluggable database created.

Elapsed: 00:00:22.04
11:29:00 SYS@ORCL> ALTER PLUGGABLE DATABASE DEV OPEN READ WRITE;

Pluggable database altered.

Elapsed: 00:00:09.39
11:29:30 SYS@ORCL>
```

Criando (Clonando) um PDB

```
11:29:31 SYS@ORCL> CREATE PLUGGABLE DATABASE DEVNEW FROM DEV
11:30:20   2  FILE_NAME_CONVERT=(
11:30:24   3  '/u01/app/oracle/oradata/ORCL/DEV/',
11:30:35   4  '/u01/app/oracle/oradata/ORCL/TantoFaz/');

Pluggable database created.

Elapsed: 00:00:29.23
11:31:33 SYS@ORCL> ALTER PLUGGABLE DATABASE DEVNEW OPEN READ WRITE;

Pluggable database altered.

Elapsed: 00:00:08.53
11:31:53 SYS@ORCL>
```

Unplug / Plug

```
08:11:38 SYS@ORCL> ALTER PLUGGABLE DATABASE DEV CLOSE;

Pluggable database altered.

Elapsed: 00:00:02.17
08:11:58 SYS@ORCL> ALTER PLUGGABLE DATABASE DEV UNPLUG INTO '/tmp/DEV.xml';

Pluggable database altered.

Elapsed: 00:00:10.13
08:12:29 SYS@ORCL> exit
```

```
[oracle@nerv06 ~]$ scp /tmp/DEV.xml oracle@192.168.0.104:/home/oracle/
oracle@192.168.0.104's password:
DEV.xml                                       100% 6199     6.1KB/s   00:00
[oracle@nerv06 ~]$ scp -Cr /u01/app/oracle/oradata/ORCL/DEV/ oracle@192.168.0.104:
01/app/oracle/oradata/ORCL/
oracle@192.168.0.104's password:
sysaux01.dbf                                  100%  195MB  19.5MB/s   00:10
system01.dbf                                  100%  210MB  26.3MB/s   00:08
undotbs01.dbf                                 100%  210MB  16.2MB/s   00:13
temp01.dbf                                    100%   20MB  20.0MB/s   00:00
[oracle@nerv06 ~]$
```

Unplug / Plug

```
08:20:37 SYS@ORCL> CREATE PLUGGABLE DATABASE DEV USING '/home/oracle/DEV.xml'
08:21:06   2  NOCOPY TEMPFILE REUSE;

Pluggable database created.

Elapsed: 00:00:06.78
08:21:24 SYS@ORCL> ALTER PLUGGABLE DATABASE DEV OPEN READ WRITE;

Pluggable database altered.

Elapsed: 00:00:21.09
08:22:04 SYS@ORCL>
```

DROP

```
[oracle@nerv06 ~]$ rlwrap sqlplus / as SYSDBA

SQL*Plus: Release 12.2.0.1.0 Production on Mon Jul 24 08:27:28 2017

Copyright (c) 1982, 2016, Oracle.  All rights reserved.


Connected to:
Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production

08:27:29 SYS@ORCL> DROP PLUGGABLE DATABASE DEV INCLUDING DATAFILES;

Pluggable database dropped.

Elapsed: 00:00:03.49
08:27:33 SYS@ORCL>
```

Backup

CDB e PDBs

```
RMAN> BACKUP DATABASE;
RMAN> BACKUP DATAFILE 1;
RMAN> BACKUP DATAFILE 1,3;
```

CDB

```
RMAN> BACKUP DATABASE ROOT;
RMAN> BACKUP TABLESPACE SYSTEM;
RMAN> BACKUP TABLESPACE SYSTEM,SYSAUX;
```

PDBs

```
RMAN> BACKUP PLUGGABLE DATABASE DEV1;
RMAN> BACKUP PLUGGABLE DATABASE DEV1,DEV2;
RMAN> BACKUP TABLESPACE DEV1:SYSTEM;
```

Restore / Recover

CDB e PDBs

```
RMAN> RESTORE DATABASE;
RMAN> RECOVER DATABASE;
RMAN> RESTORE DATAFILE 1;
RMAN> RECOVER DATAFILE 1;
```

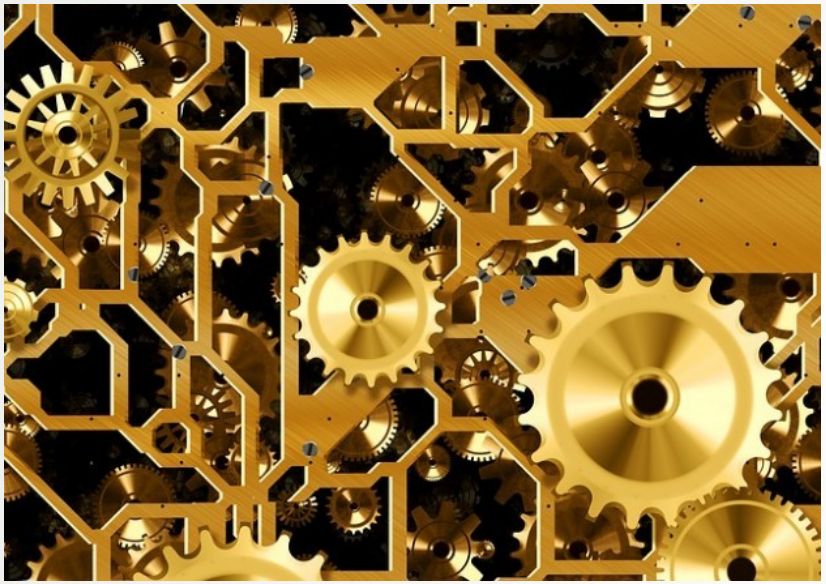
CDB

```
RMAN> RESTORE DATABASE ROOT;
RMAN> RECOVER DATABASE ROOT;
RMAN> RESTORE TABLESPACE SYSTEM;
RMAN> RECOVER TABLESPACE SYSTEM;
```

PDBs

```
RMAN> RESTORE PLUGGABLE DATABASE DEV1;
RMAN> RECOVER PLUGGABLE DATABASE DEV1;
RMAN> RESTORE TABLESPACE DEV1:USERS;
RMAN> RECOVER TABLESPACE DEV1:USERS;
```

“Dicas” de SQL Tuning?



Lendas de SQL

- O Custo não quer dizer nada?
- Índice BITMAP em baixa “cardinalidade”?
- Sintaxe Oracle ou ANSI?
- SELECT(1) ou SELECT COUNT(*)?
- Ordem da cláusula WHERE?
- Ordem de JOIN?
- CHAR é mais rápido do que VARCHAR2?
- Tabelas pequenas não utilizam índice?
- Cláusulas de negação não utilizam índice?
- Busca por NULL não utiliza índice?
- Evite Subqueries?
- Evite VIEWs?
- Evite EXISTS?
- Evite Listas IN?
- NESTED LOOPs é ruim?
- < > é melhor que BETWEEN?

Lab 1 – Lendas de SQL

Hands On !

Lab 1.1: Lendas de SQL - COUNT(*)

Execute logon com o usuário SCOTT, e verifique qual é seu arquivo de TRACE:

```
$ sqlplus SCOTT/TIGER
SQL> COLUMN TRACEFILE FORMAT A100
SQL> SELECT P.TRACEFILE FROM V$SESSION S, V$PROCESS P
    WHERE S.PADDR = P.ADDR AND S.USERNAME = 'SCOTT';
```

Coloque sua sessão em TRACE 10053, e execute os comandos abaixo.

```
SQL> ALTER SESSION SET EVENTS '10053 TRACE NAME CONTEXT FOREVER, LEVEL 1';
SQL> SELECT COUNT(EMPNO) FROM EMP;
SQL> SELECT COUNT(1) FROM EMP;
SQL> SELECT COUNT(2) FROM EMP;
SQL> SELECT COUNT(*) FROM EMP;
SQL> SELECT COUNT(ROWID) FROM EMP;
SQL> SELECT COUNT(MGR) FROM EMP;
SQL> SELECT E.ENAME, E.JOB, D.DNAME
    FROM EMP E, DEPT D WHERE E.DEPTNO = D.DEPTNO;
SQL> SELECT E.ENAME, E.JOB, D.DNAME
    FROM EMP E INNER JOIN DEPT D ON E.DEPTNO = D.DEPTNO;
SQL> SELECT E.ENAME, E.JOB, D.DNAME FROM EMP E, DEPT D
       WHERE E.DEPTNO = D.DEPTNO
       AND EMPNO IN (7369,7499,7521,7566,7654);
```

Edite o seu arquivo de TRACE.

```
$ vi /u01/app/oracle/rdbms/ORCL/orcl/trace/ORCL_ora_1234.trc
```

Lab 1.2: Lendas de SQL - COUNT(*)

Procure novamente pelo texto “CNT”:

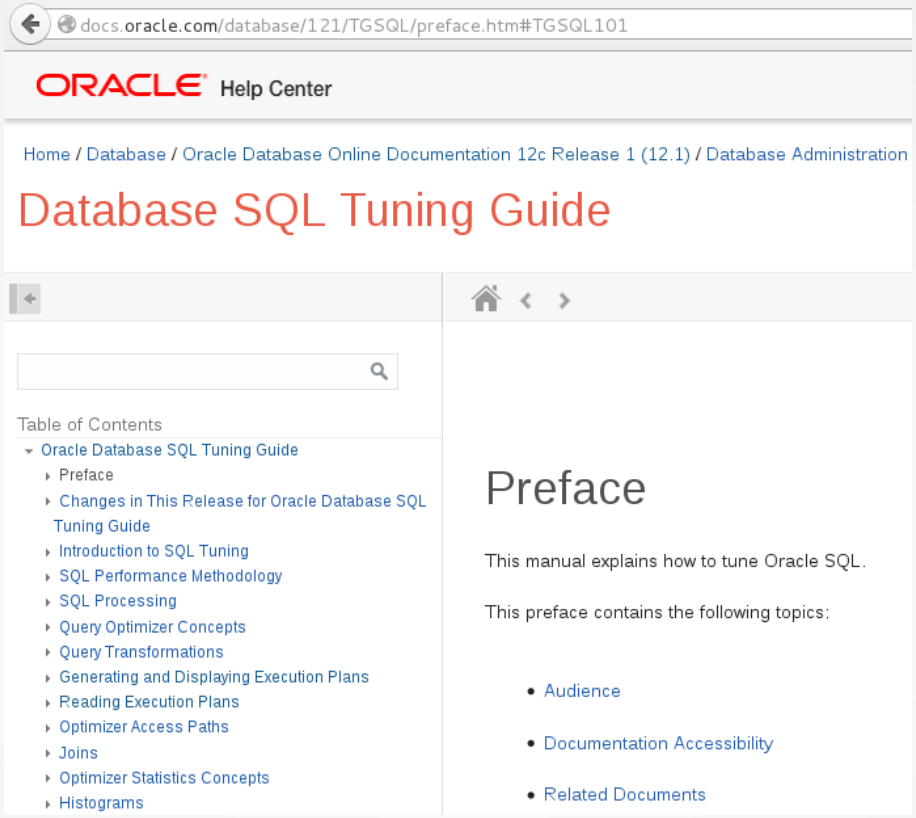
```
CNT:   Considering count(col) to count(*) on query block SEL$1 (#0)
**************************
Count(col) to Count(*) (CNT)
**************************
CNT:     Converting COUNT(EMPNO) to COUNT(*).
CNT:     COUNT() to COUNT(*) done.
```

Procure pelo texto “Final query”:

```
Final query after transformations:******* UNPARSED QUERY IS *******
SELECT COUNT(*) "COUNT(EMPNO)" FROM "SCOTT"."EMP" "EMP"
```

Cost Based Optimizer

Database SQL Tuning Guide

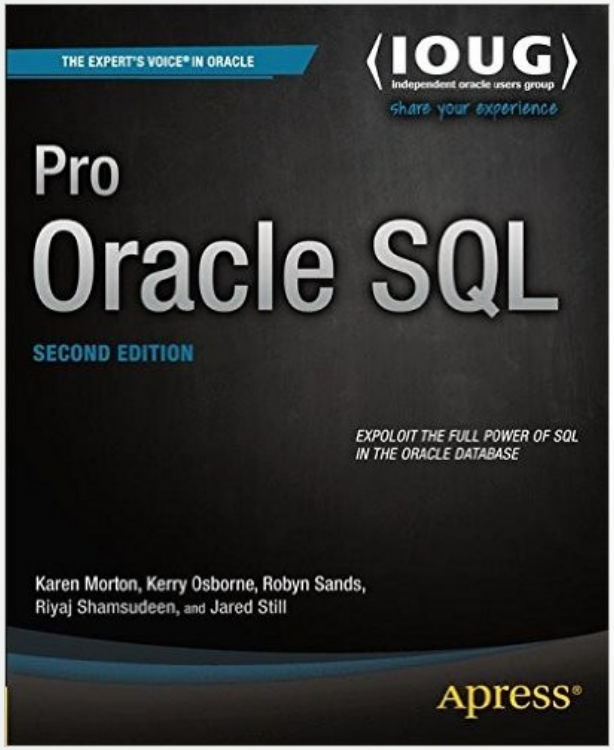


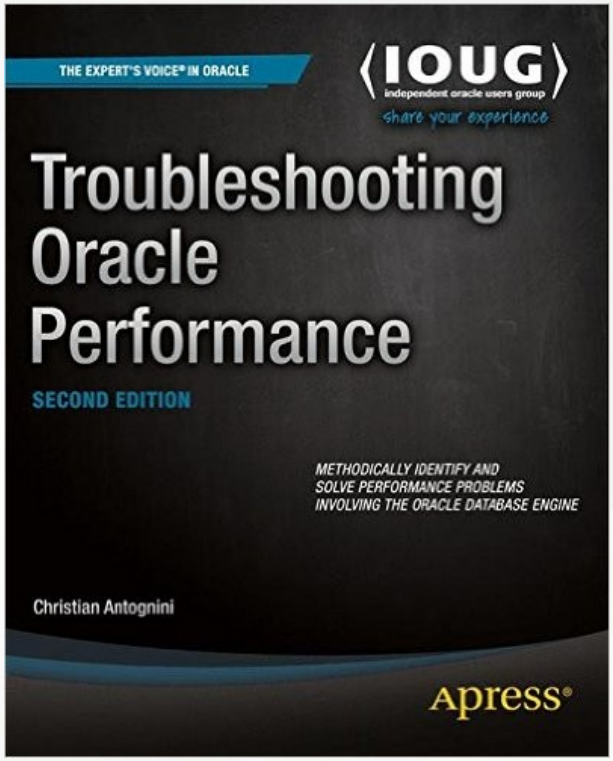
Database SQL Tuning Guide
http://docs.oracle.com/database/122/TGSQL/toc.htm

Database Performance Tuning Guide
http://docs.oracle.com/database/122/TGDBA/toc.htm

61

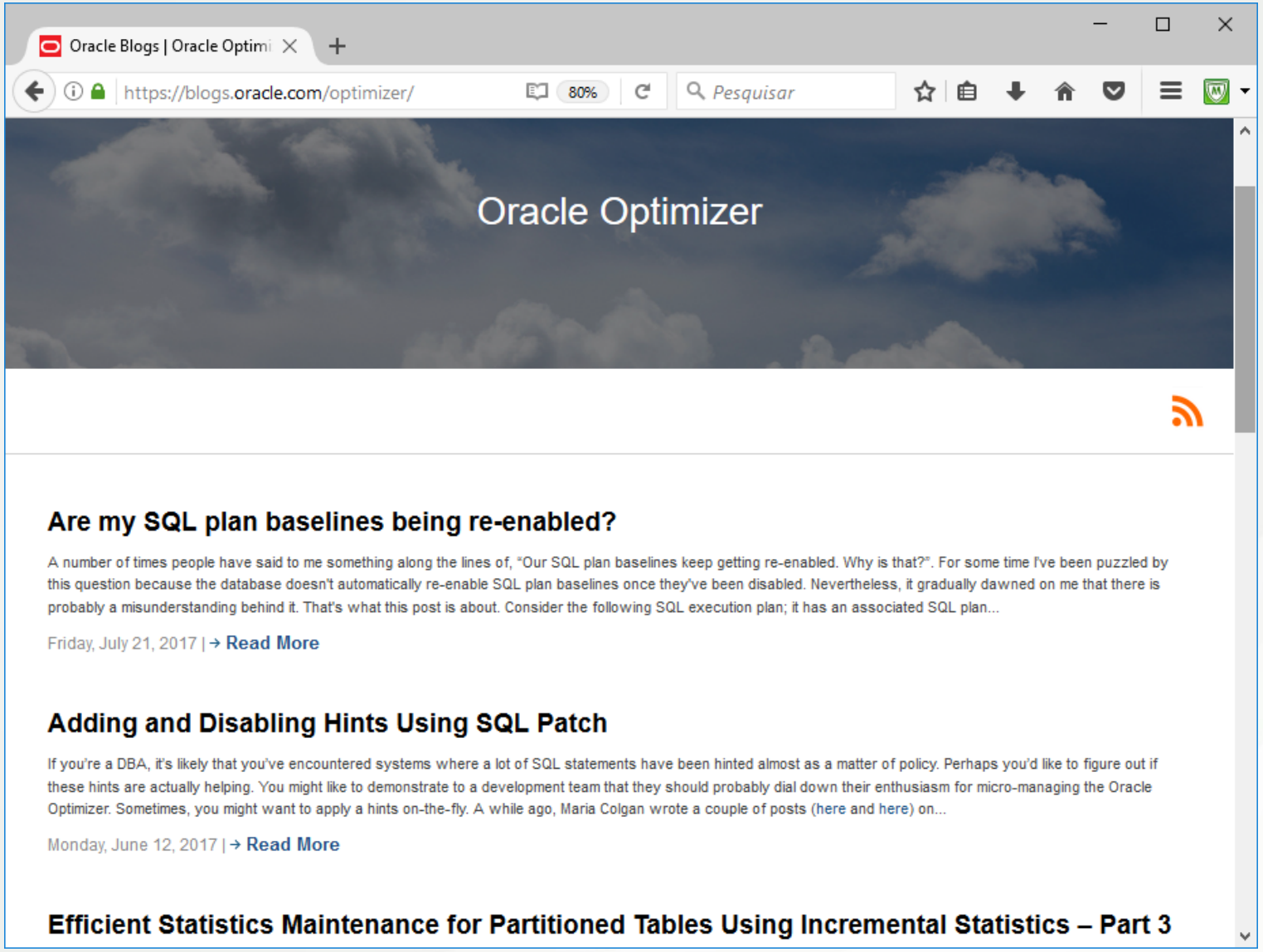
Livros


THE EXPERT'S VOICE® IN ORACLE
IOUG
independent oracle users group
share your experience
Pro
Oracle SQL
SECOND EDITION
EXPOLOIT THE FULL POWER OF SQL
IN THE ORACLE DATABASE
Karen Morton, Kerry Osborne, Robyn Sands,
Riyaj Shamsudeen, and Jared Still
Apress®


THE EXPERT'S VOICE® IN ORACLE
IOUG
independent oracle users group
share your experience
Troubleshooting
Oracle
Performance
SECOND EDITION
METHODICALLY IDENTIFY AND
SOLVE PERFORMANCE PROBLEMS
INVOLVING THE ORACLE DATABASE ENGINE
Christian Antognini
Apress®

62

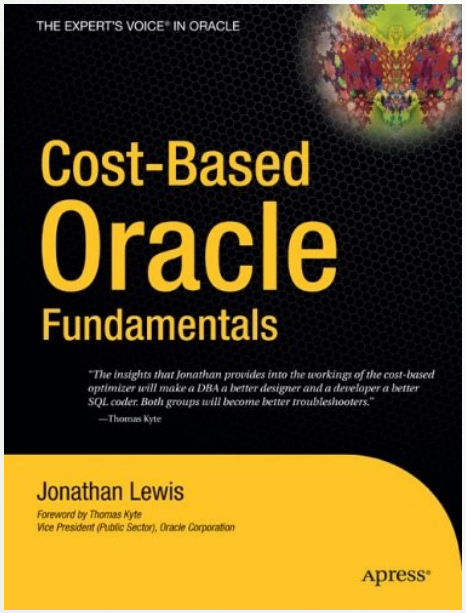
Oracle Optimizer Blog

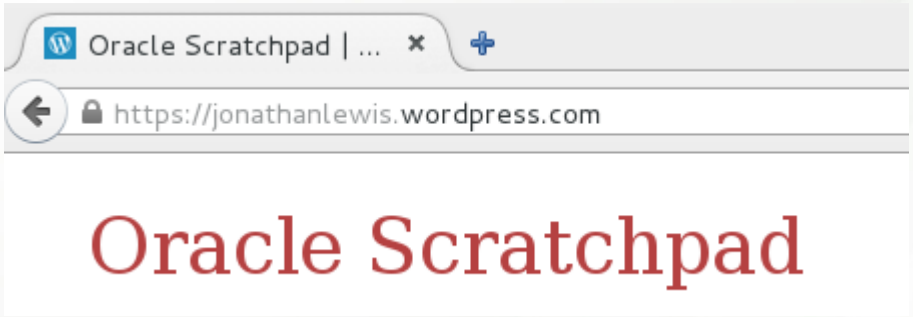


Cost Based Optimizer

"Exite uma teoria de que se alguém descobrir o que o CBO faz e como ele funciona, ele irá desaparecer instantaneamente e será substituído por algo ainda mais bizarro e inexplicável. Existe outra teoria de que isto já aconteceu... duas vezes."

Jonathan Lewis, parafraseando Douglas Adams.





64

CBO - Cost Based Optimizer – O que é o custo?

Cost = (
#SRds * sreadtim +
#MRds * mreadtim +
#CPUCycles / cpuspeed
) / sreadtim

OU

```
SQL> SELECT COUNT(ID) FROM T1 WHERE ID < 2;


Plano de Execuc?o
----------------------------------------------------------
Plan hash value: 1710841474

-------------------------------------------------------------------------------
| Id  | Operation         | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |         |     1 |     3 |  1031   (2)| 00:00:01 |
|   1 |  SORT AGGREGATE   |         |     1 |     3 |            |          |
|*  2 |   INDEX RANGE SCAN| IDX1_T1 |   129K|   378K|  1031   (2)| 00:00:01 |
-------------------------------------------------------------------------------
```

Custo = (
Quantidade de leituras de um único bloco * Tempo de leitura de um único bloco +
Quantidade de leituras de múltiplos blocos * Tempo de leitura de múltiplos blocos +
Ciclos de CPU / Velocidade da CPU
) / Tempo de leitura de um único bloco

O CBO foi lançado no Oracle 7.3.
O RBO foi considerado legado no 10g, mas existe até no 12.1.0.2.

O que é o Custo?

```
SQL> SELECT COUNT(ID) FROM T1 WHERE ID < 2;


Plano de Execuc?o
----------------------------------------------------------
Plan hash value: 1710841474

--------------------------------------------------------------------------------
| Id  | Operation         | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |         |     1 |     3 |  1031   (2)| 00:00:01 |
|   1 |  SORT AGGREGATE   |         |     1 |     3 |            |          |
|*  2 |   INDEX RANGE SCAN| IDX1_T1 |   129K|   378K|  1031   (2)| 00:00:01 |
--------------------------------------------------------------------------------
```

```
Estatisticas
----------------------------------------------------------
          0  recursive calls
          0  db block gets
        966  consistent gets
          0  physical reads
          0  redo size
        552  bytes sent via SQL*Net to client
        544  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

O que é o Custo?

```
SQL> SELECT PNAME, PVAL1 FROM SYS.AUX_STATS$ WHERE SNAME = 'SYSSTATS_MAIN'

PNAME                                PVAL1
------------------------------ ----------
CPUSPEED                              487
CPUSPEEDNW                            397
IOSEEKTIM                              10
IOTFRSPEED                           4096
MAXTHR                           12103680
MBRC                                   30
MREADTIM                             ,479
SLAVETHR                            72704
SREADTIM                           11,777
```

```
SQL> SELECT ((((0 * 0.479) + (966 * 11.777) + (1 / 487))) / 11.777) CUSTO FROM DUAL;

     CUSTO
----------
966,000174

SQL>
```

Seletividade e Cardinalidade

Seletividade
É um valor entre 0 e 1 (ou um percentual) que representa a fração de linhas obtidas por uma operação.

Cardinalidade
É o número de linhas retornadas por uma operação.

Exemplo:

```
SQL> SELECT MODELS FROM CARS;
120 rows selected.
```

Cardinalidade = 120.
Seletividade = 1.00 (120/120).

```
SQL> SELECT MODELS FROM CARS WHERE FAB = 'FORD';
18 rows selected.
```

Cardinalidade = 18.
Seletividade = 0.15 (18/120).

```
SQL> SELECT COUNT(MODELS) FROM CARS WHERE FAB = 'FORD';
1 row selected.
```

Cardinalidade = 1.
Seletividade = 0.15 (18/120).

Configuração do CBO

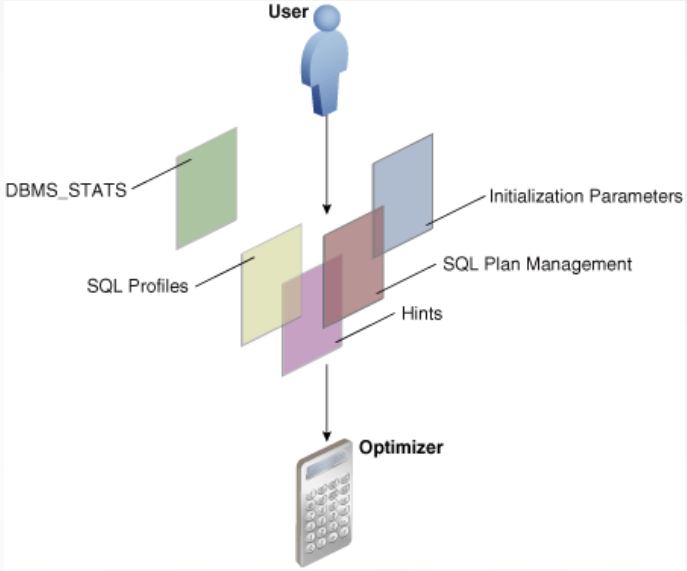
- OPTIMIZER_MODE (FIRST_ROWS_n / ALL_ROWS)
- DB_FILE_MULTIBLOCK_READ_COUNT
- RESULT_CACHE
- INMEMORY

- OPTIMIZER_SECURE_VIEW_MERGING
- QUERY_REWRITE_ENABLED
- QUERY_REWRITE_INTEGRITY
- STAR_TRANSFORMATION_ENABLED

- GATHER_SYSTEM_STATISTICS
- GATHER_DATABASE_STATISTICS
- OPTIMIZER_DYNAMIC_SAMPLING

- WORKAREA_SIZE_POLICY (AUTO / MANUAL)
- AUTO: PGA_AGGREGATE_TARGET
- MANUAL: BITMAP_MERGE_AREA_SIZE
  HASH_AREA_SIZE
  SORT_AREA_SIZE
  SORT_AREA_RETAINED_SIZE

- OPTIMIZER_INDEX_CACHING (0 a 100, padrão 0)
- OPTIMIZER_INDEX_COST_ADJ (1 a 10000, padrão 100)
- Histograms
- Extended Statistics



Configuração do CBO - OLTP

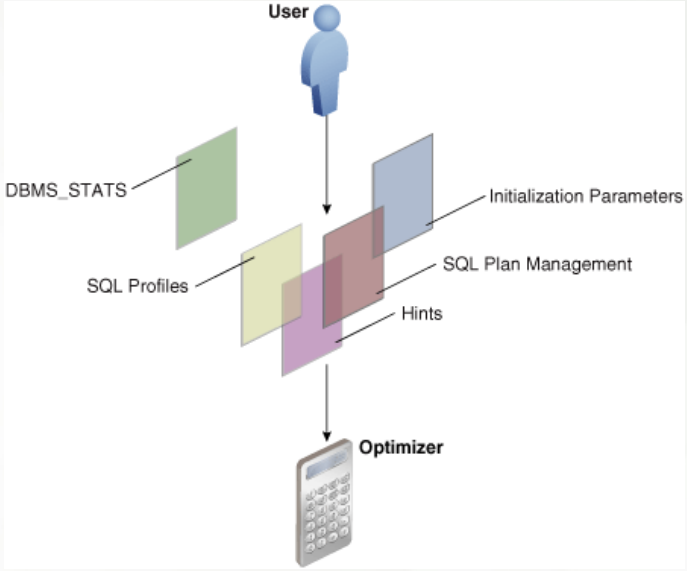
- OPTIMIZER_MODE (**FIRST_ROWS_n** / ALL_ROWS)
- **< DB_FILE_MULTIBLOCK_READ_COUNT**
- RESULT_CACHE
- INMEMORY

- OPTIMIZER_SECURE_VIEW_MERGING
- QUERY_REWRITE_ENABLED
- QUERY_REWRITE_INTEGRITY
- STAR_TRANSFORMATION_ENABLED

- **GATHER_SYSTEM_STATISTICS**
- **GATHER_DATABASE_STATISTICS**
- OPTIMIZER_DYNAMIC_SAMPLING

- WORKAREA_SIZE_POLICY (AUTO / MANUAL)
- **AUTO:** **PGA_AGGREGATE_TARGET**
- MANUAL: BITMAP_MERGE_AREA_SIZE
  HASH_AREA_SIZE
  SORT_AREA_SIZE
  SORT_AREA_RETAINED_SIZE

- **> OPTIMIZER_INDEX_CACHING**
- **< OPTIMIZER_INDEX_COST_ADJ**
- **Histograms**
- **Extended Statistics**



Configuração do CBO - OLAP

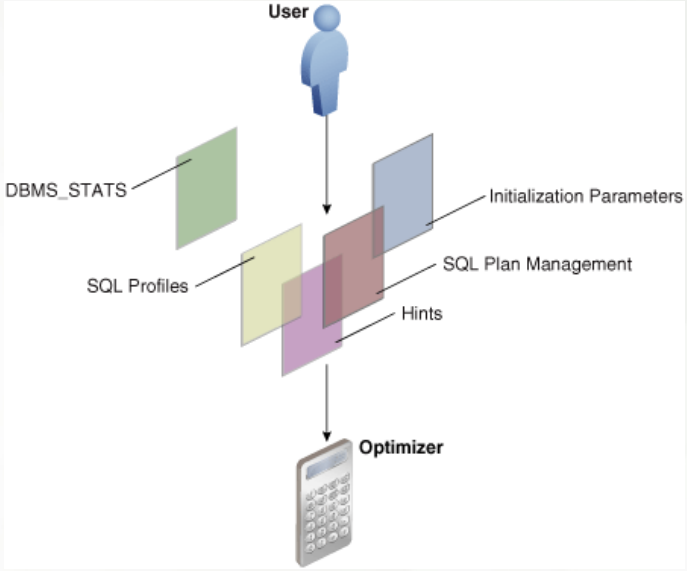
- OPTIMIZER_MODE (FIRST_ROWS_n / **ALL_ROWS**)
- **> DB_FILE_MULTIBLOCK_READ_COUNT**
- **RESULT_CACHE**
- **INMEMORY**

- **OPTIMIZER_SECURE_VIEW_MERGING**
- **QUERY_REWRITE_ENABLED**
- **QUERY_REWRITE_INTEGRITY**
- **STAR_TRANSFORMATION_ENABLED**

- **GATHER_SYSTEM_STATISTICS**
- **GATHER_DATABASE_STATISTICS**
- **OPTIMIZER_DYNAMIC_SAMPLING**

- WORKAREA_SIZE_POLICY (AUTO / MANUAL)
- AUTO: PGA_AGGREGATE_TARGET
- **MANUAL:** **BITMAP_MERGE_AREA_SIZE**
  HASH_AREA_SIZE
  SORT_AREA_SIZE
  SORT_AREA_RETAINED_SIZE

- **< OPTIMIZER_INDEX_CACHING**
- **> OPTIMIZER_INDEX_COST_ADJ**
- **Histograms**
- **Extended Statistics**



Ferramentas Básicas

AUTOTRACE

```
SQL> SET AUTOTRACE ON;
SQL> SELECT ENAME FROM EMP;
ENAME
----------
SMITH
...
14 rows selected.

Execution Plan
----------------------------------------------------------
Plan hash value: 3956160932

--------------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |    14 |    84 |     3   (0)| 00:00:01 |
|   1 |  TABLE ACCESS FULL| EMP  |    14 |    84 |     3   (0)| 00:00:01 |
--------------------------------------------------------------------------------

Statistics
----------------------------------------------------------
      1  recursive calls
...
      0  sorts (disk)
     14  rows processed
```

AUTOTRACE

```
SQL> SET AUTOTRACE TRACEONLY;
SQL> SELECT ENAME FROM EMP;

14 rows selected.

Execution Plan
----------------------------------------------------------
Plan hash value: 3956160932

--------------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |      |    14 |    84 |     3   (0)| 00:00:01 |
|   1 |  TABLE ACCESS FULL| EMP  |    14 |    84 |     3   (0)| 00:00:01 |
--------------------------------------------------------------------------------

Statistics
----------------------------------------------------------
      0  recursive calls
      0  db block gets
...
      0  sorts (disk)
     14  rows processed
```

AUTOTRACE

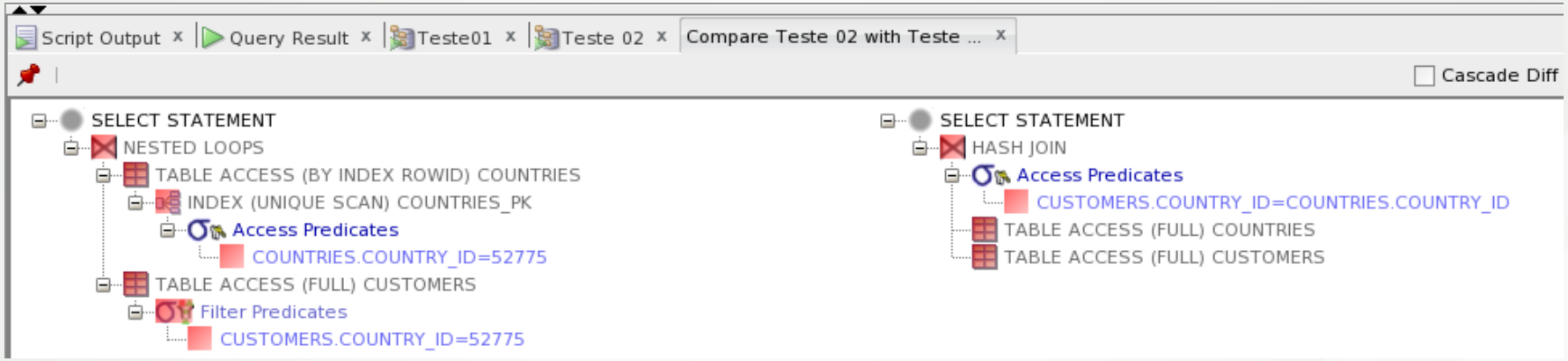
```
SQL> SET AUTOTRACE TRACEONLY EXPLAIN;
SQL> SELECT ENAME FROM EMP;

Execution Plan
----------------------------------------------------------
Plan hash value: 3956160932

--------------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |    14 |    84 |     3   (0)| 00:00:01 |
|   1 |  TABLE ACCESS FULL| EMP  |    14 |    84 |     3   (0)| 00:00:01 |
--------------------------------------------------------------------------------

SQL>
```

AUTOTRACE

```
SQL> SET AUTOTRACE TRACEONLY STATISTICS;
SQL> SELECT ENAME FROM EMP;

14 rows selected.


Statistics
----------------------------------------------------------
      0  recursive calls
      0  db block gets
      8  consistent gets
      0  physical reads
      0  redo size
    731  bytes sent via SQL*Net to client
    551  bytes received via SQL*Net from client
      2  SQL*Net roundtrips to/from client
      0  sorts (memory)
      0  sorts (disk)
     14  rows processed

SQL>
```

SQL Developer: Plano de Execução

SQL Developer: Plano de Execução



SQL Trace (Event 10046): Níveis

0 - Trace OFF
2 - Regular SQL Trace
4 - Nível 2, + Bind Variable
8 - Nível 2 + Wait Events
12 - Nível 2, + Bind Variable + Wait Events

SQL Trace (Event 10046): Ativação

Todas versões:

```
SQL> ALTER SESSION SET SQL_TRACE=TRUE;
SQL> ALTER SESSION SET SQL_TRACE=FALSE;

SQL> EXEC DBMS_SESSION.SET_SQL_TRACE(SQL_TRACE => TRUE);
SQL> EXEC DBMS_SESSION.SET_SQL_TRACE(SQL_TRACE => FALSE);

SQL> EXEC DBMS_SYSTEM.SET_SQL_TRACE_IN_SESSION
         (SID=>123, SERIAL#=>1234, SQL_TRACE=>TRUE);
SQL> EXEC DBMS_SYSTEM.SET_SQL_TRACE_IN_SESSION
         (SID=>123, SERIAL#=>1234, SQL_TRACE=>FALSE);

SQL> ALTER SESSION SET EVENTS '10046 TRACE NAME CONTEXT FOREVER, LEVEL 12';
SQL> ALTER SESSION SET EVENTS '10046 TRACE NAME CONTEXT OFF';

SQL> EXEC DBMS_SYSTEM.SET_EV(SI=>123, SE=>1234, EV=>10046, LE=>12, NM=>'');
SQL> EXEC DBMS_SYSTEM.SET_EV(SI=>123, SE=>1234, EV=>10046, LE=>0, NM=>'');
```

SQL Trace (Event 10046): Ativação

Versões >= 8i

```
SQL> SELECT P.SPID OS_PROCESS_ID
       FROM V$SESSION S, V$PROCESS P
       WHERE S.PADDR = P.ADDR AND S.USERNAME = 'SCOTT';
SQL> oradebug SETOSPID 12345;
SQL> oradebug EVENT 10046 TRACE NAME CONTEXT FOREVER, LEVEL 12;
SQL> oradebug TRACEFILE_NAME;
SQL> oradebug EVENT 10046 TRACE NAME CONTEXT OFF;
```

SQL Trace (Event 10046): Ativação

```
CREATE OR REPLACE TRIGGER SET_TRACE AFTER LOGON ON DATABASE
BEGIN
IF USER IN ('SCOTT') THEN

EXECUTE IMMEDIATE
    'ALTER SESSION SET TRACEFILE_IDENTIFIER=''SESSAO_RASTREADA_PORTILHO''';
EXECUTE IMMEDIATE
    'ALTER SESSION SET TIMED_STATISTICS=TRUE';
EXECUTE IMMEDIATE
    'ALTER SESSION SET MAX_DUMP_FILE_SIZE=UNLIMITED';
EXECUTE IMMEDIATE
    'ALTER SESSION SET EVENTS ''10046 TRACE NAME CONTEXT FOREVER, LEVEL 12''';

END IF;
 END;
/
```

SQL Trace (Event 10046): Ativação

Todas versões:

```
SQL> EXEC DBMS_SUPPORT.START_TRACE(WAITS=>TRUE, BINDS=>FALSE);
SQL> EXEC DBMS_SUPPORT.STOP_TRACE;

SQL> EXEC DBMS_SUPPORT.START_TRACE_IN_SESSION
     (SID=>123, SERIAL=>1234, WAITS=>TRUE, BINDS=>TRUE);
SQL> EXEC DBMS_SUPPORT.STOP_TRACE_IN_SESSION (SID=>123, SERIAL=>1234);
```

Versões >= 10g

```
SQL> EXEC DBMS_MONITOR.SESSION_TRACE_ENABLE
     (SESSION_ID =>1234, SERIAL_NUM=>1234, WAITS=>TRUE, BINDS=>FALSE);
SQL> EXEC DBMS_MONITOR.SESSION_TRACE_DISABLE  (SESSION_ID=>1234, SERIAL_NUM=>1234);

SQL> EXEC DBMS_MONITOR.CLIENT_ID_TRACE_ENABLE
     (CLIENT_ID=>'PORTILHO', WAITS=>TRUE, BINDS=>TRUE);
SQL> EXEC DBMS_MONITOR.CLIENT_ID_TRACE_DISABLE(CLIENT_ID=>'PORTILHO');

SQL> EXEC DBMS_MONITOR.SERV_MOD_ACT_TRACE_ENABLE
     (SERVICE_NAME=>'OLTP', MODULE_NAME=>'TESTE', ACTION_NAME=>'RUNNING',
WAITS=>TRUE, BINDS=>TRUE);
SQL> EXEC DBMS_MONITOR.SERV_MOD_ACT_TRACE_DISABLE
     (SERVICE_NAME=>'OLTP', MODULE_NAME=>'TESTE', ACTION_NAME=>'RUNNING');
```

Versões >= 11g

```
SQL> ALTER SESSION SET EVENTS 'trace[rdbms.SQL_Optimizer.*][sql:sql_id]';
```

SQL Trace (Event 10046): Ativação

```
SQL> ALTER SESSION SET EVENTS '10046 TRACE NAME CONTEXT FOREVER, LEVEL 12';
SQL> ALTER SESSION SET EVENTS '10046 TRACE NAME CONTEXT OFF';
```

OU

```
SQL> SELECT P.SPID, P.TRACEFILE
       FROM V$SESSION S, V$PROCESS P
       WHERE S.PADDR = P.ADDR AND S.USERNAME = 'SCOTT';
SQL> oradebug setospid 9999;
SQL> oradebug tracefile_name;
SQL> oradebug unlimit;
SQL> oradebug event 10046 trace name context forever, level 12;
```

SQL Trace (Event 10053)

```
SQL> ALTER SESSION SET EVENTS '10053 TRACE NAME CONTEXT FOREVER, LEVEL 1';
SQL> ALTER SESSION SET EVENTS '10053 TRACE NAME CONTEXT OFF';
```

OU

```
SQL> SELECT P.SPID, P.TRACEFILE
       FROM V$SESSION S, V$PROCESS P
       WHERE S.PADDR = P.ADDR AND S.USERNAME = 'SCOTT';
SQL> oradebug setospid 9999;
SQL> oradebug tracefile_name;
SQL> oradebug unlimit;
SQL> oradebug event 10053 trace name context forever, level 1;
```

Lab 2 – SQL Trace

Hands On !

