ORACLE DATABASE 12c

Oracle Sharding

Linear Scalability, Fault Isolation and Geo-distribution for Web-scale OLTP Applications

ORACLE WHITE PAPER | DECEMBER 2018

ORACLE

Table of Contents

Introduction

Modern applications face the challenges of petabyte-scale data management, adherence to data sovereignty regulations, cloud enablement, uninterrupted data availability and support for millions or billions of customers. In addition these web scale applications have requirements to ingest relational transactions at a high velocity, elastically scale data, customers and transactions linearly and at the same time require a sophisticated RDBMS with support for ACID.

Oracle Sharding is a scalability, availability and geo-distribution feature for OLTP applications that distributes and replicates data across a pool of discrete Oracle databases. Each database in the elastic pool is referred to as a shard. Sharding is built on shared-nothing architecture where the databases do not share storage or cluster software. Figure 1 depicts how a massive non-sharded database is split into a pool of shards.

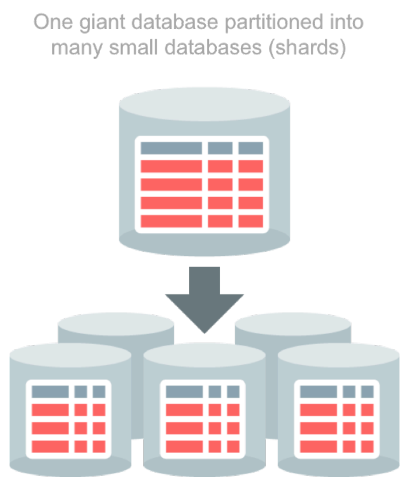
One giant database partitioned into many small databases (shards)

Figure 1: Divide and conquer methodology with Oracle Sharding

Each shard typically runs on commodity server with its own local storage, flash, and memory offering customers an opportunity to optimize performance at relatively low cost. Customers may also choose to employ an Oracle Engineered system as a shard, particularly for the geo-distribution use case. The first release of Oracle Sharding can scale up to 1,000 shards.

Oracle sharding distributes data across shards using horizontal partitioning. Horizontal partitioning splits a database table across shards so that each shard contains the table with the same columns but a different subset of rows. This partitioning is based on a sharding key. Examples of a sharding key include customer_id, account_no, email_id, country_id, etc. Each shard is replicated for performance, high availability and disaster recovery. The pool of shards is presented to an application as a single logical Oracle database (a sharded database or SDB). Applications that run on a sharded database architecture can achieve linear scalability, extreme data availability and geographic data distribution.

From the perspective of a database administrator, an SDB consists of multiple databases that can be managed either collectively or individually. However, from the perspective of an application developer, an SDB looks like a single logical database: the number of shards and the distribution of data across them are completely transparent to database applications. SQL statements issued by an application do not refer to shards nor are they dependent on the number of shards and their configuration. Oracle Sharding supports the entire lifecycle of a sharded database – deployment, sharded schema creation, data-dependent routing, elastic scaling, monitoring, patching etc.

This white paper on Oracle Sharding is intended for Enterprise Architects, Database Architects, Database Administrators, Application Architects and those who are involved in the design and architecture of distributed database systems.

Benefits of Oracle Sharding

Sharding with Oracle Database 12c Release 2 provides a number of benefits for web-scale applications.

- **Linear scalability**. OLTP applications designed for Oracle sharding can elastically scale (data, transactions and users) to any level, on any platform, simply by deploying new shards on additional stand-alone servers. Performance scales linearly as shards are added to the pool because each shard is completely independent from other shards.
- **Extreme Data Availability**. Oracle Sharding eliminates a single point of failure (shared disk, SAN, clustering, etc.) and provides strong fault isolation. The unavailability or slowdown of a shard due to either an unplanned outage or planned maintenance affects only the users of that shard, it does not affect the availability or performance of the application for users of other shards. Each shard may run a different release of the Oracle Database as long as the application is backward compatible with the oldest running version – making it simple to maintain availability of an application while performing database maintenance.
- **Data Sovereignty and Data Proximity via Geographic Data Distribution**. Sharding makes it possible to locate different parts of the data in different countries or regions – thus satisfying regulatory requirements where data has to be located in a certain jurisdiction. It also supports storing particular data closer to its consumers.

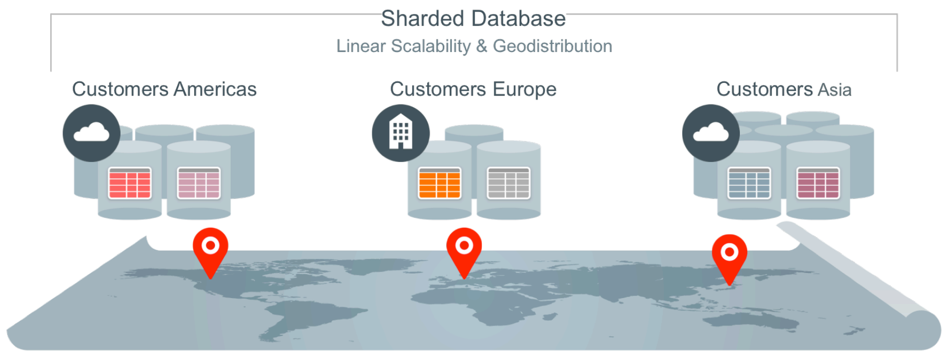

Figure 2: Oracle Sharding supports global data distribution

- **Cloud Deployment**. Since the size of a shard can be made arbitrary small, it is easy to deploy a sharded database in a cloud consisting of low-end servers. Oracle Sharding supports deployment on the cloud, on-premises and in the hybrid cloud model.

