Feature Overview

Development Flexibility

- Time to live - Auto-aging of data
  - Tables level defaults
  - Override at the record level if desired

```
create table profile(id INTEGER, firstName STRING,
                     lastName STRING, interests ARRAY(STRING),
                     contactDetails JSON,
                     primary key(id))
USING TTL 3 days
```

- Access via API or SQL
  - Get, put, scan APIs for raw key/value pairs and tables
  - SQL supports complex filtering expressions
  - SQL supports functions in filter expressions
  - SQL interoperability between schema-less and fixed schema

Sara Lipowsky
DATABASE SPECIALIST
ORACLE

Feature Overview

Development Flexibility

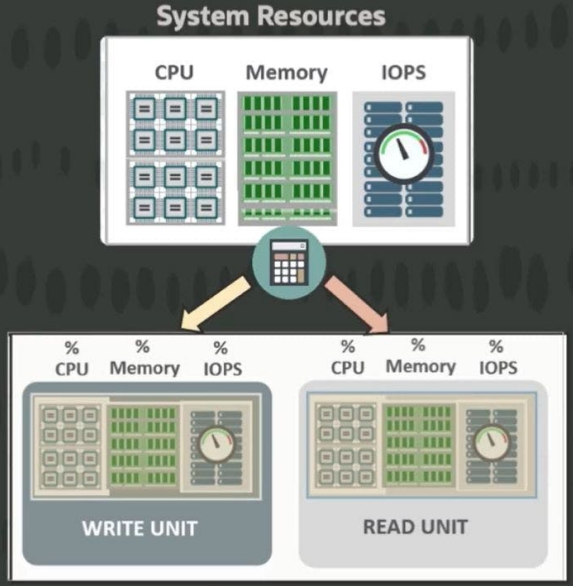
- Time to live - Auto-aging of data
  - Tables level defaults
  - Override at the record level if desired

```
create table profile(id INTEGER, firstName STRING,
                     lastName STRING, interests ARRAY(STRING),
                     contactDetails JSON,
                     primary key(id))
USING TTL 3 days
```

- Access via API or SQL
  - Get, put, scan APIs for raw key/value pairs and tables
  - SQL for rich access to JSON, more complex filtering expressions
  - Support for conjunctions and disjunctions in filter expressions
  - SQL interoperability between schema-less and fixed schema