Lab 2.1: SQL Trace (Event 10046)

```
SQL> CONN SCOTT/TIGER
SQL> SELECT P.SPID, P.TRACEFILE
       FROM V$SESSION S, V$PROCESS P
       WHERE S.PADDR = P.ADDR AND S.USERNAME = 'SCOTT';
SQL> ALTER SESSION SET EVENTS '10046 TRACE NAME CONTEXT FOREVER, LEVEL 12';
SQL> SELECT ENAME FROM EMP WHERE EMPNO = 7369;
SQL> SELECT * FROM EMP;
SQL> CREATE TABLE T21 AS SELECT * FROM ALL_OBJECTS;
```

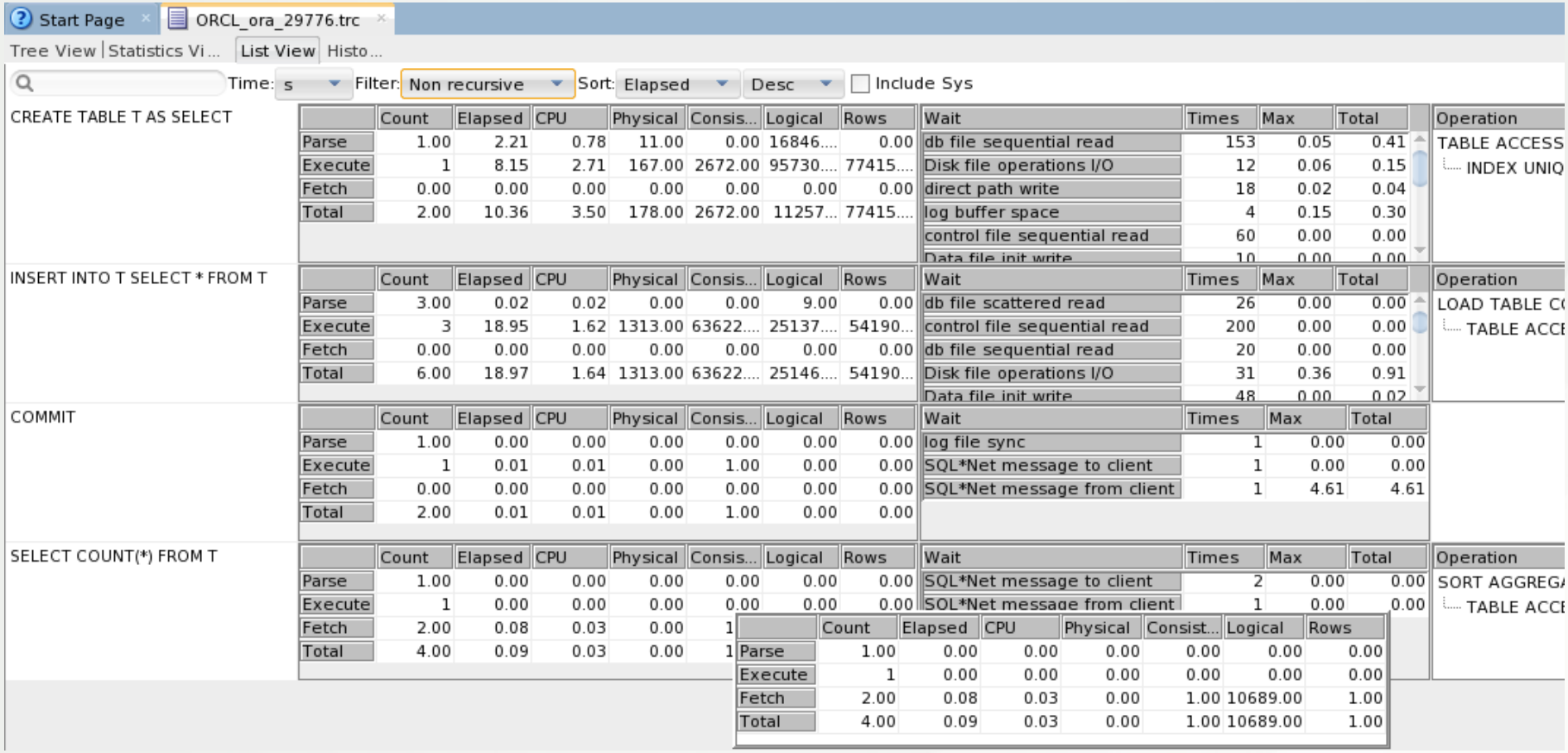
```
=====================
PARSING IN CURSOR #140195000366168 len=40 dep=0 uid=110 oct=3 lid=110 tim=14640556504 hv=565571065 ad='914beb80' sqlid='4z8r63nhvbvgt'
SELECT ENAME FROM EMP WHERE EMPNO = 7369
END OF STMT
PARSE #140195000366168:c=19996,e=117728,p=1,cr=80,cu=0,mis=1,r=0,dep=0,og=1,plh=2949544139,tim=14640556503
EXEC #140195000366168:c=0,e=39,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,plh=2949544139,tim=14640556829
WAIT #140195000366168: nam='SQL*Net message to client' ela= 9 driver id=1650815232 #bytes=1 p3=0 obj#=20366 tim=14640556973
WAIT #140195000366168: nam='Disk file operations I/O' ela= 69 FileOperation=2 fileno=6 filetype=2 obj#=93204 tim=14640557315
WAIT #140195000366168: nam='db file sequential read' ela= 14771 file#=6 block#=203 blocks=1 obj#=93204 tim=14640572178
WAIT #140195000366168: nam='db file sequential read' ela= 10267 file#=6 block#=197 blocks=1 obj#=93203 tim=14640582658
FETCH #140195000366168:c=1000,e=25719,p=2,cr=2,cu=0,mis=0,r=1,dep=0,og=1,plh=2949544139,tim=14640582786
STAT #140195000366168 id=1 cnt=1 pid=0 pos=1 obj=93203 op='TABLE ACCESS BY INDEX ROWID EMP (cr=2 pr=2 pw=0 time=25710 us cost=1 size=10 card=1)'
STAT #140195000366168 id=2 cnt=1 pid=1 pos=1 obj=93204 op='INDEX UNIQUE SCAN PK_EMP (cr=1 pr=1 pw=0 time=15259 us cost=0 size=0 card=1)'
WAIT #140195000366168: nam='SQL*Net message from client' ela= 401 driver id=1650815232 #bytes=1 p3=0 obj#=93203 tim=14640583298
FETCH #140195000366168:c=0,e=2,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=0,plh=2949544139,tim=14640583391
WAIT #140195000366168: nam='SQL*Net message to client' ela= 3 driver id=1650815232 #bytes=1 p3=0 obj#=93203 tim=14640583456
```

```
[oracle@Melquior ~]$ tkprof /u01/app/oracle/diag/rdbms/orcl/ORCL/trace/ORCL_ora_10486.trc
output = SCOTT_01

TKPROF: Release 12.1.0.2.0 - Development on Thu Feb 19 17:19:51 2015

Copyright (c) 1982, 2014, Oracle and/or its affiliates.  All rights reserved.


[oracle@Melquior ~]$ vi SCOTT_01.prf
```

Lab 2.2: tkprof (Event 10046)

```
SQL ID: 4z8r63nhvbvgt Plan Hash: 2949544139

SELECT ENAME
FROM
 EMP WHERE EMPNO = 7369


call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.05          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2      0.00       0.02          2          2          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      0.00       0.07          2          2          0           1

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 110
Number of plan statistics captured: 1

Rows (1st) Rows (avg) Rows (max)  Row Source Operation
---------- ---------- ----------  ---------------------------------------------------
         1          1          1  TABLE ACCESS BY INDEX ROWID EMP (cr=2 pr=2 pw=0 t
         1          1          1   INDEX UNIQUE SCAN PK_EMP (cr=1 pr=1 pw=0 time=15

Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                        2        0.00          0.00
  Disk file operations I/O                         1        0.00          0.00
  db file sequential read                          2        0.01          0.02
  SQL*Net message from client                      2       41.49         41.49
********************************************************************************
```

Lab 2.3: SQL Trace (Event 10046)



Start Page | ORCL_ora_29776.trc

Tree View | Statistics Vi... | List View | Histo...

Time: s | Filter: Non recursive | Sort: Elapsed | Desc | Include Sys

CREATE TABLE T AS SELECT

	Count	Elapsed	CPU	Physical	Consis...	Logical	Rows
Parse	1.00	2.21	0.78	11.00	0.00	16846....	0.00
Execute	1	8.15	2.71	167.00	2672.00	95730....	77415....
Fetch	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Total	2.00	10.36	3.50	178.00	2672.00	11257...	77415....

Wait	Times	Max	Total
db file sequential read	153	0.05	0.41
Disk file operations I/O	12	0.06	0.15
direct path write	18	0.02	0.04
log buffer space	4	0.15	0.30
control file sequential read	60	0.00	0.00
Data file init write	10	0.00	0.00

Operation: TABLE ACCESS / INDEX UNIQ

INSERT INTO T SELECT * FROM T

	Count	Elapsed	CPU	Physical	Consis...	Logical	Rows
Parse	3.00	0.02	0.02	0.00	0.00	9.00	0.00
Execute	3	18.95	1.62	1313.00	63622....	25137....	54190...
Fetch	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Total	6.00	18.97	1.64	1313.00	63622....	25146....	54190...

Wait	Times	Max	Total
db file scattered read	26	0.00	0.00
control file sequential read	200	0.00	0.00
db file sequential read	20	0.00	0.00
Disk file operations I/O	31	0.36	0.91
Data file init write	48	0.00	0.02

Operation: LOAD TABLE C / TABLE ACC

COMMIT

	Count	Elapsed	CPU	Physical	Consis...	Logical	Rows
Parse	1.00	0.00	0.00	0.00	0.00	0.00	0.00
Execute	1	0.01	0.01	0.00	1.00	0.00	0.00
Fetch	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Total	2.00	0.01	0.01	0.00	1.00	0.00	0.00

Wait	Times	Max	Total
log file sync	1	0.00	0.00
SQL*Net message to client	1	0.00	0.00
SQL*Net message from client	1	4.61	4.61

SELECT COUNT(*) FROM T

	Count	Elapsed	CPU	Physical	Consis...	Logical	Rows
Parse	1.00	0.00	0.00	0.00	0.00	0.00	0.00
Execute	1	0.00	0.00	0.00	0.00	0.00	0.00
Fetch	2.00	0.08	0.03	0.00	1		
Total	4.00	0.09	0.03	0.00	1		

Wait	Times	Max	Total
SQL*Net message to client	2	0.00	0.00
SQL*Net message from client	1	0.00	0.00

Operation: SORT AGGREGA / TABLE ACC

	Count	Elapsed	CPU	Physical	Consist...	Logical	Rows
Parse	1.00	0.00	0.00	0.00	0.00	0.00	0.00
Execute	1	0.00	0.00	0.00	0.00	0.00	0.00
Fetch	2.00	0.08	0.03	0.00	1.00	10689.00	1.00
Total	4.00	0.09	0.03	0.00	1.00	10689.00	1.00

Lab 2.4: SQL Trace (Event 10053)

```
SQL> CONN SCOTT/TIGER
SQL> SELECT P.SPID, P.TRACEFILE
       FROM V$SESSION S, V$PROCESS P
       WHERE S.PADDR = P.ADDR AND S.USERNAME = 'SCOTT';
SQL> ALTER SESSION SET EVENTS '10053 TRACE NAME CONTEXT FOREVER, LEVEL 1';
SQL> SELECT EMPNO, ENAME, DNAME, LOC
       FROM EMP, DEPT
       WHERE EMP.DEPTNO = DEPT.DEPTNO;
```

```
*** 2015-02-19 17:36:09.078
*** SESSION ID:(8.8723) 2015-02-19 17:36:09.078
*** CLIENT ID:() 2015-02-19 17:36:09.078
*** SERVICE NAME:(SYS$USERS) 2015-02-19 17:36:09.078
*** MODULE NAME:(SQL*Plus) 2015-02-19 17:36:09.078
*** CLIENT DRIVER:(SQL*PLUS) 2015-02-19 17:36:09.078
*** ACTION NAME:() 2015-02-19 17:36:09.078

Registered qb: SEL$1 0x36f6bf78 (PARSER)
---------------------
QUERY BLOCK SIGNATURE
---------------------
  signature (): qb_name=SEL$1 nbfros=2 flg=0
    fro(0): flg=4 objn=93201 hint_alias="DEPT"@"SEL$1"
    fro(1): flg=4 objn=93203 hint_alias="EMP"@"SEL$1"

SPM: statement not found in SMB
SPM: capture of plan baseline is OFF
```

Lab 2.5: SQL Trace (Event 10053)

```
***************************
Automatic degree of parallelism (AUTODOP)
***************************
Automatic degree of parallelism is disabled: Parameter.
kkopqSetForceParallelProperties: Hint:no
Query: compute:yes forced:no forceDop:0
kkopqSetDopReason: Reason why we chose this DOP is: table property.
table property forces parallelism

Global Manual DOP: 1 - Rounded?: no
```

```
PM: Considering predicate move-around in query block SEL$1 (#0)
***************************
Predicate Move-Around (PM)
***************************
```

Lab 2.6: SQL Trace (Event 10053)

```
Legend
The following abbreviations are used by optimizer trace.
CBQT - cost-based query transformation
JPPD - join predicate push-down
OJPPD - old-style (non-cost-based) JPPD
FPD - filter push-down
PM - predicate move-around
CVM - complex view merging
SPJ - select-project-join
SJC - set join conversion
SU - subquery unnesting
OBYE - order by elimination
OST - old style star transformation
ST - new (cbqt) star transformation
CNT - count(col) to count(*) transformation
JE - Join Elimination
JF - join factorization
CBY - connect by
SLP - select list pruning
DP - distinct placement
VT - vector transformation
qb - query block
LB - leaf blocks
```

Lab 2.7: SQL Trace (Event 10053)

```
***************************************
PARAMETERS USED BY THE OPTIMIZER
********************************
  *************************************
  PARAMETERS WITH ALTERED VALUES
  ******************************
Compilation Environment Dump
Bug Fix Control Environment


  *************************************
  PARAMETERS WITH DEFAULT VALUES
  ******************************
Compilation Environment Dump
optimizer_mode_hinted                  = false
optimizer_features_hinted              = 0.0.0
parallel_execution_enabled             = true
parallel_query_forced_dop              = 0
parallel_dml_forced_dop                = 0
parallel_ddl_forced_degree             = 0
parallel_ddl_forced_instances          = 0
_query_rewrite_fudge                   = 90
optimizer_features_enable              = 12.1.0.2
_optimizer_search_limit                = 5
cpu_count                              = 4
active_instance_count                  = 1
parallel_threads_per_cpu               = 2
hash_area_size                         = 131072
bitmap_merge_area_size                 = 1048576
sort_area_size                         = 65536
```

Lab 2.8: SQL Trace (Event 10053)

```
Bug Fix Control Environment
    fix  3834770 = 1
    fix  3746511 = enabled
    fix  4519016 = enabled
    fix  3118776 = enabled
    fix  4488689 = enabled
    fix  2194204 = disabled
    fix  2660592 = enabled
    fix  2320291 = enabled
    fix  2324795 = enabled
    fix  4308414 = enabled
    fix  3499674 = disabled
    fix  4569940 = enabled
    fix  4631959 = enabled
    fix  4519340 = enabled
    fix  4550003 = enabled
    fix  1403283 = enabled
    fix  4554846 = enabled
    fix  4602374 = enabled
    fix  4584065 = enabled
    fix  4545833 = enabled
    fix  4611850 = enabled
```

Lab 2.9: SQL Trace (Event 10053)

```
Considering Query Transformations on query block SEL$1 (#0)
**************************
Query transformations (QT)
**************************
JF: Checking validity of join factorization for query block SEL$1 (#0)
JF: Bypassed: not a UNION or UNION-ALL query block.
ST: not valid since star transformation parameter is FALSE
TE: Checking validity of table expansion for query block SEL$1 (#0)
TE: Bypassed: No partitioned table in query block.
VT: Initial VT validity check for query block SEL$1 (#0)
VT: Bypassed: inmemory is disabled.
CBQT bypassed for query block SEL$1 (#0): no complex view, sub-queries or UNION (ALL) queries.
CBQT: Validity checks failed for 316w41h1thpg1.
CSE: Considering common sub-expression elimination in query block SEL$1 (#0)
**************************
Common Subexpression elimination (CSE)
**************************
CSE:     CSE not performed on query block SEL$1 (#0).
OBYE:   Considering Order-by Elimination from view SEL$1 (#0)
**************************
Order-by elimination (OBYE)
**************************
OBYE:     OBYE bypassed: no order by to eliminate.
JE:   Considering Join Elimination on query block SEL$1 (#0)
**************************
Join Elimination (JE)
**************************
```

```
Final query after transformations:******* UNPARSED QUERY IS *******
SELECT "EMP"."EMPNO" "EMPNO","EMP"."ENAME" "ENAME","DEPT"."DNAME" "DNAME",
 "DEPT"."DEPTNO"=10
kkoqbc: optimizing query block SEL$1 (#0)
```

Lab 2.10: SQL Trace (Event 10053)

```
----------------------------
SYSTEM STATISTICS INFORMATION
----------------------------
Using dictionary system stats.
  Using NOWORKLOAD Stats
  CPUSPEEDNW: 1111 millions instructions/sec (default is 100)
  IOTFRSPEED: 4096 bytes per millisecond (default is 4096)
  IOSEEKTIM:  13 milliseconds (default is 10)
  MBRC:       NO VALUE blocks (default is 8)

**************************************
BASE STATISTICAL INFORMATION
**********************
Table Stats::
  Table: DEPT  Alias: DEPT
  #Rows: 4  SSZ: 0  LGR: 0  #Blks:  5  AvgRowLen:  20.00  NEB: 0  ChainCnt:  0.00
  #IMCUs: 0  IMCRowCnt: 0  IMCJournalRowCnt: 0  #IMCBlocks: 0  IMCQuotient: 0.00000
Index Stats::
  Index: PK_DEPT  Col#: 1
  LVLS: 0  #LB: 1  #DK: 4  LB/K: 1.00  DB/K: 1.00  CLUF: 1.00  NRW: 4.00 SSZ: 0.00
  KKEISFLG: 1
**********************
Table Stats::
  Table: EMP  Alias: EMP
  #Rows: 14  SSZ: 0  LGR: 0  #Blks:  5  AvgRowLen:  38.00  NEB: 0  ChainCnt:  0.00
  #IMCUs: 0  IMCRowCnt: 0  IMCJournalRowCnt: 0  #IMCBlocks: 0  IMCQuotient: 0.00000
Index Stats::
  Index: PK_EMP  Col#: 1
  LVLS: 0  #LB: 1  #DK: 14  LB/K: 1.00  DB/K: 1.00  CLUF: 1.00  NRW: 14.00 SSZ: 0.0
  KKEISFLG: 1
**************************************
```

Lab 2.11: SQL Trace (Event 10053)

```
Access path analysis for DEPT
***************************************
SINGLE TABLE ACCESS PATH
  Single Table Cardinality Estimation for DEPT[DEPT]
  SPD: Return code in qosdDSDirSetup: NOCTX, estType = TABLE
  Column (#1): DEPTNO(NUMBER)
    AvgLen: 3 NDV: 4 Nulls: 0 Density: 0.250000 Min: 10.000000 Max: 40.000000
  Table: DEPT  Alias: DEPT
    Card: Original: 4.000000  Rounded: 1  Computed: 1.000000  Non Adjusted: 1.000000
  Scan IO  Cost (Disk) =   3.000000
  Scan CPU Cost (Disk) =   36247.200000
  Total Scan IO  Cost  =   3.000000 (scan (Disk))
                         + 0.000000 (io filter eval) (= 0.000000 (per row) * 4.000000 (#rows))
                       =   3.000000
  Total Scan CPU  Cost =   36247.200000 (scan (Disk))
                         + 200.000000 (cpu filter eval) (= 50.000000 (per row) * 4.000000 (#rows))
                       =   36447.200000
  Access Path: TableScan
    Cost:  3.002187  Resp: 3.002187  Degree: 0
      Cost_io: 3.000000  Cost_cpu: 36447
      Resp_io: 3.000000  Resp_cpu: 36447
 ****** Costing Index PK_DEPT
  SPD: Return code in qosdDSDirSetup: NOCTX, estType = INDEX_SCAN
  SPD: Return code in qosdDSDirSetup: NOCTX, estType = INDEX_FILTER
  Access Path: index (UniqueScan)
    Index: PK_DEPT
    resc_io: 1.000000  resc_cpu: 8361
    ix_sel: 0.250000  ix_sel_with_filters: 0.250000
    Cost: 1.000502  Resp: 1.000502  Degree: 1
```

```
  Best:: AccessPath: TableScan
         Cost: 3.002422  Degree: 1  Resp: 3.002422  Card: 4.666667  Bytes: 0.000000

***************************************
```

Lab 2.12: SQL Trace (Event 10053)

```
***************
Now joining: EMP[EMP]#1
***************
NL Join
  Outer table: Card: 1.000000  Cost: 1.000502  Resp: 1.000502  Degree: 1  Bytes:
Access path analysis for EMP
  Scan IO  Cost (Disk) =   3.000000
  Scan CPU Cost (Disk) =   39667.200000
  Total Scan IO  Cost  =   3.000000 (scan (Disk))
                         + 0.000000 (io filter eval) (= 0.000000 (per row) * 14.000000 (#rows))
                       =   3.000000
  Total Scan CPU  Cost =   39667.200000 (scan (Disk))
                         + 700.000000 (cpu filter eval) (= 50.000000 (per row) * 14.000000 (#rows))
                       =   40367.200000
  Inner table: EMP  Alias: EMP
  Access Path: TableScan
    NL Join:  Cost: 4.002924  Resp: 4.002924  Degree: 1
      Cost_io: 4.000000  Cost_cpu: 48729
      Resp_io: 4.000000  Resp_cpu: 48729

  Best NL cost: 4.002924
          resc: 4.002924  resc_io: 4.000000  resc_cpu: 48729
          resp: 4.002924  resp_io: 4.000000  resc_cpu: 48729
  SPD: Return code in qosdDSDirSetup: NOCTX, estType = JOIN
Join Card:  4.666667 = outer (1.000000) * inner (4.666667) * sel (1.000000)
Join Card - Rounded: 5 Computed: 4.666667
Best:: JoinMethod: NestedLoop
       Cost: 4.002924  Degree: 1  Resp: 4.002924  Card: 4.666667 Bytes:
```

Lab 2.13: SQL Trace (Event 10053)

```
_
============
Plan Table
============
-------------------------------------------------+-----------------------------------+
| Id  | Operation                     | Name    | Rows  | Bytes | Cost  | Time      |
-------------------------------------------------+-----------------------------------+
| 0   | SELECT STATEMENT              |         |       |       |     4 |           |
| 1   |  NESTED LOOPS                 |         |     5 |   165 |     4 |  00:00:01 |
| 2   |   TABLE ACCESS BY INDEX ROWID | DEPT    |     1 |    20 |     1 |  00:00:01 |
| 3   |    INDEX UNIQUE SCAN          | PK_DEPT |     1 |       |     0 |           |
| 4   |   TABLE ACCESS FULL           | EMP     |     5 |    65 |     3 |  00:00:01 |
-------------------------------------------------+-----------------------------------+
Predicate Information:
----------------------
3 - access("DEPT"."DEPTNO"=10)
4 - filter("EMP"."DEPTNO"=10)
```

```
/*+
  BEGIN_OUTLINE_DATA
    IGNORE_OPTIM_EMBEDDED_HINTS
    OPTIMIZER_FEATURES_ENABLE('12.1.0.2')
    DB_VERSION('12.1.0.2')
    ALL_ROWS
    OUTLINE_LEAF(@"SEL$1")
    INDEX_RS_ASC(@"SEL$1" "DEPT"@"SEL$1" ("DEPT"."DEPTNO"))
    FULL(@"SEL$1" "EMP"@"SEL$1")
    LEADING(@"SEL$1" "DEPT"@"SEL$1" "EMP"@"SEL$1")
    USE_NL(@"SEL$1" "EMP"@"SEL$1")
  END_OUTLINE_DATA
*/
```

Lab 2.14: SQL Trace (Event 10053)

```
Optimizer state dump:
Compilation Environment Dump
optimizer_mode_hinted                  = false
optimizer_features_hinted              = 0.0.0
parallel_execution_enabled             = true
parallel_query_forced_dop              = 0
parallel_dml_forced_dop                = 0
parallel_ddl_forced_degree             = 0
parallel_ddl_forced_instances          = 0
_query_rewrite_fudge                   = 90
optimizer_features_enable              = 12.1.0.2
_optimizer_search_limit                = 5
cpu_count                              = 4
active_instance_count                  = 1
parallel_threads_per_cpu               = 2
hash_area_size                         = 131072
bitmap_merge_area_size                 = 1048576
sort_area_size                         = 65536
sort_area_retained_size                = 0
_sort_elimination_cost_ratio           = 0
_optimizer_block_size                  = 8192
_sort_multiblock_read_count            = 2
_hash_multiblock_io_count              = 0
_db_file_optimizer_read_count          = 8
```

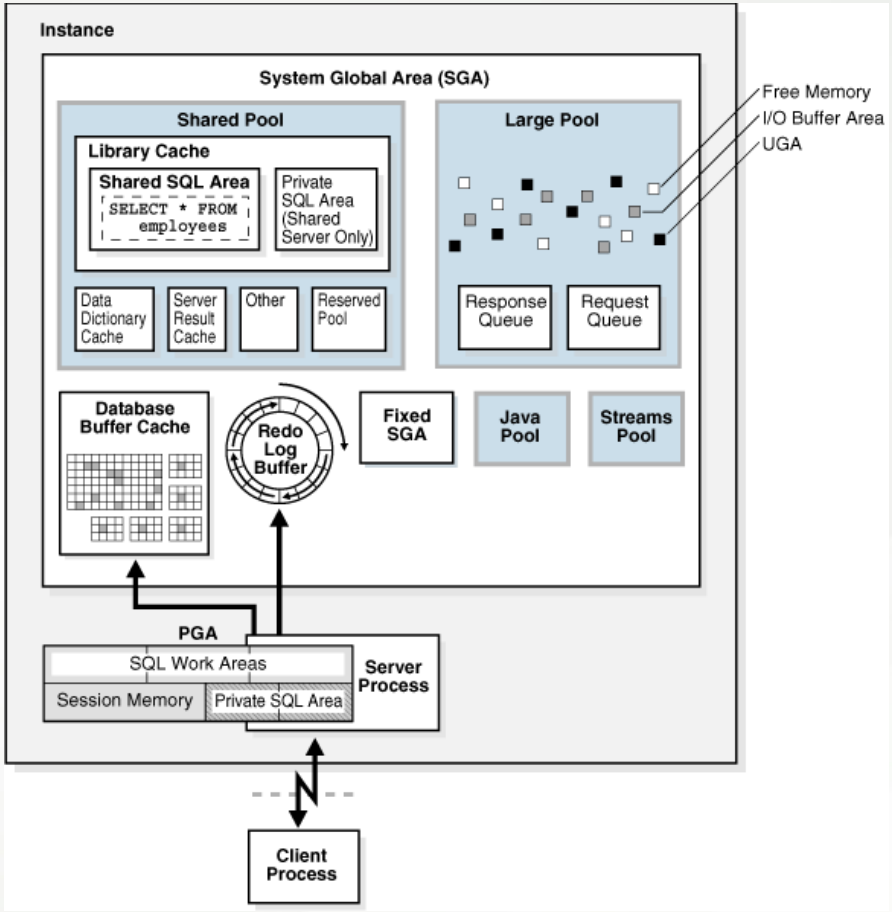
Lab 2.15: SQL Trace (Event 10053)

```
Bug Fix Control Environment
    fix  3834770 = 1
    fix  3746511 = enabled
    fix  4519016 = enabled
    fix  3118776 = enabled
    fix  4488689 = enabled
    fix  2194204 = disabled
    fix  2660592 = enabled
    fix  2320291 = enabled
    fix  2324795 = enabled
    fix  4308414 = enabled
    fix  3499674 = disabled
    fix  4569940 = enabled
    fix  4631959 = enabled
    fix  4519340 = enabled
    fix  4550003 = enabled
    fix  1403283 = enabled
    fix  4554846 = enabled
    fix  4602374 = enabled
    fix  4584065 = enabled
    fix  4545833 = enabled
    fix  4611850 = enabled
    fix  4663698 = enabled
    fix  4663804 = enabled
    fix  4666174 = enabled
    fix  4567767 = enabled
    fix  4556762 = 15
    fix  4728348 = enabled
```

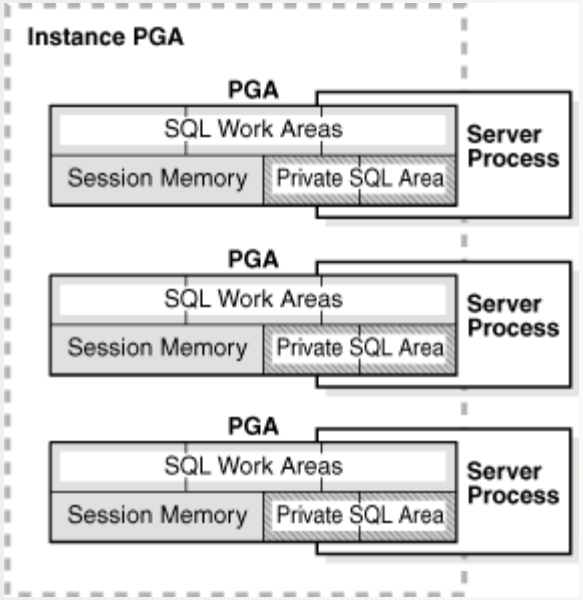
SQL Engine

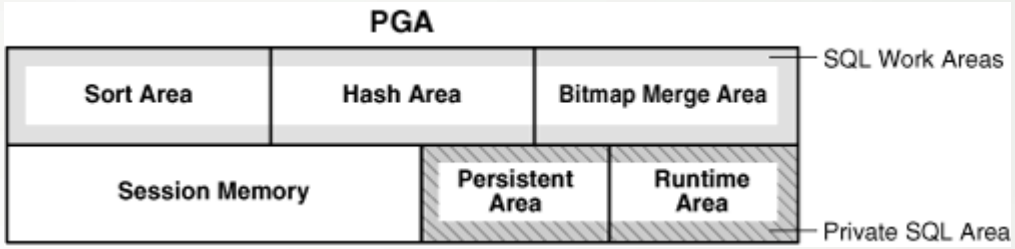
Terminologia

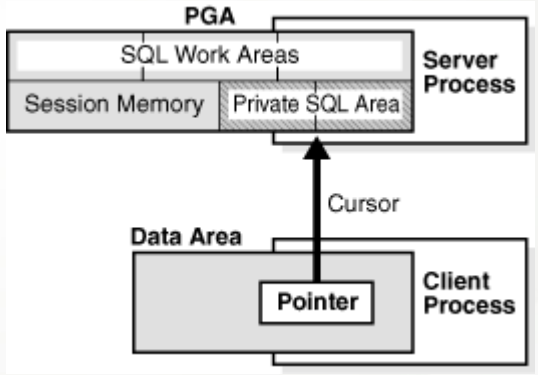
- Soft Parse / Hard Parse
- LIO (Logical Input/Output)
- PIO (Physical Input/Output)
- Latch / Mutex
- Buffer Cache
- Shared Pool
- Library Cache



PGA


Instance PGA
PGA
SQL Work Areas
Session Memory
Private SQL Area
Server Process
PGA
SQL Work Areas
Session Memory
Private SQL Area
Server Process
PGA
SQL Work Areas
Session Memory
Private SQL Area
Server Process


PGA
Sort Area
Hash Area
Bitmap Merge Area
SQL Work Areas
Session Memory
Persistent Area
Runtime Area
Private SQL Area


PGA
SQL Work Areas
Session Memory
Private SQL Area
Server Process
Cursor
Data Area
Pointer
Client Process

SELECT

5: SELECT (COLUMN / DISTINCT COLUMN / expression / scalar subquery)
1: FROM / FROM JOIN ON (fontes: TABLE, VIEW, MVIEW, PARTITION, SUBQUERY...)
2: * WHERE (condições: TRUE, FALSE, UNKNOWN)
3: * GROUP BY (opções: ROLLUP / CUBE)
4: * HAVING (condição: TRUE)
6: * ORDER BY (COLUMN)

Exemplo:

```
SELECT C.CUSTOMER_ID, COUNT(O.ORDER_ID) AS ORDER_CT
    FROM    OE.CUSTOMERS C
            JOIN OE.ORDERS O ON C.CUSTOMER_ID = O.CUSTOMER_ID
    WHERE C.GENDER = 'F'
    GROUP BY C.CUSTOMER_ID
    HAVING COUNT(O.ORDER_ID) > 4
    ORDER BY ORDERS_CT, C_CUSTOMER_ID;
```

- Na fase 2, os dados já foram selecionados (IN MEMORY Column Store).
- Qual a diferença de processamento entre WHERE e HAVING?

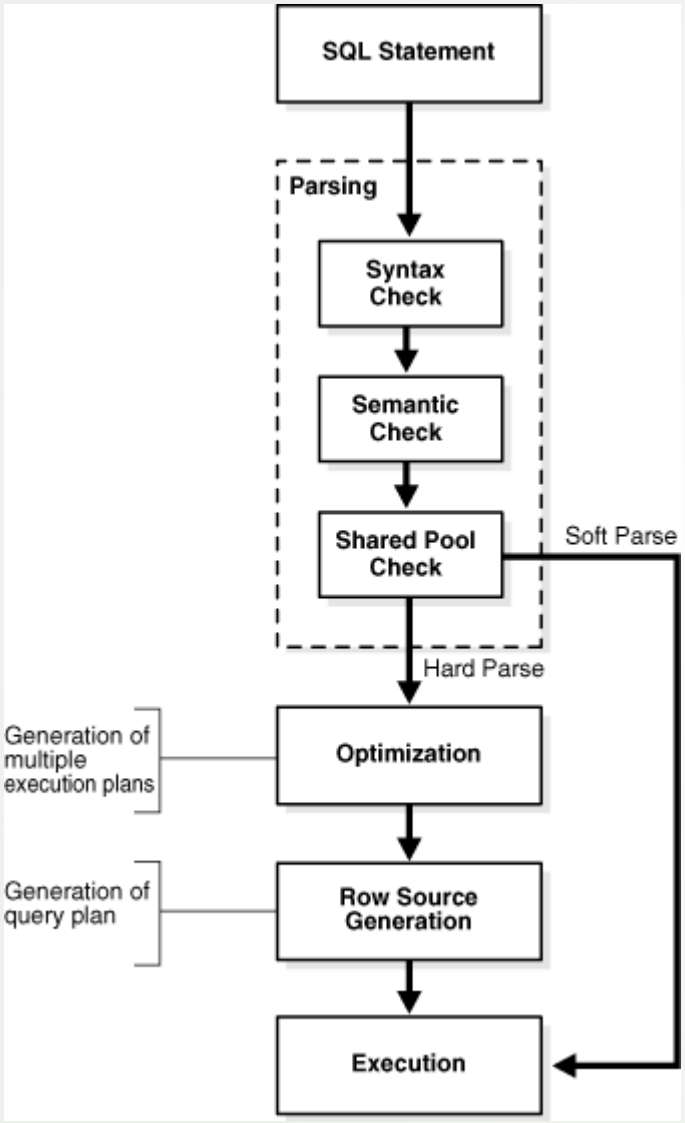
Processamento de SQL

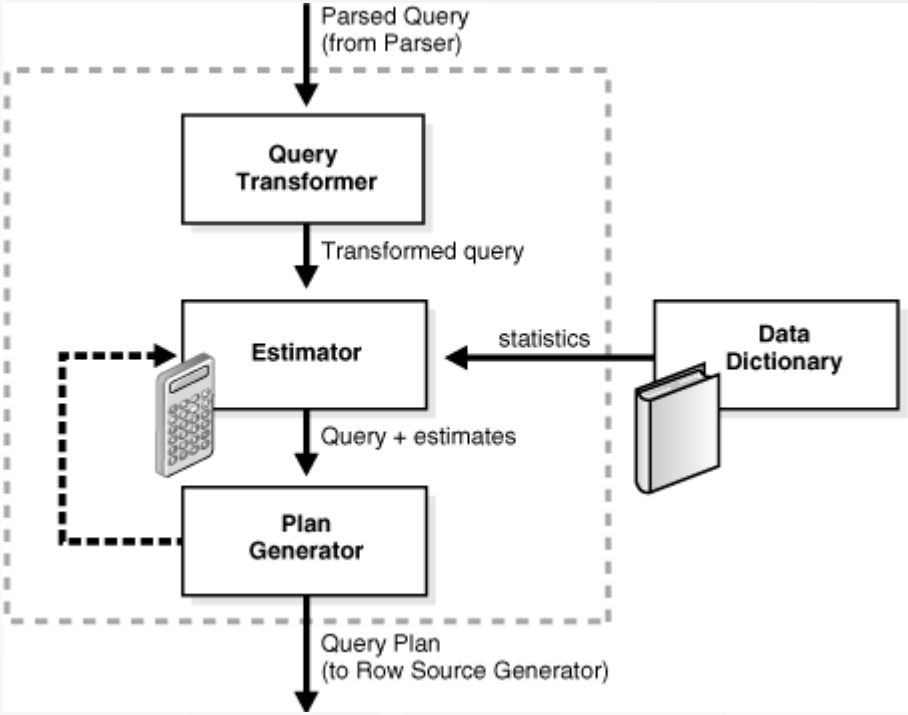
- **OPEN CURSOR**
- **PARSE**
- **BIND**
- **EXEC**
- **FETCH** (ARRAYSIZE, DEFAULT_SDU_SIZE (512 a 32767), RECV_BUF_SIZE, SEND_BUF_SIZE (sqlnet.ora)
- **CLOSE CURSOR**

```
SQL*Net Message From Client
PARSE
SQL*Net Message To Client
SQL*Net Message From Client
BIND
SQL*Net Message To Client
SQL*Net Message From Client
EXEC
SQL*Net Message To Client
SQL*Net Message From Client
FETCH
SQL*Net Message To Client
SQL*Net Message From Client
FETCH
SQL*Net Message To Client
SQL*Net Message From Client
...
```

Apenas SELECT possui a fase FETCH.

