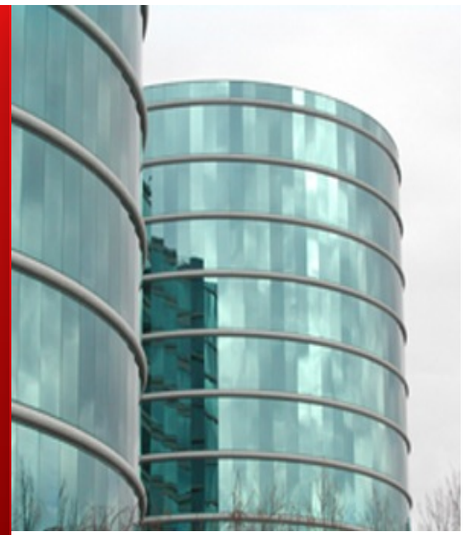
ORACLE

Oracle NoSQL Database – Application Data Modeling

Robert Greene
NoSQL Database Product Management

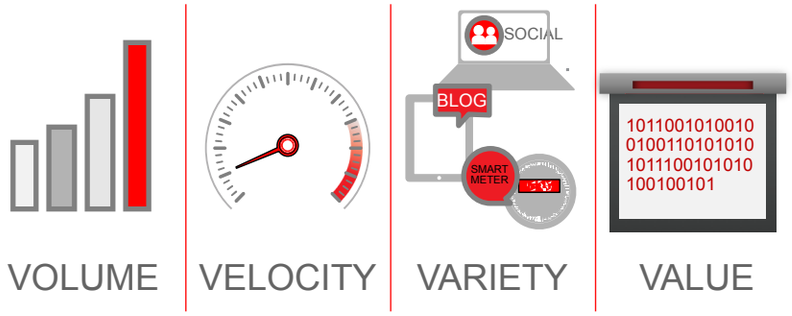


ORACLE
NOSQL DATABASE

Agenda

- The Oracle NoSQL Database
- NoSQL background
- NoSQL at Oracle
- Developing with Oracle NoSQL Database
  - Architecture
  - Deploying a cluster
  - Developing an application
- Use Cases

What is Oracle NoSQL Database?



Non-relational database designed for **cost effective simple queries** of high volume, velocity & variety data. Provides high performance & availability data storage of Big Data's **simple data** using a scale-out of servers design.

Bottom line: NoSQL is a disruptive technology that is all about **"data scalability at cost"** first and foremost. That is what is interesting to customers.

Sure, there are specific technical features that are also important to the NoSQL technology story, but they are secondary. In fact, as we'll see in a few slides, each NoSQL database implementation is a little different from the others. Just like RBDMS implementations were in the early to mid-eighties.

However, the main reason that NoSQL database technology was created was to provide more efficient **and more cost effective** data management for fast, flexible, distributed, highly available applications that what is provided today in general purpose RDBMS systems.

-- Not Part of the Script –
Additional notes:

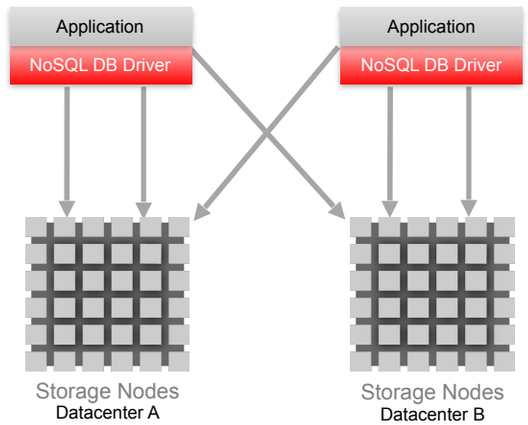
This is not new technology. Non-relational databases, and in particular, key-value databases have been around for a long time. Oracle Berkeley DB is a good example of a key-value database that has been around for 20 years. What is innovative and disruptive about this technology is the fact that scalable, horizontal distribution of data and high availability via replication are *built-in* to the architecture of the product. This combination of simple get/put, non-relational operations is along with the distributed, scalable, highly available qualities of Big Data is what makes it disruptive.

Oracle NoSQL Database

Scalable, Highly Available, Key-Value Database

Features

- Flexible Key-Value Data Model
- ACID transactions
- Horizontally Scalable
- Highly Available
- Elastic Configuration
- Simple administration
- Intelligent Driver
- Commercial grade software and support



Java SE 6 (JDK 1.6.0 u25)+; Solaris or Linux

ORACLE

The solution for these challenges is Oracle's NoSQL Database. It's a distributed key-value database.

Think of key-value as a two column table – a key and a value. Give key-value example: give customer ID as the key and customer profile (and other information) as the value/record.

NoSQL Database is good for app's where just need fast, simple db requests (key/value lookup, no join's), use a schema defined dynamically at runtime by the application itself, and have extreme scalability requirements.

The k-v technology isn't new: mainframe ISAM, Berkeley DB and other products have implemented this kind of storage in the past. What's new is:

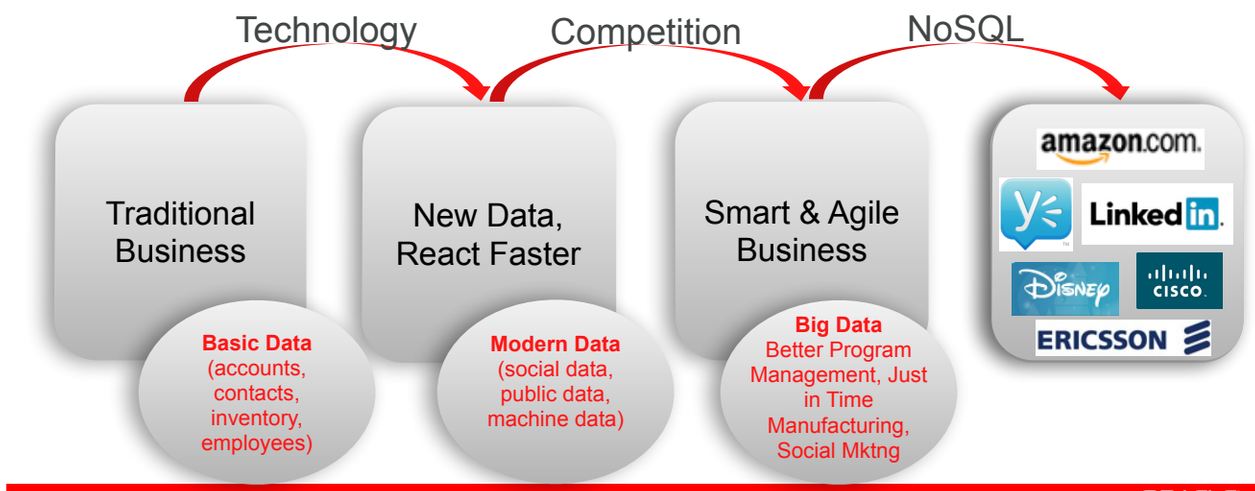
a) Distributed, high volume, high velocity nature of the storage and the applications that use them,
b) Ability to create multiple, distributed indexes and hash to/look up the appropriate one, instead of just creating a single or limited set of indexes.

