Oracle 12.2 新特性介绍

DJI DBA Team
何剑敏

1 架构篇

2 管理篇

3 开发篇

1 架构篇

架构篇

1 版本更迭

2 Application container

3 Proxy PDB

4 Flex cluster

5 ASM增强

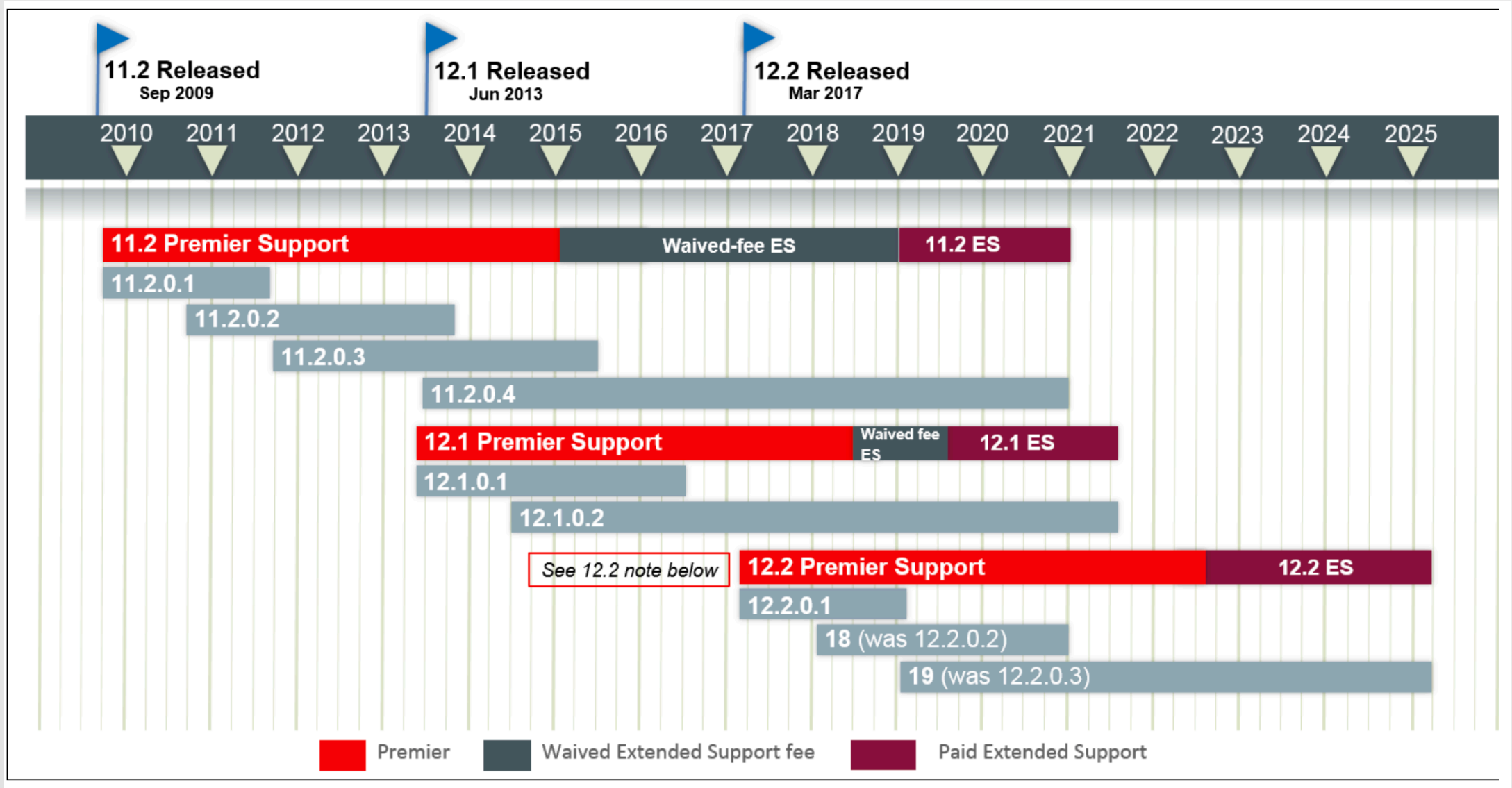
6 Sharding database

7 OFS server

8 其他

版本更迭

Release Schedule of Current Database Releases (Doc ID 742060.1)



版本更迭

Release Update and Release Update Revisions for Database Proactive Patch Program (Doc ID 2285040.1)

Frequently Asked Questions

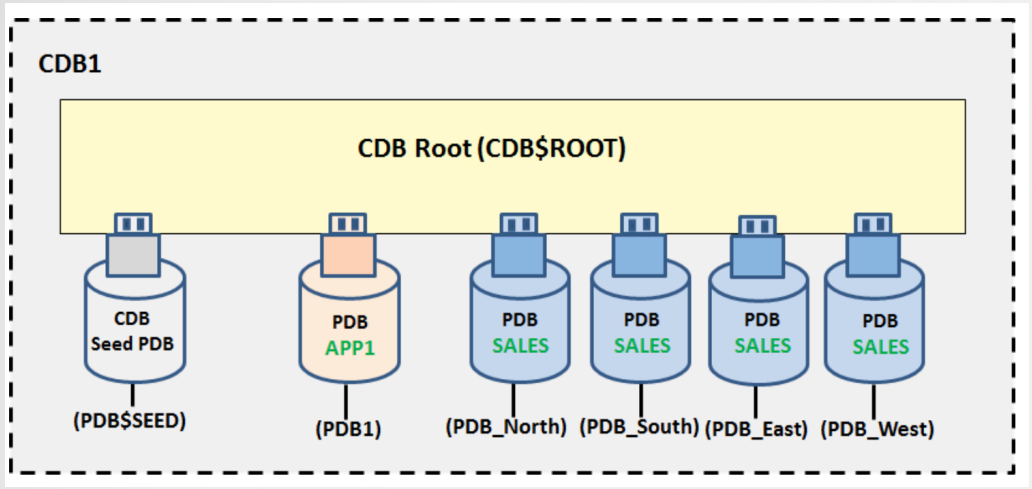
- Q1: Will individual Bug fixes be available until they are incorporated into a RU?
- Q2: Are customers required to apply RU's or RUR's in order to obtain new bug fixes?
- Q3: Are new features ever included into a RU?
- Q4: Are new features ever included into a RUR?
- Q5: Can Customers switch back and forth between RUs and RURs?
- Q6: What is changing for 12.2.0.1 in July 2017?
- Q7: Will database version 12.1 and 11.2 be affected?
- Q8: What happens to the quarterly CPU?
- Q9: How are RU's / RUR's packaged and installed?
- Q10: What database downtime does RU/RUR installation require?
- Q11: How do I find out about known issues with RU's and RUR's?
- Q12: Are optimizer fixes included into RUs?
- Q13: What lines of business are changing to this new release strategy?
- Q14: How can I easily see if all of my $ORACLE_HOMEs are at the same security level.
- Q15: Will the Windows patching model change?
- Q16: What will happen if RU 12.2.0.1.170718 is applied on top of DBBP 12.2.0.1.170620?

Production	July	October	Jan	April	July
12.2.0.1	Database Release Update 12.2.0.1.<build-date>	Database Release Update 12.2.0.1.<build-date>	Database Release Update 12.2.0.1.<build-date>	← RU	
		Database July 2017 Release Update Revision 12.2.0.1.<build-date>	Database Oct2017 Release Update Revision 12.2.0.1.<build-date>	← RUR #1	
			Database July 2017 Release Update Revision 12.2.0.1.<build-date>	← RUR #2	

大致情况：

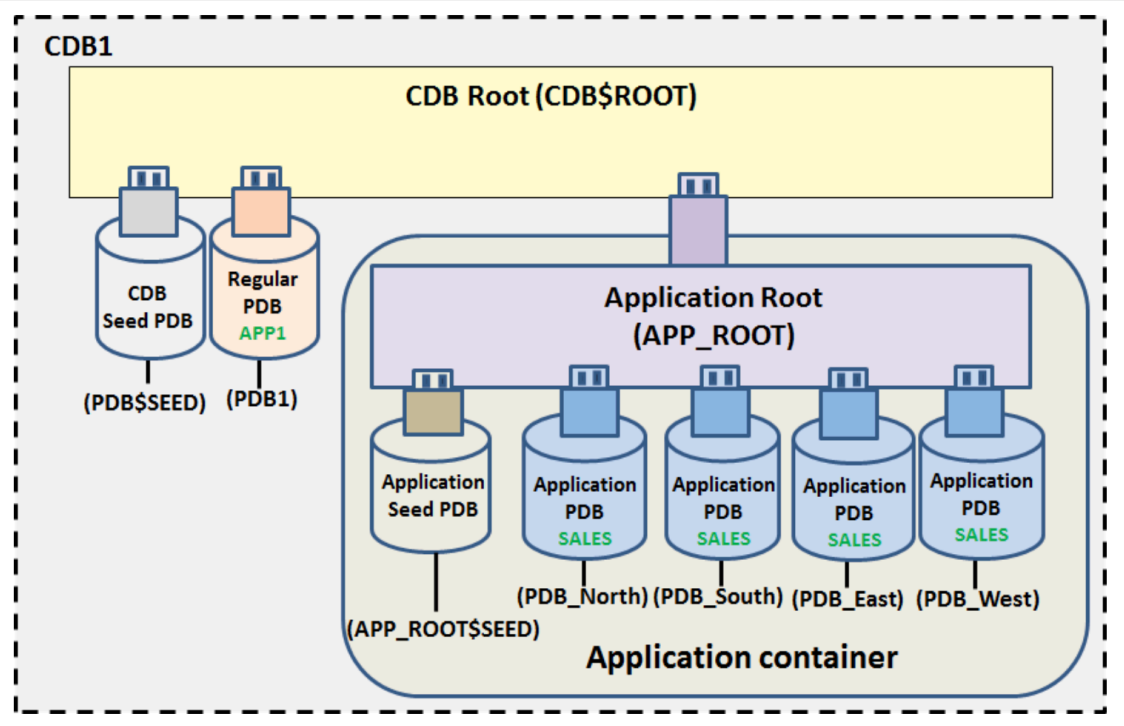
- Oracle 12.2.0.2= oracle 18c (大约2018年第一季度）；
- Oracle 12.2.0.3= oracle 19（大约2019年第一季度）；
- 当前计划是oracle19是oracle 12.2的最后一个版本，但将来也有可能oracle 20可能是oracle 12.2的最后一个版本。（涉及到paid extend support）；
- 12.2开始，不再PSU，或者DB Bundle patch；
- 12.1.0.2和11.2.0.4没有RU或者RUR；
- RU基本和BPs类似；
- RUR基本和PSU类似；
- 12.2.0.1不会有PSU和BPs；
- 2017年7月release了第一个RU（DB RU，Grid RU和OJVM RU）；
- 第一个RUR for 12.2.0.1将在2017年10月release；
- 第二个RU for 12.2.0.1将在2017年10月release；
- 第三个RU for 12.2.0.1，第二个RUR for 12.2.0.1将在2018年1月release；
- 正常情况下，每个RU只有2个RUR；

Application container

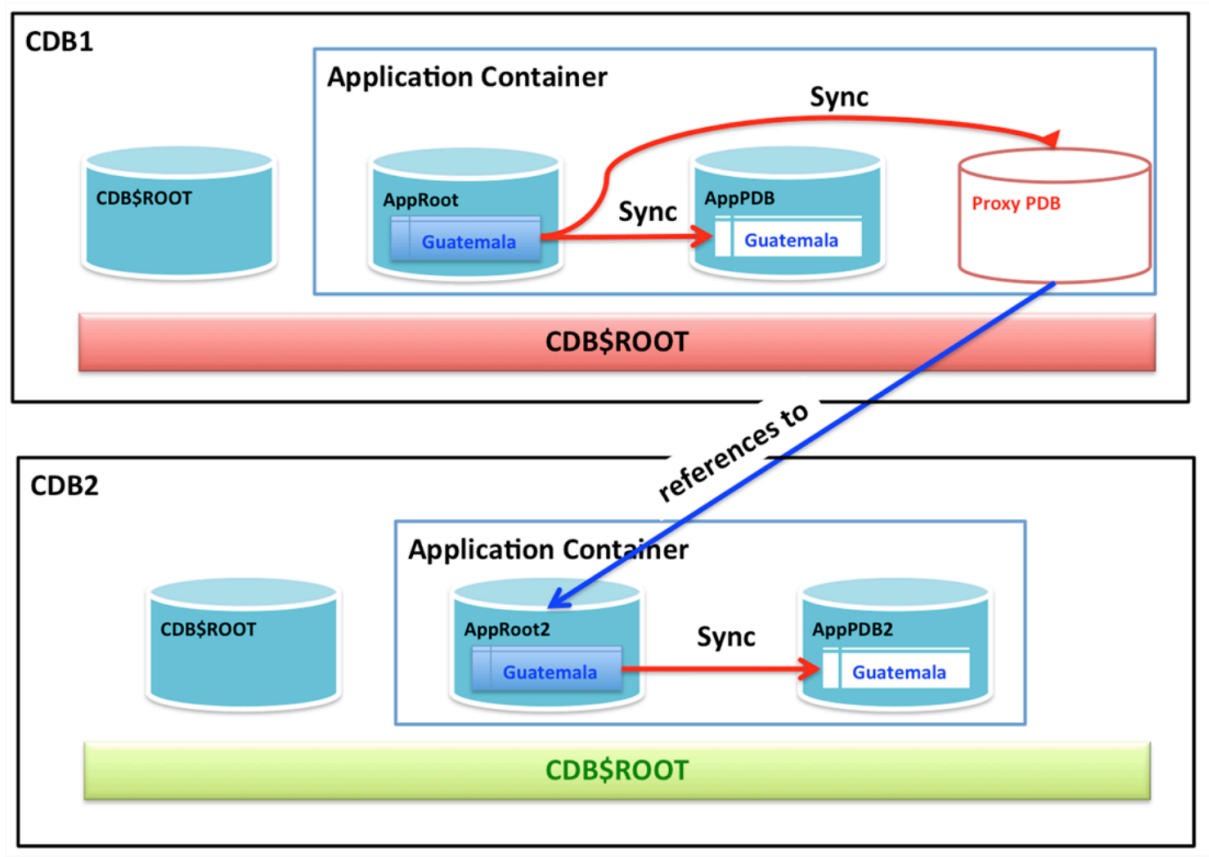


同一个application下的多个pdb的集合，组成一个application container。

目的，方便同一个的application升级脚本(同步脚本）。



Application container



有CDB root的概念，现在又有了application root的概念。

Application root可以是local的，也可以是proxy的。

Application container

创建步骤：

1.创建CDB（注意使用OMF）

2. 创建PDB_APP1 as application root:

CREATE PLUGGABLE DATABASE PDB_APP1 as APPLICATION CONTAINER …

3. 创建application seed for application pdb（可选）

4. connect to PDB_APP1

5.create PDB2 in PDB_APP1

Application container

常见操作： install

```
-- Begin the installation of application sales_app

SALES_APP_ROOT>ALTER PLUGGABLE DATABASE APPLICATION sales_app BEGIN INSTALL '1.0';

Pluggable database altered.

SALES_APP_ROOT>@get_app_status

APP_NAME                                 APP_VERSION    APP_ID APP_STATUS   IMPLICIT
---------------------------------------- ------------ ------ ------------ --------
SALES_APP                                                    3 INSTALLING   N
APP$5313F8AEAFA337B0E05364C909C08D65    1.0                  2 NORMAL     Y
```

```
-- Create common application user sales_app_user and grant necessary privileges

SALES_APP_ROOT>CREATE USER sales_app_user IDENTIFIED BY oracle  CONTAINER=ALL;
                GRANT CREATE SESSION, create procedure, CREATE TABLE,
                     unlimited tablespace TO sales_app_user;


-- Create common application metadata-linked table sales_app_user.customers

SALES_APP_ROOT>CREATE TABLE sales_app_user.customers SHARING=METADATA
               ( cust_id    NUMBER constraint cust_pk primary key,
                 cust_name  varchar2(30),
                 cust_add   varchar2(30),
                 cust_zip   NUMBER
               );
```

还可以是DATA，
或EXTENDED DATA

```
SALES_APP_ROOT> ALTER PLUGGABLE DATABASE APPLICATION sales_app END INSTALL '1.0';

SALES_APP_ROOT>@get_app_status

APP_NAME                                 APP_VERSION  APP_ID APP_STATUS   IMPLICIT
---------------------------------------- ------------ ------ ------------ --------
APP$5313F8AEAFA337B0E05364C909C08D65    1.0                2 NORMAL Y
SALES_APP                                1.0                3 NORMAL N
```

