Oracle on the distributed databases market: Globally Distributed Database 23ai (Sharding and Raft)

Franck Pachot, Developer Advocate

yugabyteDB

@FranckPachot

Franck Pachot

Developer Advocate
at Yugabyte

25+ years in databases, dev and ops, consulting
Oracle Certified Master, AWS Data Hero

fpachot@yugabyte.com
dev.to/FranckPachot
@FranckPachot

Running a database on multiple servers

Traditional monolithic database:

- one set of processes and shared memory (instance)
- one stream of write-ahead logging (online redo log)
- one set of database files (controlfile, datafiles)
- they contain metadata (dictionary) and data (tables rows and index entries)

Monolithic is a problem for:

- application downtime on instance crash, media failure, OS or DB upgrades
- a single physical location: single region for latency and data residency

Solution: distribute and replicate to multiple servers, racks, data centers, regions

Running a database on multiple servers

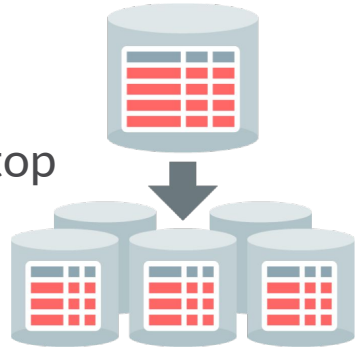
Solution: distribute and replicate to multiple servers, racks, data centers, regions

Solution 1: break the monolithic component to be distributed over the network

- multiple instances on top of one set of files: Oracle RAC
- one instance on top of distributed storage: Oracle ASM, dNFS, Exascale
- multiple read-only instances that can replace the primary: Oracle Data Guard
- Distributed SQL databases with native distribution and replication
  Spanner, CockroachDB, YugabyteDB, TiDB

Solution 2: shard to monolithic databases with a coordinator on top

- by the application (the code connects to one shard)
- or with a sharding option (Oracle Sharding in 12c)



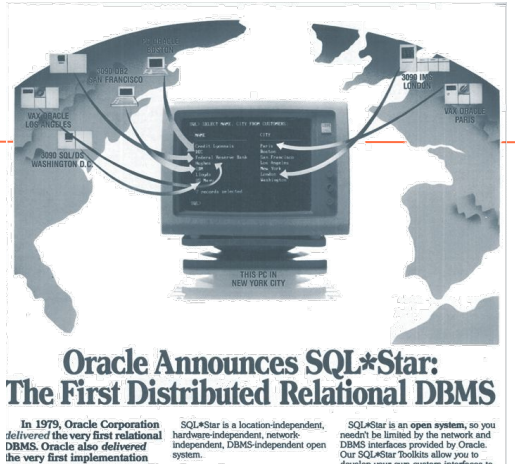
Not so new...

Sharding on top of multiple databases is not new

- table partitioning on a key
- distributed transactions (2PC)
- database link, transparent gateway

At that time, Oracle was advocating for RAC as better than shared-nothing architecture

But cloud infrastructure evolved



Partitioning tables and indexes

Choose a **partitioning key**

- rows with the same value for the partitioning key are in the same partition
- rows with different values may be in different partitions

Partitioning strategies:

- by **list**, e.g. Example Country_Code in ('D', 'A', 'CH',) into the 'DACH' partition
- by **range**, e.g: Order_Date between 2024-01-01 and 2024-12-01 in 'P2024'
- by **hash**, e.g: mod(Cust_ID,2)=0 fo to 'EVEN', mod(Cust_ID,2)=1 to 'ODD'
- by reference, on a foreign key to a partitioned parent

Partition are like tables from an operation point of view, all with same structure physically placed with tablespaces, reorganized per partition, parallel query...