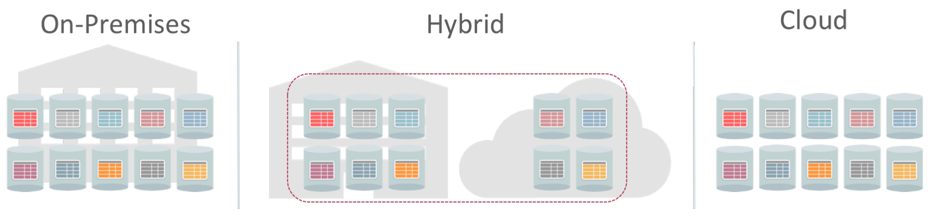

Figure 3: Oracle Sharding supports all three deployment models

- **Simplicity** via automation of many life-cycle management tasks including: automatic creation of shards and replication, system managed partitioning, single command deployment, and fine-grained rebalancing.
- **Superior run-time performance** with intelligent, data-dependent routing.
- **With the capabilities of an enterprise RDBMS**, including: relational schema, SQL, and other programmatic interfaces, complex data types, online schema changes, multi-core scalability, advanced security, compression, high-availability, ACID properties, multi-versioning concurrency control (MVCC), consistent reads, developer agility with JSON, and much more.

Oracle Sharding Components and Architecture

The following section describes the key components of the SDB infrastructure, namely, Shards, Shard Catalog, Shard Directors (GSM), Global Service and Connection Pools.

Figure 4 showcases the architectural components of Oracle Sharding:

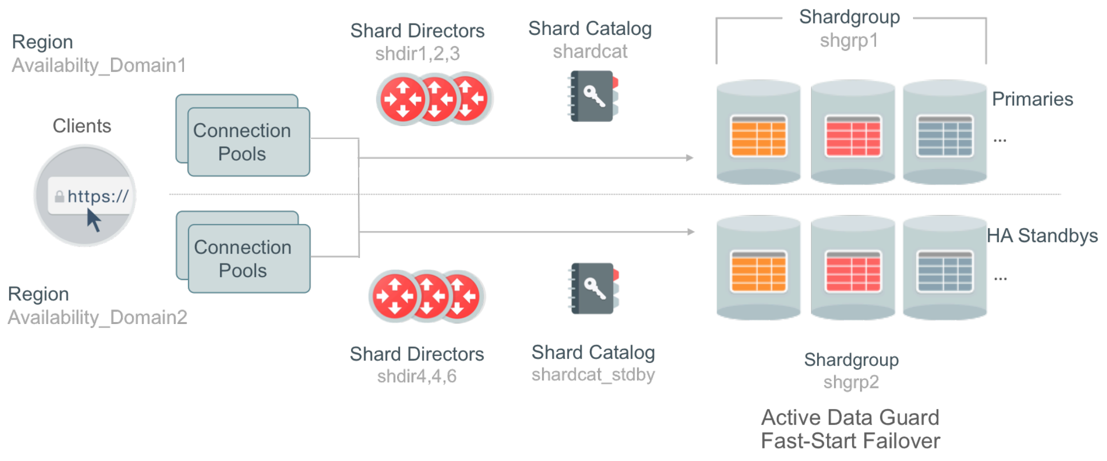

Figure 4: Oracle Sharding Architecture with no SPOF

Sharded Database and Shards

Shards are independent Oracle databases that are hosted on database servers which have their own local resources - CPU, memory, and disk . No shared-storage is required across the shards. A sharded database is a collection of shards. Shards can all be placed in one region (datacenter[s]) or can be placed in different regions. A region in the context of Oracle Sharding represents a datacenter or a multiple datacenters that are in close network proximity.

Shards are replicated for High Availability (HA) and Disaster Recovery (DR) with Oracle replication technologies such as Active Data Guard. For HA, the standby shards can be placed in the same region where the primary shards are placed. For DR, the standby shards are located in another region.

Shard Catalog

The shard catalog is a special-purpose Oracle Database that is a persistent store for SDB configuration data and plays a key role in automated deployment and centralized management of a sharded database. It also hosts the gold schema of the application and the master copies of common reference data (duplicated tables). The shard catalog database also acts as a query coordinator used to process multi-shard queries and queries that do not specify a sharding key.

All configuration changes, such as adding and removing shards and global services, are initiated on the shard catalog. All DDLs in an SDB are executed by connecting to the shard catalog.

An outage of the shard catalog does not affect the availability of the SDB. A shard catalog outage only affects the ability to perform maintenance operations or multi-shard queries during the brief period required to failover to a standby shard catalog. OLTP transactions are unaffected; they continue to be routed and executed by the SDBs. Oracle MAA recommends that a local Active Data Guard standby database be configured with Maximum Availability database protection mode with Fast-Start Failover and a remote physical standby database.

Shard Directors

The Shard Director is a regional network listener for clients that connect to an SDB. It maintains an up-to-date topology of the sharded database. Shard Directors route connections to the appropriate shards based on the sharding key passed during a connection request. The Shard Director is built upon the Oracle Global Service Manager (GSM). GSMs route connections based on database role, load, replication lag, and locality. For Oracle Sharding, GSMs have been enhanced to support management of SDB and routing of connections based on the location of data.