Application container

常见操作： install

```
-- Create new application PDB north_app_pdb

SALES_APP_ROOT>CREATE PLUGGABLE DATABASE north_app_pdb

               ADMIN USER pdb_admin IDENTIFIED BY Password1

               CREATE_FILE_DEST =
'/u01/app/oracle/oradata/orclcdb/sales_app_root/north_app_pdb';

Pluggable database created.
```

Now an application PDB **north_app_pdb** is associated with the application container **sales_app_root.**

```
SALES_APP_ROOT>@get_app_containers

  CON_ID NAME                 OPEN_MODE  APP_ROOT APP_PDB  APP_SEED
-------- -------------------- ---------- -------- -------- --------
         3 SALES_APP_ROOT      READ WRITE YES      NO       NO
         4 NORTH_APP_PDB       MOUNTED    NO       YES      NO
-- Open the application PDB north_app_pdb

SALES_APP_ROOT>ALTER PLUGGABLE DATABASE north_app_pdb OPEN;

               sho pdbs

    CON_ID CON_NAME                   OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
       3 SALES_APP_ROOT         READ WRITE NO
       4 NORTH_APP_PDB          READ WRITE NO
```

```
APP_ROOT>conn sys/oracle@host01:1522/north_app_pdb as sysdba

         set sqlprompt NORTH_APP_PDB>

NORTH_APP_PDB>desc sales_app_user.customers

ERROR:

ORA-04043: object sales_app_user.customers does not exist
```

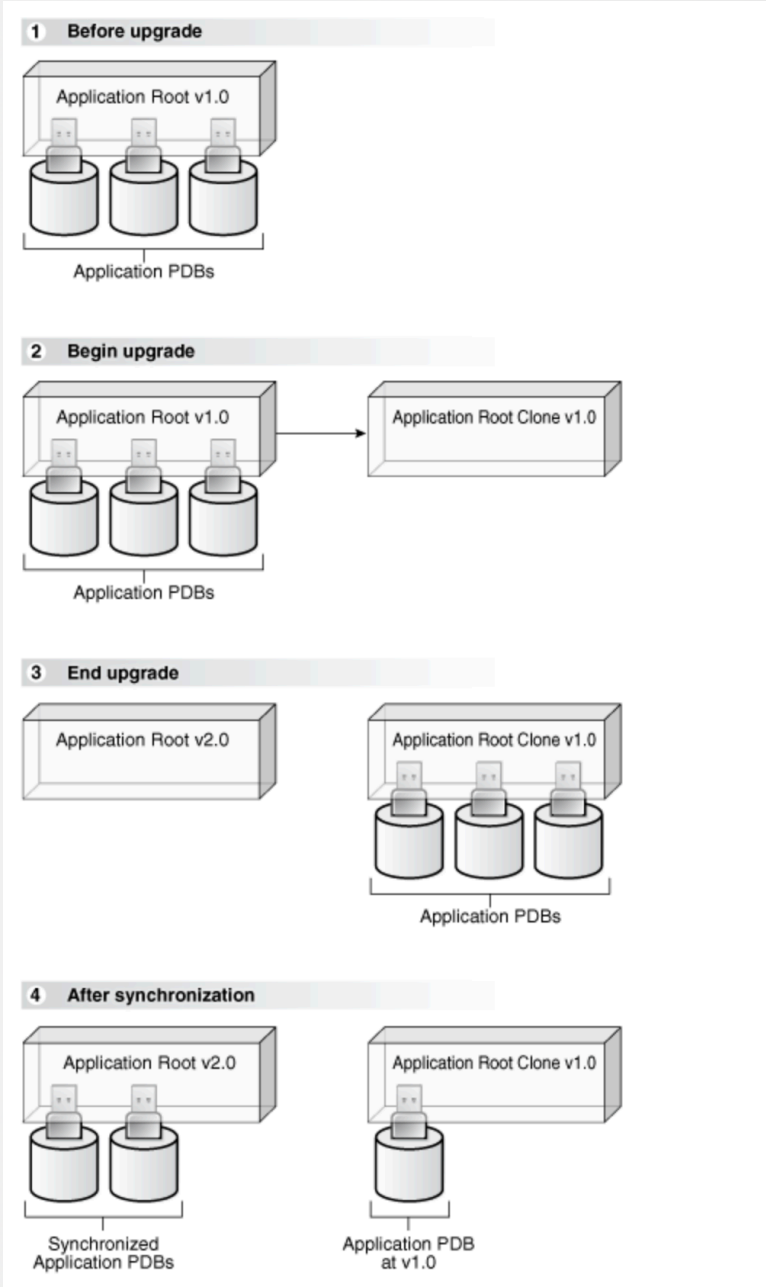
```
NORTH_APP_PDB>ALTER PLUGGABLE DATABASE APPLICATION sales_app SYNC;

Pluggable database altered.

SALES_APP_ROOT>@get_app_pdb_status

NAME                 CON_UID   APP_NAME            APP_VERSIO APP_STATUS
-------------------- ---------- -------------------- ---------- ------------
NORTH_APP_PDB        2659474630 SALES_APP           1.0          NORMAL
NORTH_APP_PDB>desc sales_app_user.customers

 Name                                  Null?      Type

 --------------------------------  --------    -------------------------

 CUST_ID                           NOT NULL    NUMBER

 CUST_NAME                                     VARCHAR2(30)

 CUST_ADD                                      VARCHAR2(30)

 CUST_ZIP                                      NUMBER
```

Application container

常见操作： patch和upgrade

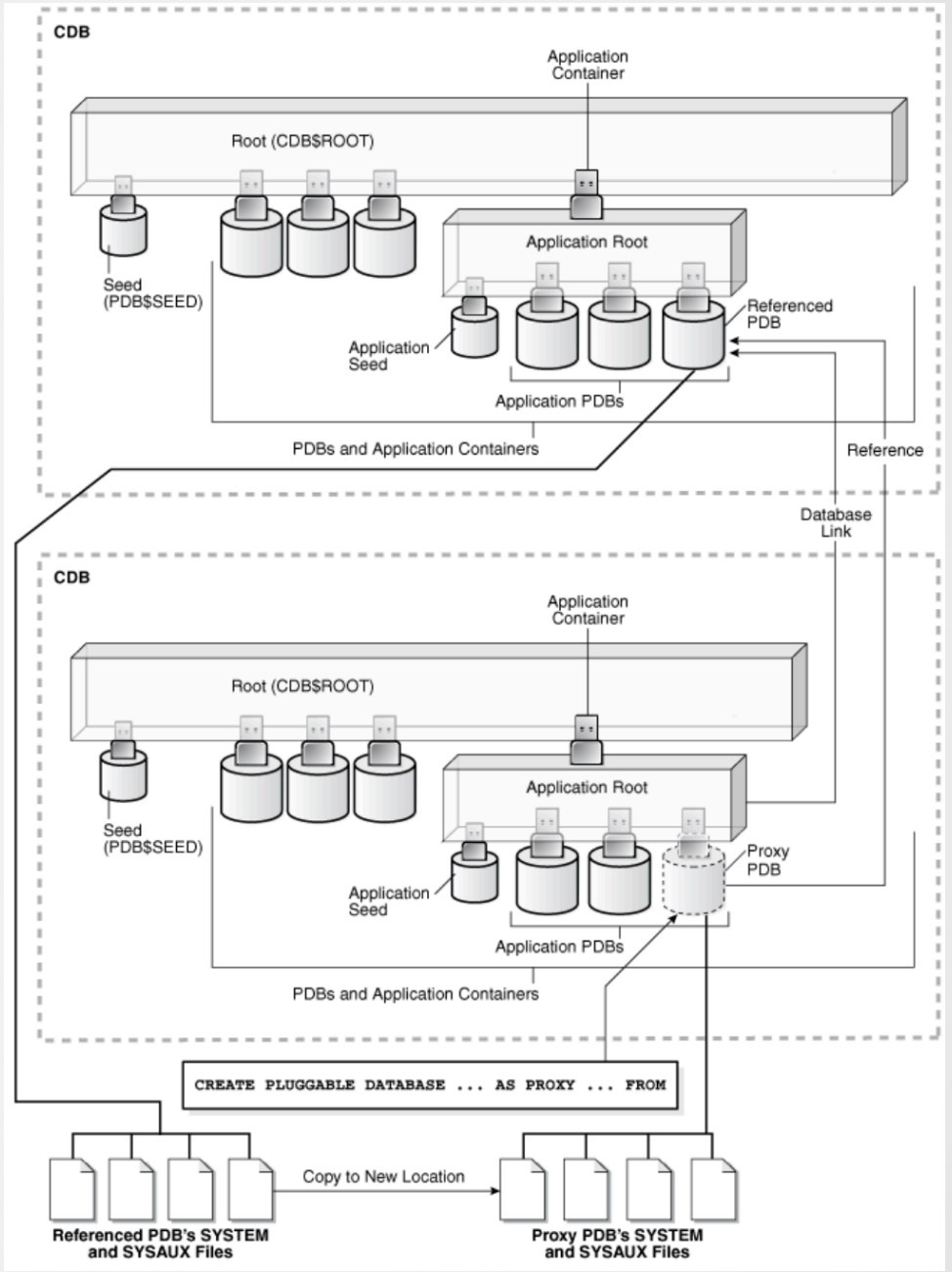
Patch：添加或者更改“Application”中的对象，但是无法删除公共对象。

Upgrade：如果我们想对公共对象进行删除操作，可以使用UPGRADE子句对现有的“Application”进行升级，在升级的过程中，Oracle会首先自动创建Application root的Clone版本，Clone版本创建完成后，Application PDBs将指向Clone版本的Application root，“Application”将继续执行升级操作，升级完成之后，Application root的Clone版本将继续被保留，并且为不同步“Application”的Application PDB提供服务，同步“Application”后的Application PDB将指向升级后的Application root。



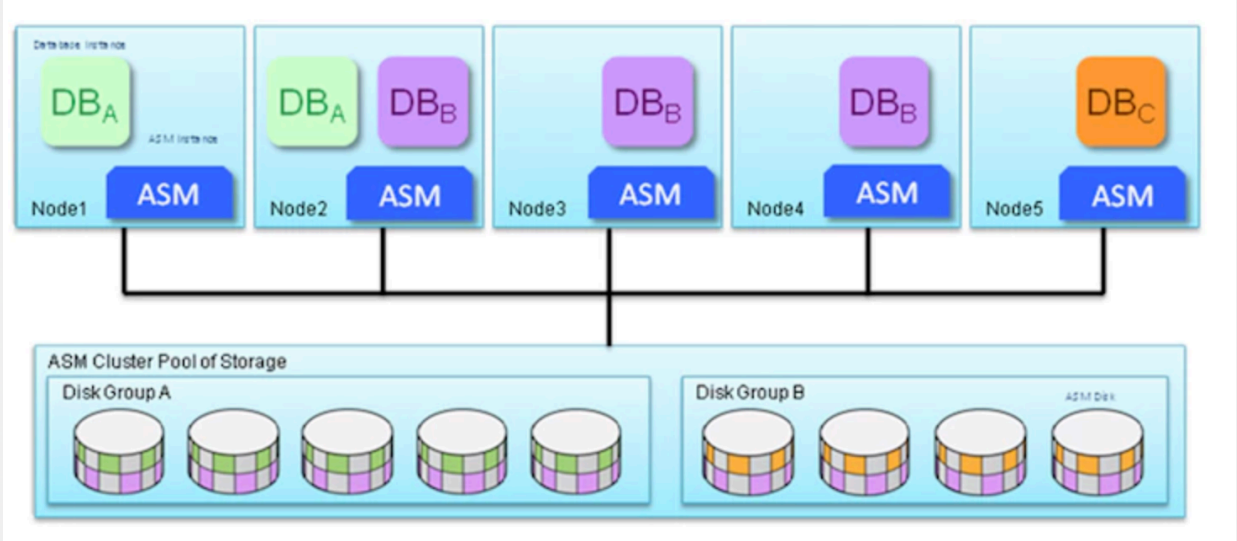
Proxy PDB

- 跨CDB进行访问
- CREATE PLUGGABLE DATABASE …**AS PROXY** … FROM dblink
- 可以结合application container使用
- 注意local undo
- 开始创建是需要dblink，创建后可以删除dblink
- 同步system，sysaux，temp和undo表空间。
- 不同步用户数据，用户数据通过网络传输。
- 注，Flashback operations on a proxy PDB are not supported.