Processamento de SQL

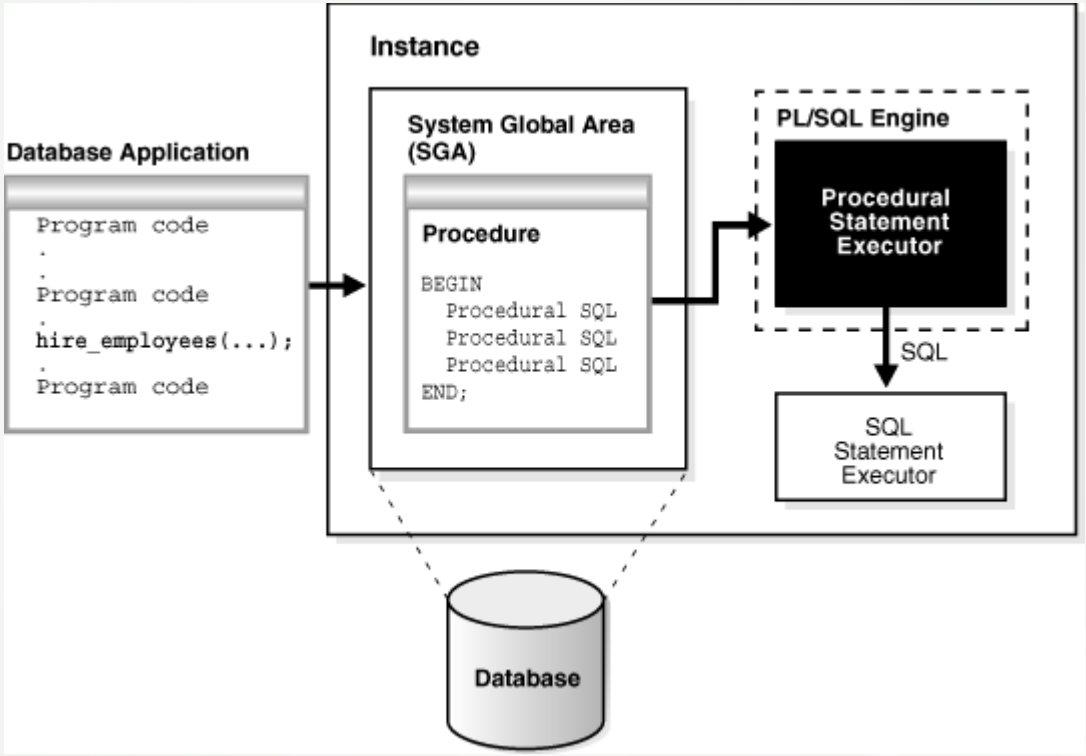




Parse Time

12.1.0.2.1	21.624 seconds
12.1.0.2	18.772 seconds
12.1.0.1	19.846 seconds
11.2.0.4	19.083 seconds
11.2.0.3	26.377 seconds
11.2.0.2	29.643 seconds
11.2.0.1	32.280 seconds
11.1.0.7	29.749 seconds
11.1.0.6	27.158 seconds
10.2.0.5	7.361 seconds
10.2.0.4	7.290 seconds
10.2.0.3	6.530 seconds
10.2.0.2	7.164 seconds
10.2.0.1	5.382 seconds
10.1.0.5	3.811 seconds
10.1.0.4	4.032 seconds
10.1.0.3	.238 seconds
10.1.0	.208 seconds
9.2.0.8	.315 seconds
9.2.0	.249 seconds
9.0.1	.594 seconds
9.0.0	.778 seconds
8.1.7	.667 seconds
8.1.6	.343 seconds
8.1.5	.257 seconds
8.1.4	.255 seconds
8.1.3	.220 seconds
8.1.0	.204 seconds
8.0.7	.313 seconds
8.0.6	.303 seconds
8.0.5	.296 seconds
8.0.4	.300 seconds
8.0.3	.232 seconds
8.0.0	.211 seconds

PL/SQL Engine



SQL Recursivos

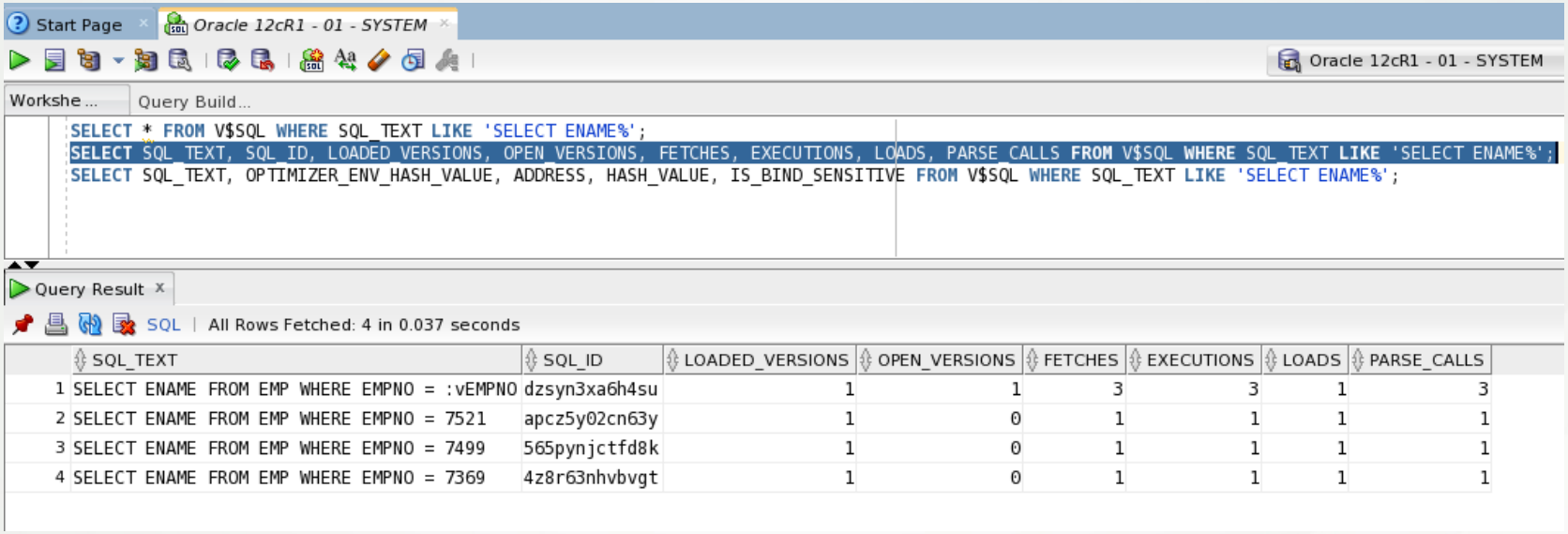
```
select name,password,datats#,tempts#,type#,defrole,resource$, ptime, decode(defschclass,NULL,'DEFAULT_CONSUMER_GROUP',c
select /*+ rule */ bucket_cnt, row_cnt, cache_cnt, null_cnt, timestamp#, sample_size, minimum, maximum, distcnt, lowval
select /*+ rule */ bucket, endpoint, col#, epvalue, epvalue_raw, ep_repeat_count from histgrm$ where obj#=:1 and intcol
select user#,password,datats#,tempts#,type#,defrole,resource$, ptime,decode(defschclass,NULL,'DEFAULT_CONSUMER_GROUP',c
select obj#,type#,ctime,mtime,stime, status, dataobj#, flags, oid$, spare1, spare2, spare3, signature, spare7, spare8,
select obj#,type#,ctime,mtime,stime, status, dataobj#, flags, oid$, spare1, spare2, spare3, signature, spare7, spare8,
select name,online$,contents$,undofile#,undoblock#,blocksize,dflmaxext,dflinit,dflincr,dflextpct,dflminext, dflminlen,
select type#,blocks,extents,minexts,maxexts,extsize,extpct,user#,iniexts,NVL(lists,65535),NVL(groups,65535),cachehint,h
select audit$ from tab$ where obj# = :1
select i.obj#,i.ts#,i.file#,i.block#,i.intcols,i.type#,i.flags,i.property,i.pctfree$,i.initrans,i.maxtrans,i.blevel,i.l
select name,intcol#,segcol#,type#,length,nvl(precision#,0),decode(type#,2,nvl(scale,-127/*MAXSB1MINAL*/),178,scale,179,
select policy#, action# from aud_object_opt$ where object# = :1 and type = 2
select pos#,intcol#,col#,spare1,bo#,spare2,spare3 from icol$ where obj#=:1
select o.owner#,o.name,o.namespace,o.remoteowner,o.linkname,o.subname from obj$ o where o.obj#=:1
select col#, grantee#, privilege#,max(mod(nvl(option$,0),2)), min(bitand(nvl(option$,0), 4) /4), max(bitand(nvl(option$
select grantee#,privilege#,nvl(col#,0),max(mod(nvl(option$,0),2)),min(bitand(nvl(option$,0), 4) /4), max(bitand(nvl(opt
select con#,obj#,rcon#,enabled,nvl(defer,0),spare2,spare3,refact from cdef$ where robj#=:1
select con#,type#,condlength,intcols,robj#,rcon#,match#,refact,nvl(enabled,0),rowid,cols,nvl(defer,0),mtime,nvl(spare1,
select intcol#,nvl(pos#,0),col#,nvl(spare1,0) from ccol$ where con#=:1
select /*+ index(do) */ dir_cnt from opt_directive_own$ do where dir_own# = :1
SELECT /* OPT_DYN_SAMP */ /*+ ALL_ROWS IGNORE_WHERE_CLAUSE NO_PARALLEL(SAMPLESUB) opt_param('parallel_execution_enabled
SELECT * FROM DEPT
```

SQL Recursivos

```
*** 2015-02-16 15:57:09.994
WAIT #139730647289880: nam='SQL*Net message from client' ela= 11311699 driver id=1650815232 #bytes=1 p3=0 obj#=93201 tim=116253102485
CLOSE #139730647289880:c=0,e=21,dep=0,type=0,tim=116253102705
=====================
PARSING IN CURSOR #139730647289880 len=18 dep=0 uid=110 oct=3 lid=110 tim=116253102883 hv=192327475 ad='931f45e0' sqlid='0ws4zp45rdbtm'
SELECT * FROM DEPT
END OF STMT
PARSE #139730647289880:c=0,e=76,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,plh=3383998547,tim=116253102882
EXEC #139730647289880:c=0,e=55,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,plh=3383998547,tim=116253103107
WAIT #139730647289880: nam='SQL*Net message to client' ela= 11 driver id=1650815232 #bytes=1 p3=0 obj#=93201 tim=116253103249
FETCH #139730647289880:c=0,e=122,p=0,cr=5,cu=0,mis=0,r=1,dep=0,og=1,plh=3383998547,tim=116253103459
WAIT #139730647289880: nam='SQL*Net message from client' ela= 375 driver id=1650815232 #bytes=1 p3=0 obj#=93201 tim=116253103932
WAIT #139730647289880: nam='SQL*Net message to client' ela= 6 driver id=1650815232 #bytes=1 p3=0 obj#=93201 tim=116253104067
FETCH #139730647289880:c=0,e=126,p=0,cr=3,cu=0,mis=0,r=3,dep=0,og=1,plh=3383998547,tim=116253104169
STAT #139730647289880 id=1 cnt=4 pid=0 pos=1 obj=93201 op='TABLE ACCESS FULL DEPT (cr=8 pr=0 pw=0 time=114 us cost=3 size=80 card=4)'
```

```
*** 2015-02-16 15:58:04.831
WAIT #139730647289880: nam='SQL*Net message from client' ela= 54835037 driver id=1650815232 #bytes=1 p3=0 obj#=93201 tim=116307939395
CLOSE #139730647289880:c=0,e=29,dep=0,type=1,tim=116307939619
=====================
PARSING IN CURSOR #139730646254864 len=31 dep=0 uid=110 oct=3 lid=110 tim=116307941824 hv=920782030 ad='9331b9e8' sqlid='dthjv3hvf416f'
SELECT * FROM DEPT ORDER BY 1,2
END OF STMT
PARSE #139730646254864:c=1000,e=2127,p=0,cr=0,cu=0,mis=1,r=0,dep=0,og=1,plh=120787663,tim=116307941823
EXEC #139730646254864:c=0,e=69,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,plh=120787663,tim=116307942100
WAIT #139730646254864: nam='SQL*Net message to client' ela= 11 driver id=1650815232 #bytes=1 p3=0 obj#=93201 tim=116307942283
FETCH #139730646254864:c=0,e=140,p=0,cr=7,cu=0,mis=0,r=1,dep=0,og=1,plh=120787663,tim=116307942503
WAIT #139730646254864: nam='SQL*Net message from client' ela= 307 driver id=1650815232 #bytes=1 p3=0 obj#=93201 tim=116307942894
WAIT #139730646254864: nam='SQL*Net message to client' ela= 4 driver id=1650815232 #bytes=1 p3=0 obj#=93201 tim=116307942982
FETCH #139730646254864:c=0,e=87,p=0,cr=0,cu=0,mis=0,r=3,dep=0,og=1,plh=120787663,tim=116307943058
STAT #139730646254864 id=1 cnt=4 pid=0 pos=1 obj=0 op='SORT ORDER BY (cr=7 pr=0 pw=0 time=142 us cost=4 size=80 card=4)'
STAT #139730646254864 id=2 cnt=4 pid=1 pos=1 obj=93201 op='TABLE ACCESS FULL DEPT (cr=7 pr=0 pw=0 time=104 us cost=3 size=80 card=4)'
```

Hard Parse / Soft Parse



	SQL_TEXT	SQL_ID	LOADED_VERSIONS	OPEN_VERSIONS	FETCHES	EXECUTIONS	LOADS	PARSE_CALLS
1	SELECT ENAME FROM EMP WHERE EMPNO = :vEMPNO	dzsyn3xa6h4su	1	1	3	3	1	3
2	SELECT ENAME FROM EMP WHERE EMPNO = 7521	apcz5y02cn63y	1	0	1	1	1	1
3	SELECT ENAME FROM EMP WHERE EMPNO = 7499	565pynjctfd8k	1	0	1	1	1	1
4	SELECT ENAME FROM EMP WHERE EMPNO = 7369	4z8r63nhvbvgt	1	0	1	1	1	1

Hard Parse / Soft Parse

Workshe ... Query Build...

```
SELECT * FROM V$SQL WHERE SQL_TEXT LIKE 'SELECT ENAME%';
SELECT SQL_TEXT, SQL_ID, LOADED_VERSIONS, OPEN_VERSIONS, FETCHES, EXECUTIONS, LOADS, PARSE_CALLS FROM V$SQL WHERE SQL_TEXT LIKE '
SELECT SQL_TEXT, OPTIMIZER_ENV_HASH_VALUE, ADDRESS, HASH_VALUE, IS_BIND_SENSITIVE FROM V$SQL WHERE SQL_TEXT LIKE 'SELECT ENAME%';
```

Query Result

SQL | All Rows Fetched: 4 in 0.035 seconds

	SQL_TEXT	OPTIMIZER_ENV_HASH_VALUE	ADDRESS	HASH_VALUE	IS_BIND_SENSITIVE
1	SELECT ENAME FROM EMP WHERE EMPNO = :vEMPNO	1557828867	0000000075C96960	1416106778	Y
2	SELECT ENAME FROM EMP WHERE EMPNO = 7521	1557828867	00000000725922C0	80353406	N
3	SELECT ENAME FROM EMP WHERE EMPNO = 7499	1557828867	0000000077B53980	1503081746	N
4	SELECT ENAME FROM EMP WHERE EMPNO = 7369	1557828867	0000000077A3A280	565571065	N

Hard Parse

```
SQL> SELECT /* Quero achar este SQL */ COUNT(*) FROM T1;
SQL> SELECT  /* Quero achar este SQL também */ COUNT(*) FROM T1;
SQL> SELECT   /* Quero achar este SQL */   COUNT(*)   FROM   T1;
SQL> select /* Quero achar este SQL */ count(*) from t1;
SQL> SELECT /* Quero achar este SQL também */ COUNT(*) FROM T1;

SQL> SELECT SQL_ID, SQL_TEXT FROM V$SQL WHERE SQL_TEXT LIKE '%Quero achar este SQL%';
```

Lab 3 – SQL Engine

Hands On !

Lab 3.1: FETCH

Execute os comandos abaixo.

```
SQL> CONN HR/HR
SQL> SET AUTOTRACE TRACEONLY STATISTICS
SQL> SELECT * FROM EMPLOYEES;

107 rows selected.
Statistics
----------------------------------------------------------
        1  recursive calls
        0  db block gets
       15  consistent gets
        6  physical reads
        0  redo size
    10499  bytes sent via SQL*Net to client
      629  bytes received via SQL*Net from client
        9  SQL*Net roundtrips to/from client
        0  sorts (memory)
        0  sorts (disk)
      107  rows processed
```

Lab 3.2: FETCH

Execute os mesmo SELECT anterior, mas com o ARRAYSIZE diferente.

```
SQL> SET ARRAYSIZE 100
SQL> SELECT * FROM EMPLOYEES;

107 rows selected.


Statistics
----------------------------------------------------------
      0  recursive calls
      0  db block gets
      9  consistent gets
      0  physical reads
      0  redo size
   9335  bytes sent via SQL*Net to client
    563  bytes received via SQL*Net from client
      3  SQL*Net roundtrips to/from client
      0  sorts (memory)
      0  sorts (disk)
    107  rows processed
```

Lab 3.3: Shared Pool

Reinicie a instância, e execute os comandos abaixo.

```
SQL> CONN / AS SYSDBA
SQL> SHUTDOWN IMMEDIATE
SQL> STARTUP
SQL> CONN SCOTT/TIGER

SQL> SET AUTOTRACE TRACEONLY STATISTICS
SQL> SELECT ENAME FROM EMP WHERE EMPNO = 7369;
    104  recursive calls
      0  db block gets
    164  consistent gets
      7  physical reads
      0  redo size
    550  bytes sent via SQL*Net to client
    551  bytes received via SQL*Net from client
      2  SQL*Net roundtrips to/from client
      9  sorts (memory)
      0  sorts (disk)
      1  rows processed
```

Lab 3.4: Shared Pool

Execute novamente o mesmo SELECT.

```
SQL> SELECT ENAME FROM EMP WHERE EMPNO = 7369;
      0  recursive calls
      0  db block gets
      2  consistent gets
      0  physical reads
      0  redo size
    550  bytes sent via SQL*Net to client
    551  bytes received via SQL*Net from client
      2  SQL*Net roundtrips to/from client
      0  sorts (memory)
      0  sorts (disk)
      1  rows processed
```

Lab 3.5: Shared Pool

Execute o mesmo SQL, duas vezes, mas solicitando outro registro.

```
SQL> SELECT ENAME FROM EMP WHERE EMPNO = 7499;
     36  recursive calls
      0  db block gets
     53  consistent gets
      0  physical reads
      0  redo size
    550  bytes sent via SQL*Net to client
    551  bytes received via SQL*Net from client
      2  SQL*Net roundtrips to/from client
      5  sorts (memory)
      0  sorts (disk)
      1  rows processed
SQL> SELECT ENAME FROM EMP WHERE EMPNO = 7499;
      0  recursive calls
      0  db block gets
      2  consistent gets
      0  physical reads
      0  redo size
    550  bytes sent via SQL*Net to client
    551  bytes received via SQL*Net from client
      2  SQL*Net roundtrips to/from client
      0  sorts (memory)
      0  sorts (disk)
      1  rows processed
```

Lab 3.6: Shared Pool

Execute o mesmo SQL, também duas vezes, mas solicitando outro registro.

```
SQL> SELECT ENAME FROM EMP WHERE EMPNO = 7521;
      1  recursive calls
      0  db block gets
      2  consistent gets
      0  physical reads
      0  redo size
    549  bytes sent via SQL*Net to client
    551  bytes received via SQL*Net from client
      2  SQL*Net roundtrips to/from client
      0  sorts (memory)
      0  sorts (disk)
      1  rows processed
SQL> SELECT ENAME FROM EMP WHERE EMPNO = 7521;
      0  recursive calls
      0  db block gets
      2  consistent gets
      0  physical reads
      0  redo size
    549  bytes sent via SQL*Net to client
    551  bytes received via SQL*Net from client
      2  SQL*Net roundtrips to/from client
      0  sorts (memory)
      0  sorts (disk)
      1  rows processed
```

Lab 3.7: Shared Pool

Execute o seguinte SQL, também duas vezes.

```
SQL> SELECT ENAME FROM EMP WHERE EMPNO IN (7654, 7698, 7782);
      5  recursive calls
      0  db block gets
     13  consistent gets
      2  physical reads
      0  redo size
    619  bytes sent via SQL*Net to client
    551  bytes received via SQL*Net from client
      2  SQL*Net roundtrips to/from client
      0  sorts (memory)
      0  sorts (disk)
      3  rows processed
SQL> SELECT ENAME FROM EMP WHERE EMPNO IN (7654, 7698, 7782);
      0  recursive calls
      0  db block gets
      4  consistent gets
      0  physical reads
      0  redo size
    619  bytes sent via SQL*Net to client
    551  bytes received via SQL*Net from client
      2  SQL*Net roundtrips to/from client
      0  sorts (memory)
      0  sorts (disk)
      3  rows processed
```

Lab 3.8: Shared Pool

Remova as estatísticas da tabela, e execute o primeiro SQL, também duas vezes.

```
SQL> EXEC DBMS_STATS.DELETE_TABLE_STATS('SCOTT', 'EMP');
SQL> SELECT ENAME FROM EMP WHERE EMPNO = 7369;
      0  recursive calls
      0  db block gets
      2  consistent gets
     16  physical reads
      0  redo size
...
      0  sorts (memory)
      0  sorts (disk)
      1  rows processed
SQL> SELECT ENAME FROM EMP WHERE EMPNO = 7369;
      0  recursive calls
      0  db block gets
      2  consistent gets
      0  physical reads
      0  redo size
...
      0  sorts (memory)
      0  sorts (disk)
      1  rows processed
```

Lab 3.9: Shared Pool

Execute o SQL abaixo, que retorna mais dados, também duas vezes.

```
SQL> SELECT * FROM DEPT;
      40  recursive calls
       0  db block gets
      88  consistent gets
      41  physical reads
       0  redo size
...
       6  sorts (memory)
       0  sorts (disk)
       4  rows processed

SQL> SELECT * FROM DEPT;
       0  recursive calls
       0  db block gets
       8  consistent gets
       0  physical reads
       0  redo size
...
       0  sorts (memory)
       0  sorts (disk)
       4  rows processed
```

Lab 3.10: Shared Pool

Limpe a Shared Pool, e execute novamente o SQL, também duas vezes.

```
SQL> CONN / AS SYSDBA
SQL> ALTER SYSTEM FLUSH SHARED_POOL;

SQL> CONN SCOTT/TIGER
SQL> SET AUTOTRACE TRACEONLY STATISTICS
SQL> SELECT * FROM DEPT;
1a execução:
      71  recursive calls
       0  db block gets
      90  consistent gets
       0  physical reads
       0  redo size
...
       6  sorts (memory)
       0  sorts (disk)
       4  rows processed

SQL> SELECT * FROM DEPT;
       0  recursive calls
       0  db block gets
       8  consistent gets
       0  physical reads
       0  redo size
...
       0  sorts (memory)
       0  sorts (disk)
       4  rows processed
```

Lab 3.11: Shared Pool

Execute novamente o primeiro SQL, mas com variáveis Bind, também duas vezes.

```
SQL> VARIABLE vEMPNO NUMBER
SQL> EXEC :vEMPNO := 7369
SQL> SELECT ENAME FROM EMP WHERE EMPNO = :vEMPNO;
      0  recursive calls
      0  db block gets
      2  consistent gets
      0  physical reads
      0  redo size
...
      1  rows processed

SQL> EXEC :vEMPNO := 7499
SQL> SELECT ENAME FROM EMP WHERE EMPNO = :vEMPNO;
      0  recursive calls
      0  db block gets
      2  consistent gets
      0  physical reads
      0  redo size
...
      1  rows processed
```

Lab 3.12: Hard Parse / Soft Parse

Execute novamente o SELECT na DEPT, verificado os Parses.

```
SQL> SET AUTOTRACE OFF
SQL> SELECT S.NAME, M.VALUE FROM V$STATNAME S, V$MYSTAT M WHERE S.STATISTIC# =
M.STATISTIC# AND S.NAME IN ('parse count (total)', 'parse count (hard)');
parse count (total)                                30
parse count (hard)                                  6

SQL> SELECT * FROM DEPT;
...
parse count (total)                                32
parse count (hard)                                  6

SQL> SELECT * FROM DEPT;
...
parse count (total)                                34
parse count (hard)                                  6

SQL> SELECT * FROM DEPT ORDER BY 1;
...
parse count (total)                                43
parse count (hard)                                  7

SQL> SELECT * FROM DEPT ORDER BY 1,2;
...
parse count (total)                                45
parse count (hard)                                  8
```

Lab 3.13: Buffer Cache

Limpe o Buffer Cache, e execute novamente o SQL da DEPT, também duas vezes.

```
SQL> CONN / AS SYSDBA
SQL> ALTER SYSTEM FLUSH BUFFER_CACHE;

SQL> CONN SCOTT/TIGER
SQL> SET AUTOTRACE TRACEONLY STATISTICS
SQL> SELECT * FROM DEPT;
1a execução:
      0  recursive calls
      0  db block gets
      8  consistent gets
      6  physical reads
      0  redo size
...
      0  sorts (memory)
      0  sorts (disk)
      4  rows processed

2a execução:
      0  recursive calls
      0  db block gets
      8  consistent gets
      0  physical reads
      0  redo size
...
      0  sorts (memory)
      0  sorts (disk)
      4  rows processed
```

Lab 3.14: Connect / Parse / Commit

Crie a tabela abaixo com o usuário SCOTT.

```
SQL> CREATE TABLE T314 (C1 NUMBER);
```

Observe o conteúdo dos seguintes scripts Perl, os execute, e compare.

```
$ time perl /home/oracle/ConnectBAD_CommitBAD_BindsBAD.pl 10000
$ time perl /home/oracle/ConnectBAD_CommitBAD_BindsGOOD.pl 10000
$ time perl /home/oracle/ConnectBAD_CommitGOOD_BindsBAD.pl 10000
$ time perl /home/oracle/ConnectBAD_CommitGOOD_BindsGOOD.pl 10000
$ time perl /home/oracle/ConnectGOOD_CommitBAD_BindsBAD.pl 10000
$ time perl /home/oracle/ConnectGOOD_CommitBAD_BindsGOOD.pl 10000
$ time perl /home/oracle/ConnectGOOD_CommitGOOD_BindsBAD.pl 10000
$ time perl /home/oracle/ConnectGOOD_CommitGOOD_BindsBAD_ONE.pl 10000
$ time perl /home/oracle/ConnectGOOD_CommitGOOD_BindsGOOD.pl 10000
$ time perl /home/oracle/ConnectGOOD_CommitGOOD_BindsGOOD_PERFECT.pl 10000
```

Re-execute os testes ConnectGOOD com os parâmetros abaixo alterados.

```
ALTER SYSTEM SET CURSOR_SHARING=FORCE;
ALTER SYSTEM SET COMMIT_LOGGING=BATCH;
```

Lab 3.15: PL/SQL Engine

Crie esta tabela com o usuário SCOTT:

```
SQL> CREATE TABLE T315 (C1 NUMBER);
```

Observe o conteúdo dos seguintes scripts Perl, os execute, e compare:

```
$ time perl /home/oracle/SemPLSQL.pl 10000
$ time perl /home/oracle/ComPLSQL.pl 10000
```

Access Paths

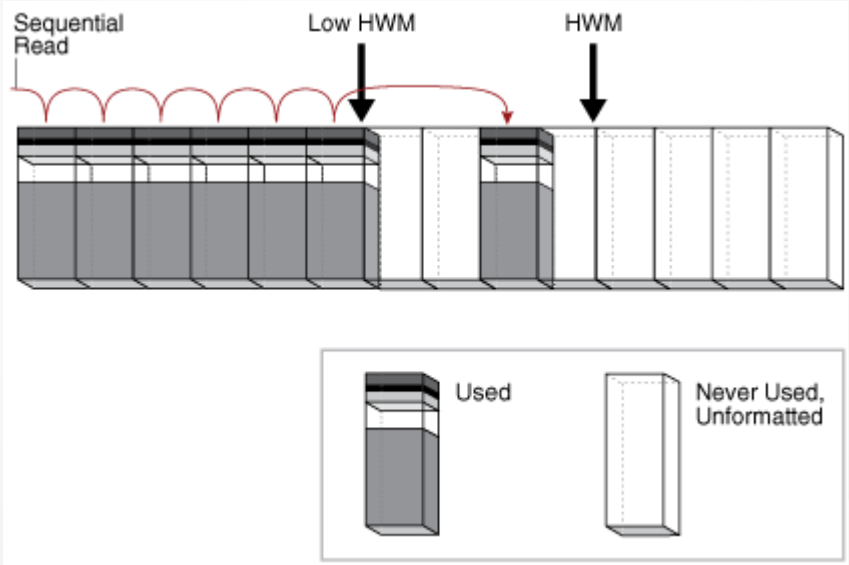
Access Paths

- Full Table Scan (FTS)
- Table Access by ROWID
- Index Unique Scan
- Index Range Scan
- Index Range Scan descending
- Index Skip Scan
- Full Index Scan (FIS)
- Fast Full Index Scan (FFIS)

Full Table Scan

Depende de:

- Percentual de dados que serão acessados;
- Quantidade de blocos lidos em leituras múltiplas (db_file_multiblock_read_count);
- Velocidade de leitura de múltiplos blocos x único bloco (System Statistics);
- Distribuição das linhas nos blocos;
- HWM.



Lab 4.1: FTS e Clustering Factor

Crie as duas tabelas abaixo com o usuário SCOTT, e compare as duas.

```
SQL> CONN SCOTT/TIGER

SQL> CREATE TABLE T1 AS
    SELECT TRUNC((ROWNUM-1)/100) ID, RPAD(ROWNUM,100) NAME
    FROM DBA_SOURCE
    WHERE ROWNUM <= 10000;
SQL> CREATE INDEX T1_IDX1 ON T1(ID);

SQL> CREATE TABLE T2 AS
    SELECT MOD(ROWNUM,100) ID, RPAD(ROWNUM,100) NAME
    FROM DBA_SOURCE
    WHERE ROWNUM <= 10000;
SQL> CREATE INDEX T2_IDX1 ON T2(ID);

SQL> SELECT COUNT(*) FROM T1;
SQL> SELECT COUNT(*) FROM T2;
SQL> SELECT MIN(ID) FROM T1;
SQL> SELECT MIN(ID) FROM T2;
SQL> SELECT MAX(ID) FROM T1;
SQL> SELECT MAX(ID) FROM T2;
SQL> SELECT COUNT(*) FROM T1 WHERE ID = 1;
SQL> SELECT COUNT(*) FROM T2 WHERE ID = 1;
```

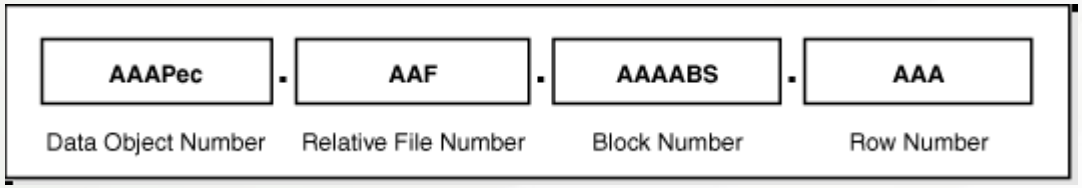
Lab 4.2: FTS e Clustering Factor

Compare os planos de execução de SQL iguais para as duas tabelas.

```
SQL> SET AUTOTRACE TRACEONLY EXPLAIN
SQL> SELECT ID, NAME FROM T1 WHERE ID = 1;
SQL> SELECT ID, NAME FROM T2 WHERE ID = 1;
SQL> SELECT ID, NAME FROM T1 WHERE ID < 5;
SQL> SELECT ID, NAME FROM T2 WHERE ID < 5;
SQL> SELECT ID, NAME FROM T1 WHERE ID < 10;
SQL> SELECT ID, NAME FROM T2 WHERE ID < 10;
```

Verifique a ordenação física dos dados das tabelas.

```
SQL> SET AUTOTRACE OFF
SQL> SELECT ID, NAME FROM T1;
SQL> SELECT ID, NAME FROM T2;
SQL> SELECT ROWID, ID, NAME FROM T1 ORDER BY 2;
SQL> SELECT ROWID, ID, NAME FROM T2 ORDER BY 2;
```



Lab 4.3: FTS e Clustering Factor

Compare as estatísticas das duas tabelas.

```
SQL> SET AUTOTRACE OFF
SQL> COL TABLE_NAME FORMAT A20
SQL> COL INDEX_NAME FORMAT A20

SQL> SELECT
       T.TABLE_NAME,
       I.INDEX_NAME,
       I.CLUSTERING_FACTOR,
       T.BLOCKS,
       T.NUM_ROWS
   FROM DBA_TABLES T, DBA_INDEXES I
   WHERE  T.TABLE_NAME = I.TABLE_NAME AND
          T.TABLE_NAME IN ('T1', 'T2') AND
          T.OWNER = 'SCOTT'
          ORDER BY T.TABLE_NAME, I.INDEX_NAME;
```

Lab 4.4: FTS e HWM

Compare as estatísticas deste SELECT, antes e depois do DELETE.

```
SQL> CREATE TABLE T4 AS SELECT * FROM ALL_OBJECTS;
SQL> SET AUTOTRACE TRACEONLY
SQL> SELECT COUNT(*) FROM T4;
SQL> SELECT COUNT(*) FROM T4;
SQL> SELECT COUNT(*) FROM T4;

SQL> DELETE FROM T4;
SQL> SELECT COUNT(*) FROM T4;
SQL> SELECT COUNT(*) FROM T4;
SQL> SELECT COUNT(*) FROM T4;
```

Lab 4.5: FTS e HWM

Verifique os blocos utilizados pela tabela, antes e depois do DELETE.

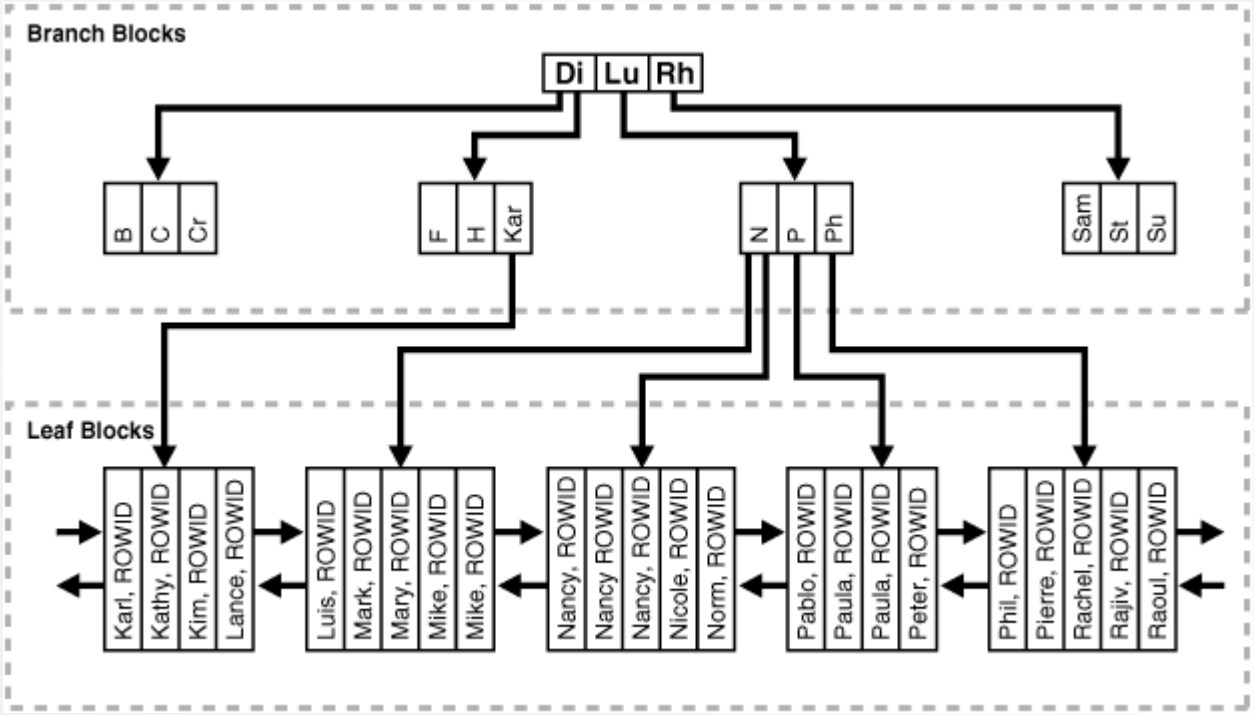
```
SQL> SET AUTOTRACE OFF
SQL> DROP TABLE T4;

SQL> CREATE TABLE T4 AS SELECT * FROM ALL_OBJECTS;
SQL> SELECT MIN(DBMS_ROWID.ROWID_BLOCK_NUMBER(ROWID)) MIN_BLK,
MAX(DBMS_ROWID.ROWID_BLOCK_NUMBER(ROWID)) MAX_BLK FROM T4;

SQL> DELETE FROM T4;
SQL> SELECT MIN(DBMS_ROWID.ROWID_BLOCK_NUMBER(ROWID)) MIN_BLK,
MAX(DBMS_ROWID.ROWID_BLOCK_NUMBER(ROWID)) MAX_BLK FROM T4;

SQL> ROLLBACK;
SQL> SELECT MIN(DBMS_ROWID.ROWID_BLOCK_NUMBER(ROWID)) MIN_BLK,
MAX(DBMS_ROWID.ROWID_BLOCK_NUMBER(ROWID)) MAX_BLK FROM T4;
```

Index Scan

- Index Unique Scan
- Index Range Scan
- Index Skip Scan
- Index Full Scan (*Por que ler todos blocos de um índice E os da tabela, e não só os da tabela?*)
- Index Fast Full Scan

Index Scan

- B-tree = Árvore Balanceada
- Root Block / Branch Blocks / Leaf Blocks
- Height / BEVEL (quando o Height / BLEVEL aumenta?)
- Average Leaf Blocks per Key / Average Data Blocks per Key
- Clustering Factor



Lab 5.1: Index Scan

Compare as estatísticas destes SELECTs.

```
SQL> SET AUTOTRACE TRACEONLY STATISTICS
SQL> SELECT NAME FROM T1 WHERE ID = 0 AND NAME = 1;
      0  recursive calls
      0  db block gets
      6  consistent gets
      0  physical reads
      0  redo size
    636  bytes sent via SQL*Net to client
    552  bytes received via SQL*Net from client
      2  SQL*Net roundtrips to/from client
      0  sorts (memory)
      0  sorts (disk)
      1  rows processed

SQL> SELECT COUNT(NAME) FROM T1 WHERE ID = 0 AND NAME = 1;
      0  recursive calls
      0  db block gets
      4  consistent gets
      0  physical reads
      0  redo size
    545  bytes sent via SQL*Net to client
    552  bytes received via SQL*Net from client
      2  SQL*Net roundtrips to/from client
      0  sorts (memory)
      0  sorts (disk)
      1  rows processed
```

Lab 5.2: Index Scan

Compare as estatísticas deste SELECTs, com os do Lab 5.1.

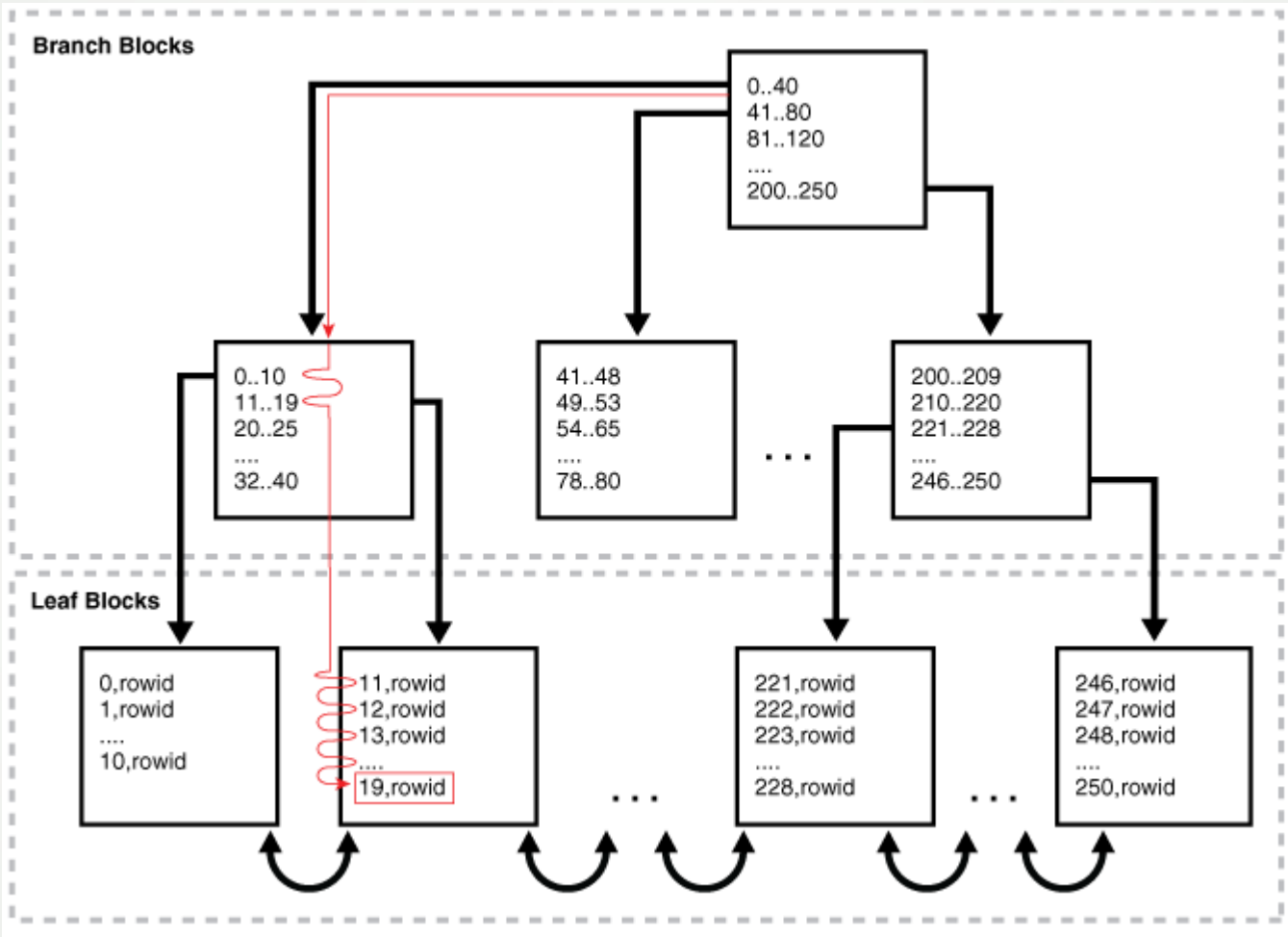
```
SQL> SELECT NAME FROM T2 WHERE ID = 0 AND NAME = 1;
      0  recursive calls
      0  db block gets
    156  consistent gets
      0  physical reads
      0  redo size
    348  bytes sent via SQL*Net to client
    540  bytes received via SQL*Net from client
      1  SQL*Net roundtrips to/from client
      0  sorts (memory)
      0  sorts (disk)
      0  rows processed

SQL> SELECT COUNT(NAME) FROM T2 WHERE ID = 0 AND NAME = 1;
      0  recursive calls
      0  db block gets
    156  consistent gets
      0  physical reads
      0  redo size
    552  bytes sent via SQL*Net to client
    551  bytes received via SQL*Net from client
      2  SQL*Net roundtrips to/from client
      0  sorts (memory)
      0  sorts (disk)
      1  rows processed
```

Lab 5.3: Index Scan

Compare as estatísticas deste SELECTs.

```
SQL> SELECT NAME FROM T1 WHERE ID = 1;
      1  recursive calls
      0  db block gets
      7  consistent gets
      0  physical reads
      0  redo size
  11484  bytes sent via SQL*Net to client
    552  bytes received via SQL*Net from client
      2  SQL*Net roundtrips to/from client
      0  sorts (memory)
      0  sorts (disk)
    100  rows processed

SQL> SELECT NAME FROM T2 WHERE ID = 1;
      1  recursive calls
      0  db block gets
    157  consistent gets
      0  physical reads
      0  redo size
  11176  bytes sent via SQL*Net to client
    552  bytes received via SQL*Net from client
      2  SQL*Net roundtrips to/from client
      0  sorts (memory)
      0  sorts (disk)
    100  rows processed
```

