Introduction to Oracle Sharding - A 12.2 New Feature

Author:
NirmalaDevi Selvakumar, Principal Software Engineer
High Availability, Oracle

This Article is intended to detail the features of Sharding which is available in Oracle 12.2. Oracle Sharding is a database scaling technology based on horizontal partitioning of data across multiple databases.

Table of Contents

What is Sharding?

Oracle Sharding is a Data Tier Architecture in which Data is horizontally partitioned across different databases. It is scalability and availability feature for custom designed OLTP applications that enable distribution and replication of data across a pool of discrete Oracle databases that share no hardware or software. The pool of database is presented to application as a single logical database. Applications elastically scale any level on any platform by adding database (shards) into pool. Oracle Sharding scaled upto 1000 Shards in current version. In regards to replication Sharding supports Oracle Data Guard and Oracle Golden Gate.

Each database in a pool (Shards) runs in dedicated storage and different server. All Shards together makeup the single logical group called Sharded Databases called SDB. Each database in a pool (Shards) contains a table with same column but different

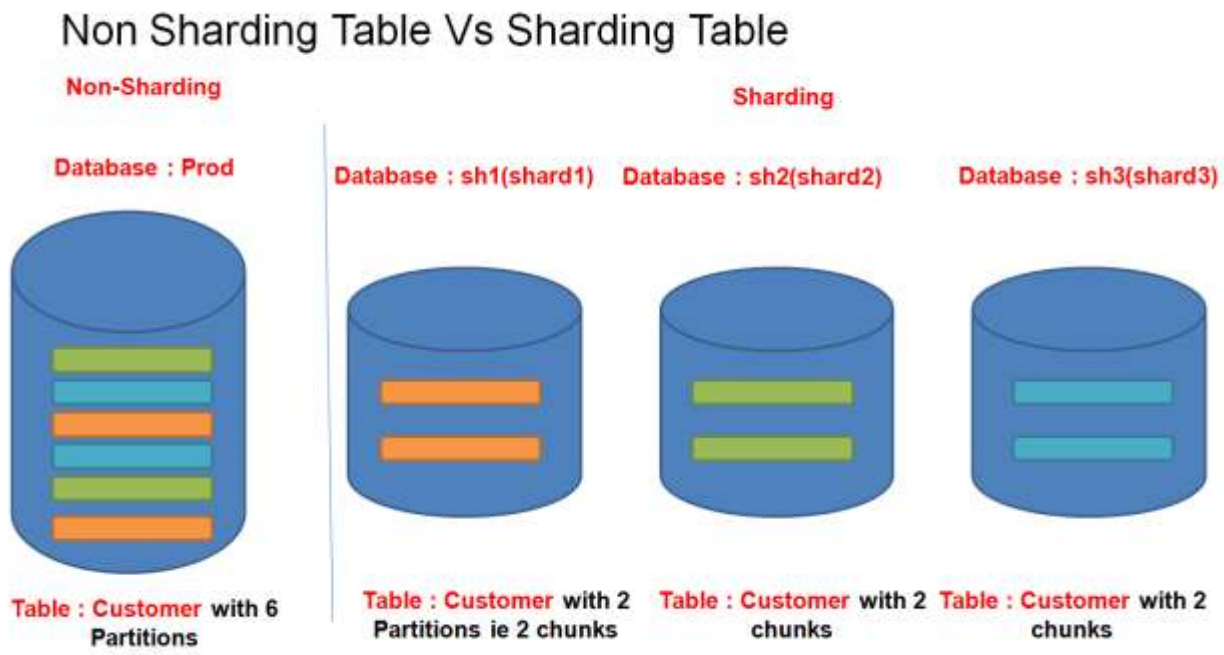
subsets of rows called CHUNKS. Sharding based on Oracle Partitioning. Partitions decompose a large table in to small partitions. Sharding keeps each partition into different databases called Shards. Number of partitions decided number of Shards.