For a typical sharded database, shard directors (GSMs) are installed on dedicated low-end commodity servers in each region. Multiple shard directors should be deployed for high availability. In Oracle Database 12.2, up to 5 shard directors can be deployed in a given region. Oracle MAA recommends deploying three shard directors per region for availability and scalability.

Shard Directors provide the following set of functions

- Maintain runtime data about SDB configuration and availability of shards
- Measure network latency between its own and other regions
- Act as a regional listeners for clients to connect to an SDB
- Manage global services
- Perform connection and runtime load balancing information to connection pools
- Monitor availability of database instances and global services, and notify clients via FAN HA events upon failure incidents

Global Service

A Global Service is a database service that can run across multiple databases. A global service allows clients to access data on any shard in the SDB. Global Services offer additional properties for sharded databases - e.g., database role, replication lag tolerance, region affinity, etc. You can create role-based global services: create a read-write global service that accesses data from primary shards, and a separate read-only global service for Active Data Guard shards.

Connection Pools

At runtime, connection pools act as shard directors by routing database requests across pooled connections. Oracle supports connection pooling in various data access drivers such as OCI, JDBC, and ODP.NET etc.

Capabilities of Oracle Sharding

In Oracle Database 12.2.0.1, Oracle Sharding provides the capabilities that support the complete lifecycle of a Sharded Database which include the following:

Automated Data Distribution

In Oracle Database 12.2.0.1, Oracle Sharding supports two methods of sharding: system-managed and composite.

System-managed sharding:

System-managed sharding method automatically distributes data across shards using consistent hash (sharding_key). This algorithm evenly and randomly distributes data across shards to eliminate hot spots and provide uniform performance across shards. Oracle Sharding automatically rebalances data when shards are added to or removed. System-managed sharding is the most used form of sharding.

Oracle Sharding is built upon Oracle Partitioning. In essence, Sharding is distributed partitioning where tablespaces that host the partitions are distributed among the shards. It uses the familiar SQL syntax for table partitioning to specify how table rows are partitioned across shards. A partitioning key for a sharded table is also the sharding key. For example, the `CREATE SHARDED TABLE` statement is used to create a sharded table based on `cust_id` as the sharding key.

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

Oracle Sharding automates the creation of tablespaces on all the shards as a unit called a tablespace set. The `PARTITIONS AUTO` clause specifies that the number of partitions should be automatically determined and mapped to the tablespaces.

The unit of data migration between shards is a chunk. A chunk is a set of tablespaces that collocate corresponding partitions of all tables in a table family. A chunk contains a single partition from each table of a set of related tables. This guarantees that related data from different sharded tables can be moved together. The number of chunks within each shard is specified when the SDB is created (default is 120 chunks per shard).

In addition to sharded tables, many applications require common reference data (e.g. Products) that needs to be accessed as part of the queries in the transactions. Multi-shard queries can be avoided by replicating the small number of read-only or read-mostly "non-shardable" tables across all shards. Replication of complete tables is a good choice for relatively small tables that are often accessed together with sharded tables. A table with the same contents in each shard is called a Duplicated Table. For example a Customers–Orders–Line Items schema may also include a Products table. This table contains data shared by all customers and cannot be sharded by the customer number. Instead, the entire table is duplicated on all databases to prevent multi-shard queries during order processing. In the example of the Products table, it is created using the `CREATE DUPLICATED TABLE` statement.

Oracle Sharding uses materialized view replication to synchronize contents of duplicated tables. A duplicated table on each shard is represented by a read-only materialized view. The master table for the materialized views is located in the Shard Catalog. The materialized views on all shards are automatically refreshed with configurable frequency. `CREATE DUPLICATED TABLE` automatically creates the master table, materialized views and other objects required for materialized view replication.

The combination of sharded and duplicated tables enables all transactions associated with a sharding key to be processed by a single shard. This technique enables linear scalability and fault isolation.

Composite Sharding:

With composite sharding method, data is first partitioned by list or range (super_sharding_key) and then further partitioned by consistent hash (sharding_key). The two levels of sharding make it possible to map data to a set of shards, and then automatically maintain balanced distribution of data across that set of shards. Composite sharding is ideal for global data distribution where shards are placed in each geography and within a given geography data is uniformly distributed and enables linear scalability.

The composite sharding method allows you to create multiple shardspaces for different subsets of data in a table partitioned by consistent hash. A shardspace is set of shards that store data that corresponds to a range or list of key values. For example, the following `CREATE SHARDED`

TABLE statement is used to create a table with composite sharding method based on geo as the super_sharding_key and cust_id as the sharding key.

```
CREATE SHARDED TABLE customers
( cust_id NUMBER NOT NULL
, name VARCHAR2(50)
, address VARCHAR2(250)
, geo VARCHAR2(20)
, class VARCHAR2(3)
, signup_date DATE
, CONSTRAINT cust_pk PRIMARY KEY(geo, cust_id)
)
PARTITIONSET BY LIST (geo)
  PARTITION BY CONSISTENT HASH (cust_id)
  PARTITIONS AUTO
(PARTITIONSET AMERICA VALUES (‘AMERICA’) TABLESPACE SET tbs1,
 PARTITIONSET ASIA VALUES (‘ASIA’) TABLESPACE SET tbs2);
```

Oracle Sharding also supports subpartitioning methods provided by Oracle Database and enables information lifecycle management (ILM) by placing subpartitions on separate tablespaces and moving them between storage tiers. Migration of subpartitions between storage tiers can be done without sacrificing the scalability and availability benefits of sharding and the ability to perform partition pruning and partition-wise joins on a primary key. This composite Sharding with sub-partitioning provides three-levels of data organization.

