Things to remember when working with Oracle... (for UDB specialists)

Kris Van Thillo, ABIS

ABIS Training & Consulting
www.abis.be
training@abis.be

2013

Document number: DB2LUWUserMeeting2013Front.fm
5 March 2013

Address comments concerning the contents of this publication to:
ABIS Training & Consulting, P.O. Box 220, B-3000 Leuven, Belgium
Tel.: (+32)-16-245610, Fax: (+32)-16-245639

Think about ...

- (1) Instances and Databases
- (2) Bufferpools
- (3) Tablespaces, Datafiles and extents
- (4) About UNDO
- (5) About physical row structures

- **(6) Tables and indexes**
- **(7) Nulls**
- **(8) Security**
- **(9) About Static and Dynamic SQL**
- **(10) Utilities**

Think about ...

1. Instances and Databases
2. Bufferpools
3. Tablespaces, Datafiles, and Extents
4. UNDO - tablespaces, segments
5. About physical row structures
6. Tables and Indexes
7. Nulls
8. Security
9. About static and dynamic SQL
10. Utilities

Instances and Databases

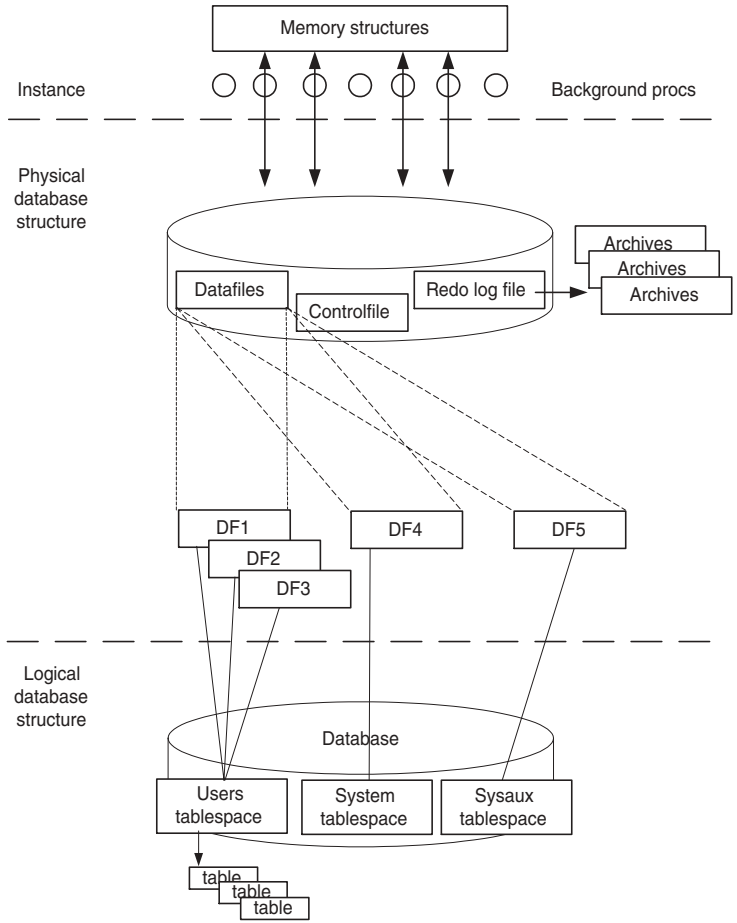
An instance is a database is an instance, right?

Basically, yes, ...

- an instance ‘contains’ - services - ONE database [ie. physical structure]