Throughput Capacity

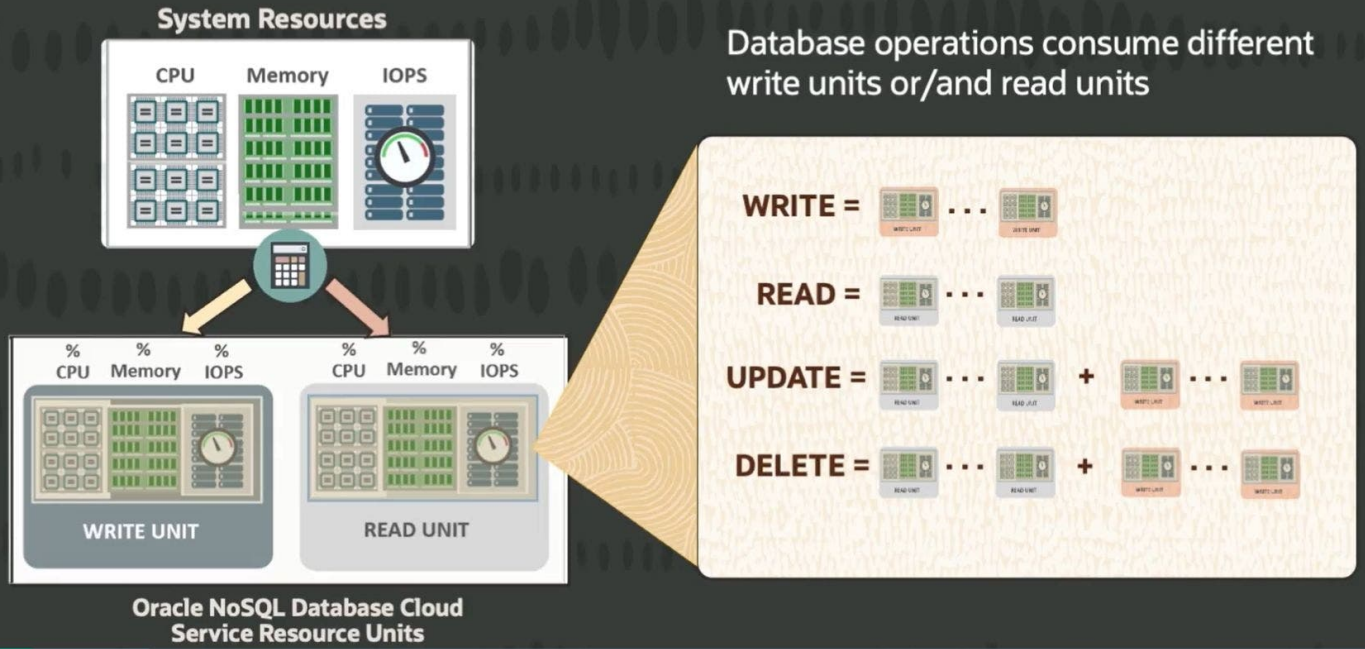
Resource usage and capacity



- Resource usage and capacity measured in write and read units
- System resources usage are transparent to users
- Write and read units are used for capacity planning and scaling
- Provision capacity based on application workloads, query pattern, record size, data model
- Simple, granular, cost effective

Write Unit, Read Unit

Database operation and throughput consumption