Index Unique Scan

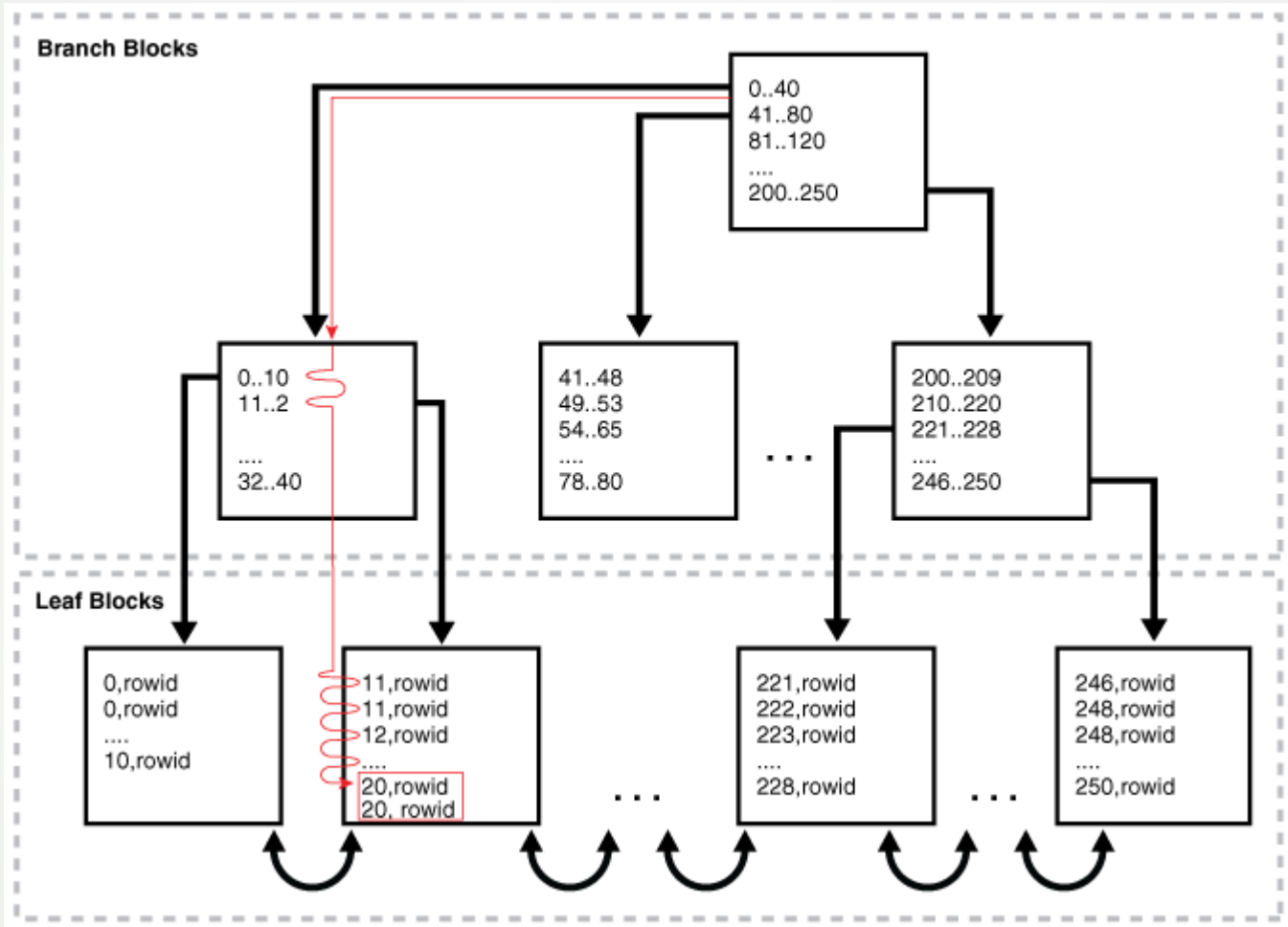


Index Unique Scan

- Utilizado com Primary Key ou Unique Key;
- Consistent Gets mínimo = (Rows x 2) + BLEVEL + 1.

```
HR> SELECT * FROM EMPLOYEES WHERE EMPLOYEE_ID = 100;

Execution Plan
----------------------------------------------------------
Plan hash value: 1833546154

--------------------------------------------------------------------------------------------
| Id  | Operation                   | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |               |     1 |    69 |     0   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| EMPLOYEES     |     1 |    69 |     0   (0)| 00:00:01 |
|*  2 |   INDEX UNIQUE SCAN         | EMP_EMP_ID_PK |     1 |       |     0   (0)| 00:00:01 |
--------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("EMPLOYEE_ID"=100)

HR>
```

Index Range Scan



Index Range Scan

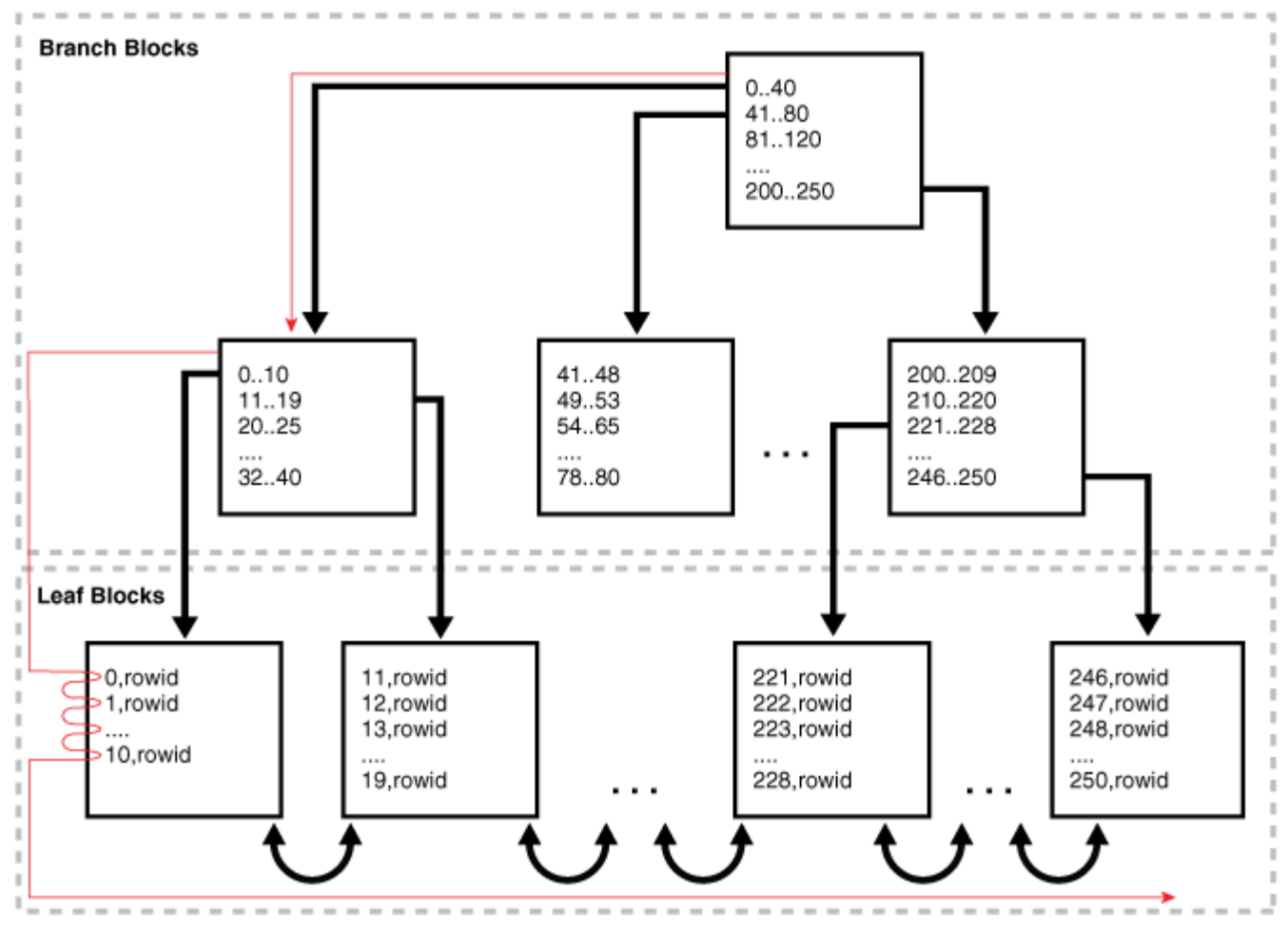
- Utilizado com Primary Key, Unique Key, ou Non-unique Key;
- Consistent Gets mínimo = (Rows x 2) + BLEVEL + 1.

```
HR> SELECT * FROM EMPLOYEES WHERE DEPARTMENT_ID = 60;

Execution Plan
----------------------------------------------------------
Plan hash value: 235881476

-------------------------------------------------------------------------------------------------------
| Id  | Operation                           | Name              | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |                   |     5 |   345 |     1   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID BATCHED| EMPLOYEES         |     5 |   345 |     1   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN                  | EMP_DEPARTMENT_IX |     1 |       |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("DEPARTMENT_ID"=60)

HR>
```

Index Range Scan - Sort

```
HR> SELECT * FROM EMPLOYEES WHERE MANAGER_ID IN (100,103,120) ORDER BY MANAGER_ID;

Execution Plan
----------------------------------------------------------
Plan hash value: 2962718280

-------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                |    26 |  1794 |     0   (0)| 00:00:01 |
|   1 |  INLIST ITERATOR             |                |       |       |            |          |
|   2 |   TABLE ACCESS BY INDEX ROWID| EMPLOYEES      |    26 |  1794 |     0   (0)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN          | EMP_MANAGER_IX |     1 |       |     0   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   3 - access("MANAGER_ID"=100 OR "MANAGER_ID"=103 OR "MANAGER_ID"=120)
```

Index Full Scan



Index Full Scan

Utilizado quando:

- **Não há predicado, mas uma das colunas está indexada;**
- Predicado não é a primeira coluna de um índice;
- Um índice pode economizar um SORT.

```
HR> SELECT EMAIL FROM EMPLOYEES;

Execution Plan
----------------------------------------------------------
Plan hash value: 2196514524

-------------------------------------------------------------------------------------
| Id  | Operation         | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |              |   107 |   856 |     0   (0)| 00:00:01 |
|   1 |  INDEX FULL SCAN  | EMP_EMAIL_UK |   107 |   856 |     0   (0)| 00:00:01 |
-------------------------------------------------------------------------------------

HR>
```

Index Full Scan

- Sem predicado, mas uma das colunas está indexada;
- **Predicado não é a primeira coluna de um índice;**
- Um índice pode economizar um SORT.

```
HR> SELECT FIRST_NAME, LAST_NAME FROM EMPLOYEES WHERE FIRST_NAME LIKE 'J%' OR FIRST_NAME LIKE 'A%';

Execution Plan
----------------------------------------------------------
Plan hash value: 2228653197

-------------------------------------------------------------------------------------
| Id  | Operation         | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |             |     9 |   135 |     0   (0)| 00:00:01 |
|*  1 |  INDEX FULL SCAN  | EMP_NAME_IX |     9 |   135 |     0   (0)| 00:00:01 |
-------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("FIRST_NAME" LIKE 'J%' OR "FIRST_NAME" LIKE 'A%')

HR>
```

Index Full Scan

- Sem predicado, mas uma das colunas está indexada;
- Predicado não é a primeira coluna de um índice;
- **Um índice pode economizar um SORT.**

```
HR> SELECT * FROM EMPLOYEES;

Execution Plan
----------------------------------------------------------
Plan hash value: 1445457117

--------------------------------------------------------------------------------
| Id  | Operation         | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |           |   107 |  7383 |     3   (0)| 00:00:01 |
|   1 |  TABLE ACCESS FULL| EMPLOYEES |   107 |  7383 |     3   (0)| 00:00:01 |
--------------------------------------------------------------------------------
```

```
HR> SELECT * FROM EMPLOYEES ORDER BY EMPLOYEE_ID;

Execution Plan
----------------------------------------------------------
Plan hash value: 2186312383

------------------------------------------------------------------------------------------
| Id  | Operation                   | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |               |   107 |  7383 |     0   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| EMPLOYEES     |   107 |  7383 |     0   (0)| 00:00:01 |
|   2 |   INDEX FULL SCAN           | EMP_EMP_ID_PK |   107 |       |     0   (0)| 00:00:01 |
------------------------------------------------------------------------------------------
```

Index Full Scan

- Sem predicado, mas uma das colunas está indexada;
- Predicado não é a primeira coluna de um índice;
- **Um índice pode economizar um SORT.**

```
HR> SELECT * FROM EMPLOYEES ORDER BY EMPLOYEE_ID DESC;

Execution Plan
----------------------------------------------------------
Plan hash value: 2761389396

-------------------------------------------------------------------------------------------------
| Id  | Operation                   | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |               |   107 |  7383 |     0   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| EMPLOYEES     |   107 |  7383 |     0   (0)| 00:00:01 |
|   2 |   INDEX FULL SCAN DESCENDING| EMP_EMP_ID_PK |   107 |       |     0   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------
```

Index Full Scan

```
HR> SELECT MIN(DEPARTMENT_ID) FROM EMPLOYEES;

Execution Plan
----------------------------------------------------------
Plan hash value: 613773769

-------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name              | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                   |     1 |     3 |     1   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE              |                   |     1 |     3 |            |          |
|   2 |   INDEX FULL SCAN (MIN/MAX)| EMP_DEPARTMENT_IX |     1 |     3 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------

HR> SELECT MAX(DEPARTMENT_ID) FROM EMPLOYEES;

Execution Plan
----------------------------------------------------------
Plan hash value: 613773769

-------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name              | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                   |     1 |     3 |     1   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE              |                   |     1 |     3 |            |          |
|   2 |   INDEX FULL SCAN (MIN/MAX)| EMP_DEPARTMENT_IX |     1 |     3 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------
```

Index Full Scan

```
HR> SELECT MIN(DEPARTMENT_ID), MAX(DEPARTMENT_ID) FROM EMPLOYEES;

Execution Plan
----------------------------------------------------------
Plan hash value: 1756381138

--------------------------------------------------------------------------------
| Id  | Operation          | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |           |     1 |     3 |     3   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE    |           |     1 |     3 |            |          |
|   2 |   TABLE ACCESS FULL| EMPLOYEES |   107 |   321 |     3   (0)| 00:00:01 |
--------------------------------------------------------------------------------
```

```
HR> SELECT
  2     (SELECT MIN(DEPARTMENT_ID) FROM EMPLOYEES) MIN_DEPARTAMENT_ID,
  3     (SELECT MAX(DEPARTMENT_ID) FROM EMPLOYEES) MAX_DEPARTMENT_ID
  4  FROM DUAL;

Execution Plan
----------------------------------------------------------
Plan hash value: 2189307159

-----------------------------------------------------------------------------------------------
| Id  | Operation                    | Name              | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                   |     1 |       |     4   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE              |                   |     1 |     3 |            |          |
|   2 |   INDEX FULL SCAN (MIN/MAX)| EMP_DEPARTMENT_IX |     1 |     3 |     1   (0)| 00:00:01 |
|   3 |  SORT AGGREGATE              |                   |     1 |     3 |            |          |
|   4 |   INDEX FULL SCAN (MIN/MAX)| EMP_DEPARTMENT_IX |     1 |     3 |     1   (0)| 00:00:01 |
|   5 |  FAST DUAL                   |                   |     1 |       |     2   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------
```

Index Skip Scan

- O predicado contém uma condição em uma coluna indexada, mas esta coluna não é a primeira do índice, e as primeiras colunas tem um baixo NDV.

```
HR> SELECT FIRST_NAME, LAST_NAME FROM EMPLOYEES WHERE FIRST_NAME LIKE 'A%';

Execution Plan
----------------------------------------------------------
Plan hash value: 234440638

-------------------------------------------------------------------------------------
| Id  | Operation        | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT |             |    10 |   150 |     1   (0)| 00:00:01 |
|*  1 |  INDEX SKIP SCAN | EMP_NAME_IX |    10 |   150 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("FIRST_NAME" LIKE 'A%')
       filter("FIRST_NAME" LIKE 'A%')
```

Index Fast Full Scan

- **Utilizado quando todas as colunas do SELECT estão incluídas no índice;**
- Utiliza MBRC;
- Não há acesso à tabela;
- Não pode ser utilizado para evitar um SORT;
- Pelo menos uma das colunas do índice deve ser NOT NULL.

```
SCOTT> SELECT ID FROM T1;

Execution Plan
----------------------------------------------------------
Plan hash value: 3617692013

--------------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |      | 10000 | 30000 |    46   (0)| 00:00:01 |
|   1 |  TABLE ACCESS FULL| T1   | 10000 | 30000 |    46   (0)| 00:00:01 |
--------------------------------------------------------------------------------

SCOTT> SELECT COUNT(ID) FROM T1;

Execution Plan
----------------------------------------------------------
Plan hash value: 4099882575

--------------------------------------------------------------------------------------
| Id  | Operation             | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |         |     1 |     3 |     7   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE       |         |     1 |     3 |            |          |
|   2 |   INDEX FAST FULL SCAN| T1_ODX1 | 10000 | 30000 |     7   (0)| 00:00:01 |
--------------------------------------------------------------------------------------
```

Modelagem

PK: Natural ou Artificial?

Natural (RG, CPF, Nota Fiscal, Matrícula, Apólice...)

- Naturalidade no entendimento das colunas;
- Redução da largura da linha;
- Menor quantidade de JOINs para exibir o resultado final;
- Validação natural de regras de negócio.

Artificial (SEQUENCE, IDENTITY, MAX + 1 com FOR UPDATE)

- Alterações com menor impacto;
- Redução da largura das chaves;
- Redução da possibilidade de concorrência em alterações de campos;
- Composição desnecessária;
- Simplicidade de JOINs.
- Possibilidade de uso para Partitionamento.

Lab 6.1: Data Types

Verifique os planos de execução dos SQL abaixo.

```
SQL> CONN SCOTT/TIGER

SQL> CREATE TABLE T61 (ID VARCHAR(255), NAME VARCHAR(255));
SQL> CREATE INDEX T61_IDX ON T61(ID);
SQL> INSERT INTO T61 SELECT * FROM T1;
SQL> COMMIT;

SQL> SET AUTOTRACE TRACEONLY EXPLAIN
SQL> SELECT COUNT(*) FROM T1 WHERE ID=1;
SQL> SELECT COUNT(*) FROM T61 WHERE ID=1;
SQL> SELECT COUNT(*) FROM T61 WHERE ID='1';

SQL> SET AUTOTRACE OFF
SQL> INSERT INTO T61 VALUES ('X', 'X');
SQL> SELECT COUNT(*) FROM T61 WHERE ID=1;
SQL> ROLLBACK;
SQL> SELECT COUNT(*) FROM T61 WHERE ID=1;
```

Lab 6.2: Data Types

Verifique o plano de execução do SQL abaixo.

```
SQL> CONN SCOTT/TIGER
SQL> CREATE TABLE T62 AS
WITH GENERATOR AS (
     SELECT     --+ materialize
          ROWNUM ID FROM ALL_OBJECTS WHERE ROWNUM <= 2000)
SELECT
     DECODE(
          MOD(ROWNUM - 1,1000),
          0, TO_DATE('31-Dec-4000'),
          TO_DATE('01-Jan-2008') + trunc((rownum - 1)/100)
     ) DATA_PEDIDO
FROM GENERATOR V1, GENERATOR V2
WHERE ROWNUM <= 1827 * 100;

SQL> SELECT COUNT(*) FROM T62;
SQL> SELECT COUNT(*) FROM T62 WHERE DATA_PEDIDO = TO_DATE('31-Dec-4000');

SQL> SET AUTOTRACE TRACEONLY EXPLAIN
SQL> SELECT COUNT(*) FROM T62 WHERE DATA_PEDIDO BETWEEN TO_DATE('01-Jan-
2010','dd-mon-yyyy') AND TO_DATE('31-Dec-2010','dd-mon-yyyy');
```

Lab 6.3: Data Types

Verifique o plano de execução do SQL abaixo.

```
SQL> CONN SCOTT/TIGER
SQL> CREATE TABLE T63 AS
SELECT
     DATA_DATE,
     TO_NUMBER(TO_CHAR(DATA_DATE,'yyyymmdd'))    DATA_NUMBER,
     TO_CHAR(DATA_DATE,'yyyymmdd')               DATA_CHAR
FROM (SELECT TO_DATE('31-Dec-2007') + ROWNUM DATA_DATE FROM ALL_OBJECTS
WHERE ROWNUM <= 1827);

SQL> SET AUTOTRACE TRACEONLY EXPLAIN
SQL> SELECT COUNT(*) FROM T63 WHERE DATA_DATE
       BETWEEN TO_DATE('01-Jan-2010','dd-mon-yyyy')
       AND TO_DATE('31-Dec-2010','dd-mon-yyyy');
SQL> SELECT COUNT(*) FROM T63 WHERE DATA_NUMBER
       BETWEEN 20100101 AND 20101231;
SQL> SELECT COUNT(*) FROM T63 WHERE DATA_CHAR
       BETWEEN '20100101' AND '20101231';
```

Query Transformation

Query Rewrite

De acordo com a Documentação:

- OR Expansion
- View Merging
- Predicate Pushing
- Subquery Unnesting
- In-Memory Aggregation
- Table Expansion
- Join Factorization
- Query Rewrite with Materialized Views
- Star Transformation

Query Rewrite

- Controle: Hints QUERY_TRANSFORMATION / NO_QUERY_TRANSFORMATION

```
SQL> CONN HR/HR
SQL> ALTER SESSION SET EVENTS '10053 TRACE NAME CONTEXT FOREVER, LEVEL 1';
SQL> SELECT * FROM EMPLOYEES WHERE DEPARTMENT_ID IN (SELECT DEPARTMENT_ID
FROM DEPARTMENTS);
```

```
***********************************
Cost-based predicate pushdown (JPPD)
***********************************
kkqctdrvTD-start on query block SEL$0404685D (#1)
kkqctdrvTD-start: :
    call(in-use=44144, alloc=49328), compile(in-use=149360, alloc=162648), execution(in-use=4368, alloc=8088)

kkqctdrvTD-cleanup: transform(in-use=0, alloc=0) :
    call(in-use=44144, alloc=49328), compile(in-use=150272, alloc=162648), execution(in-use=4368, alloc=8088)

kkqctdrvTD-end:
    call(in-use=44144, alloc=49328), compile(in-use=150872, alloc=162648), execution(in-use=4368, alloc=8088)

JPPD: Applying transformation directives
query block SEL$1 transformed to SEL$0404685D (#1)
FPD: Considering simple filter push in query block SEL$0404685D (#1)
"EMPLOYEES"."DEPARTMENT_ID" IS NOT NULL
try to generate transitive predicate from check constraints for query block SEL$0404685D (#1)
constraint: "EMPLOYEES"."SALARY">0

finally: "EMPLOYEES"."DEPARTMENT_ID" IS NOT NULL
```

```
Final query after transformations:******* UNPARSED QUERY IS *******
SELECT "EMPLOYEES"."EMPLOYEE_ID" "EMPLOYEE_ID","EMPLOYEES"."FIRST_NAME" "FIRST_NAME","EMPLOYEES"."LAST_NAME" "LAST_NAME","EMPLOYEES"."EMAIL" "EMAIL","EMPLOYEES"."PHONE
_NUMBER" "PHONE_NUMBER","EMPLOYEES"."HIRE_DATE" "HIRE_DATE","EMPLOYEES"."JOB_ID" "JOB_ID","EMPLOYEES"."SALARY" "SALARY","EMPLOYEES"."COMMISSION_PCT" "COMMISSION_PCT","
EMPLOYEES"."MANAGER_ID" "MANAGER_ID","EMPLOYEES"."DEPARTMENT_ID" "DEPARTMENT_ID" FROM "HR"."EMPLOYEES" "EMPLOYEES" WHERE "EMPLOYEES"."DEPARTMENT_ID" IS NOT NULL
kkoqbc: optimizing query block SEL$0404685D (#1)
```

View Merging

Controle: MERGE / NO_MERGE / Parâmetro _complex_view_merging (TRUE de 9i em diante).

Geralmente utilizada quando o *Outer Query Block* contém:
- Uma coluna que pode ser utilizada em um índice com outro Query Block;
- Uma coluna que pode ser utilizada para Partition Pruning com outro Query Block;
- Uma condição que limita as linhas retornadas de uma das tabelas em uma Joined View.

Geralmente não será utilizada se:
- Um Query Block contiver uma função de agregação ou analítica;
- Um Query Block contiver Set Operations (UNION, INTERSECT, MINUS);
- Um Query Block contiver ORDER BY;
- Um Query Block contiver ROWNUM.

Fomos treinados para entender que estas duas expressões retornam resultados diferentes:
- *6 + 4 / 2 = 8*
- *(6 + 4) / 2 = 5*

Lab 7.1: View Merging

Execute o SELECT abaixo, e encontre no arquivo trace o View Merging.

```
SQL> CONN OE/OE
SQL> ALTER SESSION SET EVENTS '10053 TRACE NAME CONTEXT FOREVER, LEVEL 1';
SQL> SELECT * FROM ORDERS O, (SELECT SALES_REP_ID FROM ORDERS) O_VIEW
       WHERE O.SALES_REP_ID = O_VIEW.SALES_REP_ID (+)
       AND O.ORDER_TOTAL > 100000;
```

```
CVM:   Merging SPJ view SEL$2 (#0) into SEL$1 (#0)
Registered qb: SEL$F5BB74E1 0x10f69898 (VIEW MERGE SEL$1; SEL$2)
```

```
Final query after transformations:******* UNPARSED QUERY IS *******
SELECT "O"."ORDER_ID" "ORDER_ID","O"."ORDER_DATE" "ORDER_DATE","O"."ORDER_MODE" "ORDER_MODE","O"."CU
STOMER_ID" "CUSTOMER_ID","O"."ORDER_STATUS" "ORDER_STATUS","O"."ORDER_TOTAL" "ORDER_TOTAL","O"."SALE
S_REP_ID" "SALES_REP_ID","O"."PROMOTION_ID" "PROMOTION_ID","ORDERS"."SALES_REP_ID" "SALES_REP_ID" FR
OM "OE"."ORDERS" "O","OE"."ORDERS" "ORDERS" WHERE "O"."SALES_REP_ID"="ORDERS"."SALES_REP_ID"(+) AND
"O"."ORDER_TOTAL">100000
```

Execute o SELECT abaixo, e encontre no arquivo trace o View Merging.

```
SQL> CONN OE/OE
SQL> ALTER SESSION SET EVENTS '10053 TRACE NAME CONTEXT FOREVER, LEVEL 1';
SQL> SELECT * FROM ORDERS O, (SELECT SALES_REP_ID FROM ORDERS ORDER BY
SALES_REP_ID) O_VIEW WHERE O.SALES_REP_ID = O_VIEW.SALES_REP_ID (+) AND
O.ORDER_TOTAL > 100000 AND ROWNUM < 10 ORDER BY ORDER_TOTAL;
```

Subquery Unnesting

- Controle: HINT NO_UNNEST, unnest_subquery (TRUE de 9i em diante).
- Similar a View Merging, ocorre quando a Subquery está localizada na cláusula WHERE;
- A transformação mais comum é em um JOIN;

```
SQL> CONN HR/HR
SQL> ALTER SESSION SET EVENTS '10053 TRACE NAME CONTEXT FOREVER, LEVEL 1';
SQL> SELECT * FROM EMPLOYEES WHERE DEPARTMENT_ID IN (SELECT DEPARTMENT_ID
FROM DEPARTMENTS);
```

```
****************************
Cost-Based Subquery Unnesting
****************************
SU: Unnesting query blocks in query block SEL$1 (#1) that are valid to unnest.
Subquery removal for query block SEL$2 (#2)
RSW: Not valid for subquery removal SEL$2 (#2)
Subquery unchanged.
Subquery Unnesting on query block SEL$1 (#1)SU: Performing unnesting that does not require costing.
SU: Considering subquery unnest on query block SEL$1 (#1).
SU:   Checking validity of unnesting subquery SEL$2 (#2)
SU:   Passed validity checks.
SU:   Transforming ANY subquery to a join.
Registered qb: SEL$5DA710D3 0x12f55d38 (SUBQUERY UNNEST SEL$1; SEL$2)
```

Subquery Unnesting + JE

```
************************
Join Elimination (JE)
************************
SQL:******* UNPARSED QUERY IS *******
SELECT "EMPLOYEES"."EMPLOYEE_ID" "EMPLOYEE_ID","EMPLOYEES"."FIRST_NAME" "FIRST_NAME","EMPLOYEES"."LA
ST_NAME" "LAST_NAME","EMPLOYEES"."EMAIL" "EMAIL","EMPLOYEES"."PHONE_NUMBER" "PHONE_NUMBER","EMPLOYEE
S"."HIRE_DATE" "HIRE_DATE","EMPLOYEES"."JOB_ID" "JOB_ID","EMPLOYEES"."SALARY" "SALARY","EMPLOYEES"."
COMMISSION_PCT" "COMMISSION_PCT","EMPLOYEES"."MANAGER_ID" "MANAGER_ID","EMPLOYEES"."DEPARTMENT_ID" "
DEPARTMENT_ID" FROM "HR"."DEPARTMENTS" "DEPARTMENTS","HR"."EMPLOYEES" "EMPLOYEES" WHERE "EMPLOYEES".
"DEPARTMENT_ID"="DEPARTMENTS"."DEPARTMENT_ID"
JE:   cfro: DEPARTMENTS objn:92593 col#:1 dfro:EMPLOYEES dcol#:11
JE:   cfro: DEPARTMENTS objn:92593 col#:1 dfro:EMPLOYEES dcol#:11
JE:   cfro: EMPLOYEES objn:92595 col#:11 dfro:DEPARTMENTS dcol#:1
JE:   cfro: EMPLOYEES objn:92595 col#:11 dfro:DEPARTMENTS dcol#:1
Query block (0x7fe18a654f78) before join elimination:
SQL:******* UNPARSED QUERY IS *******
SELECT "EMPLOYEES"."EMPLOYEE_ID" "EMPLOYEE_ID","EMPLOYEES"."FIRST_NAME" "FIRST_NAME","EMPLOYEES"."LA
ST_NAME" "LAST_NAME","EMPLOYEES"."EMAIL" "EMAIL","EMPLOYEES"."PHONE_NUMBER" "PHONE_NUMBER","EMPLOYEE
S"."HIRE_DATE" "HIRE_DATE","EMPLOYEES"."JOB_ID" "JOB_ID","EMPLOYEES"."SALARY" "SALARY","EMPLOYEES"."
COMMISSION_PCT" "COMMISSION_PCT","EMPLOYEES"."MANAGER_ID" "MANAGER_ID","EMPLOYEES"."DEPARTMENT_ID" "
DEPARTMENT_ID" FROM "HR"."DEPARTMENTS" "DEPARTMENTS","HR"."EMPLOYEES" "EMPLOYEES" WHERE "EMPLOYEES".
"DEPARTMENT_ID"="DEPARTMENTS"."DEPARTMENT_ID"
JE:   eliminate table: DEPARTMENTS (DEPARTMENTS)
Registered qb: SEL$0404685D 0x8a654f78 (JOIN REMOVED FROM QUERY BLOCK SEL$5DA710D3; SEL$5DA710D3; "D
EPARTMENTS"@"SEL$2")
```

```
Final query after transformations:******* UNPARSED QUERY IS *******
SELECT "EMPLOYEES"."EMPLOYEE_ID" "EMPLOYEE_ID","EMPLOYEES"."FIRST_NAME" "FIRST_NAME","EMPLOYEES"."LAST_NAME" "LAST_NAME","EMPLOYEES"."EMAIL"
 "EMAIL","EMPLOYEES"."PHONE_NUMBER" "PHONE_NUMBER","EMPLOYEES"."HIRE_DATE" "HIRE_DATE","EMPLOYEES"."JOB_ID" "JOB_ID","EMPLOYEES"."SALARY" "S
ALARY","EMPLOYEES"."COMMISSION_PCT" "COMMISSION_PCT","EMPLOYEES"."MANAGER_ID" "MANAGER_ID","EMPLOYEES"."DEPARTMENT_ID" "DEPARTMENT_ID" FROM
"HR"."EMPLOYEES" "EMPLOYEES" WHERE "EMPLOYEES"."DEPARTMENT_ID" IS NOT NULL
kkoqbc: optimizing query block SEL$0404685D (#1)
```

O plano de execução utiliza quais tabelas?

Lab 7.2: Subquery Unnesting

Execute os SELECTs abaixo, e compare os Planos de Execução.

```
SQL> CONN HR/HR
SQL> SET AUTOTRACE TRACEONLY EXPLAIN

SQL> SELECT OUTER.EMPLOYEE_ID, OUTER.LAST_NAME, OUTER.SALARY,
OUTER.DEPARTMENT_ID FROM EMPLOYEES OUTER
WHERE OUTER.SALARY > (SELECT AVG(INNER.SALARY) FROM EMPLOYEES INNER WHERE
INNER.DEPARTMENT_ID = OUTER.DEPARTMENT_ID);

SQL> SELECT OUTER.EMPLOYEE_ID, OUTER.LAST_NAME, OUTER.SALARY,
OUTER.DEPARTMENT_ID FROM EMPLOYEES OUTER
WHERE OUTER.SALARY > (SELECT /*+ NO_UNNEST */ AVG(INNER.SALARY) FROM
EMPLOYEES INNER WHERE INNER.DEPARTMENT_ID = OUTER.DEPARTMENT_ID);

SQL> SELECT * FROM EMPLOYEES WHERE DEPARTMENT_ID IN (SELECT DEPARTMENT_ID
FROM DEPARTMENTS);
SQL> SELECT * FROM EMPLOYEES WHERE DEPARTMENT_ID IN (SELECT /*+ NO_UNNEST
*/ DEPARTMENT_ID FROM DEPARTMENTS);
```

Lab 7.3: Subquery Unnesting

Execute o SELECT abaixo, e encontre no arquivo trace o Subquery Unnesting.

```
SQL> CONN HR/HR
SQL> ALTER SESSION SET EVENTS '10053 TRACE NAME CONTEXT FOREVER, LEVEL 1';
SQL> SELECT
           OUTER.EMPLOYEE_ID,
           OUTER.LAST_NAME,
           OUTER.SALARY,
           OUTER.DEPARTMENT_ID
           FROM EMPLOYEES OUTER
       WHERE OUTER.SALARY >
           (SELECT AVG(INNER.SALARY)
              FROM EMPLOYEES INNER
              WHERE INNER.DEPARTMENT_ID = OUTER.DEPARTMENT_ID);
```

Lab 7.4: Subquery Unnesting

```
Registered qb: SEL$1 0x33b33e90 (COPY SEL$1)
---------------------
QUERY BLOCK SIGNATURE
---------------------
  signature(): NULL
Registered qb: SEL$2 0x33b33600 (COPY SEL$2)
---------------------
QUERY BLOCK SIGNATURE
---------------------
  signature(): NULL
*****************************
Cost-Based Subquery Unnesting
*****************************
SU: Unnesting query blocks in query block SEL$1 (#1) that are valid to unnest.
Subquery Unnesting on query block SEL$1 (#1)SU: Performing unnesting that does not require costing.
SU: Considering subquery unnest on query block SEL$1 (#1).
SU:   Checking validity of unnesting subquery SEL$2 (#2)
SU:   Passed validity checks, but requires costing.
SU: Using search type: exhaustive
SU: Starting iteration 1, state space = (2) : (1)
SU:   Unnesting subquery query block SEL$2 (#2)Subquery removal for query block SEL$2 (#2)
RSW: Not valid for subquery removal SEL$2 (#2)
Subquery unchanged.
Registered qb: SEL$683B0107 0x33b2d1f0 (SUBQ INTO VIEW FOR COMPLEX UNNEST SEL$2)
---------------------
QUERY BLOCK SIGNATURE
---------------------
  signature (): qb_name=SEL$683B0107 nbfros=1 flg=0
    fro(0): flg=0 objn=92593 hint_alias="INNER"@"SEL$2"

Registered qb: SEL$7511BFD2 0x33b2d9d0 (VIEW ADDED SEL$1)
```

```
Final query after transformations:******* UNPARSED QUERY IS *******
SELECT "OUTER"."EMPLOYEE_ID" "EMPLOYEE_ID","OUTER"."LAST_NAME" "LAST_NAME","OUTER"."SALARY" "SALARY","OUTER"."DEPARTMENT_ID" "DEPARTMENT_ID"
 FROM  (SELECT AVG("INNER"."SALARY") "AVG(INNER.SALARY)","INNER"."DEPARTMENT_ID" "ITEM_1" FROM "HR"."EMPLOYEES" "INNER" GROUP BY "INNER"."DE
PARTMENT_ID") "VW_SQ_1","HR"."EMPLOYEES" "OUTER" WHERE "OUTER"."SALARY">"VW_SQ_1"."AVG(INNER.SALARY)" AND "VW_SQ_1"."ITEM_1"="OUTER"."DEPART
MENT_ID"
kkoqbc: optimizing query block SEL$683B0107 (#2)
```

Predicate Pushing

- Controle: Hints PUSH_PRED / NO_PUSH_PRED;
- Geralmente utilizado para aplicar os predicados de um Query Block em outro Query Block, quando não pode ocorrer uma transformação de Merging, para permitir a utilização de um índice e filtrar mais cedo no plano de execução.

Lab 7.5: Predicate Pushing

Execute o SELECT abaixo, e encontre no arquivo trace o Predicate Pushing.

```
SQL> CONN HR/HR
SQL> ALTER SESSION SET EVENTS '10053 TRACE NAME CONTEXT FOREVER, LEVEL 1';
SQL> SELECT E1.LAST_NAME, E1.SALARY, V.AVG_SALARY
  FROM EMPLOYEES E1,
       (SELECT DEPARTMENT_ID, AVG(SALARY) AVG_SALARY
       FROM EMPLOYEES E2
       GROUP BY DEPARTMENT_ID) V
  WHERE E1.DEPARTMENT_ID = V.DEPARTMENT_ID
    AND E1.SALARY > V.AVG_SALARY
    AND E1.DEPARTMENT_ID = 60;
```

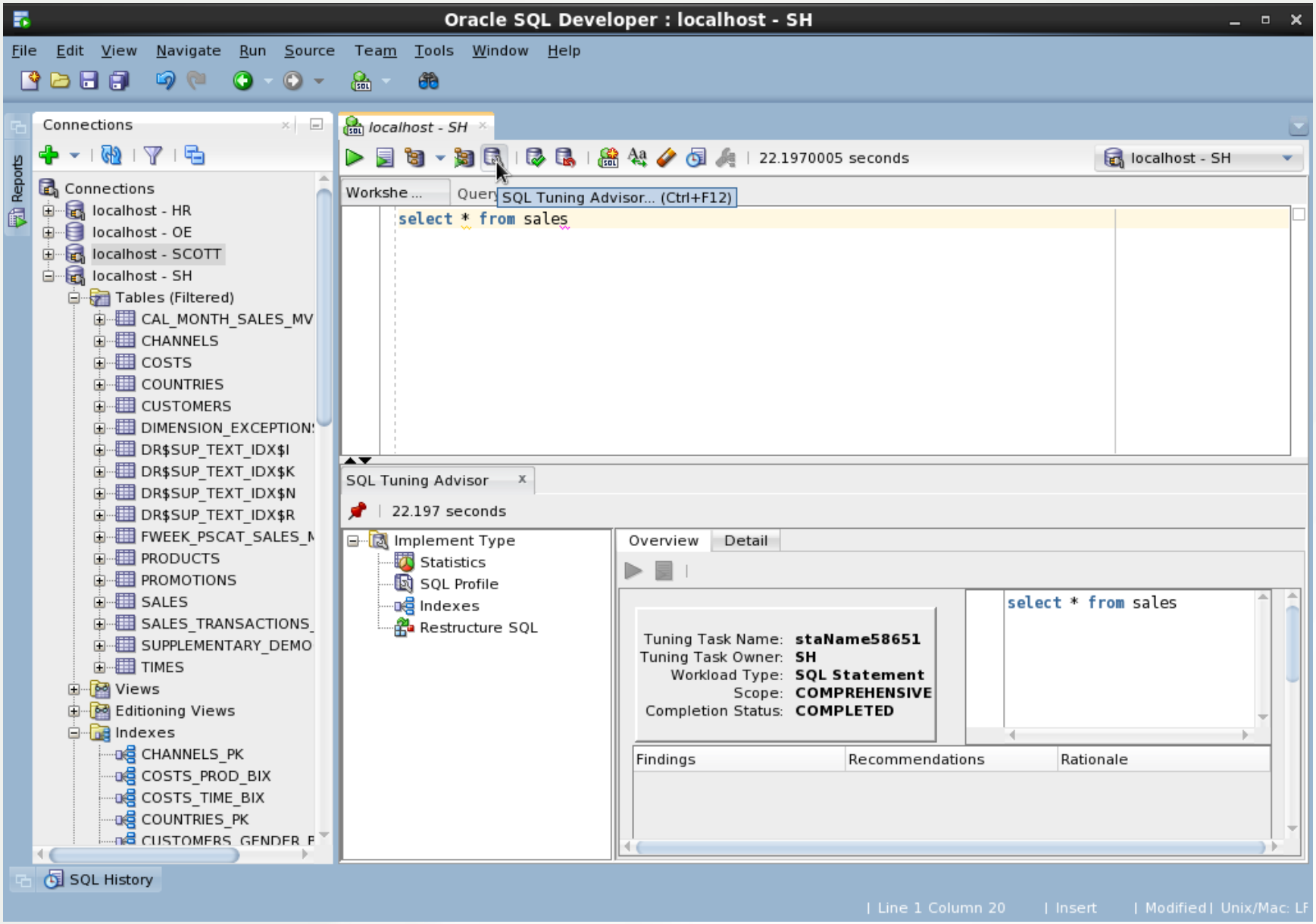
```
FPD: Considering simple filter push (pre rewrite) in query block SEL$1 (#0)
FPD:  Current where clause predicates "E1"."DEPARTMENT_ID"="V"."DEPARTMENT_ID" AND "E1"."SALARY">"V"."AVG_SALARY" AND "E1"."DEPARTMENT_ID"=60

try to generate transitive predicate from check constraints for query block SEL$1 (#0)
constraint: "E1"."SALARY">0

finally: "E1"."DEPARTMENT_ID"="V"."DEPARTMENT_ID" AND "E1"."SALARY">"V"."AVG_SALARY" AND "E1"."DEPARTMENT_ID"=60 AND "V"."DEPARTMENT_ID"=60

FPD:   transitive predicates are generated in query block SEL$1 (#0)
"E1"."DEPARTMENT_ID"="V"."DEPARTMENT_ID" AND "E1"."SALARY">"V"."AVG_SALARY" AND "E1"."DEPARTMENT_ID"=60 AND "V"."DEPARTMENT_ID"=60
FPD:   Following are pushed to where clause of query block SEL$2 (#0)
"E2"."DEPARTMENT_ID"=60
FPD: Considering simple filter push (pre rewrite) in query block SEL$2 (#0)
FPD:  Current where clause predicates "E2"."DEPARTMENT_ID"=60
```

```
Final query after transformations:******* UNPARSED QUERY IS *******
SELECT "E1"."LAST_NAME" "LAST_NAME","E1"."SALARY" "SALARY","V"."AVG_SALARY" "AVG_SALARY" FROM "HR"."EMPLOYEES" "E1", (SELECT 60 "DEPARTMENT_
ID",AVG("E2"."SALARY") "AVG_SALARY" FROM "HR"."EMPLOYEES" "E2" WHERE "E2"."DEPARTMENT_ID"=60 AND 60=60 GROUP BY 60) "V" WHERE "E1"."DEPARTME
NT_ID"="V"."DEPARTMENT_ID" AND "E1"."SALARY">"V"."AVG_SALARY" AND "E1"."DEPARTMENT_ID"=60
```

E com DEPARTMENT_ID IN (10,40,60)?