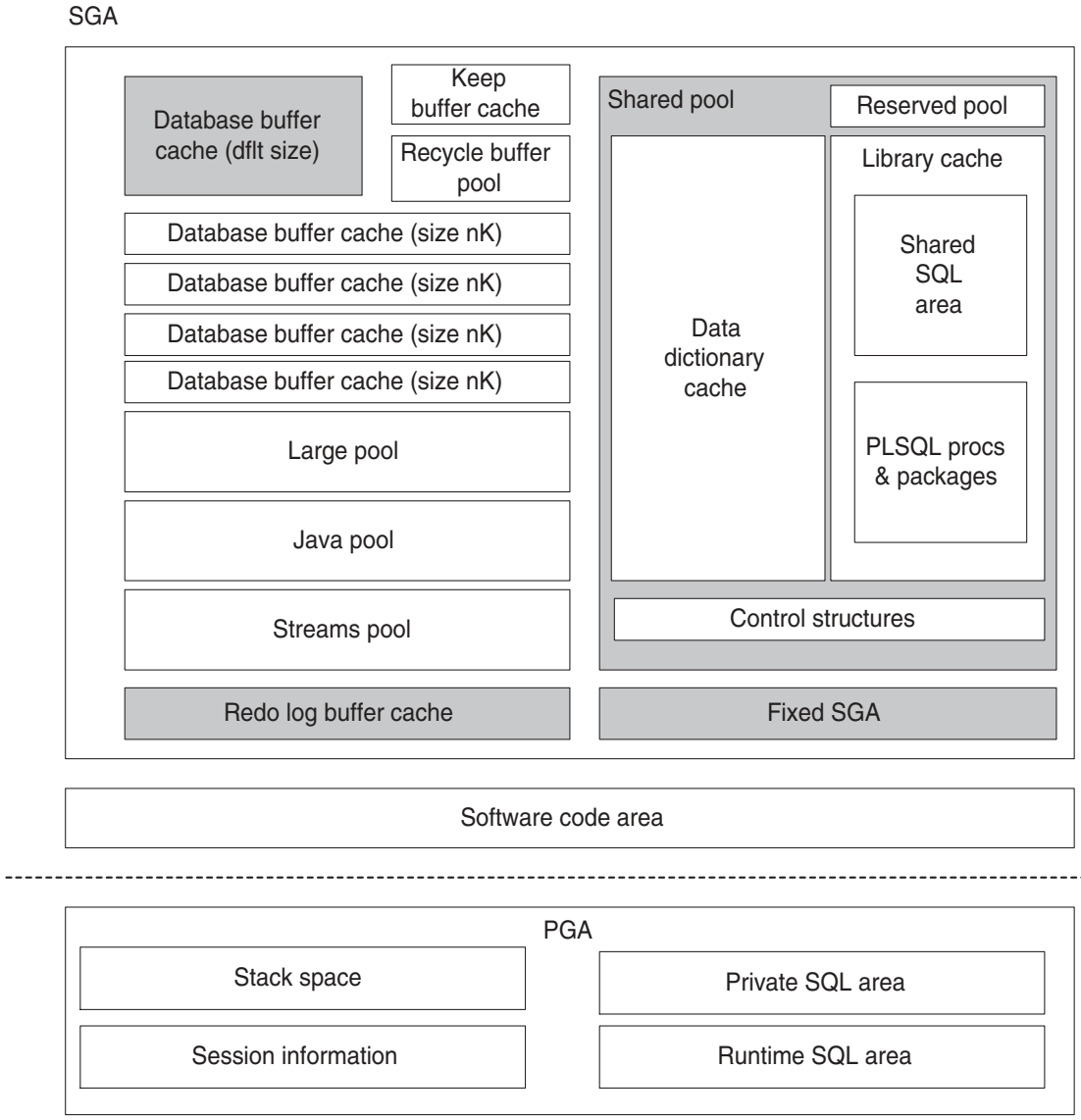
  [a database can be serviced by multiple instances in a RAC environment]
- the name of the database is *generally* of no importance, as the instance is the object to be managed
- you connect to the instance, not to the database
- if remote access is required, the instance should be locally ‘catalogued’ - added to local name resolution facilities
  (basic: tnsnames.ora)
- security by default implemented on an instance-by-instance (db by db) level

Think about ...



Think about ...

1. Instances and Databases
2. Bufferpools
3. Tablespaces, Datafiles, and Extents
4. UNDO - tablespaces, segments
5. About physical row structures
6. Tables and Indexes
7. Nulls
8. Security
9. About static and dynamic SQL
10. Utilities



Think about ...

1. Instances and Databases
2. Bufferpools
3. Tablespaces, Datafiles, and Extents
4. UNDO - tablespaces, segments
5. About physical row structures
6. Tables and Indexes
7. Nulls
8. Security
9. About static and dynamic SQL
10. Utilities

Bufferpools

- **Basically the same idea - cache data!**
- **Implemented differently:**
  - not CREATEd, but ‘activated’ or ‘enabled’
  - not explicitely named (cfr. z/OS)
  - typically one per required page/block size type
    [4 k, 8 k, 16 k, 32 k]
    [subs: default, keep, recycle]
  - no prefetchers - SPs read-in big blocks if deemed helpful
    [scan, index scan]
    DBA can influence what stays in memory, and, for how long [access type, object definition]
  - link to storage object through tablespaces

Think about ...