From Partitioning to Sharding

Partitioning can break the monolithic tables

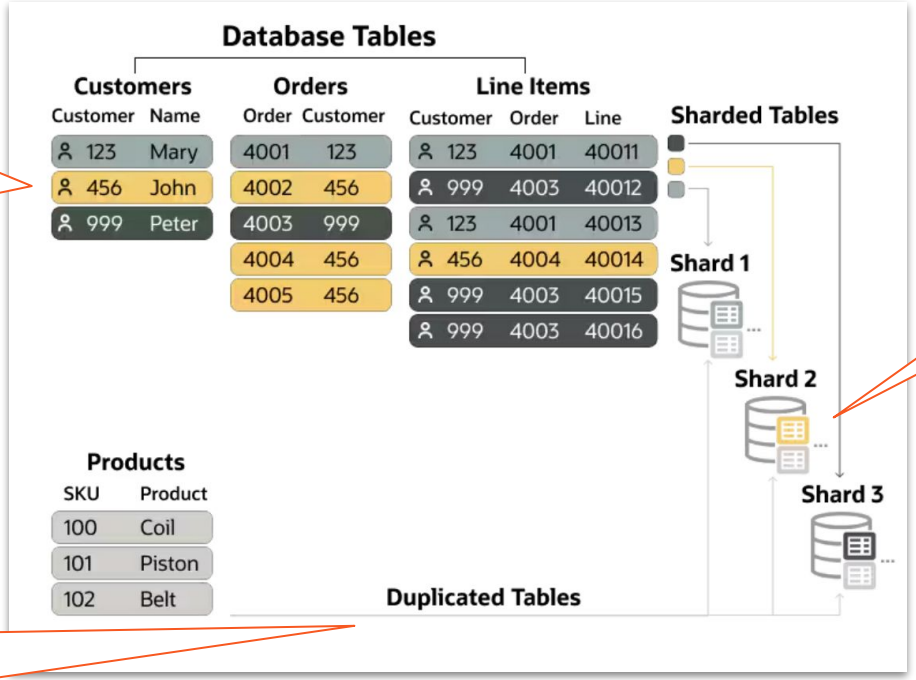
- located to different storage (**tablespaces**)
- cached by different **instances** (RAC)
  application connects to different RAC instances
- stored in different databases (**sharding**)
  application provides a sharding key

Examples:

- store old data in cheaper storage
- distribute the load to multiple instances
- store customer data in their own region (GDPR)

Relational tables in a shard (a shard is an Oracle database)

One SHARDING KEY for a hierarchy of tables (family)



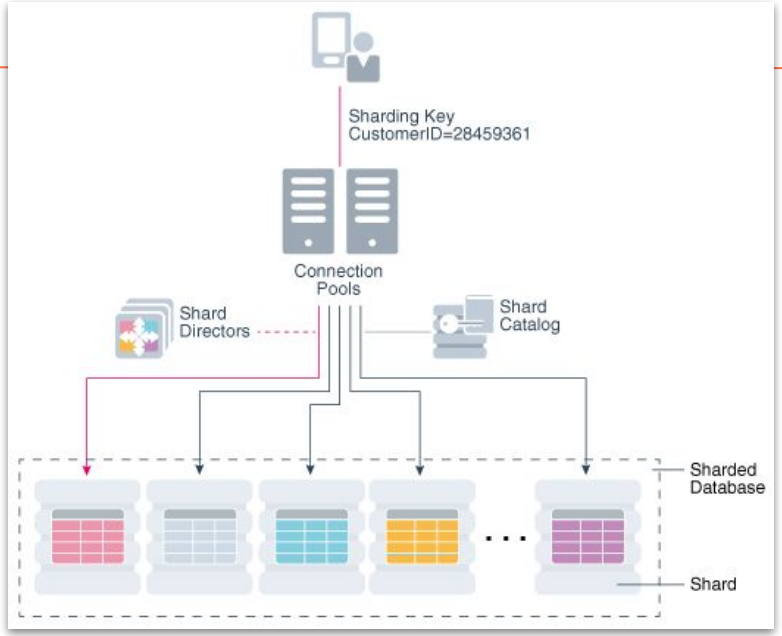
One Oracle Database per SHARD

Tables out of the hierarchy are broadcasted

SHARD = PDB

No cross-shard referential integrity (Foreign Key) or global indexes (unique key)