Sharding uses Global Data Service (GDS) framework for automatic deployment and management. GDS uses GDSCTL utility. The Advantages of GDS is, it provides connection load balancing and role based routing in SDB (Shard Catalog). GSM Manager is a central component of GDS which provides direct routing of client connections to Shards. GSM is called as Shard Director. Shard catalog (SDB) stores the information about Shard table. Additionally it provides centralized schema maintenance and cross shard queries.

Overall Oracle Sharding DEPLOYMENT steps,

- **Create Shard Catalog**
- **Add GSM and Start GSM**
- **Create Shard(s)**
- **DEPLOY**

Overview



The diagram outlines the difference between non Sharding with Sharding environment. Database PRODS with table CUSTOMER having six partitions on the non Shard environment. Unlikely in Sharding environment three different database with dedicated hardware sharing partitions of single table CUSTOMER.

On Sharding environment shown in the diagram has the table CUSTOMER with 2 partitions referred as CHUNKS. Table load is shard across three different Shards. For application it is a single logical database (SDB). Application/Clients connect to GSM and it internally routes the connection to the respective Shard. Now Shard directly sends the requested data to clients directly.

TYPE

Sharding divided into three major categories.

- System Managed Sharding
- User Defined Sharding
- Composite Sharding

System Managed Sharding

System managed Sharding does not require the user to specify mapping of data to Shards. DATA is automatically distributed across the Shards using partitioning by consistent HASH. User has no control over data. The advantage of this method is, it will avoid HOT spots and provide uniform performance across Shards.

- Example

```
CREATE TABLESPACE SET TSP1;

CREATE SHARDED TABLE City
(
ccode VARCHAR2(60) NOT NULL,
cname VARCHAR2(60),
population VARCHAR2(60),
CONSTRAINT pk_ccode PRIMARY KEY (ccode)
) TABLESPACE SET TSP1
PARTITION BY CONSISTENT HASH (ccode) PARTITIONS AUTO;
```

User Defined Sharding

User has control over database means user specifies the mapping of data to individual Shards. It is useful in cases where application decides certain data need to be kept in a particular Shard and user have control on moving data between Shards. The advantage of this method is that in case of maintenance planned and unplanned outage of a Shard, the user knows exactly what data is not available. But the disadvantage of this model is user has to keep track of data to maintain the balanced distribution and workload across shards. User defined Sharding uses partitioning by Range or list. User defined Partitioning not available in 12.2.0.1 Beta.

Example:

```
CREATE TABLESPACE TSP1;
CREATE TABLESPACE TSP2;

CREATE SHARDED TABLE City
(
ccode VARCHAR2(60) NOT NULL,
...
gender VARCHAR2(10),
) Partition by list (gander)
(partition male_detail values ('MALE') tablesapce tbs1,
partition female_detail values ('FEMALE') tablesapce tbs2);
```

Composite Sharding

Composite Sharding is a combination of system managed and user defined Sharding. Data first partitioned by list or range and then further partitioned in to consistent hash. Consistent partition maintains balanced distribution of data across set of Shards.

- Combination of system managed and user defined Sharding
- Data is partitioned using LIST or RANGE.
- Subset of partitions further partitioned using composite Partitioning. Example,

```
CREATE TABLESPACE SET TSP1;

CREATE SHARDED TABLE City
(
ccode VARCHAR2(60) NOT NULL,
...
gender VARCHAR2(10),
) Partition by list (gander)
partition by consistent HASH (ccode)
Partition auto
(partition male_detail values ('MALE') tablesapce tbs1,
partition female_detail values ('FEMALE') tablesapce tbs2);
```

Why Sharding?

Sharding offers unique benefits such as

- Linear Scalability – Upon Adding/Removing Shards automatic chunks reorganization.
- Strong Fault Isolation - individual database carries unique set of data called Chunks.
- Horizontal Partitioning – Column Data (chunks) span across multiple databases (shards).