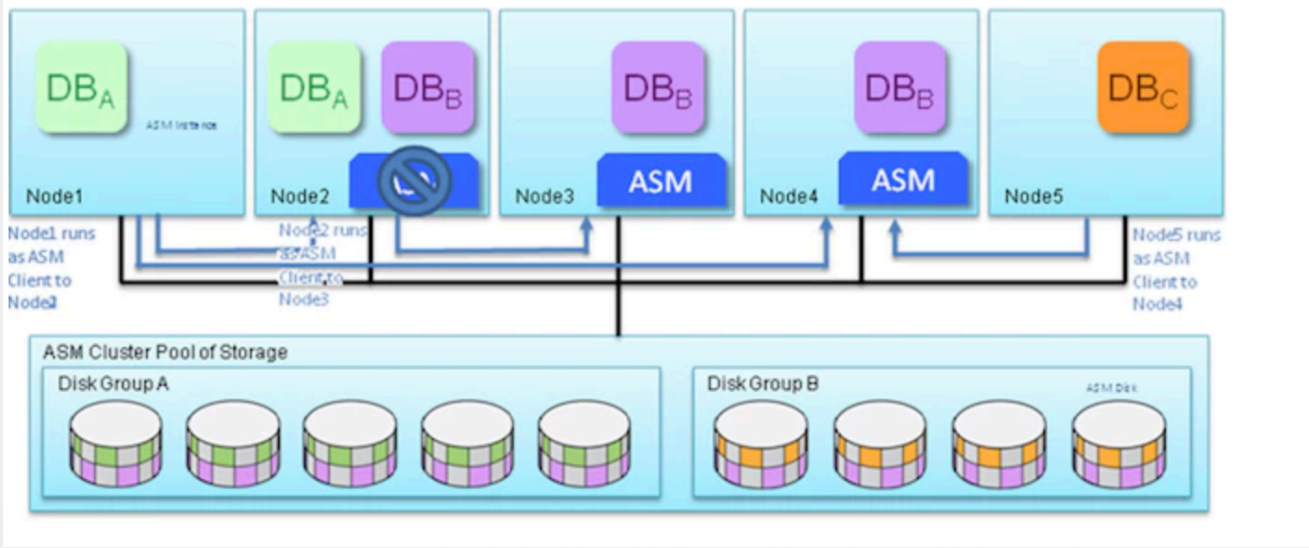


Flex Cluster

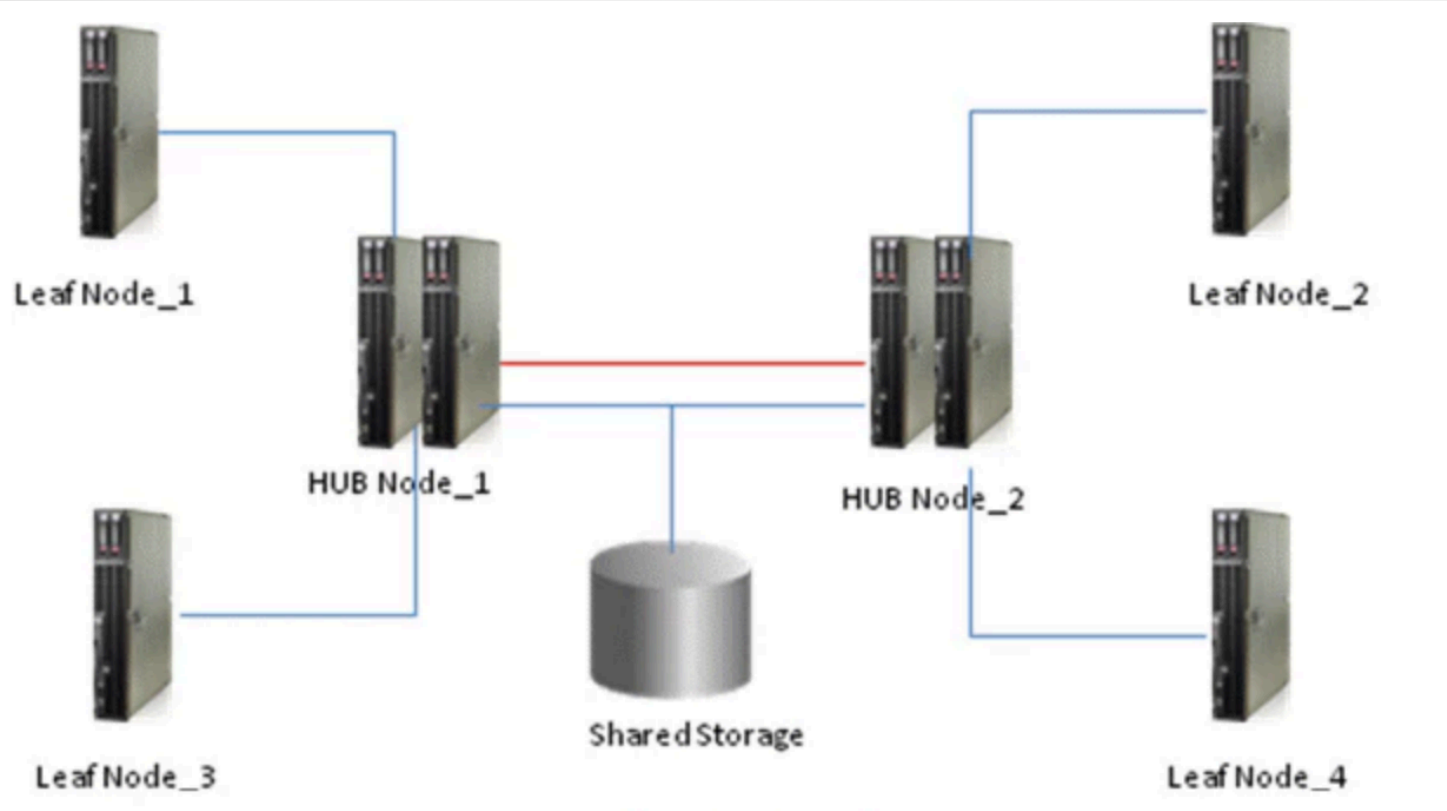
Standard ASM



Flex ASM



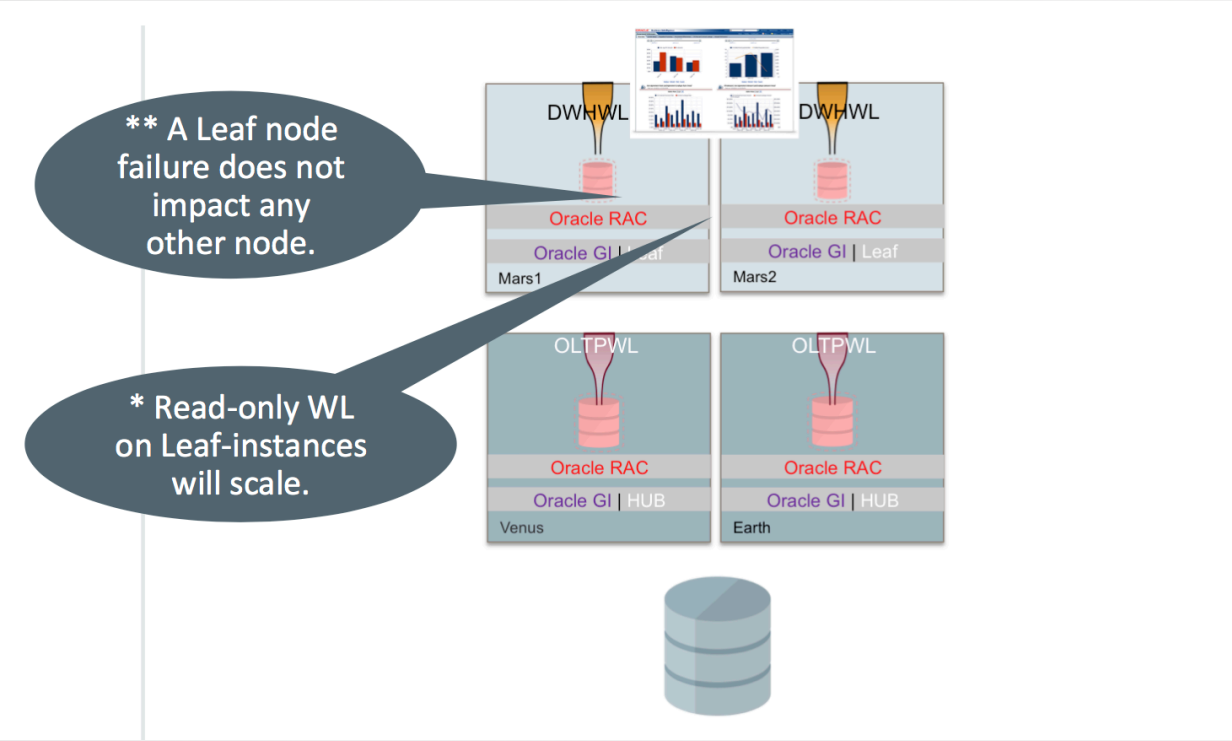
Flex Cluster



- 12.1 leaf node 不能包含数据库实例
- 12.2 leaf node可以包含只读的数据库实例
- 注，角色转换，需要重启该节点crs
- 可以从一般的asm转成flex asm（单向转换，不可逆转）

Flex Cluster

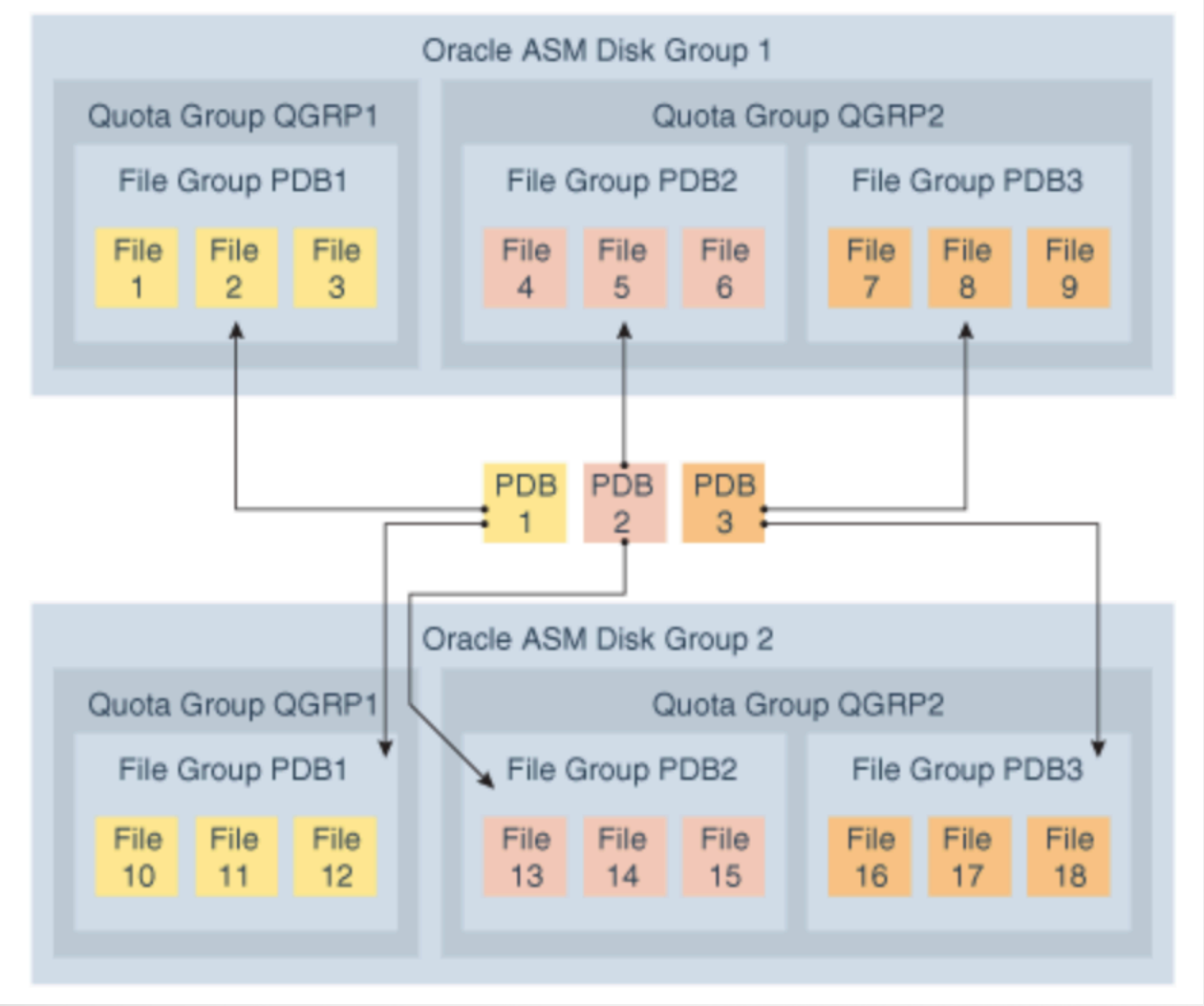
场景：



Use Case : Starting with Oracle RAC 12c Release 2, it is now possible to run a read-only workload on instances running on Leaf Nodes, which in essence are “Reader Nodes.” A Reader Node failure does not impact the overall database activity, making it easy to scale to hundreds of nodes. Customers will use this deployment option to handle Read-Only workloads running on read-mostly Leaf Node instances for ad hoc data analysis. **Read-only work can scale across hundreds of nodes** and does not result in any delay for accessing updated data and does not adversely impact OLTP performance work running on the Hub Nodes. The Leaf nodes supporting a large volume of read activity

ASM增强

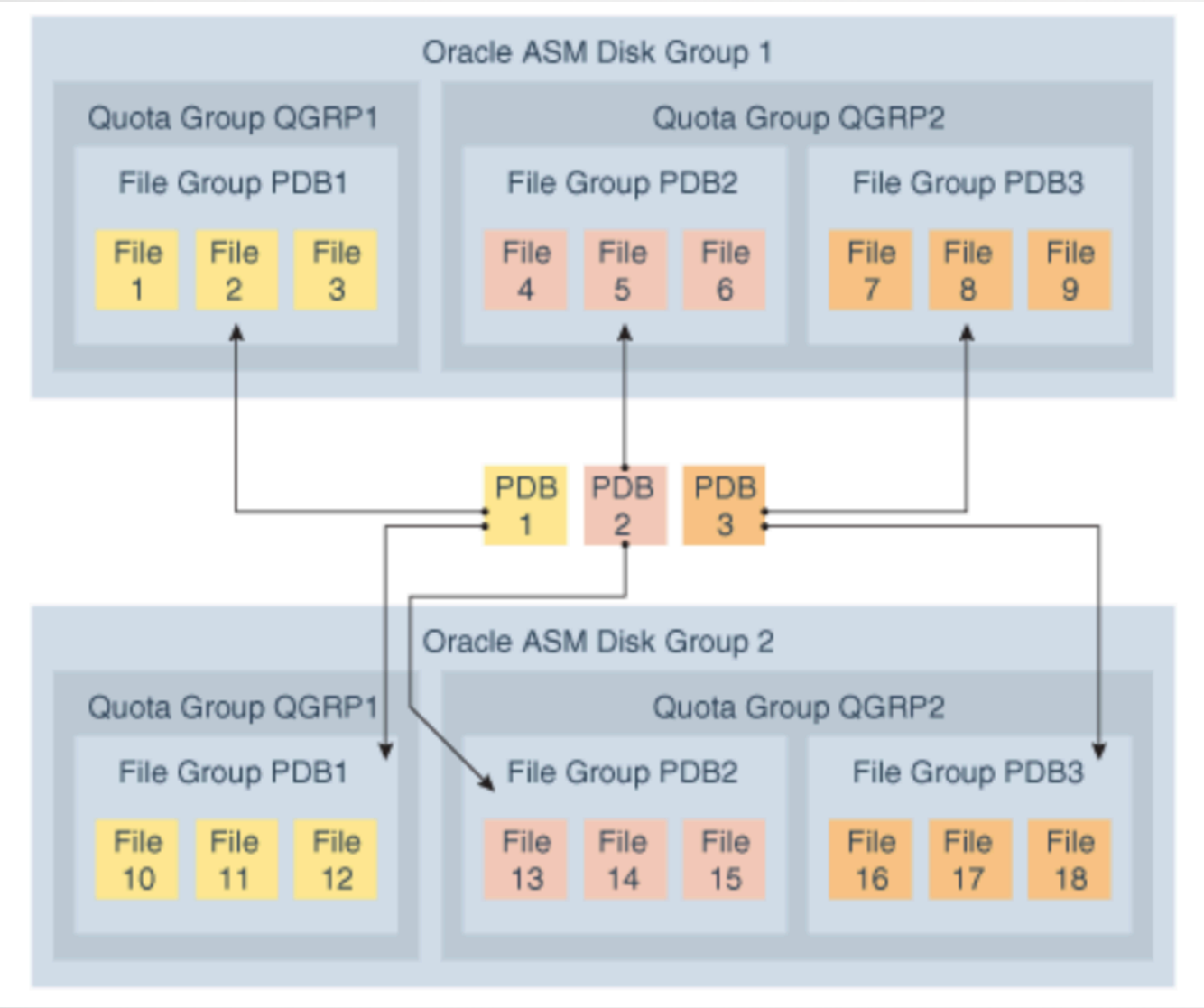
Flex diskgroup：




```
CREATE DISKGROUP xxx FLEX REDUNDANCY DISK disk_path;

ALTER DISKGROUP yyy CONVERT REDUNDANCY TO FLEX;
```

Flex主要体现在redundancy可以灵活。同一个diskgroup内，可以使用不同的redundancy

ASM增强



Quota group：

- 分配给一组asm file group一定大小的配额；
- 在file group上的一个概念。可以多个file group一个quota group，也可以一个file group 一个quota group；

File group:

- 同一冗余程度；
- 同一平衡优先级；
- 按照pdb来划分不同的file group；

ASM增强

AFD（或者叫ASMFD，始于12.1）

1. config

```
[root@vm140 install]# $ORACLE_HOME/bin/asmcmd afd_configure
Connected to an idle instance.
AFD-627: AFD distribution files found.
AFD-636: Installing requested AFD software.
AFD-637: Loading installed AFD drivers.
AFD-9321: Creating udev for AFD.
AFD-9323: Creating module dependencies - this may take some time.
AFD-9154: Loading 'oracleafd.ko' driver.
AFD-649: Verifying AFD devices.
AFD-9156: Detecting control device '/dev/oracleafd/admin'.
AFD-638: AFD installation correctness verified.
Modifying resource dependencies - this may take some time.
ASMCMD-9524: AFD configuration failed 'ERROR: OHASD start failed'
[root@vm140 install]# $ORACLE_HOME/bin/asmcmd afd_state
Connected to an idle instance.
ASMCMD-9526: The AFD state is 'LOADED' and filtering is 'DISABLED' on host 'vm140'
```

2. label

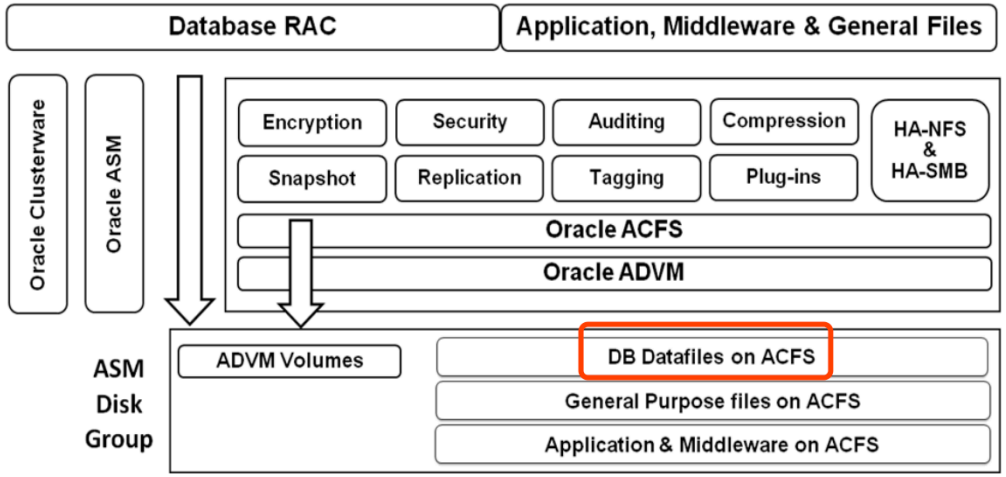
```
[root@vm140 install]# $ORACLE_HOME/bin/asmcmd afd_label GI /dev/xvdb1
Connected to an idle instance.
[root@vm140 install]# $ORACLE_HOME/bin/asmcmd afd_label DATA /dev/xvdc1
Connected to an idle instance.
[root@vm140 install]# $ORACLE_HOME/bin/asmcmd afd_label FRA /dev/xvdd1
Connected to an idle instance.

[root@vm140 install]# $ORACLE_HOME/bin/asmcmd afd_lsdsk
Connected to an idle instance.
--------------------------------------------------------------------------------
Label                     Filtering   Path
================================================================================
GI                         DISABLED   /dev/xvdb1
DATA                       DISABLED   /dev/xvdc1
FRA                        DISABLED   /dev/xvdd1
```