Centralized schema management

The SDB schema, including the structure of sharded and duplicated tables and the data distribution, is maintained in the Shard Catalog. The Shard Directors in conjunction with the Shard Catalog propagate the schema to all the shards.

Automated creation and replication of shards

In the Oracle Database 12.2.0.1 release, Oracle Sharding supports three automatically configured replication options: Data Guard, Active Data Guard, or Oracle GoldenGate (Oracle GoldenGate 12.3 is planned to support Oracle Sharding).

Sharding supports two SDB deployment methods.

1. The first method is with the “CREATE SHARD” GDSCTL command. With this method, the shards and their respective listeners are automatically created.

   Once the primary shards are created, the corresponding standby shards are automatically built using the RMAN ‘duplicate’ command. After which the Data Guard Broker configuration with Fast-Start Failover (FSFO) is automatically enabled. The FSFO observers are automatically started on the regional shard director.

In system-managed and composite sharding, the logical unit of replication is a group of shards called a shardgroup. In system-managed sharding, a shardgroup contains all of the data stored in the SDB. The data is sharded by consistent hash across shards that make up the shardgroup. Shards that belong to a shardgroup are usually located in the same data center. An entire shardgroup can be fully replicated to one or more shardgroups in the same or different data centers.

2. The second method is with the "ADD SHARD" GDSCTL command. Many organizations have their own database creation standards and they may opt to deploy the SDB using their own pre-created databases (shards). The ADD SHARD based deployment method supports this requirement by simply adding the shards, which are pre-built by the user.

These two deployment methods support both the initial and incremental deployments.

Data-dependent routing

Oracle Sharding supports Direct and Proxy routing of database requests to shards.

Direct Routing:

Key enhancements have been made to Oracle connection pools and drivers to support Sharding. Starting from 12.2, JDBC/UCP, OCI and Oracle Data Provider for .NET (ODP.NET) recognize the sharding keys as part of the connection check out. Apache Tomcat, JBoss, IBM WebSphere and Oracle WebLogic can use UCP support for sharding. PHP, Python, Perl, and Node.js can use OCI support.

Sharding Key is used for routing the database connection requests at a user session level during connection checkout. Based on this information, connection is established to the relevant shard which contains the data pertinent to the given sharding_key. Once the session is made to a shard, all SQL queries and DMLs are supported, executed in the scope of the given shard and require no modification.

Upon the first connection to a given shard, the sharding key range mapping is collected from the shards to dynamically build the shard topology cache. This routing map is cached in the client. This allows subsequent requests for sharding keys within the cached range to be routed directly to the shard, bypassing the shard director. Such data-dependent routing of database requests eliminates an extra network-hop - thereby decreasing the latency for high volume OLTP applications.

When a connection request is made with a sharding key, connection pool looks up the corresponding shards on which this particular sharding key exists (from its topology cache). If a matching connection is available in the pool, then the pool returns a connection to one of these shards by applying its internal connection selection algorithm. If a connection is not available, then forwarding the request with the sharding key (by the connection pool) to the shard director creates a new connection.

As illustrated in Figure 5, DB connection request for a given sharding key that is in any of the cached topology map, goes directly to the shard (i.e., bypassing the shard director).

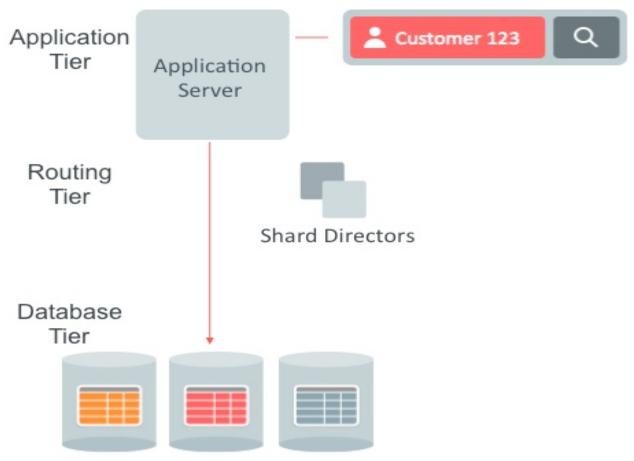

Figure 5. Logical flow of direct routing – Connection pool as a Shard Director

Note: Super_sharding_key is needed only in the case of composite sharding.

The routing map automatically refreshes when a shard becomes unavailable or changes occur to the sharding topology. This is enabled by the Fast Application Notification (FAN) published by Shard Director via Oracle Notification Server (ONS).

Proxy Routing for Multi-Shard Queries:

Proxy routing is an ancillary usage pattern targeted for developer convenience. This requires connection be established to the coordinator. In Oracle Database 12.2.0.1, the shard catalog database assumes the role of the coordinator database. Once the session is made to the coordinator, SQL queries and DMLs are executed and require no modification. Proxy routing is suitable for the following scenarios:

» When the application cannot pass the sharding key during connect.

» When the application needs to access data from multiple shards in the same query. For example, a query to aggregate data across all shards.

