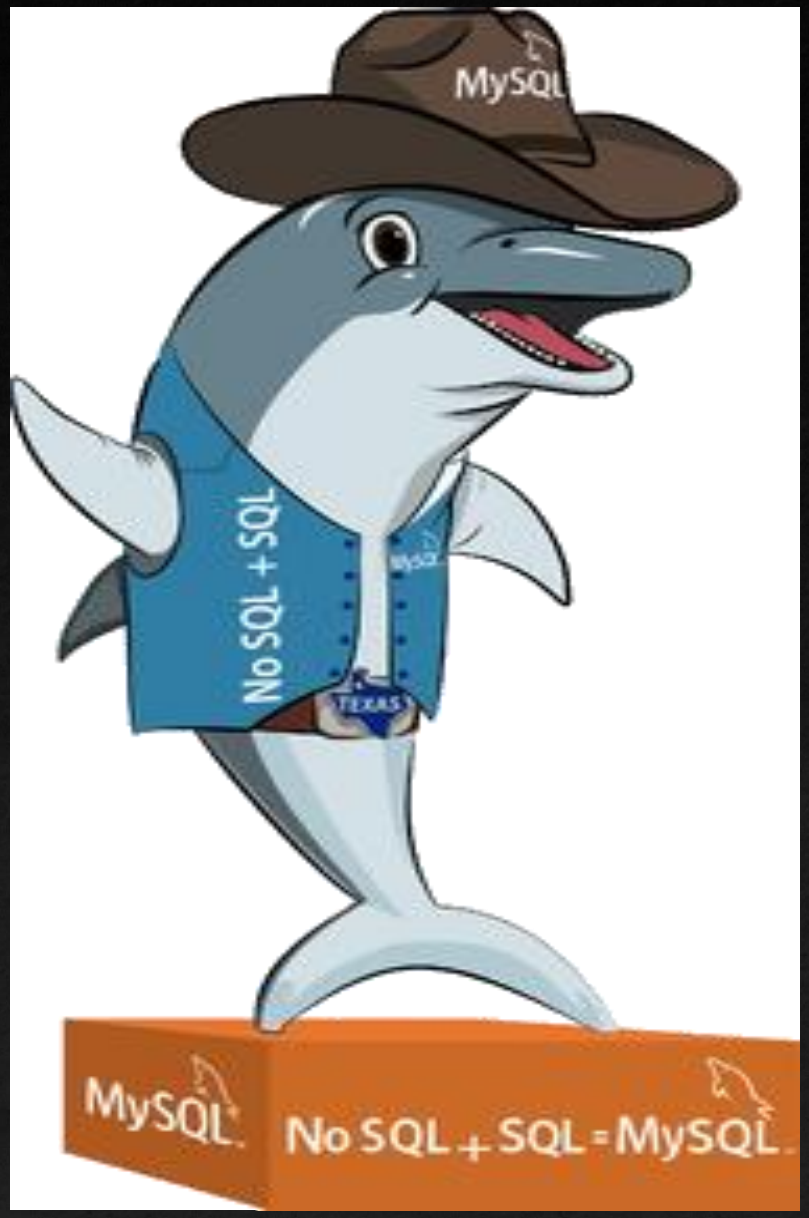
MySQL As NoSQL



Introduction

- Srinivas Tantravahi, 16 years of Experience as DBA
- Worked on Retail ,Telecom,Insurance,Technology,Healthcare Domains.
- Worked on Mysql,Sqlserver ,Azure Cosmos,Postgres,MongoDB,Cassandra,Influx DB.
- Contact : srinimysqldba@gmail.com
- Ktexperts mysqlDB group
- https://www.linkedin.com/in/tantrasunil/

Course Outline

- Background of NoSQL
- Use cases
- MySQLX dev API architecture
- Classic MySQL protocol vs X protocol (NoSQL)
- MySQL shell overview
- Working with MySQL Vs Mongodb (commands)

Course Outline

- Using tables and collections in the database (MySQL + NoSQL)
- Importing JSON to collections
- Query Performance and Fine tuning
- High Availability
- MySQL on Cloud
- Questions

