

Continuous Availability Workshop

Part III - Database sharding



Contents

CONTINUOUS AVAILABILITY WORKSHOP	1
INITIAL REQUIREMENTS	2
DATABASE SHARDING	3
<i>Check environment</i>	3
<i>Deploy a sharded database</i>	6
<i>Create a non sharded application</i>	24
<i>Migrate to sharded database</i>	33
<i>Setup and Run the Demo Application</i>	45
<i>Database requests routing to shards</i>	48
<i>Sharded database dynamic scaling</i>	58

Initial requirements

- SSH private key to Access the database servers in the cloud. This private key is provided along with this manual.
- SSH client app, to login to the database servers
- Database servers public IP



Database sharding

Check environment

For additional details about this lab, especially about sharding concepts, please check "https://apexapps.oracle.com/pls/apex/dbpm/r/livelabs/workshop-attendee-2?p210_workshop_id=835&p210_type=3&session=4182485884819".

In the following workshop, you will:

- Deploy a shard database with two shards using system managed sharding
- Migrate an application to the shard database
- Work with the sharded database
- Extend the sharded database with a third shard

Your environment is made of 4 servers. You have been provided with the public IP of each of the servers, along with the private key to ssh them. The servers are:

- Shard catalog
- Shard 1
- Shard 2
- Shard 3

To start-up with the lab, connect to the 4 servers and check that a database is up and running on each of them, as well as a listener:

```
## Connect to the shard catalog

ssh -i privateKey opc@<public ip of shard catalog>

## Connect as "oracle" user and connect to the database

sudo su - oracle

[oracle@cata ~]$ sqlplus / as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Tue Nov 9 11:36:07 2021
Version 19.11.0.0.0

Copyright (c) 1982, 2020, Oracle. All rights reserved.
```



Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.11.0.0.0

SQL>

SQL> show pdbs

CON_ID	CON_NAME	OPEN MODE	RESTRICTED
2	PDB\$SEED	READ ONLY	NO
3	CATAPDB	READ WRITE	NO

SQL> exit

lsnrctl status LISTENER

Connect to shard 1

ssh -i privateKey opc@<public ip of shard 1>

Connect as "oracle" user and connect to the database

sudo su - oracle

[oracle@shd1 ~]\$ sqlplus / as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Tue Nov 9 11:37:10 2021
Version 19.11.0.0.0

Copyright (c) 1982, 2020, Oracle. All rights reserved.

Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.11.0.0.0

SQL> show pdbs

CON_ID	CON_NAME	OPEN MODE	RESTRICTED
2	PDB\$SEED	READ ONLY	NO
3	SHDPDB1	READ WRITE	NO

exit

lsnrctl status LISTENER

Connect to shard 2

ssh -i privateKey opc@<public ip of shard 2>

Connect as "oracle" user and connect to the database

sudo su - oracle



```
[oracle@shd2 ~]$ sqlplus / as sysdba
```

```
SQL*Plus: Release 19.0.0.0.0 - Production on Tue Nov 9 11:37:10 2021  
Version 19.11.0.0.0
```

```
Copyright (c) 1982, 2020, Oracle. All rights reserved.
```

```
Connected to:  
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production  
Version 19.11.0.0.0
```

```
SQL> show pdbs
```

CON_ID	CON_NAME	OPEN MODE	RESTRICTED
2	PDB\$SEED	READ ONLY	NO
3	SHDPDB2	READ WRITE	NO

```
exit
```

```
lsnrctl status LISTENER
```

```
## Connect to shard 3  
ssh -i privateKey opc@<public ip of shard 3>
```

```
## Connect as "oracle" user and connect to the database
```

```
sudo su - oracle
```

```
[oracle@shd3 ~]$ sqlplus / as sysdba
```

```
SQL*Plus: Release 19.0.0.0.0 - Production on Tue Nov 9 11:37:10 2021  
Version 19.11.0.0.0
```

```
Copyright (c) 1982, 2020, Oracle. All rights reserved.
```

```
Connected to:  
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production  
Version 19.11.0.0.0
```

```
SQL> show pdbs
```

CON_ID	CON_NAME	OPEN MODE	RESTRICTED
2	PDB\$SEED	READ ONLY	NO
3	SHDPDB3	READ WRITE	NO

```
exit
```

```
lsnrctl status LISTENER
```



Deploy a sharded database

In the following section, we will deploy a sharded database. Let's sum-up what we've done so far, and how we will use the components in the coming steps.

So far, we have 4 databases instances, and their unique names are:

- cata: we will use it as the catalog database
- shd1: will be database shard 1
- shd2: will be database shard 2
- shd3: will be used as database shard 3, added to the first two shards at the end of the lab, to illustrate sharding horizontal scalability

Before going further, review the public and private IP of your 4 servers, and complete a table like the one given as an example below:

Public IP	Private IP	CDB name	PDB name
<catalog_public_ip>	<catalog_private_ip>	cata	catapdb
<shard1_public_ip>	<shard1_private_ip>	shd1	shdpdb1
<shard2_public_ip>	<shard2_private_ip>	shd2	shdpdb2
<shard3_public_ip>	<shard3_private_ip>	shd3	shdpdb3

Replace all the IP by your own values.

Configure the shard hosts: Connect to the catalog host and each of the shard hosts with root user, then edit the /etc/hosts file:

```
ssh -i privateKey opc@<catalog host public IP>

## Sudo to root user
sudo -i

## Modify the /etc/hosts file, with the name and private IP of the 4 machines:

10.0.1.125 cata
10.0.1.75 shd1
10.0.1.98 shd2
10.0.1.131 shd3
```



```
## Substitute the values by your own IPs
## Do this step on cata, shd1, shd2 and shd3 !!!
```

For each of the shard host (**shard1**, **shard2**, **shard3**), open 1521 port:

```
ssh -i privateKey opc@<shd1 host public IP>
## Use sudo and firewall command to open port 1521

[opc@shd1 ~]$ sudo firewall-cmd --add-port=1521/tcp --permanent
success
[opc@shd1 ~]$ sudo firewall-cmd --reload
success
[opc@shd1 ~]$ sudo firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: ens3
  sources:
  services: dhcpv6-client ssh
  ports: 1521/tcp
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:

## Repeat these steps on shd2 and shd3 servers !!!
```

On the catalog host (cata), we will install GSM (Global Service Manager). GSM is a component of Oracle Global Data Services solution (GDS), that acts as a global listener on top of all the shards of the sharded database.

As the catalog host will also be the shard director, we need to open port 1522 (shard director default listener port), along with port 1521. For the demo application we will use in this lab, we need to open port 8081 as well.

Connect to the catalog host (cata) as "opc", and open the required ports:

```
## Connect to cata host as opc

ssh -i privateKey opc@<cata host public IP>

## Open the required ports

[opc@cata ~]$ sudo firewall-cmd --add-port=1521/tcp --permanent
success
```



```
[opc@cata ~]$ sudo firewall-cmd --add-port=1522/tcp --permanent
success
[opc@cata ~]$ sudo firewall-cmd --add-port=8081/tcp --permanent
success
[opc@cata ~]$ sudo firewall-cmd --reload
success
[opc@cata ~]$ sudo firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: ens3
  sources:
  services: dhcpv6-client ssh
  ports: 1521/tcp 1522/tcp 8081/tcp
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

On the catalog host, install the shard director software.

```
## Connect to the cata server and gain access to "oracle" user:

[opc@cata ~]$ sudo -i
[root@cata ~]# su - oracle
Last login: Tue Nov  9 11:36:00 GMT 2021 on pts/0

## Create a file named gsm.sh
## This file will be used further to load the GSM environment variables

touch /home/oracle/gsm.sh

## Copy and paste the following 4 lines in your file:

export ORACLE_BASE=/u01/app/oracle
export ORACLE_HOME=/u01/app/oracle/product/19c/gsmhome_1
export LD_LIBRARY_PATH=$ORACLE_HOME/lib
export PATH=$ORACLE_HOME/bin:$PATH

## Create a file named cata.sh
## This file will be used further to load the catalog database environment
variables

touch /home/oracle/cata.sh

## Copy and paste the following 4 lines in your file:
```




```

export ORACLE_BASE=/u01/app/oracle
export ORACLE_HOME=/u01/app/oracle/product/19c/dbhome_1
export LD_LIBRARY_PATH=$ORACLE_HOME/lib
export PATH=$ORACLE_HOME/bin:$PATH

## Load GSM environment and install GSM software:

. ./gsm.sh

## Download GSM distribution

wget https://bit.ly/3bNtHTy -O GSM.19.3.V982067-01.zip

[oracle@cata ~]$ ls -ltr
total 937408
drwxr-xr-x. 12 oracle oinstall      4096 Jul 23  2020 swingbench
-rw-r--r--.  1 oracle oinstall      166 Nov 10 10:20 cata.sh
-rw-r--r--.  1 oracle oinstall      167 Nov 10 16:56 gsm.sh
-rw-r--r--.  1 oracle oinstall 959891519 Nov 11 08:48 GSM.19.3.V982067-01.zip

## Unzip the distribution in /home/oracle

unzip GSM.19.3.V982067-01.zip

[oracle@cata ~]$ ls -ltr
total 937408
drwxr-xr-x.  5 oracle oinstall      90 Apr 17  2019 gsm
drwxr-xr-x. 12 oracle oinstall      4096 Jul 23  2020 swingbench
-rw-r--r--.  1 oracle oinstall      166 Nov 10 10:20 cata.sh
-rw-r--r--.  1 oracle oinstall      167 Nov 10 16:56 gsm.sh
-rw-r--r--.  1 oracle oinstall 959891519 Nov 11 08:48 GSM.19.3.V982067-01.zip

## Edit the ./gsm/response/gsm_install.rsp file. Specify the variables like
following.

    UNIX_GROUP_NAME=oinstall
    INVENTORY_LOCATION=/u01/app/oraInventory
    ORACLE_HOME=/u01/app/oracle/product/19c/gsmhome_1
    ORACLE_BASE=/u01/app/oracle

## Create the gsm home directory

mkdir -p /u01/app/oracle/product/19c/gsmhome_1

## Install the GSM software in silent mode

./gsm/runInstaller -silent -responseFile
/home/oracle/gsm/response/gsm_install.rsp -showProgress -ignorePrereq

Starting Oracle Universal Installer...

Checking Temp space: must be greater than 551 MB.   Actual 32844 MB       Passed
Preparing to launch Oracle Universal Installer from /tmp/OraInstall2021-11-
11_08-55-36AM. Please wait ...[oracle@cata ~]$

```



```

[oracle@cata ~]$ [WARNING] [INS-13014] Target environment does not meet some
optional requirements.
  CAUSE: Some of the optional prerequisites are not met. See logs for details.
/u01/app/oraInventory/logs/installActions2021-11-11_08-55-36AM.log
  ACTION: Identify the list of failed prerequisite checks from the log:
/u01/app/oraInventory/logs/installActions2021-11-11_08-55-36AM.log. Then either
from the log file or from installation manual find the appropriate
configuration to meet the prerequisites and fix it manually.
The response file for this session can be found at:
/u01/app/oracle/product/19c/gsmhome_1/install/response/gsm_2021-11-11_08-55-
36AM.rsp

You can find the log of this install session at:
/u01/app/oraInventory/logs/installActions2021-11-11_08-55-36AM.log

Prepare in progress.
..... 8% Done.

Prepare successful.

Copy files in progress.
..... 13% Done.
..... 19% Done.
..... 27% Done.
..... 33% Done.
..... 38% Done.
..... 43% Done.
..... 48% Done.
..... 53% Done.
..... 58% Done.
..... 64% Done.
..... 69% Done.
..... 74% Done.
..... 79% Done.

Copy files successful.

Link binaries in progress.

Link binaries successful.

Setup files in progress.
.....
Setup files successful.

Setup Inventory in progress.

Setup Inventory successful.
.....
Finish Setup in progress.
..... 84% Done.

Finish Setup successful.

```



```

The installation of Oracle Distributed Service and Load Management was
successful.
Please check '/u01/app/oraInventory/logs/silentInstall2021-11-11_08-55-
36AM.log' for more details.

Setup Oracle Base in progress.

Setup Oracle Base successful.
..... 95% Done.

As a root user, execute the following script(s):
    1. /u01/app/oracle/product/19c/gsmhome_1/root.sh

Successfully Setup Software with warning(s).
..... 100% Done.

## As root, execute the root.sh script

[root@cata ~]# /u01/app/oracle/product/19c/gsmhome_1/root.sh
Check /u01/app/oracle/product/19c/gsmhome_1/install/root_cata_2021-11-11_08-58-
23-323548564.log for the output of root script

## Check the logfile

[root@cata ~]$ cat
/u01/app/oracle/product/19c/gsmhome_1/install/root_cata_2021-11-11_08-58-23-
323548564.log
Performing root user operation.

The following environment variables are set as:
    ORACLE_OWNER= oracle
    ORACLE_HOME= /u01/app/oracle/product/19c/gsmhome_1
Copying dbhome to /usr/local/bin ...
Copying oraenv to /usr/local/bin ...
Copying coraenv to /usr/local/bin ...