Oracle NoSQL Database Cloud Service

Provisioned throughput

Provision reads/sec, writes/sec, GB storage at table creation time

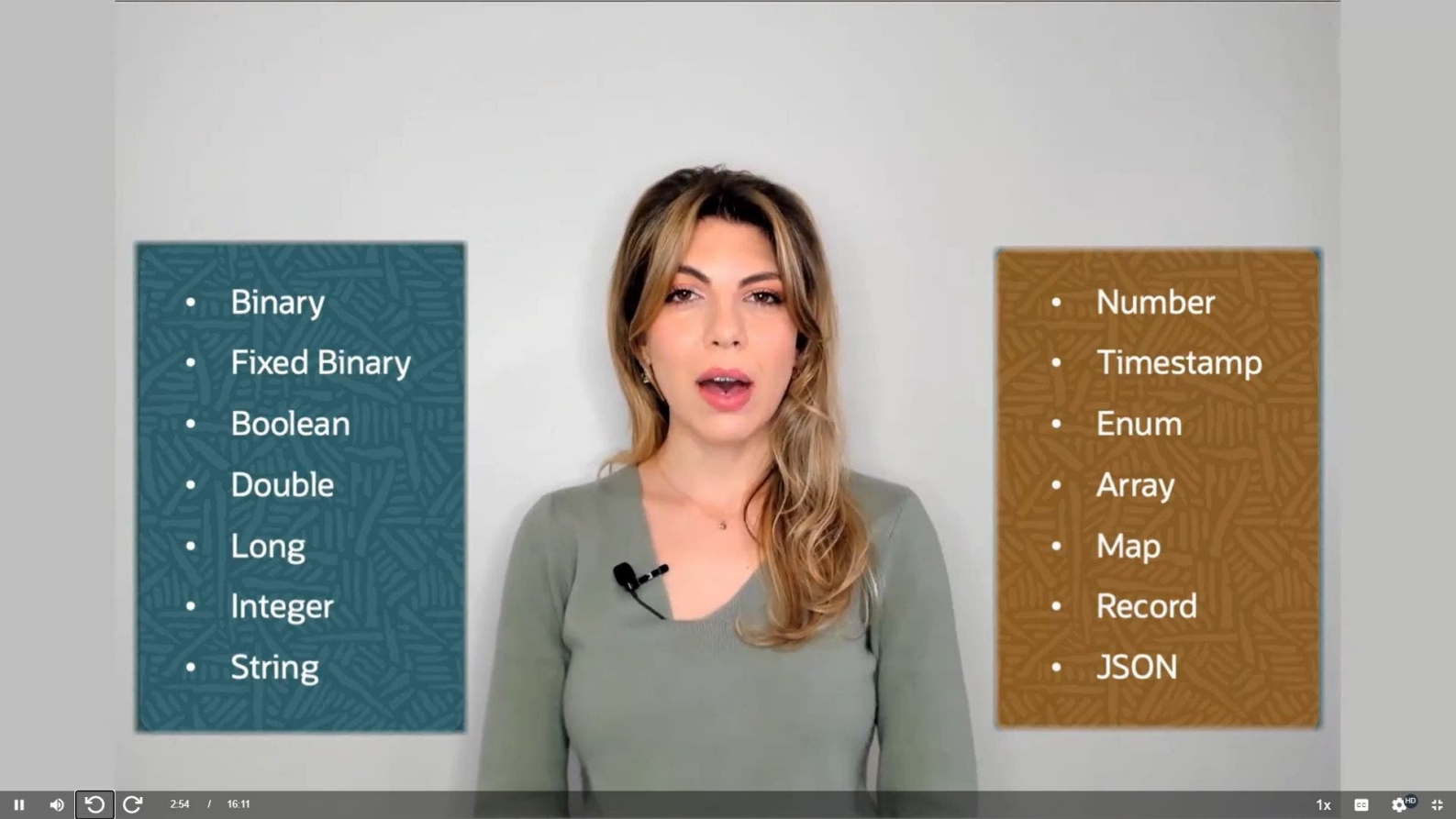
- Dynamically increase
- Dynamically decrease


```
TableRequest tableRequest = new TableRequest()
        .setStatement("create table if not exists foo (id integer,
value JSON)")
        .setTableLimits(new TableLimits(2000, 100, 500))
        .setTimeout(1000);
TableResult res = NoSQLHandle.tableRequest(tableRequest);
```