Tablespaces, Datafiles, and Extents

or if you prefer: Tablespaces, Containers, and Extents

An Oracle tablespace:

- is an automatically managed DMS tablespace
- assigned [default ?] to a bufferpool
- in which NO Oracle managed extent balancing/striping occurs

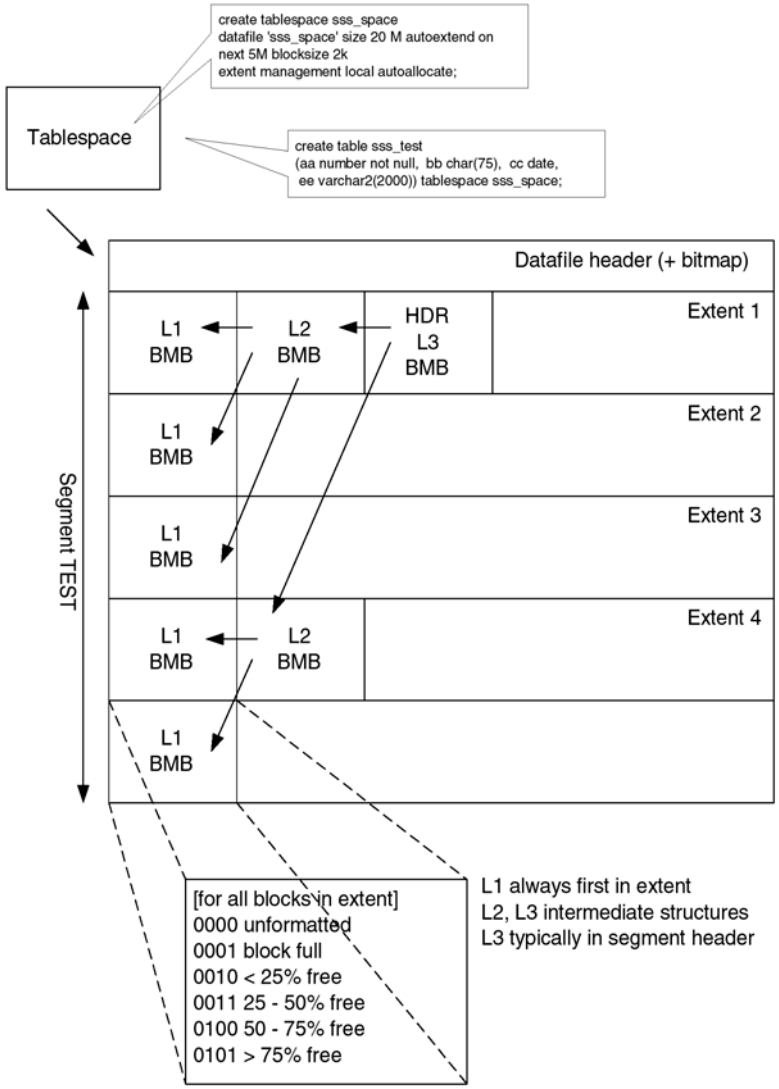
  if striping is important:

  °) do it manually - based on sizing/DBA intervention

  °) rely on other Oracle (ASM) or third party software (SAN, NAS based solutions) <=> bigfile tablespaces
- [system, sysaux, temp, undo, users (?) - and application based tablespaces]
- there is no such thing as an indexspace!

Think about ...

1. Instances and Databases
2. Bufferpools
3. Tablespaces, Datafiles, and Extents
4. UNDO - tablespaces, segments
5. About physical row structures
6. Tables and Indexes
7. Nulls
8. Security
9. About static and dynamic SQL
10. Utilities



Think about ...