Agenda

- The Oracle NoSQL Database
- NoSQL background
- NoSQL at Oracle
- Developing with Oracle NoSQL Database
  - Architecture
  - Deploying a cluster
  - Developing an application
- Use Cases

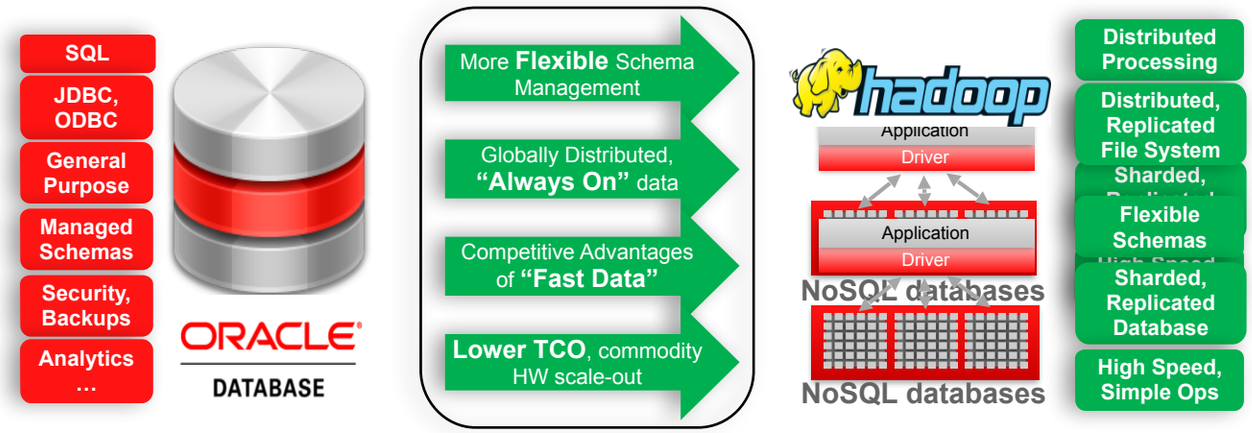
NoSQL

Where did it come from and What is it?



NoSQL and Big Data

What's changing ?



Things to highlight…

- **RDBMS is not going away – it's been around for the last 30 years and is the backbone of most transactional and lots of analytic systems. It's rich in it's capabilities with SQL, JDBC/ODBC, support for complex schemas, security, backup, high availability, etc.**

  The internet and internet-related businesses in general have brought us huge increases in the variety and volume of data. Additionally, the problem of rapidly evolving data requirements, which tend to take much longer to manage within the processes around conventional relational databases.

- **Customers and Application Developers need more flexible data management. Business are looking for support of real time data feeds and high availability for highly distributed data sets as some of the requirements to keep the competitive advantage, while not increasing the overall cost of operation.**

  New disruptive technologies that have recently emerged and are getting more and more widely adopted

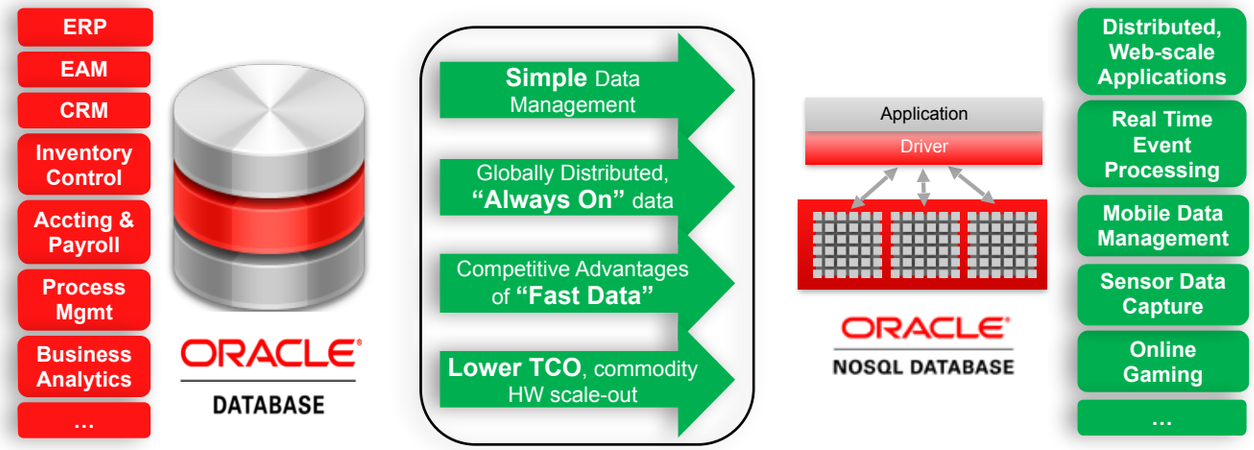
- **These technologies are not replacements for RDBMS. They are specifically designed to address the specific needs of *certain* kinds of data processing and management.** In fact, they can all co-exist with all 3 technologies, side by side on the same stack.

  Hadoop is not the same as NoSQL – they are separate technologies, each is useful within a specific content.

  Hadoop and NoSQL are both built to run on commodity hardware, addressing the low TCO scale-out requirements. They can also run on Oracle engineered systems (BDA).

Oracle NoSQL Database

Where is it used?



things to highlight…

- **RDBMS is not going away – it's been around for the last 30 years and is the backbone of most transactional and lots of analytic systems. It's rich in it's capabilities with SQL, JDBC/ODBC, support for complex schemas, security, backup, high availability, etc.**

  Some examples of applications that require the features of an RDBMS include [see list on left hand side]. These applications require transactions, multi-table complex schemas, security, SQL, etc.

  ERP == Enterprise Resource Planning
  EAM == Enterprise Asset Management
  CRM == Customer Relationship Management

  The internet and internet-related businesses in general have brought us huge increases in the variety and volume of data. Additionally, the problem of rapidly evolving data requirements, which tend to take much longer to manage within the processes around conventional relational databases.

- **Web-scale applications typically need very low-latency, simple read/write operations. Business are looking for support of real time data feeds and high availability for highly distributed data sets as some of the requirements to keep the competitive advantage, while not increasing the overall cost of operation.**

  NoSQL Database is designed to handle this kinds of requirements – high volume, low-latency simple read/write operations for globally distributed data sets.

- **NoSQL Database is not a replacement for RDBMS. It was specifically designed to address the specific needs of certain kinds of data processing and management.** In fact, it typically co-exists with both the Oracle Database and Hadoop, side by side on the same stack.

  Oracle NoSQL Database is designed to handle a huge volume of simple read/write operations (hundreds of thousands to millions of operations per **second**). It is perfectly suited for applications like the list on the right. Basically, any time that you need to read and write lots of records per second and require low-latency read and write operations. These horizontal use cases can be applied to multiple businesses and industries.

  - Most companies today have large web-based applications for both internal and external-facing operations.
  - Many companies perform real-time or near-real-time operations, where capturing and evaluating incoming data is a crucial part of the business. Evaluating that data requires looking up additional information like profile data (customers, accounts), configuration information (telco switches, manufacturing systems) and contextual information (past history, related events, warnings, alerts), etc. NoSQL Database provides a scalable, distributed low-latency platform for executing those low-latency reads.
  - D2D (Device-to-Data Center), Mobile Device content Management and IoT (Internet of Things) have many large companies pushing the limits of scalability in order to **capture** and **interact** in real-time with millions of devices in the field. NoSQL Database offers a perfect platform that answers the requirements for Always On, Distributed, Scalable, High Performance, Simple/Flexible data management at a lower TCO.
  - NoSQL Database is also commonly used Online Gaming applications. These types of applications tend to combine the requirements of Distributed Web-scale application processing **plus** Real Time Event Processing.

  NoSQL Database can run on commodity hardware, addressing the low TCO scale-out requirements. It can also run on, and is included with, Oracle engineered systems (BDA).