Entries will be added to the /etc/oratab file as needed by
Database Configuration Assistant when a database is created
Finished running generic part of root script.
Now product-specific root actions will be performed.

```

Now we will setup the catalog database. Connect back to "oracle" on cata server, and perform the following steps:

```

[root@cata ~]# su - oracle
Last login: Wed Nov 10 10:18:34 GMT 2021 on pts/0

## Load the catalog database environment

```



```
[oracle@cata ~]$ . ./cata.sh

## Because the shard catalog database can run multi-shard queries which connect
to shards over database links, the OPEN_LINKS and OPEN_LINKS_PER_INSTANCE
database initialization parameter values must be greater than or equal to the
number of shards that will be part of the sharded database configuration.

## Review and change some instance parameters

sqlplus / as sysdba

SQL> alter system set open_links=20 scope=spfile;

System altered.

SQL> alter system set open_links_per_instance=20 scope=spfile;

System altered.

SQL> alter system set db_files=1024 scope=spfile;

System altered.

SQL> alter system set db_create_file_dest='/u01/app/oracle/oradata'
scope=spfile;

System altered.

## Unlock the catalog user schema, and change its password

SQL> alter user gsmcatuser account unlock;

User altered.

SQL> alter user gsmcatuser identified by Ora_DB4U;

User altered.

## Connect to the catalog pdb, Unlock the gsmcatalog user and create a shard
catalog administrator account

SQL> show pdbs

  CON_ID CON_NAME          OPEN MODE  RESTRICTED
-----
      2 PDB$SEED             READ ONLY   NO
      3 CATAPDB            READ WRITE  NO

SQL> alter session set container=catapdb;

Session altered.

SQL> alter user gsmcatuser account unlock;
```



```

User altered.

SQL> create user mysdbadmin identified by Ora_DB4U;

User created.

SQL> grant gsmadmin_role to mysdbadmin;

Grant succeeded.

## Connect as sysdba. Check the database archivelog mode and flashback mode

SQL> connect / as sysdba
Connected.
SQL> archive log list
Database log mode          No Archive Mode
Automatic archival         Disabled
Archive destination        /u01/app/oracle/product/19c/dbhome_1/dbs/arch
Oldest online log sequence 16
Current log sequence       18

SQL> select flashback_on from v$database;

FLASHBACK_ON
-----
NO

SQL> show parameter db_recovery_file

NAME                                TYPE        VALUE
-----
db_recovery_file_dest               string
db_recovery_file_dest_size          big integer 0
SQL>

## Enable archivelog and flashback

SQL> !mkdir -p /u01/app/oracle/fast_recovery_area

SQL> alter system set db_recovery_file_dest_size=50G scope=both;

System altered.

SQL> alter system set
db_recovery_file_dest='/u01/app/oracle/fast_recovery_area' scope=both;

System altered.

SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL>
SQL>

```



```
SQL>
SQL> startup mount
ORACLE instance started.

Total System Global Area 4630509232 bytes
Fixed Size          9143984 bytes
Variable Size       855638016 bytes
Database Buffers    3758096384 bytes
Redo Buffers        7630848 bytes
Database mounted.
```

```
SQL> alter database archivelog;
```

Database altered.

```
SQL> alter database flashback on;
```

Database altered.

```
SQL> alter database open;
```

Database altered.

In the following steps, we are going to setup the shards of the sharded database. **These steps need to be performed on shd1, shd2 and shd3.**

```
## Gain access to shd1 as "oracle", and run the following steps
ssh -i privateKey opc@<shd1 public IP>
```

```
sudo su - oracle
```

```
[oracle@shd1 ~]$ sqlplus / as sysdba
```

```
SQL*Plus: Release 19.0.0.0.0 - Production on Wed Nov 10 11:44:28 2021
Version 19.11.0.0.0
```

```
Copyright (c) 1982, 2020, Oracle. All rights reserved.
```

```
Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.11.0.0.0
```

```
SQL> alter user gsmrootuser account unlock;
```

User altered.

```
SQL> alter user gsmrootuser identified by Ora_DB4U;
```

User altered.

```
SQL> grant SYSDG, SYSBACKUP to gsmrootuser;
```



Grant succeeded.

A directory object named DATA_PUMP_DIR must be created and accessible in the shard database from the GSMADMIN_INTERNAL account

```
SQL> select directory_path from dba_directories where
directory_name='DATA_PUMP_DIR';
```

DIRECTORY_PATH

```
-----
-
/u01/app/oracle/admin/shd1/dpdump/
```

```
SQL> grant read, write on directory DATA_PUMP_DIR to gsmadmin_internal;
```

Grant succeeded.

Unlock schema gsmuser

```
SQL> alter user gsmuser account unlock;
```

User altered.

```
SQL> alter user gsmuser identified by Ora_DB4U;
```

User altered.

```
SQL> grant SYSDG, SYSBACKUP to gsmuser;
```

Grant succeeded.

```
SQL> alter system set db_files=1024 scope=spfile;
```

System altered.

```
SQL> alter system set dg_broker_start=true scope=both;
```

System altered.

```
SQL> alter system set db_file_name_convert='/SHDSTB1/', '/SHD1/' scope=spfile;
```

System altered.

```
SQL> show pdbs
```

CON_ID	CON_NAME	OPEN MODE	RESTRICTED
2	PDB\$SEED	READ ONLY	NO
3	SHDPDB1	READ WRITE	NO

```
SQL> alter session set container=shdpdb1;
```

Session altered.



```

SQL> alter user gsmuser account unlock;

User altered.

SQL> grant SYSDG, SYSBACKUP to gsmuser;

Grant succeeded.

SQL> show parameter db_create_file_dest

NAME                                TYPE        VALUE
-----
db_create_file_dest                  string
SQL> alter system set db_create_file_dest='/u01/app/oracle/oradata' scope=both;

System altered.

SQL> grant read, write on directory DATA_PUMP_DIR to gsmadmin_internal;

Grant succeeded.

## Connect to the CDB and enable archivelog and flashback

SQL> connect / as sysdba
Connected.
SQL> !mkdir -p /u01/app/oracle/fast_recovery_area

SQL> alter system set db_recovery_file_dest_size=50G scope=both;

System altered.

SQL> alter system set
db_recovery_file_dest='/u01/app/oracle/fast_recovery_area' scope=both;

System altered.

SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> startup mount
ORACLE instance started.

Total System Global Area 4630509232 bytes
Fixed Size                  9143984 bytes
Variable Size              855638016 bytes
Database Buffers          3758096384 bytes
Redo Buffers                7630848 bytes
Database mounted.

SQL> alter database archivelog;

Database altered.

```




```
SQL> alter database flashback on;
```

Database altered.

```
SQL> alter database open;
```

Database altered.

(Optional) If your shard database will use standby shard databases, you must enable the FORCE LOGGING mode.

In this lab we won't setup Data Guard on the shards, so the next step can be skipped.

```
SQL> alter database force logging;
```

Database altered.

Connect to the shard pdb and validate the shard.

The validateShard procedure can and should be run against primary, mounted (unopened) standby, and Active Data Guard standby databases that are part of the sharded database configuration.

```
SQL> alter session set container=shdpdb1;
```

Session altered.

```
SQL> set serveroutput on
```

```
SQL> execute dbms_gsm_fix.validateShard
```

INFO: Data Guard shard validation requested.

INFO: Database role is PRIMARY.

INFO: Database name is SHD1.

INFO: Database unique name is shd1.

INFO: Database ID is 785208895.

INFO: Database open mode is READ WRITE.

INFO: Database in archivelog mode.

INFO: Flashback is on.

INFO: Force logging is on.

INFO: Database platform is Linux x86 64-bit.

INFO: Database character set is AL32UTF8. This value must match the character set of the catalog database.

INFO: 'compatible' initialization parameter validated successfully.

INFO: Database is a multitenant container database.

INFO: Current container is SHDPDB1.

INFO: Database is using a server parameter file (spfile).

INFO: db_create_file_dest set to: '/u01/app/oracle/oradata'

INFO: db_recovery_file_dest set to: '/u01/app/oracle/fast_recovery_area'

INFO: db_files=1024. Must be greater than the number of chunks and/or tablespaces to be created in the shard.

INFO: dg_broker_start set to TRUE.

INFO: remote_login_passwordfile set to EXCLUSIVE.

INFO: db_file_name_convert set to: '/SHDSTB1/, /SHD1/'

INFO: GSMUSER account validated successfully.

INFO: DATA_PUMP_DIR is



```
 '/u01/app/oracle/admin/shd1/dpdump/D04B91BB4408489EE055000017074120'.
```

```
PL/SQL procedure successfully completed.
```

Repeat these configuration steps on shd2 and shd3 !!!

Once we have configured shd1, shd2 and shd3, we are ready to configure the sharded database topology. Connect to the catalog host (cata), and gain access to the "oracle" user:

```
sudo su - oracle
```

```
## Switch to the GSM environment
```

```
[oracle@cata ~]$ . ./gsm.sh
```

```
## Enter the GSM command line interface.
```

```
[oracle@cata ~]$ gdsctl
```

```
GDSCtl: Version 19.0.0.0.0 - Production on Thu Nov 11 09:18:26 GMT 2021
```

```
Copyright (c) 2011, 2019, Oracle. All rights reserved.
```

```
Welcome to GDSCtl, type "help" for information.
```

```
Warning: GSM is not set automatically because gsm.ora does not contain GSM entries. Use "set gsm" command to set GSM for the session.
```

```
Current GSM is set to GSMORA
```

```
GDSCtl>
```

```
## Create the shard catalog using the System-Managed sharding method. In this workshop, we have no data guard environment, so just set one region.
```

```
## In this workshop, we set the chunks to 12, the default value is 120 for each of the shard database.
```

```
GDSCtl> create shardcatalog -database cata:1521/catapdb -user  
mysdbadmin/Ora_DB4U -chunks 12 -region region1 -SHARDING system
```

```
Catalog is created
```

```
## Add and start the shard director
```

```
GDSCtl> connect mysdbadmin/Ora_DB4U@cata:1521/catapdb  
Catalog connection is established
```

```
GDSCtl> add gsm -gsm shardedirector1 -catalog cata:1521/catapdb -pwd Ora_DB4U -  
region region1
```

```
GSM successfully added
```

```
GDSCtl> start gsm -gsm shardedirector1
```

```
GSM is started successfully
```

```
GDSCtl> set gsm -gsm shardedirector1
```



```

## Add shard group, each shardspace must contain at least one primary
shardgroup and may contain any number or type of standby shardgroups.
## In this workshop, we have only one primary shardgroup.

GDSTCL> add shardgroup -shardgroup shardgroup_primary -deploy_as primary -
region region1
The operation completed successfully

## Verify the Sharding Topology.
## Before adding information about your shard databases to the catalog,
verify that your sharding topology is correct by using some GDSTCL CONFIG
commands.

GDSTCL> config

Regions
-----
region1

GSMs
-----
sharddirector1

Sharded Database
-----
orasdb

Databases
-----

Shard Groups
-----
shardgroup_primary

Shard spaces
-----
shardspaceora

Services
-----

GDSTCL pending requests
-----

```

Command	Object	Status
-----	-----	-----

```

Global properties
-----
Name: oradbcloud
Master GSM: sharddirector1
DDL sequence #: 0

## Add shard CDB. Repeat the ADD CDB command for all of the CDBs that contain a
shard PDB in the configuration. for now, we only add shd1 and shd2.

```



```

GDSTCL> add cdb -connect shd1:1521/shd1 -pwd Ora_DB4U
DB Unique Name: shd1
The operation completed successfully

GDSTCL> add cdb -connect shd2:1521/shd2 -pwd Ora_DB4U
DB Unique Name: shd2
The operation completed successfully

GDSTCL> config cdb
shd1
shd2

## Add the primary shard information to the shard catalog. The shard group is
shardgroup_primary.

GDSTCL> add shard -connect shd1:1521/shdpdb1 -pwd Ora_DB4U -shardgroup
shardgroup_primary -cdb shd1
INFO: Data Guard shard validation requested.
INFO: Database role is PRIMARY.
INFO: Database name is SHD1.
INFO: Database unique name is shd1.
INFO: Database ID is 785208895.
INFO: Database open mode is READ WRITE.
INFO: Database in archivelog mode.
INFO: Flashback is on.
INFO: Force logging is on.
INFO: Database platform is Linux x86 64-bit.
INFO: Database character set is AL32UTF8. This value must match the character
set of the catalog database.
INFO: 'compatible' initialization parameter validated successfully.
INFO: Database is a multitenant container database.
INFO: Current container is SHDPDB1.
INFO: Database is using a server parameter file (spfile).
INFO: db_create_file_dest set to: '/u01/app/oracle/oradata'
INFO: db_recovery_file_dest set to: '/u01/app/oracle/fast_recovery_area'
INFO: db_files=1024. Must be greater than the number of chunks and/or
tablespaces to be created in the shard.
INFO: dg_broker_start set to TRUE.
INFO: remote_login_passwordfile set to EXCLUSIVE.
INFO: db_file_name_convert set to: '/SHDSTB1/, /SHD1/'
INFO: GSMUSER account validated successfully.
INFO: DATA_PUMP_DIR is
'/u01/app/oracle/admin/shd1/dpdump/D04B91BB4408489EE055000017074120'.
DB Unique Name: shd1_shdpdb1
The operation completed successfully

GDSTCL> add shard -connect shd2:1521/shdpdb2 -pwd Ora_DB4U -shardgroup
shardgroup_primary -cdb shd2
INFO: Data Guard shard validation requested.
INFO: Database role is PRIMARY.
INFO: Database name is SHD2.
INFO: Database unique name is shd2.
INFO: Database ID is 1343741747.

```



```

INFO: Database open mode is READ WRITE.
INFO: Database in archivelog mode.
INFO: Flashback is on.
INFO: Force logging is on.
INFO: Database platform is Linux x86 64-bit.
INFO: Database character set is AL32UTF8. This value must match the character
set of the catalog database.
INFO: 'compatible' initialization parameter validated successfully.
INFO: Database is a multitenant container database.
INFO: Current container is SHDPDB2.
INFO: Database is using a server parameter file (spfile).
INFO: db_create_file_dest set to: '/u01/app/oracle/oradata'
INFO: db_recovery_file_dest set to: '/u01/app/oracle/fast_recovery_area'
INFO: db_files=1024. Must be greater than the number of chunks and/or
tablespaces to be created in the shard.
INFO: dg_broker_start set to TRUE.
INFO: remote_login_passwordfile set to EXCLUSIVE.
INFO: db_file_name_convert set to: '/SHDSTB2/, /SHD2/'
INFO: GSMUSER account validated successfully.
INFO: DATA_PUMP_DIR is
'/u01/app/oracle/admin/shd2/dpdump/D04B984A766F48D6E055000017017509'.
DB Unique Name: shd2_shdpdb2
The operation completed successfully

```

Run CONFIG SHARD to view the shard metadata on the shard catalog.