1. Instances and Databases
2. Bufferpools
3. Tablespaces, Datafiles, and Extents
4. UNDO - tablespaces, segments
5. About physical row structures
6. Tables and Indexes
7. Nulls
8. Security
9. About static and dynamic SQL
10. Utilities

UNDO - tablespaces, segments

- **store rollback or 'undo' information**
  [undo is a more correct **term]**
- **used for** locking and read consistency**,** transaction rollback**, and** recovery - ***versioning!***
- [created/activated/enabled] **undo segments are stored in undo tablespaces - ie. physical objects**
- **general idea:**
  - data change:

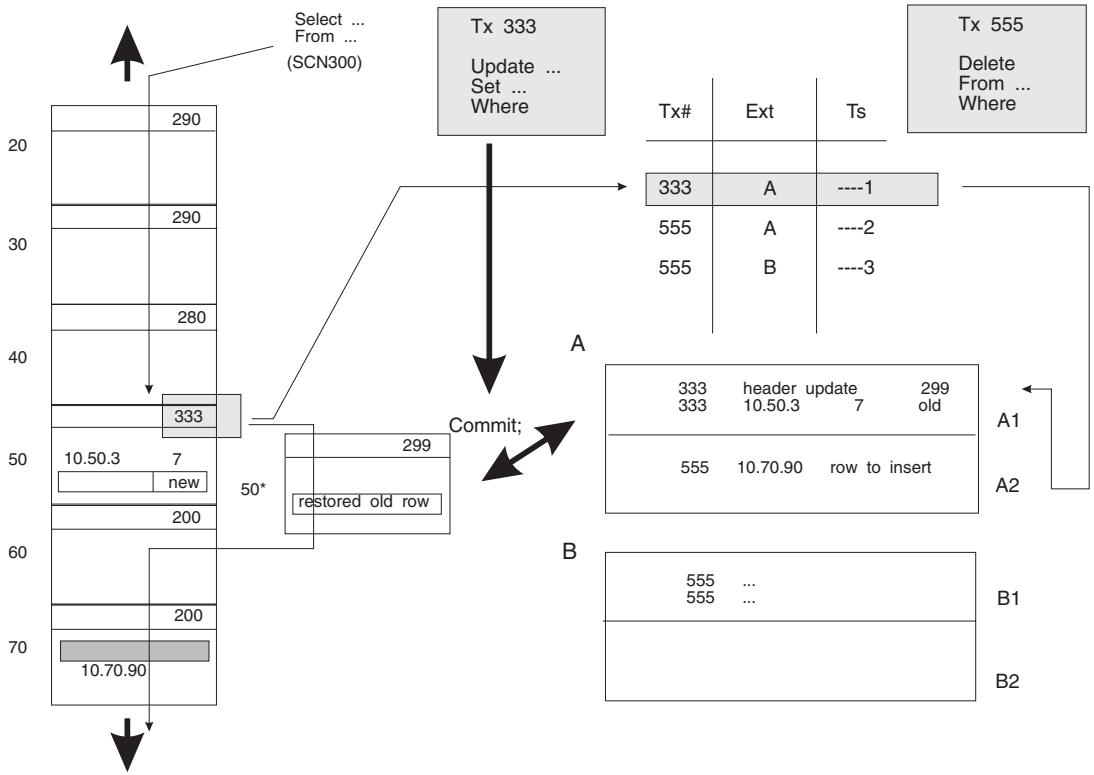
    -> after image (AI) to transaction log (redo log)

    -> before image (BI) to undo segment; this modifies the undo segment. Consequently:

    => AI undo modification is sent to the transaction log

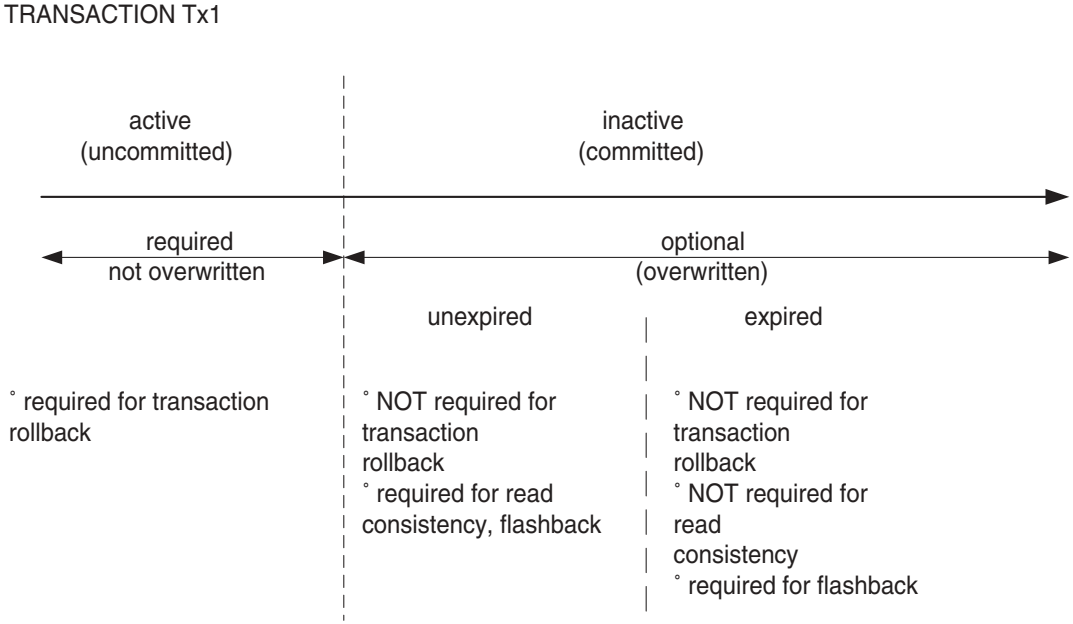
    => BI undo modification is sent to the undo segment
    [one extra generation kept]