  Hadoop (not shown here) is not the same as NoSQL – they are separate technologies, each is useful within a specific content. Hadoop is designed primarily for batch processing. NoSQL Database is designed for low-latency, high volume simple read/write operations (when you need to access the data *now*).

Agenda

- The Oracle NoSQL Database
- NoSQL background
- NoSQL at Oracle
- Developing with Oracle NoSQL Database
  - Architecture
  - Deploying a cluster
  - Developing an application
- Use Cases

The NoSQL Landscape

Pioneers – Built their own

Oracle continues to invested in NoSQL as a key data management technology

Global Enterprise Technology Adopters

Source: NoSQL Market Forecast 2013-2018 by Market Research Media

ORACLE

The NoSQL landscape is often characterized by the few pioneers who hired bright engineers and built their own databases and deployed them into solutions. Cassandra from Facebook, DynamoDB from Amazon, Voldermot from LinkedIn are some of the notable NoSQL technologies in the market today.

Click
And now as the technology matured we see some large enterprises adopting different NoSQL technologies.

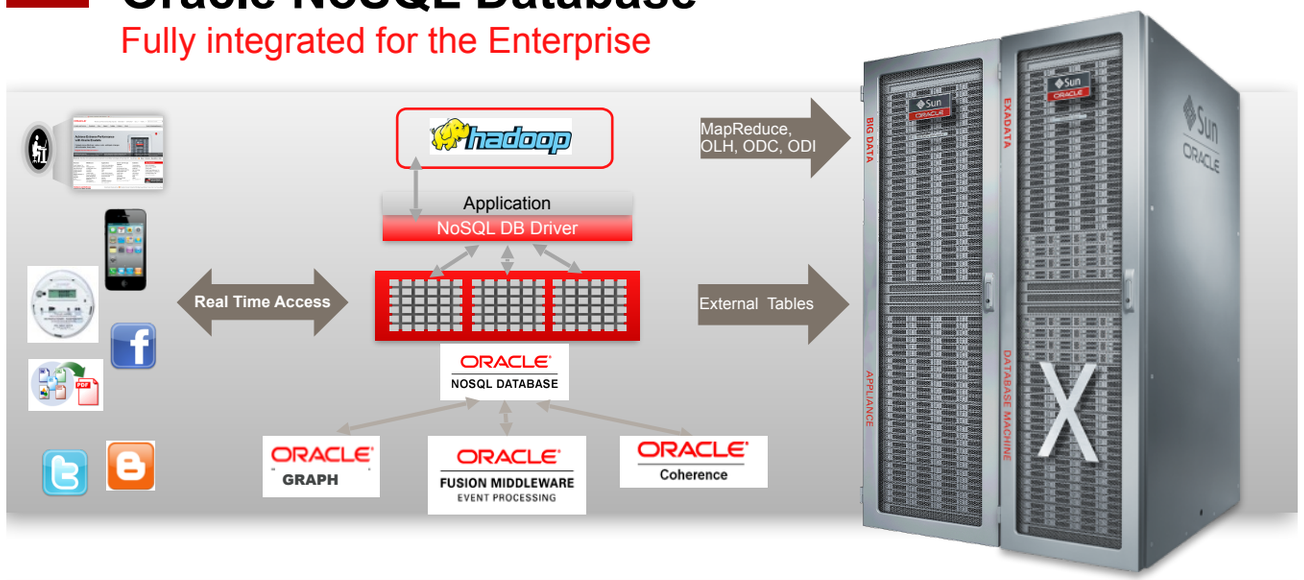
Click
It is now a real market growing 21% annually, much faster than the rest of the IT market! It is slated to be as large as $3.4 B by 2018.

Click
Oracle is not just a “relational database company”. Oracle a “data management” company and our goal is to address the data management needs of our enterprise customers. Oracle is the only major database technology provider to offer an enterprise class, integrated NoSQL database product as part of our data management technology solution stack.

==========

Oracle NoSQL Database

Fully integrated for the Enterprise



The story for this slide is really our Enterprise Integration approach to NoSQL development.
Oracle understands the Enterprise's needs better than any data management company, having supported the largest market share of Enterprise level customers than any other vendor.

In that regard, we understand the value of integrating infrastructure components to work together toward more holistic solutions.

So, we've integrated the **O**racle **N**oSQL **D**ata**b**ase (ONDB) into an increasing number of oracle products and solutions including: Oracle Database, Hadoop, Coherence, OEP (Oracle Event Processing), Spatial and Graph along with a growing number of Oracle complete vertical solutions like Telecom and Commerce.

Choose the RIGHT tool for the job

Hadoop	Oracle NoSQL Database	Oracle Database
File Parallel Processing	***Key-Value Database***	***Relational Database***
No inherent structure	***Simple data structure***	*Complex data structures, rich SQL*
High volume writes	***High volume random reads and writes***	*High volume OLTP with 2-PC*
Limited functionality, roll-your-own applications	*Simple get/put high speed storage, flex configuration*	*Security, Backup/Restore, Data life cycle mgmt, XML, etc.*
Batch Oriented	*Real-Time, web-scale specialized applications*	*General purpose SQL platform, multiple applications, ODBC, JDBC*

If we look at the various storage options available to handle Big Data – there are essentially three types: Hadoop, NoSQL Databases, Relational Databases

HDFS is a great distributed file system. Parallel, highly scalable and no inherent structure. However, it's tuned primarily for bulk sequential read/write of file blocks. There are no indices for fast access to specific data records, it's not well suited for lots of small files or updating files that have already been written. Primarily a batch system, write lots of data, then read it all in parallel over and over. Sounds like a datawarehouse, but more unstructured.

The Relational Database on the other hand, is usually deployed on a big machine, and supports complex data structures stored in tables with plenty of relationships. Data is manipulated and accessed using rich SQL to build mission critical applications. There is support for variety of data access protocols like ODBC/JDBC along with an elaborate life cycle management infrastructure involving security and backup/restore operations. Enterprises run their mission critical transaction processing systems on relational databases.

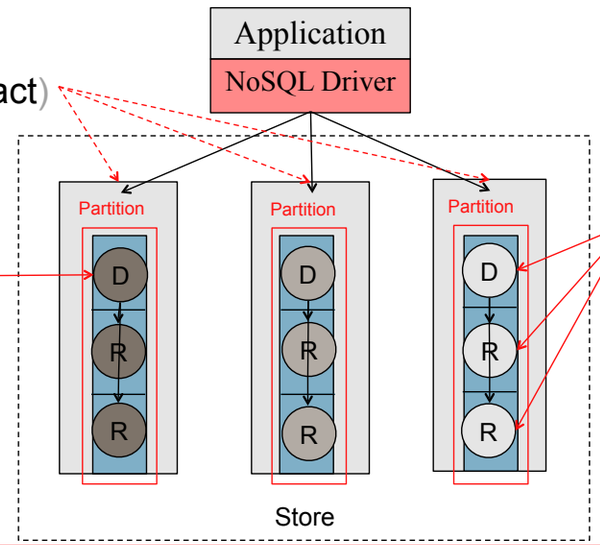
NoSQL database is the middle ground: a distributed key-value database with a simple data structure. It has indices. It can handle large volumes of data and is usually deployed on a distributed architecture consisting of several small machines. **It's designed for low latency high volume reads and writes of simple data, that is typical with real-time and web-scale specialized applications.** It's not tuned for reading/writing huge files – use a file system for that. It has flex configuration capabilities that make it very suitable to rapid application development requirements. **Data scalability at low cost.**