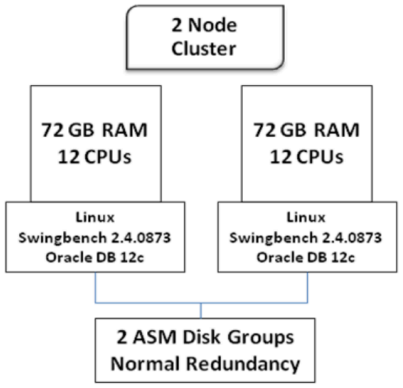
作用：

IO Filter，即使root权限dd整个asm磁盘，也不会导致asm磁盘失效！

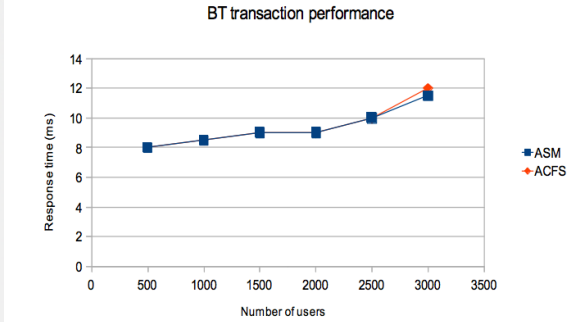
ASM增强 ACFS

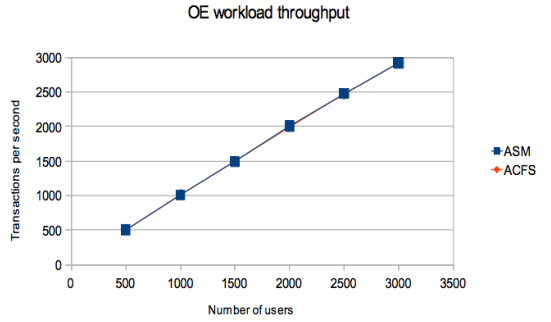
从12.1开始支持数据文件

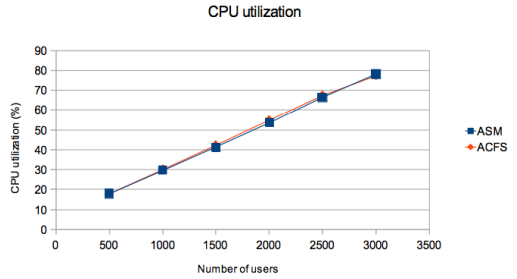


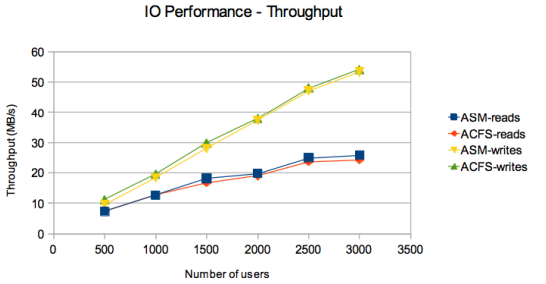


- One disk group for data and one for recovery
- Data:
  - 80 Physical disks configured as RAID10 (5 shelves)
  - 120 disks for DSS (5 shelves)
- Recovery:
  - 16 Physical disks configured as RAID10 (1 shelf).
- Disk groups with enough space to accommodate 2 databases:
  - One Database uses ACFS the other uses ASM directly
  - One ACFS file system create on each disk group
- Database:
  - Two identical database configured
  - Two database instances per database
  - Archivelog and flashback database enabled (disabled for DSS).









PERFORMANCE NUMBERS

	ASM	ACFS	% difference
SH Throughput (TX/min)	3.29	3.26	-1%
IO Throughput (Reads / sec)	3501	3312	-5%
IO Throughput (MB Read / sec)	724	712	-2%
CPU Usage (%)	16.5 %	16.7 %	+1%

ASM增强 ACFS

- 支持snapshot remast，将旧的filesystem和snapshot删除
  - ✓ The previous Oracle ACFS primary file system and all previous snapshots are deleted.

- Snapshot rename改名
  - ✓ This renames an existing ACFS snapshot.

- acfsutil snap delete， 支持force delete，即使file 还是open的，snapshot还是一样能被删除。
  - ✓ It is a force snapshot delete that enables a snapshot with open file references to be removed from an Oracle ACFS.

- 压缩增强
  - ✓ acfsutil compress on
  - ✓ Redo logs, flashback logs, and control files不压缩
  - ✓ 新开启压缩，则老文件不压缩
  - ✓ 新禁用压缩，则老文件不解压缩
  - ✓ database file，压缩单位是db block size
  - ✓ 非 datbase file，压缩单位是32kb

ASM增强 ACFS

- 性能增强 —Accelerator Volume 存放metadata

- 基于snapshot的replication
  - ✓ acfsutil snap duplicate create 和 acfsutil snap duplicate apply —— 从一个snapshot创建另一个snapshot并且进行跟踪其变化，apply其变化

- Snapshot level quota
  - ✓ Limit the new storage that can be allocated for a given snapshot. This provides a quota on space usage for each snapshot.

- auto resize 增强
  - ✓ This feature enables you to specify file system size quotas and have the file system automatically increase in size, if required.

- acfsutil defrag dir/file — ACFS碎片整理（存放OLTP数据库的话）

- 支持4k sector size的文件系统metadata block

ASM增强

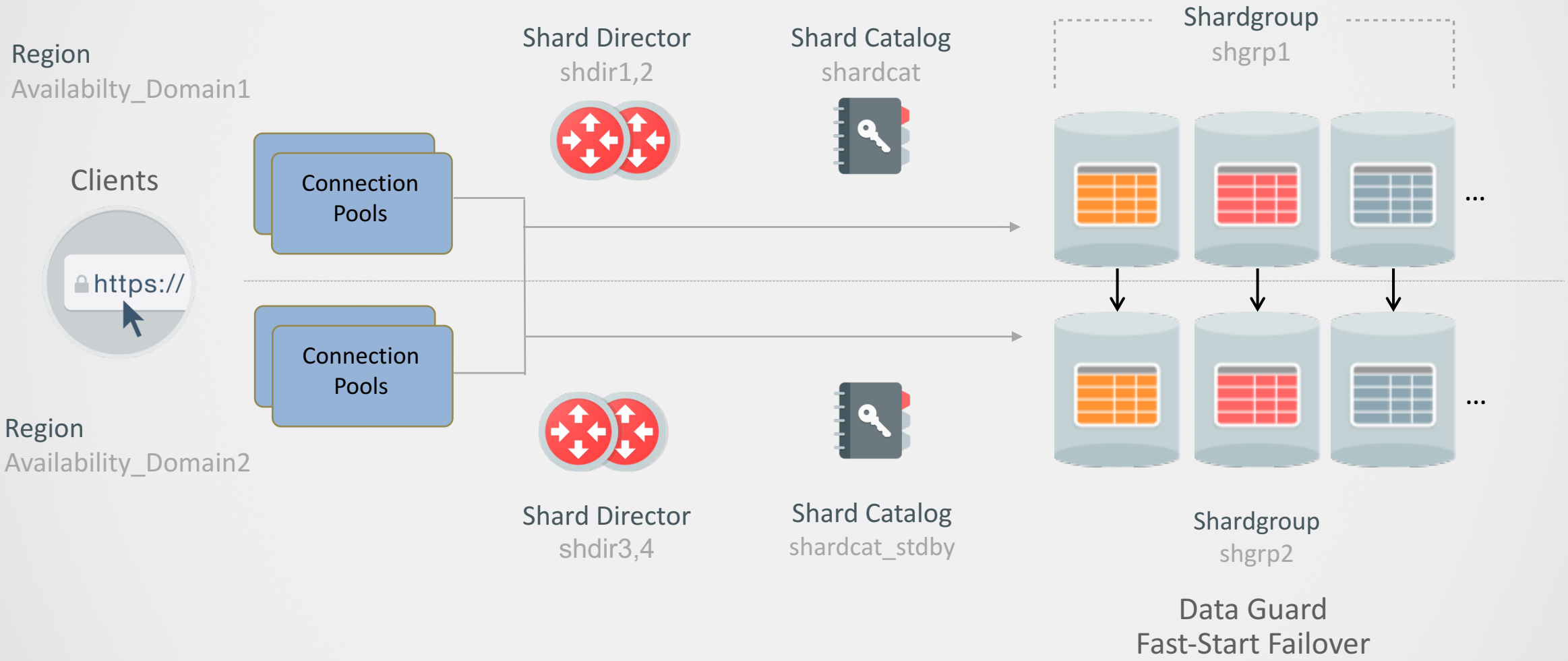
其他：

- ASM默认memory target 大于1G（_asm_allow_small_memory_target）

- Prioritized rebalancing，优先级高的file group先完成rebalance

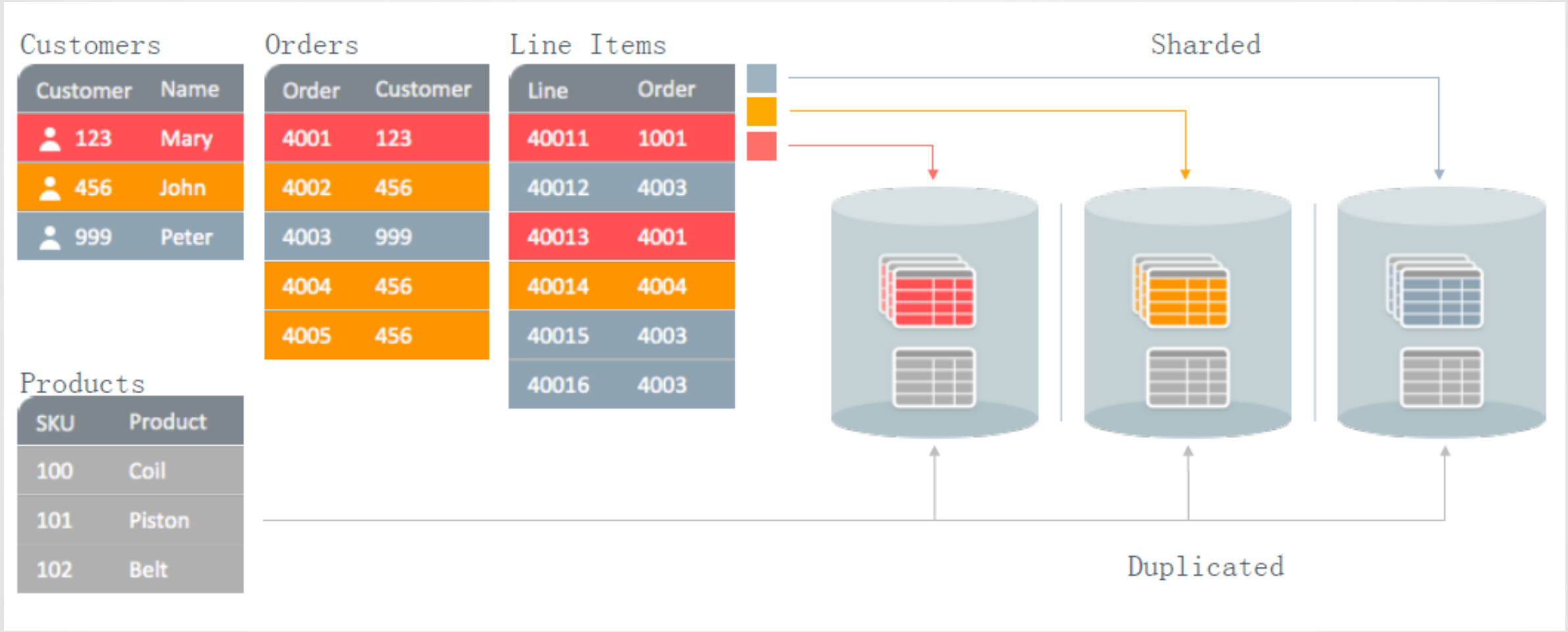
  级别：
  - HIGHEST
  - HIGH
  - MEDIUM（默认）
  - LOW
  - LOWEST

Sharding Database



Sharding Database

分片表的分类



Oracle分片方法

- System Managed分片
  - by Consistent Hash
    - Range of hash values assigned to each chunk

```
CREATE SHARDED TABLE customers
( cust_id     NUMBER NOT NULL
, name        VARCHAR2(50)
, address     VARCHAR2(250)
, location_id VARCHAR2(20)
, class       VARCHAR2(3)
, signup      DATE
, CONSTRAINT cust_pk PRIMARY KEY(cust_id)
)
PARTITION BY CONSISTENT HASH (cust_id)
PARTITIONS AUTO
TABLESPACE SET ts1
;
```

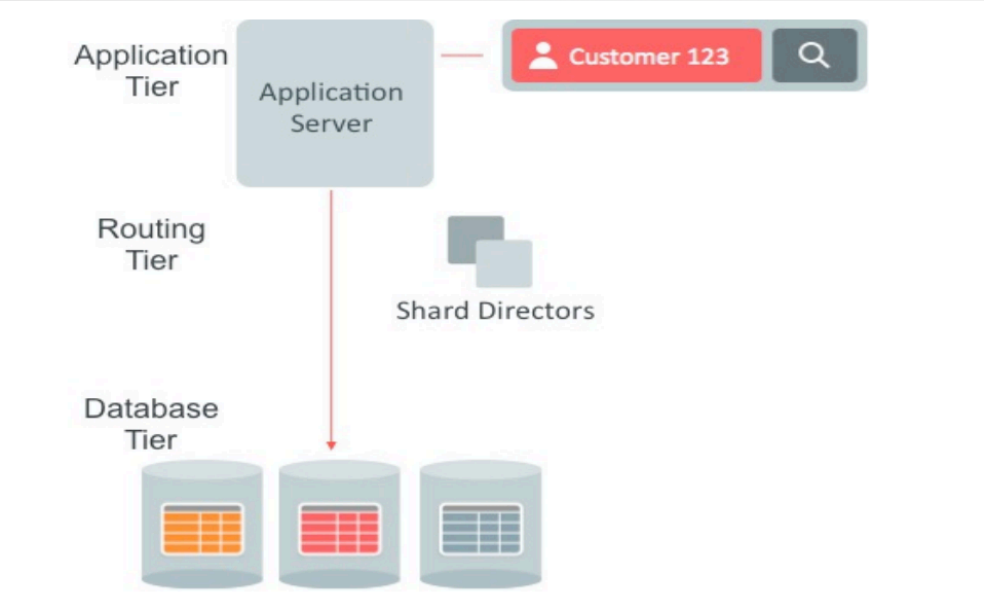
- Composite 分片
  - by Range - Consistent Hash or by List - Consistent Hash
    - Two-level sharding, uses two keys

```
CREATE SHARDED TABLE customers
( cust_id NUMBER NOT NULL
, name VARCHAR2(50)
, address VARCHAR2(250)
, location_id VARCHAR2(20)
, class VARCHAR2(3)
, signup_date DATE
, CONSTRAINT cust_pk PRIMARY KEY(class, cust_id)
)
PARTITIONSET BY LIST (class)
  PARTITION BY CONSISTENT HASH (cust_id)
  PARTITIONS AUTO
(PARTITIONSET gold VALUES ('gld') TABLESPACE SET tbs1,
 PARTITIONSET silver VALUES ('slv') TABLESPACE SET tbs2)
;
```