Oracle Sharding



Leverages existing features

- Partitioning strategies
- Global Data Services
- Distributed Transactions

Difference with partitioning

➖ one shard key: no global indexes, no cross-shard foreign key

➕ additional strategy: consistent hashing for transparent re-sharding

Consistent Hashing

List and Range partitioning are good when we know the range of values
To distribute before knowing the range of value: hash sharding
(but used only with equality predicates)

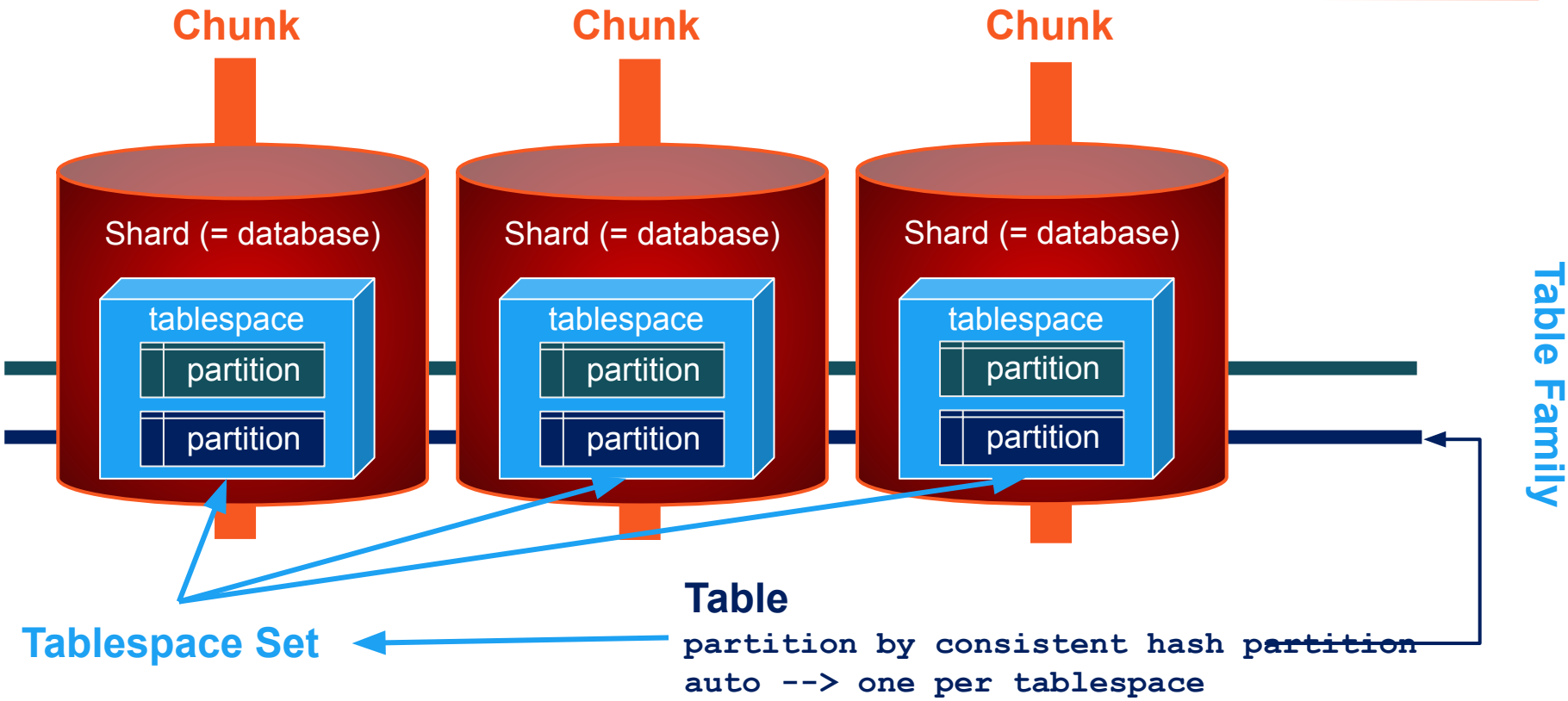
HASH partitioning uses modulo on a hash function, but
splitting a partition has to re-write it (hint: use powers of two to keep it balanced)