- Rolling Upgrade – While Upgrade on one shard application can still connect to others and easy to validate with individual chunks.

Oracle Sharding SYSTEM MANAGED SHARDING Creation steps:

Sharding require following environment,

- One Shard Director(minimum) – GMS
- One Shard Catalog Database (SDB)
- Shard Space
    - Two Primary Shard
    - Two Standby Shards

Prerequisites:

- Install GDS - refer KM Article 'Installation and GDS Configuration for Data guard and Golden Gate (Doc ID 2144138.1)'
- Install Oracle 12.2 Software in all Shard(s) including Shard catalog server
- Create Shard catalog Database (NON CDB)

IMPLEMENTATIN:

Once the environment is ready invoke GDSCTL to create Shard catalog (SDB)

- **Create Shard Catalog (SDB)**
- **Add GSM and Start GSM**
- **Create Shard(s)**
- **DEPLOY**
- **Create Shard Table**

Create Shard Catalog (SDB)

Before create Shard catalog do the following on Shard catalog server and all Shard servers.

- Disable FIREWALL
- Create necessary directories
- Create Shard Catalog Schema on SDB
- Configure Remote Scheduler
- Start scheduler on SDB and all Shard servers

Disable iptable on Catalog and all Shard Databases

Login as root,

```
service itables stop
chkconfig iptables off
```

Check for following directories,

```
chown root $ORACLE_HOME/bin/extjob
chmod 4750 $ORACLE_HOME/bin/extjob
chown root $ORACLE_HOME/rdbms/admin/externaljob.ora
chmod 640 $ORACLE_HOME/rdbms/admin/externaljob.ora
chown root $ORACLE_HOME/bin/jssu
chmod 4750 $ORACLE_HOME/bin/jssu
```

Create Base directory on all Shard DB

Login as Oracle OS user and create directory,

```
mkdir -p $ORACLE_BASE/oradata
mkdir -p $ORACLE_BASE/fast_recovery_area
ls -lst $ORACLE_BASE/oradata
ls -lst $ORACLE_BASE/fast_recovery_area
```

Create Shard Catalog Schema in SDB

Connect to Catalog database use SYS user, create Shard catalog user and assign necessary priviledge.

```
$ sqlplus / as sysdba
 alter system set db_create_file_dest='/u01/app/oracle/oradata' scope=both;
 alter system set open_links=16 scope=spfile;
 alter system set open_links_per_instance=16 scope=spfile;
 alter user gsmcatuser identified by gsmcatuser account unlock;
 create user mygds identified by mygds;
 grant connect, create session, gsmadmin_role to mygds;
 grant inherit privileges on user SYS to GSMADMIN_INTERNAL;
 alter system set events 'immediate trace name GWM_TRACE level 7';
 alter system set event='10798 trace name context forever, level 7'
scope=spfile;
 startup force
```

Configure the REMOTE scheduler on SDB server

On SDB database connect to SYS user and execute following command to configure remote scheduler.

```
set echo on
set termout on
spool config_remote_scheduler.lst
@?/rdbms/admin/prvtrsch.plb
SET FEEDBACK 1
SET NUMWIDTH 10
SET LINESIZE 80
SET TRIMSPOOL ON
SET TAB OFF
SET PAGESIZE 100
```

```
SET SERVEROUTPUT ON;
WHENEVER SQLERROR EXIT;
Begin
dbms_isched.agent_install_pre_steps;
end;
 /
alter session set "_ORACLE_SCRIPT" = false;
begin
 dbms_isched.agent_install_post_steps;
end;
/
```

Set the Scheduler Port and Password

On the same session set the AGNET port and password. NOTE: Same port will be used in all the Shard server agents to communicate with SDB server.

```
exec DBMS_XDB.sethttpport(8080);
commit;
exec DBMS_SCHEDULER.SET_AGENT_REGISTRATION_PASS('welcome');
alter system register;
```

Start scheduler on SDB server and all Shard servers,

Register SDB database on all the Shards. On all the Shard server connect to 12.2 ORACLE_HOME, start the agent and register SDB database.

```
schagent -start
schagent -status
echo welcome | schagent -registerdatabase blr 8080
```

ADD GSM and Start GSM

GSM acts as a Sharddirector for Sharding. It receives the client connection and route to respective Shard based on Shard_key. Once connection is established then Shard database sends the requested information back to clients directly.