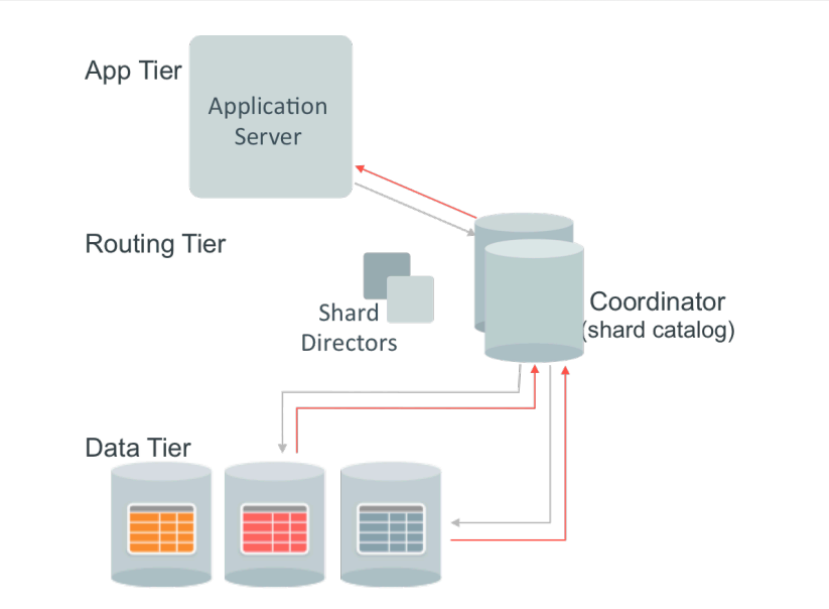
Sharding Database

路由方式

- Direct routing

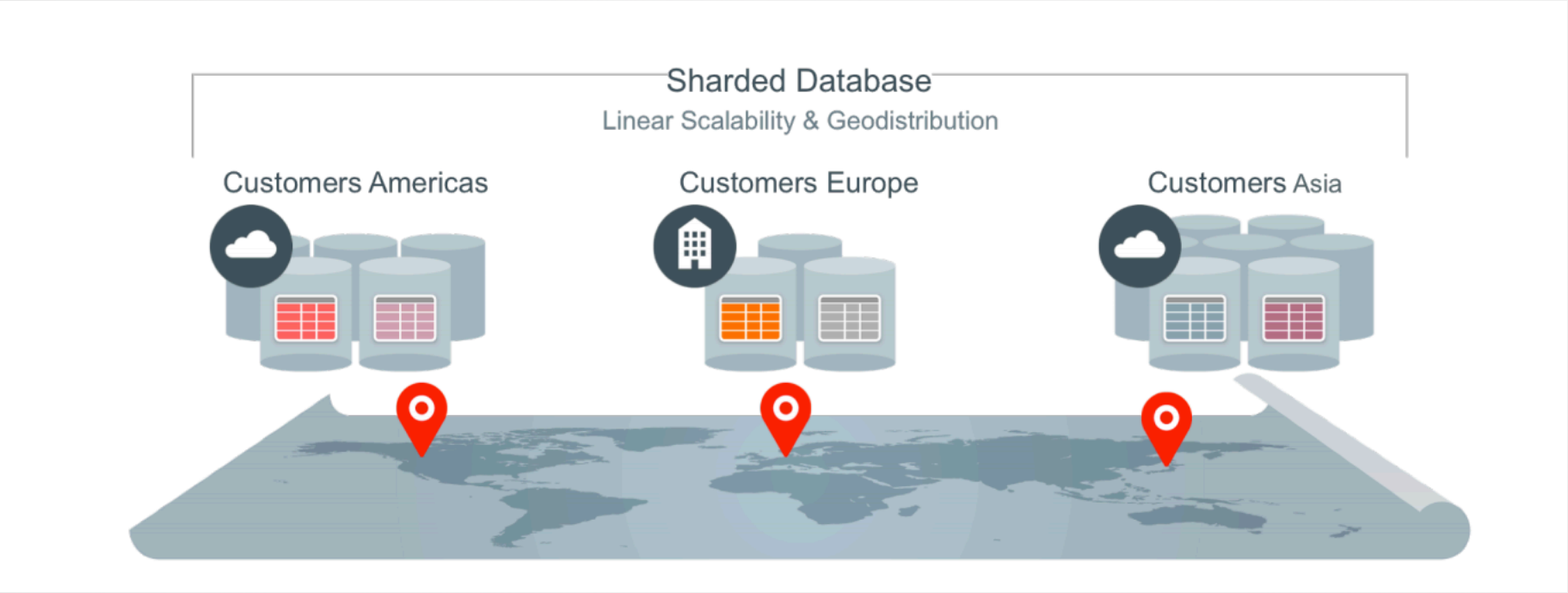


- Proxy routing



Sharding Database

场景：



OFS server

- 利用dbfs
- OFS，一种新的database file system
- 步骤：

1. yum install kernel-devel fuse fuse-libs
2. 修改/etc/fuse.conf，取消user_allow_other的注释
3. chmod +x /usr/bin/fusermountreboot

OFS server

- 步骤：
  4. 选择file system的方式（如果只能pdb级别，选择dbfs）
     - OFS：

```
mkdir -p /u01/ofs/ofs_fs1
CREATE TABLESPACE ofs_ts DATAFILE SIZE 500M AUTOEXTEND ON NEXT 1M;
BEGIN
  DBMS_FS.make_oracle_fs (
    fstype    => 'ofs',
    fsname    => 'ofs_fs1',
    fsoptions => 'TABLESPACE=ofs_ts');
END;
/
BEGIN
  DBMS_FS.make_oracle_fs (
    fstype    => 'ofs',
    fsname    => 'ofs_fs1',
    fsoptions => 'TABLESPACE=ofs_ts');
END;
/
```

     - dbfs

```
mkdir -p /u01/dbfs/dbfs_fs1
@dbfs_create_filesystem.sql dbfs_ts dbfs_fs1
BEGIN
  DBMS_FS.mount_oracle_fs (
    fstype           => 'dbfs',
    fsname           => 'dbfs_fs1',
    mount_point      => '/u01/dbfs/dbfs_fs1',
    mount_options    => 'default_permissions, allow_other, persist'
  );
END;
/
```

OFS server

- 步骤：

5. yum install nfs-utils –y
6. systemctl start nfs

   systemctl enable nfs
7. exportfs –ra

   showmount –e
8. mkdir -p /mnt/ofs_fs1

   mount ol7-122.localdomain:/u01/ofs/ofs_fs1 /mnt/ofs_fs1 -t nfs -o vers=3

其他

- 利用逻辑standby进行滚动升级增强
- OGG需要大于12.3才能支持database 12.2

1 管理篇

管理篇

1 CPB/PDB新特性

2 工具的增强

3 Online operation增强

4 In Memory增强

5 ADG增强

6 SPA和STS增强

7 安全增强

8 其他

CPB/PDB新特性

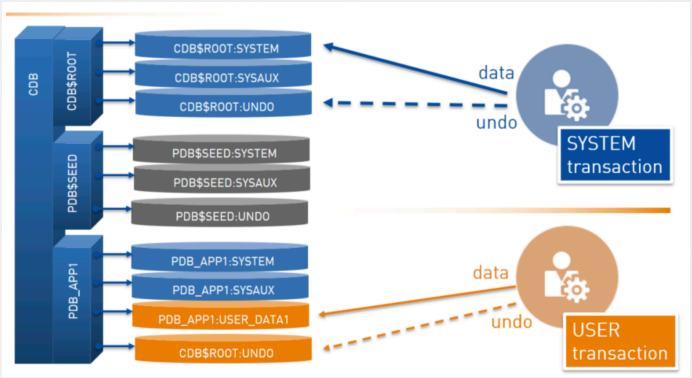
- **IO限制in PDB**
  - **MAX_IOPS： PDB每秒最大的I/O操作次数。默认是0，不推荐设置MAX_IOPS的值小于100.**
  - **MAX_MBPS： PDB每秒最大的I/O带宽（MB）。默认是0，不推荐设置MAX_MBPS的值小于 25 MBPS。**

- **ILM增强**
  - Heatmap支持PDB： PDB级别， ALTER SYSTEM SET heat_map = ON; 通过dba_heat_map_seg_histogram查看结果。
  - ADO策略 in PDB ， 通过user_ilmobjects查看结果。

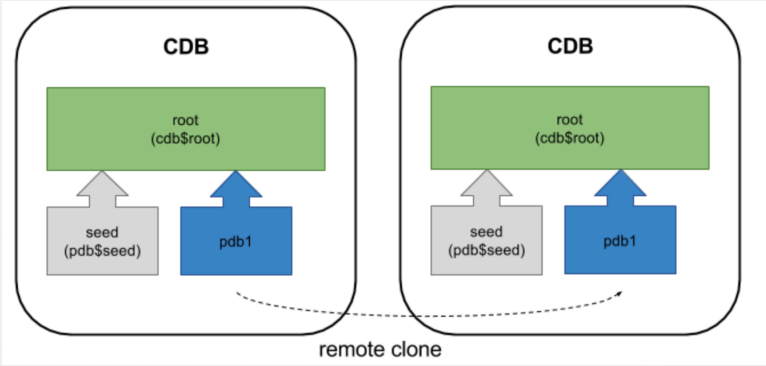
```
CONN test/test@pdb1

DROP TABLE invoices PURGE;

CREATE TABLE invoices (
  invoice_no    NUMBER NOT NULL,
  invoice_date  DATE   NOT NULL,
  comments      VARCHAR2(500)
)
PARTITION BY RANGE (invoice_date)
(
  PARTITION invoices_2016_q1 VALUES LESS THAN (TO_DATE('01/04/2016', 'DD/MM/YYYY')) TABLESPACE slow_storage_ts,
  PARTITION invoices_2016_q2 VALUES LESS THAN (TO_DATE('01/07/2016', 'DD/MM/YYYY')) TABLESPACE slow_storage_ts,
  PARTITION invoices_2016_q3 VALUES LESS THAN (TO_DATE('01/09/2016', 'DD/MM/YYYY')) TABLESPACE medium_storage_ts
    ILM ADD POLICY TIER TO slow_storage_ts READ ONLY SEGMENT AFTER 6 MONTHS OF NO ACCESS,
  PARTITION invoices_2016_q4 VALUES LESS THAN (TO_DATE('01/01/2017', 'DD/MM/YYYY')) TABLESPACE medium_storage_ts
    ILM ADD POLICY TIER TO slow_storage_ts READ ONLY SEGMENT AFTER 6 MONTHS OF NO ACCESS,
  PARTITION invoices_2017_q1 VALUES LESS THAN (TO_DATE('01/04/2017', 'DD/MM/YYYY')) TABLESPACE fast_storage_ts
    ILM ADD POLICY TIER TO medium_storage_ts READ ONLY SEGMENT AFTER 3 MONTHS OF NO ACCESS,
  PARTITION invoices_2017_q2 VALUES LESS THAN (TO_DATE('01/07/2017', 'DD/MM/YYYY')) TABLESPACE fast_storage_ts
    ILM ADD POLICY TIER TO medium_storage_ts READ ONLY SEGMENT AFTER 3 MONTHS OF NO ACCESS
)
ILM ADD POLICY ROW STORE COMPRESS BASIC SEGMENT AFTER 3 MONTHS OF NO ACCESS;
```

CPB/PDB新特性

- **独立UNDO in PDB**
  - 便于flashback pdb，无须close pdb再create clean restore point
  - I/O 隔离，减少对UNDO tablespace的争用

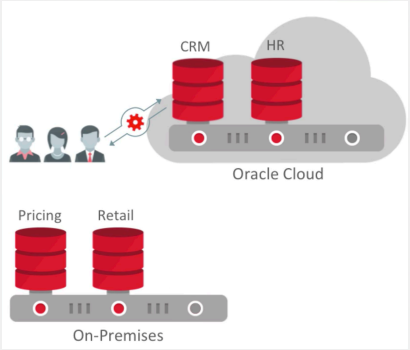


- **PDB hot clone**
  - 可以在不同pdb之间存在不同的字符集（前提：cdb是al32utf8）
  - 同一主机不同CBD之间迁移pdb:CREATE PLUGGABLE DATABASE FROM RELOCATE

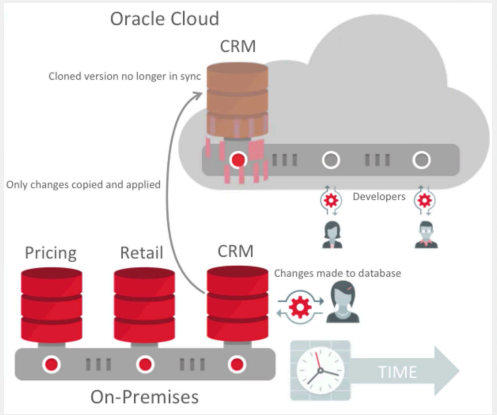


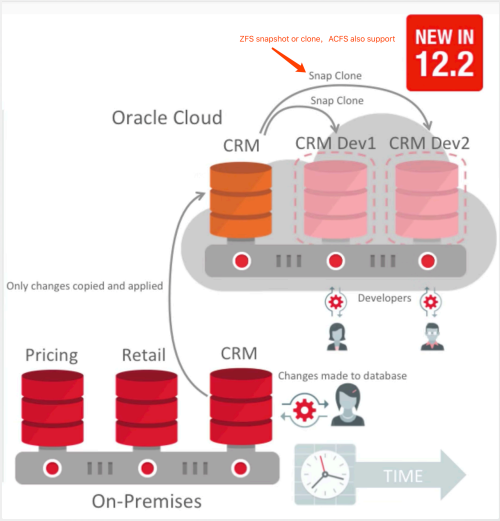
CPB/PDB新特性

- **PDB relocate**
  - 适合迁移到别的CDB，完全online，云上云下迁移



- **PDB refresh**
  - 在别的CDB中建立只读库，穷人的ADG
  - CREATE PLUGGABLE DATABASE pdb5_ro FROM pdb5@clone_link REFRESH MODE MANUAL;
    - refresh mode
      - ✓ MANUAL
      - ✓ EVERY 60 MINUTES
      - ✓ NONE
  - ALTER SESSION SET CONTAINER=pdb5_ro;
  - ALTER PLUGGABLE DATABASE CLOSE IMMEDIATE;
  - ALTER PLUGGABLE DATABASE REFRESH;
  - ALTER PLUGGABLE DATABASE OPEN READ ONLY;





CPB/PDB新特性

- **PGA 限制in PDB**
  - PDB级别参数
    - ✓ PGA_AGGREGATE_LIMIT
    - ✓ PGA_AGGREGATE_TARGET
  - 利用resource manager

```
CONN / AS SYSDBA
ALTER SESSION SET CONTAINER=pdb1;

BEGIN
  DBMS_RESOURCE_MANAGER.clear_pending_area();
  DBMS_RESOURCE_MANAGER.create_pending_area();

  -- Create plan
  DBMS_RESOURCE_MANAGER.create_plan(
    plan    => 'pga_plan',
    comment => 'Plan for a combination of high and low PGA usage.');

  -- Create consumer groups
  DBMS_RESOURCE_MANAGER.create_consumer_group(
    consumer_group => 'high_pga_cg',
    comment        => 'High PGA usage allowed');

  -- Assign consumer groups to plan and define priorities
  DBMS_RESOURCE_MANAGER.create_plan_directive (
    plan               => 'pga_plan',
    group_or_subplan   => 'high_pga_cg',
    session_pga_limit => 100);

  DBMS_RESOURCE_MANAGER.validate_pending_area;
  DBMS_RESOURCE_MANAGER.submit_pending_area();
END;
/
```

CPB/PDB新特性

- **单个CDB支持的pdb个数**
  - 12.1 ： 252
  - 12.2 参数max_pdbs，取值范围0~4098
- **flashback in PDB**
  - 大前提（即不是PDB flashback特有）：
    - ✓ archive模式
    - ✓ 启用flashback area
    - ✓ flashback log被打开
    - ✓ flashback log在flashback area，OMF管理
    - ✓ 如果重建或者restore控制文件，那么无法flashback到重建或restore控制文件的时间点之前
  - 前提
    - ✓ compatible>=12.2.0.0
    - ✓ rman需要以sysdba或者sysbackup权限连接
    - ✓ 需要flashback的pdb必须close状态，其他pdb可以open状态
    - ✓ root 必须open，当pdb open resetlog的时候

CPB/PDB新特性

PDB lockdown profile

用于限制pdb级别进行类似alter system的操作

- 选项（Option）

```
ALTER LOCKDOWN PROFILE my_profile ENABLE OPTION ALL;（默认）

ALTER LOCKDOWN PROFILE my_profile ENABLE OPTION = ('DATABASE QUEUING');

ALTER LOCKDOWN PROFILE my_profile ENABLE OPTION = ('PARTITIONING');

ALTER LOCKDOWN PROFILE my_profile ENABLE OPTION ALL EXCEPT = ('PARTITIONING');
```

- 特性（Feature）

```
ALTER LOCKDOWN PROFILE my_profile ENABLE FEATURE ALL;（默认）

ALTER LOCKDOWN PROFILE my_profile ENABLE  FEATURE = ('UTL_HTTP');

ALTER LOCKDOWN PROFILE my_profile ENABLE  FEATURE = ('NETWORK_ACCESS');

ALTER LOCKDOWN PROFILE my_profile ENABLE FEATURE ALL EXCEPT = ('NETWORK_ACCESS');
```

- 语句（Statement）

```
ALTER LOCKDOWN PROFILE my_profile ENABLE STATEMENT = ('ALTER DATABASE', 'ALTER PLUGGABLE DATABASE');

ALTER LOCKDOWN PROFILE my_profile DISABLE STATEMENT = (‘ALTER PLUGGABLE DATABASE’) CLAUSE = ('DEFAULT TABLESPACE', 'DEFAULT TEMPORARY TABLESPACE');

ALTER LOCKDOWN PROFILE my_profile DISABLE STATEMENT = (‘ALTER SYSTEM’) CLAUSE ALL EXCEPT = ('FLUSH SHARED_POOL');

ALTER LOCKDOWN PROFILE my_profile DISABLE STATEMENT = (‘ALTER SYSTEM’) CLAUSE = ('SET') OPTION = ('CPU_COUNT') MINVALUE = '1' MAXVALUE = '3';
```

CPB/PDB新特性

- 支持pdb级别AWR

```
SQL> @?/rdbms/admin/awrrpt.sql

Specify the Report Type
~~~~~~~~~~~~~~~~~~~~~~~
AWR reports can be generated in the following formats.  Please enter the
name of the format at the prompt.  Default value is 'html'.