Background of NoSQL

- NoSQL databases are non-tabular databases and store data differently than relational tables
- The main types are document, key-value, wide-column, and graph
- They provide flexible schemas and scale easily with large amounts of data and high user loads
- NoSQL and SQL applications can simultaneously access the same data

Background of NoSQL

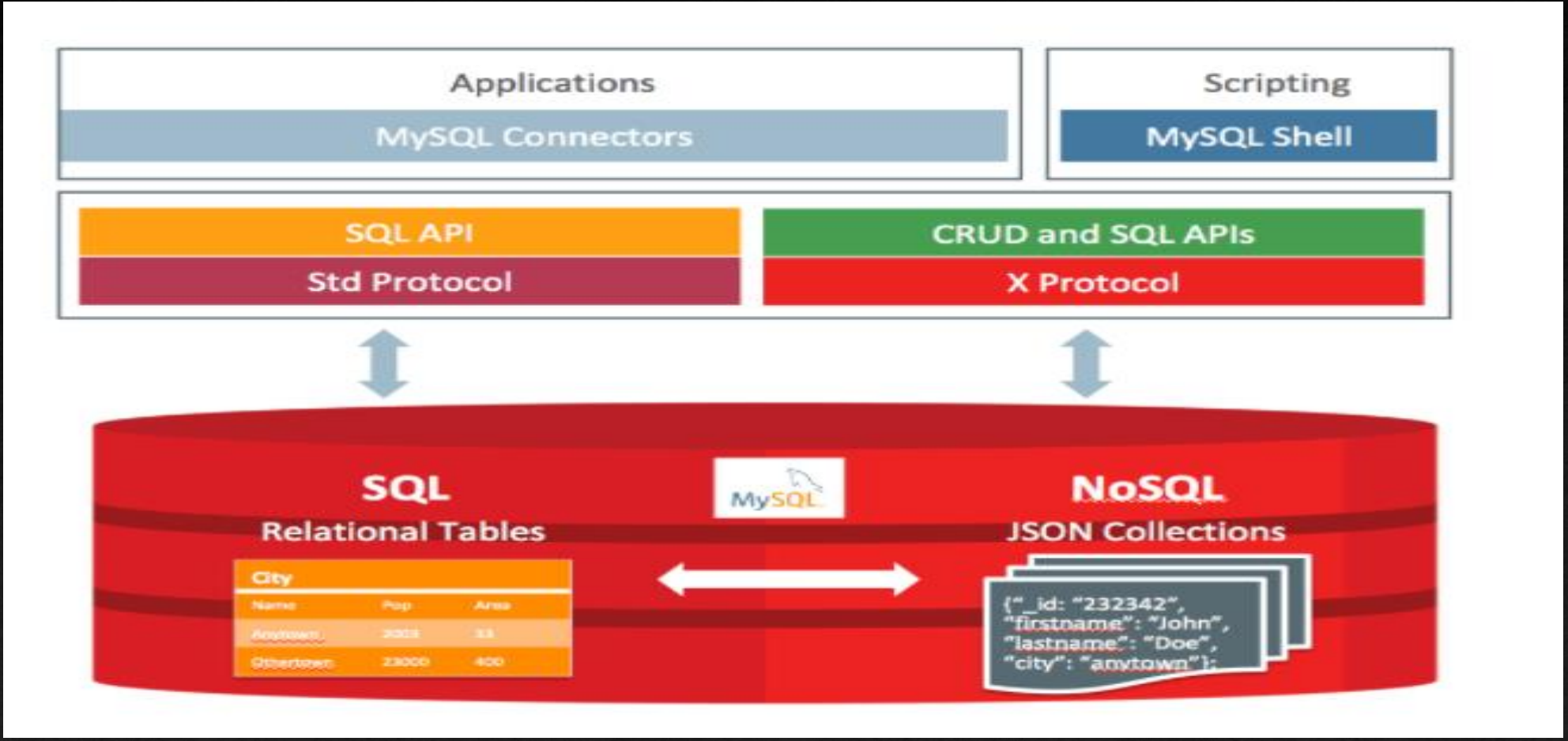
Terminoogy in Nosql

- Database : A database base Is a collection of documents and meta data
- Document : Is the collection of data
- Collection : Data