Modify the table lowering the read units to 1000

```
tableRequest.setTableLimits(new TableLimits(1000, 100, 500))
```

Note: Every TableRequest is a DDL call to the NoSQL store and may be performed 4 times within a minute.


Binary
Fixed Binary
Boolean
Double
Long
Integer
String
Number
Timestamp
Enum
Array
Map
Record
JSON
2:54 / 16:11
1x

Oracle NoSQL Database Cloud Service

Service connectivity and resources types

Connection Credentials

- Tenancy ID (an OCI ID)
- User ID (an OCI ID)
- API Signing Key
- Signing Key Fingerprint
- Signing Key Pass Phrase (optional)

Resource Types

- nosql-tables
- nosql-row
- nosql-indexes

Note: Every OCI ID consists of ocid1.<RESOURCE TYPE>.<REALM>.[REGION][.FUTURE USE].<UNIQUE ID>

Oracle NoSQL Database Cloud Service

Permissions for NoSQL-Tables

Verb	Permissions	REST APIs Fully Covered	NoSQL Cloud Driver Request Covered
INSPECT	NOSQL_TABLE_INSPECT	ListTables	ListTableRequest
READ	INSPECT + NOSQL_TABLE_READ	GetTable	GetTableRequest
		ListWorkRequests GetWorkRequest ListWorkRequestErrors ListWorkRequestLogs	None
		ListTableUsage	TableUsageRequest
USE	READ + NOSQL_TABLE_ALTER	UpdateTable DeleteWorkRequest	TableRequest • change TableLimits • ALTER TABLE
MANAGE	USE + NOSQL_TABLE_CREATE	CreateTable	TableRequest (CREATE TABLE)
	NOSQL_TABLE_DROP	DeleteTable	TableRequest (DROP TABLE)
	NOSQL_TABLE_MOVE	ChangeTableCompartment	Not supported

Oracle NoSQL Database Cloud Service

Permissions for NoSQL-Rows

Verb	Permissions	REST APIs Fully Covered	NoSQL Cloud Driver Request Covered
INSPECT	None	None	None
READ	NOSQL_ROWS_READ	GetRow Query (SELECT) PrepareStatement SummarizeStatement	• GetRequest • PrepareRequest • QueryRequest (SELECT)
USE	READ + NOSQL_ROWS_INSERT	UpdateRow Query (INSERT/UPSERT, UPDATE)	• PutRequest • WriteMultipleRequest(Put) • QueryRequest(INSERT/UPSERT, UPDATE)
MANAGE	USE + NOSQL_ROWS_DELETE	DeleteRow Query (DELETE)	• DeleteRequest • MultiDeleteRequest • WriteMultipleRequest(Delete) • QueryRequest(DELETE)