```
GDSTCL> config shard
```

Name	Shard Group	Status	State	Region	Availability
shd1_shdpdb1	shardgroup_primary	U	none	region1	-
shd2_shdpdb2	shardgroup_primary	U	none	region1	-

Add all of the host names and IP addresses of your shard hosts to the shard catalog. First, View a list of trusted hosts.

```
GDSTCL> config vn timer
```

Name	Group ID
10.0.1.125	
shd1	
shd2	

Run the ADD INVITEDNODE command to manually add all host names and IP addresses of your shard hosts to the shard catalog metadata.

```

GDSTCL> add invitednode 127.0.0.1
GDSTCL> add invitednode cata
GDSTCL> add invitednode 10.0.1.75 <===== Substitute by private IP of shd1
GDSTCL> add invitednode 10.0.1.98 <===== Substitute by private IP of shd2
GDSTCL>

```



```

GDSCTL> config vnchr
Name                               Group ID
----                               -
10.0.1.125
10.0.1.75
10.0.1.98
127.0.0.1
cata
shd1
shd2

## Deploy the sharding configuration

## When the sharded database topology has been fully configured, run the GDSCTL
DEPLOY command to deploy the sharded database configuration

GDSCTL> deploy
deploy: examining configuration...
deploy: requesting Data Guard configuration on shards via GSM
deploy: shards configured successfully
The operation completed successfully

## Check the shard status, It may look similar to the following.

GDSCTL> config shard
Name                               Shard Group      Status   State   Region
Availability                      -----
-----
shd1_shdpdb1                      shardgroup_primary Ok        Deployed region1  ONLINE
shd2_shdpdb2                      shardgroup_primary Ok        Deployed region1  ONLINE

## Observe all shard are registered.

GDSCTL> databases
Database: "shd1_shdpdb1" Registered: Y State: Ok ONS: N. Role: PRIMARY
Instances: 1 Region: region1
  Registered instances:
    orasdb%1
Database: "shd2_shdpdb2" Registered: Y State: Ok ONS: N. Role: PRIMARY
Instances: 1 Region: region1
  Registered instances:
    orasdb%11

## Create and start a global service named oltp_rw_srvc that a client can use
to connect to the sharded database.
## The oltp_rw_srvc service runs read/write transactions on the primary shards.

GDSCTL> add service -service oltp_rw_srvc -role primary
The operation completed successfully

GDSCTL> start service -service oltp_rw_srvc
The operation completed successfully

```



Check the status of the service.

```
GDSCTL> config service
```

Name	Network name	Pool	Started	Preferred
all				
----	-----	----	-----	-----
--				
oltp_rw_srvc	oltp_rw_srvc.orasdb.oradbcloud	orasdb	Yes	Yes

Exit the GDSCTL.

```
GDSCTL> exit  
[oracle@cata ~]$
```

Check the shard director listener status. You can see listening on 1522 port there is a service named **oltp_rw_srvc.orasdb.oradbcloud** which we create previously and a service named **GDS\$CATALOG.oradbcloud** which connects to the catalog instance.

```
[oracle@cata ~]$ lsnrctl status SHARDDIRECTOR1
```

LSNRCTL for Linux: Version 19.0.0.0.0 - Production on 11-NOV-2021 10:24:26

Copyright (c) 1991, 2019, Oracle. All rights reserved.

Connecting to
(DESCRIPTION=(ADDRESS=(HOST=cata)(PORT=1522)(PROTOCOL=tcp))(CONNECT_DATA=(SERVICE_NAME=GDS\$CATALOG.oradbcloud)))
STATUS of the LISTENER

```
-----  
Alias                SHARDDIRECTOR1  
Version              TNSLSNR for Linux: Version 19.0.0.0.0 - Production  
Start Date           11-NOV-2021 10:05:22  
Uptime               0 days 0 hr. 19 min. 4 sec  
Trace Level          off  
Security             ON: Local OS Authentication  
SNMP                 OFF
```

Listener Parameter File

/u01/app/oracle/product/19c/gsmhome_1/network/admin/gsm.ora

Listener Log File

/u01/app/oracle/diag/gsm/cata/sharddirector1/alert/log.xml

Listening Endpoints Summary...

(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=cata)(PORT=1522)))

Services Summary...

Service "GDS\$CATALOG.oradbcloud" has 1 instance(s).

Instance "cata", status READY, has 1 handler(s) for this service...

Service "GDS\$COORDINATOR.oradbcloud" has 1 instance(s).

Instance "cata", status READY, has 1 handler(s) for this service...

Service "_MONITOR" has 1 instance(s).



```
Instance "SHARDDIRECTOR1", status READY, has 1 handler(s) for this service...
Service "_PINGER" has 1 instance(s).
Instance "SHARDDIRECTOR1", status READY, has 1 handler(s) for this service...
Service "oltp_rw_srv.orasdb.oradbcloud" has 2 instance(s).
Instance "orasdb%1", status READY, has 1 handler(s) for this service...
Instance "orasdb%11", status READY, has 1 handler(s) for this service...
The command completed successfully
```

Create a non sharded application

In the following steps, we will create a non sharded application on shd3. Then we will convert that application to a sharded application, to illustrate the conversion steps. This might be a pretty common situation to start from a non-sharded existing application, and convert it to sharded to gain scalability.

Connect to the shard3 host, switch to the oracle user:

```
ssh -i privateKey opc@<shard3 public IP>

## Gain access to "oracle" user

sudo su - oracle

## Create a new PDB named NSPDB

[oracle@shd3 ~]$ sqlplus / as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Thu Nov 11 11:39:19 2021
Version 19.11.0.0.0

Copyright (c) 1982, 2020, Oracle. All rights reserved.

Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.11.0.0.0

SQL>
SQL>
SQL> CREATE PLUGGABLE DATABASE nspdb ADMIN USER admin IDENTIFIED BY Ora_DB4U
      DEFAULT TABLESPACE users DATAFILE
      '/u01/app/oracle/oradata/SHD3/nspdb/users01.dbf'
      SIZE 10G AUTOEXTEND ON
      FILE_NAME_CONVERT = ('/pdbseed/', '/nspdb/');

Pluggable database created.
```




```
SQL> alter pluggable database NSPDB open;
```

Pluggable database altered.

```
SQL> show pdbs
```

CON_ID	CON_NAME	OPEN MODE	RESTRICTED
2	PDB\$SEED	READ ONLY	NO
3	SHDPDB3	READ WRITE	NO
4	NSPDB	READ WRITE	NO

Create a service named GDS\$CATALOG.ORADBCLOUD and start it in order to run the demo application correctly

```
SQL> alter session set container = nspdb;
```

Session altered.

```
SQL> BEGIN
  DBMS_SERVICE.create_service(
    service_name => 'GDS$CATALOG.ORADBCLOUD',
    network_name => 'GDS$CATALOG.ORADBCLOUD'
  );
END;
/
```

PL/SQL procedure successfully completed.

```
SQL> BEGIN
  DBMS_SERVICE.start_service(
    service_name => 'GDS$CATALOG.ORADBCLOUD'
  );
END;
/
```

PL/SQL procedure successfully completed.

```
exit
```

Create the demo schema.

Still in the shard3 host with oracle user. Download the SQL script nonshard-app-schema.sql

```
cd /home/oracle
```

```
wget https://bit.ly/3nF6jdz -O nonshard-app-schema.sql
```

```
--2021-11-11 16:40:26-- https://objectstorage.us-ashburn-
1.oraclecloud.com/p/_wAzMJHX9Kz8sFn3kd12KMov3HxTPiAyX0winrn7sbh9T7RXYSsR6f_tyAx
IdYhi/n/c4u04/b/data-management-library-files/o/Oracle%20Sharding/nonshard-app-
schema.sql
Resolving objectstorage.us-ashburn-1.oraclecloud.com (objectstorage.us-ashburn-
1.oraclecloud.com)... 134.70.28.1, 134.70.24.1, 134.70.32.1
```



```

Connecting to objectstorage.us-ashburn-1.oraclecloud.com (objectstorage.us-
ashburn-1.oraclecloud.com)|134.70.28.1|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2938 (2.9K) [application/octet-stream]
Saving to: 'nonshard-app-schema.sql'

100%[=====
=====>] 2,938      --.-K/s   in 0s

2021-11-11 16:40:26 (28.2 MB/s) - 'nonshard-app-schema.sql' saved [2938/2938]

[oracle@shd3 ~]$ ls -ltr
total 8
drwxr-xr-x. 12 oracle oinstall 4096 Jul 23  2020 swingbench
-rw-r--r--.  1 oracle oinstall 2938 Jul  2 19:51 nonshard-app-schema.sql

## Use Sql*Plus to run the script

sqlplus /nolog
@nonshard-app-schema.sql

```

Setup and Run the Demo Application: connect to the **catalog host**, switch to the oracle user.

```

ssh -i privateKey opc@<cata public IP>
sudo su - oracle

## Download the sdb_demo_app.zip file

wget https://bit.ly/3P7Pcg3 -O sdb_demo_app.zip

## Unzip the file. This will create sdb_demo_app directory under the
/home/oracle

unzip sdb_demo_app.zip

## View the content of the nonshard_demo_app_ext.sql. Make sure the connect
string is correct to the non-sharded instance pdb

cd sdb_demo_app/sql
cat nonshard_demo_app_ext.sql

-- Create catalog monitor packages
connect sys/Ora_DB4U@shd3:1521/nspdb as sysdba;

@catalog_monitor.sql

connect app_schema/app_schema@shd3:1521/nspdb;

alter session enable shard ddl;

CREATE OR REPLACE VIEW SAMPLE_ORDERS AS

```



```

SELECT OrderId, CustId, OrderDate, SumTotal FROM
  (SELECT * FROM ORDERS ORDER BY OrderId DESC)
  WHERE ROWNUM < 10;

alter session disable shard ddl;

-- Allow a special query for dbaview
connect sys/Ora_DB4U@shd3:1521/nspdb as sysdba;

-- For demo app purposes
grant shard_monitor_role, gsmadmin_role to app_schema;

alter session enable shard ddl;

create user dbmonuser identified by TEZiPP4MsLLL;
grant connect, alter session, shard_monitor_role, gsmadmin_role to dbmonuser;

grant all privileges on app_schema.products to dbmonuser;
grant read on app_schema.sample_orders to dbmonuser;

alter session disable shard ddl;
-- End workaround

exec dbms_global_views.create_any_view('SAMPLE_ORDERS',
  'APP_SCHEMA.SAMPLE_ORDERS', 'GLOBAL_SAMPLE_ORDERS', 0, 1);

## Use Sql*Plus to run the script

sqlplus /nolog

SQL*Plus: Release 19.0.0.0.0 - Production on Tue Nov 16 16:41:37 2021
Version 19.11.0.0.0

Copyright (c) 1982, 2020, Oracle. All rights reserved.

SQL> @nonshard_demo_app_ext.sql

## The result screen like the following. Ignore the ORA-02521 error because
it's not a shard database.

Connected.
ERROR:
ORA-02521: attempted to enable shard DDL in a non-shard database

Role created.

Grant succeeded.

Grant succeeded.

```



Grant succeeded.

Grant succeeded.

Session altered.

Package created.

No errors.

Package body created.

No errors.

PL/SQL procedure successfully completed.

Type created.

Type created.

Package created.

No errors.

Package body created.

No errors.

Package body created.

No errors.

Grant succeeded.

Grant succeeded.

Grant succeeded.

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.



```
PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

Connected.
ERROR:
ORA-02521: attempted to enable shard DDL in a non-shard database

View created.

Session altered.

Connected.

Grant succeeded.

ERROR:
ORA-02521: attempted to enable shard DDL in a non-shard database

User created.