DBMS_SQLTUNE

DBMS_SQLTUNE



Lab 8.1: DBMS_SQLTUNE

```
SQL> CONN SCOTT/TIGER
SQL> CREATE TABLE T8 AS SELECT * FROM ALL_OBJECTS;
SQL> SELECT COUNT(DISTINCT(OWNER)) FROM T8;
SQL> SELECT SQL_ID FROM V$SQL WHERE SQL_TEXT LIKE '%T8%';
```

Execute o SQL_TUNE nos SQLs analizados.

```
SQL> CONN / AS SYSDBA
SQL> DECLARE RET_VAL VARCHAR2(4000);
BEGIN
    RET_VAL := DBMS_SQLTUNE.CREATE_TUNING_TASK(SQL_ID => '3a7wvqqf1tsun',
    SCOPE => DBMS_SQLTUNE.SCOPE_COMPREHENSIVE,
    TIME_LIMIT => 60,
    TASK_NAME => 'Portilho Tuning Task',
    DESCRIPTION => 'Portilho Tuning Task');
END;
/

SQL> EXEC DBMS_SQLTUNE.EXECUTE_TUNING_TASK('Portilho Tuning Task');

SQL> SET LONG 9000
SQL> SELECT DBMS_SQLTUNE.REPORT_TUNING_TASK('Portilho Tuning Task') FROM DUAL;
SQL> SELECT DBMS_SQLTUNE.SCRIPT_TUNING_TASK('Portilho Tuning Task') FROM DUAL;
```

Remova o SQL_TUNE executado, após executar a correção.

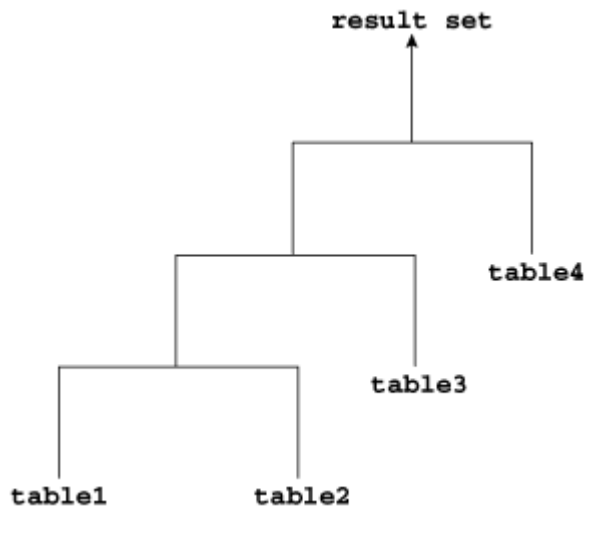
```
SQL> EXEC DBMS_SQLTUNE.DROP_TUNING_TASK('Portilho Tuning Task');
```

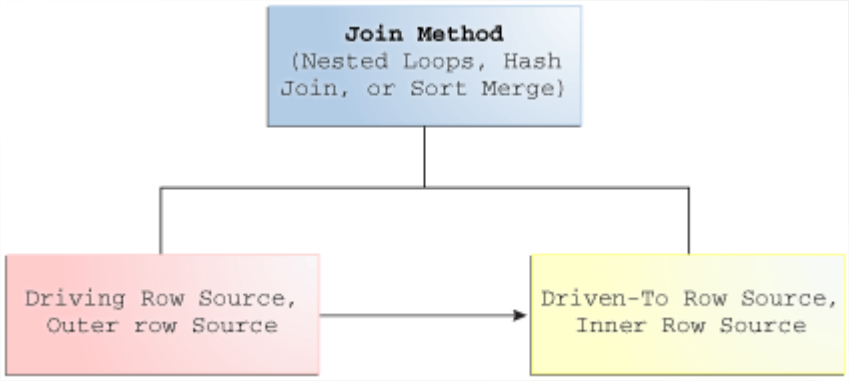
Join Methods & Options

Join Methods

- Nested Loops
- Hash Joins
- Sort-merge Joins

Driving Table (maior / OUTER / BUILD / PK / UNIQUE) / Probe Table (menor / INNER):





A silent movie on joins
https://www.youtube.com/watch?v=-ldqQY5XoKg

Join Options

- Inner Join
- Outer Join
- Cross Join / Cartesian Joins
- Semi-Join
- Anti-Join

Nested Loops

- É um LOOP dentro de um LOOP.
- É mais eficiente com pequenos Result Sets;
- Geralmente ocorre quando há índices nas colunas utilizadas pelo Join;
- Utiliza pouca memória, pois o Result Set é construído uma linha por vez;
- HINT: /*+ leading(ORDER_ITENS ORDERS) use_nl(ORDERS) index(ORDERS(ORDER_ID)) */
- HINT: /*+ leading(ORDERS ORDER_ITENS) use_nl(ORDER_ITENS) index(ORDER_LINES(ORDER_ID)) */

Nested Loops

```
SCOTT> SELECT EMPNO, ENAME, DNAME, LOC FROM EMP, DEPT WHERE EMP.DEPTNO = DEPT.DEPTNO AND DEPT.DEPTNO = 10;

Execution Plan
----------------------------------------------------------
Plan hash value: 568005898

-------------------------------------------------------------------------------------------
| Id  | Operation                    | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |         |     5 |   165 |     4   (0)| 00:00:01 |
|   1 |  NESTED LOOPS                |         |     5 |   165 |     4   (0)| 00:00:01 |
|   2 |   TABLE ACCESS BY INDEX ROWID| DEPT    |     1 |    20 |     1   (0)| 00:00:01 |
|*  3 |    INDEX UNIQUE SCAN         | PK_DEPT |     1 |       |     0   (0)| 00:00:01 |
|*  4 |   TABLE ACCESS FULL          | EMP     |     5 |    65 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

```
SQL> SELECT EMPNO, ENAME, DNAME, LOC FROM EMP, DEPT WHERE EMP.DEPTNO =
DEPT.DEPTNO AND DEPT.DEPTNO IN (10);

SQL> SELECT EMPNO, ENAME, DNAME, LOC FROM EMP, DEPT WHERE EMP.DEPTNO =
DEPT.DEPTNO AND DEPT.DEPTNO IN (10,20);
SQL> SELECT EMPNO, ENAME, DNAME, LOC FROM EMP, DEPT WHERE EMP.DEPTNO =
DEPT.DEPTNO;
```

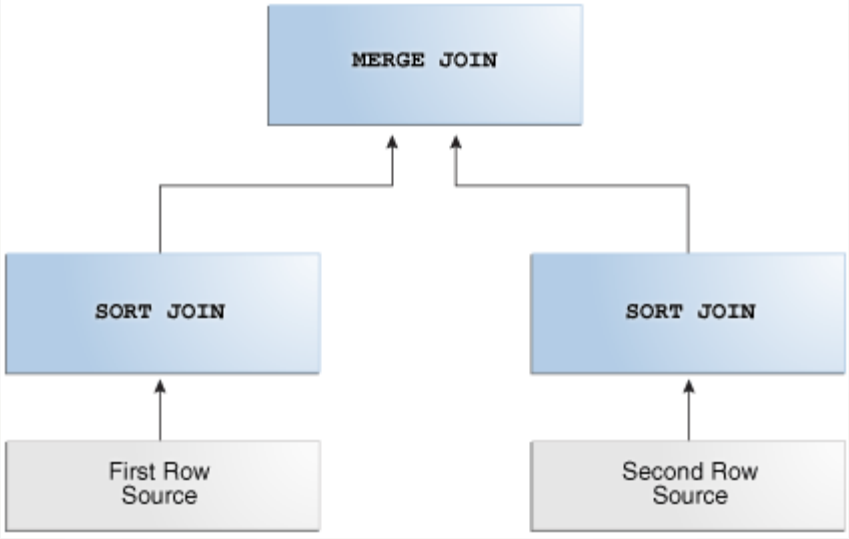
Por que Rows = 5?

```
SELECT COUNT(*) FROM EMP / SELECT COUNT(DISTINCT(DEPTNO)) FROM EMP;
```

Nested Loops

```
SQL> CONN SCOTT/TIGER

SQL> SET AUTOTRACE ON EXPLAIN
SQL> SELECT EMPNO, ENAME, DNAME, LOC FROM EMP, DEPT WHERE EMP.DEPTNO =
DEPT.DEPTNO AND DEPT.DEPTNO IN (10);
SQL> SELECT /*+ leading(DEPT EMP) use_nl(EMP) */ EMPNO, ENAME, DNAME, LOC
FROM EMP, DEPT WHERE EMP.DEPTNO = DEPT.DEPTNO AND DEPT.DEPTNO IN (10);
SQL> SELECT /*+ leading(EMP DEPT) use_nl(DEPT) */ EMPNO, ENAME, DNAME,
LOC FROM EMP, DEPT WHERE EMP.DEPTNO = DEPT.DEPTNO AND DEPT.DEPTNO IN
(10);

SQL> SET AUTOTRACE OFF
SQL> SELECT /*+ GATHER_PLAN_STATISTICS */ EMPNO, ENAME, DNAME, LOC FROM
EMP, DEPT WHERE EMP.DEPTNO = DEPT.DEPTNO AND DEPT.DEPTNO IN (10);
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(NULL, NULL, 'ALLSTATS
LAST'));
```

Sort-Merge Joins



Sort-Merge Joins

- Lê as duas tabelas de forma independente, ordena, e junta os Result Sets, descartando linhas que não combinam;
- Geralmente é utilizado para Result Sets maiores, e quando não há índices;
- Geralmente é utilizado quando é uma operação de desigualdade;
- O maior custo é a ordenação;
- Poderá ser utilizada apenas PGA, ou pode ser necessário TEMP;
- HINTs: USE_MERGE / NO_USE_MERGE

```
SCOTT> SELECT EMPNO, ENAME, DNAME, LOC FROM EMP, DEPT WHERE EMP.DEPTNO = DEPT.DEPTNO;

Execution Plan
----------------------------------------------------------
Plan hash value: 844388907

-------------------------------------------------------------------------------------------
| Id  | Operation                    | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |         |    14 |   462 |     6  (17)| 00:00:01 |
|   1 |  MERGE JOIN                  |         |    14 |   462 |     6  (17)| 00:00:01 |
|   2 |   TABLE ACCESS BY INDEX ROWID| DEPT    |     4 |    80 |     2   (0)| 00:00:01 |
|   3 |    INDEX FULL SCAN           | PK_DEPT |     4 |       |     1   (0)| 00:00:01 |
|*  4 |   SORT JOIN                  |         |    14 |   182 |     4  (25)| 00:00:01 |
|   5 |    TABLE ACCESS FULL         | EMP     |    14 |   182 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

```
SQL> SELECT EMPNO, ENAME, DNAME, LOC FROM EMP, DEPT WHERE EMP.DEPTNO =
DEPT.DEPTNO AND DEPT.DEPTNO NOT IN (10);
```

Hash Joins

- Só ocorre em equi-joins;
- Geralmente é utilizado para grandes Result Sets;
- Geralmente é utilizado se o menor Result Set cabe em memória;
- A tabela com o menor Result Set é lida e armazenada em memória como um HASH;
- Em seguida a outra tabela (maior Result Set) é lida, é aplicado o HASH, e então comparada com a menor;
- Poderá ser utilizada apenas PGA, ou pode ser necessário TEMP;

```
SCOTT> SELECT T1.ID, T2.NAME FROM T1, T2 WHERE T1.ID = T2.ID;

Execution Plan
----------------------------------------------------------
Plan hash value: 3167886034

-------------------------------------------------------------------------------------
| Id  | Operation             | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |         |  1000K|   102M|    56   (6)| 00:00:01 |
|*  1 |  HASH JOIN            |         |  1000K|   102M|    56   (6)| 00:00:01 |
|   2 |   INDEX FAST FULL SCAN| T1_ODX1 | 10000 | 30000 |     7   (0)| 00:00:01 |
|   3 |   TABLE ACCESS FULL   | T2      | 10000 |  1015K|    46   (0)| 00:00:01 |
-------------------------------------------------------------------------------------
```

Hash Joins

```
SCOTT> SELECT DISTINCT DEPTNO, ora_hash(DEPTNO,1000) HV FROM EMP ORDER BY DEPTNO;

    DEPTNO         HV
---------- ----------
        10        547
        20        486
        30        613

SCOTT> SELECT DISTINCT DEPTNO, ora_hash(DEPTNO,1000,50) HV FROM EMP ORDER BY DEPTNO;

    DEPTNO         HV
---------- ----------
        10        839
        20        850
        30        290
```

Hash Joins

```
SQL> CONN SCOTT/TIGER
SQL> SET AUTOTRACE TRACEONLY EXPLAIN

SQL> SELECT T1.ID, T2.NAME FROM T1, T2 WHERE T1.ID = T2.ID;
SQL> SELECT /*+ USE_HASH (T1 T2) */ T1.ID, T2.NAME FROM T1, T2 WHERE T1.ID = T2.ID;
SQL> SELECT /*+ USE_MERGE (T1 T2) */ T1.ID, T2.NAME FROM T1, T2 WHERE T1.ID = T2.ID;
SQL> SELECT /*+ USE_NL (T1 T2) */ T1.ID, T2.NAME FROM T1, T2 WHERE T1.ID = T2.ID;
```

Cartesian Joins / Cross Joins

- Geralmente trata-se da ausência de condição do Join (DISTINCT!);
- O tamanho do resultado é o produto dos dois Result Sets;
- Poderá ser utilizada apenas PGA, ou pode ser necessário TEMP.

```
SCOTT> SELECT T1.ID, T2.NAME FROM T1, T2;

Execution Plan
----------------------------------------------------------
Plan hash value: 1323614827

--------------------------------------------------------------------------------
| Id  | Operation            | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |      |   100M|  9918M|   444K  (1)| 00:00:18 |
|   1 |  MERGE JOIN CARTESIAN|      |   100M|  9918M|   444K  (1)| 00:00:18 |
|   2 |   TABLE ACCESS FULL  | T2   | 10000 |   986K|    46   (0)| 00:00:01 |
|   3 |   BUFFER SORT        |      | 10000 | 30000 |   444K  (1)| 00:00:18 |
|   4 |    TABLE ACCESS FULL | T1   | 10000 | 30000 |    44   (0)| 00:00:01 |
--------------------------------------------------------------------------------
```

O BUFFER SORT de um Cartesian Join não é um SORT.

Cartesian Joins / Cross Joins

```
SCOTT> SELECT DISTINCT T1.ID, T2.NAME FROM T1, T2;

Execution Plan
----------------------------------------------------------
Plan hash value: 2798497149

-------------------------------------------------------------------------------------------------------
| Id  | Operation             | Name            | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |                 |   707K|    70M|       | 23641   (1)| 00:00:01 |
|   1 |  HASH UNIQUE          |                 |   707K|    70M|   108M| 23641   (1)| 00:00:01 |
|   2 |   MERGE JOIN CARTESIAN|                 |  1000K|    99M|       |  4497   (1)| 00:00:01 |
|   3 |    VIEW               | VW_DTP_BA8ECEFB |   100 |   300 |       |    47   (3)| 00:00:01 |
|   4 |     HASH UNIQUE       |                 |   100 |   300 |       |    47   (3)| 00:00:01 |
|   5 |      TABLE ACCESS FULL| T1              | 10000 | 30000 |       |    46   (0)| 00:00:01 |
|   6 |    BUFFER SORT        |                 | 10000 |   986K|       |  4497   (1)| 00:00:01 |
|   7 |     TABLE ACCESS FULL | T2              | 10000 |   986K|       |    45   (3)| 00:00:01 |
-------------------------------------------------------------------------------------------------------
```

```
SQL> SELECT T1.ID, T2.NAME FROM T1, T2 WHERE T1.ID = T2.ID;
```

Outer Joins

- Sintaxe Oracle: (+);
- A sintaxe Oracle não suporta FULL OUTER JOIN;
- Sintaxe ANSI: LEFT [OUTER] JOIN / RIGHT [OUTER] JOIN;
- Um OUTER JOIN pode utilizar qualquer método de JOIN;
- 12c: Multi-table LEFT OUTER JOIN.

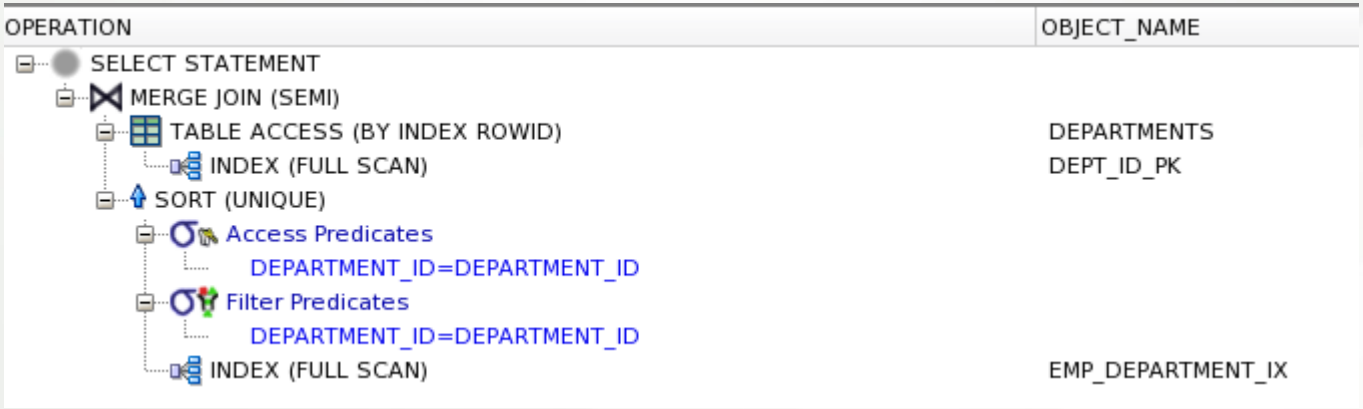
```
HR> SELECT EMPLOYEES.EMPLOYEE_ID, END_DATE FROM EMPLOYEES, JOB_HISTORY WHERE EMPLOYEES.EMPLOYEE_ID = JOB_HISTORY.EMPLOYEE_ID (+);

EMPLOYEE_ID END_DATE
----------- ---------
        100
        101 27-OCT-01
        101 15-MAR-05
        102 24-JUL-06
        103
        104
        105
```

```
HR> SELECT EMPLOYEES.EMPLOYEE_ID, END_DATE FROM EMPLOYEES, JOB_HISTORY WHERE EMPLOYEES.EMPLOYEE_ID = JOB_HISTORY.EMPLOYEE_ID (+);

Execution Plan
----------------------------------------------------------
Plan hash value: 1914381899

-------------------------------------------------------------------------------------------------------------
| Id  | Operation                           | Name              | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |                   |   110 |  1760 |     1   (0)| 00:00:01 |
|   1 |  NESTED LOOPS OUTER                 |                   |   110 |  1760 |     1   (0)| 00:00:01 |
|   2 |   INDEX FULL SCAN                   | EMP_EMP_ID_PK     |   107 |   428 |     1   (0)| 00:00:01 |
|   3 |   TABLE ACCESS BY INDEX ROWID BATCHED| JOB_HISTORY       |     1 |    12 |     0   (0)| 00:00:01 |
|*  4 |    INDEX RANGE SCAN                 | JHIST_EMPLOYEE_IX |     1 |       |     0   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------------------
```

Semi-Joins (IN)

- Controle: HINTs SEMIJOIN, NO_SEMIJOIN, NL_SJ, MERGE_SJ, HASH_SJ
- Uma otimização de INNER JOIN, de custo menor;
- Utilizado em listas IN ou EXISTS;
- É necessário utilizar uma Subquery;
- Se for utilizado EXISTS, é necessário utilizar uma Correlated Subquery;
- As cláusulas IN ou EXISTS não podem estar dentro de um OR.

```
SQL> CONN HR/HR
SQL> SELECT DEPARTMENT_NAME
       FROM DEPARTMENTS DEPT
       WHERE DEPARTMENT_ID IN
           (SELECT DEPARTMENT_ID FROM HR.EMPLOYEES EMP);
```



Semi-Joins (EXISTS)

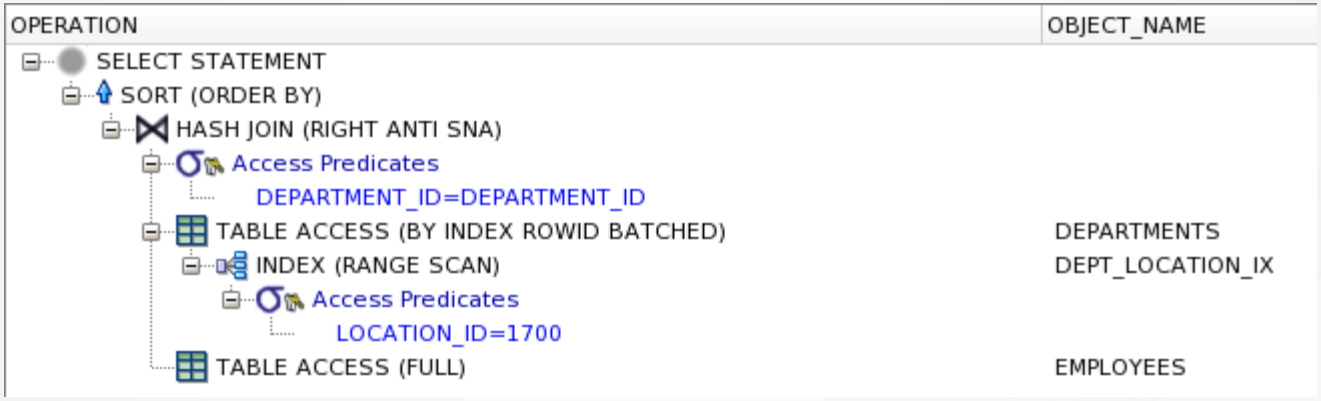
```
SQL> CONN HR/HR
SQL> SELECT DEPARTMENT_NAME
       FROM DEPARTMENTS DEPT
       WHERE EXISTS
           (SELECT NULL
              FROM HR.EMPLOYEES EMP
              WHERE EMP.DEPARTMENT_ID = DEPT.DEPARTMENT_ID);
```

Anti-Joins (NOT IN)

- Uma otimização de INNER JOIN, de custo menor;
- Utilizado em listas NOT IN ou NOT EXISTS;
- É necessário utilizar uma Subquery;
- Se for utilizado NOT EXISTS, é necessário utilizar uma Correlated Subquery;
- As cláusulas NOT IN ou NOT EXISTS não podem estar dentro de um OR;
- A Subquery não pode retornar NULL (<=10g) (SNA).
- Controle: HINTs ANTIJOIN, NL_AJ, MERGE_AJ, HASH_AJ
- Controle: Parâmetros _always_anti, _gs_anti_semi_join_allowed, _optimizer_null_aware_antijoin, _optimizer_outer_to_anti_enabled

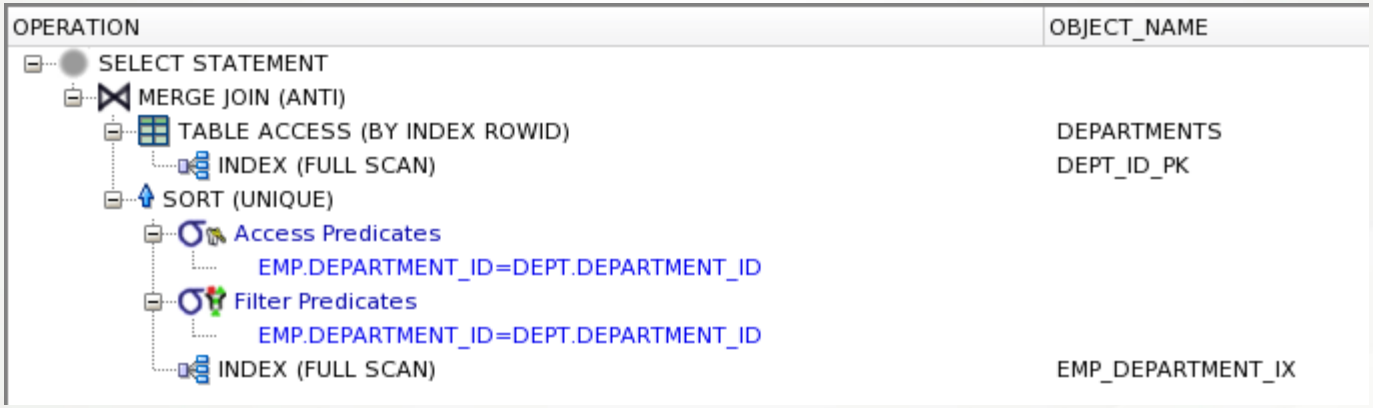
Anti-Joins (NOT IN)

```
SQL> SELECT * FROM EMPLOYEES
       WHERE DEPARTMENT_ID NOT IN
           (SELECT DEPARTMENT_ID
              FROM DEPARTMENTS
              WHERE LOCATION_ID = 1700)
              ORDER BY LAST_NAME;
```



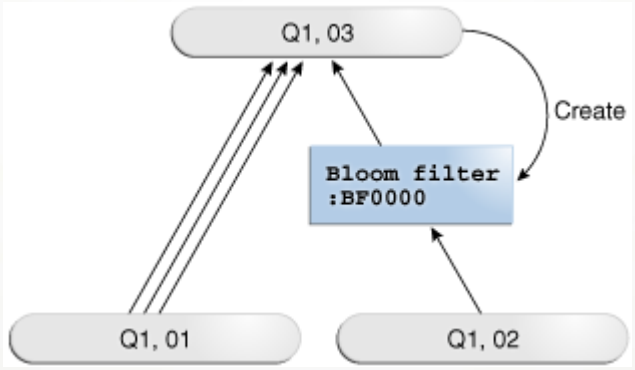
Anti-Joins (NOT EXISTS)

```
SQL> SELECT DEPARTMENT_NAME
       FROM DEPARTMENTS DEPT
       WHERE NOT EXISTS
           (SELECT NULL
              FROM EMPLOYEES EMP
              WHERE EMP.DEPARTMENT_ID = DEPT.DEPARTMENT_ID);
```



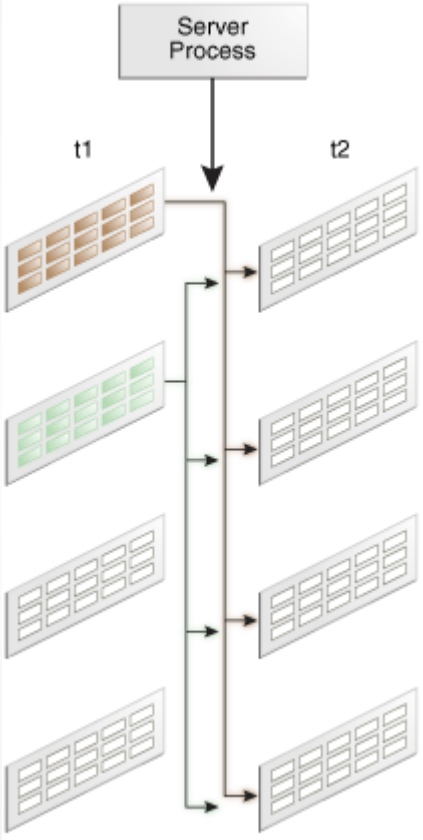
Bloom Filters

- Controle: HINTs PX_JOIN_FILTER, NO_PX_JOIN_FILTER;
- Utiliza pouca memória para o filtro em relação do conjunto de dados;
- É mais utilizado quando a maioria das linhas são descartadas em um JOIN;
- Reduz a quantidade de dados transferidos entre processos paralelos;
- É mais utilizado se mais partições forem eliminadas por Partition Pruning;
- Pode utilizar Result Cache;
- É utilizado em Exatada, especialmente em JOINs em STAR SCHEMAs.



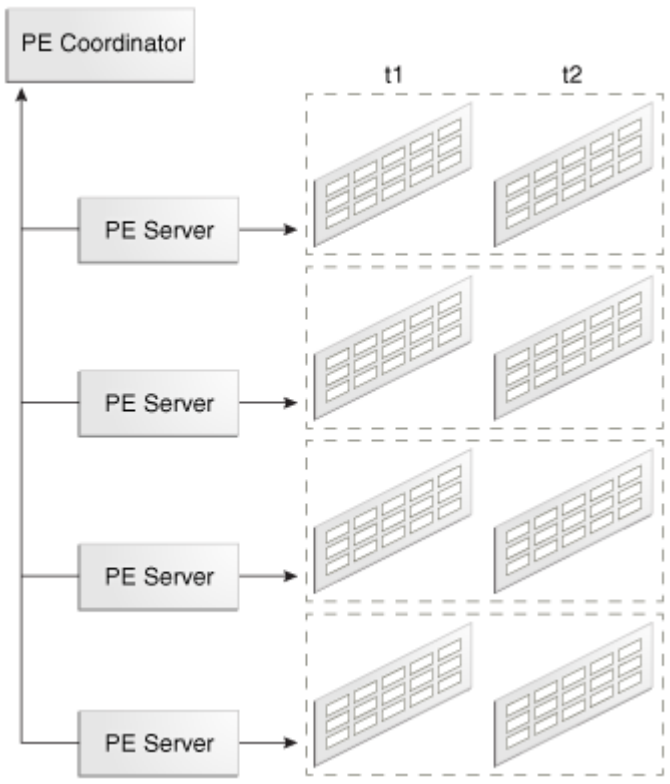
Partition-wise Joins

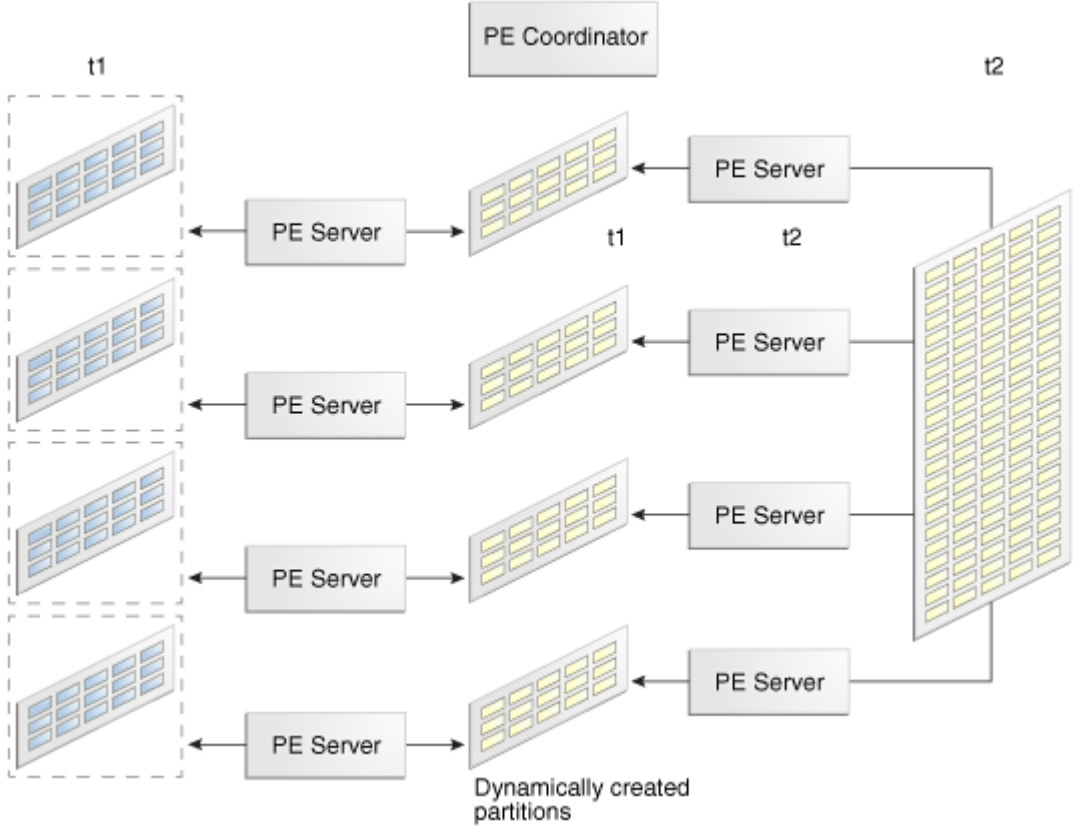
- No partition-wise join



Partition-wise Joins

- Full partition-wise join
- Partial partition-wise joins





Índices

Índices

- B-tree
- Bitmap
- Bitmap Join
- IOT (Index-Organized Table)
- Function-Based
- Bitmap Function-Based
- Reverse Key
- Invisible Indexes (11g / OPTIMIZER_USE_INVISIBLE_INDEXES)
- Virtual Indexes (_use_nosegment_indexes)
- Partitioned Indexes
- Partial Indexes (12c)
- Domain Indexes
- Compressed
- Ascending / Descending
- Table Clusters

Índices

Bitmap Join

```
SQL> CREATE BITMAP INDEX cust_sales_bji
       ON    sales(customers.state)
       FROM  sales, customers
       WHERE sales.cust_id = customers.cust_id;
```

IOT

```
CREATE TABLE locations
    (id            NUMBER(10)    NOT NULL,
     description   VARCHAR2(50)  NOT NULL,
     map           BLOB,
     CONSTRAINT pk_locations PRIMARY KEY (id)
    )
    ORGANIZATION INDEX
    TABLESPACE iot_tablespace
    PCTTHRESHOLD 20
    INCLUDING description
    OVERFLOW TABLESPACE overflow_tablespace;
```

Reverse

```
CREATE INDEX bowie_reverse_idx ON bowie(id) REVERSE;
```

Índices

Invisible

```
SQL> ALTER INDEX IDX_T INVISIBLE;
SQL> ALTER INDEX IDX_T VISIBLE;
```

Virtual

```
SQL> CREATE INDEX IDX_T ON T(OBJECT_NAME) NOSEGMENT;
```

Function Based

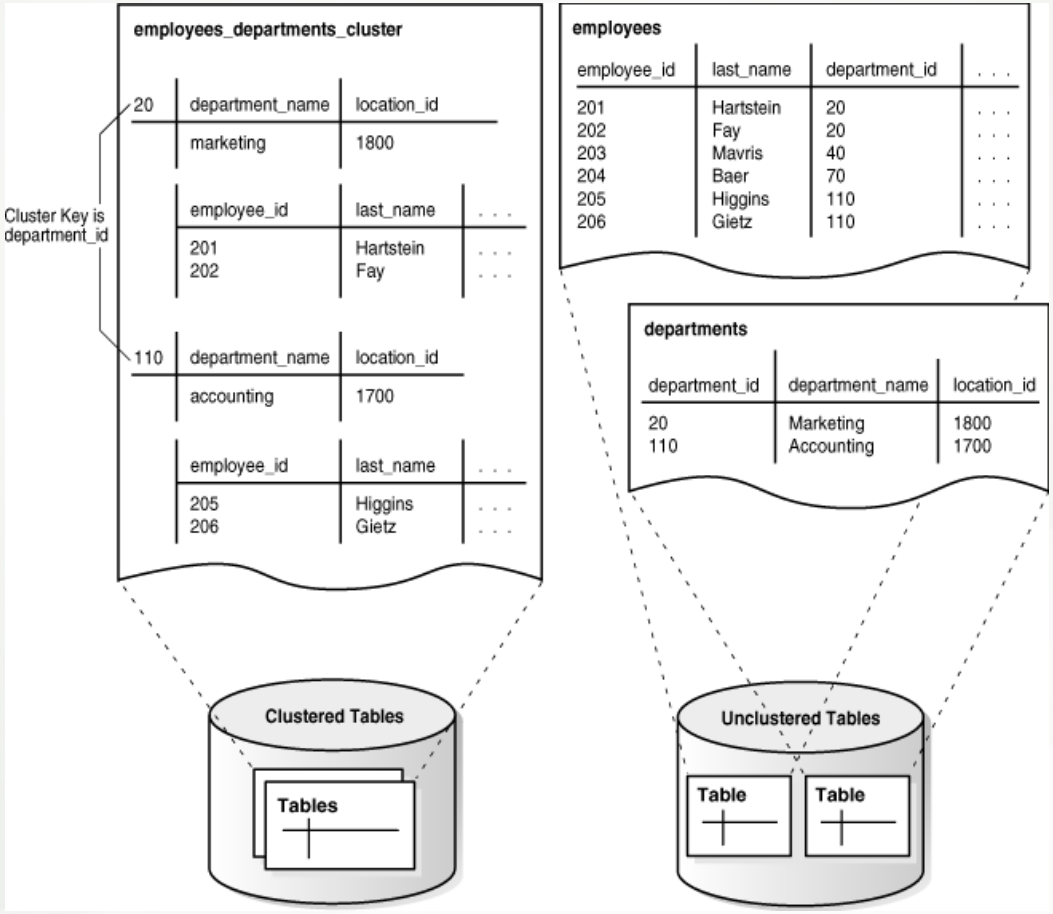
```
SQL> CREATE INDEX IDX_T ON T(UPPER(OBJECT_NAME));
```

Partial Index (12c)

```
SQL> ALTER TABLE T1 MODIFY PARTITION ANO_2014 INDEXING OFF;
SQL> CREATE INDEX T1_IDX ON T1(YEAR) LOCAL;
SQL> CREATE INDEX T1_IDX ON T1(YEAR) LOCAL INDEXING FULL;
SQL> CREATE INDEX T1_IDX ON T1(YEAR) LOCAL INDEXING PARTIAL;
SQL> CREATE INDEX T1_IDX ON T1(YEAR) GLOBAL;
SQL> CREATE INDEX T1_IDX ON T1(YEAR) GLOBAL INDEXING FULL;
SQL> CREATE INDEX T1_IDX ON T1(YEAR) GLOBAL INDEXING PARTIAL;
```

Índices - Table Clusters

- Utilize se as tabelas são frequentemente utilizadas em JOINs (NF e ITEM_NF);
- Utilize se as tabelas sofrem mais JOINs do que operações DML;
- Evite se é utilizado FTS de apenas uma das tabelas;
- Evite para linhas largas (PEDIDO_NOTA_FISCAL com OBS);
- Evite se o número de linhas nas chaves das duas tabelas é muito diferente.



Índices e Access Paths

Access Path	Heap-Organized Tables	B-Tree Indexes and IOTs	Bitmap Indexes	Table Clusters
Full Table Scans	x			
Table Access by Rowid	x			
Sample Table Scans	x			
Index Unique Scans		x		
Index Range Scans		x		
Index Full Scans		x		
Index Fast Full Scans		x		
Index Skip Scans		x		
Index Join Scans		x		

Índices e Access Paths

Access Path	Heap-Organized Tables	B-Tree Indexes and IOTs	Bitmap Indexes	Table Clusters
Bitmap Index Single Value			x	
Bitmap Index Range Scans			x	
Bitmap Merge			x	
Bitmap Index Range Scans			x	
Cluster Scans				x
Hash Scans				x

Índices - Guidelines

Geral

- Controle: HINTs INDEX, INDEX_COMBINE, NO_INDEX, FULL;
- Crie índices em colunas utilizadas na cláusula WHERE;
- Crie índices em colunas utilizadas em JOINs;
- Crie índices em colunas de alta seletividade;
- Crie índices em colunas de baixa seletividade mas que contenham dados com seletividades muito distintas;
- Prefira índices PRIMARY KEY, se o modelo permitir;
- Prefira índices UNIQUE, se o modelo permitir, mas PRIMARY KEY não é possível;
- Crie índices compostos em colunas utilizadas frequentemente na mesma cláusula WHERE;
- Em índices compostos, utilize as colunas com maior seletividade à esquerda;
- Se um valor de uma coluna indexada não for utilizado em uma cláusula WHERE, verifique se este valor pode ser trocado para NULL;
- Busque sempre minimizar a quantidade de índices de uma tabela;
- Considere o espaço utilizado por um índice (60% - 40%).

Índices - Guidelines

DML

- Crie índices em Foreign Keys (FKs) que sofrem DML de forma concorrente;
- Evite índices em colunas que sofrem muitos UPDATEs;
- Evite índices em tabelas que sofrem muitos INSERTs ou DELETEs.