Invoke GDSCTL and login using shard catalog user. First cerate SahrdCatalog(SDB) on the SDB server. In my example ‘blr’ is the SDB server hostname. Later on add GSM with any unused port. If any issue on add GSM and start GSM refer to GSM log which will be present in GDS_HOME.

```
$.oraenv
GSM
$gdsctl

gdsctl connect mygds/mygds@blr:1526:scat

GDSCTL>create shardcatalog -database blr:1526:scat -user mygds/mygds -chunks 12 -user
mygds/mygds -region region1, region2 -sdb scat

GDSCTL>add gsm -gsm shardDGdirector -listener 12106 -pwd gsmcatuser -catalog blr:1526:scat -
region region1 -trace_level 16

GDSCTL>start gsm -gsm shardDGdirector

GDSCTL>set  gsm -gsm shardDGdirector
GDSCTL>status gsm
Alias                     SHARDDGDIRECTOR
Version                   12.2.0.0.0
Start Date                07-MAY-2016 16:25:35
Trace Level               support
Listener Log File         /u02/app/oracle/diag/gsm/blr/sharddgdirector/alert/log.xml
Listener Trace File
/u02/app/oracle/diag/gsm/blr/sharddgdirector/trace/ora_23825_140392884396416.trc
Endpoint summary          (ADDRESS=(HOST=blr)(PORT=12106)(PROTOCOL=tcp))
GSMOCI Version            2.1.1
Mastership                Y
Connected to GDS catalog  Y
Process Id                23827
Number of reconnections   0
Pending tasks.     Total  0
Tasks in  process. Total  0
Regional Mastership       TRUE
Total messages published  0
Time Zone                 +05:30
Orphaned Buddy Regions:
     None
GDS region                region1
```

Create Shard(s)

Before adding Shardgroup check if GSM status shows correct information. If everything is perfect go ahead and add shardspaces one for primary and another one for standby (ADG). Now add Shard i.e primary shard server in primary shardspace and standby shard in standby shardspace respectively.

```
GDSCTL>modify catalog -agent_password welcome
GDSCTL>add shardgroup -shardgroup primary_shardgroup -deploy_as primary -region region1
GDSCTL>add shardgroup -shardgroup standby_shardgroup -deploy_as active_standby -region
region2
GDSCTL>add shardgroup -shardgroup standby_shardgroup -deploy_as active_standby -region
region2
GDSCTL>create shard -shardgroup primary_shardgroup -destination blr -osaccount oracle -
ospassword oracle
GDSCTL>add invitednode 10.10.10.30
GDSCTL>create shard -shardgroup standby_shardgroup -destination cbe -osaccount oracle -
ospassword oracle
GDSCTL>add invitednode 10.10.10.40
GDSCTL>create shard -shardgroup standby_shardgroup -destination hyd -osaccount oracle -
ospassword oracle
```

Deploy invokes netca and dbca to configure listener and create database (shards) on each servers added to the SDB. Before DEPLOY the GDSCTL>config shard Availability shows NULL. After deploy Availability shows the correct status. For example for primary it is ONLINE and for standby it is READ_ONLY. Chunks from 1 to 12 stored on primary and the same chunks replicated on standby for failover purpose.

```
GDSCTL>config shard
Name                Shard Group          Status    State       Region    Availability
----                -----------          ------    -----       ------    ------------
sh103               primary_shardgroup   U         none        region1   -
sh104               standby_shardgroup   U         none        region2   -

GDSCTL>deploy
GDSCTL>config shard
Name                Shard Group          Status    State       Region    Availability
----                -----------          ------    -----       ------    ------------
sh103               primary_shardgroup   Ok        Deployed    region1   ONLINE
sh104               standby_shardgroup   Ok        Deployed    region2   READ_ONLY

GDSCTL>config chunks
Chunks
------------------
Database                  From   To
---------                 -----  --
sh103                     1            12
sh104                     1            12
```