Think about ...



Think about ...

1. Instances and Databases
2. Bufferpools
3. Tablespaces, Datafiles, and Extents
4. UNDO - tablespaces, segments
5. About physical row structures
6. Tables and Indexes
7. Nulls
8. Security
9. About static and dynamic SQL
10. Utilities



Think about ...

1. Instances and Databases
2. Bufferpools
3. Tablespaces, Datafiles, and Extents
4. UNDO - tablespaces, segments
5. About physical row structures
6. Tables and Indexes
7. Nulls
8. Security
9. About static and dynamic SQL
10. Utilities

UNDO - tablespaces, segments - cont.

Flashback and Rewind!

- **undo based:**
  - flashback query
  - flashback version query
  - flashback transaction query
  - flashback transaction backout
  - flashback table
- **not undo based:**
  - flashback drop (recycle bin)
  - flashback database (flashback logs)

Think about ...

1. Instances and Databases
2. Bufferpools
3. Tablespaces, Datafiles, and Extents
4. UNDO - tablespaces, segments
5. About physical row structures
6. Tables and Indexes
7. Nulls
8. Security
9. About static and dynamic SQL
10. Utilities

UNDO - tablespaces, segments - cont.

```
select count(*)
from test
as of timestamp(TO_TIMESTAMP('07-04-2005 06:30:00',
                             'DD-MM-YYYY HH24:MI:SS'));


declare
   CURSOR PrevVersion is
   SELECT * FROM test
   as of timestamp TO_TIMESTAMP('07-04-2005 06:25:00',
                                'DD-MM-YYYY HH24:MI:SS');
   Data     PrevVersion%ROWTYPE;
   RowExist number := 0;
begin
....
end;


execute dbms_flashback.enable_at_time(
        TO_TIMESTAMP('07-04-2005 06:30:00',
                     'DD-MM-YYYY HH24:MI:SS'));
```

Think about ...

1. Instances and Databases
2. Bufferpools
3. Tablespaces, Datafiles, and Extents
4. UNDO - tablespaces, segments
5. About physical row structures
6. Tables and Indexes
7. Nulls
8. Security
9. About static and dynamic SQL
10. Utilities

UNDO - tablespaces, segments - cont.

Remember!

- **versioning - and currently_committed for that matter - is an** ‘acquired taste’**!**

  advantages and disadvantages!

- **developping applications against a dbms enabled for versioning requires a distinct development skill set!**

  it will in all likelyhood require application changes!

Think about ...

1. Instances and Databases
2. Bufferpools
3. Tablespaces, Datafiles, and Extents
4. UNDO - tablespaces, segments
5. About physical row structures
6. Tables and Indexes
7. Nulls
8. Security
9. About static and dynamic SQL
10. Utilities