Oracle NoSQL Database Cloud Service

Permissions for NoSQL-Indexes

Verb	Permissions	REST APIs Fully Covered	NoSQL Cloud Driver Request Covered
INSPECT	None	None	None
READ	NOSQL_INDEX_READ	ListIndexes	GetIndexesRequest + indexName
		GetIndex	GetIndexesRequest
USE	READ + NONE	ListIndexes	GetIndexesRequest + indexName
		GetIndex	GetIndexesRequest
MANAGE	READ + NOSQL_INDEX_CREATE	CreateIndex	TableRequest(CREATE INDEX)
	NOSQL_INDEX_DROP	DeleteIndex	TableRequest(DROP INDEX)

Oracle NoSQL Database Cloud Service

Permissions for NoSQL Cloud Driver Request

Request	Permission	Operation (Request.operation)
DeleteRequest	NOSQL_ROWS_DELETE	DeleteRow
GetIndexesRequest	NOSQL_INDEX_READ	GetIndex
GetRequest	NOSQL_ROWS_READ	GetRow
GetTableRequest	NOSQL_TABLE_READ	GetTable
ListTablesRequest	NOSQL_TABLE_INSPECT	ListTables
MultiDeleteRequest	NOSQL_ROWS_DELETE	DeleteRow
PrepareRequest	NOSQL_ROWS_READ	GetRow
PutRequest	NOSQL_ROWS_INSERT	UpdateRow
QueryRequest (SELECT)	NOSQL_ROWS_READ	GetRow
QueryRequest (INSERT, UPSERT, UPDATE)	NOSQL_ROWS_INSERT	UpdateRow
QueryRequest (DELETE)	NOSQL_ROWS_DELETE	DeleteRow
TableRequest (CREATE TABLE)	NOSQL_TABLE_CREATE	CreateTable
TableRequest (ALTER TABLE)	NOSQL_TABLE_ALTER	UpdateTable
TableRequest (DROP TABLE)	NOSQL_TABLE_DROP	DeleteTable
TableUsageRequest	NOSQL_TABLE_READ	GetTable
WriteMultipleRequest	has PutRequest: NOSQL_ROWS_INSERT has DeleteRequest: NOSQL_ROWS_DELETE	UpdateRow DeleteTable

Seamless multi-model

Seamless SQL interoperability with fixed schema

Schema-less

Data Definition

```
create table profile(cookieID STRING,
        content JSON, primary cookieID))
```

Question

Find all visitors to my site in November who are males between 24 and 30 years of age

Query

```
select cookieID from profile
where
   cast(content.lastVisit as TIMSTAMP) >=
        cast("2019-11-01" as TIMESTAMP) and
   cast(content.lastVisit as TIMESTAMP) <=
         cast("2019-11-30as TIMESTAMP) and
   content.demographic.gender = 'M' and
   content.demographic.age >= 24 and
   content.demographic.age <= 30
```

Identical SQL

Fixed Schema

Data Definition

```
create table profile(cookieID STRING,
     content RECORD(lastVisit TIMESTAMP,
               demographic RECORD(age INTEGER,
                     gender ENUM(M, F))),
     primary key(id))
```

Question

Find all visitors to my site in November who are males between 24 and 30 years of age

Query

```
select cookieID from profile
where
   cast(content.lastVisit as TIMSTAMP) >=
        cast("2019-11-01" as TIMESTAMP) and
   cast(content.lastVisit as TIMESTAMP) <=
         cast("2019-11-30as TIMESTAMP) and
   content.demographic.gender = 'M' and
   content.demographic.age >= 24 and
   content.demographic.age <= 30
```

Feature Overview

Rich Secondary Indexing

```
create table profile(id INTEGER, firstName STRING,
                     lastName STRING, interests ARRAY(STRING),
                     contactDetails JSON,
                     primary key(id))
```

```
{
  "billingAddress" : {
     "street" : "127 15th",
     "type" : "Avenue",
     "city" : San Francisco",
     "state" : "California",
     "zipcode" : 94116,
     "country" : "USA"
  },
  "shipTo" : {
     "street" : "127 15th",
     "type" : "Avenue",
     "city" : San Francisco",
     "state" : "California",
     "zipcode" : 94116,
     "country" : "USA"
  }
}
```