Create Service for Primary (read_write) and standby (read_only) database

Create role based services for the applications to connect read_write operations on primary and read_only operation on Standby. Additionally services with load balancing created for Sharding with DUPLCATED tables.

```
GDSCTL>add service -service prim_srv -role primary
GDSCTL>config service
GDSCTL>start service -service prim_srv
GDSCRL>add service -service stdby_srv -role physical_standby
GDSCTL>config service
GDSCTL>start service -service stdby_srv
```

```
GDSCTL>databases
Database: "sh103" Registered: Y State: Ok ONS: N. Role: PRIMARY Instances: 1 Region:
region1
   Service: "prim_srv" Globally started: Y Started: Y
            Scan: N Enabled: Y Preferred: Y
   Service: "stdby_srv" Globally started: Y Started: N
            Scan: N Enabled: Y Preferred: Y
   Registered instances:
     scat%1
```

```
Database: "sh104" Registered: N State: Errors ONS: N. Role: N/A Instances: 0 Region:
region2
Service: "prim_srv" Globally started: Y Started: N
            Scan: N Enabled: Y Preferred: Y
   Service: "stdby_srv" Globally started: Y Started: N
            Scan: N Enabled: Y Preferred: Y
```

Create Shard Tables/Shard schema

On Shard catalog Database (SDB) before creating Shard table enable shard ddl at the session level, create application schema, optionally create Tablespace for sharing table and create Shard table.

- Create Application Schema
- Create Tablespace for Shard
- Create Shard Table

Create Application Schema

On SDB create application schema and grant necessary privileges.

```
$ sqlplus / as sysdba
alter session enable shard ddl;
create user app_schema identified by app_schema;
grant all privileges to app_schema;
grant gsmadmin_role to app_schema;
grant select_catalog_role to app_schema;
grant connect, resource to app_schema;
grant dba to app_schema;
grant execute on dbms_crypto to app_schema;
```

Create Tablespace for Shard

```
CREATE TABLESPACE SET TSP_SET_2 using template (datafile size 100m extent management
local segment space management auto );
CREATE TABLESPACE products_tsp_1 datafile size 100m extent management local uniform
size 1m;
```

Create Shard table

Connect to application schema and create three dependent tables namely Customer, Orders and LineItems

```
SQL>connect app_schema/app_schema;

SQL>CREATE SHARDED TABLE cust
(
CustId VARCHAR2(60) NOT NULL,
FirstName VARCHAR2(60),
LastName VARCHAR2(60),
```

```
CONSTRAINT pk_cust PRIMARY KEY (CustId)
) TABLESPACE SET TSP1 PARTITION BY CONSISTENT HASH (CustId) PARTITIONS AUTO;
CREATE SHARDED TABLE Orders
(
OrderId INTEGER NOT NULL,
CustId VARCHAR2(60) NOT NULL,
constraint pk_orders primary key (CustId, OrderId),
constraint fk_orders_parent foreign key (CustId)
references cust on delete cascade
) partition by reference (fk_orders_parent);

SQL>CREATE SHARDED TABLE LineItems
(
OrderId INTEGER NOT NULL,
CustId VARCHAR2(60) NOT NULL,
ProductId INTEGER NOT NULL,
constraint pk_items primary key (CustId, OrderId, ProductId),
constraint fk_items_parent foreign key (CustId, OrderId)
references Orders on delete cascade
) partition by reference (fk_items_parent);

SQL>CREATE DUPLICATED TABLE Products
(
ProductId INTEGER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
Name VARCHAR2(128),
DescrUri VARCHAR2(128),
LastPrice NUMBER(19,4)
) TABLESPACE products_tsp_1;
```

Post Verification TEST

Run following query from SDB to verify the Tablespace name and size.