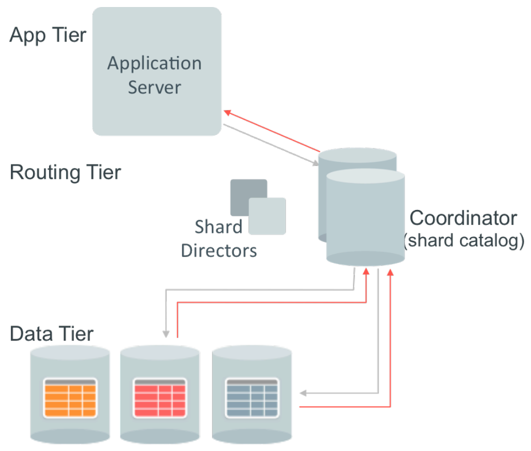

Figure 6. Logical flow of proxy routing

As illustrated in Figure 6, routing via the coordinator allows users to submit SQL statements without a sharding key value passed during connect. The Coordinator's SQL compiler analyzes and rewrites the query into query fragments that are sent and executed by the participating shards. The queries are rewritten so that most of the query processing is done on the participating shards and then aggregated by the coordinator.

Applications should separate their workloads for direct vs. proxy routing. Separate connection pools must be created for these workloads.

Full Support for the Lifecycle Management of an SDB

Oracle Sharding platform allows shards to be added or removed online. It also supports the data reorganization at the chunk level.

Online scale-out with auto-resharding

Oracle Sharding provides the capability to elastically scale-out the sharded database via incremental deployment - shards are added online to expand the pool. In System Managed sharding, the addition of new shards to the pool will automatically trigger resharding wherein the chunks are automatically moved in order to attain balanced distribution of data.

Ability to scale-back

Sharding allows scaling back the sharded database by removing the shards. In order to shrink the pool, you can use the "REMOVE SHARD" command. In 12.2.0.1, the chunks must be explicitly moved to the other shards before removing the shard.

Chunk Move or Split

In addition to shard addition and removal, Oracle Sharding supports splitting and moving of chunk from one shard to another. Chunks can also be moved from one shard to another when data or workload skew occurs without a change in the number of shards. Chunk migration can be initiated automatically or by the DBA.

Structural modifications to the Sharded Database are done transparently to the application. Connection pools get notified (via ONS) about split, move, add/remove shards and auto-resharding. Application can choose either to reconnect or access read-only.

Patching Automation

Sharding enables patching all the shards with one command via *opatchauto*. OPatchauto supports all sharding schemes and replication methods. It also supports rolling mode and parallel mode.

Supports Command-Line and Graphical User Interfaces

Oracle Sharded databases can be deployed, managed and monitored with two interfaces:

<u>GDSCTL:</u>

GDSCTL is a command-line interface that provides a simple declarative way of specifying the configuration of an SDB and automating its deployment. For example, just a few GDSCTL commands are required to create the SDB:

CREATE SHARDCATALOG

ADD GSM; START GSM (create and start shard directors)

CREATE SHARD (for each shard)

DEPLOY (for automated creation of shards and replication setup)

<u>Oracle Enterprise Manager:</u>

Enterprise Manager enables management of the life cycle of a sharded database with graphical user interface. For example, with few clicks, a shard director and shard software can be provisioned. Likewise, the deployment of shard directors and the creation of sharded database can be performed. With EM, you can also manage and monitor an SDB for availability and performance.

Application Requirements

Oracle Sharding is intended for OLTP applications that are designed for a sharded database architecture. Sharded applications should meet the following requirements:

» Have a well-defined data model and data distribution strategy (consistent hash or composite) that primarily accesses data via some key. Examples of keys include customer ID, account number, country_id, etc.

» Transactions that are frequently executed should operate on a single shard. These transactions provide a sharding key for high performance routing and typically access 10s or 100s of rows. For example, lookup and update of a customer's billing record; lookup and update of a subscriber's documents etc.

» The data must be modeled as a hierarchical table family that consists of a root table and many child and grandchild tables. In 12.2.0.1, one table family is supported per sharded database.

» Every table in the table family must contain the sharding _key.

» Sharding _key must be the leading column of the primary key of the root table.

- Referential integrity makes it simpler to create a sharded database but is not required. Oracle Sharding allows the creation of a sharded table family in the absence of referential integrity constraints.
- Common reference tables that are not shard-able must be identified and created as *Duplicated Tables*.
- Use Oracle integrated connection pools (UCP, OCI, ODP.NET, JDBC).
- Use separate connection pools for workloads that use direct routing and those that use proxy routing.
- Use separate global services for read/write and read-only workloads.
- For each key-based request, the application should check out a new connection from the pool by specifying the sharding key using the API provided with Oracle Sharding.

Oracle Sharding Benchmark on Oracle Bare Metal Cloud

This section covers the results of the Oracle Sharding MAA benchmark on Oracle Bare Metal Cloud.

