Oracle Sharding

Geo-distributed, Linearly Scalable, Multi-model, Cloud-native Database

Nagesh Battula, Sr. Principal Product Manager, Oracle
Mark Dilman, Sr. Director, Product Development, Oracle
Gairik Chakraborty, Sr. Director, Database, Epsilon

Oct 23rd, 2018

@NageshBattula
Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described for Oracle’s products may change and remains at the sole discretion of Oracle Corporation.
Program Agenda

1. Oracle Sharding Overview
2. Customer Case Study – Epsilon
3. Sharding 19c Features
4. Sharding Use Cases
Oracle Sharding | Concepts

• Horizontal partitioning of data across up to 1000 independent Oracle Databases (shards)

• Shared-nothing hardware architecture
  – Each shard runs on a separate server
  – No shared storage
  – No clusterware

• Data is partitioned using a sharding key (i.e. account_id)
Oracle Sharding | Benefits

Linear Scalability

Add shards online to scale database size and throughput. Online split and rebalance.

Ultra Availability

Shared-nothing hardware architecture. Fault of one shard has no impact on others.

Geographic Distribution

User defined data placement for performance, availability, DR or to meet regulatory requirements.
Oracle Sharding | Architecture & Key Features

- Automated deployment of up to 1000 shards with replication (Active Data Guard and Oracle GoldenGate)
- Sharding methods
  - System-managed, User-defined and Composite
- Centralized schema maintenance
  - Native SQL for sharded and duplicated tables
  - Relational, JSON, LOBs and Spatial support
- Direct routing and Proxy routing
- Online scale-out w/ auto-resharding or scale-in
- Mid-tier Sharding
  - Scale mid-tiers along with shards
- RAC Sharding
  - Gives a RAC DB, the performance and scalability of Oracle Sharding with minimal application changes
Oracle Sharding | Sharded and Duplicated Tables

Database Tables

Customers
- Customer: 123 Mary
- Customer: 456 John
- Customer: 999 Peter

Orders
- Order 4001, Customer 123
- Order 4002, Customer 456
- Order 4003, Customer 999
- Order 4004, Customer 456
- Order 4005, Customer 456

Line Items
- Customer: 123, Order: 4001, Line: 40011
- Customer: 999, Order: 4003, Line: 40012
- Customer: 123, Order: 4001, Line: 40013
- Customer: 456, Order: 4004, Line: 40014
- Customer: 999, Order: 4003, Line: 40015
- Customer: 999, Order: 4003, Line: 40016

Products
- SKU: 100, Product: Coil
- SKU: 101, Product: Piston
- SKU: 102, Product: Belt

Sharded Tables
- Shard 1
- Shard 2
- Shard N

Duplicated Table
Oracle Sharding | Data Modeling Considerations

- To reap sharding benefits, schema must be designed to maximize number of single-shard requests

- Sharding-amenable schema consists of:
  - **Sharded Table Family:**
    - Consists of a single root table and hierarchy of child and grandchild tables
    - Set of tables **equi-partitioned by the sharding key**
      - Related data is always stored and moved together
      - Joins & integrity constraint checks are done within a shard
    - **Sharding method and key** are based on App requirements
    - Sharding key must be the **leading column of a primary key**
  - **Duplicated Tables:**
    - Non-sharded tables are replicated to all shards
    - Usually contain common reference data
    - Can be read or updated on each shard
Sharding Method | **System-managed**

**Linear Scalability & Ultra Availability**

- Partition data across shards by CONSISTENT HASH
  - Ranges of hash values of sharding keys are assigned to shards
- Data is sharded / re-sharded automatically
- Data is evenly distributed across shards
- All shards are managed and replicated as unit
- Many relatively small equally sized chunks (120 per shard)

+ Automatic balanced data distribution
- User has no control on location of data

---

Customer profile applications
sharded by customer_profile_id

Sharded Database

Linear Scalability & Ultra Availability

...
System-managed Sharded Table Family | Example

CREATE TABLESPACE SET tbs1;
CREATE TABLESPACE products_tsp;