Tipos

- Prefira índices BTREE em colunas de alta seletividade (CPF, NF);
- Prefira índices BITMAP em colunas de baixa seletividade (ESTADO, CIDADE);
- Evite índices em colunas utilizadas em cláusula WHERE apenas com funções;
- Prefira índices BITMAP para grandes tabelas;
- Evite índices BITMAP em colunas que sofrem muito DML, principalmente de forma concorrente;
- Prefira partições HASH em índices pequenos que sofrem DML em alta concorrência;
- Utilize IOTs em PKs frequentemente utilizadas na cláusula WHERE;
- Utilize Function Based Index em colunas utilizadas em cláusula WHERE mais frequentemente com funções;
- Utilize índices REVERSE em colunas que sofrem DML em alta concorrência;
- Se um valor de uma coluna indexada utilizado em uma cláusula WHERE for raro, considere um Function Based Index:

```
CREATE INDEX IDX_ORDER_NEW
ON ORDERS(CASE STATUS WHEN 'N' THEN 'N' ELSE NULL END);
```

BITMAP x BTREE

```
SQL> SELECT COUNT(*) FROM T4;  - - Sem índice.
COUNT(1)
___
10936000

SQL> SELECT COUNT(DISTINCT(OWNER)) FROM T; - - Sem índice.
COUNT(DISTINCT(OWNER))
_______
28
Decorrido: 00:00:26.75

SQL> SELECT COUNT(DISTINCT(OWNER)) FROM T; - - Com índice BTREE.
COUNT(DISTINCT(OWNER))
_______
28
Decorrido: 00:00:05.29

SQL> SELECT COUNT(DISTINCT(OWNER)) FROM T; - - Com índice BITMAP.
COUNT(DISTINCT(OWNER))
_______
28
Decorrido: 00:00:01.84
```

BITMAP x BTREE

```
SQL> SELECT COUNT(*) FROM T;
COUNT(1)
___
10936000

SQL> SELECT COUNT(DISTINCT(OWNER)) FROM T;
COUNT(DISTINCT(OWNER))
_______
28

SQL> SELECT COUNT(DISTINCT(OBJECT_NAME)) FROM T; -- Com índice BTREE
COUNT(DISTINCT(OBJECT_NAME))
_________
40998
Decorrido: 00:00:08.95


SQL> SELECT COUNT(DISTINCT(OBJECT_NAME)) FROM T; – Com índice BITMAP
COUNT(DISTINCT(OBJECT_NAME))
_________
40998
Decorrido: 00:00:03.15
```

Lab 9.1: DML e BITMAP Index

1ª Sessão:

```
SQL> CREATE BITMAP INDEX
IDX_BITMAP_T314 ON T314(C1);

SQL> INSERT INTO T314 VALUES (1);

SQL> COMMIT;

SQL> INSERT INTO T314 VALUES (1);

SQL> COMMIT;

SQL> INSERT INTO T314 VALUES (1);

SQL> INSERT INTO T314 VALUES (10);
```

2ª Sessão:

```
SQL> INSERT INTO T314 VALUES (10);

SQL> COMMIT;

SQL> INSERT INTO T314 VALUES (1);

COMMIT;

SQL> INSERT INTO T314 VALUES (10);

SQL> INSERT INTO T314 VALUES (1);
```

Lab 9.2: Impacto de Índices

Execute novamente o melhor script de INSERT, mas com a adição de índices como abaixo.

```
$ cd /home/oracle
$ time perl ConnectGOOD_CommitGOOD_BindsGOOD_PERFECT.pl 10000
$ time perl ConnectGOOD_CommitGOOD_BindsGOOD_PERFECT.pl 10000
$ time perl ConnectGOOD_CommitGOOD_BindsGOOD_PERFECT.pl 10000

SQL> DROP INDEX IDX_BITMAP_T314;
SQL> CREATE INDEX IDX_BTREE_T314 ON T314(C1);
$ time perl ConnectGOOD_CommitGOOD_BindsGOOD_PERFECT.pl 10000
$ time perl ConnectGOOD_CommitGOOD_BindsGOOD_PERFECT.pl 10000
$ time perl ConnectGOOD_CommitGOOD_BindsGOOD_PERFECT.pl 10000

SQL> DROP INDEX IDX_BTREE_T314;
$ time perl ConnectGOOD_CommitGOOD_BindsGOOD_PERFECT.pl 10000
$ time perl ConnectGOOD_CommitGOOD_BindsGOOD_PERFECT.pl 10000
$ time perl ConnectGOOD_CommitGOOD_BindsGOOD_PERFECT.pl 10000
```

Lab 9.3: Impacto de Índices

Com o usuário SCOTT, crie uma tabela de testes, e verifique o tempo de sua duplicação.

```
SQL> CREATE TABLE T91 AS SELECT * FROM ALL_OBJECTS;
SQL> INSERT INTO T91 SELECT * FROM T91;
SQL> INSERT INTO T91 SELECT * FROM T91;
SQL> INSERT INTO T91 SELECT * FROM T91;
SQL> INSERT INTO T91 SELECT * FROM T91;
SQL> COMMIT;

SQL> CREATE TABLE T92 AS SELECT * FROM T91 WHERE 1=0;
SQL> SET TIMING ON

SQL> INSERT INTO T92 SELECT * FROM T91;
SQL> TRUNCATE TABLE T92;
SQL> INSERT INTO T92 SELECT * FROM T91;
SQL> TRUNCATE TABLE T92;
SQL> INSERT INTO T92 SELECT * FROM T91;
SQL> TRUNCATE TABLE T92;
```

Lab 9.4: Impacto de Índices

Verifique o tempo de sua duplicação, mas com índices.

```
SQL> CREATE INDEX T92_IDX_01 ON T92(OWNER);
SQL> INSERT INTO T92 SELECT * FROM T91;
SQL> TRUNCATE TABLE T92;
SQL> INSERT INTO T92 SELECT * FROM T91;
SQL> TRUNCATE TABLE T92;
SQL> INSERT INTO T92 SELECT * FROM T91;
SQL> TRUNCATE TABLE T92;

SQL> CREATE INDEX T92_IDX_02 ON T92(OBJECT_NAME);
SQL> INSERT INTO T92 SELECT * FROM T91;
SQL> TRUNCATE TABLE T92;
SQL> INSERT INTO T92 SELECT * FROM T91;
SQL> TRUNCATE TABLE T92;
SQL> INSERT INTO T92 SELECT * FROM T91;
SQL> TRUNCATE TABLE T92;
```

Lab 9.5: Impacto de Índices

Verifique o tempo de sua duplicação, mas um índice composto.

```
SQL> DROP INDEX T92_IDX_01;
SQL> DROP INDEX T92_IDX_02;
SQL> CREATE INDEX T92_IDX_03 ON T92(OWNER,OBJECT_NAME);
SQL> INSERT INTO T92 SELECT * FROM T91;
SQL> TRUNCATE TABLE T92;
SQL> INSERT INTO T92 SELECT * FROM T91;
SQL> TRUNCATE TABLE T92;
SQL> INSERT INTO T92 SELECT * FROM T91;
SQL> TRUNCATE TABLE T92;
```

Lab 9.6: Impacto de Índices

Verifique o uso dos índices.

```
SQL> CONN SCOTT/TIGER
SQL> DROP INDEX T92_IDX_03;
SQL> INSERT INTO T92 SELECT * FROM T91;
SQL> CREATE INDEX T92_IDX_01 ON T92(OWNER);
SQL> CREATE INDEX T92_IDX_02 ON T92(OBJECT_NAME);
SQL> ALTER INDEX T92_IDX_01 MONITORING USAGE;
SQL> ALTER INDEX T92_IDX_02 MONITORING USAGE;

SQL> COL INDEX_NAME FORMAT A40
SQL> SELECT INDEX_NAME, MONITORING, USED FROM V$OBJECT_USAGE;
SQL> SELECT * FROM T92 WHERE OWNER = 'SCOTT';
SQL> SELECT COUNT(*) FROM T92 WHERE OWNER = 'SYS';
SQL> SELECT COUNT(*) FROM T92 WHERE OWNER = 'SYSTEM';
SQL> SELECT INDEX_NAME, MONITORING, USED, START_MONITORING
FROM V$OBJECT_USAGE;
```

Problemas a se considerar ao remover índices:

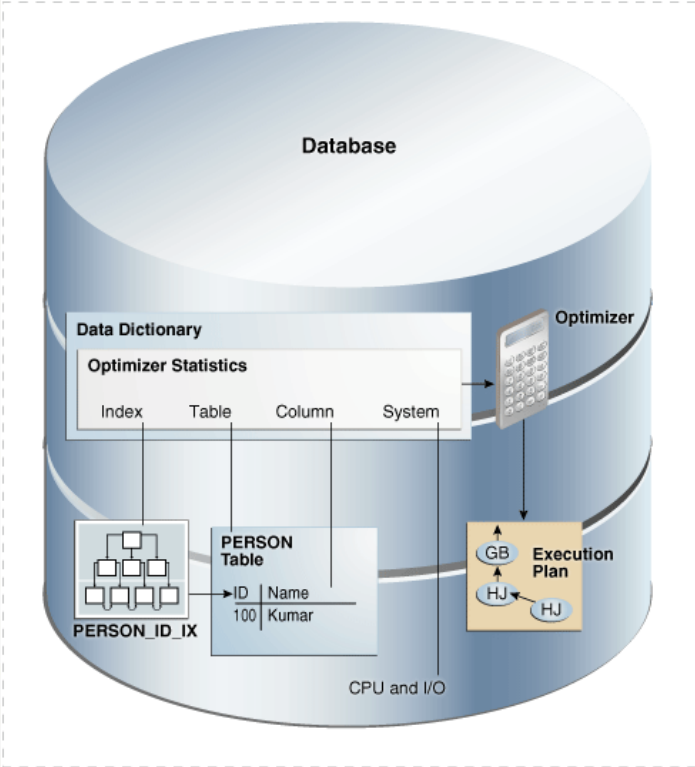
- Não está utilizando o índice, mas deveria utilizar;
- Após o DROP, não é utilizado outro índice;
- Uso da seletividade em índices compostos, mesmo sem utilizar a coluna;
- FKs (Enqueue TM).

Estatísticas

Estatísticas e SQL Engine

Optimizer Statistics

- Table statistics
  - Number of rows
  - Number of blocks
  - Average row length
- Column statistics
  - Number of distinct values (NDV) in column
  - Number of nulls in column
  - Data distribution (histogram)
  - Extended statistics
- Index statistics
  - Number of leaf blocks
  - Average data blocks per Key
  - Levels
  - Index clustering factor



System Statistics

- I/O performance and utilization
- CPU performance and utilization

Estatísticas - Tabela

DBA_TABLES / ALL_TABLES / USER_TABLES

```
SELECT TABLE_NAME, NUM_ROWS, BLOCKS, EMPTY_BLOCKS, AVG_SPACE,
CHAIN_CNT, AVG_ROW_LEN, SAMPLE_SIZE, LAST_ANALYZED FROM USER_TABLES
ORDER BY 1;
```

TABLE_NAME	NUM_ROWS	BLOCKS	EMPTY_BLOCKS	AVG_SPACE	CHAIN_CNT	AVG_ROW_LEN	SAMPLE_SIZE	LAST_ANALYZED
BONUS	0	0	0	0	0	0	0	10/03/15
DEPT	4	5	0	0	0	20	4	17/03/15
EMP	14	5	0	0	0	38	14	17/03/15
LAB3	619472	10150	0	0	0	110	619472	24/03/15
LAB314	0	0	0	0	0	0	0	24/03/15
LAB5	755225	12414	0	0	0	110	755225	24/03/15
SALGRADE	5	5	0	0	0	10	5	10/03/15
T	1177984	19941	0	0	0	110	1177984	16/03/15
T1	10000	164	0	0	0	104	10000	24/03/15
T2	10000	164	0	0	0	104	10000	24/03/15
T3	109989	244	0	0	0	4	109989	15/03/15
T314	199156	370	0	0	0	4	199156	24/03/15
T315	79992	244	0	0	0	4	79992	24/03/15
T4	77442	1275	0	0	0	110	77442	24/03/15
T7	1239136	20325	0	0	0	110	1239136	24/03/15
T73	1239104	20325	0	0	0	110	1239104	24/03/15
T732	0	0	0	0	0	0	0	24/03/15
T8	0	0	0	0	0	0	0	24/03/15

Estatísticas - Índices

DBA_INDEXES / ALL_INDEXES / USER_INDEXES

```
SELECT TABLE_NAME, INDEX_NAME, NUM_ROWS, BLEVEL, LEAF_BLOCKS,
DISTINCT_KEYS, CLUSTERING_FACTOR, AVG_LEAF_BLOCKS_PER_KEY,
AVG_DATA_BLOCKS_PER_KEY, SAMPLE_SIZE, LAST_ANALYZED FROM
USER_INDEXES ORDER BY 1,2;
```

TABLE_NAME	INDEX_NAME	NUM_ROWS	BLEVEL	LEAF_BLOCKS	DISTINCT_KEYS	CLUSTERING_FACTOR	AVG_LEAF_BLOCKS_PER_KEY	AVG_DATA_BLOCKS_PER_KEY	SAMPLE_SIZE	LAST_ANALYZED
DEPT	PK_DEPT	4	0	1	4	1	1	1	4	17/03/15
EMP	PK_EMP	14	0	1	14	1	1	1	14	17/03/15
T	T_IDX01	1177984	2	3318	23	43194	144	1878	1177984	16/03/15
T1	T1_IDX1	10000	1	20	100	152	1	1	10000	24/03/15
T2	T2_IDX1	10000	1	20	100	10000	1	100	10000	24/03/15
T3	IDX_BTREE_T3	109989	1	230	9999	109989	1	11	109989	24/03/15

Estatísticas - Colunas

DBA_TAB_COLUMNS / ALL_TAB_COLUMNS / USER_TAB_COLUMNS

```
SELECT TABLE_NAME, COLUMN_NAME, NUM_DISTINCT, NUM_NULLS, DENSITY,
LOW_VALUE, HIGH_VALUE, DATA_LENGTH, AVG_COL_LEN, SAMPLE_SIZE,
LAST_ANALYZED  FROM USER_TAB_COLUMNS ORDER BY 1,2;
```

TABLE_NAME	COLUMN_NAME	NUM_DISTINCT	NUM_NULLS	DENSITY	LOW_VALUE	HIGH_VALUE	DATA_LENGTH	AVG_COL_LEN	SAMPLE_SIZE	LAST_ANALYZED
T	CREATED	652	0	0,00153374233128834	78720707062804	7873030F0C322F	7	8	1177984	16/03/15
T	DATA_OBJECT_ID	193	1174896	0,00518134715025907	C2022B	C30A2414	22	2	3088	16/03/15
T	EDITIONABLE	2	514352	0,5	4E	59	1	2	663632	16/03/15
T	EDITION_NAME	0	1177984	0	(null)	(null)	128	0	(null)	16/03/15
T	GENERATED	2	0	0,5	4E	59	1	2	1177984	16/03/15
T	LAST_DDL_TIME	634	0	0,00157728706624606	78660A010D2A32	7873030F0C322F	7	8	1177984	16/03/15
T	NAMESPACE	7	0	0,142857142857143	C102	C141	22	3	1177984	16/03/15
T	OBJECT_ID	74672	0	0,0000133919005785301	C20222	C30A2414	22	5	1177984	16/03/15
T	OBJECT_NAME	40208	0	0,0000248706725029845	2F31303030333...	79436243725375...	128	26	1177984	16/03/15
T	OBJECT_TYPE	23	0	0,0000004244539823...	434F4E53554D4...	584D4C20534348...	23	10	1177984	16/03/15
T	ORACLE_MAINTAINED	2	0	0,5	4E	59	1	2	1177984	16/03/15
T	OWNER	17	0	0,0588235294117647	415045585F303...	584442	128	6	1177984	16/03/15
T	SECONDARY	1	0	1	4E	4E	1	2	1177984	16/03/15
T	SHARING	3	0	0,333333333333333	4D45544144415...	4F424A45435420...	13	8	1177984	16/03/15
T	STATUS	1	0	1	56414C4944	56414C4944	7	6	1177984	16/03/15

Coleta de Estatísticas - ANALYZE

```
ANALYZE TABLE emp VALIDATE STRUCTURE;
ANALYZE TABLE emp VALIDATE STRUCTURE CASCADE;
ANALYZE TABLE emp VALIDATE STRUCTURE CASCADE FAST;
ANALYZE TABLE emp VALIDATE STRUCTURE CASCADE ONLINE;

utlchain.sql / utlchn1.sql
ANALYZE TABLE emp LIST CHAINED ROWS INTO CHAINED_ROWS;
```

Note:

Do not use the `COMPUTE` and `ESTIMATE` clauses of `ANALYZE` to collect optimizer statistics. These clauses have been deprecated. Instead, use the `DBMS_STATS` package, which lets you collect statistics in parallel, collect global statistics for partitioned objects, and fine tune your statistics collection in other ways. The cost-based optimizer, which depends upon statistics, will eventually use only statistics that have been collected by `DBMS_STATS`. See *Oracle Database PL/SQL Packages and Types Reference* for more information on the `DBMS_STATS` package.

You must use the `ANALYZE` statement (rather than `DBMS_STATS`) for statistics collection not related to the cost-based optimizer, such as:

- To use the `VALIDATE` or `LIST CHAINED ROWS` clauses
- To collect information on freelist blocks

Quando coletar? Coleta automática

Table 26-1 Predefined Maintenance Windows

Window Name	Description
MONDAY_WINDOW	Starts at 10 p.m. on Monday and ends at 2 a.m.
TUESDAY_WINDOW	Starts at 10 p.m. on Tuesday and ends at 2 a.m.
WEDNESDAY_WINDOW	Starts at 10 p.m. on Wednesday and ends at 2 a.m.
THURSDAY_WINDOW	Starts at 10 p.m. on Thursday and ends at 2 a.m.
FRIDAY_WINDOW	Starts at 10 p.m. on Friday and ends at 2 a.m.
SATURDAY_WINDOW	Starts at 6 a.m. on Saturday and is 20 hours long.
SUNDAY_WINDOW	Starts at 6 a.m. on Sunday and is 20 hours long.

Quando coletar? Coleta automática

```
BEGIN
  DBMS_AUTO_TASK_ADMIN.DISABLE (
    client_name  => 'auto optimizer stats collection'
,   operation    => NULL
,   window_name  => NULL
);
END;
/
```

```
SQL> SELECT CLIENT_NAME, STATUS FROM DBA_AUTOTASK_CLIENT;

CLIENT_NAME                                                      STATUS
---------------------------------------------------------------- --------
auto optimizer stats collection                                  ENABLED
auto space advisor                                               ENABLED
sql tuning advisor                                               DISABLED
```

```
BEGIN
  DBMS_AUTO_TASK_ADMIN.ENABLE (
    client_name  => 'auto optimizer stats collection'
,   operation    => NULL
,   window_name  => NULL
);
END;
/
```

Quando coletar? Coleta automática

```
BEGIN
  DBMS_AUTO_TASK_ADMIN.disable(
    client_name => 'auto optimizer stats collection',
    operation   => NULL,
    window_name => 'MONDAY_WINDOW');

  DBMS_AUTO_TASK_ADMIN.disable(
    client_name => 'auto space advisor',
    operation   => NULL,
    window_name => 'MONDAY_WINDOW');

  DBMS_AUTO_TASK_ADMIN.disable(
    client_name => 'sql tuning advisor',
    operation => NULL,
    window_name => 'MONDAY_WINDOW');
END;
/
```

```
BEGIN
  DBMS_SCHEDULER.disable(name => 'SYS.MONDAY_WINDOW', force => TRUE);

  DBMS_SCHEDULER.set_attribute(
    name      => 'SYS.MONDAY_WINDOW',
    attribute => 'DURATION',
    value     => numtodsinterval(180, 'minute'));

  DBMS_SCHEDULER.enable(name=>'SYS.MONDAY_WINDOW');
END;
/
```

```
BEGIN
  DBMS_SCHEDULER.SET_ATTRIBUTE (
    'MONDAY_WINDOW'
,   'repeat_interval'
,   'freq=daily;byday=MON;byhour=05;byminute=0;bysecond=0'
);
END;
/
```

Quando coletar? OPTIMIZER_DYNAMIC_SAMPLING

Nível 0 = Não há coleta.
Nível 1 = Coleta 32 blocos.
Se há pelo menos 1 tabela particionada no SQL sem estatísticas.
Se esta tabela não tem índices.
Se esta tabela tem mais que 64 blocos.
Nível 2 = Coleta 64 blocos.
Coleta se há pelo menos uma tabela do SQL sem estatísticas.
Nível 3 = Coleta 64 blocos.
Coleta se o Nível 2 é atendido OU se é utilizada expressão no WHERE.
Nível 4 = Coleta 64 blocos.
Coleta se o nível 3 é atendido OU se o SQL utiliza AND ou OR entre múltiplos predicados.
Nível 5 = Coleta 128 blocos.
Coleta se o nível 4 é atendido.
Nível 6 = Coleta 256 blocos.
Coleta se o nível 4 é atendido.
Nível 7 = Coleta 512 blocos.
Coleta se o nível 4 é atendido.
Nível 8 = Coleta 1024 blocos.
Coleta se o nível 4 é atendido.
Nível 9 = Coleta 4086 blocos.
Coleta se o nível 4 é atendido.
Nível 10 = Coleta todos os blocos.
Coleta se o nível 4 é atendido.
Nível 11 (Adaptive Dynamic Sampling: >= 11.2.0.4) = Coleta ? Blocos. Coleta quando?

Lab 10.1: Coleta de Estatísticas

Verifique os planos de execução dos SQL abaixo.

```
SQL> CONN SCOTT/TIGER
SQL> SET AUTOTRACE ON EXPLAIN
SQL> SELECT ID, NAME FROM T1 WHERE ID = 1;
SQL> SELECT ID, NAME FROM T2 WHERE ID = 1;
SQL> SELECT ID, NAME FROM T1 WHERE ID < 5;
SQL> SELECT ID, NAME FROM T2 WHERE ID < 5;
SQL> SELECT ID, NAME FROM T1 WHERE ID < 10;
SQL> SELECT ID, NAME FROM T2 WHERE ID < 10;
```

Remova as estatísticas e verifique novamente os planos de execução.

```
SQL> EXEC DBMS_STATS.DELETE_TABLE_STATS('SCOTT','T1');
SQL> EXEC DBMS_STATS.DELETE_TABLE_STATS('SCOTT','T2');
```

Desabilite as estatísticas dinâmicas, e verifique novamente os planos de execução.

```
SQL> ALTER SESSION SET OPTIMIZER_DYNAMIC_SAMPLING=0;
```

Colete as estatísticas com os parâmetros padrão, e verifique novamente os planos de execução.

```
SQL> EXEC DBMS_STATS.GATHER_TABLE_STATS('SCOTT','T1');
SQL> EXEC DBMS_STATS.GATHER_TABLE_STATS('SCOTT','T2');
```

Estatísticas: Coleta Manual

Coleta de todos os objetos.

```
SQL> EXEC DBMS_STATS.GATHER_DATABASE_STATS;
SQL> EXEC DBMS_STATS.GATHER_SCHEMA_STATS('SOE');
SQL> EXEC DBMS_STATS.GATHER_TABLE_STATS('SOE','CUSTOMERS');
SQL> EXEC DBMS_STATS.GATHER_INDEX_STATS('SOE','CUSTOMERS_PK');
```

Coleta apenas de objetos EMPTY e STALE.

```
SQL> EXEC DBMS_STATS.GATHER_DATABASE_STATS(OPTIONS=>'GATHER EMPTY');
SQL> EXEC DBMS_STATS.GATHER_SCHEMA_STATS('SOE',OPTIONS=>'GATHER EMPTY');

SQL> EXEC DBMS_STATS.GATHER_DATABASE_STATS(OPTIONS=>'GATHER STALE');
SQL> EXEC DBMS_STATS.GATHER_SCHEMA_STATS('SOE',OPTIONS=>'GATHER STALE');
```


228

Como coletar? Opções

ESTIMATE_PERCENT

DBMS_STATS.AUTO_SAMPLE_SIZE / N

BLOCK_SAMPLE

FALSE / TRUE

DEGREE

NULL / N

GRANULARITY

AUTO / ALL / DEFAULT / GLOBAL / GLOBAL AND PARTITION / PARTITION / SUBPARTITION

CASCADE

DBMS_STATS.AUTO_CASCADE / TRUE / FALSE

OPTIONS

GATHER / GATHER AUTO / GATHER STALE / GATHER EMPTY

GATHER_SYS

TRUE / FALSE

NO_INVALIDATE

DBMS_STATS.AUTO_INVALIDATE / TRUE / FALSE

Estatísticas - Histogramas

DBA_TAB_COLUMNS / ALL_TAB_COLUMNS / USER_TAB_COLUMNS
DBA_TAB_HISTOGRAMS / ALL_TAB_HISTOGRAMS / USER_TAB_HISTOGRAMS

```
SELECT H.TABLE_NAME, H.COLUMN_NAME, C.HISTOGRAM, H.ENDPOINT_NUMBER,
H.ENDPOINT_ACTUAL_VALUE, H.ENDPOINT_REPEAT_COUNT FROM
USER_TAB_HISTOGRAMS H, USER_TAB_COLUMNS C WHERE H.TABLE_NAME =
C.TABLE_NAME AND H.COLUMN_NAME = C.COLUMN_NAME AND HISTOGRAM <>
'NONE' ORDER BY 1,2,4;
```

TABLE_NAME	COLUMN_NAME	HISTOGRAM	NUM_BUCKETS	NUM_DISTINCT	LAST_ANALYZED	SAMPLE_SIZE
EMP	DEPTNO	FREQUENCY	3	3	17/03/15	14
T	OBJECT_TYPE	FREQUENCY	23	23	16/03/15	1177984
T1	ID	FREQUENCY	100	100	24/03/15	10000
T2	ID	FREQUENCY	100	100	24/03/15	10000

TABLE_NAME	COLUMN_NAME	HISTOGRAM	ENDPOINT_NUMBER	ENDPOINT_ACTUAL_VALUE	ENDPOINT_REPEAT_COUNT
EMP	DEPTNO	FREQUENCY	3	10	0
EMP	DEPTNO	FREQUENCY	8	20	0
EMP	DEPTNO	FREQUENCY	14	30	0
T	OBJECT_TYPE	FREQUENCY	32	CONSUMER GROUP	0
T	OBJECT_TYPE	FREQUENCY	64	DESTINATION	0
T	OBJECT_TYPE	FREQUENCY	80	EDITION	0
T	OBJECT_TYPE	FREQUENCY	96	EVALUATION CONTEXT	0
T	OBJECT_TYPE	FREQUENCY	4112	FUNCTION	0
T	OBJECT_TYPE	FREQUENCY	5840	INDEX	0
T	OBJECT_TYPE	FREQUENCY	5952	INDEXTYPE	0
T	OBJECT_TYPE	FREQUENCY	497872	JAVA CLASS	0
T	OBJECT_TYPE	FREQUENCY	514064	JAVA RESOURCE	0
T	OBJECT_TYPE	FREQUENCY	514288	JOB CLASS	0
T	OBJECT_TYPE	FREQUENCY	515152	OPERATOR	0
T	OBJECT_TYPE	FREQUENCY	522208	PACKAGE	0

Como coletar? Histogramas

METHOD_OPT

FOR ALL COLUMNS SIZE AUTO

FOR ALL [INDEXED | HIDDEN] COLUMNS SIZE [N | REPEAT | AUTO | SKEWONLY]

FOR COLUMNS *column* SIZE [N | REPEAT | AUTO | SKEWONLY]

Exemplos:

FOR ALL COLUMNS **SIZE 1**
FOR ALL COLUMNS SIZE 100
FOR ALL COLUMNS SIZE AUTO
FOR ALL COLUMNS SIZE REPEAT
FOR ALL COLUMNS SIZE SKEWONLY
FOR ALL INDEXED COLUMNS **SIZE 1**
FOR ALL INDEXED COLUMNS SIZE 100
FOR ALL INDEXED COLUMNS SIZE AUTO
FOR ALL INDEXED COLUMNS SIZE REPEAT
FOR ALL INDEXED COLUMNS SIZE SKEWONLY
FOR COLUMNS C1 **SIZE 1**
FOR COLUMNS C1 SIZE 100
FOR COLUMNS C1 SIZE AUTO
FOR COLUMNS C1 SIZE REPEAT
FOR COLUMNS C1 SIZE SKEWONLY

Lab 10.2: Coleta de Estatísticas

Verifique os planos de execução dos SQL abaixo.

```
SQL> SELECT COUNT(OBJECT_TYPE), OBJECT_TYPE FROM T91 GROUP BY OBJECT_TYPE
ORDER BY 1;
SQL> CREATE INDEX IND_T91_01 ON T91(OBJECT_TYPE);
SQL> EXEC DBMS_STATS.DELETE_TABLE_STATS('SCOTT','T91');
SQL> SET AUTOTRACE ON EXPLAIN
SQL> SELECT COUNT(OBJECT_NAME) FROM T91 WHERE OBJECT_TYPE = 'SYNONYM';
SQL> SELECT COUNT(OBJECT_NAME) FROM T91 WHERE OBJECT_TYPE = 'PACKAGE';
SQL> SELECT COUNT(OBJECT_NAME) FROM T91 WHERE OBJECT_TYPE = 'TABLE';
SQL> SELECT COUNT(OBJECT_NAME) FROM T91 WHERE OBJECT_TYPE = 'DIRECTORY';
```

Habilite o Dynamic Sampling e verifique novamente os planos de execução.

```
SQL> ALTER SESSION SET OPTIMIZER_DYNAMIC_SAMPLING=2;
```

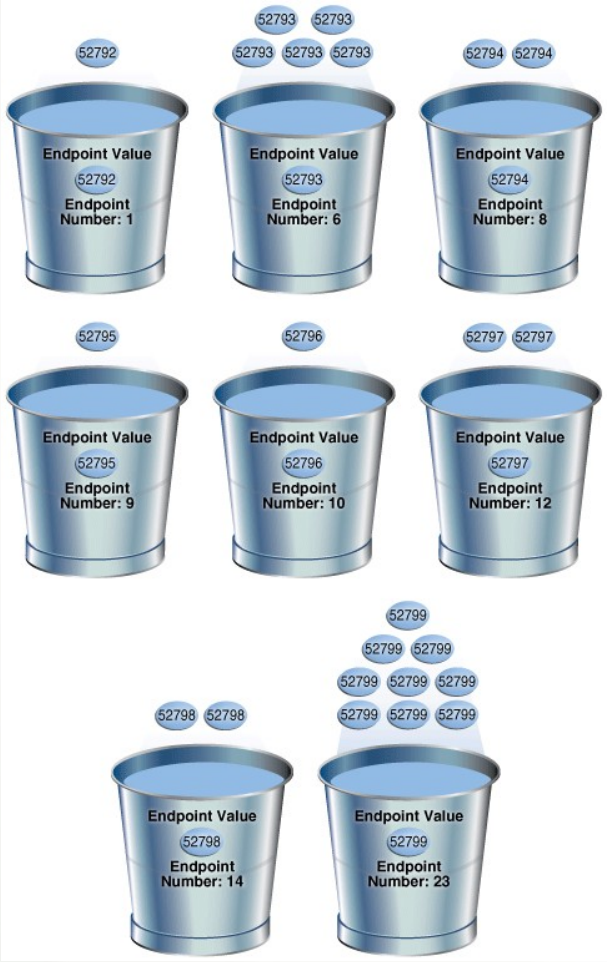
Colete as estatísticas e verifique novamente os planos de execução.

```
SQL> EXEC DBMS_STATS.GATHER_TABLE_STATS('SCOTT','T91', METHOD_OPT=>'FOR
COLUMNS OBJECT_TYPE SIZE AUTO');
SQL> EXEC DBMS_STATS.GATHER_TABLE_STATS('SCOTT','T91', METHOD_OPT=>'FOR
COLUMNS OBJECT_TYPE SIZE 10');
SQL> EXEC DBMS_STATS.GATHER_TABLE_STATS('SCOTT','T91', METHOD_OPT=>'FOR
COLUMNS OBJECT_TYPE SIZE 5');
SQL> EXEC DBMS_STATS.GATHER_TABLE_STATS('SCOTT','T91', METHOD_OPT=>'FOR
COLUMNS OBJECT_TYPE SIZE 1');
```

Frequency Histograms

Utilizados se:

- Se o NDV é menor ou igual que a quantidade de Buckets indicados na coleta (ou o máximo);
- *É utilizado AUTO_SAMPLE_SIZE na execução da coleta.*



Height Balanced Histograms

Utilizados se:

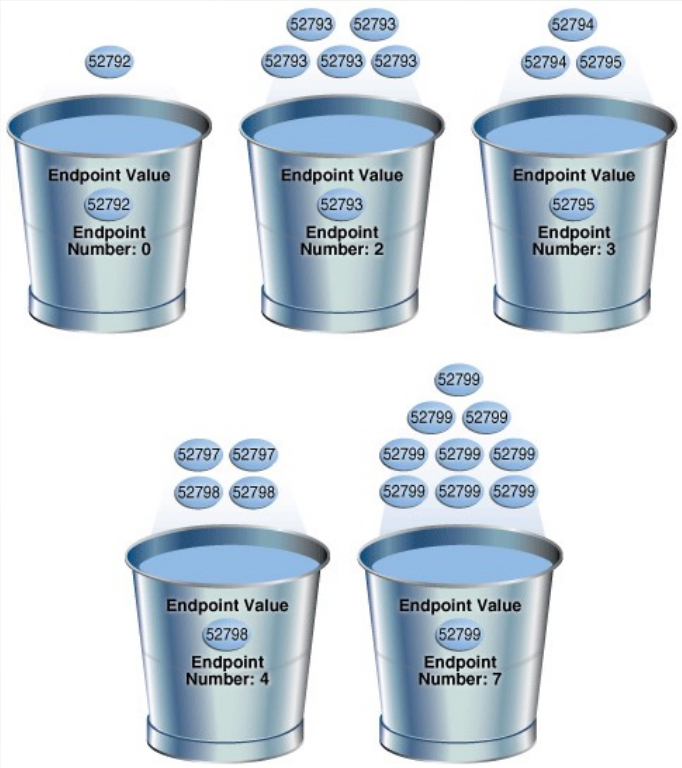
- Se o número de Buckets indicados na coleta (ou o máximo) é menor que o NDV.

```
5, 6, 6, 6, 9, 11, 11, 12, 12, 12, 12, 12, 13, 13, 13, 13, 13, 16, 17, 17
```

```
5, 6, 6, 6, 9, 11, 11, 12, 12, 12, 12, 12, 13, 13, 13, 13, 13, 16, 17, 17
```

0	5
1	6
2	12
3	12
4	13
5	17

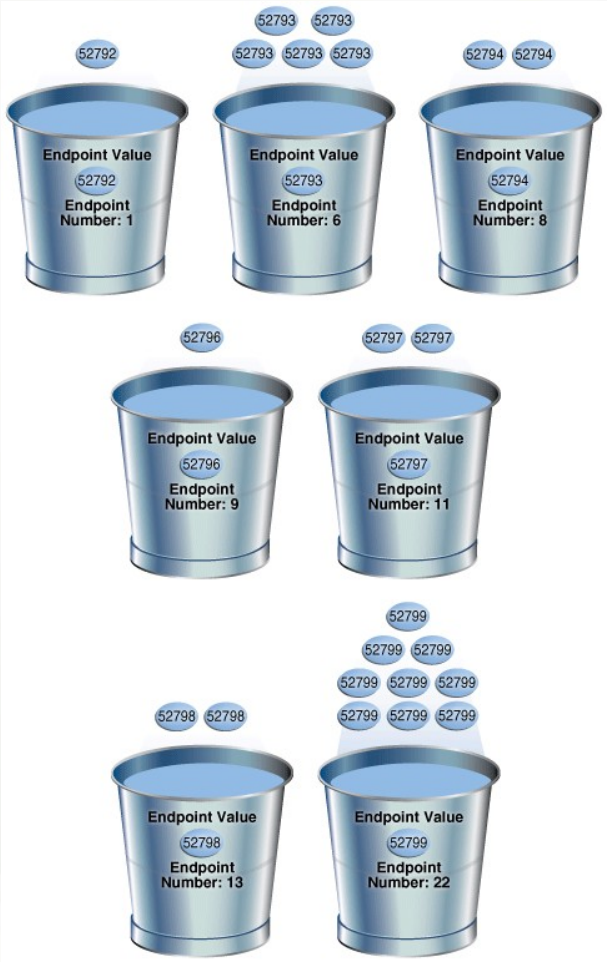
0	5
1	6
3	12
4	13
5	17



Top Frequency Histograms (12c)

Utilizados se:

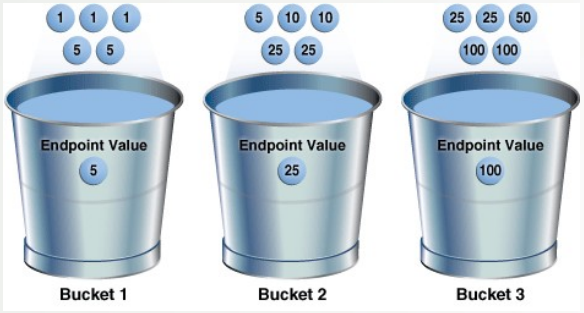
- Se o NDV é maior que a quantidade de Buckets indicados na coleta (ou o máximo);
- É utilizado AUTO_SAMPLE_SIZE na execução da coleta;
- Se o percentual de linhas ocupadas pelos Top Values é igual ou maior que p, sendo que p = (1-(1/Buckets))*100.

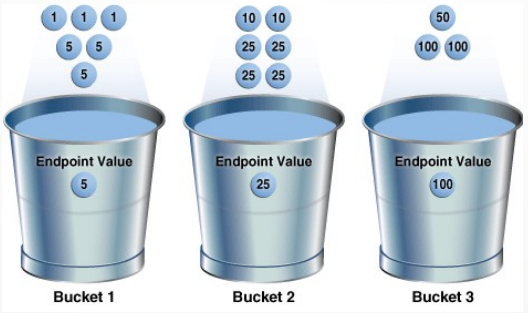


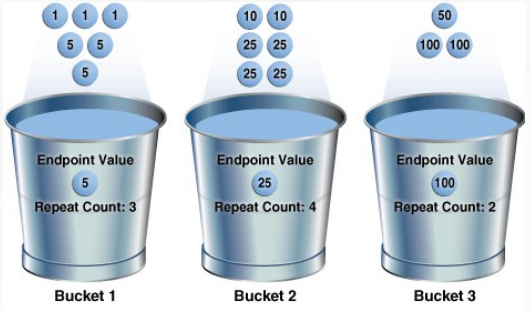
Hybrid Histograms (12c)

Utilizados se:

- Se o número de Buckets indicados na coleta (ou o máximo) é menor que o NDV;
- É utilizado AUTO_SAMPLE_SIZE na execução da coleta;
- Se os critétios para Top Frequency Histograms não se aplicam.







Histogramas

- Buckets: máximo de 254 / 127 (2048 no 12c);
- Frequency Histograms;
- Height-Balanced Histograms;
- Top Frequency Histograms (12c);
- Hybrid Histograms (12c).

Problemas:

- Colunas que não precisam de Histogramas;
- Custo de coleta;
- Sensibilidade a coleta por SAMPLE;
- Sensibilidade a momento da coleta (Novos pedidos às 22:00?);
- Bind Variables (pré ACS);
- Frequency: Se um valor não está em um Bucket, será considerada metade da cardinalidade do valor menos popular;
- Height-Balanced: Se um valor não é popular (só está em 1 Bucket), será considerada a cardinalidade = número de linhas não populares / número de valor não populares (11gR2).

http://docs.oracle.com/database/121/TGSQL/tgsql_histo.htm

Como coletar?

Coleta geral

```
EXEC DBMS_STATS.GATHER_DATABASE_STATS
(ESTIMATE_PERCENT=>DBMS_STATS.AUTO_SAMPLE_SIZE,
BLOCK_SAMPLE=>FALSE,
DEGREE=>8,
GRANULARITY=>'AUTO',
CASCADE=>TRUE,
OPTIONS=>'GATHER EMPTY',
GATHER_SYS=>FALSE,
NO_INVALIDATE=>FALSE,
METHOD_OPT=>'FOR ALL COLUMNS SIZE AUTO');

EXEC DBMS_STATS.GATHER_DATABASE_STATS
(ESTIMATE_PERCENT=>DBMS_STATS.AUTO_SAMPLE_SIZE,
BLOCK_SAMPLE=>FALSE,
DEGREE=>8,
GRANULARITY=>'AUTO',
CASCADE=>TRUE,
OPTIONS=>'GATHER STALE',
GATHER_SYS=>FALSE,
NO_INVALIDATE=>FALSE,
METHOD_OPT=>'FOR ALL COLUMNS SIZE AUTO');
```

Como coletar?

Controle de Opções

```
SQL> EXEC DBMS_STATS.SET_DATABASE_PREFS('DEGREE','2');
SQL> EXEC DBMS_STATS.SET_SCHEMA_PREFS('SOE','CASCADE','TRUE');
SQL> EXEC DBMS_STATS.SET_TABLE_PREFS('SOE','CUSTOMERS','STALE_PERCENT',5);
```

- CASCADE
- DEGREE
- ESTIMATE_PERCENT
- GRANULARITY
- INCREMENTAL
- INCREMENTAL_LEVEL
- INCREMENTAL_STALENESS
- METHOD_OPT
- NO_INVALIDATE
- PUBLISH
- STALE_PERCENT
- TABLE_CACHED_BLOCKS
- OPTIONS

Como coletar?

Coleta por exceção

```
EXEC DBMS_STATS.UNLOCK_TABLE_STATS('SCOTT','EMP');

EXEC DBMS_STATS.GATHER_TABLE_STATS
('SCOTT', 'EMP',
ESTIMATE_PERCENT=>DBMS_STATS.AUTO_SAMPLE_SIZE,
BLOCK_SAMPLE=>FALSE,
DEGREE=>16,
GRANULARITY=>'PARTITION',
CASCADE=>TRUE,
OPTIONS=>'GATHER',
NO_INVALIDATE=>FALSE,
METHOD_OPT=>'FOR COLUMNS EMPNO SIZE REPEAT');

EXEC DBMS_STATS.LOCK_TABLE_STATS('SCOTT','EMP');
```

Como coletar? Coleta de exceção

- HINT /*+ GATHER_PLAN_STATISTICS */
- Parâmetro STATISTICS_LEVEL = ALL

```
SQL> SELECT COUNT(OBJECT_NAME) FROM T5 WHERE OBJECT_TYPE = 'TABLE';

COUNT(OBJECT_NAME)
------------------
             86656

SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(NULL, NULL, 'ALLSTATS LAST'));

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------
SQL_ID  aug9hpqkvv7vc, child number 0
-------------------------------------
SELECT COUNT(OBJECT_NAME) FROM T5 WHERE OBJECT_TYPE = 'TABLE'

Plan hash value: 748540854

-------------------------------------------------------------------------------------------------
| Id  | Operation         | Name        | Starts | E-Rows | A-Rows |   A-Time   | Buffers |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |             |      1 |        |      1 |00:00:00.02 |     208 |
|   1 |  SORT AGGREGATE   |             |      1 |      1 |      1 |00:00:00.02 |     208 |
|*  2 |   INDEX RANGE SCAN| IDX_T4_TYPE |      1 |   2327 |  86656 |00:00:00.01 |     208 |
-------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("OBJECT_TYPE"='TABLE')
```

Como coletar? Coleta de exceção

SQL: [+]

ID	Exec Ord	Operation	Go To	More	Cost[2]	Estim Card	LAST Starts	LAST Output Rows	LAST Over/Under Estimate[1]	Work Area
0	19	SELECT STATEMENT			2217	4	1	18	5x under	
1	18	SORT GROUP BY		[+]	2217	4	1	18	5x under	[+]
2	17	. GENERATE CUBE		[+]	2217	4	1	32	8x under	
3	16	.. SORT GROUP BY		[+]	2217	4	1	4	1x	[+]
4	15	... HASH JOIN		[+]	2216	1067	1	1014	1x	[+]
5	1	.... TABLE ACCESS FULL COUNTRIES	[+]	[+]	2	2	1	2	1x	
6	14	.... HASH JOIN		[+]	2214	11736	1	21964	2x under	[+]
7	12	....+ ~~NESTED LOOPS~~		[+]	2214	11736	1	1	*** 11736x over	
8	10	....+. ~~NESTED LOOPS~~		[+]	1694		1	1		
9	8	....+.. ~~STATISTICS COLLECTOR~~		[+]	1694		1	21964		
10	7	....+... HASH JOIN		[+]	1694	11736	1	21964	2x under	[+]
11	5	....+.... MERGE JOIN CARTESIAN		[+]	4	28	1	28	1x	
12	2	....+....+ TABLE ACCESS FULL TIMES	[+]	[+]	2	14	1	14	1x	
13	4	....+....+ BUFFER SORT		[+]	2	2	14	28	1x	[+]
14	3	....+....+. TABLE ACCESS FULL CHANNELS	[+]	[+]	0	2	1	2	1x	
15	6	....+.... TABLE ACCESS FULL SALES	[+]	[+]	1593	14728759	1	14728759	1x	
16	9	....+.. ~~INDEX UNIQUE SCAN CUSTOMERS_PK~~	[+]	[+]			0	0		
17	11	....+. ~~TABLE ACCESS BY INDEX ROWID CUSTOMERS~~	[+]	[+]	510	1	0	0		
18	13	....+ TABLE ACCESS FULL CUSTOMERS	[+]	[+]	510	1626064	1	1626064	1x	

Performance statistics are only available when parameter "statistics_level" was set to "ALL" at hard-parse time, or SQL contains "gather_plan_statistics" hint.
Last Starts and Last Output Rows were captured by SQL Plan Monitor. SQL Exec Start is "2016-07-25/16:52:04". SQL Exec ID is "16777216". Status is "DONE (ALL ROWS)". Key is "433791697552".
*(1) If estim_card * starts < output_rows then under-estimate. If estim_card * starts > output_rows then over-estimate. Color highlights when exceeding * 10x, ** 100x and *** 1000x over/under-estimates.*
(2) Largest contributors for cumulative-statistics columns are shown in red.