Agenda

- The Oracle NoSQL Database
- NoSQL background
- NoSQL at Oracle
- Developing with Oracle NoSQL Database
  - Architecture & Features
  - Deploying a cluster
  - Developing an application
- Use Cases

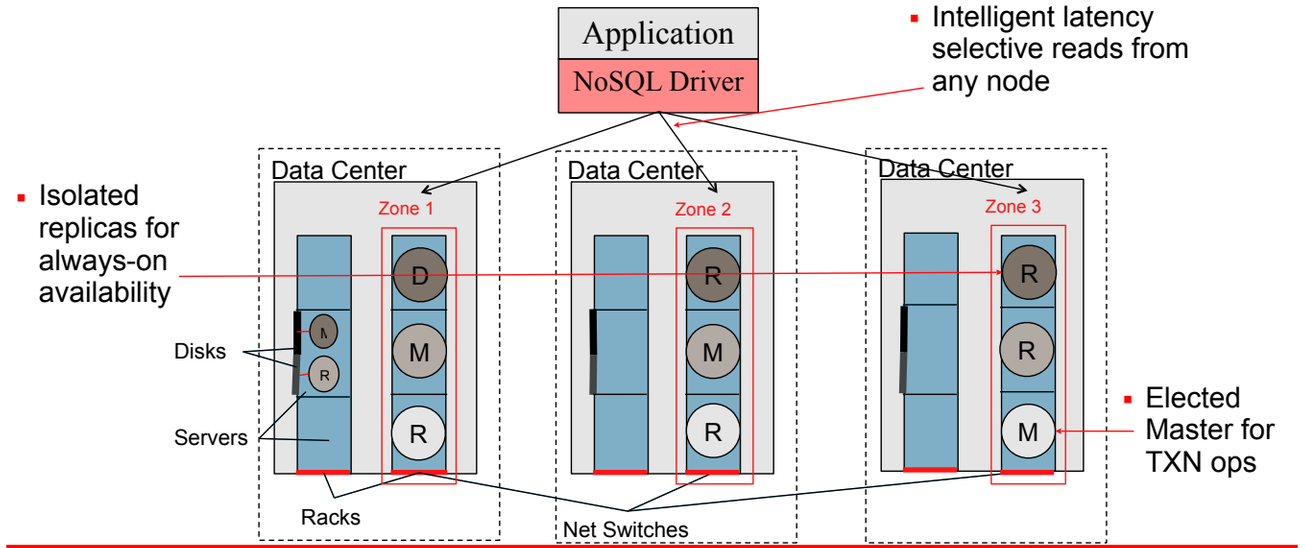
Logical Architecture – Applications View

- Elastic partitions (split, add, contract)
- Writes to elected node



- Reads from any node in system

Physical Architecture

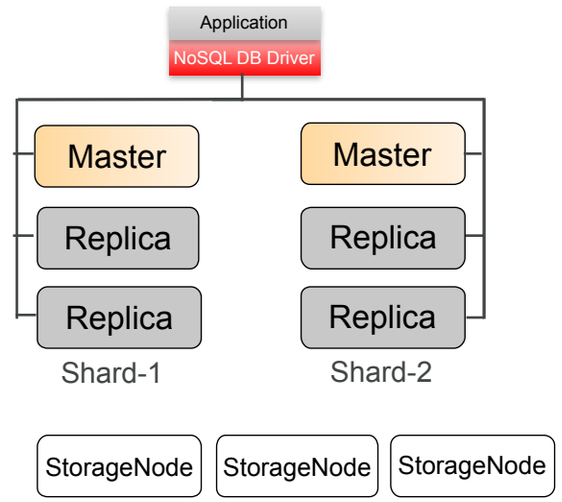


Elasticity

On-Demand Cluster Expansion

On Demand

- Increase Data Capacity
  - Add more storage nodes
  - New shards automatically created
- Increase Data Throughput
  - More shards = better write throughput
  - More replicas/shard = better read throughput



ORACLE

Elasticity refers to dynamic/online expansion changes in a deployed store configuration. New storage nodes are added to a store to increase performance, reliability, or both.

Increase Data Capacity - A Company's Oracle NoSQL Database application is now obtaining it's data from several unplanned new sources. The utilization of the existing configuration as more than adequate to meet requirements, with one exception, they anticipate running out of disk space later this year.

The company would like to add the needed disks to the existing servers in existing slots, establish mount points, ask NoSQL Database to fully utilize the new disks along with the disks already in place while the system is up and running Oracle NoSQL Database.

The Administrator after installing the new disks, defines a new topology using the Administrator with the new mount points and capacity value such that new replication nodes can be created on the existing storage nodes. The administrator can review the plan for errors and then when ready the new topology is deployed while the Oracle NoSQL Database is online and continues to serve the running application with CRUD operations.

Increase Throughput - As a result of an unplanned corporate merger, the live Oracle NoSQL Database will see a substantial increase in write operations. The read write mix of transactions will go from 50/50 to 85/15. The need workload will exceeds the I/O capacity available of the available storage nodes.

The company would like to add new hardware and have it be utilized by the existing Oracle NoSQL Database (kvstore) currently in place. Oh, and of course the Application needs to continue to be available while this upgrade is occurring.

With the new elasticity capabilities and topology planning, the administrator can add the new hardware and define a new topology with the new Storage Nodes. The administrator can then look at the resulting topology (storage nodes, replication nodes, shards, etc) to confirm it meets their requirements. Once they are satisfied with the new topolgy they can also determine when they want to deploy the new topology in the background and while the existing application continues to operate. As partitions/chunks of data are moved they are made available to the live system.

Increase Replication Factor- A new requirement has been placed on an existing Oracle NoSQL Database to increase the overall availability of the Oracle NoSQL Database by increasing the replication factor by utilizing new storage nodes added in a second geographic location. This is accomplished by adding at least 1 replication node for every existing shard. The current configuration has a replication factor of 3.

While the system is live, the administrator changes the topology to define the new storage nodes and define the replication factor. Again the administrator can validate the topology and review it before deploying. As a side point, the administrator could validate several changes to evaluate alternatives and then decide which topology to deploy. Just like the other scenarios described the data is automatically moved and partitions are made available as they are moved as part of a background activity. Meanwhile the KVStore continues to service the existing workload starting to use the new replicas as they become available. Once the topology is deployed a new replication node has been created and populated for each shard.

We have increased availability by increasing the replication factor where the new storage nodes are in another geographic location.

We have increased read throughput capability with the new Replication nodes for each shard and the Replication Factor is now 4.