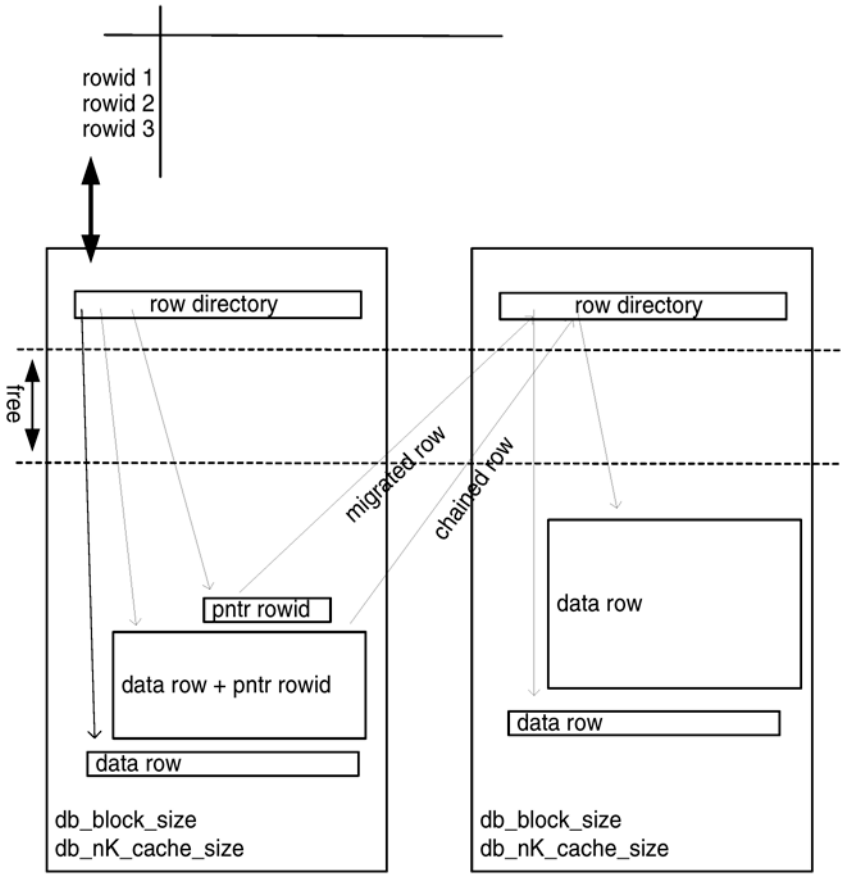
About physical row structures

- **row length is typically dynamic**
  - all columns are stored using ‘variable length’ semantics
  - numbers are typically stored ‘in decimal format’ (not binary)
  - nulls values are either implied or stored using placeholder

    [most accessed first, fixed first/variable last, null columns last]
- **no ‘max # rows per page’ concept; row length not limited to page size**
  - migration (growing below page size) - *moved to*
  - chaining (growing beyond page size) - *pieces*
- **rows are assigned to pages based on page free space,** not on any kind of ‘logical’, ie. value or key based, allocation scheme!
  [based on bitmaps - LHWM, HHWM]

Think about ...

1. Instances and Databases
2. Bufferpools
3. Tablespaces, Datafiles, and Extents
4. UNDO - tablespaces, segments
5. About physical row structures
6. Tables and Indexes
7. Nulls
8. Security
9. About static and dynamic SQL
10. Utilities



Think about ...

1. Instances and Databases
2. Bufferpools
3. Tablespaces, Datafiles, and Extents
4. UNDO - tablespaces, segments
5. About physical row structures
6. Tables and Indexes
7. Nulls
8. Security
9. About static and dynamic SQL
10. Utilities

Tables and Indexes

- **Basically the same**
- **Tables are divided in segments, to be assigned to a specific tablespace (or one, if so desired)**
  - column based [data type based]
  - row based - partitions [partitioned tables]
  - matrix structure [?]
  - different data types, LOBs stored in row OR out of row - your call

  Learn about/Read up on ... IOTs, [Hash] clusters, external tables, ... **and on the available partitioning options:** range, hash, list, interval, system, reference [ie. auto], ...
- **Indexes are created on tables, and stored independently**

  Learn about/Read up on ... reverse indexes, invisible indexes, function based indexes

Think about ...

1. Instances and Databases
2. Bufferpools
3. Tablespaces, Datafiles, and Extents
4. UNDO - tablespaces, segments
5. About physical row structures
6. Tables and Indexes
7. Nulls
8. Security
9. About static and dynamic SQL
10. Utilities

Nulls

- **null** is **an empty string** is **null**
- **nulls are:**
  - stored - or not!
  - never stored in traditional B*tree indexes [dense - sparse]

    stored in bitmap indexes
  - what about your where conditions? [not null with default?]

Think about ...