CREATE SHARDED TABLE Customers
( CustId VARCHAR2(60) NOT NULL,
  FirstName VARCHAR2(60),
  LastName VARCHAR2(60),
  ...
  CONSTRAINT pk_customers
    PRIMARY KEY(CustId)
)
PARTITION BY CONSISTENT HASH (CustId)
PARTITIONS AUTO
TABLESPACE SET tbs1;

CREATE SHARDED TABLE Orders
( CustId VARCHAR2(60),
  OrderId INTEGER,
  OrderDate TIMESTAMP,
  ...
  CONSTRAINT pk_orders
    PRIMARY KEY(CustId, OrderId),
  CONSTRAINT fk_orders_parent
    FOREIGN KEY(CustId) REFERENCES Customers(CustId)
)
PARTITION BY REFERENCE (fk_orders_parent) ;

CREATE DUPLICATED TABLE Products
( ProductId INTEGER PRIMARY KEY,
  Name VARCHAR2(128),
  LastPrice NUMBER(19,4),
  ...
)
TABLESPACE products_tsp ;
Sharding Method | User-defined

Explicit mapping of data to shards for better control, compliance & performance

- Partition data across shards by RANGE or LIST
  - Ranges or lists of sharding key values are assigned to shards by the user

- User-controlled data distribution provides:
  + Regulatory compliance
    • Data remains in country of origin
  + Hybrid cloud and cloud bursting
    • Some shards on premises; other shards in the cloud
  + Control of data availability for planned maintenance
  + Ability to customize hardware resources and HA configuration for subsets of data
  + More efficient range queries
  - User needs to maintain balanced data distribution

- Fast, direct access from within each region
- Multi-shard queries access data from all regions

Sharded Database

Geodistribution for Data Sovereignty & Privacy Compliance

Customers Americas

Customers Europe

Customers Asia
Sharding Method | Composite

Geographic Distribution, Linear Scalability and Ultra Availability

• Provides two-levels of sharding

• The sharded database is divided into N sets of shards called shardspaces
  – Data is partitioned across shardspaces by LIST or RANGE on super-sharding key (e.g. geography)
  – Within each shardspace, data is partitioned across shards by CONSISTENT HASH using sharding key (e.g. customer id)

+ For geo-distribution or hybrid clouds with linear scalability

- Requires two sharding keys: super_sharding_key and sharding_key
Fast Path for Sharding Key based access

- Connection pool maintains the shard topology cache
  - Upon first connection to a shard
    - Connection pool retrieves all sharding key ranges in the shard
    - Connection pool caches the key range mappings
  - DB request for a key that is in any of the cached key ranges goes directly to the shard (i.e., bypasses shard director)
Oracle Sharding | Direct Routing with Sharding Key

Oracle Universal Connection Pooling

• Must use direct routing for linear scalability & fault isolation
  – Requires building and passing the sharding key as part of connection check-out from the pool
  – All data required for a transaction is contained within a single shard

• Example: Using UCP Sharding APIs:

```java
OracleShardingKey keyMaryEmail =
    pds.createShardingKeyBuilder()
    .subkey("mary.smith@xyz.com", OracleType.VARCHAR2)
    .build();

Connection connection =
    pds.createConnectionBuilder()
    .shardingKey(keyMaryEmail)
    .build();
```
Oracle Sharding | Proxy Routing for Multi-shard Queries

For Workloads that cannot pass sharding key during connection check-out

• Applications connect to Query coordinator/Shard Catalog
  – E.g. SELECT GEO, CLASS, COUNT(*) FROM CUSTOMERS GROUP BY GEO, ROLLUP(CLASS);

• Coordinator rewrites the query to do most processing on the shards

• Supports shard pruning and scatter-gather

• Final aggregation performed on the coordinator
RAC Sharding | For Partition-amenable Applications