Elasticity

Add more hardware and tell store its capacity

Change logical topology

Spread writes and/or give more places to read

Review topology plan

Deploy the plan to create the topology on the hardware

Issue Rebalance Command

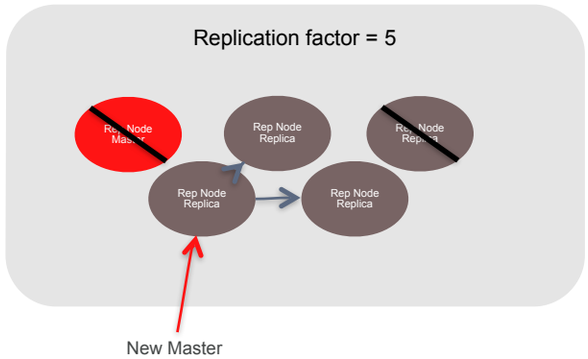
Physical data partitions move to new processes

Does it smart for reliability and low latency

Features - Failover

Automatic Failover

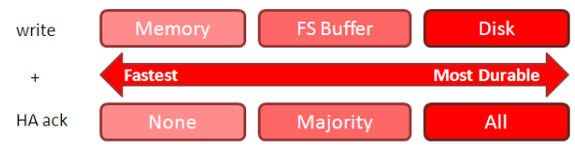
- Automatic election of new Master
- Rejoining nodes automatically synchronize with the Master
- Isolated nodes can still service reads
- All nodes are symmetric

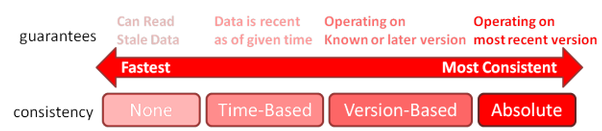


Features – Configurable ACID Transactions

Greater Flexibility

- Configurable Durability per operation
- Configurable Consistency per operation
- ACID by default
- Transaction scope is single API call
- Records share same major key
- Multiple operations supported





You can think of a transaction as a single auto-commit API call. That API call can be for a single record, multiple records or multiple operations AS LONG AS all of the records are for the same Major Key. However many records/operations are in that API call, they are all committed atomically (all or nothing). Because they all share the same Major Key, all of the data being affected resides on a single storage node, so we can guarantee the transactional semantics of the transaction commit. We will replicate that transaction to the replicas (copies of the data) as part of the transaction.

Of course, not all operations are created equal. In some cases you may want operations that are not completely ACID. One of the benefits of NoSQL is that it relaxes transactional guarantees in order to provide faster throughput. The Oracle NoSQL Database allows you to override the default and relax the ACID properties on a per-operation basis, allowing the application to specify the transactional behavior that is most appropriate.

Oracle NoSQL Database allows you to relax/configure the **C**onsistency and **D**urability policies for a given operation.

Durability is controlled by defining the Write Policy and the HA Acknowledgement Policy. You can increase write transactions performance by relaxing the Durability constraints. The default is Write-to-memory, Majority Ack.

Consistency is controlled by defining the Read Guarantees that you require from the system. You can increase read transaction performance by relaxing the Consistency constraints. The default is None.

Transactions & Consistency

Set at store or operation level
BASE and ACID
Write – Server Memory or OS Memory or DISK
Do this for user specified number of replicas
Master Only, Majority Quorum, ALL
Read – Any or Quorum or Master
Do this for user specified consistency

Agenda

- The Oracle NoSQL Database
- NoSQL background
- NoSQL at Oracle
- Developing with Oracle NoSQL Database
  - Architecture & Features
  - Deploying a cluster
  - Developing an application
- Use Cases

Deploying a Cluster

Hardware declaration

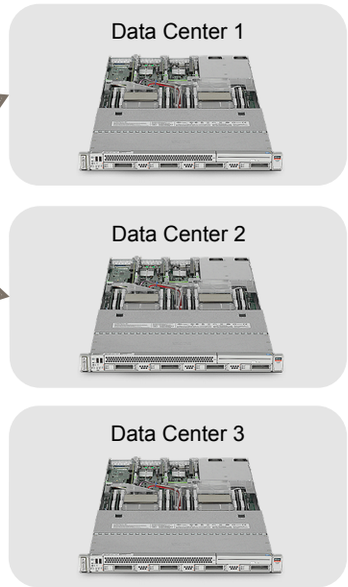
java -jar %KVHOME%\lib\kvstore.jar makebootconfig
-root \dataFilesDir
-port 5000 ← listener
-admin 5001
-host 10.256.128.64
-harange 5010,5020
-capacity 9 ← usually #disks on machine
-num_cpus 4
-memory_mb 1000

each server

# of potential processes on the server

start the listener

nohup java –jar \lib\kvstore.jar start -root \dataFilesDir

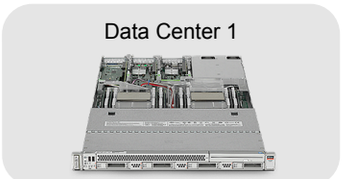


Deploying a Cluster

Start Administration Utility

java -jar \lib\kvstore.jar runadmin -host 10.256.128.64 -port 5000

- Lets you plan a deployment topology
- Lets you verify adequate resources and reliable process distribution
- Lets you deploy a validated plan
- Accepts scripts to automate a series of steps