For the best distribution: consistent hashing

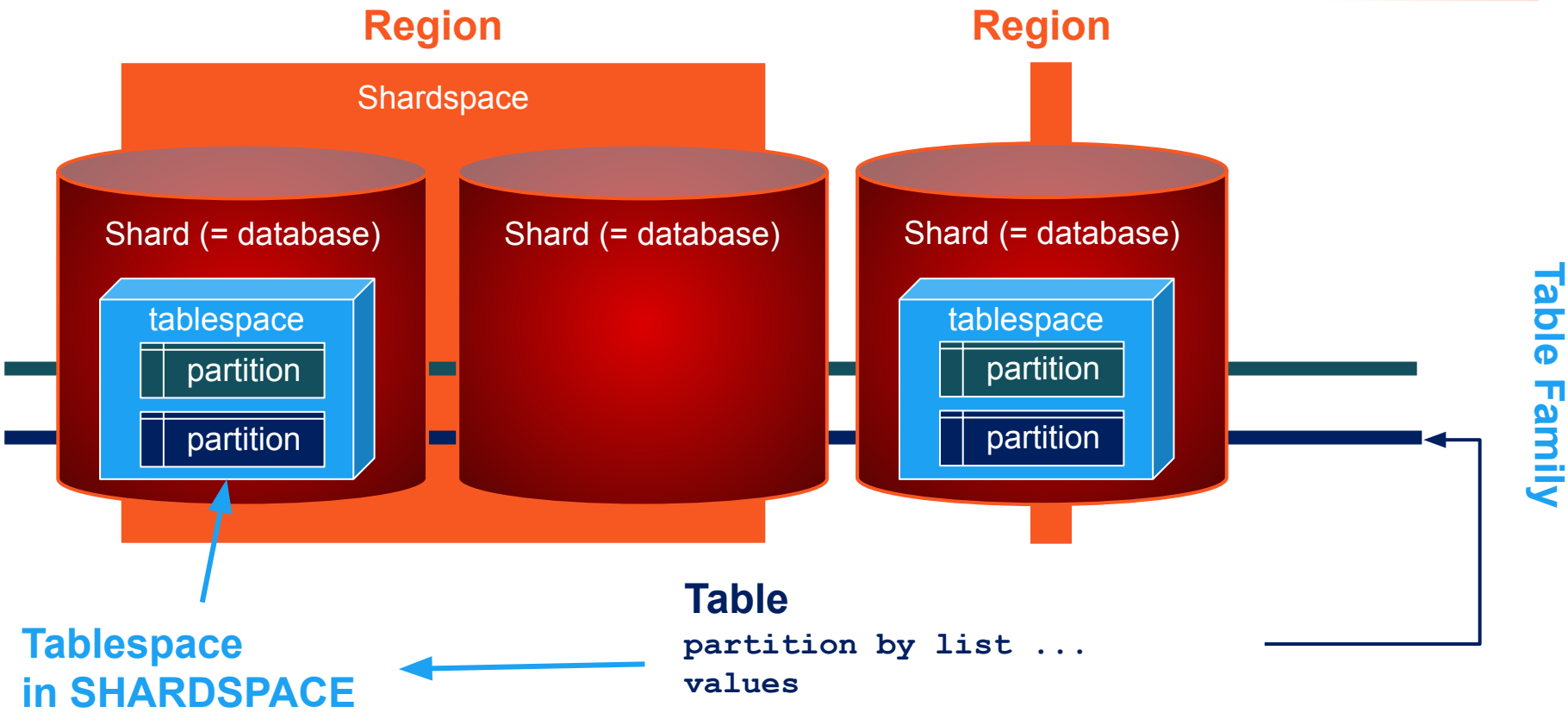
- applies a hash function to the sharding key (1 to 2^32)
- partition by range of this hash function (chunks)

It makes it easier to split a partition when the size grows
-> System-Managed Sharding (automatic mapping to shards)