'html'          HTML format (default)
'text'          Text format
'active-html'   Includes Performance Hub active report

Enter value for report_type: html
old   1: select 'Type Specified: ',lower(nvl('&&report_type','html')) report_type from dual
new   1: select 'Type Specified: ',lower(nvl('html','html')) report_type from dual

Type Specified:  html

Specify the location of AWR Data
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
AWR_ROOT - Use AWR data from root (default)
AWR_PDB - Use AWR data from PDB
```

WORKLOAD REPOSITORY PDB report (PDB snapshots)

DB Name	DB Id	Unique Name	Role	Edition	Release	RAC	CDB
ORA122	631936275	ora122	PRIMARY	EE	12.2.0.1.0	NO	YES

Instance	Inst Num	Startup Time
ora122	1	06-Mar-17 18:03

Container DB Id	Container Name	Open Time
631936275	TEST1	06-Mar-17 18:03

Host Name	Platform	CPUs	Cores	Sockets	Memory (GB)
isrvmrh541.prod.quest.corp	Linux x86 64-bit	4	4	1	3.74

	Snap Id	Snap Time	Sessions	Cursors/Session
Begin Snap:	1	06-Mar-17 21:20:00	0	3.0
End Snap:	2	06-Mar-17 21:20:10	0	7.0
Elapsed:		0.16 (mins)		
DB Time:		0.06 (mins)		

工具的增强

- **DATAPUMP**
  - 导入metadata也支持parallel参数
  - 导出metadata也支持parallel参数
  - %L参数，比%U参数更好，从2位到10位，即原来的dump文件名为dmp%U为dmp01～dmp99，现在是dmp01～dmp2147483646

online operation

- **在线从定义dbms_redefinition**
  - 新增ROLLBACK参数
  - 通过V$online_redef观察从定义的进度
  - 支持bfile列
  - 注，非分区表到分区表，可以通过alter table modify online实现，不再需要dbms_redefinition
- **online TDE**
  - 在12.2之前，如果对表空间进行透明数据加密，这是需要停机时间的，可参考 Oracle Advanced Security 透明数据加密最佳实践，但是在12.2中，我们可以不用停机的进行TDE加密了。
  - alter tablespace tbs_test encryption online using 'AES192' encrypt ;
- **online DDL**
  - alter table move online

online operation

- **online DDL**
  - alter table modify online

```
ALTER TABLE employees_convert MODIFY
  PARTITION BY RANGE (employee_id) INTERVAL (100)
  ( PARTITION P1 VALUES LESS THAN (100),
    PARTITION P2 VALUES LESS THAN (500)
   ) ONLINE
  UPDATE INDEXES
 ( IDX1_SALARY LOCAL,
   IDX2_EMP_ID GLOBAL PARTITION BY RANGE (employee_id)
  ( PARTITION IP1 VALUES LESS THAN (MAXVALUE))
 );
```

  - alter table split/merge patition online

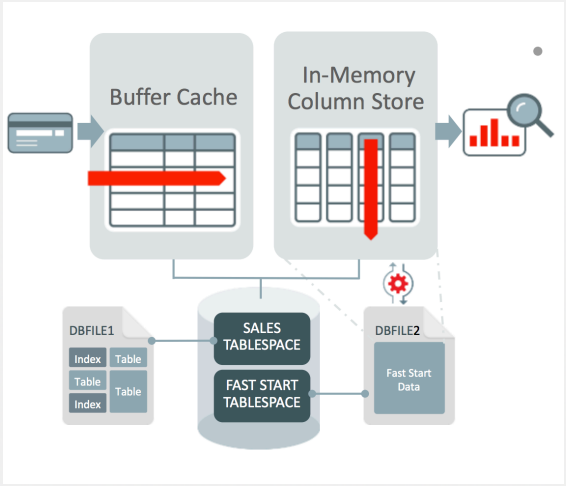
注意，分区维护过滤（Filtered Partition Maintenance Operations）

```
ALTER TABLE T_ORACLEBLOG MOVE PARTITION p4
TABLESPACE SYSAUX
INCLUDING ROWS WHERE REGION = 'CHINA' --Note keyword INCLUDING ROW WHERE;
```

  - 为交换分区做准备的表，可以直接通过create table FOR EXCHANGE WITH TABLE命令生成。

In memory增强

- **In-memory json column**
  - 注1，json文档需要小于32k
  - 注2，使用json_textcontains做全文检索的，不被IM column store支持
  - 注3，初始化参数max_string_size需要设置成'extended’
  - 注4，存储json数据的列，必须有is json的check约束
- **ADO heatmap**
  - 支持冷热数据的分离，不仅仅在存储层面，也在是否in memory层面。
- **支持adg**
  - **支持备库的in memory查询**
- **in memory faststart**
  - **解决快速加载问题**



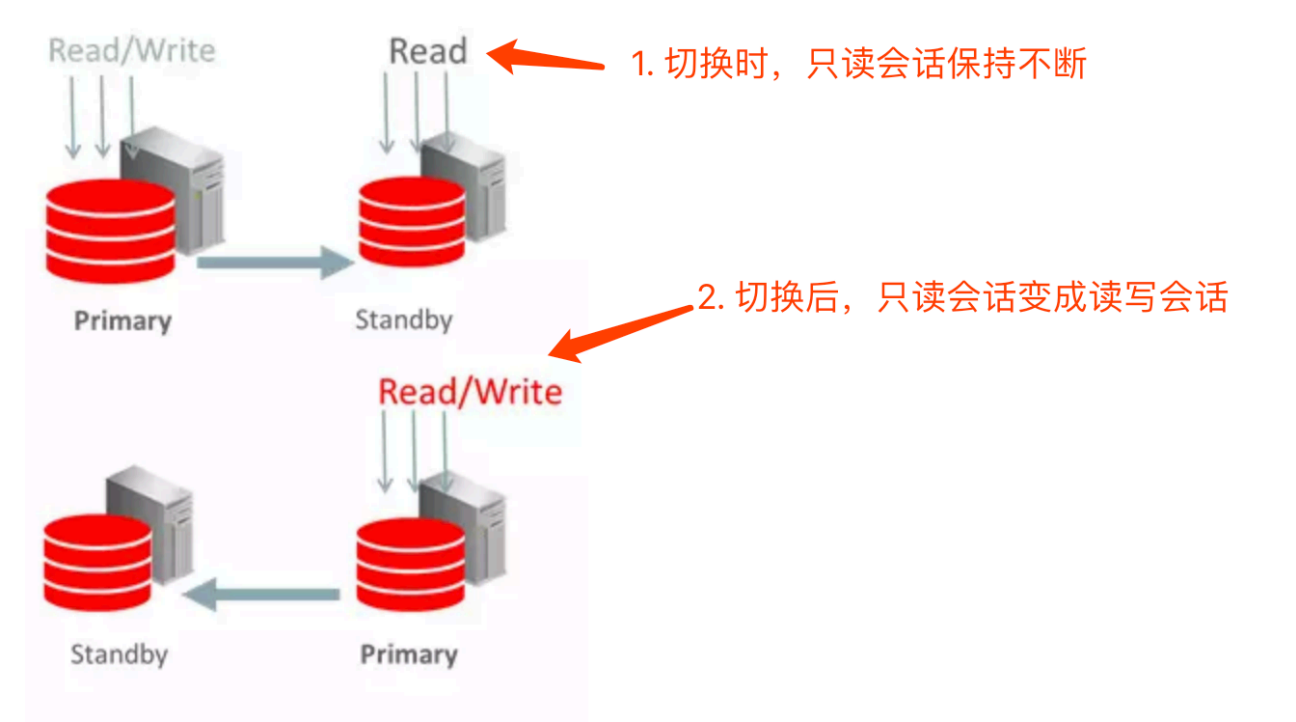
ADG增强

数据保护增强

- 在主库上的nonlogging的block，rman在备库上进行校验和修复
  - RMAN > RECOVER DATABASE NONLOGGED BLOCK
- 增强的自动块修复
- 即使在异步模式存储故障时，数据零丢失（far-sync）
- 主备数据库高速对比 DBMS_DBCOMP.DBCOMP
  - 进度：v$session_longops
  - 可以在主库，也可以在备库执行 可以在主库，但是非全部备库启动；
  - 可以在主库，但是非全部备库启动；也可以在某个备库，但是主库启动；非全部备库启动

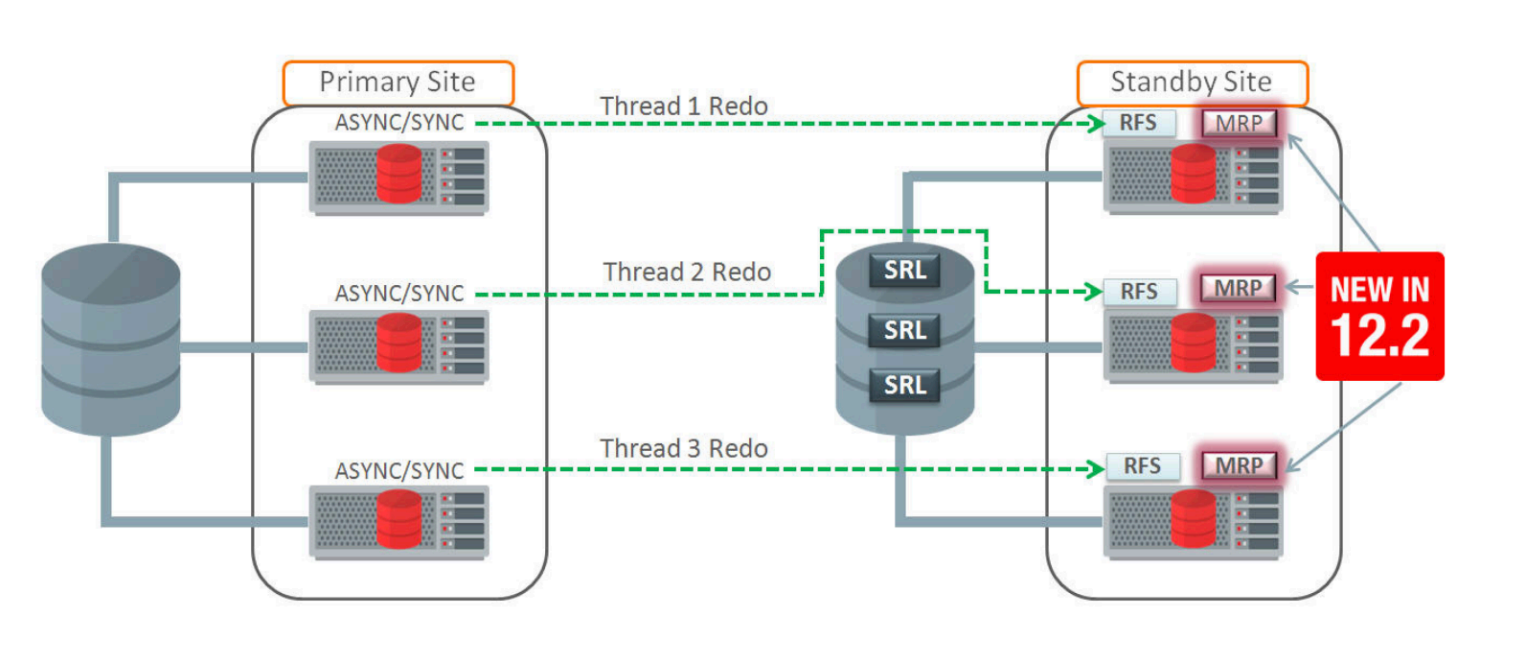
ADG增强

- **快速failover增强**
  - 在备库上的只读会话在故障切换时保持连接
  - 备库上的只读会话，切换后会变成读写会话
  - 上述2点。目的是为了降低连接风暴



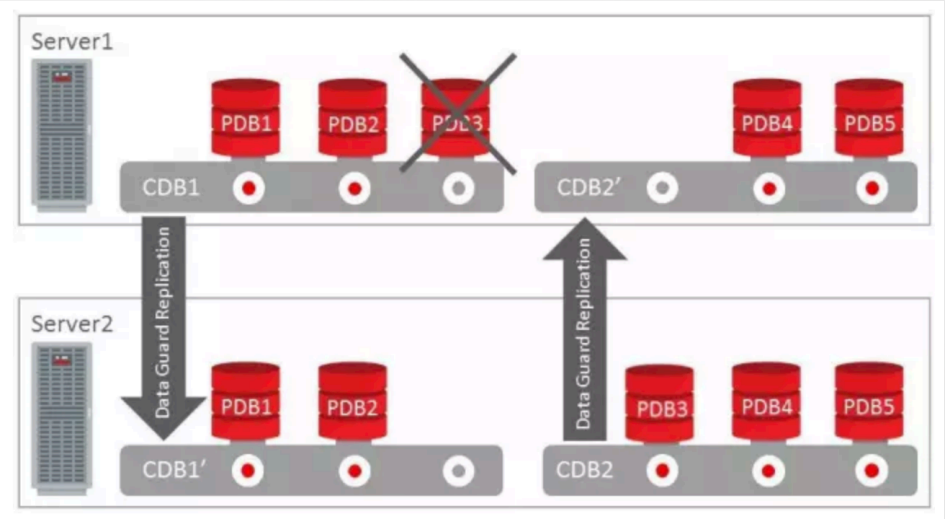
ADG增强

- **性能增强**
  - 在支持 in-memory
  - RAC的dataguard支持多实例的mrp恢复（multi-node MRP，即原来只能启动一个实例的mrp恢复，其他实例无mrp进程）



ADG增强

- **管理增强**
  - dgbroker
    - ✓ 支持dgbroker的多observer
    - ✓ MIGRATE PLUGGABLE DATABASE,实现在同一个server上将PDB从CBD_1迁移到CBD_2
      - 可以实现pdb的migrate from primary CDB to another Primary CDB
      - 可以实现pdb的failover from a standby CDB to another primary CDB
  - Dataguard复制增强
    - ✓ subset ADG，即只复制部分PDB到ADG
  - 支持AWR
  - 支持sql tuning advisor
  - PDB级的故障切换
  - 支持duplicate命令创建far sync dg
  - 支持duplicate primary from standby



SPA增强

- 三个新task
  - EXECUTE_FULLDML_TRIGGERS

    使用此参数可以启用或禁用在FULLDML模式下运行SPA试用时递归触发的数据库触发器。
  - EXECUTE_WITH_FIXED_DATE

    在SPA试用中使用此参数为引用SYSDATE函数的SQL设置固定日期。
  - NUM_ROWS_TO_FETCH

    此参数允许限制SQL语句根据优化程序模式设置获取的行数。

STS增强

- 收集所有rac节点的workload（如db replay）
- 收集关于exadata，sql monitoring，in memory的信息

安全增强

- **TDE 全加密，包括system，sysaux，undo表空间**
- **TDE支持更多加密算法：ARIA，SEED，GOST**
- **RAS（Real Application Security，Virtual Private Database的下一代）**
  - 应用程序用户 <--> VPD策略制定者 <--> 不同部门的用户 (VPD， 基于行或者列，添加where条件)
  - 粒度更小
    - ✓ Update操作：用户拥有行权限和列权限，才可以更新表中的某行的某个字段
    - ✓ Insert操作：用户需要行权限和列权限才能insert对应列的对应字段，而非保护字段，可以正常插入
    - ✓ Delete操作，用户需要行权限，无需列权限
    - ✓ Select操作，也需要行权限和列权限
  - 结合label security实现session级数据保护

注，Oracle Advanced Security相关：TDE，Data Masking，Data redaction，VPD，DB Vault，OLS，SSL，tcp.invited_nodes

其他

- **指定PDB升级的优先级，让某些pdb先升级**
- **awr和autotask参数**
  - AWR_PDB_AUTOFLUSH_ENABLED
  - ENABLE_AUTOMATIC_MAINTENANCE_PDB
  - AUTOTASK_MAX_ACTIVE_PDBS
- **MGMTDB**
  - 安装可以选择NO（仅仅是不安装在OCRVD，实际还是需要安装）

其他

- **cluster评估语句增强**
  - 12.1
    - ✓ crsctl eval=srvctl -eval
    - ✓ crsctl eval fail=srvctl predict