- Simple scalars

  ```
  create index myidx1 on profile(lastName)
  ```

- Non-scalars

  ```
  create index myidx1 on profile(interests)
  ```

- Composites

  ```
  create index myidx2 on profile(firstName, lastName)
  ```

- JSON

  ```
  create index myidx1 on profile(contactDetails.shipTo.zipcode as ANYATOMIC)
  ```

Feature Overview

Rich SQL Query Support

Predicates, projections, paging

```
select
  cookieID,
  content.demographic.values($key="income")
from
  profile
where
  starts_with(lower(content.demographic.gender),
              "f"),
order by content.demographic.age asc
limit 25
offset 26
```

- Projections
  - Simple scalars
  - JSON document fragments
- String functions
- Sorted results
- Paging

Group by, aggregates

```
select
   content.demographic.age,
   avg(content.demographic.income) as avgIncome
from
    profile
where
   year(cast (content.lastVisit as timeStamp)) >= 2019
group by
     age
order by avgIncome desc
```

- Simple aggregates
  - Min, max, avg, sum, count
- Time extraction functions
- Group by expressions

Feature Overview

Rich SQL Query Support

Document Upsert

```
update profile p
   set p.content.demographic.income =
       p.content.demographic.income  + 2000,
   add p.info.address.phones 0
     { "areacode":831, "number":5294368, "kind":"mobile" }
   remove p.info.address.phones[$element.kind = "office"]
   put p.info.children.Ron { "likes" : ["skiing"] }
where cookieID = "787cd009871"
```

- Set simple scalars
- Add new elements to arrays
- Remove elements of arrays
- Put new arrays into an existing document

Shard local joins & Regular Expressions

```
select
  profile.content.demographic.age,
  profile.content.demographic.gender
from
  NESTED TABLES(profile.shoppingCart ancestors(profile) on
                profile.cookieID = shoppingCart.cookieID)
where
  regex_like(shoppingCart.productDesc, *Mac*)
```

- Nested table clause with left outer join semantics
- Regular expressions

Feature Overview

Rich SQL Query Support

GeoJSON

```
select
   profile.cookieID
from
   profile
where
   profile.deomgraphic.gender = 'M' and
   geo_inside(profile.location,
         { "type" : "polygon",
          "coordinates" : [
            [ [-121.94, 36.28],
              [-117.52, 37.38],
               [-119.99, 39.00]
            ]
          ]})
```

- Points, line, and polygons
- Bounding box
- Intersection
- Within distance (near)

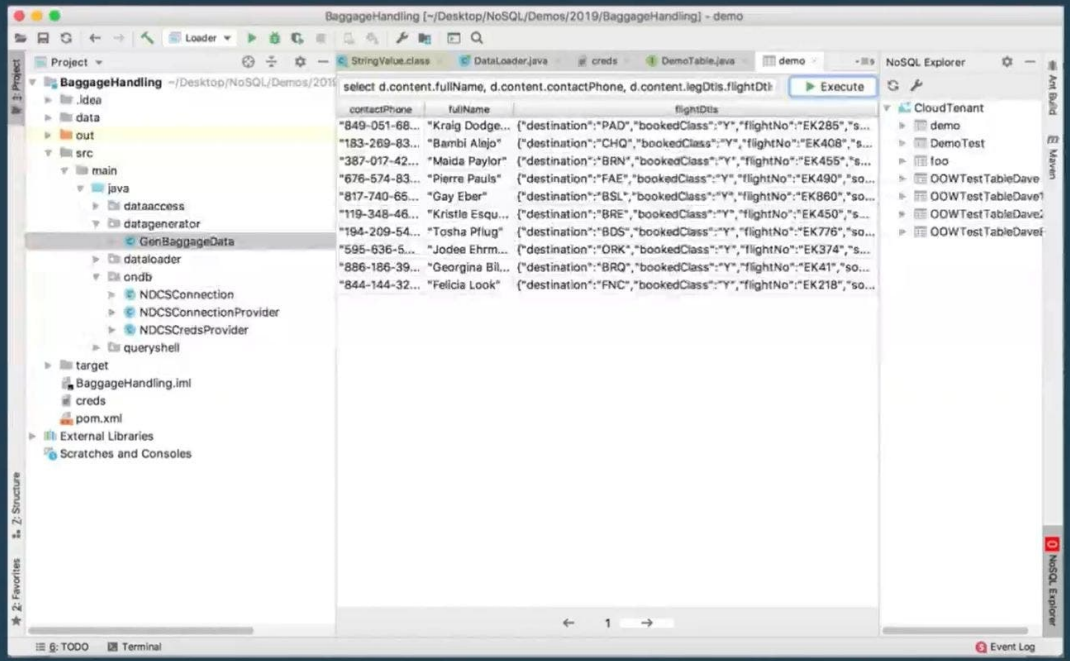
ID Generation

```
CREATE Table profile(id INTEGER GENERATED BY DEFAULT AS IDENTITY (
                                   START WITH 1
                                   INCREMENT BY 1
                                   MAXVALUE 100000000
                                   CACHE 1000
                    cookieID STRING,
                    content JSON,
    PRIMARY KEY(id))
```