System-Managed Sharding with Consistent Hashing and Tablespace Sets



User-Defined Sharding with List Partitioning and ShardSpaces



Composite Sharding

Combines both:

- System-Managed sharding
  to distribute the primary key to the tablespace set within a shardspace

- User-Defined sharding
  for data placement of partition sets (on a sharding key) to tablespace sets

```
partition set by list (region)
... partition by consistent hash ... auto
partitionset ... values ... tablespace set ...
```

Sharding + Replication

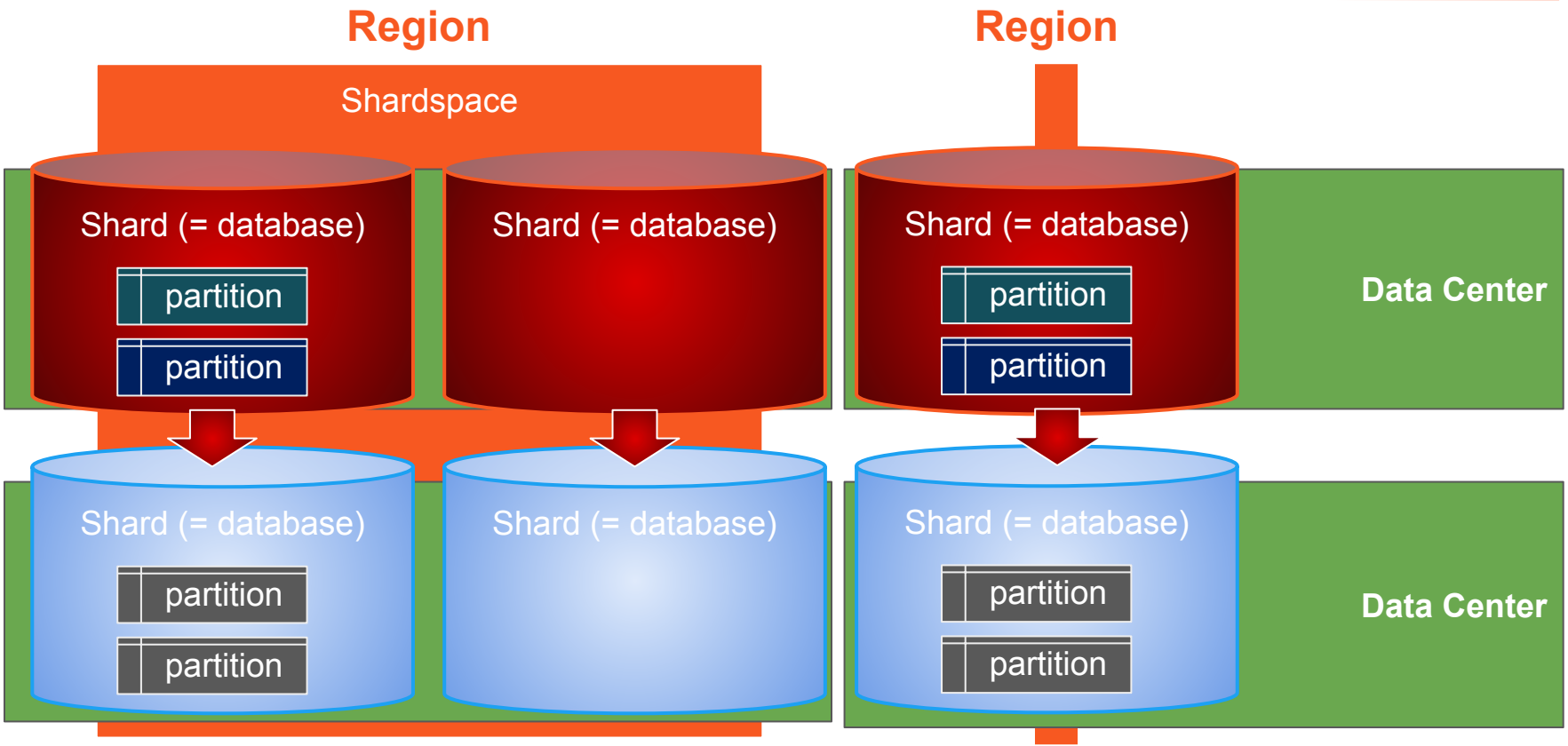
- Define a sharding key for **data placement**
- Distribute on the primary key for **distribution**