```
[oracle@12102-rac1 ~]$ crsctl eval stop res ora.cdbrac.db -unsupported

Stage Group 1:
--------------------------------------------------------------------------------
Stage Number    Required        Action
--------------------------------------------------------------------------------

     1              Y           Resource 'ora.cdbrac.db' (2/1) will be in state
                                [OFFLINE]

--------------------------------------------------------------------------------
[oracle@12102-rac1 ~]$
```

  - 12.2 多了 -explain参数
    - ✓ Not only what, but also why

```
$ crsctl eval delete server mjk-node2-3 -explain

Stage Group 1:
--------------------------------------------------------------------------------
Stage Required    Action
--------------------------------------------------------------------------------
   1 E Server 'mjk-node2-3' is removed from server pool 'sp1'.
       E Server pool 'sp1' is below the MIN_SIZE value of 2 with 1
            servers.
       E Looking at other server pools to see whether MIN_SIZE value 2 of
            server pool 'sp1' can be met.
       E Scanning server pools with MIN_SIZE or fewer servers in
            ascending order of IMPORTANCE.
       E Considering server pools (IMPORTANCE): sp2(2) for suitable
            servers.
       E Considering server pool 'sp2' because its MIN_SIZE is 2 and it
            has 0 servers above MIN_SIZE.
       E Relocating server 'mjk-node2-0' to server pool 'sp1'.
       Y Server 'mjk-node2-3' will be removed from pools 'sp1'.
       Y Server 'mjk-node2-0' will be moved from pools 'sp2' to
            pools 'sp1'
================================================================================
```

1 开发篇

开发篇

1. JDBC Driver
2. JSON
3. CDB字符集增强
4. SQLPLUS支持history
5. 分区增强
6. Real time Materialized view
7. listagg函数(行转列）
8. Approx query
9. ACCESSIBLE BY增强

JDBC Driver（12.1开始）

SQLNET.ALLOWED_LOGON_VERSION_SERVER 在12c中默认值是11，所以如10.2.0.5的JDBC连接过来，就会报错ora-28040。虽然可以设置SQLNET.ALLOWED_LOGON_VERSION_SERVER=8来解决这个问题，但是由于今后11.1以下版本的JDBC不再被oracle支持，因此还是建议升级JDBC驱动来实现。
jar file和JDBC驱动之间的关系，参考Doc ID 401934.1。

What are the Oracle JDBC versions ?

The Oracle JDBC driver is shipped with the Oracle database server.
Starting with 9.2.0.1 database version, the Oracle JDBC driver versions are :

- JDBC 9.2.0.X
- JDBC 10.1.0.X
- JDBC 10.2.0.X
- JDBC 11.1.0.X
- JDBC 11.2.0.X
- JDBC 12.1.0.X
- JDBC 12.2.0.x

The supported JDBC drivers versions are : 12.2, 12.1 , and 11.2 .

JDBC Version	JDK version	JDBC File Name
12.2.0	8.x	ojdbc8.jar
12.1.0	8.x 7.x 6.x	ojdbc7.jar ojdbc7.jar ojdbc6.jar
11.2.0	8.x ** 7.x ** 6.x 5.x	ojdbc6.jar ojdbc6.jar ojdbc6.jar ojdbc5.jar
11.1.0	6.x 5.x	ojdbc6.jar ojdbc5.jar
10.2.0	1.2.x 1.3.x 1.4.x 5.x	classes12.jar classes12.jar ojdbc14.jar ojdbc14.jar
10.1.0	1.2.x 1.3.x 1.4.x	classes12.jar classes12.jar ojdbc14.jar
9.2.0	1.1.x 1.2.x 1.3.x 1.4.x	classes111.zip* classes12.zip* classes12.zip* ojdbc14.jar

JDBC drivers	Oracle database
12.2.0	12.2.0 12.1.0 11.2.0
12.1.0	12.2.0 12.1.0 11.2.0 11.1.0
11.2.0	12.2.0 12.1.0 11.2.0 11.1.0 10.2.0 10.1.0 9.2.0
11.1.0	12.1.0 11.2.0 11.1.0 10.2.0 10.1.0 9.2.0
10.2.0	12.1.0 11.2.0 11.1.0 10.2.0 10.1.0 9.2.0
10.1	11.2.0 11.1.0 10.2.0 10.1.0 9.2.0
9.2.0	11.2.0 11.1.0 10.2.0 10.1.0 9.2.0

JSON（12.1开始原生支持json）

- **JSON 文件使用现有的数据类型存到数据库中**
  - VARCHAR2（32k限制，超过32k可以使用blob或clob）
  - BLOB
  - CLOB
  - HDFS(外部表）
- **JSON search索引新语法**
  - CREATE SEARCH INDEX [index_name] ON [table_name](column_name) FOR JSON

```
-- 12.1 Syntax
CREATE INDEX json_docs_search_idx ON json_documents (data)
  INDEXTYPE IS CTXSYS.CONTEXT
  PARAMETERS ('section group CTXSYS.JSON_SECTION_GROUP SYNC (ON COMMIT)');

-- 12.2 Syntax
CREATE SEARCH INDEX json_docs_search_idx ON json_documents (data) FOR JSON;

EXEC DBMS_STATS.gather_table_stats(USER, 'JSON_DOCUMENTS');
```

- **JSON search增强**
  - 索引支持range和list分区

JSON（12.1开始原生支持json）

- **JSON运算符**
  - IS JSON/IS JSON (STRICT)，常常用在某字段的约束，检查其是否为JSON，而加STRICT表示同时检查json语法

```
CREATE TABLE json_documents (
  id    RAW(16) NOT NULL,
  data  CLOB,
  CONSTRAINT json_documents_pk PRIMARY KEY (id),
  CONSTRAINT json_documents_json_chk CHECK (data IS JSON)
);
```

```
INSERT INTO json_documents (id, data)
VALUES (SYS_GUID(),
        '{
          "FirstName"      : "John",
          "LastName"       : "Doe",
          "Job"            : "Clerk",
          "Address"        : {
                               "Street"   : "99 My Street",
                               "City"     : "My City",
                               "Country"  : "UK",
                               "Postcode" : "A12 34B"
                             },
          "ContactDetails" : {
                               "Email"    : "john.doe@example.com",
                               "Phone"    : "44 123 123456",
                               "Twitter"  : "@johndoe"
                             },
          "DateOfBirth"    : "01-JAN-1980",
          "Active"         : true
        }');

INSERT INTO json_documents (id, data)
VALUES (SYS_GUID(),
        '{
          "FirstName"      : "Jayne",
          "LastName"       : "Doe",
          "Job"            : "Manager",
          "Address"        : {
                               "Street"   : "100 My Street",
                               "City"     : "My City",
                               "Country"  : "UK",
                               "Postcode" : "A12 34B"
                             },
          "ContactDetails" : {
                               "Email"    : "jayne.doe@example.com",
                               "Phone"    : ""
                             },
          "DateOfBirth"    : "01-JAN-1982",
          "Active"         : false
        }');

COMMIT;
```

  - JSON_EXISTS-12.2支持在where 中

```
SELECT a.data.FirstName,
       a.data.LastName,
       a.data.ContactDetails.Email AS Email
FROM   json_documents a
WHERE  JSON_EXISTS(a.data.ContactDetails, '$.Phone' FALSE ON ERROR)
AND    a.data.ContactDetails.Phone IS NULL;

FIRSTNAME       LASTNAME        EMAIL
--------------- --------------- -------------------------
Jayne           Doe             jayne.doe@example.com

1 row selected.

SQL>
```

JSON（12.1开始原生支持json）

- **JSON运算符**
  - JSON_TEXTCONTAINS（注意，只有在json search索引建立的情况下，才能使用JSON_TEXTCONTAINS 全文检索）

```
SELECT COUNT(*)
FROM   json_documents
WHERE  JSON_TEXTCONTAINS(data, '$.ContactDetails.Email', 'john.doe@example.com');

----------------------------------------------------------------------------------------------
| Id  | Operation         | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |                     |     1 |  2014 |     4   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE   |                     |     1 |  2014 |            |          |
|*  2 |   DOMAIN INDEX    | JSON_DOCS_SEARCH_IDX |     1 |  2014 |     4   (0)| 00:00:01 |
----------------------------------------------------------------------------------------------

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

   2 - access("CTXSYS"."CONTAINS"("JSON_DOCUMENTS"."DATA",'{john.doe@example.com}
              INPATH(/ContactDetails/Email)')>0)
```

```
INSERT INTO json_documents (id, data)
VALUES (SYS_GUID(),
        '{
          "FirstName"      : "John",
          "LastName"       : "Doe",
          "Job"            : "Clerk",
          "Address"        : {
                               "Street"   : "99 My Street",
                               "City"     : "My City",
                               "Country"  : "UK",
                               "Postcode" : "A12 34B"
                             },
          "ContactDetails" : {
                               "Email"    : "john.doe@example.com",
                               "Phone"    : "44 123 123456",
                               "Twitter"  : "@johndoe"
                             },
          "DateOfBirth"    : "01-JAN-1980",
          "Active"         : true
         }');

INSERT INTO json_documents (id, data)
VALUES (SYS_GUID(),
        '{
          "FirstName"      : "Jayne",
          "LastName"       : "Doe",
          "Job"            : "Manager",
          "Address"        : {
                               "Street"   : "100 My Street",
                               "City"     : "My City",
                               "Country"  : "UK",
                               "Postcode" : "A12 34B"
                             },
          "ContactDetails" : {
                               "Email"    : "jayne.doe@example.com",
                               "Phone"    : ""
                             },
          "DateOfBirth"    : "01-JAN-1982",
          "Active"         : false
         }');

COMMIT;
```

  - JSON_QUERY

```
SELECT a.data.FirstName,
       a.data.LastName,
       JSON_QUERY a.data, '$.ContactDetails' WITH WRAPPER) AS contact_details
FROM   json_documents a
ORDER BY a.data.FirstName,
         a.data.Last_name;

FIRSTNAME       LASTNAME        CONTACT_DETAILS
--------------- --------------- ----------------------------------------
Jayne           Doe             [{"Email":"jayne.doe@example.com","Phone
                                ":""}]

John            Doe             [{"Email":"john.doe@example.com","Phone"
                                :"44 123 123456","Twitter":"@johndoe"}]


2 rows selected.

SQL>
```

JSON（12.1开始原生支持json）

- **JSON可索引**
  - 函数索引

```
CREATE INDEX json_docs_email_idx
  ON json_documents (JSON_VALUE(data, '$.ContactDetails.Email' RETURNING VARCHAR2 ERROR ON ERROR));

SET AUTOTRACE TRACE EXPLAIN

SELECT a.data.FirstName,
       a.data.LastName,
       a.data.ContactDetails.Email AS Email
FROM   json_documents a
WHERE  JSON_VALUE(data, '$.ContactDetails.Email' RETURNING VARCHAR2 ERROR ON ERROR) = 'john.doe@example.com';

-----------------------------------------------------------------------------------------------------------
| Id  | Operation                            | Name               | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                     |                    |     1 |  1499 |     2   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID BATCHED| JSON_DOCUMENTS     |     1 |  1499 |     2   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN                   | JSON_DOCS_EMAIL_IDX |     1 |       |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------------

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

   2 - access(JSON_VALUE("DATA" FORMAT JSON , '$.ContactDetails.Email' RETURNING VARCHAR2(4000)
              ERROR ON ERROR)='john.doe@example.com')
```

```
INSERT INTO json_documents (id, data)
VALUES (SYS_GUID(),
        '{
          "FirstName"      : "John",
          "LastName"       : "Doe",
          "Job"            : "Clerk",
          "Address"        : {
                               "Street"   : "99 My Street",
                               "City"     : "My City",
                               "Country"  : "UK",
                               "Postcode" : "A12 34B"
                             },
          "ContactDetails" : {
                               "Email"    : "john.doe@example.com",
                               "Phone"    : "44 123 123456",
                               "Twitter"  : "@johndoe"
                             },
          "DateOfBirth"    : "01-JAN-1980",
          "Active"         : true
        }');

INSERT INTO json_documents (id, data)
VALUES (SYS_GUID(),
        '{
          "FirstName"      : "Jayne",
          "LastName"       : "Doe",
          "Job"            : "Manager",
          "Address"        : {
                               "Street"   : "100 My Street",
                               "City"     : "My City",
                               "Country"  : "UK",
                               "Postcode" : "A12 34B"
                             },
          "ContactDetails" : {
                               "Email"    : "jayne.doe@example.com",
                               "Phone"    : ""
                             },
          "DateOfBirth"    : "01-JAN-1982",
          "Active"         : false
        }');

COMMIT;
```

JSON（12.1开始原生支持json）

- **JSON可索引**
  - 虚拟列复合索引

```
-- Create the virtual columns and index.
ALTER TABLE json_documents ADD (first_name VARCHAR2(50)
  GENERATED ALWAYS AS (JSON_VALUE(data, '$.FirstName' RETURNING VARCHAR2(50))));

ALTER TABLE json_documents ADD (last_name VARCHAR2(50)
  GENERATED ALWAYS AS (JSON_VALUE(data, '$.LastName' RETURNING VARCHAR2(50))));

CREATE INDEX json_docs_name_idx ON json_documents (first_name, last_name);


-- Test a query against the virtual columns,
SET AUTOTRACE TRACE EXPLAIN

SELECT COUNT(*)
FROM   json_documents
WHERE  first_name = 'John'
AND    last_name  = 'Doe';

----------------------------------------------------------------------------------------------
| Id  | Operation         | Name               | Rows  | Bytes | Cost (%CPU)| Time     |
----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |                    |     1 |    54 |     1   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE   |                    |     1 |    54 |            |          |
|*  2 |   INDEX RANGE SCAN| JSON_DOCS_NAME_IDX |     1 |    54 |     1   (0)| 00:00:01 |
----------------------------------------------------------------------------------------------

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

   2 - access("FIRST_NAME"='John' AND "LAST_NAME"='Doe')
```

```
INSERT INTO json_documents (id, data)
VALUES (SYS_GUID(),
        '{
          "FirstName"      : "John",
          "LastName"       : "Doe",
          "Job"            : "Clerk",
          "Address"        : {
                               "Street"   : "99 My Street",
                               "City"     : "My City",
                               "Country"  : "UK",
                               "Postcode" : "A12 34B"
                             },
          "ContactDetails" : {
                               "Email"    : "john.doe@example.com",
                               "Phone"    : "44 123 123456",
                               "Twitter"  : "@johndoe"
                             },
          "DateOfBirth"    : "01-JAN-1980",
          "Active"         : true
        }');