1. Instances and Databases
2. Bufferpools
3. Tablespaces, Datafiles, and Extents
4. UNDO - tablespaces, segments
5. About physical row structures
6. Tables and Indexes
7. Nulls
8. Security
9. About static and dynamic SQL
10. Utilities

Security

- **user** identification **and** validation **basically different!**
  - **Oracle managed -** eg. password reuse requirements, ...
  - **verified by Oracle or externally (by default 'not verified'** ie. trusted**)**
  - **no integration with default OS platform security**
- **Oracle uses privileges and roles:**

  Oracle roles <=> UDB authorisations
  Roles can however be added
- **'Advanced' features supported -** encryption, SSO, LDAP integration, ...

Think about ...

1. Instances and Databases
2. Bufferpools
3. Tablespaces, Datafiles, and Extents
4. UNDO - tablespaces, segments
5. About physical row structures
6. Tables and Indexes
7. Nulls
8. Security
9. About static and dynamic SQL
10. Utilities

About static and dynamic SQL

All Oracle SQL - both static and dynamic SQL - is DB2 UDB dynamic, that is: optimized at run time.

[optimized => parsed]

[QEP <=> plan; cached in pools cfr. dyn statement cache **(*)**]

- **hard parse:**
  - first time ever parse
- **soft parse**
  - reuse existing parse statement
- **softer-the-soft**
  - application sends 'parsed' statement to the server

(*) QEP stored in cursor area's - parent area, child area based on assumed and observed access path

stored outlines, profiles

Think about ...

1. Instances and Databases
2. Bufferpools
3. Tablespaces, Datafiles, and Extents
4. UNDO - tablespaces, segments
5. About physical row structures
6. Tables and Indexes
7. Nulls
8. Security
9. About static and dynamic SQL
10. Utilities

Utilities

Oracle: RMAN, Import/Export aka Dpump, Loader, Analyze (1), **db_verify, ??** (2), **??**(2)

DB2 UDB: Backup/Restore, Import/Export, Load, Runstats, ??, reorg, reorgchk

(1) dbms_stats.gather_<object>_stats
(2) use command sequences to verify multiple storage characteristics

Think about ...

1. Instances and Databases
2. Bufferpools
3. Tablespaces, Datafiles, and Extents
4. UNDO - tablespaces, segments
5. About physical row structures
6. Tables and Indexes
7. Nulls
8. Security
9. About static and dynamic SQL
10. Utilities

So what does Oracle compatibility mode do?

- **Datatypes:**
  - DATE data type based on TIMESTAMP(0)
    [supports oracle style computations, eg '+1']
  - NUMBER
    [decfloat(16), decimal(p,s)]
  - VARCHAR2 and NVARCHAR2 (*)
    [varchar, non-padded comparison semantics]
    (*) hardly ever used!
- **SQL data access level enforcement**
  [what to expect in a stored object - SQL, NoSQL, Write?]
- **Outer join operator ==> '+' (*)**
  (*) old school most Oracle apllis should use ansi SQL
  '+' refers to column where the missing values should be 'assumed present', to generate the outer rows
- **Hierarchical queries - connect by, start with, prior**
  [CTE in Oracle still have somewhat limited functionality]
- **INOUT** parameter used when creating Oracle stored procs

Think about ...

1. Instances and Databases
2. Bufferpools
3. Tablespaces, Datafiles, and Extents
4. UNDO - tablespaces, segments
5. About physical row structures
6. Tables and Indexes
7. Nulls
8. Security
9. About static and dynamic SQL
10. Utilities

So what does Oracle compatibility mode do? (II)

- **ROWNUM**
- **DUAL** table in Oracle is typically references with schema name [eg. old style coding requires sequence referencing using access to a table]
- **Oracle data dictionary-compatible views [sys.dba_... sys.all_... sys.user_...**
- **PL/SQL support**
- **truncate table**

Think about ...

1. Instances and Databases
2. Bufferpools
3. Tablespaces, Datafiles, and Extents
4. UNDO - tablespaces, segments
5. About physical row structures
6. Tables and Indexes
7. Nulls
8. Security
9. About static and dynamic SQL
10. Utilities

Thank you!

ABIS Training & Consulting
Kris Van Thillo
kvanthillo@abis.be

Think about ...

1. Instances and Databases
2. Bufferpools
3. Tablespaces, Datafiles, and Extents
4. UNDO - tablespaces, segments
5. About physical row structures
6. Tables and Indexes
7. Nulls
8. Security
9. About static and dynamic SQL
10. Utilities