In term of High Availability, Sharding doesn't provide resilience
but it avoids full failure (one region down, the other continues)

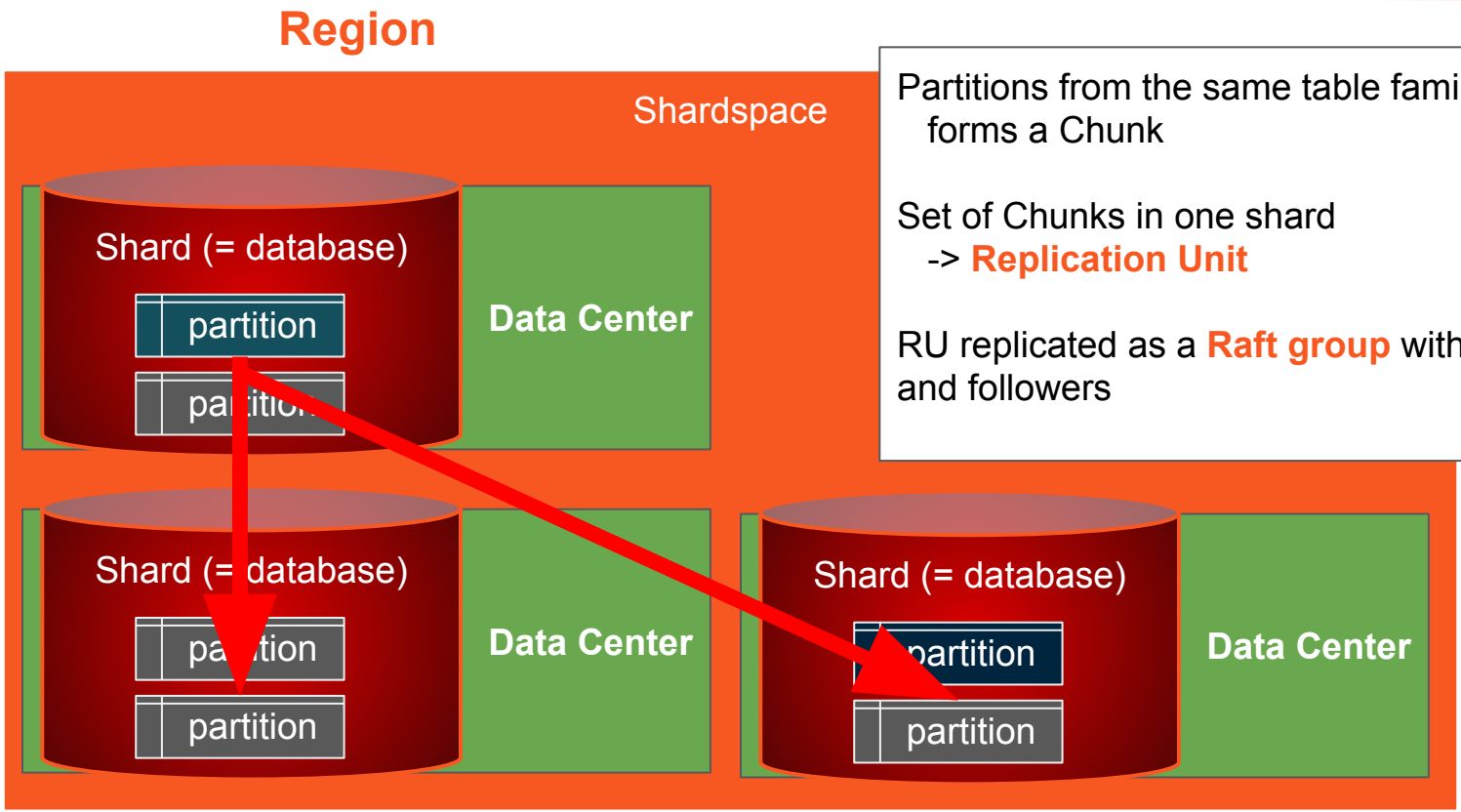
But that's not enough: we need to replicate for **high availability**