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

To verify the CHUNKS distribution

```
SQL>
set echo off
select a.name Shard, count( b.chunk_number) Number_of_Chunks from
gsmadmin_internal.database a, gsmadmin_internal.chunk_location b where
a.database_num=b.database_num group by a.name;
```

On each Shard databases verify the partition detail

```
SQL>set linesize 140
column table_name format a20
column tablespace_name format a20
column partition_name format a20
show parameter db_unique_name
select table_name, partition_name, tablespace_name from dba_tab_partitions where
tablespace_name like 'C%TSP_SET_2' order by tablespace_name;
```

Connect to application schema and check if Shard tables get created on each Shard

```
SQL>connect app_schema/app_schema_password
col TNAME for a20
select table_name from user_tables;

TNAME                TABTYPE  CLUSTERID
-------------------- ------- ----------
CUST                 TABLE
LINEITEMS            TABLE
MLOG$_PRODUCTS       TABLE
ORDERS               TABLE
PRODUCTS             TABLE
RUPD$_PRODUCTS       TABLE
```

On Primary Shard DGMGRL show configuration will give you the details of primary and replicated standby shards detail with FSFO enabled.

```
dgmgrl /
dgmgrl>connect sys

DGMGRL>show configuration;

Configuration - sh103

       Protection Mode: MaxPerformance
       Members:
       sh103 - Primary database
              sh104 - (*) Physical standby database

Fast-Start Failover: Enabled

Configuration Status;
Success      (status updated 51 seconds ago)
```

TROBUBLESHOOTING

Any issue in GSM check the GSM log which will be present in GSM_HOME, `$ORACLE_BASE/diag/gsm/<host_name>/GSM_NAME>/trace`.

Example,

Get the GSM name

```
GDSCTL>config
GDSCTL>status gsm -gsm <gsm_name>
GDSCTL>status gsm gsmDGDirector
Alias                     SHARDDGDIRECTOR
Version                   12.2.0.0.0
Start Date                07-MAY-2016 16:25:35
Trace Level               support
Listener Log File         /u02/app/oracle/diag/gsm/blr/sharddgdirector/alert/log.xml
```

```
Listener Trace File
/u02/app/oracle/diag/gsm/blr/sharddgdirector/trace/ora_23825_140392884396416.trc
Endpoint summary          (ADDRESS=(HOST=blr)(PORT=12106)(PROTOCOL=tcp))
GSMOCI Version            2.1.1
Mastership                Y
Connected to GDS catalog  Y
Process Id                23827
Number of reconnections   0
Pending tasks.     Total  0
Tasks in  process. Total  0
Regional Mastership       TRUE
Total messages published  0
Time Zone                 +05:30
Orphaned Buddy Regions:
     None
GDS region                region1
```

GSM log is under, '/u01/app/oracle/diag/gsm/blr/sharddgdirector/trace'.

If Deploy fails then check the logs on SDB $ORACLE_HOME/data/pendingjobs.dat as well as cfgtools for netca and dbca. A very important point is that do not open the wallet file which is present on $ORACLE_HOME/data. Opening wallet will cause damage to security keys and DBCA will fail with 'Invalid User authentication' on the server in which you opened the wallet. The details can be found on respective Shard 12.2, $ORACLE_HOME/data/pendingjobs.dat

CONCLUSION

The document offers basic to intermittent knowledge on what/why/how to implement Oracle database 12.2 New feature Sharding. The implementation example given in the article mainly concentrates on Sharing Replication rather than scaling. Details shown are with one primary and one Standby shard however it can be extended later on by adding more shards on primary and standby Shardspace for scalability.