Grant succeeded.

Grant succeeded.

Grant succeeded.

Session altered.

PL/SQL procedure successfully completed.

## Exit the sqlplus. Then change directory to the sdb_demo_app.
exit

cd /home/oracle/sdb_demo_app

## Review the nonsharddemo.properties file content. Make sure the
connect_string and service name is correct

[oracle@cata sdb_demo_app]$ cat nonsharddemo.properties
```



```

name=demo
connect_string=(ADDRESS_LIST=(LOAD_BALANCE=off)(FAILOVER=on)(ADDRESS=(HOST=shd3
)(PORT=1521)(PROTOCOL=tcp)))
monitor.user=dbmonuser
monitor.pass=TEZiPP4MsLLL
app.service.write=nspdb
app.service.readonly=nspdb
app.user=app_schema
app.pass=app_schema
app.threads=7

## Start the workload by executing command: ./run.sh demo
nonsharddemo.properties

./run.sh demo nonsharddemo.properties

```

RO Queries	RW Queries	RO Failed	RW Failed	APS
133194	22428	0	0	819
135368	22801	0	0	794
137639	23162	0	0	816
139983	23514	0	0	857
142154	23923	0	0	791
144423	24326	0	0	821
146604	24720	0	0	790
148820	25111	0	0	812
151074	25509	0	0	809
153302	25899	0	0	793
155798	26347	0	0	913
158566	26841	0	0	1013
161386	27335	0	0	1019
164235	27820	0	0	1031
167050	28272	0	0	1008
169731	28729	0	0	976
172676	29238	0	0	1078
175631	29737	0	0	1083
178483	30231	0	0	1043
181422	30730	0	0	1074

```

[...]
```

Wait the application run several minutes and press Ctrl-C to exit the application. Remember the values of the APS(transaction per second).

Export the demo data and copy the DMP file.

In this step, you will export the demo application data and copy the dmp file to the catalog and each of the shard hosts. You will import the data to the shard database in the next lab.

Connect to the shard3 host, switch to the oracle user.



```

--- Connect to the shard3 host, switch to the oracle user.

ssh -i privateKey opc@<shd3 public IP>

[opc@shd3 ~]$ sudo su - oracle
Last login: Tue Nov 23 16:10:11 GMT 2021 on pts/0

## Connect to the non-sharded database as app_schema user with SQLPLUS.
## Create a DIRECTORY

sqlplus app_schema/app_schema@shd3:1521/nspdb

create directory demo_pump_dir as '/home/oracle';
exit

## Run the following command to export the demo data
## GROUP_PARTITION_TABLE_DATA: Unloads all partitions as a single operation
producing a single partition of data in the dump file.
## Subsequent imports will not know this was originally made up of multiple
partitions.

expdp app_schema/app_schema@shd3:1521/nspdb directory=demo_pump_dir \
  dumpfile=original.dmp logfile=original.log \
  schemas=app_schema data_options=group_partition_table_data

[...]
Dump file set for APP_SCHEMA.SYS_EXPORT_SCHEMA_01 is:
/home/oracle/original.dmp
Job "APP_SCHEMA"."SYS_EXPORT_SCHEMA_01" successfully completed at Tue Nov 16
17:06:36 2021 elapsed 0 00:01:27

```

Now we will copy the DMP file to cata, shd1 and shd2. First we generate a RSA key pair:

```

## From the shard3 host, create a ssh key pair. Press Enter to accept all the
default values.

[oracle@shd3 sdb_demo_app]$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/oracle/.ssh/id_rsa):
Created directory '/home/oracle/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/oracle/.ssh/id_rsa.
Your public key has been saved in /home/oracle/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:BakV9l9zcUn/FuS8evQKNoE7WhrEsFc9/74QxUGVfmE oracle@shd3
The key's randomart image is:
+---[RSA 2048]---+
|           +o    o** |
|          .oo .  =E= |
|         .o  + o  =*+ |
|        .+ o o  +.+= |

```



```

|      . S . o..o+|
|      o      . .+o.|
|      . + +o .o|
|      = o ooo|
|      o      ..o|
+-----[SHA256]-----+

```

```

[oracle@shd3 sdb_demo_app]$ cat /home/oracle/.ssh/id_rsa.pub
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQDC2EjG8bsTKnvQpjlnDtbdKFUb9X0ik3PRnW99BbDfR0vAiYp
2rojcbMCed2YKzcZr5UX8x7p8HpB3u8Bp/J1wJxW/OVuwW3oaSkQ8QRL60tX6KdyTbwVGwxK0YoUaCb
gYemXVHGa/TjuRY/csSesTBIRuCSL1SPYBBGLC0pOnl84+PDhVsf+TxfeIFK+b0zcevr3y++1Yz96+E
wS66h1RSs9d6QQ/Uf0dx4WQnbxWM5lyXdwyJKInDoBxRgoDgv/+Zo+RCOk2n0SCqqaXwTb6cA8Vimup
7dmd+9e8wPX9Wo0rDI1CfdEjBStBhK2sDTtq+8ju9tXDhguiTZ53LD4f oracle@shd3

```

Now we will copy the public key to cata, shd1 and shd2:

```

--- Connect to cata host and gain access to "oracle" user
ssh -i privateKey opc@<cata public IP>
sudo su - oracle

## Make a .ssh directory and edit the authorized_keys file.

[oracle@cata ~]$ mkdir .ssh
[oracle@cata ~]$ vi .ssh/authorized_keys

## Copy and paste the public key from shd3
## Save and chmod the file

[oracle@cata ~]$ chmod 600 .ssh/authorized_keys

## Repeat this steps on shd1 and shd2 !!!

```

Now we are ready to copy the DMP file on each host, using scp and the generated private key: go back to your shd3 session and scp the DMP file to the other hosts.

```

[oracle@shd3 ~]$ scp original.dmp oracle@cata:/home/oracle
original.dmp
100% 12MB 47.6MB/s 00:00
[oracle@shd3 ~]$ scp original.dmp oracle@shd1:/home/oracle
original.dmp
100% 12MB 49.5MB/s 00:00
[oracle@shd3 ~]$ scp original.dmp oracle@shd2:/home/oracle
original.dmp
100% 12MB 51.4MB/s 00:00

```



Migrate to sharded database

Before the existing database can be migrated to the sharded database, you must decide how to organize the sharded database.

You must decide which tables in the application are sharded and which tables are duplicated tables. In this lab, we have already created a scripts for the sharded demo schema.

It creates a sharded table family: "Customers-->Orders-->LineItems" using the sharding key CustId, and Products is the duplicated table.

Next we will create a sharded schema, and then load the exported data into that sharded schema.

Login to the catalog database host, switch to oracle user.

```
ssh -i privateKey opc@cata public IP>
sudo su - oracle

## Download the sharded demo schema SQL scripts sdb-app-schema.sql

[oracle@cata ~]$ wget https://bit.ly/3yKp59X -O sdb-app-schema.sql

--2021-11-17 09:57:44-- https://objectstorage.us-ashburn-
1.oraclecloud.com/p/ZkoZi3PVSwYGZAscZNDRzOLlqdKypfJEnM15czI6ud6nM5POU8MHkcXHXnp
1NJ27/n/c4u04/b/data-management-library-files/o/Oracle%20Sharding/sdb-app-
schema.sql
Resolving objectstorage.us-ashburn-1.oraclecloud.com (objectstorage.us-ashburn-
1.oraclecloud.com)... 134.70.24.1, 134.70.28.1, 134.70.32.1
Connecting to objectstorage.us-ashburn-1.oraclecloud.com (objectstorage.us-
ashburn-1.oraclecloud.com)|134.70.24.1|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3554 (3.5K) [application/octet-stream]
Saving to: 'sdb-app-schema.sql'

100%[=====
=====] 3,554      --.-K/s   in 0s

2021-11-17 09:57:44 (29.6 MB/s) - 'sdb-app-schema.sql' saved [3554/3554]

[oracle@cata ~]$ ls -ltr
total 955464
drwxr-xr-x.  5 oracle oinstall          90 Apr 17  2019 gsm
drwxr-xr-x. 12 oracle oinstall       4096 Jul 23  2020 swingbench
-rw-r--r--.  1 oracle oinstall      3554 Jul  2 19:51 sdb-app-schema.sql
-rw-r--r--.  1 oracle oinstall   5897389 Jul  6 19:04 sdb_demo_app.zip
-rw-r--r--.  1 oracle oinstall    166 Nov 10 10:20 cata.sh
-rw-r--r--.  1 oracle oinstall 959891519 Nov 11 08:48 GSM.19.3.V982067-01.zip
-rw-r--r--.  1 oracle oinstall    167 Nov 11 09:18 gsm.sh
drwxr-xr-x.  3 oracle oinstall     26 Nov 16 16:39 __MACOSX
drwxr-xr-x.  9 oracle oinstall    4096 Nov 16 16:46 sdb_demo_app
-rw-r-----.  1 oracle oinstall 12582912 Nov 16 17:28 original.dmp

## Use SQLPLUS to run this sql scripts
```



```

## First load the cata database environment

. ./cata.sh

[oracle@cata ~]$ sqlplus /nolog

SQL*Plus: Release 19.0.0.0.0 - Production on Wed Nov 17 09:59:48 2021
Version 19.11.0.0.0

Copyright (c) 1982, 2020, Oracle. All rights reserved.

SQL> @sdb-app-schema.sql
SQL> set termout on
SQL> set time on
10:00:00 SQL> spool /home/oracle/sdb_app_schema.lst
10:00:00 SQL> REM
10:00:00 SQL> REM Connect to the Shard Catalog and Create Schema
10:00:00 SQL> REM
10:00:00 SQL> connect / as sysdba
Connected.
10:00:00 SQL> alter session set container=catapdb;

Session altered.

10:00:00 SQL> alter session enable shard ddl;

Session altered.

10:00:00 SQL> create user app_schema identified by app_schema;

User created.

10:00:01 SQL> grant connect, resource, alter session to app_schema;

Grant succeeded.

10:00:01 SQL> grant execute on dbms_crypto to app_schema;

Grant succeeded.

10:00:01 SQL> grant create table, create procedure, create tablespace, create
materialized view to app_schema;

Grant succeeded.

10:00:01 SQL> grant unlimited tablespace to app_schema;

Grant succeeded.

10:00:01 SQL> grant select_catalog_role to app_schema;

Grant succeeded.

10:00:01 SQL> grant all privileges to app_schema;

```



Grant succeeded.

```
10:00:01 SQL> grant gsmadmin_role to app_schema;
```

Grant succeeded.

```
10:00:01 SQL> grant dba to app_schema;
```

Grant succeeded.

```
10:00:01 SQL>
```

```
10:00:01 SQL>
```

```
10:00:01 SQL> REM
```

```
10:00:01 SQL> REM Create a tablespace set for SHARDED tables
```

```
10:00:01 SQL> REM
```

```
10:00:01 SQL> CREATE TABLESPACE SET TSP_SET_1 using template (datafile size
100m autoextend on next 10M maxsize unlimited extent management local segment
space management auto );
```

Tablespace created.

```
10:00:02 SQL>
```

```
10:00:02 SQL> REM
```

```
10:00:02 SQL> REM Create a tablespace for DUPLICATED tables
```

```
10:00:02 SQL> REM
```

```
10:00:02 SQL> CREATE TABLESPACE products_tsp datafile size 100m autoextend on
next 10M maxsize unlimited extent management local uniform size 1m;
```

Tablespace created.

```
10:00:03 SQL>
```

```
10:00:03 SQL> REM
```

```
10:00:03 SQL> REM Create Sharded and Duplicated tables
```

```
10:00:03 SQL> REM
```

```
10:00:03 SQL> connect app_schema/app_schema@catapdb
```

Connected.

```
10:00:03 SQL> alter session enable shard ddl;
```

Session altered.

```
10:00:03 SQL> REM
```

```
10:00:03 SQL> REM Create a Sharded table for Customers (Root table)
```

```
10:00:03 SQL> REM
```

```
10:00:03 SQL> CREATE SHARDED TABLE Customers
```

```
10:00:03 2 (
```

```
10:00:03 3     CustId    VARCHAR2(60) NOT NULL,
```

```
10:00:03 4     FirstName VARCHAR2(60),
```

```
10:00:03 5     LastName  VARCHAR2(60),
```

```
10:00:03 6     Class    VARCHAR2(10),
```

```
10:00:03 7     Geo      VARCHAR2(8),
```

```
10:00:03 8     CustProfile VARCHAR2(4000),
```

```
10:00:03 9     Passwd    RAW(60),
```

```
10:00:03 10    CONSTRAINT pk_customers PRIMARY KEY (CustId),
```



```

10:00:03 11          CONSTRAINT json_customers CHECK (CustProfile IS JSON)
10:00:03 12 ) TABLESPACE SET TSP_SET_1
10:00:03 13 PARTITION BY CONSISTENT HASH (CustId) PARTITIONS AUTO;

```

Table created.