Deploying a Cluster

Define and Create a Store

configure -name ONDB

plan deploy-**datacenter** -name "dataCenter1" -rf 1

plan deploy-**sn** -dcname dataCenter1 -host 10.256.128.64 -port 5000 -wait

plan deploy-**admin** -sn sn1 -port 5001 -wait

……plan deploy-datacenter -name "dataCenter2" -rf 1

……plan deploy-datacenter -name "dataCenter3" -rf 1

topology create -name MyTopology *-pool AllStorageNodes* -partitions 900

topology validate -name MyTopology

Copies of Data in DC

One SN for Each RF

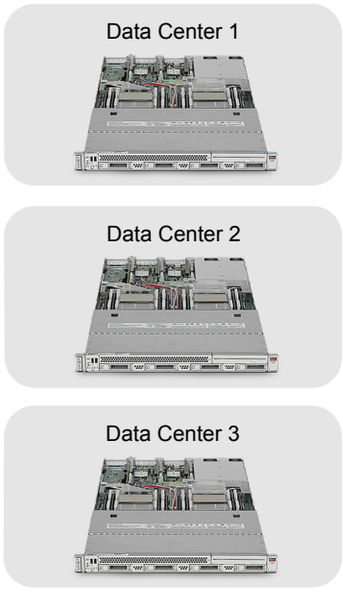
MUST do 1 of these after first SN

Repeat for more SN's and Each DC

Create and Validate

plan deploy-topology -name MyTopology

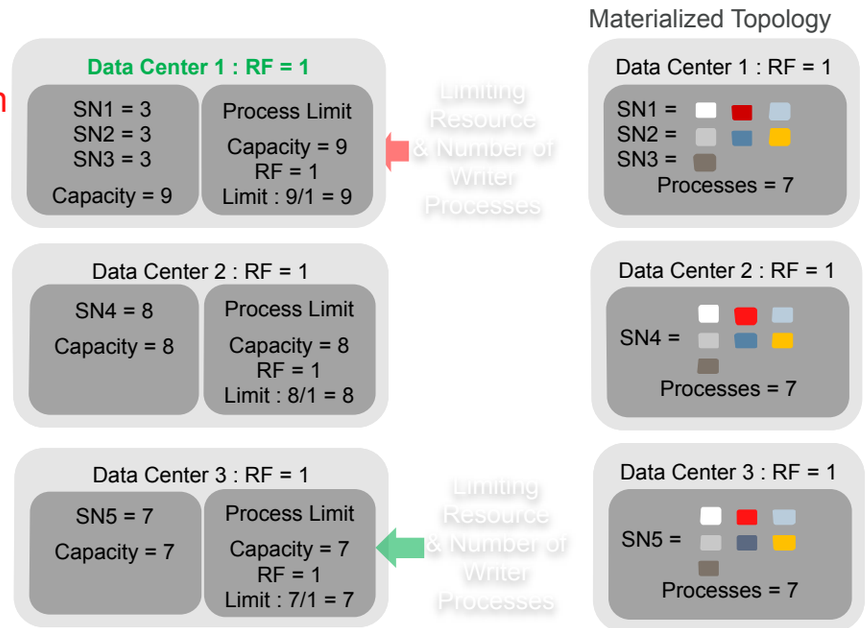
Deploy the Topology

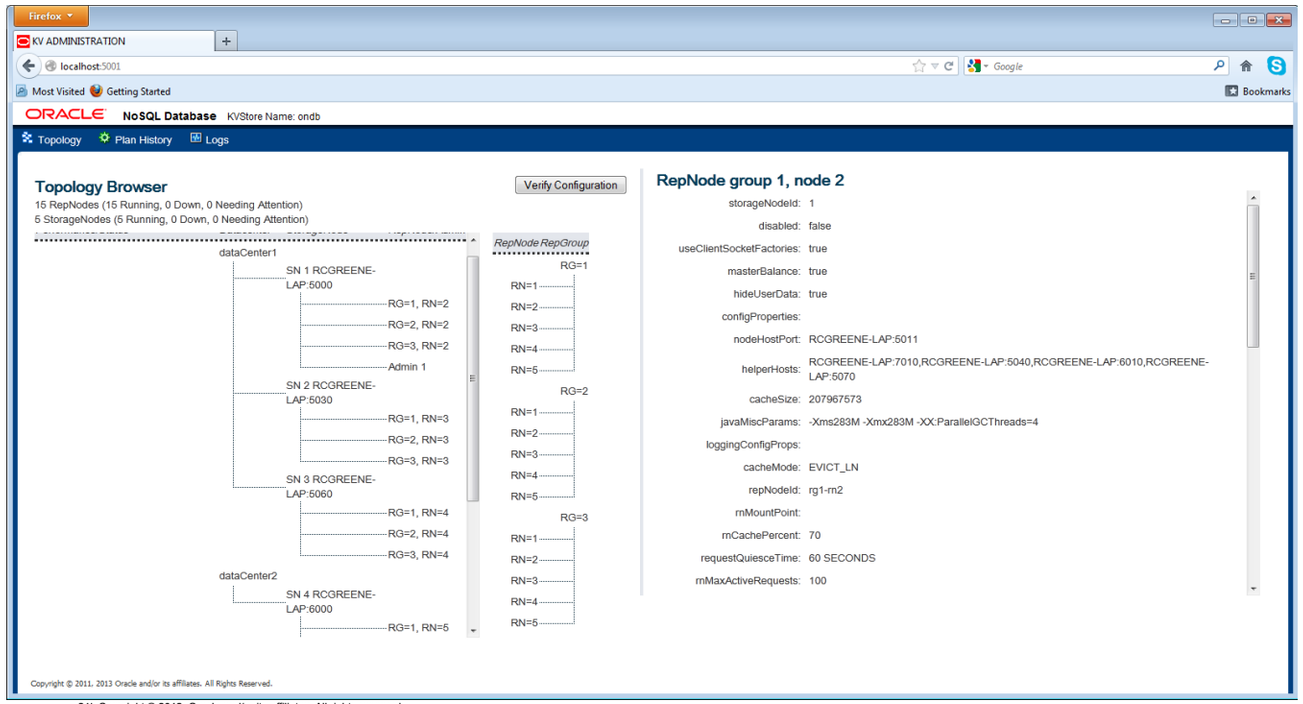


Topology

Resource Allocation

- DC Capacity
  - Sum SN's
- DC Copies
  - RF for DC
- DC Process Limit
  - Capacity/RF

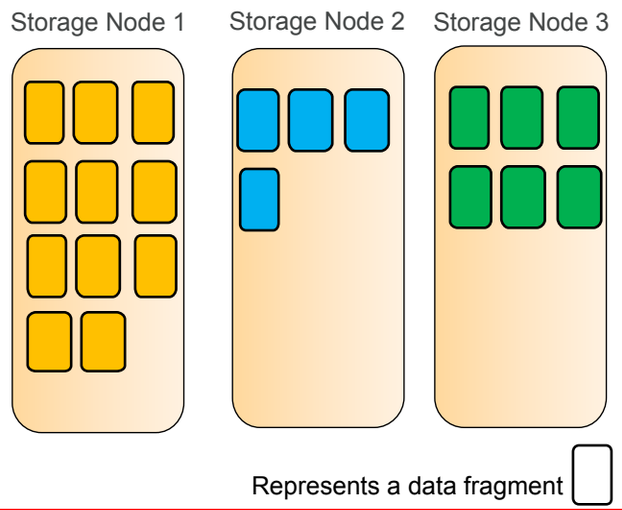




Rebalance an Unbalanced Store

Improve Performance

- Supports heterogeneous storage topology
- Replicas move from over-utilized to under-utilized storage nodes
- Number of shards and replication factor remain unchanged



ORACLE

Rebalance a configuration :

A storage node has failed and must be replaced (KVStore continues to run). The new hardware is a much more powerful machine (9 Cores, 64 GB of real (compared to 8 GB), multiple 400 GB Solid State Drives). The hardware is a heterogenous hardware mix. The new hardware replaces the failed storage node and the System administrator add the new Storage node to the pool of available storage modes and then migrates the old (failed) Storage node to the new one. After successful migration (KVStore continues to run) the failed storage node is deleted and all Storage nodes are active again.

Continuing to monitor the performance of the system and the existing topology, the administrator notices that some of the older storage nodes have 2 replication nodes on them and the CPU/IO utilization is high and latency is high as well, while the new much faster storage node is under utilized.

By using the new physical topology planning support available in this release, Oracle NoSQL Database will rebalance the configuration and redistribute the data . In other words, Oracle NoSQL Database will make optimal use of heterogeneous storage nodes.
The new Storage nodes will likely have multiple replication nodes running on them while many of the older systems may go from 2 to 1. The replication nodes will automatically be moved.
Again this can all happen while the system is online and at the convenience of the company.

By using the new physical topology planning support available in this release, Oracle NoSQL Database will rebalance the configuration and redistribute the data . In other words, Oracle NoSQL Database will make optimal use of heterogeneous storage nodes.
The new Storage nodes will likely have multiple replication nodes running on them while many of the older systems may go from 2 to 1. The replication nodes will automatically be moved.
Again this can all happen while the system is online and at the convenience of the company.

Data Movement:

- **Idempotent:** Can be run multiple times with the same result
- **Interruptible**: You can interrupt at any time and the KVStore will continue running. The company may have a peak workload period daily and may want to interrupt the data movement (as part of the new topology) and restart it after the peak period.
- **Restartable**:

Agenda

- The Oracle NoSQL Database
- NoSQL background
- NoSQL at Oracle
- Developing with Oracle NoSQL Database
  - Architecture & Features
  - Deploying a cluster
  - Developing an application
- Use Cases

Developing Applications

Getting Connected

- KVStore kvstore = KVStoreFactory.getStore(kconfig);
  - Use kvstore handle to database for CRUD operations
    - e.g. kvstore.put(“key”, “value”);

- KVStoreConfig(“*store name*", hosts[ ]);
  - String[] hosts = {"n1.example.org:5088", "n2.example.org:4129"};

KVStoreConfig settings *[Name, Hosts, Durability, Consistency, Timeout]*

Developing Applications

CRUD Operations

```
 /**
  * Perform PUT and Get operation.
  */
 void putExample( String sector, String id, int reading ) {

     final String sensorKey = "PS" + sector + id;
     final byte[] measure = ByteBuffer.allocate(4).putInt(reading).array();

     final Key k = Key.createKey(sensorKey);
     final Value v = Value.createValue(measure);

     store.put(k,v);

     final ValueVersion valueVersion = store.get(Key.createKey(sensorKey));

     System.out.println(sensorKey + " " + new String(valueVersion.getValue()
 }
```