- Data Guard: one DG configuration per shard (primary + standbys)
- Golden Gate: deprecated for sharding
- Sharding Native Replication: Raft replication per chunk sets

Data Guard



Sharding Native Replication



Partitions from the same table family forms a Chunk

Set of Chunks in one shard
-> **Replication Unit**

RU replicated as a **Raft group** with a leader and followers

Raft Consensus

In a Raft group, only one replica has a leader lease
- executes DML (consistent reads and writes)

Each Replication Unit is a raft group
- RF3: one leader and two followers
- RF5: one leader and four followers

Reads and Writes are sent to the Raft leader (according to the shard key)

Writes are replicated to the followers (asynchronous, commit is synchronous)
If the leader is inaccessible, a majority of the followers can elect a new leader

Raft Consensus

Resilience and consistency with Raft:

- Leader waits for the majority of followers to acknowledge (write quorum)
- A majority of followers can serve the latest state (read quorum)
- They elect a new leader, which can then guarantee a consistent state alone

Sync to quorum for each write is not efficient

- Oracle waits only for the commit record
- Other writes are asynchronous
  - in case of new leader election, some transactions are aborted `(ORA-03970)`
  - can be re-played by the driver (Application Continuity)

Logical Change Record

Data Guard replicates the binary changes to all blocks

- the standby is a binary copy of all files

Raft alternative (Shard Native Replication) is a logical replication

- similar to Streams (use by logical standby, GoldenGate)

With Shard Native Replication, DML generates two transactions logs:

- online redo log records (physical, to recover blocks)
- LCRs (logical, to replicate table row changes)

What about DDL?

Shard Native Replication is about DML,
but DDL must be synchronized between shards

DDL is run on the central catalog (GDS) and propagated to the shards

- Barrier DDL is synchronous and blocks DML on the shards
  (example: rename a column)
- Some Non-Barrier DDL just synchronizes with the replication
  (example: drop a column)

What is Oracle Globally Distributed Database?

It is a Distributed Database? Yes
- it distributes the data on a sharding key,
- connect to a coordinator for distributed transactions

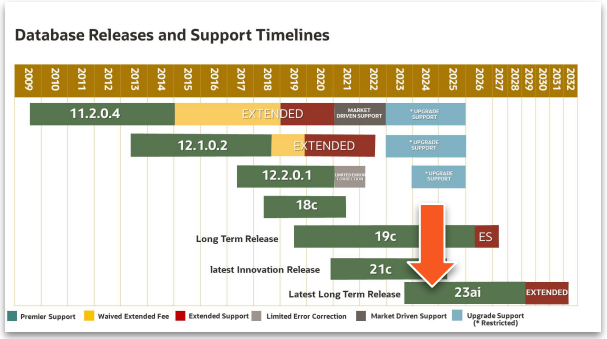
What is different from Distributed SQL Database?
- not all SQL features are available at global level (foreign keys, unique constraints)
- application needs to use a sharding key (SQL database must provide data independence)

🔍**Sharding in Other RDBMS:**
- Vitess (MySQL)
- Citus (PostgreSQL)
- Aurora Limitless (AWS)

🔍**Distributed SQL RDBMS:**
- Spanner
- TiDB
- CockroachDB
- YugabyteDB

It is Sharding with Shard Native Replication + Global DDL (in GDS)

Licensing / Availability



Licensing:

- Not in Standard Edition
- Max 3 shards if no ADG, GG or RAC option

Availability:

- Linux, Solaris, Windows: end 2024?
- ODA, Exadata: 22-Jul-2024
- Cloud (BaseDB, ExaCS): 2-May-2024

Oracle Globally Distributed Database

High Availability

- Oracle Database FREE
- Standard Edition 2
- Enterprise Edition
- Oracle Database Appliance
- Exadata
- Exadata Cloud Service / Cloud@Customer
- Oracle Base Database Service Standard Edition
- Oracle Base Database Service Enterprise Edition
- Oracle Base Database Service Enterprise Edition - High Performance
- Oracle Base Database Service Enterprise Edition - Extreme Performance

Formerly called Oracle Sharding.