```

10:00:04 SQL>
10:00:04 SQL> REM
10:00:04 SQL> REM Create a Sharded table for Orders
10:00:04 SQL> REM
10:00:04 SQL> CREATE SHARDED TABLE Orders
10:00:04 2 (
10:00:04 3     OrderId      INTEGER NOT NULL,
10:00:04 4     CustId       VARCHAR2(60) NOT NULL,
10:00:04 5     OrderDate    TIMESTAMP NOT NULL,
10:00:04 6     SumTotal     NUMBER(19,4),
10:00:04 7     Status       CHAR(4),
10:00:04 8     constraint pk_orders primary key (CustId, OrderId),
10:00:04 9     constraint fk_orders_parent foreign key (CustId)
10:00:04 10          references Customers on delete cascade
10:00:04 11 ) partition by reference (fk_orders_parent);

```

Table created.

```

10:00:04 SQL>
10:00:04 SQL> REM
10:00:04 SQL> REM Create the sequence used for the OrderId column
10:00:04 SQL> REM
10:00:04 SQL> CREATE SEQUENCE Orders_Seq;

```

Sequence created.

```

10:00:04 SQL>
10:00:04 SQL> REM
10:00:04 SQL> REM Create a Sharded table for LineItems
10:00:04 SQL> REM
10:00:04 SQL> CREATE SHARDED TABLE LineItems
10:00:04 2 (
10:00:04 3     OrderId      INTEGER NOT NULL,
10:00:04 4     CustId       VARCHAR2(60) NOT NULL,
10:00:04 5     ProductId    INTEGER NOT NULL,
10:00:04 6     Price        NUMBER(19,4),
10:00:04 7     Qty          NUMBER,
10:00:04 8     constraint pk_items primary key (CustId, OrderId,
10:00:04 ProductId),
10:00:04 9     constraint fk_items_parent foreign key (CustId, OrderId)
10:00:04 10          references Orders on delete cascade
10:00:04 11 ) partition by reference (fk_items_parent);

```

Table created.

```

10:00:04 SQL>
10:00:04 SQL> REM
10:00:04 SQL> REM Create Duplicated table for Products

```



```

10:00:04 SQL> REM
10:00:04 SQL> CREATE DUPLICATED TABLE Products
10:00:04 2  (
10:00:04 3      ProductId  INTEGER GENERATED BY DEFAULT AS IDENTITY PRIMARY
KEY,
10:00:04 4      Name      VARCHAR2(128),
10:00:04 5      DescrUri   VARCHAR2(128),
10:00:04 6      LastPrice  NUMBER(19,4)
10:00:04 7  ) TABLESPACE products_tsp;

```

Table created.

```

10:00:05 SQL>
10:00:05 SQL> REM
10:00:05 SQL> REM Create functions for Password creation and checking - used by
the REM demo loader application
10:00:05 SQL> REM
10:00:05 SQL>
10:00:05 SQL> CREATE OR REPLACE FUNCTION PasswCreate(PASSW IN RAW)
10:00:05 2      RETURN RAW
10:00:05 3  IS
10:00:05 4      Salt RAW(8);
10:00:05 5  BEGIN
10:00:05 6      Salt := DBMS_CRYPTO.RANDOMBYTES(8);
10:00:05 7      RETURN UTL_RAW.CONCAT(Salt,
DBMS_CRYPTO.HASH(UTL_RAW.CONCAT(Salt, PASSW), DBMS_CRYPTO.HASH_SH256));
10:00:05 8  END;
10:00:05 9  /

```

Function created.

```

10:00:05 SQL>
10:00:05 SQL> CREATE OR REPLACE FUNCTION PasswCheck(PASSW IN RAW, PHASH IN RAW)
10:00:05 2      RETURN INTEGER IS
10:00:05 3  BEGIN
10:00:05 4      RETURN UTL_RAW.COMPARE(
10:00:05 5          DBMS_CRYPTO.HASH(UTL_RAW.CONCAT(UTL_RAW.SUBSTR(PHASH, 1,
8), PASSW), DBMS_CRYPTO.HASH_SH256),
10:00:05 6          UTL_RAW.SUBSTR(PHASH, 9));
10:00:05 7  END;
10:00:05 8  /

```

Function created.

```

10:00:05 SQL>
10:00:05 SQL> REM
10:00:05 SQL> REM
10:00:05 SQL> select table_name from user_tables;

```

TABLE_NAME

```

-----
-
CUSTOMERS
ORDERS

```



```
LINEITEMS
PRODUCTS
MLOG$_PRODUCTS
RUPD$_PRODUCTS
```

```
6 rows selected.
```

```
10:00:05 SQL> REM
10:00:05 SQL> REM
10:00:05 SQL> spool off
10:00:05 SQL> exit
```

Once we have created the sharded demo schema, we will connect to GSM and perform some checking steps. On cata host, as "oracle", load the GSM environment:

```
[oracle@cata ~]$ . ./gsm.sh
[oracle@cata ~]$
[oracle@cata ~]$
[oracle@cata ~]$ gdsctl
GDSCCTL: Version 19.0.0.0.0 - Production on Wed Nov 17 11:17:51 GMT 2021

Copyright (c) 2011, 2019, Oracle. All rights reserved.

Welcome to GDSCCTL, type "help" for information.

Current GSM is set to SHARDDIRECTOR1

## Run the "show ddl" command to see the last DDL executed on the sharded DB

GDSCCTL> show ddl
Catalog connection is established
id      DDL Text                                     Failed shards
--      -
9       grant dba to app_schema
10      CREATE TABLESPACE SET TSP_SET_1 usin...
11      CREATE TABLESPACE products_tsp datafi...
12      CREATE SHARDED TABLE Customers ( Cu...
13      CREATE SHARDED TABLE Orders ( Order...
14      CREATE SEQUENCE Orders_Seq
15      CREATE SHARDED TABLE LineItems ( Or...
16      CREATE MATERIALIZED VIEW "APP_SCHEMA"...
17      CREATE OR REPLACE FUNCTION PasswCreat...
18      CREATE OR REPLACE FUNCTION PasswCheck...

GDSCCTL>

## Run the config commands as shown below for each of the shards and verify if
there are any DDL error.

GDSCCTL> config shard -shard shd1_shdpdb1
Name: shd1_shdpdb1
Shard Group: shardgroup_primary
```



```
Status: Ok
State: Deployed
Region: region1
Connection string: shd1:1521/shdpdb1
SCAN address:
ONS remote port: 0
Disk Threshold, ms: 20
CPU Threshold, %: 75
Version: 19.0.0.0
Failed DDL:
DDL Error: ---
Failed DDL id:
Availability: ONLINE
Rack:
```

Supported services

Name	Preferred
Status	
----	-----
-	
oltp_rw_srvc	Yes
Enabled	

Show the created chunks.

GDSCTL> **config chunks**

Chunks

Database	From	To
-----	----	--
shd1_shdpdb1	1	6
shd2_shdpdb2	7	12

With Sql*plus, connect to shdpdb1 and check the created tablespaces:

```
[oracle@cata ~]$ ./cata.sh
```

```
[oracle@cata ~]$ sqlplus sys/Ora_DB4U@shd1:1521/shdpdb1 as sysdba
```

```
SQL*Plus: Release 19.0.0.0.0 - Production on Wed Nov 17 13:02:08 2021
Version 19.3.0.0.0
```

```
Copyright (c) 1982, 2019, Oracle. All rights reserved.
```

```
Connected to:
```

```
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.11.0.0.0
```



```
SQL> select TABLESPACE_NAME, BYTES/1024/1024 MB from sys.dba_data_files order
by tablespace_name;
```

TABLESPACE_NAME	MB
C001TSP_SET_1	100
C002TSP_SET_1	100
C003TSP_SET_1	100
C004TSP_SET_1	100
C005TSP_SET_1	100
C006TSP_SET_1	100
PRODUCTS_TSP	100
SYS_AUX	520
SYSTEM	350
TSP_SET_1	100
UNDOTBS1	215
USERS	5

12 rows selected.

Verify that the chunks and chunk tablespaces are created.

```
set linesize 140
column table_name format a20
column tablespace_name format a20
column partition_name format a20
select table_name, partition_name, tablespace_name from dba_tab_partitions
where tablespace_name like 'C%TSP_SET_1' order by tablespace_name;
```

TABLE_NAME	PARTITION_NAME	TABLESPACE_NAME
LINEITEMS	CUSTOMERS_P1	C001TSP_SET_1
CUSTOMERS	CUSTOMERS_P1	C001TSP_SET_1
ORDERS	CUSTOMERS_P1	C001TSP_SET_1
CUSTOMERS	CUSTOMERS_P2	C002TSP_SET_1
ORDERS	CUSTOMERS_P2	C002TSP_SET_1
LINEITEMS	CUSTOMERS_P2	C002TSP_SET_1
CUSTOMERS	CUSTOMERS_P3	C003TSP_SET_1
LINEITEMS	CUSTOMERS_P3	C003TSP_SET_1
ORDERS	CUSTOMERS_P3	C003TSP_SET_1
LINEITEMS	CUSTOMERS_P4	C004TSP_SET_1
CUSTOMERS	CUSTOMERS_P4	C004TSP_SET_1

TABLE_NAME	PARTITION_NAME	TABLESPACE_NAME
ORDERS	CUSTOMERS_P4	C004TSP_SET_1
CUSTOMERS	CUSTOMERS_P5	C005TSP_SET_1
ORDERS	CUSTOMERS_P5	C005TSP_SET_1
LINEITEMS	CUSTOMERS_P5	C005TSP_SET_1
CUSTOMERS	CUSTOMERS_P6	C006TSP_SET_1
ORDERS	CUSTOMERS_P6	C006TSP_SET_1
LINEITEMS	CUSTOMERS_P6	C006TSP_SET_1

18 rows selected.




```
## Connect to shdpdb2 with Sql*Plus
```

```
SQL> connect sys/Ora_DB4U@shd2:1521/shdpdb2 as sysdba  
Connected.
```

```
SQL> select TABLESPACE_NAME, BYTES/1024/1024 MB from sys.dba_data_files order  
by tablespace_name;
```

TABLESPACE_NAME	MB
-----	-----
C007TSP_SET_1	100
C008TSP_SET_1	100
C009TSP_SET_1	100
C00ATSP_SET_1	100
C00BTSP_SET_1	100
C00CTSP_SET_1	100
PRODUCTS_TSP	100
SYSAUX	530
SYSTEM	350
TSP_SET_1	100
UNDOTBS1	215

TABLESPACE_NAME	MB
-----	-----
USERS	5

12 rows selected.

```
SQL> select table_name, partition_name, tablespace_name from dba_tab_partitions  
where tablespace_name like 'C%TSP_SET_1' order by tablespace_name;
```

TABLE_NAME	PARTITION_NAME	TABLESPACE_NAME
-----	-----	-----
ORDERS	CUSTOMERS_P7	C007TSP_SET_1
LINEITEMS	CUSTOMERS_P7	C007TSP_SET_1
CUSTOMERS	CUSTOMERS_P7	C007TSP_SET_1
ORDERS	CUSTOMERS_P8	C008TSP_SET_1
CUSTOMERS	CUSTOMERS_P8	C008TSP_SET_1
LINEITEMS	CUSTOMERS_P8	C008TSP_SET_1
LINEITEMS	CUSTOMERS_P9	C009TSP_SET_1
ORDERS	CUSTOMERS_P9	C009TSP_SET_1
CUSTOMERS	CUSTOMERS_P9	C009TSP_SET_1
LINEITEMS	CUSTOMERS_P10	C00ATSP_SET_1
ORDERS	CUSTOMERS_P10	C00ATSP_SET_1

TABLE_NAME	PARTITION_NAME	TABLESPACE_NAME
-----	-----	-----
CUSTOMERS	CUSTOMERS_P10	C00ATSP_SET_1
ORDERS	CUSTOMERS_P11	C00BTSP_SET_1
LINEITEMS	CUSTOMERS_P11	C00BTSP_SET_1
CUSTOMERS	CUSTOMERS_P11	C00BTSP_SET_1
LINEITEMS	CUSTOMERS_P12	C00CTSP_SET_1
CUSTOMERS	CUSTOMERS_P12	C00CTSP_SET_1
ORDERS	CUSTOMERS_P12	C00CTSP_SET_1



18 rows selected.

Connect to the shardcatalog database

Query the gsmadmin_internal.chunk_loc table to observe that the chunks are uniformly distributed between shards

```
SQL> connect sys/Ora_DB4U@cata:1521/catapdb as sysdba
```

```
column shard format a40
```

```
select a.name Shard,count( b.chunk_number) Number_of_Chunks from
gsmadmin_internal.database a, gsmadmin_internal.chunk_loc b where
a.database_num=b.database_num group by a.name;
```

SHARD	NUMBER_OF_CHUNKS
shd1_shdpdb1	6
shd2_shdpdb2	6

Connect into the appschema/appschema on the catadb, shard1, shard2 databases and verify that the sharded and duplicated tables were created.

```
SQL> connect app_schema/app_schema@cata:1521/catapdb
Connected.
```

```
SQL> select table_name from user_tables;
```

TABLE_NAME
CUSTOMERS
ORDERS
LINEITEMS
PRODUCTS
MLOG\$_PRODUCTS
RUPD\$_PRODUCTS

6 rows selected.