ORACLE

Developing Applications

CRUD Operations – Reading

- get( ) – a single value beneath a key
- multiGet( )
  - Result should fit in memory
  - multiple values beneath a complete major key
- multiGetIterator( )
  - If the results may not fit in memory
  - Can specify a range
- storeIterator( )
  - Read via partial key components
- keyRange( ) – to filter multi-get operations

Exceptions:

- ConsistencyException
- RequestTimeoutException
- FaultException

Consistency.ABSOLUTE
Consistency.NONE_REQUIRED

Consistency.TIME …..need NTP to sync clocks
Consistency.VERSION

Developing Applications

CRUD Operations – Writing

- put( ) – a single value beneath a key
- putIfAbsent( )
  - Only if the key ***does not*** already exist
  - Returns null if already exists
- putIfPresent( )
  - Only if the key ***does*** exist
  - Returns null if does not exist
- putIfVersion( )
  - Only if the version of the value has not changed since you read it
- delete( ) , deleteIfVersion( ), multiDelete( )

Exceptions:

- DurabilityException
- RequestTimeoutException
- FaultException

Durability.ReplicaAckPolicy.
ALL
NONE
SIMPLE_MAJORITY

Durability.SyncPolicy.
SYNC
WRITE_NO_SYNC
NO_SYNC

```
Durability defaultDurability =
new Durability(Durability.SyncPolicy.SYNC, // Master sync
Durability.SyncPolicy.NO_SYNC, // Replica sync
Durability.ReplicaAckPolicy.SIMPLE_MAJORITY);

kconfig.setDurability(defaultDurability);
```

Developing Apps

CRUD Operations

- Custom durability
- Versioning
- Exception handling
- Retry logic

```
/**
 * Perform Update operation.
 */
void updateExample( Key psKey, int newReading ) {

   final byte[] measure = ByteBuffer.allocate(4).putInt(newReading).array();
   final Value newValue = Value.createValue(measure);

   final Version matchVersion = store.get(psKey).getVersion();

   Durability durability =
         new Durability(Durability.SyncPolicy.SYNC, // Master sync
         Durability.SyncPolicy.NO_SYNC, // Replica sync
         Durability.ReplicaAckPolicy.SIMPLE_MAJORITY);

   ReturnValueVersion prevValue = new ReturnValueVersion(
                                 ReturnValueVersion.Choice.VERSION);

   try{
         store.putIfVersion( psKey, newValue, matchVersion, prevValue,
                           durability, 1000, TimeUnit.MILLISECONDS);
   }catch( DurabilityException de ){
      //Lets Discuss Durability  ......
   }catch( RequestTimeoutException re ){
      //Lets Discuss Timeout
   }catch( FaultException fe ){
      //Lets Discuss Faults
   }
}
```


Developing Applications

Other stuff

- AVRO / JSON
  - A highly efficient serialization format
- Sequences
  - Combine many operations into a single transaction
- LOBs
  - A streaming API interface for large objects
  - Does split/merge operations for highly parallelized access

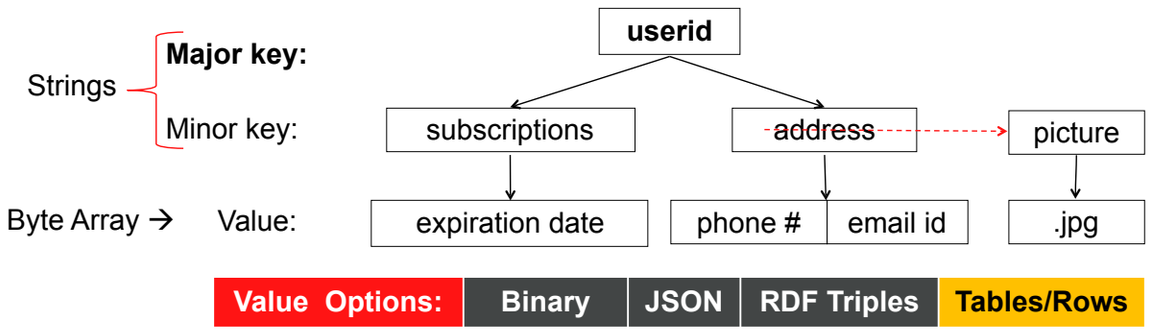
How to Approach Data Modeling

- Use functional requirements to drive critical major key modeling and ACID semantics
- Think of major key as the master in a master detail type of relationship
- Query access patterns are crucial in deciding how to model the key space (CRUD, range, ACID)
- Know the APIs – Knowing the functionality exposed through the Oracle NoSQL Database APIs will be invaluable in helping to successfully model your application
- Don’t think that values must be utilized just because it is a key/value store