Virtual sharding of a non-sharded RAC database

• Affinitizes table partitions to RAC instances
  – Affinity gives better cache utilization and reduced block pings across instances
• Takes advantage of **direct routing API** of Sharding
  – Requests that specify sharding key are routed to the instance that logically holds the corresponding partition
  – Requests that don’t specify sharding key still work transparently
• Add sharding key to improve performance of OLTP operations; no changes to the database schema
  – *alter system enable affinity <TableName>*;
• Benefits
  – Gives RAC the performance and scalability of a Sharded Database with minimal application changes
Program Agenda

1. Oracle Sharding Overview
2. Customer Case Study – Epsilon
3. Sharding 19c Features
4. Sharding Use Cases
Epsilon – Oracle Sharding Use Case in Public Cloud

Gairik Chakraborty
Senior Director, Database Administration, Epsilon
Agenda

• Snapshot of Epsilon
• Oracle Sharding Use Case in Public Cloud
Epsilon at a Glance

• Epsilon is all-encompassing global marketing company, we are global leader in turning data-driven marketing into personalized customer experience and lasting relationships

• More than 9000 associates and 70 offices worldwide

• Largest permission-based e-mailer in the world, delivering over 75 billion emails annually

• World’s leading source of data with information covering over 1.5B individual records and 278M devices

• More than 2,000 global clients, including 26 of the Fortune 100
  ▶ 9 out of 10 Top Banks
  ▶ 8 out of 10 Top Retailers
  ▶ 9 out of Top 10 Pharmaceutical Companies
We deliver personalized connections, build loyalty and drive business for brands around the world.

**Data**

Know each of your customers on a meaningful level with Agility Audience, our premier solution offering unrivaled customer information, data resources and tools.

**Loyalty**

Create a one-of-a-kind loyalty program and grow long-lasting customer relationships with Agility Loyalty® and our full suite of loyalty capabilities and services.

**Digital Messaging**

Orchestrate personalized conversations taking your marketing where it needs to go with Agility Harmony®, the first platform built to be omnichannel from the ground up.

**Media Reach**

Optimize your media mix with the customer data, marketing technology and channels expertise that Epsilon and Conversant provide. We deliver personalized content that gets results.

**Multichannel**

Consolidate your approach to insights-driven multichannel marketing. Agility Unite combines unmatched consumer insights and machine learning to tailor every interaction to an individual’s wants and needs across any channel.
Oracle Sharding Use Case in Public Cloud
High Level OLTP Application Business Requirements

• Loyalty real time marketing solution with extreme availability, performance and scalability requirement

• Deployment should support any public cloud as well as on premises if needed – product offering is cloud first

• Platform supports real time POS integration with average call time from 200-500ms average and each API call can have multiple (20–100) SQLs internally

• Scale out solution which can grow flexibly based on workload demand

• Scalable, reliable and highly available infrastructure along with industry standard security and auditing
Why we like Oracle Sharding

• Loyalty marketing solution includes processing of high volume of complex financial transaction which requires strong multi version concurrency control, data protection, security, etc.
  ▪ NoSQL solutions are not very good fit for those use cases

• Application is using many complex joins and a solution needs to support the existing code with minimal change

• Use all other existing oracle feature like data consistency, security, availability, robust performance optimizer, backup and recovery which are key for business critical application deployment and are already part of Oracle Sharding framework

• Using Oracle Sharding, OLTP database can scale up and scale out and we expect to use lesser number of shards based on workload demand
Consideration before using Oracle Sharding

• Application data model needs to change to support sharding
• Identify transaction tables which can be sharded, reference table needs to be duplicated
• Select appropriate sharding key and sharding method
• Non default global database service created using GSDCTL needs to be used for connection
• Data access for OLTP should use sharding key as much as possible to avoid cross shard operation which is expensive in terms of execution time
• Application need to use driver which has sharding support in built. Example includes Oracle JDBC, OCI, ODP.NET (unmanaged driver), Oracle UCP, etc.
• Shard catalog setup is protected with active data guard with maximum availability protection and fast start failover enabled
OLTP System Deployment Architecture at Public Cloud
Application Service Placement: Current State