```
SQL> connect app_schema/app_schema@shd1:1521/shdpdb1
Connected.
```

```
SQL> select table_name from user_tables;
```

TABLE_NAME
CUSTOMERS
ORDERS
LINEITEMS
PRODUCTS
USLOG\$_PRODUCTS

```
SQL> connect app_schema/app_schema@shd2:1521/shdpdb2
Connected.
```

```
SQL> select table_name from user_tables;
```



```
TABLE_NAME
-----
CUSTOMERS
ORDERS
LINEITEMS
PRODUCTS
USLOG$_PRODUCTS
```

Now, we will load data into sharded database using the dump file which created in the previous lab.

The duplicated tables reside in the shard catalog, they are always loaded into the shard catalog database using any of available data loading utilities, or plain SQL.

When loading a sharded table, each database shard accommodates a distinct subset of the data set, so the data in each table must be split (partitioned) across shards during the load.

You can use the Oracle Data Pump utility to load the data across database shards in subsets. Data from the source database can be exported into a Data Pump dump file. Then Data Pump import can be run on each shard concurrently by using the same dump file.

Loading the data directly into the database shards is much faster, because each shard is loaded separately.

The Data Pump Import detects that you are importing into a shard and only load rows that belong to that shard.

Use SQLPLUS, connect to the catalog pdb with app_schema user:

```
ssh -i privateKey opc@cata public IP>
sudo su - oracle
. ./cata.sh

## Create a directory
## When shard ddl is enabled, it will be created in catalog db and each of the
sharded db

sqlplus app_schema/app_schema@cata:1521/catapdb

alter session enable shard ddl;
create directory demo_pump_dir as '/home/oracle';
exit

## From the catalog host, run the following command to import the duplicated
table data

[oracle@cata ~]$ impdp app_schema/app_schema@cata:1521/catapdb
directory=demo_pump_dir \
dumpfile=original.dmp logfile=imp.log \
```



```
tables=Products \  
content=DATA_ONLY
```

```
Import: Release 19.0.0.0.0 - Production on Thu Nov 18 10:27:58 2021  
Version 19.11.0.0.0
```

```
Copyright (c) 1982, 2019, Oracle and/or its affiliates. All rights reserved.
```

```
Connected to: Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 -  
Production
```

```
Master table "APP_SCHEMA"."SYS_IMPORT_TABLE_01" successfully loaded/unloaded
```

```
Starting "APP_SCHEMA"."SYS_IMPORT_TABLE_01":
```

```
app_schema/*****@cata:1521/catapdb directory=demo_pump_dir
```

```
dumpfile=original.dmp logfile=imp.log tables=Products content=DATA_ONLY
```

```
*/
```

```
Processing object type SCHEMA_EXPORT/TABLE/TABLE_DATA
```

```
. . imported "APP_SCHEMA"."PRODUCTS" 27.25 KB 480 rows
```

```
Job "APP_SCHEMA"."SYS_IMPORT_TABLE_01" successfully completed at Thu Nov 18
```

```
10:28:13 2021 elapsed 0 00:00:10
```

```
## The data was imported in duplicated table PRODUCTS
```

```
## Run the following command to import data into the shard1 tables
```

```
impdp app_schema/app_schema@shd1:1521/shdpdb1 directory=demo_pump_dir \  
dumpfile=original.dmp logfile=imp.log \  
tables=Customers, Orders, LineItems \  
content=DATA_ONLY
```

```
Import: Release 19.0.0.0.0 - Production on Thu Nov 18 10:29:46 2021  
Version 19.11.0.0.0
```

```
Copyright (c) 1982, 2019, Oracle and/or its affiliates. All rights reserved.
```

```
Connected to: Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 -  
Production
```

```
Master table "APP_SCHEMA"."SYS_IMPORT_TABLE_01" successfully loaded/unloaded
```

```
Starting "APP_SCHEMA"."SYS_IMPORT_TABLE_01":
```

```
app_schema/*****@shd1:1521/shdpdb1 directory=demo_pump_dir
```

```
dumpfile=original.dmp logfile=imp.log tables=Customers, Orders, LineItems
```

```
content=DATA_ONLY
```

```
*/
```

```
Processing object type SCHEMA_EXPORT/TABLE/TABLE_DATA
```

```
. . imported "APP_SCHEMA"."CUSTOMERS" 6.169 MB 13717 out
```

```
of 27430 rows <===== Roughly half of the rows are loaded into shard1
```

```
. . imported "APP_SCHEMA"."ORDERS" 2.118 MB 21188 out
```

```
of 42386 rows
```

```
. . imported "APP_SCHEMA"."LINEITEMS" 3.027 MB 38011 out
```

```
of 76034 rows
```

```
Job "APP_SCHEMA"."SYS_IMPORT_TABLE_01" successfully completed at Thu Nov 18
```

```
10:30:19 2021 elapsed 0 00:00:28
```

```
## Run the following command to load data into shard2 tables.
```



```

[oracle@cata ~]$ impdp app_schema/app_schema@shd2:1521/shdpdb2
directory=demo_pump_dir \
dumpfile=original.dmp logfile=imp.log \
tables=Customers, Orders, LineItems \
content=DATA_ONLY

Import: Release 19.0.0.0.0 - Production on Thu Nov 18 10:31:03 2021
Version 19.11.0.0.0

Copyright (c) 1982, 2019, Oracle and/or its affiliates. All rights reserved.

Connected to: Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 -
Production
Master table "APP_SCHEMA"."SYS_IMPORT_TABLE_01" successfully loaded/unloaded
Starting "APP_SCHEMA"."SYS_IMPORT_TABLE_01":
app_schema/*****@shd2:1521/shdpdb2 directory=demo_pump_dir
dumpfile=original.dmp logfile=imp.log tables=Customers, Orders, LineItems
content=DATA_ONLY
*/
Processing object type SCHEMA_EXPORT/TABLE/TABLE_DATA
. . imported "APP_SCHEMA"."CUSTOMERS"          6.169 MB   13713 out
of 27430 rows    <===== Roughly half of the rows are loaded into shard2
. . imported "APP_SCHEMA"."ORDERS"             2.118 MB   21198 out
of 42386 rows
. . imported "APP_SCHEMA"."LINEITEMS"          3.027 MB   38023 out
of 76034 rows
Job "APP_SCHEMA"."SYS_IMPORT_TABLE_01" successfully completed at Thu Nov 18
10:31:37 2021 elapsed 0 00:00:30

```

Setup and Run the Demo Application

Migrate application to the sharded database requires a slight change to the application code. This will be illustrated in a further chapter.

In this workshop, the demo application is designed for sharded database. You need to create additional objects needed by the demo application.

```

## From the catalog host, as "oracle" user, make sure your are in the catalog
environment.

cd ~/sdb_demo_app/sql

## View the content of the sdb_demo_app_ext.sql. Make sure the connect string
is correct.

cat sdb_demo_app_ext.sql

-- Create catalog monitor packages
connect / as sysdba
alter session set container=catapdb;

```



```

@catalog_monitor.sql

connect app_schema/app_schema@cata:1521/catapdb;

alter session enable shard ddl;

CREATE OR REPLACE VIEW SAMPLE_ORDERS AS
  SELECT OrderId, CustId, OrderDate, SumTotal FROM
    (SELECT * FROM ORDERS ORDER BY OrderId DESC)
    WHERE ROWNUM < 10;

alter session disable shard ddl;

-- Allow a special query for dbaview
connect / as sysdba
alter session set container=catapdb;

-- For demo app purposes
grant shard_monitor_role, gsmadmin_role to app_schema;

alter session enable shard ddl;

create user dbmonuser identified by TEZiPP4MsLLL;
grant connect, alter session, shard_monitor_role, gsmadmin_role to dbmonuser;

grant all privileges on app_schema.products to dbmonuser;
grant read on app_schema.sample_orders to dbmonuser;

alter session disable shard ddl;
-- End workaround

exec dbms_global_views.create_any_view('SAMPLE_ORDERS',
'APP_SCHEMA.SAMPLE_ORDERS', 'GLOBAL_SAMPLE_ORDERS', 0, 1);

## Connect to Sql*Plus and run the script

sqlplus /nolog
@sdb_demo_app_ext.sql
exit

## Change directory to the sdb_demo_app and review sdbdemo.properties file

cd ~/sdb_demo_app

[oracle@cata sdb_demo_app]$ cat sdbdemo.properties
name=demo
connect_string=(ADDRESS_LIST=(LOAD_BALANCE=off)(FAILOVER=on)(ADDRESS=(HOST=loca
lhost)(PORT=1522)(PROTOCOL=tcp)))
monitor.user=dbmonuser
monitor.pass=TEZiPP4MsLLL
#app.service.write=oltp_rw_srvc.cust_sdb.oradbcloud
app.service.write=oltp_rw_srvc.orasdb.oradbcloud
#app.service.readonly=oltp_rw_srvc.cust_sdb.oradbcloud
app.service.readonly=oltp_rw_srvc.orasdb.oradbcloud

```



```
app.user=app_schema
app.pass=app_schema
app.threads=7
```

```
## Start the workload by executing the command:
```

```
./run.sh demo sdbdemo.properties
```

RO Queries	RW Queries	RO Failed	RW Failed	APS
217471	37038	0	0	1338
221685	37782	0	0	1573
226016	38510	0	0	1620
230522	39264	0	0	1697
235169	39980	0	0	1749
239513	40703	0	0	1653
243503	41472	0	0	1488
247781	42238	0	0	1617
252090	43001	0	0	1628
256371	43791	0	0	1635
260649	44523	0	0	1604
264750	45230	0	0	1540
268903	45943	0	0	1587
273267	46659	0	0	1689
277461	47343	0	0	1591
281727	48096	0	0	1619
286116	48777	0	0	1651

```
## Compare the APS values with the ones obtained with the non-sharded application.
```

```
## Open another terminal, connect to the catalog host, switch to oracle user. Change the directory to sdb_demo_app.
```

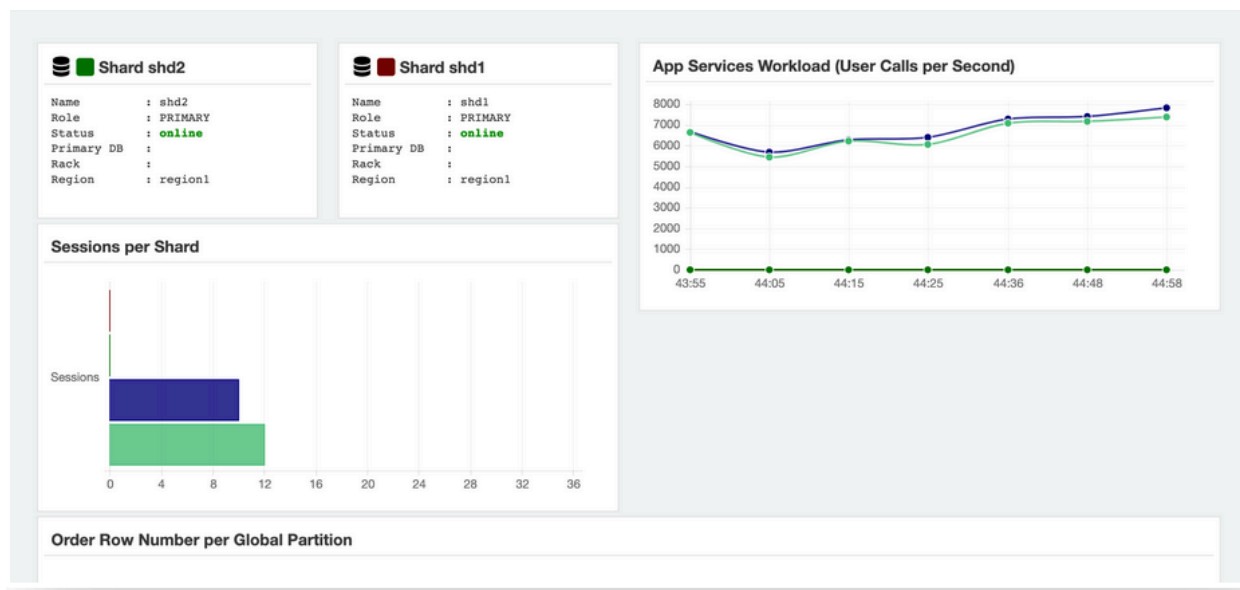
```
## Start the monitoring tool via the following command. Ignore the FileNotFoundException message. (Note: due to the resource limit, start monitor may impact the demo application performance).
```

```
./run.sh monitor sdbdemo.properties
```

Now you can access to the monitor application through the URL: <http://<catalog host public IP>:8081/>

This allows you to monitor the shards, for example:





Hit Ctrl-C to stop both the application and the monitor commands.

Database requests routing to shards

In the following chapter, we will see how to route SQL statements directly to shards. For clarity, we will use Sql*Plus, but the same kind of concepts apply for any application working with a sharded database.

Connect to cata host and gain access to "oracle" user. Load the catalog database environment:

```
ssh -i privateKey opc@<cata public IP>
sudo su - oracle

. ./cata.sh

## Connect to the sharded database with a known sharding key
## Observer the "SHARDING_KEY" clause at the end of the connection string

[oracle@cata ~]$ sqlplus
app_schema/app_schema@'(description=(address=(protocol=tcp)(host=cata)(port=152
2))(connect_data=(service_name=oltp_rw_srvc.orasdb.oradbcloud)(region=region1)(
SHARDING_KEY=tom.edwards@y.bogus)))'

SQL*Plus: Release 19.0.0.0.0 - Production on Thu Nov 18 11:10:43 2021
Version 19.11.0.0.0

Copyright (c) 1982, 2020, Oracle. All rights reserved.