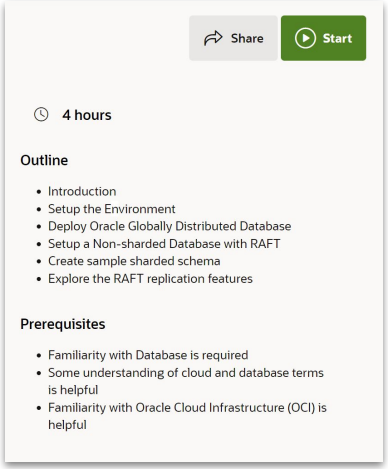
Free: Use is limited to three shards

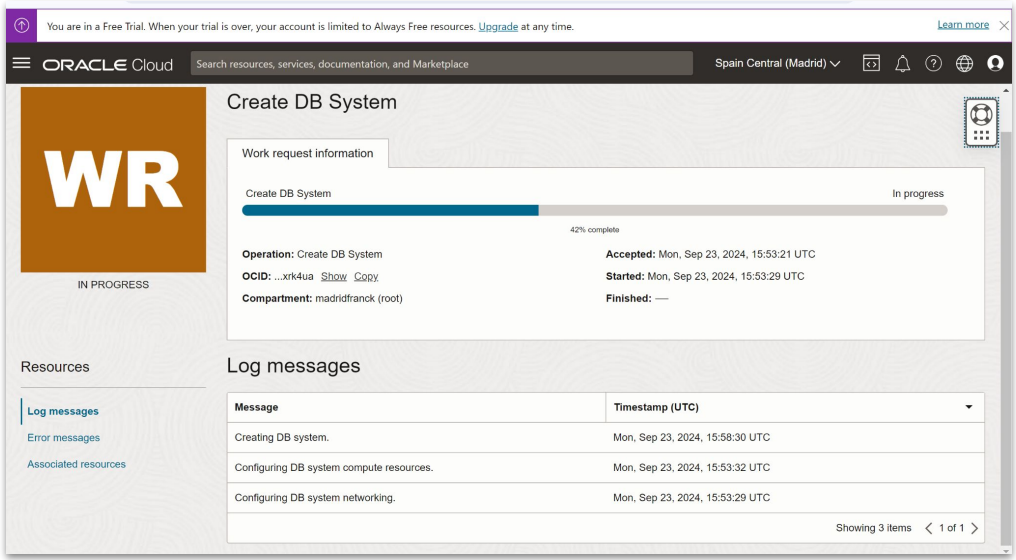
EE, **ODA**, and **Exa**: No limit on the number of either primary shards or standby shards if every shard has an Oracle Active Data Guard, Oracle GoldenGate, or Oracle RAC license. Without an Oracle Active Data Guard, Oracle GoldenGate, or Oracle RAC license, use is limited to three primary shards, with basic Data Guard standbys.

BaseDB EE, **BaseDB EE-HP**, and **Authorized Cloud Environments**: Use is limited to three primary shards; there is no limit on the number of standby shards.

ExaCS/CC and **BaseDB EE-EP**: No limit on the number of either primary shards or standby shards.

No Demo :(





Oracle LiveLabs workshop (on OCI):
https://apexapps.oracle.com/pls/apex/r/dbpm/livelabs/view-workshop?wid=835

E-mail:
fpachot@yugabyte.com

Open Community Hours:
https://www.youtube.com/@YugabyteDB

Blogs:
dev.to/FranckPachot
blog.yugabyte.com/author/fpachot

Twitter:
@FranckPachot

LinkedIn:
www.linkedin.com/in/franckpachot

yugabyteDB