The objectives of this benchmark are to:

» Elastically scale-out the Sharded Database (SDB) on Oracle Bare Metal IaaS Cloud – following the Oracle Maximum Availability Architecture (MAA) best practices

» Demonstrate the linear scalability of relational transactions with Oracle Sharding

» Observe the fault isolation aspects of a sharded database

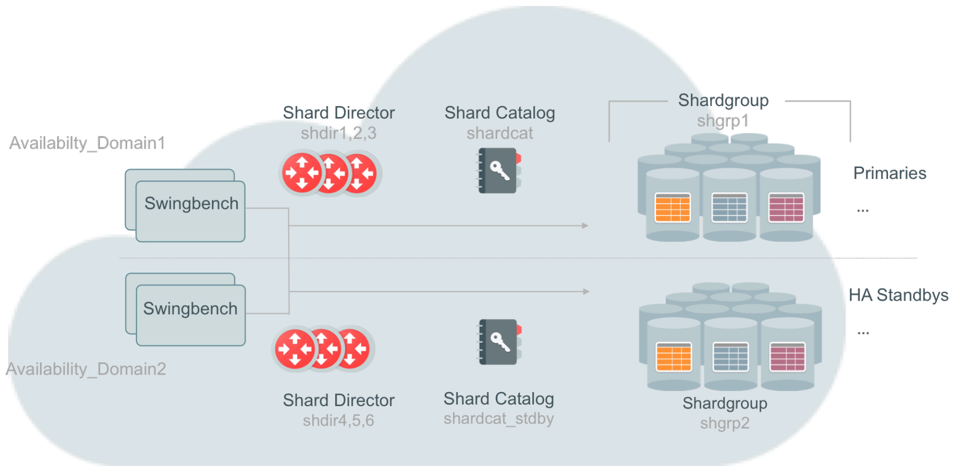

Figure 7. Sharded Database Topology on Oracle Bare Metal Cloud

As shown in Figure 7, the sharded database used in this benchmark has two shard groups - one in each of the availability domains of the Oracle Bare Metal Cloud. The network latency between the availability domains is less than 1ms. Shardgroup1 in Availability_Domain1 has 100 primary shards and the shardgroup2 in Availability_Domain2 has 100 HA standby shards. Each shard is hosted on dedicated bare metal server, which has 36 cores, 512G RAM and four 12.8TB NVMe flash disks. These flash disks are used for the creation of two ASM Diskgroups (+DATA, +RECO) with normal redundancy. Three shard directors were deployed in each of the Availability Domains for high availability. The shard catalog is placed in Availability_Domain1 and its standby is located in Availability_Domain2.

This benchmark uses the Swingbench Order-entry application. Customer_id column is defined as the sharding key. The data model has been modified so that every sharded table in the table family contains the sharding key. Tables used for reference data have been defined as duplicated tables. The connection pooling code has been modified to use the Oracle Sharding API calls - to check out a connection on a given shard based the sharding key.

Role-based global services have been created so that read-write transactions run on primary shards and queries run on standby shards. The total size of the sharded database is 50TB with 500G on each of the 200 shards.

This sharded database uses Active Data Guard in Max Availability with Fast-Start Failover for replication. The SDB is deployed automatically using the "CREATE SHARD" and Oracle Sharding automatically configured the replication. The FSFO observers are automatically started on the regional shard director.

Here are the steps taken for the execution of the benchmark:

1) Begin application workload on 25 primary shards and 25 standby shards
2) Bring up 25 more primary and 25 more standby shards
3) Repeat step #2 until all the 200 shards are online on both Availability Domains
4) Observe linear scaling of transactions per second (TPS) and queries per second (QPS)
5) Compute the total read-write transaction per second and queries per second across all shards in the sharded database

Following is the table that shows the TPS and QPS observed, as the SDB is elastically scaled-out.

TABLE 1 LINEAR SCALABILITY OF TRANSACTIONS AS SHARDS ARE ADDED

Primary shards	Standby shards	Read/Write Transactions per Second	Ready Only Queries per Second
25	25	1,180,000	1,625,000
50	50	2,110,000	3,260,000
75	75	3,570,000	5,050,000
100	100	4,380,000	6,820,000

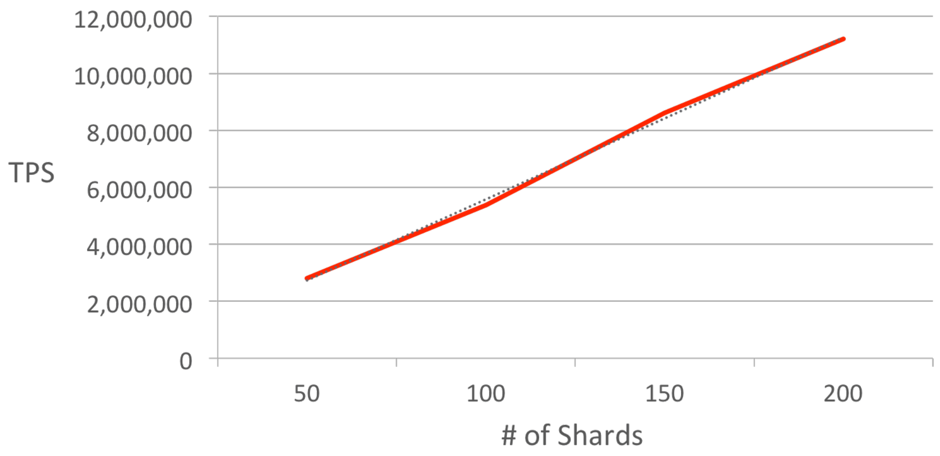

Figure 8. Transactions and queries scaled linearly as shards are added

Figure 8 illustrates that as shards were doubled, tripled and quadrupled, we were able to observe that the rate of transactions and queries doubled, tripled and quadrupled accordingly. This demonstrated the frictionless linear scaling due to shared-nothing hardware architecture of Oracle Sharding. Altogether we were able to execute 11.2 Million transactions per second that includes 4.38 Million read-write transactions per second across all the 100 primary shards and 6.82 Million read-only transactions per second across all the 100 active standby shards.