Site 1
- Application Load Balancer
- Web Service Call
- Web Servers – ODP.NET Connection Pool
  - OLTP
  - Batch
  - Reports
- FAN/FCF
  - OLTP Service
  - Batch Service
- Primary DB

Site 2
- Application Load Balancer
- Web Service Call
- Web Servers – ODP.NET Connection Pool
  - OLTP
  - Batch
  - Reports
- FAN/FCF
- Report Service
- Standby DB

Active Data Guard
Application Service Placement: Target State with Sharding

Site 1
- Application Load Balancer
- Web Service Call
- Web Servers – ODP.NET Connection Pool
  - OLTP
  - Batch
  - Reports
- FAN/FCF
  - OLTP Service
  - Batch Service
    - Shard Directors
    - Shard Catalog
- Primary Shards

Site 2
- Application Load Balancer
- Web Service Call
- Web Servers – ODP.NET Connection Pool
  - OLTP
  - Batch
  - Reports
- FAN/FCF
  - Shard Service
  - Shard Catalog
  - Shard Director
- Standby Shards

Active Data Guard

- FAN/FCF
- Shard Services
POC Result – Horizontal Scalability

Mixed Workload - TPS Analysis

Mixed Workload - Response time analysis
Best Practices

• Global service used for ODP.NET needs to have –notification set to TRUE using GDSCTL while configuring service for FAN

• Client side TNS should point to multiple shard directors for high availability instead of shard nodes.

• Backups should not run during chunk movement ( either during re-shard operation or manual chunk movement ) as that will not have correct data layout.

• During recovery, validate or recover shard option may be needed to sync it with shard catalog
Future Plans

• Collaborate with oracle engineering team to convert Epsilon loyalty solution to be sharding capable, configure application for zero downtime using features like application continuity during shard failure or re-shard operation, etc.

• Plan to deploy Oracle sharding for many customers as next generation true horizontally scalable multiple cloud global solution across any region as part of Epsilon’s global loyalty platform deployment plan
Program Agenda

1 Oracle Sharding Overview
2 Customer Case Study – Epsilon
3 Sharding 19c Features
4 Sharding Use Cases
Oracle Sharding Evolution

Oracle 19c
- Multiple PDB shards per CDB
- Multiple Table Families
- Multiple Proxy Coordinators
- Global Sequences
- Parameter setting across shards

Oracle 18c
- User-defined Sharding
- PDB Sharding
- RAC Sharding
- Mid-tier Sharding
- Update-able Duplicated Tables

Oracle 12.2
- Automated Deployment with Replication
- System-managed & Composite Sharding
- Centralized Sharded Schema Management
- Direct & Proxy Routing
- Online Scale-out w/Auto-resharding
- Online Scale-in
- Chunk Move and Split
Oracle Sharding | 19c Features

• Support multiple PDB-shards in the same CDB
  – Allows consolidation of sharded databases (SDBs)

• Enables shard catalog's ADG standbys to act as multi-shard query coordinators
  – Improves HA & scalability of proxy routing for reporting and analytical workloads

• Multiple Table Families
  – Allows an SDB to support multiple table families, each of which can be sharded with a different sharding key

• Support for fast data load
  – Data is split and loaded directly to shards in parallel
Consolidation of Sharded Databases
Consolidation of Sharded Databases

- DB-A
  - SH-A1
  - SH-A2
  - SH-A3

- DB-B
  - SH-B1
  - SH-B2
  - SH-B3

- DB-C
  - SH-C1
  - SH-C2
  - SH-C3

CDB-1
CDB-2
CDB-3
Consolidation of Sharded Databases
Consolidation of Sharded Databases
Multitenant Database for Microservices

Reduce the cost and complexity of data management for new apps

• Microservices use PDBs
  – Each microservice allocates its own dataset (can be a PDB, Schema, or a subset)
  – Each microservice has private data model
• Multitenant database containers deliver
  – Manage many databases as one
  – Secure separation of data
  – Easy sharing and querying of data across PDBs
• Database views isolate what each microservice sees
PDB Sharding for Microservices