- Identity column
  - Min/max values determine storage size
  - Optionally cache blocks of values on client

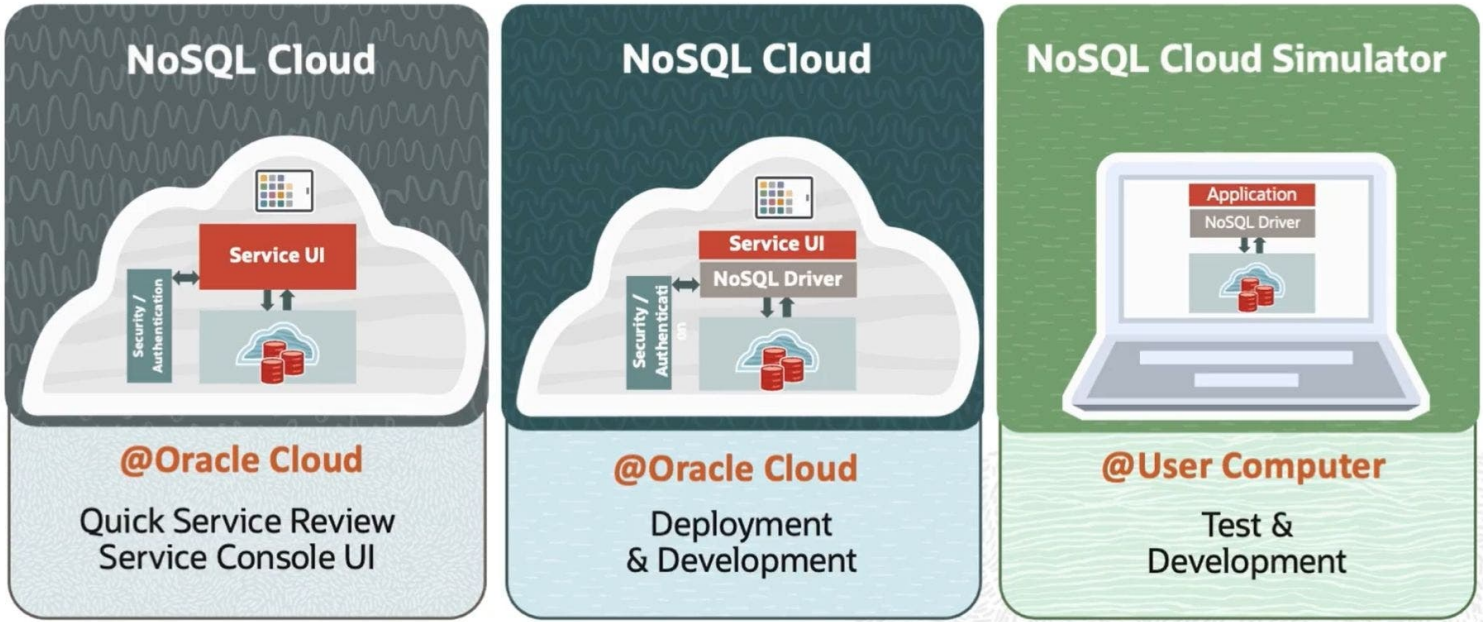
Feature Overview

IDE Plugins



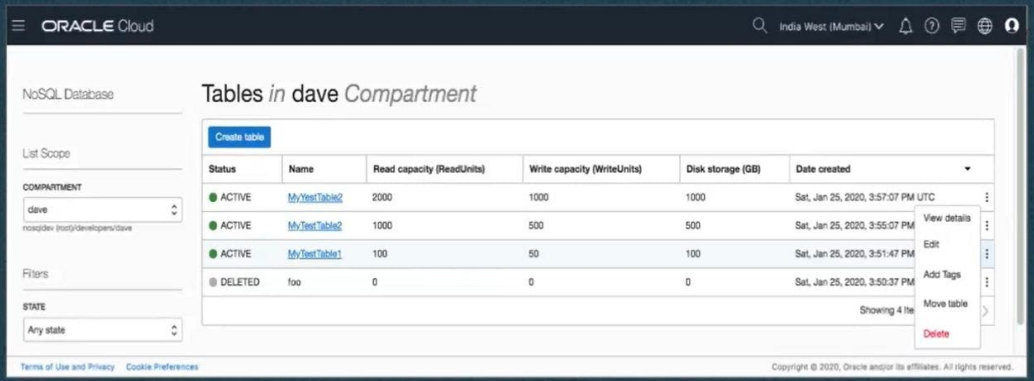
Oracle NoSQL Database Cloud Service

Service Interface, Development Tool



Fully Managed Cloud Service

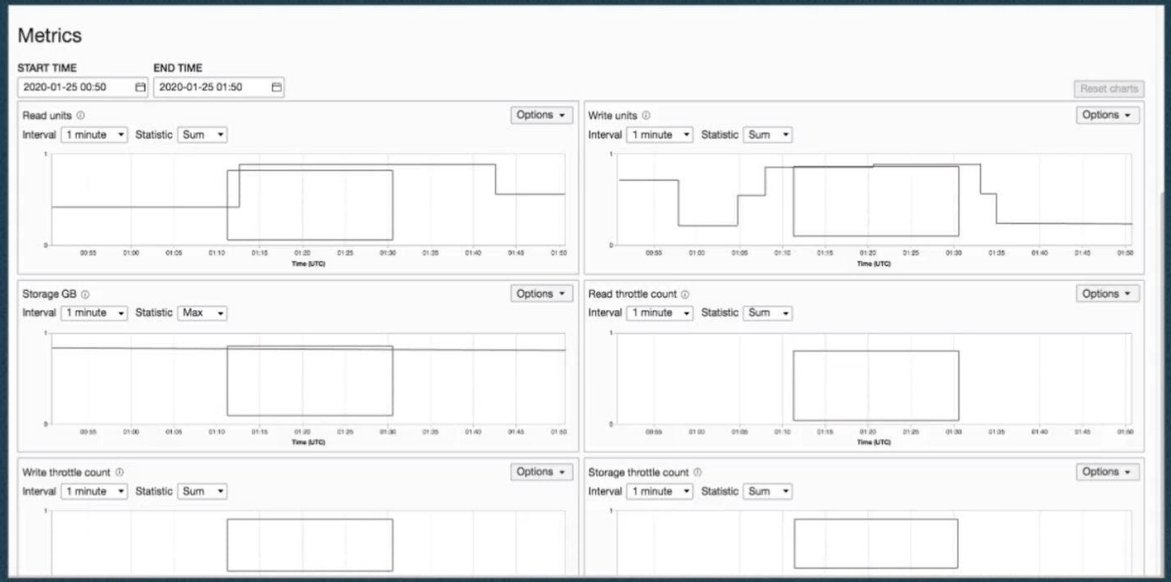
Integrated with OCI cloud console



- Cloud identity
  - Authentication
  - Authorization
- Create and manage tables
- Monitor
  - Throughput
  - Latency
  - Throttling
  - Storage usage

Fully Managed Cloud Service

Integrated monitoring via OCI Console



Differentiators

What make Oracle NoSQL Different?

- Seamless multi-model
  - Key/value, fixed schema, schema-less all in one data store
  - Seamless SQL interoperability between schema-less and fixed schema worlds
- Tunable ACID
  - Shard local full ACID
    - Parent/child tables for easy multi object ACID
  - Carefully balance the tradeoffs of ACID and scale
- Fully managed cloud, throughput provisioned flexibility, no lock-in
  - Run as a fully managed service in Oracle Cloud
  - Run anywhere you like
  - Hybrid cloud via HTTP access