The study also illustrated that Oracle Sharding provided extreme data availability. When a failure was induced on a given shard, there was absolutely no impact to the other shards in the SDB. This is due to zero shared hardware or software among the shards.

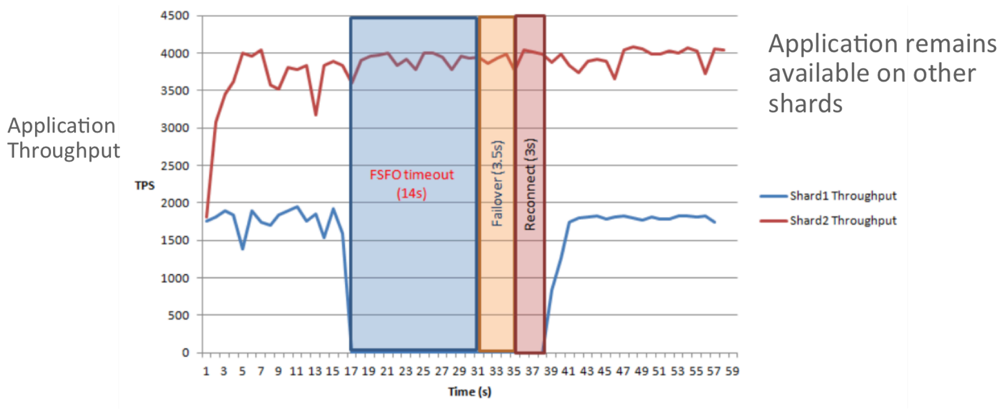

Figure 9. A shard outage has zero impact on surviving shards

As can be seen, Oracle Sharding provides both linear scalability and fault isolation.

Building Document Store Applications with Oracle Sharding and Oracle JSON

Starting with Oracle Database 12c Release 12.1.0.2, Oracle Database provides the same application-development experience as a special-purpose NoSQL document store. It can store, manage, and index JSON documents, and it provides a NoSQL-like document-store API that enables rapid schemaless development. Oracle delivers the schemaless development paradigm that many developers want today, but without compromising on core enterprise capabilities. While many NoSQL systems are now recognizing the need for a tabular, structured format for accessing data, and some are even introducing SQL-like languages, Oracle Database delivers the full power of SQL to JSON document stores today, with its advanced SQL analytic capabilities and scalable parallel SQL infrastructure.

Unlike NoSQL databases, Oracle Database provides sophisticated SQL querying and reporting over JSON documents. This lets you integrate JSON and relational data, joining them in the same query. And because JSON features are integrated into Oracle Database 12c, all of its enterprise features for availability, security, scalability, and manageability are fully supported for JSON data. And now, with the advent of Oracle Sharding which naturally supports Oracle JSON, document-oriented databases can be built very easily. The following table illustrates how simple it is to build a system managed sharded table for a document store application.

```
CREATE SHARDED TABLE Customers
( CustId     VARCHAR2(60) NOT NULL,
  CustProfile CLOB,
  CONSTRAINT pk_customers PRIMARY KEY(CustId),
  CONSTRAINT cons_json CHECK (CustProfile IS  JSON)
)
```

```
PARTITION BY CONSISTENT HASH (CustId)
PARTITIONS AUTO
TABLESPACE SET tbs1 ;
```

The combination of Oracle JSON and Oracle Sharding provides data model flexibility, developer productivity plus the linear scalability of a sharded database.

How Oracle Sharding compares with Cassandra and Mongo DB?

Oracle Sharding, in contrast to NoSQL data stores, provides all the capabilities of an enterprise RDBMS. These enterprise capabilities include: SQL and other programmatic interfaces, support for complex data types, online schema changes, multi-core scalability, advanced security, compression, extensive high-availability features, ACID properties, consistent reads, the developer agility of JSON, and much more. For example, Oracle's inherent support for transactions means concurrent updates/reads never get inconsistent or incorrect results; Oracle also extends read consistency to multi-shard operations as well using global consistent read. Following are the tables that compare and contrast Oracle Sharding with Cassandra and Mongo DB NoSQL data stores.

	Cassandra	Oracle Sharding
Linear scalability and fault isolation	✓	✓
Deploy on commodity hardware on premises or on Cloud	✓	✓
Replication	✓	✓
Relational/SQL – joins, transactions, no pre-defined queries, multi-shard queries	○	✓
ACID transactions, concurrency control & strong consistency	○	✓
Enterprise grade partitioning, compression, security, backup/recovery, DR	○	✓
Supports Multi-core over SMP (Shard-level)	○	✓
Leverage in-house and world-wide Oracle DBA skillset	○	✓

Figure 10: Oracle Sharding compared with Cassandra

	Mongo DB	Oracle Sharding
Linear scalability and fault isolation	✓	✓
Document-store capability (store, index, and query JSON documents)	✓	✓
Deploy on commodity hardware on premises or on Cloud	✓	✓
Multi-master Replication	○	✓
Relational/SQL & Multi-shard queries	○	✓
Joins within documents, within collections and across Collections	Within Documents Only	✓
Joins with Relational, XML, Spatial and Text content	Limited Support for Text	✓
ACID transactions, concurrency control & strong consistency	○	✓
Enterprise grade partitioning, compression, security, backup/recovery, DR	○	✓
Leverage in-house and world-wide Oracle DBA skillset	○	✓

Figure 11: Oracle Sharding compared with Mongo DB

In summary, Oracle Sharding combines Oracle Enterprise Edition with a comprehensive solution for deploying a sharded architecture that includes automation to simplify many aspects of life-cycle management, advanced partitioning methods for increased flexibility, and intelligent data-dependent routing for superior run-time performance. The combination of Oracle Sharding and the Oracle RDBMS provide customers with the best of both worlds: the ability to massively scale using a sharded database architecture without the compromises of a NoSQL data store.