**Scalability, fault isolation and geo-distribution**

- Multitenant simplifies manageability
- 19c PDB Sharding feature adds benefits of sharding to multitenant
  - Each PDB can be sharded individually across multiple CDBs
- One of the main reasons for using microservices is the ability to scale each service individually
  - Allows to save a lot of resources comparing to scaling a monolithic application.
- PDB Sharding also provides fault isolation and geo-distribution for microservices
Multiple Query Coordinators

- Enables shard catalog's Active Data Guard standbys to act as multi-shard query coordinators
  - Improves scalability of proxy routing for reporting and analytical workloads
  - Coordinators can be placed in different availability domains and geo regions
Multiple Table Families

- Allows you to make a trade-off:
  - Reduces amount of duplicated data
  - May increase time for executing joins

- Supported for system-managed sharding (by consistent hash)
  - All table families share the same set of chunks

- For direct routing to shards, clients need to provide table family ID (name of the root table) in addition to the sharding key value
High Speed Data Load

- Utilized fully parallel direct-to-shard data loader
- Sharded architecture scales the CPU, flash, and network interfaces
- Can add shards as needed to accommodate higher ingest rates
- Architecture applies to IIoT, IoT and edge compute scenarios

[Diagram showing data sources splitting to multiple loaders, connected to data shards]

Open Source
https://github.com/oracle/db-sharding
Evaluation Results

- Need to load 30m rows every 5 min
- Loading from S3 to Redshift
  - 3 to 4 minutes
- Loading from Oracle Object Storage to Shards
  - 500k rows / second
  - Full load in 1 minute
  - 4 minutes of headroom for queries
Program Agenda

1. Oracle Sharding Overview
2. Customer Case Study – Epsilon
3. Sharding 19c Features
4. Sharding Use Cases
Use Case #1 | Scalability w/ Oracle Sharded Database

Order Entry application sharded by customer_id needs massive linear scalability

System-managed SDB

Oracle Cloud

Shard-level Active Data Guard

Availability Domain 1
(100 Primary Shards)

Availability Domain 2
(100 HA Standby Shards)

Graph showing TPS vs. # of Shards
Use Case #2 | Ultra Availability w/ Oracle Sharded Database

Airline Ticketing requires - failure of a database to not take down an entire service

User-defined SDB
Sharded over Airline_id

Shard Outage or Slowdown has Zero Impact on Other Shards

Application remains available on other shards
Use Case #3 | Oracle Sharded Database for Data Sovereignty

Global Financial Customer wants to place data in specific geographies for privacy compliance

User-defined Sharded Database
Geo-distribution for Data Sovereignty & Privacy Compliance

Customers USA

Customers UK

Customers China
Use Case #4 | **Oracle Sharded Database for Data Proximity**

Billing System Sharded by Geo and then by Account_id - With Composite Sharding

**Composite Sharded Database**

Geo-distribution for Data Proximity & Linear Scalability

Customers West

Customers East
Use Case #5 | Multi-cloud Deployment w/ Oracle Sharding

Cloud-based SaaS Provider needs to support cloud workload portability with no lock-in

User-defined Sharded Database

Customers Americas

Azure

Customers Europe

Oracle Cloud

Customers Asia

AWS
Case #6 | Cloud Bursting with Oracle Sharded Database

Online Retailer needs to elastically scale for seasonal demand & new product launch

- Scale capacity for **holiday shopping** – transactions, customers and data
- Scale capacity for new customers from **new product launches** or services
Oracle Sharding = Best of NoSQL + Best of Oracle

• Oracle Sharding Key features
  – Automatic data distribution across a set of commodity servers (using consistent hash)
  – Built-in and customizable replication (Active Data Guard and GoldenGate)
  – Data Sovereignty, Linear scalability, Ultra-availability
  – Plus: Composite and user-defined sharding, duplicated tables, centralized management, ...