Developing Applications

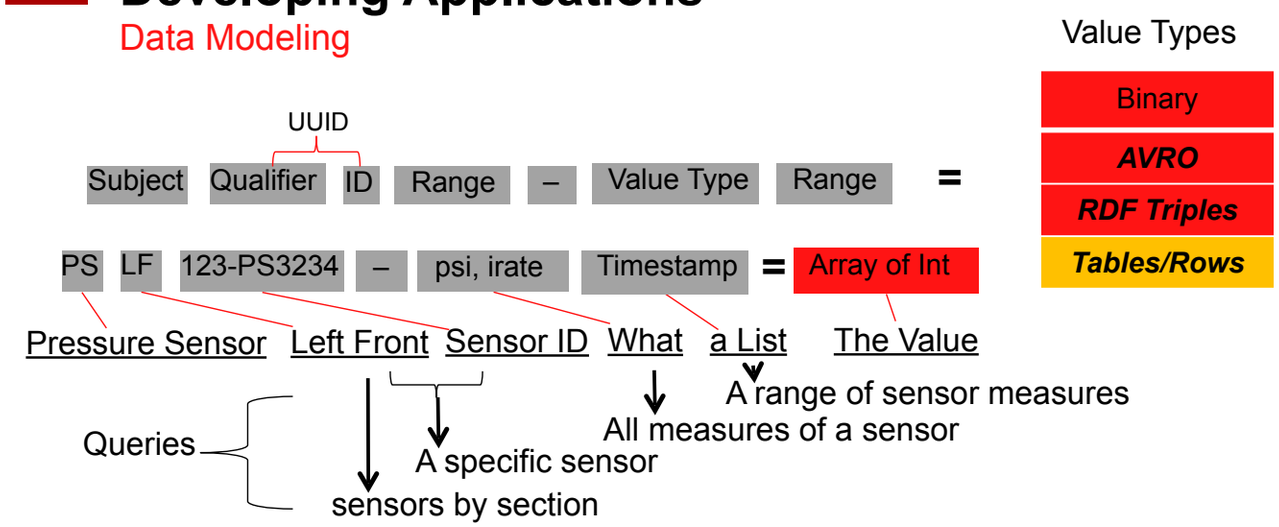
Data Modeling

- Major-Minor (hash – local)
- Keep small (memory), align with queries



Developing Applications

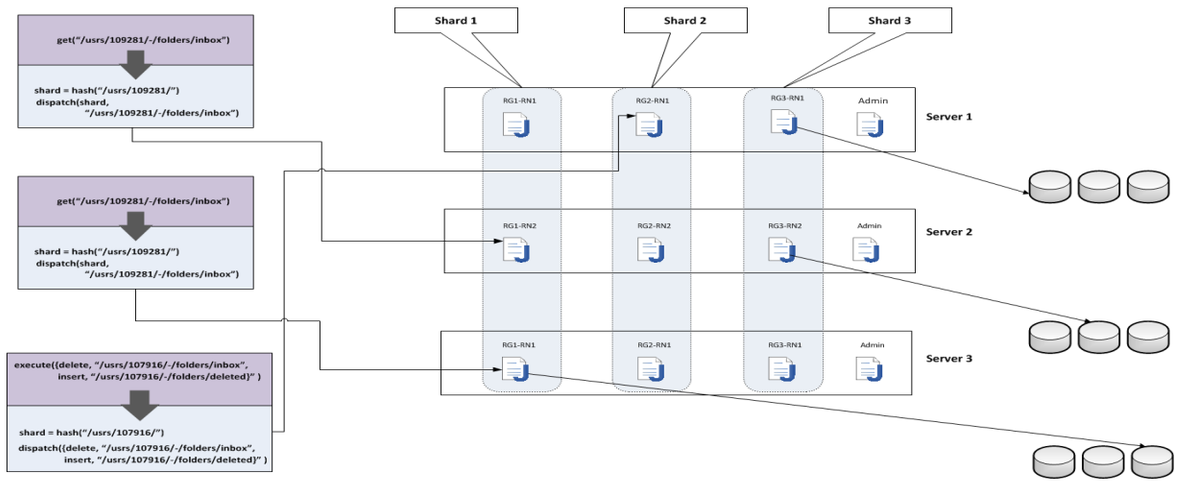
Data Modeling



Basic Key Structure

- Key composed of two pieces – Major and minor
  - /maj1/maj2/-/min1/min2/
  - /users/109281/-/profile
  - /users/109281/-/address
- MD5 Hash(major component) – Maps major key to shard
- Shard defines the scope of the transaction
- Full key path is unique – Utilized in B-tree scans
- Intermediate B-tree nodes kept in cache – Compress keys as much as possible

Major Key Mapping

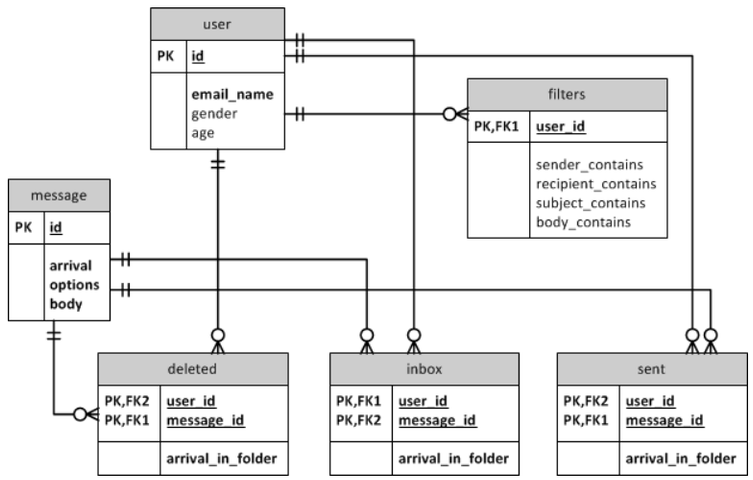


Example Application – Email at Scale

Functional Requirements

- User's must have the following folders available
  - Inbox – Stores messages that have been read, not deleted, and not moved to other folders
  - Sent folder - Stores messages that have been sent
  - Deleted – Stores messages that haven been deleted from other folders
    - Messages must be permanently deleted from the deleted folder after 90 days from arrival in the deleted folder
  - Calendar – Stores meetings, appointments, etc.
- When a user deletes a message and refreshes the browser, the message must show up under the deleted folder.
- Message displayed in each folder must (by default) be sorted by the time of arrival to the email system.
- Calendar event editing must be atomic, consistent , isolated, and durable.

Email Example – RDBMS Schema



Translating to Oracle NoSQL Key Spaces

- Try to model everything in a simple hierarchy
- Parent-child tables as hierarchical key paths
- Parent entities (master data) becomes major key component
  - First think about ACID semantics
  - Next think about key distribution and sufficient cardinality for better distribution
- Add secondary keys when necessary
  - Modeled as separate keys whose value contains a pointer to the primary key

Email Example - Key Space Model

- /usrs/id/-/folders/inbox/message_date/message_id
  - /usrs/791096/-/inbox/20130923/789765
- /usrs/id/-/folders/del/message_date/message_id
  - /usrs/791096/-/del/20130923/893671
- /usrs/id/-/folders/sent/message_date/message_id
  - /usrs/791096/-/sent/20130923/787361
- /usrs/id/-/folders/cal/event_date/event_id
  - /usrs/791096/-/cal/20130923/534891
- /usrs/id/-/folders/filters/filter_id
  - /usrs/791096/-/filters/20130923/898517

Email Example - Key Space Modeling Rationale

- Scale – User messages will be distributed throughout every shard of the Oracle NoSQL Database Cluster
- ACID – Multiple operations against one or more user's folders will be atomic, consistent, isolated, and durable
  - Atomic – Deleting a user's message = Remove from inbox, insert into deleted folder
  - Consistent – Refreshing the browser after a user deletes a message must render the message in the deleted folder
  - Isolated – Threads updating arriving calendar events on behalf of a user must be isolated from that user's edits to the same calendar time slots
- Sorting – Storing dates in the form YYYYMMDD as a component in the path will be crucial to ordering our results

Email Example – API Drivers of Model

- execute(List<Operation> operations)
  - Move message from one folder to another
    - Atomic delete followed by an insert
- multidelete(Key parentKey, KeyRange subRange, Depth depth)
  - Empty the deleted folder
    - Single network roundtrip to delete many messages
- multiGetIterator(Direction direction, int batchSize, Key parentkey, KeyRange subRange, Depth depth)
  - Retrieve messages from a folder ordered by date
- multiGet(Key parentkey, KeyRange subRange, Depth depth)
  - Retrieve ordered events from calendar, fully materialized

Email Example – Application Queries to API Calls

- Get the first 25 messages in the inbox for user 109876 for the last week and order descending
  - multiGetIterator(Direction.REVERSE, 25, “/usr/109876/-/folders/inbox”, {20130923, 20130916}, Depth.DESCENDENTS_ONLY)
- Move message 78711 for user 109876 from the inbox to the deleted folder
  - execute({Operation.TYPE.DELETE, “/usr/109876/-/folders/inbox/20120923/78711”, Operation.TYPE.PUT, ““/usr/109876/-/folders/deleted/20120923/78711”})
- Empty the deleted folder for user 109876
  - multiDelete(, “/usr/109876/-/folders/deleted”, null, null)
- Find all calendar events for this month for user 109876
  - multiGet(“/usr/109876/-/folders/cal”, {20130901, 20130930}, Depth.DESCENDENTS_ONLY)

Oracle NoSQL DB Resources

- Oracle Big Data Handbook (Amazon, Barnes & Noble, Oracle Press)
- NoSQL DB Use Cases, White Papers, Data Sheets, Benchmarks

  http://www.oracle.com/technetwork/products/nosqldb/overview/index.html
- NoSQL DB Documentation

  http://www.oracle.com/technetwork/products/nosqldb/documentation/index.html
- NoSQL DB Downloads

  http://www.oracle.com/technetwork/products/nosqldb/downloads/index.html
- NoSQL DB OTN Forum

  http://forums.oracle.com/forums/forum.jspa?forumID=1388
- OU Training Classes

  http://bit.ly/V5qbmY

Hardware and Software

ORACLE

Engineered to Work Together