Last Successful login time: Thu Nov 18 2021 10:45:45 +00:00
```




```

Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.11.0.0.0

SQL> select instance_name from v$instance;

INSTANCE_NAME
-----
shd1

## You have been connected to shard1: this is because tom.edwards@y.bogus hash
value corresponds to shard1 !!!

SQL> INSERT INTO Customers (CustId, FirstName, LastName, CustProfile, Class,
Geo, Passwd) VALUES ('tom.edwards@y.bogus', 'Tom', 'Edwards', NULL, 'Gold',
'east', hextoraw('8d1c00e'));

1 row created.

SQL> commit;

Commit complete.

## Select from the customer table. You can see there is one record which you
just insert in the table

SQL> select * from customers where custid like '%y.bogus';

CUSTID
-----
FIRSTNAME
-----
LASTNAME                                CLASS    GEO
-----
-
CUSTPROFILE
-----
-
PASSWD
-----
-
tom.edwards@y.bogus
Tom
Edwards                                Gold     east
CUSTID
-----
FIRSTNAME
-----
LASTNAME                                CLASS    GEO
-----
-
CUSTPROFILE
-----

```



```

-----
-
PASSWD
-----
-

08D1C00E

## Exit and connect to a shard with another shard key.

[oracle@cata ~]$ sqlplus
app_schema/app_schema@'(description=(address=(protocol=tcp)(host=cata)(port=152
2))(connect_data=(service_name=oltp_rw_srvc.orasdb.oradbcloud)(region=region1)(
SHARDING_KEY=james.parker@y.bogus)))'

SQL*Plus: Release 19.0.0.0.0 - Production on Thu Nov 18 11:13:58 2021
Version 19.11.0.0.0

Copyright (c) 1982, 2020, Oracle. All rights reserved.

Last Successful login time: Thu Nov 18 2021 10:45:45 +00:00

Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.11.0.0.0

SQL> select instance_name from v$instance;

INSTANCE_NAME
-----
shd2

SQL> INSERT INTO Customers (CustId, FirstName, LastName, CustProfile, Class,
Geo, Passwd) VALUES ('james.parker@y.bogus', 'James', 'Parker', NULL, 'Gold',
'west', hextoraw('9a3b00c'));

1 row created.

SQL> commit;

Commit complete.

SQL> select * from customers where custid like '%y.bogus';

CUSTID
-----
FIRSTNAME
-----
LASTNAME                                CLASS    GEO
-----
CUSTPROFILE
-----
-

```



```

PASSWD
-----
-
james.parker@y.bogus
James
Parker                                Gold    west

CUSTID
-----
FIRSTNAME
-----
LASTNAME                                CLASS   GEO
-----
-
CUSTPROFILE
-----
-
PASSWD
-----
-

09A3B00C

```

Depending on the value of your sharding key, you are routed to a shard or another by the shard director.

In the next steps, we will illustrate Routing Queries and DMLs by Proxy: connect to the shardcatalog (coordinator database) using the GDS\$CATALOG service (from catalog or any shard host):

```

[oracle@cata ~]$ sqlplus
app_schema/app_schema@cata:1522/GDS\GDS$CATALOG.oradbccloud

SQL*Plus: Release 19.0.0.0.0 - Production on Thu Nov 18 11:22:30 2021
Version 19.11.0.0.0

Copyright (c) 1982, 2020, Oracle. All rights reserved.

Last Successful login time: Thu Nov 18 2021 11:22:24 +00:00

Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.11.0.0.0

SQL> select instance_name from v$instance;

INSTANCE_NAME
-----
cata

SQL> select custid from customers where custid like '%y.bogus';

```



```

CUSTID
-----
tom.edwards@y.bogus
james.parker@y.bogus

## The query returns a consolidated set of rows, one row in each shard.

exit

```

Now we will illustrate a multi-shard query use case. A multi-shard query is a query that must scan data from more than one shard, and the processing on each shard is independent of any other shard.

A multi-shard query maps to more than one shard and the coordinator might need to do some processing before sending the result to the client.

The inline query block is mapped to every shard just as a remote mapped query block (SHARD ITERATOR operation).

The coordinator performs further aggregation and GROUP BY on top of the result set from all shards. The unit of execution on every shard is the inline query block.

From the catalog host, connect to the catalog database:

```

## Let's run a multi-shard query which joins sharded and duplicated table (join
on non sharding key) to get the fast moving products (qty sold > 10).
The output that you will observe will be different (due to data load
randomization).

sqlplus app_schema/app_schema@catapdb

set echo on
set lines 255
column name format a40
explain plan for SELECT name, SUM(qty) qtysold FROM lineitems l, products p
WHERE l.productid = p.productid GROUP BY name HAVING sum(qty) > 10 ORDER BY
qtysold desc;

SQL> set echo off
SQL> select * from table(dbms_xplan.display());

PLAN_TABLE_OUTPUT
-----
-
Plan hash value: 2044377012

-----
-
-----

| Id | Operation          | Name                | Rows  | Bytes | Cost (%CPU)| Ti
me      | Inst|IN-OUT|

```



```

-----
-
-----

PLAN_TABLE_OUTPUT
-----
-
| 0 | SELECT STATEMENT | | 1 | 79 | 4 (50)|
00
:00:01 | | |
| 1 | SORT ORDER BY | | 1 | 79 | 4 (50)|
00
:00:01 | | |
|* 2 | HASH GROUP BY | | 1 | 79 | 4 (50)|
00
:00:01 | | |
| 3 | VIEW | VW_SHARD_372F2D25 | 1 | 79 | 4 (50)|
00
:00:01 | | |

PLAN_TABLE_OUTPUT
-----
-
| 4 | SHARD ITERATOR | | | |
| 5 | REMOTE | | | |
| ORA_S~ | R->S |

-----
-
-----

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

2 - filter(SUM("ITEM_1")>10)

Remote SQL Information (identified by operation id):
-----

5 - EXPLAIN PLAN INTO PLAN_TABLE@! FOR SELECT SUM("A2"."QTY"), "A1"."NAME"
FRO

```



```
M "LINEITEMS"
```

```
PLAN_TABLE_OUTPUT
```

```
-----  
-  
      "A2","PRODUCTS" "A1" WHERE "A2"."PRODUCTID"="A1"."PRODUCTID" GROUP BY  
"A1  
"."NAME" /*  
  
      coord_sql_id=g415vyfr9rg2a */ (accessing  
'ORA_SHARD_POOL@ORA_MULTI_TARGET'  
T' )
```

```
25 rows selected.
```

```
SQL> SELECT name, SUM(qty) qtysold FROM lineitems l, products p WHERE  
l.productid = p.productid GROUP BY name HAVING sum(qty) > 10 ORDER BY qtysold  
desc;
```

NAME	QTYSOLD
Fuel tank	1823
Thermostat	1772
Distributor	1734
Radiator	1718
Fastener	1704
Center console	1698
Master cylinder	1685
seal	1677
Starter motor	1672
Battery	1607
Engine block	1558
[...]	
Pinion bearing	722
Ammeter	721
Power steering	717
Oil pump	715
Suspension link and bolt	705
Engine shake damper and vibration absorber	691
Coil wire	685

```
469 rows selected.
```

```
## Let's run a multi-shard query which runs an IN subquery to get orders that  
includes product with price > 900000.
```

```
set echo on
```



```
column name format a20
explain plan for SELECT COUNT(orderid) FROM orders o WHERE orderid IN (SELECT
orderid FROM lineitems l, products p WHERE l.productid = p.productid AND
o.custid = l.custid AND p.lastprice > 900000);
```

```
set echo off lines 120
select * from table(dbms_xplan.display());
```

PLAN_TABLE_OUTPUT

Plan hash value: 2403723386

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
Inst	IN-OUT					
0	SELECT STATEMENT		1	13	2 (0)	
1	SORT AGGREGATE		1	13		
2	VIEW	VW_SHARD_72AE2D8F	1	13	2 (0)	
3	SHARD ITERATOR					
4	REMOTE					ORA_S~
R->S						

PLAN_TABLE_OUTPUT

Remote SQL Information (identified by operation id):

```
4 - EXPLAIN PLAN INTO PLAN_TABLE@! FOR SELECT COUNT(*) FROM "ORDERS" "A1"
WHERE
      "A1"."ORDERID"=ANY (SELECT "A3"."ORDERID" FROM "LINEITEMS"
"A3", "PRODUCTS" "A2" WHERE
      "A3"."PRODUCTID"="A2"."PRODUCTID" AND "A1"."CUSTID"="A3"."CUSTID" AND
"A2"."LASTPRICE">900000)
/* coord_sql_id=ff5nrpzr2ddnf */ (accessing
'ORA_SHARD_POOL@ORA_MULTI_TARGET' )
```

20 rows selected.

The following query may take some minutes, be patient !



```
SQL> SELECT COUNT(orderid) FROM orders o WHERE orderid IN (SELECT orderid FROM
lineitems l, products p WHERE l.productid = p.productid AND o.custid = l.custid
AND p.lastprice > 900000);
```

```
COUNT(ORDERID)
```

```
-----
7860
```

Let's run a multi-shard query that calculates customer distribution based on the number of orders placed. Please wait several minutes for the results return.

```
set echo off
column name format a40
explain plan for SELECT ordercount, COUNT(*) as custdist
FROM (SELECT c.custid, COUNT(orderid) ordercount
FROM customers c LEFT OUTER JOIN orders o
ON c.custid = o.custid AND
orderdate BETWEEN sysdate-4 AND sysdate GROUP BY c.custid)
GROUP BY ordercount
ORDER BY custdist desc, ordercount desc;
```

```
select * from table(dbms_xplan.display());
```

```
PLAN_TABLE_OUTPUT
```

```
-----
Plan hash value: 313106859
```

```
-----
| Id | Operation          | Name                | Rows  | Bytes | Cost (%CPU)| Time |
| Inst | IN-OUT | | | | | |
|---|---|---|---|---|---|---|
| 0 | SELECT STATEMENT    |                     |      1 |    13 |    5  (20)|      |
00:00:01 |         |                     |      1 |    13 |    5  (20)|      |
| 1 | SORT ORDER BY       |                     |      1 |    13 |    5  (20)|      |
00:00:01 |         |                     |      1 |    13 |    5  (20)|      |
| 2 | HASH GROUP BY       |                     |      1 |    13 |    5  (20)|      |
00:00:01 |         |                     |      1 |    13 |    5  (20)| 00:00:01
| 3 | VIEW                |                     |      1 |    13 |    5  (20)|      |
| 4 | HASH GROUP BY       |                     |      1 |    45 |    5  (20)|      |
00:00:01 |         |                     |      1 |    45 |    5  (20)|      |
| 5 | VIEW                | VW_SHARD_28C476E6 |      1 |    45 |    5  (20)|      |
00:00:01 |         |
```

```
PLAN_TABLE_OUTPUT
```

```
-----
| 6 | SHARD ITERATOR      |                     |      1 |    45 |    5  (20)|      |
| 7 |
```




```
| 7 | REMOTE | | | |
| ORA_S~ | R->S |
```

 Remote SQL Information (identified by operation id):

```
7 - EXPLAIN PLAN INTO PLAN_TABLE@! FOR SELECT
COUNT("A1"."ORDERID"), "A2"."CUSTID" FROM
  "CUSTOMERS" "A2", "ORDERS" "A1" WHERE "A2"."CUSTID"="A1"."CUSTID"(+) AND
  "A1"."ORDERDATE"(+) >= CAST(SYSDATE@! -4 AS TIMESTAMP) AND
  "A1"."ORDERDATE"(+) <= CAST(SYSDATE@! AS
    TIMESTAMP) GROUP BY "A2"."CUSTID" /* coord_sql_id=972ysbafqgcav */
(accessing
```

PLAN_TABLE_OUTPUT

 'ORA_SHARD_POOL@ORA_MULTI_TARGET')

Note

 - dynamic statistics used: dynamic sampling (level=2)

28 rows selected.

```
SQL> SELECT ordercount, COUNT(*) as custdist
FROM (SELECT c.custid, COUNT(orderid) ordercount
FROM customers c LEFT OUTER JOIN orders o
ON c.custid = o.custid AND
orderdate BETWEEN sysdate-4 AND sysdate GROUP BY c.custid)
GROUP BY ordercount
ORDER BY custdist desc, ordercount desc;
```

ORDERCOUNT	CUSTDIST
1	58516
2	21298
3	8151
4	3335
5	1468
6	752
7	417
8	242
9	160
10	120
11	69
12	67
13	50



16	27
15	27
14	18
18	16
17	13
19	11
20	10
21	7
32	6
ORDERCOUNT	CUSTDIST
-----	-----
0	6
22	5
24	4
38	3
27	3
26	3
25	3
47	2
42	2
40	2
37	2
ORDERCOUNT	CUSTDIST
-----	-----
28	2
23	2
59	1
58	1
54	1
51	1
46	1
41	1
39	1
36	1
35	1
ORDERCOUNT	CUSTDIST
-----	-----
33	1
30	1
29	1

47 rows selected.

Sharded database dynamic scaling

Now, we will add the shard (on shd3) to the Shard Database and thus elastically scale the sharded database.



We will see that we can add a new shard to the database without downtime. The chunks and data will be dynamically re-balanced on the new shards, to achieve even data distribution.

To add a new shard to the sharded database, connect to the catalog database host. Switch to oracle user.