• With the strength of Oracle RDBMS Engine
  – Familiar and powerful relational data model and query language, including full native support for modern datatypes like JSON, Spatial ...
  – Transactions, recovery, in-memory ...
  – Industrial-strength and scalable storage engine
  – Data security and access controls
  – Familiar to current staff, and supported by Oracle
Summary | 3 Takeaways

• Complete platform for sharding an Oracle database

• Licensing
  – Oracle Cloud: Extreme Performance
  – On-premises: Enterprise Edition + (either Active Data Guard or GoldenGate or RAC)

• Ideal for applications whose dominant access pattern is via a sharding key and that require:
  – Linear scalability for workload, data size and network throughput
  – Data sovereignty
  – Ultra availability
Oracle Sharding | Sessions and Demos in OOW 2018

Monday, Oct 22nd

• 9 AM | High Availability and Sharding Deep Dive with Next-Gen Oracle Database [TRN4032] | Moscone West-3007

• 3.45 PM | Industrial-strength Microservice Architectures with Next-Gen Oracle Database [TRN5515] | Moscone West-3003

Tuesday, Oct 23rd

• 12:30 PM | Oracle Sharding: Geo-Distributed, Scalable, Multimodel Cloud-Native DBMS [PRO4037] | Moscone West-3007

• 3.45 PM | Data and Application Modeling in the Brave New World of Oracle Sharding [BUS1845] | Moscone West-3007

Thursday, Oct 24th


• 1 PM | Using Oracle Sharding on Oracle Cloud Infrastructure [CAS5896] | Moscone West-160

Demos

Sharding Demo Booth - 1615

• High Availability, Moscone South Exhibition Hall

Contact:

Nagesh Battula
Sharding Product Management
srinagesh.battula@oracle.com

@OracleSharding & @NageshBattula
https://www.oracle.com/goto/sharding
<table>
<thead>
<tr>
<th>Monday, Oct 22nd</th>
<th>Tuesday, Oct 23rd</th>
<th>Wednesday, Oct 24th</th>
<th>Thursday, Oct 25th</th>
</tr>
</thead>
<tbody>
<tr>
<td>9:00am – 9:45am</td>
<td>10:30am – 11:15am</td>
<td>11:15am – 12:00pm</td>
<td>9:00am – 9:45am</td>
</tr>
<tr>
<td>11:15am – 12:00pm</td>
<td>11:15am – 12:00pm</td>
<td>11:15am – 12:00pm</td>
<td>11:00 – 11:45</td>
</tr>
<tr>
<td>12:30pm – 1:15pm</td>
<td>12:30pm – 1:15pm</td>
<td>12:30pm – 1:15pm</td>
<td>12:00pm – 12:45</td>
</tr>
<tr>
<td>1:45pm – 2:30pm</td>
<td>1:45pm – 2:30pm</td>
<td>1:45pm – 2:30pm</td>
<td>1:00pm – 1:45pm</td>
</tr>
<tr>
<td>3:45pm – 4:30pm</td>
<td>3:45pm – 4:30pm</td>
<td>3:45pm – 4:30pm</td>
<td>3:45pm – 4:30pm</td>
</tr>
<tr>
<td>4:45pm – 5:30pm</td>
<td>4:45pm – 5:30pm</td>
<td>4:45pm – 5:30pm</td>
<td>4:45pm – 5:30pm</td>
</tr>
<tr>
<td>5:45pm – 6:30pm</td>
<td>5:45pm – 6:30pm</td>
<td>5:45pm – 6:30pm</td>
<td>5:45pm – 6:30pm</td>
</tr>
</tbody>
</table>
Oracle Sharding | Resources

Oracle Maximum Availability Architecture
http://www.oracle.com/goto/maa

Social Media:
https://twitter.com/NageshBattula
https://twitter.com/OracleSharding

https://www.oracle.com/goto/oraclesharding
https://nageshbattula.com