SQLT (MOS 215187.1)
oratop (MOS 1500864.1)

```
Oracle 12c - CAT 17:18:38 up: 1.3d,    1 ins,    2 sn,    1 us, 1.4G mt,  4.6% db
ID %CPU LOAD %DCU   AAS  ASC  ASI  ASW  AST IOPS %FR   PGA UTPS UCPS SSRT   %DBT
 1   10    0    3     4    1    0    0    1   20   8  373M    0    6   1m    100

EVENT (C)                                TOT WAITS   TIME(s)  AVG_MS  PCT    WAIT_CLASS
DB CPU                                                   746              58
Datapump dump file I/O                        5704       279    49.0   22      User I/O
db file sequential read                      46252       167     3.6   13      User I/O
direct path write                              714        58    82.6    5      User I/O
direct path read temp                          367        35    97.1    3      User I/O

ID   SID     SPID USR PROG S   PGA SQLID/BLOCKER OPN   E/T STA STE EVENT/*LA   W/T

Enter sql_id: dyrgn2atcaukd

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
SQL_ID: dyrgn2atcaukd, child number 0
-------------------------------------------------
SELECT T1.ID, T2.NAME FROM T1, T2 WHERE T1.ID = T2.ID

Plan hash value: 2691992084

-------------------------------------------------------------------------------
|   Id | Operation                      | Name       | Rows | Cost | Stale |
-------------------------------------------------------------------------------
|    0 | SELECT STATEMENT               |            |      |   68 |       |
|    1 |  HASH JOIN                     |            | 1.0M |   68 |       |
|    2 |   INDEX FAST FULL SCAN         | T1_IDX1    |  10k |    7 | YES   |
|    3 |   TABLE ACCESS FULL            | T2         |  10k |   43 | YES   |
-------------------------------------------------------------------------------
```

Outras estatísticas

Fixed Objects Statistics (VSQL, VSESSION, etc.)

```
SQL> EXEC DBMS_STATS.GATHER_FIXED_OBJECTS_STATS;
```

Dictionary Statistics (DBA_SEGMENTS, DBA_TABLES, etc.);

```
SQL> EXEC DBMS_STATS.GATHER_DICTIONARY_STATS;
```

System Statistcs (CPU e I/O)

```
SQL> EXEC DBMS_STATS.GATHER_SYSTEM_STATS;
```

OU

```
SQL> EXEC DBMS_STATS.GATHER_SYSTEM_STATS('START');
...
SQL> EXEC DBMS_STATS.GATHER_SYSTEM_STATS('STOP');
```

OU

```
SQL> EXEC DBMS_STATS.GATHER_SYSTEM_STATS('EXADATA');
```

Lab 10.3: System Statistics

Verifique e guarde os planos de execução do capítulo de JOINs.

Colete as estatísticas de sistema durante uma carga, e verifique sua alteração.

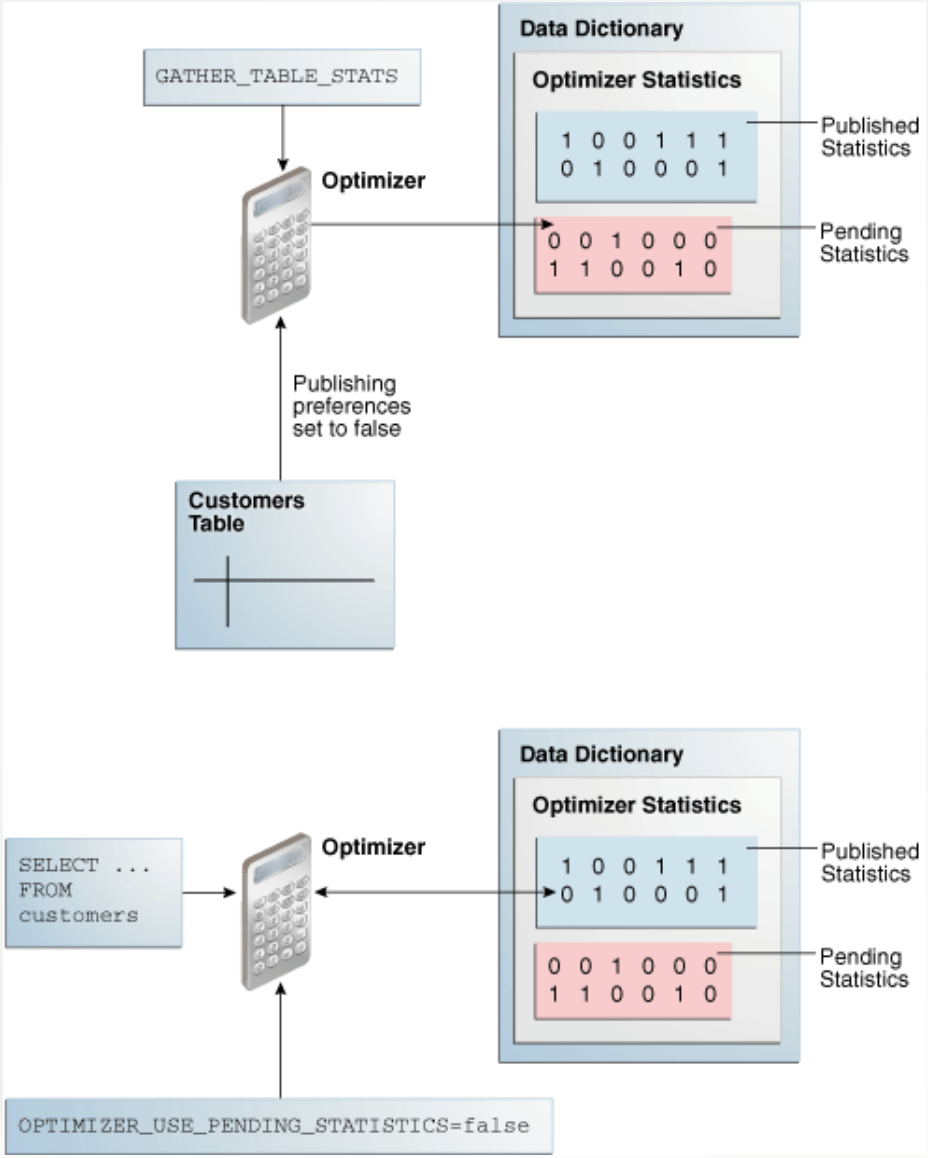
```
SQL> SELECT * FROM SYS.AUX_STATS$;
SQL> EXEC DBMS_STATS.GATHER_SYSTEM_STATS('START');

$ unzip -q swingbench261046.zip
$ cd /home/oracle/swingbench/bin/
$ ./charbench -uc 10 -cs //nerv01/PROD -c ../configs/SOE_Server_Side_V2.xml
...

SQL> SELECT * FROM SYS.AUX_STATS$;
SQL> EXEC DBMS_STATS.GATHER_SYSTEM_STATS('STOP');
SQL> SELECT * FROM SYS.AUX_STATS$;
```

Extended Statistics – Column Groups e Expression

```
SQL> EXEC DBMS_STATS.GATHER_TABLE_STATS( 'SH','CUSTOMERS',
    METHOD_OPT=>'FOR ALL COLUMNS SIZE SKEWONLY ' || 'FOR COLUMNS SIZE
SKEWONLY (CUST_STATE_PROVINCE,COUNTRY_ID)');

SQL> EXEC DBMS_STATS.GATHER_TABLE_STATS('SH','CUSTOMERS'
    ,METHOD_OPT => 'FOR ALL COLUMNS SIZE SKEWONLY FOR COLUMNS
     (LOWER(CUST_STATE_PROVINCE)) SIZE SKEWONLY');
```


245

Estatísticas Pendentes



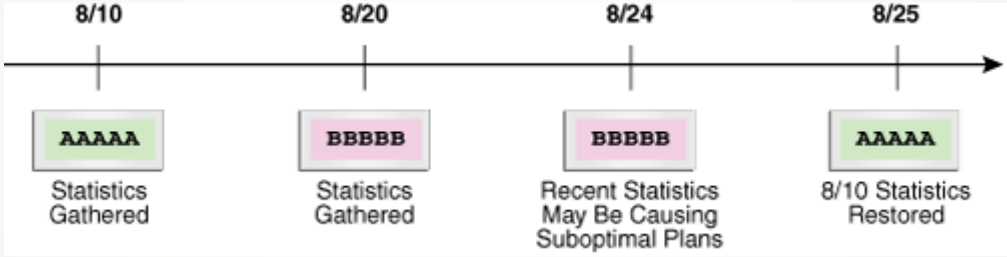
Lab 10.4: Estatísticas Pendentes

Colete estatísticas, e verifique-as antes de publica-las.

```
SQL> CONN SH/SH
SQL> EXEC DBMS_STATS.SET_TABLE_PREFS
      ('SH','CUSTOMERS','PUBLISH','FALSE');
SQL> SELECT * FROM USER_TAB_PENDING_STATS;
SQL> EXEC  DBMS_STATS.GATHER_TABLE_STATS('SH','CUSTOMERS');
SQL> SELECT * FROM USER_TAB_PENDING_STATS;
SQL> EXEC DBMS_STATS.PUBLISH_PENDING_STATS('SH','CUSTOMERS');
SQL> SELECT * FROM USER_TAB_PENDING_STATS;

SQL> EXEC DBMS_STATS.SET_TABLE_PREFS ('SH','SALES','PUBLISH','FALSE');
SQL> SELECT * FROM USER_TAB_PENDING_STATS;
SQL> EXEC DBMS_STATS.GATHER_TABLE_STATS('SH','SALES');
SQL> SELECT * FROM USER_TAB_PENDING_STATS;
SQL> EXEC DBMS_STATS.DELETE_PENDING_STATS('SH','SALES');
SQL> SELECT * FROM USER_TAB_PENDING_STATS;
```


optimizer_use_pending_statistics

Restore de Estatísticas

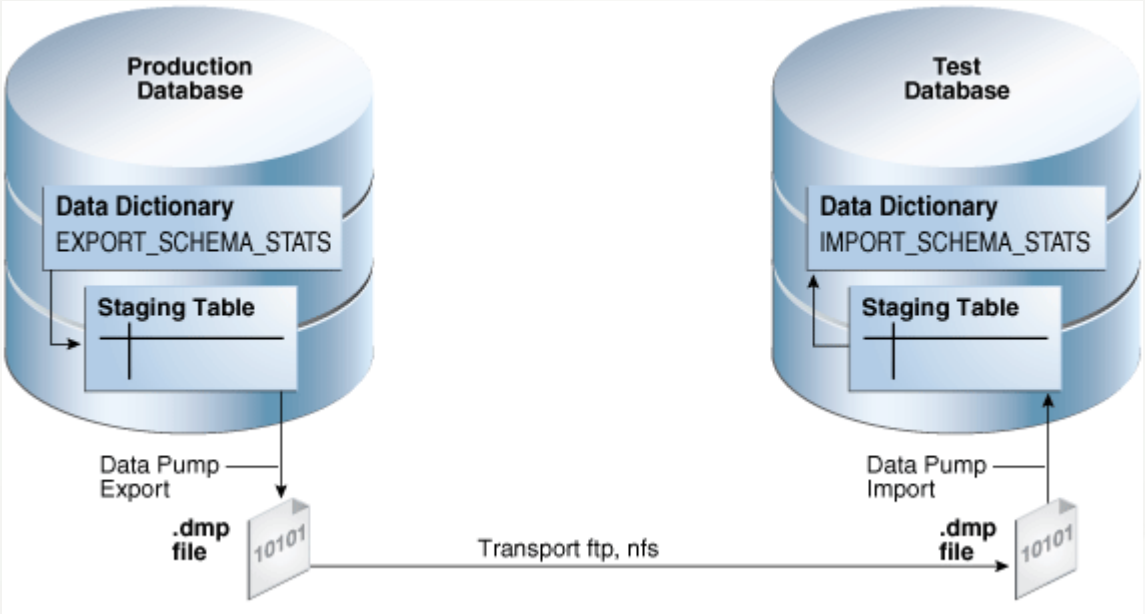


Lab 10.5: Restore de Estatísticas

Execute o Restore de uma estatística anterior.

```
SQL> CONN SCOTT/TIGER
SQL> COL TABLE_NAME FORMAT A10
SQL> SELECT
        TABLE_NAME,
            TO_CHAR(STATS_UPDATE_TIME,'YYYY-MM-DD:HH24:MI:SS') AS STATS_MOD_TIME
        FROM   DBA_TAB_STATS_HISTORY
        WHERE  TABLE_NAME='T92'
        AND OWNER='SCOTT'
        ORDER BY STATS_UPDATE_TIME DESC;
SQL> EXEC DBMS_STATS.GATHER_TABLE_STATS('SCOTT', 'T92');

SQL> EXEC DBMS_STATS.RESTORE_TABLE_STATS( 'SCOTT','T92',TO_TIMESTAMP('2018-03-
03:16:11:14','YYYY-MM-DD:HH24:MI:SS'), NO_INVALIDATE=>FALSE);

EXEC DBMS_STATS.RESTORE_SCHEMA_STATS('SCOTT',TO_TIMESTAMP('2018-03-
03:16:11:14','YYYY-MM-DD:HH24:MI:SS') );
```

Transporte de Estatísticas



Transporte de Estatísticas

Origem

```
EXEC DBMS_STATS.CREATE_STAT_TABLE (ownname => 'HR', stattab =>
TESTE_STATS'));

EXEC DBMS_STATS.GATHER_SCHEMA_STATS ('HR');
BEGIN DBMS_STATS.EXPORT_SCHEMA_STATS (ownname => 'HR', stattab =>
'TESTE_STATS';

$ expdp HR/HR DIRECTORY=dump_directory DUMPFILE=TESTE_STATS.dmp
TABLES=TESTE_STATS
```

Destino:

```
$ impdp HR/HR DIRECTORY=dump_directory DUMPFILE=TESTE_STATS.dmp
TABLES=TESTE_STATS

SQL> EXEC DBMS_STATS.IMPORT_SCHEMA_STATS(ownname => 'HR', stattab =>
'TESTE_STATS');
```

Otimizações

SQL ou PL/SQL?

"Se você pode fazer algo em SQL, faça-o em SQL.
Se você não pode faze-lo em SQL, faça em PL/SQL.
Se você não pode faze-lo em PL/SQL, faça em Java.
Se você não pode faze-lo em Java, faça em C++.
Se você não pode fazer em C++, não o faça."

Thomas Kyte

- O que você aprendeu sobre carros, quando aprendeu a dirigir?
- Quanto tempo você levou para aprender SQL?
- Utilizamos apenas cerca de 20% das capacidades de um programa ou linguagem.

Rows x Sets

Rows x Sets

Isto não representa uma tabela!

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	7369	SMITH	CLERK	7902	17-DEC-80	800	(null)	20
2	7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
3	7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
4	7566	JONES	MANAGER	7839	02-APR-81	2975	(null)	20
5	7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
6	7698	BLAKE	MANAGER	7839	01-MAY-81	2850	(null)	30
7	7782	CLARK	MANAGER	7839	09-JUN-81	2450	(null)	10
8	7788	SCOTT	ANALYST	7566	19-APR-87	3000	(null)	20
9	7839	KING	PRESIDENT	(null)	17-NOV-81	5000	(null)	10
10	7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
11	7876	ADAMS	CLERK	7788	23-MAY-87	1100	(null)	20
12	7900	JAMES	CLERK	7698	03-DEC-81	950	(null)	30
13	7902	FORD	ANALYST	7566	03-DEC-81	3000	(null)	20
14	7934	MILLER	CLERK	7782	23-JAN-82	1300	(null)	10

Rows x Sets

Quais são os funcionários que ganham mais do que a média de seu departamento.

Método procedural:

- Calcule a média salarial de cada departamento;
- Armazene a média de forma temporária;
- Liste todos os empregados e seu departamento;
- Compare o salário de cada empregado com a média de seu departamento;
- Liste todos os empregados que tem o salário acima da média.

Método por conjunto:

```
SELECT * FROM EMP EMP1
WHERE SAL > (SELECT AVG(EMP2.SAL)
             FROM EMP EMP2
             WHERE EMP2.DEPTNO = EMP1.DEPTNO);
```

Rows x Sets

Quais os empregados que passaram a mesma quantidade de anos em cada cargo ocupado?

Método procedural:

- Listar todos os empregados;
- Listas todos os cargos, data inicial e final destes empregados, calculando a quantidade de anos;
- Armazenar estes dados em formato temporário;
- Para cada empregado, verificar se todos as quantidades de anos são iguais;
- Descartar os empregados que não possuem todas as quantidade de anos iguais.

Método por conjunto:

```
SELECT EMPLOYEE_ID
    FROM JOB_HISTORY
    GROUP BY EMPLOYEE_ID
        HAVING
            MIN(ROUND(MONTHS_BETWEEN(START_DATE, END_DATE) / 12,0)) =
            MAX(ROUND(MONTHS_BETWEEN(START_DATE, END_DATE) / 12,0)
```

Rows x Sets

Qual a quantidade média de dias entre os pedidos de um cliente?

Método procedural:

- Liste todos os pedidos do cliente X;
- Selecione os pedidos e suas datas;
- Para cada pedido, selecione a data do pedido anterior;
- Calcule qual a quantidade de dias entre a data do pedido e a data anterior;
- Calcule a média destas quantidades.

Método por conjunto:

```
SELECT
    (MAX(TRUNC(ORDER_DATE)) - MIN(TRUNC(ORDER_DATE))) / COUNT(ORDER_ID)
    FROM ORDERS
    WHERE CUSTOMER_ID = 102 ;
```

Rows x Sets

Crie uma tabela com os clientes pequenos, outra com os clientes médios, e outra com os clientes grandes.

Método procedural:

- Selecionar os clientes que compraram menos de 10000;
- Inseri-los na tabela SMALL_CUSTOMERS;
- Selecionar os clientes que compraram entre 10000 e 99999;
- Inseri-los na tabela MEDIUM_CUSTOMERS;
- Selecionar os clientes que compraram mais de 100000;
- Inseri-los na tabela LARGE_CUSTOMERS;

Método por conjunto:

```
INSERT ALL
    WHEN SUM_ORDERS < 10000 THEN INTO SMALL_CUSTOMERS
    WHEN SUM_ORDERS >= 10000 AND SUM_ORDERS < 100000 THEN INTO
MEDIUM_CUSTOMERS
    ELSE INTO LARGE_CUSTOMERS
    SELECT CUSTOMER_ID, SUM(ORDER_TOTAL) SUM_ORDERS
        FROM OE.ORDERS
        GROUP BY CUSTOMER_ID;
```

Rows x Sets

Altere o bônus para 20% de quem é candidato mas ainda não tem bônus, remova de quem ganha mais de 7.500, e marque como 10% o bônus de quem ainda não é candidato mas recebe menos que 7.500.

Método procedural:

- Selecione os empregados que devem receber a alteração de 20%;
- Faça a alteração dos empregados que devem receber a alteração de 20%;
- Selecione os empregados que devem receber a alteração de 10%;
- Faça a alteração dos empregados que devem receber a alteração de 10%;
- Selecione os empregados que não devem mais ser candidatos a bônus;
- Remova os empregados que não devem mais ser candidatos a bônus.

Método por conjunto:

```
MERGE INTO BONUS_DEPT60 B
    USING (SELECT EMPLOYEE_ID, SALARY, DEPARTMENT_ID
        FROM EMPLOYEES WHERE DEPARTMENT_ID = 60) E
    ON (B.EMPLOYEE_ID = E.EMPLOYEE_ID)
        WHEN MATCHED THEN
            UPDATE SET BONUS = E.SALARY * 0.2 WHERE B.BONUS = 0
            DELETE WHERE (E.SALARY > 7500)
        WHEN NOT MATCHED THEN
        INSERT (B.EMPLOYEE_ID, B.BONUS)
            VALUES (E.EMPLOYEE_ID, E.SALARY * 0.1)
            WHERE (E.SALARY < 7500);
```

Otimizações Gerais

- Menos é mais.
- A qualidade do trabalho do CBO é inversamente proporcional ao tamanho do SQL.
- O CBO pode ser beneficiado por CONSTRAINTs em SELECTs (Query Transformation).
- Evite funções na cláusula WHERE.
- Evite DISTINCT, se possível.
- Evite ORDER BY, se possível.
- Prefira UNION ALL a UNION, se possível.
- SELECT COL1, COL2 … ao invés de SELECT * … irá economizar SQL*Net e PGA.
- Procure reduzir o tamanho do SQL (Shared Pool).
- Prefira SQL, se lhe atender.
- Stored Procedures podem reduzir tráfego de rede.
- Stored Procedures forçam o uso de BINDs.
- Evite trocar entre PL/SQL e SQL (context switch).
- Evite acesso desnecessário a DUAL (operações aritméticas, SEQUENCEs).
- Em PL/SQL, evite Row-by-Row.
- Em PL/SQL, evite Nested Row-by-Row.
- Utilize variáveis BIND em SQLs repetitivos.
- Variáveis BINDs impedem SQL Injection.
- Não utilize Binds se o tempo total de compilação não for relevante quanto ao tempo total de resposta.
- Utilize CACHE em Sequences, se possível (em RAC, adicione NOORDER, se possível).

Otimizações Gerais

- A maneira mais rápida de se fazer um grande UPDATE / DELETE é não fazer. Faça CTAS.
- Em alterações em lote, desabilite as CONSTRAINTs, se possível.
- Em alterações em lote, remova os índices, se possível.
- Em alterações em lote, considere o PARALLELISM.
- Em INSERT em lote, considere o uso APPEND (Direct Path).
- TRUNCATE não gera REDO e UNDO, não dispara Triggers, e abaixa a HWM.
- Conheça BULK BINDING (LOOP com FORALL).
- Conheça INSERT com BULK COLLECT.
- Conheça INSERT com Pipelined Table Function
- Estude o SQL*Loader / External Tables.
- Troque UTL_FILE por EXTERNAL TABLES.
- Em geral, Segurança (Auditoria, Criptografia, etc.) prejudica Desempenho.
- Evite WHERE CAMPO LIKE 'A%' em colunas de baixa seletividade.
- Evite mais ainda WHERE CAMPO LIKE '%A%'.
- Se precisar utilizar LIKE '%A%', implemente Oracle Text.
- Estude a função MERGE.
- Estude a Cláusula WITH.
- Estude as Funções Analíticas.
- Estude Queries Hierárquicas (CONNECT BY).
- Considere Materialized Views.
- Evite o abuso de VIEWs, TRIGGERs, DBLINKs.

Lab 11.1: Append e Redo

Crie uma tabela de testes, e execute uma gravação, com parâmetros diferentes.

```
SQL> CONN SCOTT/TIGER
SQL> CREATE TABLE T111 AS SELECT * FROM ALL_OBJECTS WHERE 1=0;

SQL> SET AUTOTRACE TRACEONLY STATISTICS
SQL> INSERT INTO T111 SELECT * FROM ALL_OBJECTS;
SQL> INSERT INTO T111 SELECT * FROM ALL_OBJECTS;
SQL> INSERT INTO T111 SELECT * FROM ALL_OBJECTS;

SQL> INSERT /*+ APPEND */ INTO T111 SELECT * FROM ALL_OBJECTS;
SQL> INSERT /*+ APPEND */ INTO T111 SELECT * FROM ALL_OBJECTS;
```

O que aconteceu?

```
SQL> INSERT /*+ APPEND */ INTO T111 SELECT * FROM ALL_OBJECTS;

SQL> ALTER TABLE T111 NOLOGGING;
SQL> INSERT /*+ APPEND */ INTO T111 SELECT * FROM ALL_OBJECTS;
SQL> INSERT /*+ APPEND */ INTO T111 SELECT * FROM ALL_OBJECTS;
SQL> INSERT /*+ APPEND */ INTO T111 SELECT * FROM ALL_OBJECTS;
```

Qual a diferença exibida pelo SET AUTOTRACE TRACEONLY STATISTICS?

Paralelismo

Permite Query, DML e DDL.

Quantos processos de paralelismo utilizar?

Um objeto pode ter Parallel permanente, independente do SQL:

```
SQL> ALTER TABLE SCOTT.T PARALLEL 4;

SQL> ALTER SESSION FORCE PARALLEL QUERY PARALLEL 5;
SQL> ALTER SESSION FORCE PARALLEL DML PARALLEL 5;
SQL> ALTER SESSION FORCE PARALLEL DDL PARALLEL 5;
```

O Parallel SQL pode ser utilizado diretamente no SQL:

```
SQL> SELECT /*+ PARALLEL(T2 4) */ COUNT(*) FROM T2;
SQL> SELECT /*+ PARALLEL(T2 4,2) */ COUNT(*) FROM T2;
```

Paralelismo

Parâmetros:
PARALLEL_MIN_SERVERS = Número entre 0 e PARALLEL_MAX_SERVERS.
PARALLEL_MAX_SERVERS = De 0 a 3600.
PARALLEL_MIN_PERCENT = De 0 a 100.

PARALLEL_DEGREE_POLICY = MANUAL, LIMITED ou AUTO.
PARALLEL_MIN_TIME_THRESHOLD = AUTO | Segundos.
PARALLEL_ADAPTIVE_MULTI_USER = true ou false.
PARALLEL_DEGREE_LIMIT = CPU, IO ou Número.
PARALLEL_SERVERS_TARGET = Número entre 0 e PARALLEL_MAX_SERVERS.
PARALLEL_THREADS_PER_CPU = Qualquer número.

PARALLEL_EXECUTION_MESSAGE_SIZE = De 2148 a 32768
PARALLEL_FORCE_LOCAL = true ou **false**.
PARALLEL_INSTANCE_GROUP = Oracle RAC service_name ou group_name.

PARALLEL_AUTOMATIC_TUNING: Deprecated.
PARALLEL_IO_CAP_ENABLED = Deprecated.

Paralelismo

```
SQL> SELECT SID, SERIAL#, QCSID, QCSERIAL# FROM V$PX_SESSION;

       SID    SERIAL#      QCSID  QCSERIAL#
---------- ---------- ---------- ----------
       202       5249         12        387
        20       3587         12        387
        75       4043         12        387
       141        233         12        387
       204        751         12        387
        16        229         12        387
        73       3279         12        387
       137        403         12        387
       203       1137         12        387
        18        103         12        387
        79          5         12        387
       134       3431         12        387
       206          5         12        387
        19          5         12        387
        76         31         12        387
       140          5         12        387
        12        387         12
```

Lab 11.2: Paralelismo

Abra a sessão com o SCOTT com SET TIMING ON.
Em seguida, faça o teste do PARALLEL.

```
SQL> CONN SCOTT/TIGER
SQL> CREATE TABLE T112 AS SELECT * FROM ALL_OBJECTS;
7 x SQL> INSERT INTO T112 SELECT * FROM T112;
SQL> COMMIT;
```

Repita a operação com PARALLEL, e compare.

```
SQL> SET TIMING ON
SQL> SELECT COUNT(*) FROM T112;
SQL> SELECT /*+ PARALLEL(T112 4) */ COUNT(*) FROM T112;
SQL> SELECT /*+ PARALLEL(T112 20) */ COUNT(*) FROM T112;
SQL> SELECT /*+ PARALLEL(T112 40) */ COUNT(*) FROM T112;
```

Qual a diferença exibida pelo SET TIMING ON?

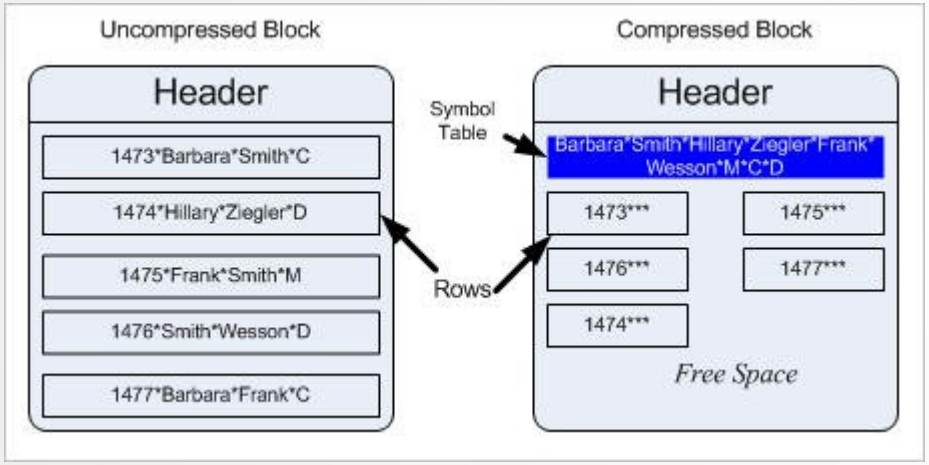
Lab 11.3: RESULT_CACHE

Execute o teste do RESULT_CACHE.

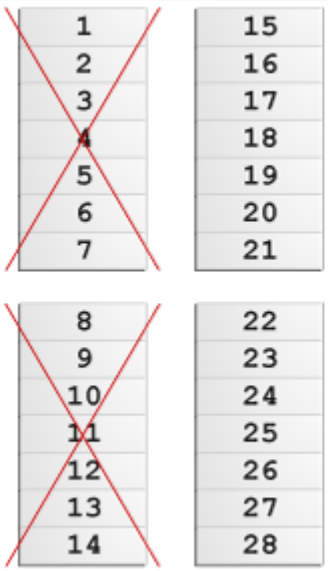
```
SQL> CONN SCOTT/TIGER

SQL> SET TIMING ON
SQL> SET AUTOTRACE ON
SQL> SELECT COUNT(*) FROM T112;
SQL> SELECT COUNT(*) FROM T112;
SQL> SELECT COUNT(*) FROM T112;
SQL> SELECT /*+ RESULT_CACHE */ COUNT(*) FROM T112;
SQL> SELECT /*+ RESULT_CACHE */ COUNT(*) FROM T112;
SQL> SELECT /*+ RESULT_CACHE */ COUNT(*) FROM T112;

SQL> DELETE FROM T112 WHERE ROWNUM = 1;
SQL> SELECT /*+ RESULT_CACHE */ COUNT(*) FROM T112;
SQL> SELECT /*+ RESULT_CACHE */ COUNT(*) FROM T112;
SQL> SELECT /*+ RESULT_CACHE */ COUNT(*) FROM T112;
```

Compression

- 9i OLAP
- 11g OLTP
- 12c InMemory




```
SELECT cust_id, time_id, channel_id
FROM   sales
WHERE  prod_id > 14
AND    prod_id < 29
```



Storage Index

```
ext 1:   min 1, max 7
ext 2:   min 8, max 14
ext 3:   min 15, max 21
ext 4:   min 22, max 28
```

Lab 11.4: Compression

Execute o teste do COMPRESSION.

```
SQL> SET TIMING ON
SQL> SELECT COUNT(*) FROM T112;
SQL> SELECT COUNT(*) FROM T112;
SQL> SELECT COUNT(*) FROM T112;

SQL> SELECT TO_CHAR(SUM(BYTES)) FROM DBA_SEGMENTS WHERE SEGMENT_NAME
= 'T112';
SQL> ALTER TABLE T112 COMPRESS;
SQL> ALTER TABLE T112 MOVE;
SQL> SELECT TO_CHAR(SUM(BYTES)) FROM DBA_SEGMENTS WHERE SEGMENT_NAME
= 'T112';

SQL> SELECT COUNT(*) FROM T112;
SQL> SELECT COUNT(*) FROM T112;
SQL> SELECT COUNT(*) FROM T112;
```

Lab 11.5: CTAS

Execute o teste de DELETE x CTAS (CREATE TABLE AS SELECT).

```
SQL> SELECT COUNT(*) FROM T113;
SQL> SELECT COUNT(*) FROM T113 WHERE OBJECT_TYPE = 'SYNONYM';

SQL> SET TIMING ON
SQL> DELETE FROM T113 WHERE OBJECT_TYPE = 'SYNONYM';
SQL> ROLLBACK;

SQL> CREATE TABLE T113TEMP AS SELECT * FROM T113 WHERE
OBJECT_TYPE != 'SYNONYM';
SQL> DROP TABLE T113;
SQL> ALTER TABLE T113TEMP RENAME TO T113;
```

Extração de Planos de Execução

Fontes de Planos de Execução

- PLAN_TABLE, carregada por EXPLAIN PLAN / DBMS_XPLAN.DISPLAY ou AUTOTRACE (e SQL Developer, Toad, etc.);
- VIEWs de planos compilados e armazenados na Library Cache;
- Tabelas de AWR / STATSPACK;
- Arquivos Trace (10046, 10053, etc.).

Explain Plan

Sintaxe:

```
EXPLAIN PLAN [SET STATEMENT_ID=id] [INTO table] FOR statement;
```

Exemplo:

```
SQL> EXPLAIN PLAN SET STATEMENT_ID='TESTE1' FOR
       SELECT ENAME FROM EMP;
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);
```

Limitações do Explain Plan:

- Não executa o SQL (AUTOTRACE TRACEONLY EXPLAIN também não executa).
- É o Plano, não a Execução;
- Não utiliza Bind Peeking / Adaptive Cursor Sharing (ACS);
- Todas Variáveis Bind são consideradas VARCHAR2;
- Depende do ambiente de execução (parâmetros, trigger de logon?).

Recuperação de Ambiente de Execução:

```
SQL> SELECT * FROM
    TABLE(DBMS_XPLAN.DISPLAY_CURSOR(NULL,NULL,'OUTLINE PEEKED_BINDS'));

SQL> SELECT NAME, VALUE_STRING, VALUE_ANYDATA FROM V$SQL_BIND_CAPTURE
WHERE SQL_ID = 'gsd1tngqwc6q9';

SQL> SELECT NAME, VALUE FROM V$SQL_OPTIMIZER_ENV
    WHERE SQL_ID = '3whrvcrw7p78q' AND CHILD_NUMBER = 0
    AND ISDEFAULT = 'NO';
```

Lab 12.1: Explain Plan

Execute os SQLs abaixo, comparando os Planos de Execução.

```
SQL> CONN SCOTT/TIGER
SQL> CREATE TABLE T121 AS SELECT * FROM ALL_OBJECTS;
SQL> CREATE INDEX IDX_T121_OBJECT_ID ON T121(OBJECT_ID);

SQL> SET AUTOTRACE ON EXPLAIN
SQL> SELECT COUNT(OBJECT_ID) FROM T121 WHERE OBJECT_ID = 1;
SQL> SELECT COUNT(OBJECT_ID) FROM T121 WHERE OBJECT_ID = '1';
SQL> VARIABLE VOBJECT_ID NUMBER;
SQL> EXEC :VOBJECT_ID := 1;
SQL> SELECT COUNT(OBJECT_ID) FROM T121 WHERE OBJECT_ID = :VOBJECT_ID;
```

Lab 12.2: Explain Plan

Execute o SQL abaixo duas vezes, comparando o Plano de Execução.

```
$ sqlplus SCOTT/TIGER
SQL> CREATE TABLE T1122 AS SELECT * FROM ALL_OBJECTS;
SQL> CREATE INDEX T1122_IDX1 ON T1122(OBJECT_ID);
SQL> CREATE TABLE T2122 AS SELECT * FROM ALL_OBJECTS;
SQL> CREATE INDEX T2122_IDX1 ON T2122(OBJECT_ID);
SQL> EXEC DBMS_STATS.GATHER_TABLE_STATS('SCOTT', 'T1122');
SQL> EXEC DBMS_STATS.GATHER_TABLE_STATS('SCOTT', 'T2122');

SQL> SET AUTOTRACE TRACEONLY EXPLAIN

SQL> SELECT T1122.OBJECT_ID, T2122.OBJECT_NAME FROM T1122, T2122 WHERE
T1122.OBJECT_ID = T2122.OBJECT_ID AND T2122.OBJECT_TYPE = 'SYNONYM' AND
T1122.OBJECT_ID BETWEEN 1000 and 10000;

SQL> VARIABLE vSTART NUMBER
SQL> VARIABLE vEND NUMBER
SQL> EXEC :vSTART := 1000
SQL> EXEC :vEND := 10000
SQL> SELECT T1122.OBJECT_ID, T2122.OBJECT_NAME FROM T1122, T2122 WHERE
T1122.OBJECT_ID = T2122.OBJECT_ID AND T2122.OBJECT_TYPE = 'SYNONYM' AND
T1122.OBJECT_ID BETWEEN :vSTART and :vEND;
```

AWR / Statspack

SQL Statement

```
@$ORACLE_HOME/rdbms/admin/awrsqrpt.sql
Enter value for report_type: html
Enter value for num_days: 1
Enter value for begin_snap: 40
Enter value for end_snap: 41
Enter value for sql_id: 062savj8zgzut
Enter value for report_name: AWR_SQL_01.html
```

SQL Statement (Specific Database Instance)

```
@$ORACLE_HOME/rdbms/admin/awrsqrpi.sql
```

STATSPACK

```
@$ORACLE_HOME/rdbms/admin/sprepsql.sql
```


277

AWR / Statspack

Tabelas:

- STATS$SQL_PLAN (STATSPACK)
- DBA_HIST_SQL_PLAN
- DBA_HIST_SQLTEXT
- DBA_HIST_SQLSTAT
- DBA_HIST_SQLBIND

Exemplo:

```
SQL> SELECT SQL_ID, SQL_TEXT FROM DBA_HIST_SQLTEXT
       WHERE SQL_TEXT LIKE '%SELECT ENAME FROM EMP%';
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_AWR('a10jnjwd22gs8'));

SQL> EXEC DBMS_WORKLOAD_REPOSITORY.CREATE_SNAPSHOT;
SQL> EXEC DBMS_WORKLOAD_REPOSITORY.MODIFY_SNAPSHOT_SETTINGS
(RETENTION => 43200, INTERVAL => 30, TOPNSQL => 100);

SQL> EXEC DBMS_WORKLOAD_REPOSITORY.ADD_COLORED_SQL('az9p3ctumhpr8');
SQL> EXEC DBMS_WORKLOAD_REPOSITORY.REMOVE_COLORED_SQL('az9p3ctumhpr8');
```

AWR requer Licença Diagnostics Pack

Views

Dynamic Performance Views:

- V$SQL_PLAN
- V$SQL_PLAN_STATISTICS
- V$SQL_WORKAREA
- V$SQL_PLAN_STATISTICS_ALL (V$SQL_PLAN_STATISTICS + V$SQL_WORKAREA)

Chave < 10g: ADDRESS, HASH_VALUE, CHILD_NUMBER
Chave >= 10g: SQL_ID

DISPLAY_CURSOR

```
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(SQLID, CHILD_NUMBER,
FORMAT));
```

FORMAT

- TYPICAL = DEFAULT
- ALL = TYPICAL + QB + PROJECTION + ALIAS + REMOTE
- ADVANCED = ALL + OUTLINE + BINDS
- ALLSTATS = IOSTATS + MEMSTATS (all executions)
- ALLSTATS LAST (last execution)
- ADAPTIVE (12c)

Execution Plan

```
SQL> CONN SCOTT/TIGER
SQL> EXEC DBMS_STATS.DELETE_TABLE_STATS('SCOTT','T91');
SQL> SELECT /*+ GATHER_PLAN_STATISTICS */ COUNT(OBJECT_NAME) FROM T91 WHERE
OBJECT_TYPE = 'SYNONYM';
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(NULL, NULL, 'ALLSTATS LAST'));

SQL> SELECT /*+ GATHER_PLAN_STATISTICS */ COUNT(OBJECT_NAME) FROM T91 WHERE
OBJECT_TYPE = 'PACKAGE';
SQL> SELECT /*+ GATHER_PLAN_STATISTICS */ COUNT(OBJECT_NAME) FROM T91 WHERE
OBJECT_TYPE = 'DIRECTORY';
```

```
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(NULL, NULL, 'ALLSTATS LAST'));

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------
SQL_ID  g51k4ccp1a6mg, child number 0
-------------------------------------
SELECT /*+ GATHER_PLAN_STATISTICS */ EMPNO, ENAME FROM SCOTT.EMP WHERE
ENAME = 'KING'

Plan hash value: 3956160932

-----------------------------------------------------------------------------------------
| Id  | Operation         | Name | Starts | E-Rows | A-Rows |   A-Time   | Buffers |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |      |      1 |        |      1 |00:00:00.01 |       8 |
|*  1 |  TABLE ACCESS FULL| EMP  |      1 |      1 |      1 |00:00:00.01 |       8 |
-----------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("ENAME"='KING')
```