```
ssh -i privateKey opc@<cata public IP>
sudo su - oracle
## Load the GSM environment, and connect to gdsctl

. ./gsm.sh
[oracle@cata ~]$ gdsctl
GDSCTL: Version 19.0.0.0.0 - Production on Thu Nov 18 11:39:03 GMT 2021

Copyright (c) 2011, 2019, Oracle. All rights reserved.

Welcome to GDSCTL, type "help" for information.

Current GSM is set to SHARDDIRECTOR1

GDSCTL> config shard
Catalog connection is established
Name          Shard Group      Status   State      Region
Availability   -----
-----
shd1_shdpdb1    shardgroup_primary Ok        Deployed   region1    ONLINE
shd2_shdpdb2    shardgroup_primary Ok        Deployed   region1    ONLINE

## Add the new shard CDB

GDSCTL> add cdb -connect shd3:1521/shd3 -pwd Ora_DB4U
DB Unique Name: shd3
The operation completed successfully

GDSCTL> config cdb
shd1
shd2
shd3

## Add the new shard PDB

GDSCTL> add shard -connect shd3:1521/shdpdb3 -pwd Ora_DB4U -shardgroup
shardgroup_primary -cdb shd3
INFO: Data Guard shard validation requested.
INFO: Database role is PRIMARY.
INFO: Database name is SHD3.
INFO: Database unique name is shd3.
INFO: Database ID is 1393551348.
INFO: Database open mode is READ WRITE.
INFO: Database in archivelog mode.
INFO: Flashback is on.
INFO: Force logging is on.
```



```

INFO: Database platform is Linux x86 64-bit.
INFO: Database character set is AL32UTF8. This value must match the character
set of the catalog database.
INFO: 'compatible' initialization parameter validated successfully.
INFO: Database is a multitenant container database.
INFO: Current container is SHDPDB3.
INFO: Database is using a server parameter file (spfile).
INFO: db_create_file_dest set to: '/u01/app/oracle/oradata'
INFO: db_recovery_file_dest set to: '/u01/app/oracle/fast_recovery_area'
INFO: db_files=1024. Must be greater than the number of chunks and/or
tablespaces to be created in the shard.
INFO: dg_broker_start set to TRUE.
INFO: remote_login_passwordfile set to EXCLUSIVE.
INFO: db_file_name_convert set to: '/SHDSTB3/, /SHD3/'
INFO: GSMUSER account validated successfully.
INFO: DATA_PUMP_DIR is
'/u01/app/oracle/admin/shd3/dpdump/D04B9ECB98A14919E05502001701C873'.
DB Unique Name: shd3_shdpdb3
The operation completed successfully

```

```
GDSTCL> config shard
```

Name	Shard Group	Status	State	Region	Availability
shd1_shdpdb1	shardgroup_primary	Ok	Deployed	region1	ONLINE
shd2_shdpdb2	shardgroup_primary	Ok	Deployed	region1	ONLINE
shd3_shdpdb3	shardgroup_primary	U	none	region1	-

```
## View a list of trusted hosts.
```

```
GDSTCL> config vn timer
```

Name	Group ID
10.0.1.125	
10.0.1.75	
10.0.1.98	
127.0.0.1	
cata	
shd1	
shd2	
shd3	

```
## The host name of shard3 is already there. Manually add shard3 private IP
addresses to the shard catalog metadata.
```

```
GDSTCL> add invitednode 10.0.1.131 <===== substitute by your shd3 private IP
```

```
GDSTCL>
```

```
GDSTCL>
```

```
GDSTCL> config vn timer
```

Name	Group ID
10.0.1.125	
10.0.1.131	



```
10.0.1.75
```

```
10.0.1.98
```

```
127.0.0.1
```

```
cata
```

```
shd1
```

```
shd2
```

```
shd3
```

```
## Deploy and Verify the New Shard.
```

```
GDSTCL> deploy
```

```
Catalog connection is established
```

```
deploy: examining configuration...
```

```
deploy: requesting Data Guard configuration on shards via GSM
```

```
deploy: shards configured; background operations in progress
```

```
The operation completed successfully
```

```
GDSTCL>
```

```
GDSTCL> config shard
```

Name	Shard Group	Status	State	Region	
Availability					
----	-----	-----	-----	-----	-----
shd1_shdpdb1	shardgroup_primary	Ok	Deployed	region1	ONLINE
shd2_shdpdb2	shardgroup_primary	Ok	Deployed	region1	ONLINE
shd3_shdpdb3	shardgroup_primary	Ok	Deployed	region1	ONLINE

```
## Run the following command every minute or two to see the progress of  
automatic rebalancing of chunks. You can see there are 4 chunks need to move to  
the third shard.
```

```
GDSTCL> config chunks -show_reshard
```

```
Chunks
```

```
-----
```

Database	From	To
-----	----	--
shd1_shdpdb1	1	5
shd2_shdpdb2	7	12
shd3_shdpdb3	6	6

```
Ongoing chunk movement
```

```
-----
```

Chunk	Source	Target	status
-----	-----	-----	-----
5	shd1_shdpdb1	shd3_shdpdb3	scheduled
6	shd1_shdpdb1	shd3_shdpdb3	Running
11	shd2_shdpdb2	shd3_shdpdb3	scheduled
12	shd2_shdpdb2	shd3_shdpdb3	scheduled

```
GDSTCL> config chunks -show_reshard
```

```
Chunks
```

```
-----
```

Database	From	To
-----	----	--
shd1_shdpdb1	1	5



```

shd2_shdpdb2          7          12
shd3_shdpdb3          6          6

Ongoing chunk movement
-----
Chunk      Source                Target                status
-----
5          shd1_shdpdb1          shd3_shdpdb3          Running
11         shd2_shdpdb2          shd3_shdpdb3          scheduled
12         shd2_shdpdb2          shd3_shdpdb3          scheduled

```

```

GDSCCTL> config chunks -show_reshard
Chunks
-----

```

```

Database          From      To
-----
shd1_shdpdb1      1        4
shd2_shdpdb2      7        12
shd3_shdpdb3      5        6

```

```

Ongoing chunk movement
-----
Chunk      Source                Target                status
-----
11         shd2_shdpdb2          shd3_shdpdb3          scheduled
12         shd2_shdpdb2          shd3_shdpdb3          Running

```

```

GDSCCTL> config chunks -show_reshard
Chunks
-----

```

```

Database          From      To
-----
shd1_shdpdb1      1        4
shd2_shdpdb2      7        11
shd3_shdpdb3      5        6
shd3_shdpdb3      12       12

```

```

Ongoing chunk movement
-----
Chunk      Source                Target                status
-----
11         shd2_shdpdb2          shd3_shdpdb3          Running

```

After a few minutes, chunks end up rebalanced on the new shard !!!

```

GDSCCTL> config chunks -show_reshard
Chunks
-----

```

```

Database          From      To
-----
shd1_shdpdb1      1        4
shd2_shdpdb2      7        10
shd3_shdpdb3      5        6
shd3_shdpdb3      11       12

```



Ongoing chunk movement

Chunk	Source	Target	status
-----	-----	-----	-----

Observe that the “databases” are automatically registered.

GDSCTL> **databases**

Database: "shd1_shdpdb1" Registered: Y State: Ok ONS: N. Role: PRIMARY

Instances: 1 Region: region1

Service: "oltp_rw_srvc" Globally started: Y Started: Y

Scan: N Enabled: Y Preferred: Y

Registered instances:

orasdb%1

Database: "shd2_shdpdb2" Registered: Y State: Ok ONS: N. Role: PRIMARY

Instances: 1 Region: region1

Service: "oltp_rw_srvc" Globally started: Y Started: Y

Scan: N Enabled: Y Preferred: Y

Registered instances:

orasdb%11

Database: "shd3_shdpdb3" Registered: Y State: Ok ONS: N. Role: PRIMARY

Instances: 1 Region: region1

Service: "oltp_rw_srvc" Globally started: Y Started: Y

Scan: N Enabled: Y Preferred: Y

Registered instances:

orasdb%21

Observe that the “services” are automatically brought up on the newly added shard.

GDSCTL> **services**

Service "oltp_rw_srvc.orasdb.oradbccloud" has 3 instance(s). Affinity: ANYWHERE

Instance "orasdb%1", name: "shd1", db: "shd1_shdpdb1", region: "region1", status: ready.

Instance "orasdb%11", name: "shd2", db: "shd2_shdpdb2", region: "region1", status: ready.

Instance "orasdb%21", name: "shd3", db: "shd3_shdpdb3", region: "region1", status: ready.

Now, run the demo application again and observe.

Manually update the monitored shard list. The package dbms_global_views is used by the monitor tools to monitor the status of shards.

It will create a public shard_dblink_view and a public dblink to each shard. **If you skip this step, the monitor tools will not show the status of the latest added shard database.**

From the cata host, as "oracle"

. ./cata.sh

[oracle@cata ~]\$ **sqlplus / as sysdba**



```
SQL*Plus: Release 19.0.0.0.0 - Production on Thu Nov 18 13:06:58 2021
Version 19.11.0.0.0
```

```
Copyright (c) 1982, 2020, Oracle. All rights reserved.
```

```
Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.11.0.0.0
```

```
SQL> alter session set container=catapdb;
```

```
Session altered.
```

```
SQL> exec dbms_global_views.create_all_database_links();
```

```
PL/SQL procedure successfully completed.
```

```
exit
```

Now run the demo app and compare the APS results with the previous runs, without shards and with two shards.

```
cd sbd_demo_app
[oracle@cata sbd_demo_app]$ ./run.sh demo sbdemo.properties
```

RO Queries	RW Queries	RO Failed	RW Failed	APS
195539	34027	0	0	1601
199379	34670	0	0	1587
203113	35358	0	0	1524
206903	36066	0	0	1548
210737	36786	0	0	1595
214500	37493	0	0	1544
218492	38189	0	0	1639
222401	38859	0	0	1613
226386	39517	0	0	1635
230349	40210	0	0	1614
234115	40891	0	0	1560
237785	41507	0	0	1536
241644	42146	0	0	1567
245335	42785	0	0	1556

Restart the monitor in another terminal window:

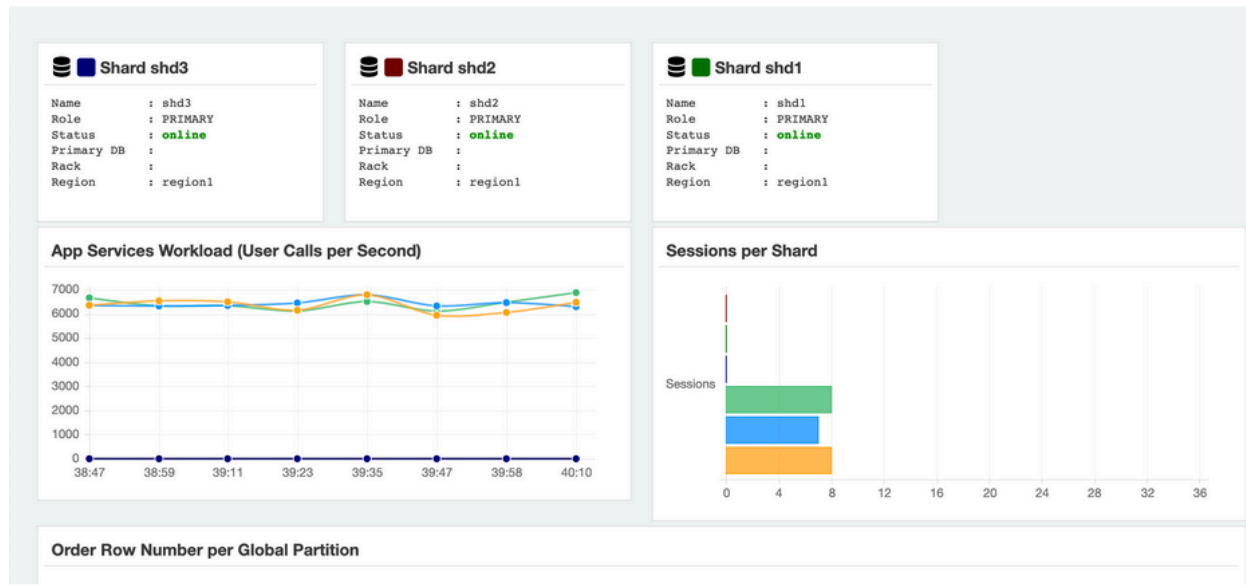
```
## Open another terminal, connect to the catalog host, switch to oracle user.
Change the directory to sbd_demo_app.
```

```
## Start the monitoring tool via the following command. Ignore the
FileNotFoundException message.
(Note: due to the resource limit, start monitor may impact the demo application
performance).
```




```
./run.sh monitor sdbdemo.properties
```

Connect to the monitor tool with the URL: <http://<cata host public IP>:8081/>



You can see the new shard in the monitor tool. The sharded database scaled up horizontally without any downtime.

This concludes the Sharding workshop.