How Oracle Sharding compares with Oracle RAC?

The combination of Oracle RAC and Active Data Guard provide transparent scale-out and availability for OLTP and analytic applications. With Oracle RAC, all transactions can act on any data within the database, there is no need to partition data or concern for the performance of multi shard operations; all RAC instances share direct access to the same physical database. Oracle Sharding trades-off transparency in return for much higher level of scalability, extreme availability and global data distribution for custom OLTP applications.

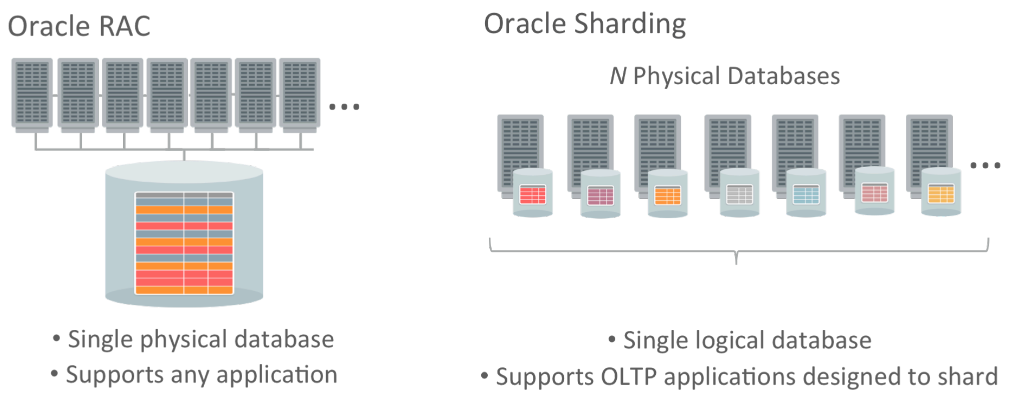
Oracle RAC
• Single physical database
• Supports any application
Oracle Sharding
N Physical Databases
• Single logical database
• Supports OLTP applications designed to shard

Figure 12: Oracle Database 12cR2 offers two choices for scalability and availability

Licensing

Oracle Sharding is available for on-premises, on Oracle Cloud and for the hybrid cloud model. This section discusses the licensing details for various scenarios.

<u>On-Premises and BYOL model for Oracle Cloud (IaaS):</u>

For a sharded database (SDB) with 3 or fewer primary shards, Oracle Sharding is included with EE (includes Data Guard). There is no limit on the number of standby shards. For the three primary shards, if Active Data Guard, Oracle GoldenGate or Oracle RAC are used, they must be licensed accordingly.

For an SDB with more than 3 primary shards, the licensing of Oracle Sharding requires that all shards be licensed for Oracle Enterprise Edition (EE) and one of the Oracle High Availability options (i.e. Active Data Guard or Oracle GoldenGate or Oracle RAC). If an organization has Oracle Unlimited License Agreement (ULA) that covers EE and one of the HA options – Active Data Guard or Oracle GoldenGate or Oracle RAC then Oracle Sharding is included at no additional cost.

<u>Oracle Cloud (using PaaS instances):</u>

With DBCS EE and DBCS EE-High Performance (HP), use is limited to three primary shards. There is no limit on the number of standby shards. With DBCS EE-Extreme Performance (EP) and Exadata Cloud Service (ECS), there is no limit on the number of primary shards or standby shards.

Oracle Database Cloud Services (PaaS) do not currently provide turnkey automation for deploying a sharded database. Automated deployment of Oracle Sharding using Oracle Database Cloud Services is on the roadmap. Till then, Oracle Sharded databases can be deployed on Oracle Cloud DBCS instances using the cookbooks (Listed in www.oracle.com/goto/oraclesharding).

Conclusion

Oracle Sharding is a scalability, availability and geo-distribution feature for OLTP applications that enables distribution and replication of data across a pool of Oracle databases that share no hardware or software. Applications elastically scale (data, transactions and concurrent users) to any level, on any platform, simply by adding additional databases (shards) to the pool. Oracle Sharding allows applications to perform high velocity relational transactions.

Oracle Sharding automates the entire lifecycle of a sharded database – deployment, schema creation, data-dependent routing with superior run-time performance, elastic scaling and simpler life-cycle management. It also provides the advantages of an enterprise DBMS, including: relational schema, SQL, and other programmatic interfaces, support for complex data types, online schema changes, multi-core scalability, advanced security, compression, high-availability, ACID properties, consistent reads, developer agility with JSON, and much more.

References

1. Oracle Sharding OTN Page – http://www.oracle.com/goto/oraclesharding
2. Oracle Sharding Documentation - http://www.oracle.com/technetwork/database/database-technologies/sharding/documentation/index.html