Execution Plan

```
SQL> CONN SCOTT/TIGER
SQL> EXEC DBMS_STATS.DELETE_SCHEMA_STATS(USER);

SQL> SELECT EMPNO, ENAME, DNAME, LOC FROM EMP, DEPT WHERE EMP.DEPTNO = DEPT.DEPTNO
AND DEPT.DEPTNO IN (10);
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(NULL, NULL, 'ALLSTATS LAST'));

SQL> SELECT EMPNO, ENAME, DNAME, LOC FROM EMP, DEPT WHERE EMP.DEPTNO = DEPT.DEPTNO
AND DEPT.DEPTNO IN (10,20);
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(NULL, NULL, 'ALLSTATS LAST'));

SQL> SELECT EMPNO, ENAME, DNAME, LOC FROM EMP, DEPT WHERE EMP.DEPTNO = DEPT.DEPTNO;
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(NULL, NULL, 'ALLSTATS LAST'));

SQL> SELECT /*+ GATHER_PLAN_STATISTICS */ EMPNO, ENAME, DNAME, LOC FROM EMP, DEPT
WHERE EMP.DEPTNO = DEPT.DEPTNO AND DEPT.DEPTNO IN (10);
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(NULL, NULL, 'ALLSTATS LAST'));

SQL> SELECT /*+ GATHER_PLAN_STATISTICS */ EMPNO, ENAME, DNAME, LOC FROM EMP, DEPT
WHERE EMP.DEPTNO = DEPT.DEPTNO AND DEPT.DEPTNO IN (10,20);
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(NULL, NULL, 'ALLSTATS LAST'));

SQL> SELECT /*+ GATHER_PLAN_STATISTICS */ EMPNO, ENAME, DNAME, LOC FROM EMP, DEPT
WHERE EMP.DEPTNO = DEPT.DEPTNO;
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(NULL, NULL, 'ALLSTATS LAST'));
```


281

Execution Plan

```
SQL> ALTER SYSTEM SET statistics_level=ALL;

System altered.

SQL> SELECT EMPNO, ENAME FROM SCOTT.EMP WHERE ENAME = 'KING';

     EMPNO ENAME
---------- ----------
      7839 KING

SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(NULL, NULL, 'ALLSTATS LAST'));

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------
SQL_ID  8mr5jjp29fudw, child number 0
-------------------------------------
SELECT EMPNO, ENAME FROM SCOTT.EMP WHERE ENAME = 'KING'

Plan hash value: 3956160932

-----------------------------------------------------------------------------------------
| Id  | Operation         | Name | Starts | E-Rows | A-Rows |   A-Time   | Buffers |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |      |      1 |        |      1 |00:00:00.01 |       8 |
|*  1 |  TABLE ACCESS FULL| EMP  |      1 |      1 |      1 |00:00:00.01 |       8 |
-----------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("ENAME"='KING')
```


282

Execution Plan

```
SQL> CONN / AS SYSDBA
SQL> ALTER SYSTEM SET STATISTICS_LEVEL=ALL;

SQL> CONN SCOTT/TIGER
SQL> ALTER SESSION SET STATISTICS_LEVEL=ALL;
SQL> SELECT COUNT(OBJECT_NAME) FROM T91 WHERE OBJECT_TYPE = 'SYNONYM';
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(NULL, NULL, 'ALLSTATS LAST'));

SQL> SELECT COUNT(OBJECT_NAME) FROM T91 WHERE OBJECT_TYPE = 'PACKAGE';
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(NULL, NULL, 'ALLSTATS LAST'));

SQL> SELECT COUNT(OBJECT_NAME) FROM T91 WHERE OBJECT_TYPE = 'DIRECTORY';
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(NULL, NULL, 'ALLSTATS LAST'));
```


283

Execution Plan

```
SQL> CONN SCOTT/TIGER
SQL> COL SQL_TEXT FORMAT A150
SQL> SELECT SQL_ID, CHILD_NUMBER, SQL_TEXT FROM V$SQL WHERE SQL_TEXT LIKE '%T91%';
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR('fnjnr3c42v3fr', NULL, 'ALLSTATS
LAST'));
```

```
SQL> SELECT SQL_ID, SQL_TEXT FROM V$SQL WHERE SQL_TEXT LIKE 'SELECT COUNT(OBJECT_NAME) FROM T5%';

SQL_ID        SQL_TEXT
------------- ------------------------------------------------------------------------------------
aug9hpqkvv7vc SELECT COUNT(OBJECT_NAME) FROM T5 WHERE OBJECT_TYPE = 'TABLE'

SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR('aug9hpqkvv7vc', 0, 'ALLSTATS LAST'));

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------
SQL_ID  aug9hpqkvv7vc, child number 0
-------------------------------------
SELECT COUNT(OBJECT_NAME) FROM T5 WHERE OBJECT_TYPE = 'TABLE'

Plan hash value: 748540854

-----------------------------------------------------------------------------------------------
| Id  | Operation         | Name        | Starts | E-Rows | A-Rows |   A-Time   | Buffers |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |             |      1 |        |      1 |00:00:00.02 |     208 |
|   1 |  SORT AGGREGATE   |             |      1 |      1 |      1 |00:00:00.02 |     208 |
|*  2 |   INDEX RANGE SCAN| IDX_T4_TYPE |      1 |  45861 |  86656 |00:00:00.01 |     208 |
-----------------------------------------------------------------------------------------------
```


284

Execution Plan

```
SQL> SELECT COUNT(OBJECT_NAME) FROM T5 WHERE OBJECT_TYPE = 'TABLE';

COUNT(OBJECT_NAME)
------------------
             86656

SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(NULL, NULL, 'ALLSTATS LAST'));

PLAN_TABLE_OUTPUT
-----------------------------------------------------------------------------------------------------
SQL_ID  aug9hpqkvv7vc, child number 0
-------------------------------------
SELECT COUNT(OBJECT_NAME) FROM T5 WHERE OBJECT_TYPE = 'TABLE'

Plan hash value: 748540854

-----------------------------------------------------------------------------------------------------
| Id  | Operation         | Name        | Starts | E-Rows | A-Rows |   A-Time   | Buffers |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |             |      1 |        |      1 |00:00:00.02 |     208 |
|   1 |  SORT AGGREGATE   |             |      1 |      1 |      1 |00:00:00.02 |     208 |
|*  2 |   INDEX RANGE SCAN| IDX_T4_TYPE |      1 |   2327 |  86656 |00:00:00.01 |     208 |
-----------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("OBJECT_TYPE"='TABLE')
```


285

Execution Plan

```
SQL> CONN SCOTT/TIGER
SQL> ALTER SESSION SET OPTIMIZER_DYNAMIC_SAMPLING = 0;
SQL> EXEC DBMS_STATS.DELETE_TABLE_STATS(USER, 'T91');
SQL> SELECT COUNT(OBJECT_NAME) FROM T91 WHERE OBJECT_TYPE = 'TABLE';
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(NULL, NULL, 'ALLSTATS LAST'));
SQL> SELECT COUNT(OBJECT_NAME) FROM T91 WHERE OBJECT_TYPE = 'DIRECTORY';
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(NULL, NULL, 'ALLSTATS LAST'));
SQL> SELECT COUNT(OBJECT_NAME) FROM T91 WHERE OBJECT_TYPE = 'SYNONYM';
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(NULL, NULL, 'ALLSTATS LAST'));
```


286

Execution Plan

```
SQL> EXEC DBMS_STATS.DELETE_TABLE_STATS(USER, 'T91');
SQL> EXEC DBMS_STATS.GATHER_TABLE_STATS(USER, 'T91', ESTIMATE_PERCENT=>100,
METHOD_OPT=>'FOR ALL COLUMNS SIZE 1');
SQL> SELECT COUNT(OBJECT_NAME) FROM T91 WHERE OBJECT_TYPE = 'TABLE';
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(NULL, NULL, 'ALLSTATS LAST'));
SQL> SELECT COUNT(OBJECT_NAME) FROM T91 WHERE OBJECT_TYPE = 'DIRECTORY';
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(NULL, NULL, 'ALLSTATS LAST'));
SQL> SELECT COUNT(OBJECT_NAME) FROM T91 WHERE OBJECT_TYPE = 'SYNONYM';
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(NULL, NULL, 'ALLSTATS LAST'));
```


287

Execution Plan

```
SQL> EXEC DBMS_STATS.DELETE_TABLE_STATS(USER, 'T91');
SQL> EXEC DBMS_STATS.GATHER_TABLE_STATS(USER, 'T91', ESTIMATE_PERCENT=>100,
METHOD_OPT=> 'FOR ALL COLUMNS SIZE 5');
SQL> SELECT COUNT(OBJECT_NAME) FROM T91 WHERE OBJECT_TYPE = 'TABLE';
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(NULL, NULL, 'ALLSTATS LAST'));
SQL> SELECT COUNT(OBJECT_NAME) FROM T91 WHERE OBJECT_TYPE = 'DIRECTORY';
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(NULL, NULL, 'ALLSTATS LAST'));
SQL> SELECT COUNT(OBJECT_NAME) FROM T91 WHERE OBJECT_TYPE = 'SYNONYM';
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(NULL, NULL, 'ALLSTATS LAST'));
```

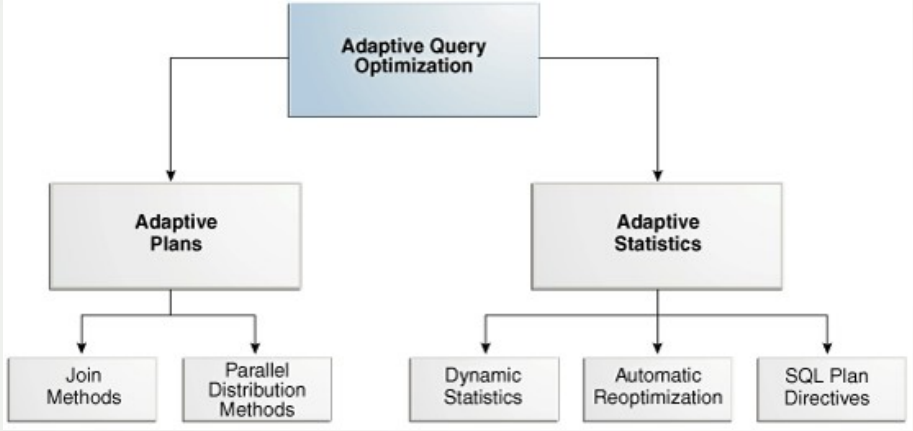

288

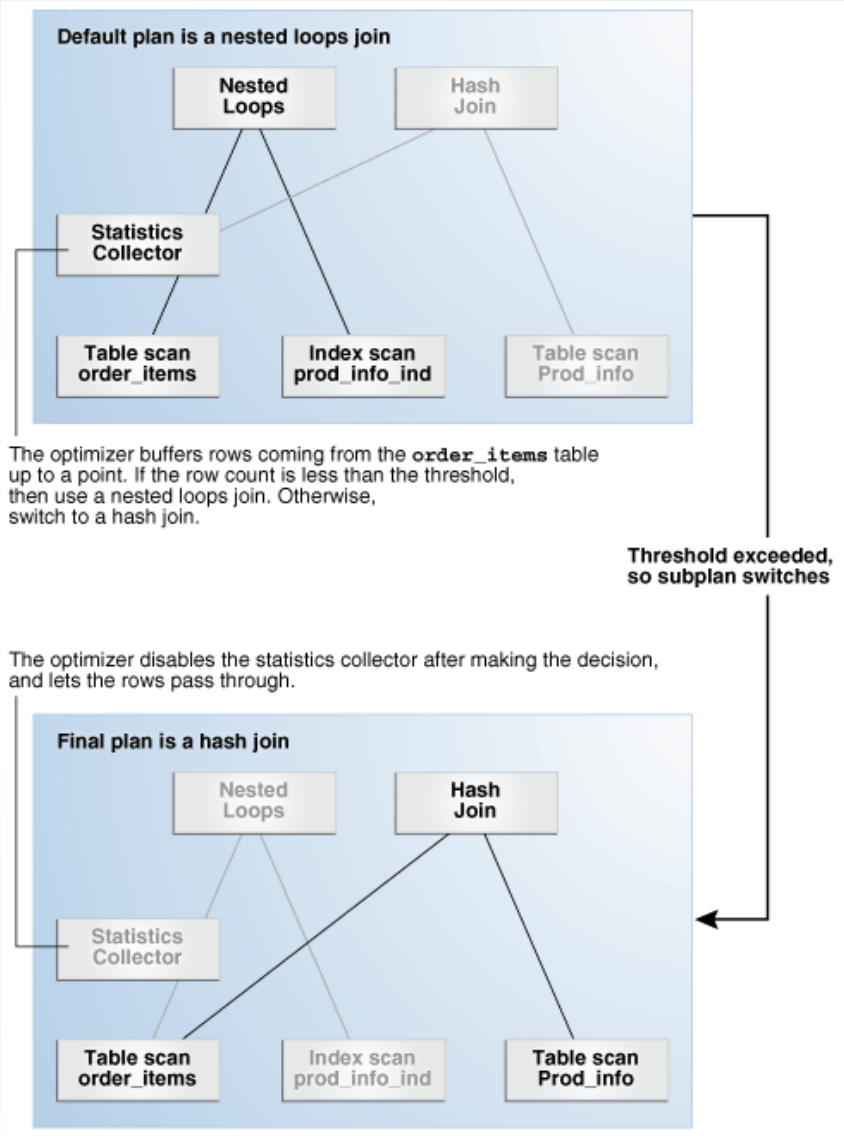
Execution Plan

```
SQL> EXEC DBMS_STATS.DELETE_TABLE_STATS(USER, 'T91');
SQL> EXEC DBMS_STATS.GATHER_TABLE_STATS(USER, 'T91');
SQL> SELECT COUNT(OBJECT_NAME) FROM T91 WHERE OBJECT_TYPE = 'TABLE';
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(NULL, NULL, 'ALLSTATS LAST'));
SQL> SELECT COUNT(OBJECT_NAME) FROM T91 WHERE OBJECT_TYPE = 'DIRECTORY';
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(NULL, NULL, 'ALLSTATS LAST'));
SQL> SELECT COUNT(OBJECT_NAME) FROM T91 WHERE OBJECT_TYPE = 'SYNONYM';
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(NULL, NULL, 'ALLSTATS LAST'));
```


289

Adaptive Plans





290

Lab 12.3: Adaptive Plans

Execute o SQL abaixo duas vezes, comparando o Plano Real de Execução.

```
SQL>  ALTER SYSTEM SET STATISTICS_LEVEL=ALL;
$ sqlplus OE/OE
SQL> SELECT o.order_id, v.product_name
FROM   orders o,
       (  SELECT order_id, product_name
          FROM   order_items o, product_information p
          WHERE  p.product_id = o.product_id
          AND    list_price < 50
          AND    min_price < 40  ) v
WHERE  o.order_id = v.order_id;
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(FORMAT=>'+ALLSTATS'));

SQL> SELECT o.order_id, v.product_name
FROM   orders o,
       (  SELECT order_id, product_name
          FROM   order_items o, product_information p
          WHERE  p.product_id = o.product_id
          AND    list_price < 50
          AND    min_price < 40  ) v
WHERE  o.order_id = v.order_id;
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(FORMAT=>'+ALLSTATS'));
```

Lab 12.4: Adaptive Plans

Verifique a diferença entre os dois Cursores.

```
SQL> SELECT SQL_ID, CHILD_NUMBER, SQL_TEXT FROM V$SQL WHERE SQL_TEXT LIKE
'%list_price <%';
SQL> SELECT CHILD_NUMBER, CPU_TIME, ELAPSED_TIME, BUFFER_GETS
FROM   V$SQL
WHERE  SQL_ID = 'b7gtpsznk6ww9';

SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR('b7gtpsznk6ww9', 0,
'ALLSTATS LAST'));
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR('b7gtpsznk6ww9', 1,
'ALLSTATS LAST'));
```

Lab 12.5: Adaptive Plans

Execute o SQL abaixo duas vezes, comparando o Plano Real de Execução.

```
$ sqlplus OE/OE
EXPLAIN PLAN FOR
  SELECT product_name FROM order_items o, product_information p
  WHERE  o.unit_price = 15
  AND    quantity > 1
  AND    p.product_id = o.product_id;
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);

SELECT product_name FROM order_items o, product_information p
  WHERE  o.unit_price = 15
  AND    quantity > 1
  AND    p.product_id = o.product_id;
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(FORMAT=>'+ALLSTATS LAST'));

SELECT product_name FROM order_items o, product_information p
WHERE  o.unit_price = 15
AND    quantity > 1
AND    p.product_id = o.product_id;
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(FORMAT=>'+ALLSTATS LAST
+ADAPTIVE'));
```

SQLT

Lab 13.1: SQLT - Instalação

MOS Doc ID 215187.1: SQLT Diagnostic Tool

Execute a instalação do SQLT.

```
$ unzip sqlt_latest.zip
$ cd sqlt/install
$ sqlplus / AS SYSDBA
SQL> ALTER SESSION SET CONTAINER = PROD;
SQL> @install
@PROD <ENTER>
Nerv2018 <ENTER>
Nerv2018 <ENTER>
YES <ENTER>
USERS <ENTER>
TEMP <ENTER>
SCOTT <ENTER>
T <ENTER>

SQL> GRANT SQLT_USER_ROLE TO OE;
SQL> GRANT SQLT_USER_ROLE TO SH;
SQL> GRANT SQLT_USER_ROLE TO SHSB;
SQL> GRANT SQLT_USER_ROLE TO HR;
```

Lab 13.2: SQLTXPLAIN - Extração

Verifique os SQL_ID dos SQL executados pelos dois SCHEMAS.

```
$ cd /home/oracle
SQL> CONN / AS SYSDBA
SQL> @SQLT.sql
SQL> SELECT SQL_ID, CHILD_NUMBER, SQL_TEXT FROM V$SQL WHERE
PARSING_SCHEMA_NAME = 'SHSB' AND CHILD_NUMBER > 2 ORDER BY 2
DESC;
```

Extraia o Relatório SQLT dos SQL_ID executados pelos dois SCHEMAS.

```
$ cd sqlt/run
$ sqlplus SHSB/SHSB
SQL> @sqltxtract 18dq002ybpmx9
Nerv2018 <ENTER>
$ unzip sqlt_s36985_xtract_grgrdq5ja4a1x.zip
$ firefox sqlt_s36985_main.html
```

Análise de Planos de Execução

Análise de Plano

```
SQL> SET LINES 300
SQL> SET AUTOTRACE ON
SQL> SET AUTOTRACE TRACEONLY EXPLAIN
SQL> SELECT COUNT(*) FROM ALL_OBJECTS;

Plano de Execuc?o
----------------------------------------------------------
Plan hash value: 3032110733

-------------------------------------------------------------------------------------------------------
| Id  | Operation                        | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                 |                     |     1 |   131 |   403   (2)| 00:00:01 |
|   1 |  SORT AGGREGATE                  |                     |     1 |   131 |            |          |
|*  2 |   FILTER                         |                     |       |       |            |          |
|*  3 |    HASH JOIN                     |                     | 99533 |    12M|   403   (2)| 00:00:01 |
|   4 |     INDEX FULL SCAN              | I_USER2             |   135 |   540 |     1   (0)| 00:00:01 |
|*  5 |     HASH JOIN                    |                     | 99533 |    12M|   402   (2)| 00:00:01 |
|   6 |      INDEX FULL SCAN             | I_USER2             |   135 |  3240 |     1   (0)| 00:00:01 |
|*  7 |      TABLE ACCESS FULL           | OBJ$                | 99533 |     9M|   400   (2)| 00:00:01 |
|*  8 |    TABLE ACCESS BY INDEX ROWID   | IND$                |     1 |     8 |     2   (0)| 00:00:01 |
|*  9 |     INDEX UNIQUE SCAN            | I_IND1              |     1 |       |     1   (0)| 00:00:01 |
|  10 |    NESTED LOOPS                  |                     |     1 |    15 |     2   (0)| 00:00:01 |
|  11 |     FIXED TABLE FULL             | X$KZSRO             |     2 |     6 |     0   (0)| 00:00:01 |
|* 12 |     INDEX RANGE SCAN             | I_OBJAUTH1          |     1 |    12 |     1   (0)| 00:00:01 |
|* 13 |    FIXED TABLE FULL              | X$KZSPR             |     1 |     7 |     0   (0)| 00:00:01 |
|* 14 |    FIXED TABLE FULL              | X$KZSPR             |     2 |    14 |     0   (0)| 00:00:01 |
|  15 |    NESTED LOOPS                  |                     |     1 |    15 |     2   (0)| 00:00:01 |
|  16 |     FIXED TABLE FULL             | X$KZSRO             |     2 |     6 |     0   (0)| 00:00:01 |
|* 17 |     INDEX RANGE SCAN             | I_OBJAUTH1          |     1 |    12 |     1   (0)| 00:00:01 |
|  18 |    NESTED LOOPS                  |                     |     1 |    15 |     2   (0)| 00:00:01 |
|  19 |     FIXED TABLE FULL             | X$KZSRO             |     2 |     6 |     0   (0)| 00:00:01 |
```

Análise de Plano

- Stand-alone Operations: apenas uma operação filho;
- Unrelated-combine Operations: operações filho são executadas de forma independente;
AND-EQUAL, BITMAP AND, BITMAP MINUS, CONCATENATION, CONNECT BY WITHOUT FILTERING, HASH JOIN, INTERSECTION, MERGE JOIN, MINUS, MULTI_TABLE INSERT, SQL MODEL, TEMP TABLE TRANSFORMATION, UNION-ALL.
- **Related-combine Operations:** a primeira operação filho é executada apenas uma vez, e controla a execução das outras operações filho, de forma intercalada;
NESTED LOOPS, UPDATE, CONNECT BY WITH FILTERING, BITMAP KEY ITERATION;
- **Blocking Operations** (SORT)
- Nonblocking Operations (FILTER)

Não há planos de execução difíceis, somente longos.
JL

Stand-alone Operation

```
SQL> SELECT DEPTNO, COUNT(*) FROM EMP WHERE JOB = 'CLERK' AND SAL < 1200 GROUP BY DEPTNO;

    DEPTNO   COUNT(*)
---------- ----------
        30          1
        20          2

SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(NULL, NULL, 'ALLSTATS LAST'));

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------------------------
SQL_ID  611zsndqdz83c, child number 1
-------------------------------------
SELECT DEPTNO, COUNT(*) FROM EMP WHERE JOB = 'CLERK' AND SAL < 1200
GROUP BY DEPTNO

Plan hash value: 4067220884

------------------------------------------------------------------------------------------------------------------------
| Id  | Operation          | Name | Starts | E-Rows | A-Rows |   A-Time   | Buffers |  OMem |  1Mem | Used-Mem |
------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |      1 |        |      2 |00:00:00.01 |       7 |       |       |          |
|   1 |  HASH GROUP BY     |      |      1 |      2 |      2 |00:00:00.01 |       7 |  1600K|  1600K|  613K (0)|
|*  2 |   TABLE ACCESS FULL| EMP  |      1 |      3 |      3 |00:00:00.01 |       7 |       |       |          |
------------------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter(("SAL"<1200 AND "JOB"='CLERK'))
```


300

Stand-alone Operation - STOPKEY

```
SQL> SELECT * FROM EMP WHERE ROWNUM <= 10;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250       1400         30
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30

10 rows selected.

SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(NULL, NULL, 'ALLSTATS LAST'));

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------
SQL_ID  8wjxk880pxg89, child number 0
-------------------------------------
SELECT * FROM EMP WHERE ROWNUM <= 10

Plan hash value: 1973284518

-----------------------------------------------------------------------------------------
| Id  | Operation          | Name | Starts | E-Rows | A-Rows |   A-Time   | Buffers |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |      1 |        |     10 |00:00:00.01 |       5 |
|*  1 |  COUNT STOPKEY     |      |      1 |        |     10 |00:00:00.01 |       5 |
|   2 |   TABLE ACCESS FULL| EMP  |      1 |     10 |     10 |00:00:00.01 |       5 |
-----------------------------------------------------------------------------------------
```


301

Stand-alone Operation - STOPKEY

```
SQL> SELECT * FROM (SELECT * FROM EMP ORDER BY SAL DESC) WHERE ROWNUM <= 10;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
```

```
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(NULL, NULL, 'ALLSTATS LAST'));

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------------------------------------
SQL_ID  62ux0y5crdj1j, child number 0
-------------------------------------
SELECT * FROM (SELECT * FROM EMP ORDER BY SAL DESC) WHERE ROWNUM <= 10

Plan hash value: 1744961472

------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                | Name | Starts | E-Rows | A-Rows |   A-Time   | Buffers |  OMem |  1Mem | Used-Mem |
------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |      |      1 |        |     10 |00:00:00.01 |       6 |       |       |          |
|*  1 |  COUNT STOPKEY           |      |      1 |        |     10 |00:00:00.01 |       6 |       |       |          |
|   2 |   VIEW                   |      |      1 |     14 |     10 |00:00:00.01 |       6 |       |       |          |
|*  3 |    SORT ORDER BY STOPKEY |      |      1 |     14 |     10 |00:00:00.01 |       6 |  2048 |  2048 | 2048  (0)|
|   4 |     TABLE ACCESS FULL    | EMP  |      1 |     14 |     14 |00:00:00.01 |       6 |       |       |          |
------------------------------------------------------------------------------------------------------------------------------------
```

Stand-alone Operation - FILTER

```
SQL> SELECT * FROM EMP WHERE JOB = 'CLERK' AND 1=2;

no rows selected

SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(NULL, NULL, 'ALLSTATS LAST'));

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------
SQL_ID  d68rp38qava95, child number 0
-------------------------------------
SELECT * FROM EMP WHERE JOB = 'CLERK' AND 1=2

Plan hash value: 3896240783

-----------------------------------------------------------------------------
| Id  | Operation          | Name | Starts | E-Rows | A-Rows |   A-Time   |
-----------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |      1 |        |      0 |00:00:00.01 |
|*  1 |  FILTER            |      |      1 |        |      0 |00:00:00.01 |
|*  2 |   TABLE ACCESS FULL| EMP  |      0 |      3 |      0 |00:00:00.01 |
-----------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter(NULL IS NOT NULL)
   2 - filter("JOB"='CLERK')
```


303

Unrelated-combine Operation

```
SQL> SELECT ENAME FROM EMP UNION ALL SELECT DNAME FROM DEPT UNION ALL SELECT '%' FROM DUAL;

ENAME
--------------
SMITH
ALLEN
```

```
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(NULL, NULL, 'ALLSTATS LAST'));

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------
SQL_ID  0ab27a051v1hd, child number 0
-------------------------------------
SELECT ENAME FROM EMP UNION ALL SELECT DNAME FROM DEPT UNION ALL SELECT
'%' FROM DUAL

Plan hash value: 4181933179

-----------------------------------------------------------------------------------------------------
| Id  | Operation          | Name | Starts | E-Rows | A-Rows |   A-Time   | Buffers | Reads  |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |      1 |        |     19 |00:00:00.01 |      16 |      4 |
|   1 |  UNION-ALL         |      |      1 |        |     19 |00:00:00.01 |      16 |      4 |
|   2 |   TABLE ACCESS FULL| EMP  |      1 |     14 |     14 |00:00:00.01 |       8 |      0 |
|   3 |   TABLE ACCESS FULL| DEPT |      1 |      4 |      4 |00:00:00.01 |       8 |      4 |
|   4 |   FAST DUAL        |      |      1 |      1 |      1 |00:00:00.01 |       0 |      0 |
-----------------------------------------------------------------------------------------------------
```


304

Related-combine Operation

```
SQL> SELECT EMPNO, ENAME, DNAME, LOC FROM EMP, DEPT WHERE EMP.DEPTNO = DEPT.DEPTNO AND DEPT.DEPTNO = 10;

     EMPNO ENAME      DNAME          LOC
---------- ---------- -------------- -------------
      7782 CLARK      ACCOUNTING     NEW YORK
      7839 KING       ACCOUNTING     NEW YORK
      7934 MILLER     ACCOUNTING     NEW YORK
```

```
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(NULL, NULL, 'ALLSTATS LAST'));

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------
SQL_ID  316w41h1thpg1, child number 0
-------------------------------------
SELECT EMPNO, ENAME, DNAME, LOC FROM EMP, DEPT WHERE EMP.DEPTNO =
DEPT.DEPTNO AND DEPT.DEPTNO = 10

Plan hash value: 568005898

-----------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name    | Starts | E-Rows | A-Rows |   A-Time   | Buffers |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |         |      1 |        |      3 |00:00:00.01 |      10 |
|   1 |  NESTED LOOPS                |         |      1 |      5 |      3 |00:00:00.01 |      10 |
|   2 |   TABLE ACCESS BY INDEX ROWID| DEPT    |      1 |      1 |      1 |00:00:00.01 |       2 |
|*  3 |    INDEX UNIQUE SCAN         | PK_DEPT |      1 |      1 |      1 |00:00:00.01 |       1 |
|*  4 |   TABLE ACCESS FULL          | EMP     |      1 |      5 |      3 |00:00:00.01 |       8 |
-----------------------------------------------------------------------------------------------------
```


305

O que procurar?

- Ponto de aumento de Cost ou Rows.
- Diferença entre A-Rows e E-Rows.
- Nested Loops com grande quantidade de Starts.
- FTS / FIS em objetos com filtros.
- Desperdício:

```
-------------------------------------------------------------------------------
| Id  | Operation                     | Name  | Starts | E-Rows | A-Rows |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |       |      1 |        |      0 |
|   1 |  MERGE JOIN                   |       |      1 |     10 |      0 |
|*  2 |   TABLE ACCESS BY INDEX ROWID | T1    |      1 |     10 |     10 |
|   3 |    INDEX FULL SCAN            | T1_PK |      1 |  10000 |  10000 |
|*  4 |   SORT JOIN                   |       |     10 |     10 |      0 |
|*  5 |    TABLE ACCESS BY INDEX ROWID| T2    |      1 |     10 |     10 |
|   6 |     INDEX FULL SCAN           | T2_PK |      1 |  10000 |  10000 |
-------------------------------------------------------------------------------
```

Lab 14.1: Análise de Plano

Execute o SQL abaixo e analise seu plano de execução.

```
SQL> CONN HR/HR
SQL> SELECT e.employee_id, j.job_title, e.salary, d.department_name
       FROM employees e, jobs j, departments d
       WHERE  e.employee_id < 103
       AND e.job_id = j.job_id
       AND e.department_id = d.department_id;

SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR
       (NULL, NULL, 'ALLSTATS LAST'));
```

Lab 14.2: Análise de Plano

Execute o SQL abaixo e analise seu plano de execução.

```
SQL> CONN SHSB/SHSB
SQL> SELECT *
FROM
  (SELECT times.calendar_quarter_desc,
     customers.cust_first_name,
     customers.cust_last_name,
     customers.cust_id,
     SUM(sales.amount_sold),
     rank() over(PARTITION BY times.calendar_quarter_desc
   ORDER BY SUM(amount_sold) DESC) AS
  rank_within_quarter
   FROM sales,
     customers,
     times
   WHERE sales.cust_id = customers.cust_id
   AND times.calendar_quarter_desc = '2002-2'
   AND times.time_id = sales.time_id
   GROUP BY customers.cust_id,
     customers.cust_first_name,
     customers.cust_last_name,
     customers.cust_id,
     times.calendar_quarter_desc)
WHERE rank_within_quarter < 16;
```

Lab 14.3: Análise de Plano

Execute o SQL abaixo e analise seu plano de execução.

```
SQL> CONN SHSB/SHSB
SQL> SELECT t.time_id,
  to_char(SUM(amount_sold),   '9,999,999,999') AS sales,
  to_char(AVG(SUM(amount_sold)) over(ORDER BY t.time_id range BETWEEN INTERVAL '2'
DAY preceding AND INTERVAL '2' DAY following),   '9,999,999,999') AS
centered_5_day_avg
FROM sales s,
  times t
WHERE t.calendar_month_desc IN('1996-01','1996-02','1996-03','1996-04')
 AND s.time_id = t.time_id
GROUP BY t.time_id
ORDER BY t.time_id;

SQL> SELECT t.time_id,
  to_char(SUM(amount_sold),   '9,999,999,999') AS sales,
  to_char(AVG(SUM(amount_sold)) over(ORDER BY t.time_id range BETWEEN INTERVAL '2'
DAY preceding AND INTERVAL '2' DAY following),   '9,999,999,999') AS
centered_5_day_avg
FROM sales s,
  times t
WHERE t.calendar_month_desc IN('2007-10','2007-11','2007-12','2008-01')
 AND s.time_id = t.time_id
GROUP BY t.time_id
ORDER BY t.time_id;
```

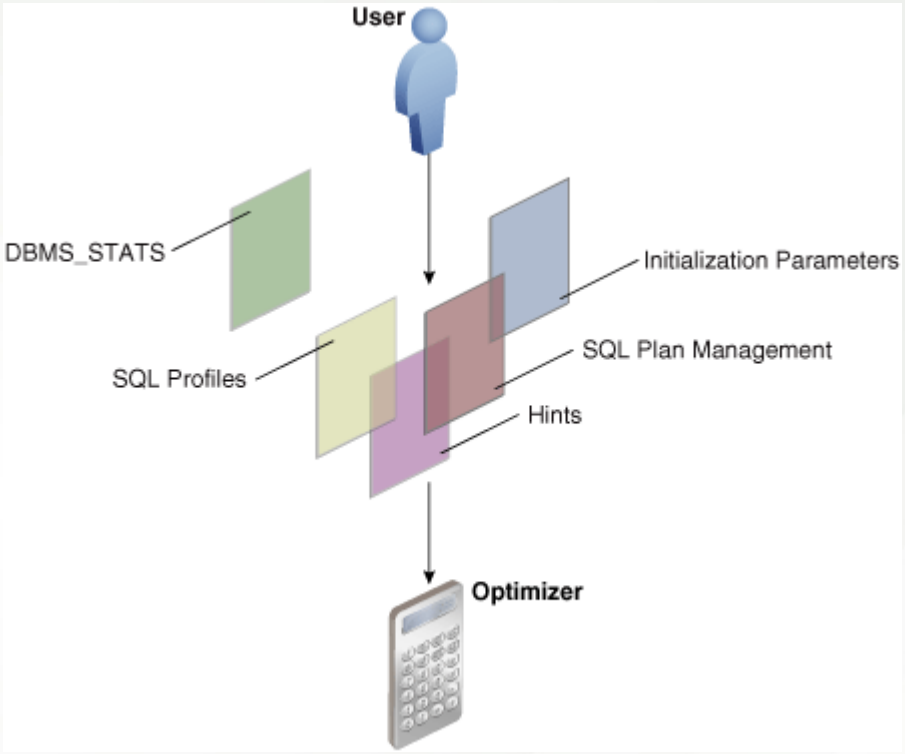
Estabilidade de Plano

Evolução de Shared SQL

- Bind Variable
- CURSOR_SHARING
- Bind Variable Peeking (9i)
- Extended Cursor Sharing / Adaptive Cursor Sharing (11gR1)
- Cardinality Feedback (11gR2)
- Adaptive Optimizer / Automatic Reoptimization / Adaptive Plan (12cR1)

Estabilidade do Plano

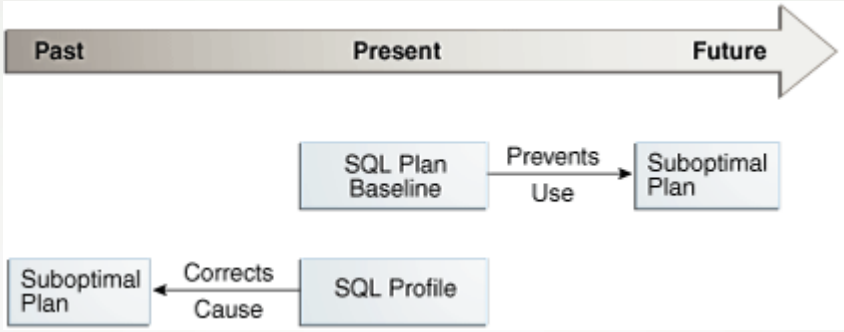
- HINTs (Session: "_OPTIMIZER_IGNORE_HINTS" = TRUE)
- Parâmetros de Controle do CBO
- Parâmetros ocultos de Controle do CBO
- Stored Outlines
- **SQL Profiles (DBA_SQL_PROFILES) (10g)**
- **SQL Plan Baselines (11g)**
- **SQL Plan Directives (12c)**

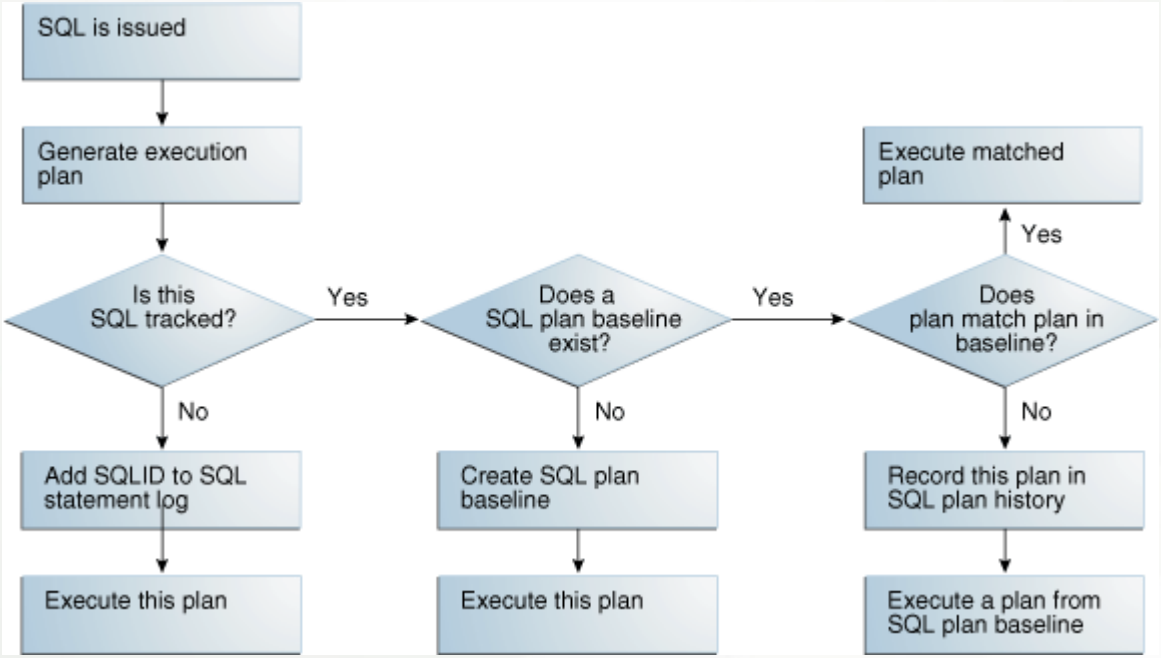


Fixar um Plano de Execução em Oracle Standard

```
SELECT ADDRESS, HASH_VALUE FROM GV$OPEN_CURSOR WHERE SQL_TEXT LIKE
'%ALL_TABLES%';
EXEC SYS.DBMS_SHARED_POOL.KEEP('1C5B28DC, 3958201300', 'C');
```

Baselines


Past
Present
Future
SQL Plan Baseline
Prevents
Use
Suboptimal Plan
Suboptimal Plan
Corrects
Cause
SQL Profile


SQL is issued
Generate execution plan
Is this SQL tracked?
Yes
No
Add SQLID to SQL statement log
Execute this plan
Does a SQL plan baseline exist?
Yes
No
Create SQL plan baseline
Execute this plan
Does plan match plan in baseline?
Yes
No
Execute matched plan
Record this plan in SQL plan history
Execute a plan from SQL plan baseline

Baselines

Consulta:
DBA_SQL_PLAN_BASELINES

Carregar Baselines Automaticamente:

```
SQL> ALTER SYSTEM SET OPTIMIZER_CAPTURE_SQL_PLAN_BASELINES=TRUE;
SQL> ALTER SYSTEM SET OPTIMIZER_USE_SQL_PLAN_BASELINES=TRUE;
```

Carregar Baselines manualmente:

```
SQL> EXEC DBMS_SPM.LOAD_PLANS_FROM_CURSOR_CACHE(ATTRIBUTE_NAME=>'SQL
TEXT', ATTRIBUTE_VALUE=>'%SELECT ID, NAME FROM T1%')
SQL> EXEC DBMS_SPM.LOAD_PLANS_FROM_CURSOR_CACHE(SQL_ID=>'1234567890',
PLAN_HASH_VALUE=>'ABCDEFGH');
```

Baselines

Evoluir Baselines:

```
SQL> SET LONG 10000
SQL> SELECT DBMS_SPM.EVOLVE_SQL_PLAN_BASELINE(sql_handle =>
'SYS_SQL_7b76323ad90440b9') FROM DUAL;
```

Remover Baselines:

```
SQL> DECLARE
       i NATURAL;
       BEGIN
       i := DBMS_SPM.DROP_SQL_PLAN_BASELINE('SQL_0c20446867a16450');
       DBMS_OUTPUT.PUT_LINE(i);
       END;
       /
```

Baselines

```
SQL> CONN / AS SYSDBA
SQL> ALTER SYSTEM SET OPTIMIZER_CAPTURE_SQL_PLAN_BASELINES=TRUE;
SQL> ALTER SYSTEM SET OPTIMIZER_USE_SQL_PLAN_BASELINES=TRUE;

SQL> CONN SCOTT/TIGER
SQL> CREATE TABLE T AS SELECT * FROM ALL_OBJECTS;
SQL> SELECT COUNT(DISTINCT(OWNER)) FROM T;
SQL> SELECT SQL_TEXT, SQL_HANDLE, PLAN_NAME, ENABLED, ACCEPTED FROM
DBA_SQL_PLAN_BASELINES;
SQL> CREATE BITMAP INDEX T_INDEX_01 ON T(OWNER);
SQL> SET SERVEROUTPUT ON
SQL> DECLARE EVOLVE_OUT CLOB;
      BEGIN
      EVOLVE_OUT := DBMS_SPM.EVOLVE_SQL_PLAN_BASELINE (SQL_HANDLE =>
      'SQL_0c20446867a16450', COMMIT => 'YES');
      DBMS_OUTPUT.PUT_LINE(EVOLVE_OUT);
      END;
      /
SQL> SELECT COUNT(DISTINCT(OWNER)) FROM T;
SQL> SELECT SQL_TEXT, SQL_HANDLE, PLAN_NAME, ENABLED, ACCEPTED FROM
DBA_SQL_PLAN_BASELINES;
```

Perguntas?

Ricardo Portilho Proni
ricardo@nervinformatica.com.br