INSERT INTO json_documents (id, data)
VALUES (SYS_GUID(),
        '{
          "FirstName"      : "Jayne",
          "LastName"       : "Doe",
          "Job"            : "Manager",
          "Address"        : {
                               "Street"   : "100 My Street",
                               "City"     : "My City",
                               "Country"  : "UK",
                               "Postcode" : "A12 34B"
                             },
          "ContactDetails" : {
                               "Email"    : "jayne.doe@example.com",
                               "Phone"    : ""
                             },
          "DateOfBirth"    : "01-JAN-1982",
          "Active"         : false
        }');

COMMIT;
```

JSON（12.1开始原生支持json）

- **JSON可索引**
  - bitmap索引

```
CREATE BITMAP INDEX json_docs_name_idx ON json_documents (
  JSON_VALUE(data, '$.FirstName')
);

DROP INDEX json_docs_name_idx;
CREATE BITMAP INDEX json_docs_name_idx ON json_documents (
  JSON_QUERY(data, '$.FirstName')
);

DROP INDEX json_docs_name_idx;
CREATE BITMAP INDEX json_docs_name_idx ON json_documents (
  JSON_EXISTS(data, '$.FirstName')
);
```

```
INSERT INTO json_documents (id, data)
VALUES (SYS_GUID(),
        '{
          "FirstName"      : "John",
          "LastName"       : "Doe",
          "Job"            : "Clerk",
          "Address"        : {
                               "Street"   : "99 My Street",
                               "City"     : "My City",
                               "Country"  : "UK",
                               "Postcode" : "A12 34B"
                             },
          "ContactDetails" : {
                               "Email"    : "john.doe@example.com",
                               "Phone"    : "44 123 123456",
                               "Twitter"  : "@johndoe"
                             },
          "DateOfBirth"    : "01-JAN-1980",
          "Active"         : true
        }');

INSERT INTO json_documents (id, data)
VALUES (SYS_GUID(),
        '{
          "FirstName"      : "Jayne",
          "LastName"       : "Doe",
          "Job"            : "Manager",
          "Address"        : {
                               "Street"   : "100 My Street",
                               "City"     : "My City",
                               "Country"  : "UK",
                               "Postcode" : "A12 34B"
                             },
          "ContactDetails" : {
                               "Email"    : "jayne.doe@example.com",
                               "Phone"    : ""
                             },
          "DateOfBirth"    : "01-JAN-1982",
          "Active"         : false
        }');

COMMIT;
```

JSON（12.1开始原生支持json）

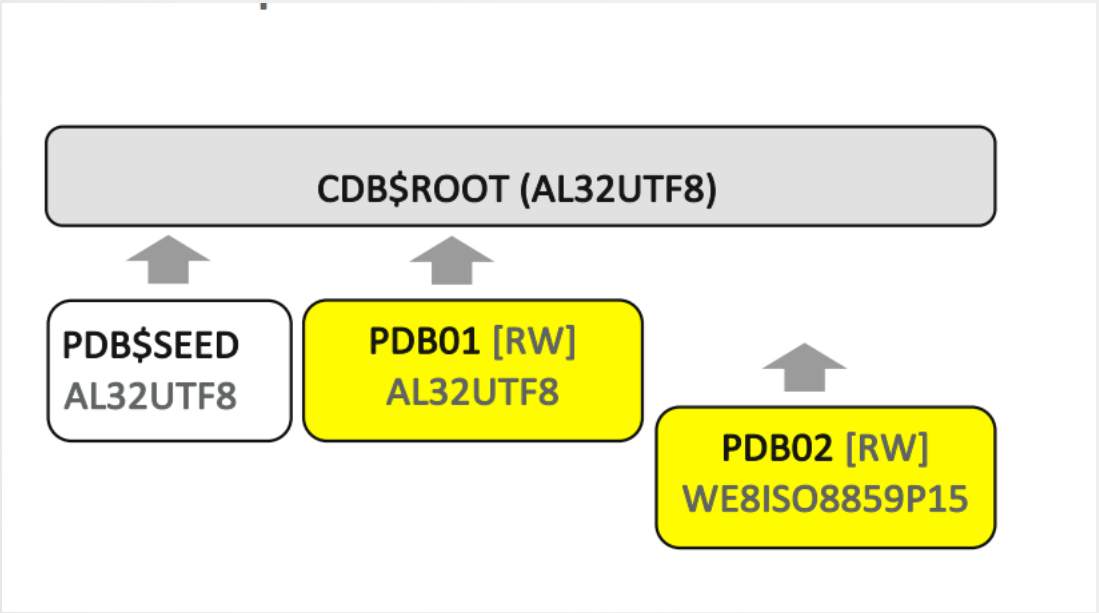
- **JSON可索引**
  - 全文检索索引，即之前提到的create search index for json

```
SELECT COUNT(*)
FROM   json_documents
WHERE  JSON_TEXTCONTAINS(data, '$.ContactDetails.Email', 'john.doe@example.com');

-----------------------------------------------------------------------------------------------
| Id  | Operation         | Name                 | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |                      |     1 |  2014 |     4   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE   |                      |     1 |  2014 |            |          |
|*  2 |   DOMAIN INDEX    | JSON_DOCS_SEARCH_IDX |     1 |  2014 |     4   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------

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

   2 - access("CTXSYS"."CONTAINS"("JSON_DOCUMENTS"."DATA",'{john.doe@example.com}
               INPATH(/ContactDetails/Email)')>0)
```

```
INSERT INTO json_documents (id, data)
VALUES (SYS_GUID(),
        '{
          "FirstName"      : "John",
          "LastName"       : "Doe",
          "Job"            : "Clerk",
          "Address"        : {
                               "Street"   : "99 My Street",
                               "City"     : "My City",
                               "Country"  : "UK",
                               "Postcode" : "A12 34B"
                             },
          "ContactDetails" : {
                               "Email"    : "john.doe@example.com",
                               "Phone"    : "44 123 123456",
                               "Twitter"  : "@johndoe"
                             },
          "DateOfBirth"    : "01-JAN-1980",
          "Active"         : true
        }');

INSERT INTO json_documents (id, data)
VALUES (SYS_GUID(),
        '{
          "FirstName"      : "Jayne",
          "LastName"       : "Doe",
          "Job"            : "Manager",
          "Address"        : {
                               "Street"   : "100 My Street",
                               "City"     : "My City",
                               "Country"  : "UK",
                               "Postcode" : "A12 34B"
                             },
          "ContactDetails" : {
                               "Email"    : "jayne.doe@example.com",
                               "Phone"    : ""
                             },
          "DateOfBirth"    : "01-JAN-1982",
          "Active"         : false
        }');

COMMIT;
```

CDB字符集增强

- 支持PDB不同于CDB的字符集（要求CDB为al32utf8，默认）
- 12.2 PDB插入CDB后报ORA-41401错误， Advance Queue的bug， BUG 22331316
- Character Sets For CDB And PDB in 12.2 (Doc ID 2231602.1)



sqlplus支持history命令

```
SQL> hist list
  1  select name from v$database;
  2  select name, database_role from v$database;
  3  select banner from v$version;

SQL>
SQL>
SQL> hist 1 run

NAME
---------
DB122
```

```
SQL> hist clear
SQL>
SQL> hist list
SP2-1651: History list is empty.
SQL>
```

分区增强

- **Multi-column list partition。注：最多支持16个列**

```
CREATE TABLE t_oracleblog (salername varchar(200),region VARCHAR2(50), channel VARCHAR2(50))
PARTITION BY LIST (region, channel)  --Note keywork :region, channel, Here are 2 columns
(
partition p1 values ('USA','Direct'),
partition p2 values ('USA','Partners'),
partition p3 values ('GERMANY','Direct'),
partition p4 values (('GERMANY','Partners'),('GERMANY','Web')),
partition p5 values ('CHINA','Direct'),
partition p6 values (('CHINA','Partners'),('CHINA','Web'),('CHINA','Oversee')),
partition p7 values ('JAPAN','Direct'),
partition p8 values (DEFAULT)
)
/
```

- **Auto-list partition**

```
CREATE TABLE t_car (brand VARCHAR2(50),model VARCHAR2(50), year char(4))
PARTITION BY LIST (brand) AUTOMATIC --Note keywork :AUTOMATIC
(
partition p1 values ('BMW'),
partition p2 values ('BENZ')
)
/
```

- **Interval subpartition**

```
Create table CARS (auto_make varchar(30),
Auto_model varchar(30),
Purchase_date date)
PARTITION BY LIST (auto_make)
SUBPARTITION BY RANGE (Purchase_Date) INTERVAL
(NUMTOYMINTERVAL(1,'month'))
SUBPARTITION TEMPLATE
(subpartition sp1 values less than
TO_DATE('15-03-2015','dd-mm-yyyy')))
(partition p_bmw values ('BMW'))
/
```

分区增强

- **Online DDL for partition**

```
ALTER TABLE employees_convert MODIFY
  PARTITION BY RANGE (employee_id) INTERVAL (100)
  ( PARTITION P1 VALUES LESS THAN (100),
    PARTITION P2 VALUES LESS THAN (500)
   ) ONLINE
  UPDATE INDEXES
 ( IDX1_SALARY LOCAL,
   IDX2_EMP_ID GLOBAL PARTITION BY RANGE (employee_id)
  ( PARTITION IP1 VALUES LESS THAN (MAXVALUE))
 );
```

- **Filtered Partition on Maintenance Operations**

```
ALTER TABLE T_ORACLEBLOG MOVE PARTITION p4
TABLESPACE SYSAUX
INCLUDING ROWS WHERE REGION = 'CHINA' --Note keyword INCLUDING ROW WHERE;
```

分区增强

- **Read only partition**

```
CREATE TABLE orders
(
order_id number,
order_date DATE,
customer_name varchar2(200)
) read only  ----Note keyword read only, which mean table read only
PARTITION BY RANGE(order_date)
(
partition q1_2015 values less than (to_date('2014-10-01','yyyy-mm-dd')),
partition q2_2015 values less than (to_date('2015-01-01','yyyy-mm-dd')),
partition q3_2015 values less than (to_date('2015-04-01','yyyy-mm-dd')),
partition q4_2015 values less than (to_date('2015-07-01','yyyy-mm-dd')) read write ----Note keyword read only, which mean partition q4 read write
)
/
```

Real time Materialized view

- 在12.2之前，如果你想获得实时的数据，那么在利用query rewrite前，你必须得用on commit的刷新方式刷新物化视图。但是on commit的刷新方式有众多限制，如sql的复杂度，如频繁对系统的压力等等。所以，我们不得不采用on command的方式来进行刷新（不管是全量刷新还是增量刷新）。那么在使用on command刷新的时候，必须得有个job来定时的刷，那么，在一次job运行之后，下一次job到来之前，如果基表有数据变化，那么此时的数据肯定不是最新的。
- 利用增量刷新的log，在查询时快速输出最新结果。
- 注，还是需要快速刷新的操作
- 使用/*+ fresh_mv */ 的hint

```
create materialized view mv_new
refresh fast on demand
enable on query computation    --- Note the key word here.
enable query rewrite
as
select y , count(*) c1
from t2
group by y;
```

listagg函数

- **listagg函数(行转列）- 12.2增强 on overflow truncate**

```
CREATE TABLE EMP (
  EMPNO NUMBER(4,0),
  ENAME VARCHAR2(10 BYTE),
  JOB VARCHAR2(9 BYTE),
  MGR NUMBER(4,0),
  HIREDATE DATE,
  SAL NUMBER(7,2),
  COMM NUMBER(7,2),
  DEPTNO NUMBER(2,0),
  CONSTRAINT PK_EMP PRIMARY KEY (EMPNO)
  );

insert into EMP (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values (7369,'SMITH','CLERK',7902,to_date('17-DEC-80','DD-MON-RR'),800,null,20);
insert into EMP (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values (7499,'ALLEN','SALESMAN',7698,to_date('20-FEB-81','DD-MON-RR'),1600,300,30);
insert into EMP (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values (7521,'WARD','SALESMAN',7698,to_date('22-FEB-81','DD-MON-RR'),1250,500,30);
insert into EMP (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values (7566,'JONES','MANAGER',7839,to_date('02-APR-81','DD-MON-RR'),2975,null,20);
insert into EMP (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values (7654,'MARTIN','SALESMAN',7698,to_date('28-SEP-81','DD-MON-RR'),1250,1400,30);
insert into EMP (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values (7698,'BLAKE','MANAGER',7839,to_date('01-MAY-81','DD-MON-RR'),2850,null,30);
insert into EMP (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values (7782,'CLARK','MANAGER',7839,to_date('09-JUN-81','DD-MON-RR'),2450,null,10);
insert into EMP (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values (7788,'SCOTT','ANALYST',7566,to_date('19-APR-87','DD-MON-RR'),3000,null,20);
insert into EMP (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values (7839,'KING','PRESIDENT',null,to_date('17-NOV-81','DD-MON-RR'),5000,null,10);
insert into EMP (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values (7844,'TURNER','SALESMAN',7698,to_date('08-SEP-81','DD-MON-RR'),1500,0,30);
insert into EMP (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values (7876,'ADAMS','CLERK',7788,to_date('23-MAY-87','DD-MON-RR'),1100,null,20);
insert into EMP (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values (7900,'JAMES','CLERK',7698,to_date('03-DEC-81','DD-MON-RR'),950,null,30);
insert into EMP (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values (7902,'FORD','ANALYST',7566,to_date('03-DEC-81','DD-MON-RR'),3000,null,20);
insert into EMP (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values (7934,'MILLER','CLERK',7782,to_date('23-JAN-82','DD-MON-RR'),1300,null,10);
commit;
```

```
SELECT deptno, LISTAGG(ename, ',' ON OVERFLOW TRUNCATE '~~~') WITHIN GROUP (ORDER BY ename) AS employees
FROM   emp
       CROSS JOIN (SELECT level FROM dual CONNECT BY level <= 1000)
WHERE  deptno = 30
GROUP BY deptno
ORDER BY deptno;

    DEPTNO EMPLOYEES
---------- --------------------------------------------------------------------
        30 ALLEN,ALLEN,ALLEN,ALLEN,ALLEN,ALLEN,ALLEN,ALLEN,ALLEN,ALLEN,ALLEN,ALLE
           N,ALLEN,ALLEN,ALLEN,ALLEN,ALLEN,ALLEN,ALLEN,ALLEN,ALLEN,ALLEN,ALLEN,AL
           LEN,ALLEN,ALLEN,ALLEN,ALLEN,ALLEN,ALLEN,ALLEN,ALLEN,ALLEN,ALLEN,ALLEN,
[removed]
           N,ALLEN,ALLEN,ALLEN,ALLEN,ALLEN,ALLEN,ALLEN,ALLEN,ALLEN,ALLEN,ALLEN,AL
           LEN,ALLEN,ALLEN,ALLEN,ALLEN,ALLEN,ALLEN,ALLEN,~~~(5339)
```

Approx query（12.1开始）

- 需要收集统计信息
- approx_count_distinct
- Approx_count_distinct_detail
- Approx_count_distinct_agg
- To_approx_count_distinct
- Approx_median
- 好处：
  - 可以group by
  - 省时间
  - 省pga
- 启用：alter session set approx_for_xxx=true
- 误差 < 0.5%

ACCESSIBLE BY增强（12.1开始）

- **Create package xxx accessible by (procedure xx_procedure)**
- **作用：package白名单。**
- **12.2增强，对invalid的package body或function也生效，即对当前不存在的package body或funcation也生效。**

```
CONN test1/test1@pdb1

CREATE OR REPLACE PROCEDURE protected_proc
  ACCESSIBLE BY (calling_proc)
AS
BEGIN
  DBMS_OUTPUT.put_line('TEST1 : protected_proc');
END;
/

Procedure created.

SQL>
```

```
CREATE OR REPLACE PROCEDURE calling_proc AS
BEGIN
  DBMS_OUTPUT.put_line('TEST1 : calling_proc');
  protected_proc;
END;
/

Procedure created.

SQL> SET SERVEROUTPUT ON
SQL> EXEC calling_proc;
TEST1 : calling_proc
TEST1 : protected_proc

PL/SQL procedure successfully completed.

SQL>
```

```
SQL> EXEC protected_proc;
BEGIN protected_proc; END;

      *
ERROR at line 1:
ORA-06550: line 1, column 7:
PLS-00904: insufficient privilege to access object PROTECTED_PROC
ORA-06550: line 1, column 7:
PL/SQL: Statement ignored

SQL>
```

Thanks

Q & A

THE FUTURE OF POSSIBLE