Use cases

- No need to subscribe / licencing to mongodb in cloud environments
- Mysql on Azure / AWS / Oracle cloud is very cheap when compared to NO sql database like cosmosdb / dynamodb.
- Since no configurations are necessary from Mysql its easy to deploy with existing innodb engine

MySQLX API Architecture



Classic MySQL protocol vs X protocol (NoSQL)

- Classisc protocol : Mysql listens on port 3306
- X protocol :

➢ In order to use Mysql as document DB we need to connect to Mysql as X protocol,which listens to port 33060.

➢ Fully integrated with Mysqlshell

➢ X Plugin is enabled by default in MySQL Server as of MySQL 8.0

➢ Refer to the below link for further documentation (https://dev.mysql.com/doc/x-devapi-userguide/en/)

MySQL shell overview

MySQL Shell includes the following APIs implemented in JavaScript and Python which you can use to develop code that interacts with MySQL.

Admin API :

AdminAPI enables you to administer MySQL instances, using them to create InnoDB Cluster, InnoDB ClusterSet, and InnoDB ReplicaSet deployments, and integrating MySQL Router.

AdminAPI also provides operations to configure users for MySQL Router, to make integration with InnoDB Cluster, InnoDB ClusterSet, and InnoDB ReplicaSet as simple as possible. For more information on AdminAPI, see Chapter 6, MySQL AdminAPI.

X DevAPI :

Enables developers to work with both relational and document data when MySQL Shell is connected to a MySQL server using the X Protocol

X Protocol Support :

MySQL Shell is designed to provide an integrated command-line client for all MySQL products which support X Protocol. The development features of MySQL Shell are designed for sessions using the X Protocol. MySQL Shell can also connect to MySQL Servers that do not support the X Protocol using the classic MySQL protocol.

MySQL shell overview

Mysql shell Modes :

SQL mode : Used to write sql queries

JS mode : Used to write Nosql querie when connected as x protocol and used to manage innodb cluster when connected as classic protocol

Python moe : used to write python programming

Working with MySQL Vs Mongodb (commands)

Command	MongoDB	MySQL Document Store
Login	mongo -u <user> -p <password> --database <dbname>	mysqlsh root@127.0.0.1 --database <dbname>
Select schema	use database_name	\use database_name
Show schemas/dbs	show dbs	session.getSchemas()
Create schema	use database_name	session.createSchema(<schema_name>)
Show collections	show collections or db.getCollectionNames();	db.getCollections()
Create collection	db.createCollection("collectionName");	db.createCollection("collectionName");
Insert document	db.<collectionName>.insert({field1: "value", field2: "value"})	db.<collectionName>.insert({field1: "value", field2: "value"})
Remove document	db.<collectionName>.remove(<deletion criteria>)	db.<collectionName>.remove(<deletion criteria>)
Run SQL command	<Not available>	session.runSql(SQL_command)

Using tables and collections in the database (MySQL + NoSQL)

- A collection is a table with 2+ columns:

  Primary key: `_id`

  JSON document: `doc`

- The document's `_id` field can be supplied or be automatically generated by server as UUID
- This field is also used to populate the primary key
- Can add extra columns and indexes to a collection
- SQL, NoSQL, tables, collections, all can be used simultaneously
- Operations compatible with replication

Using tables and collections in the database (MySQL + NoSQL)

Collection people

```
{
_id: 101,
FirstName
: "
LastName
: "
Street: "123 Elm Street",
State_ID
: 5
}
```

Table states

ID	Name
1	Virginia
2	New York
3	California

```
mysql >
SELECT 'people'.'doc'-->>>'$.firstname','people'.'doc'-->>>'$.lastname'
FROM
'people','states'
WHERE
'people'.'doc'->'$.State_ID'='states'.'id'
AND
'states'.'name'='California'
```

Importing JSON to collections

util. importJson (("restaurants.{ collection : "restaurants",convertBsonOid : true }

Migration from MongoDb to Mysql document store

Export from MongoDB

mongoexport --db test --collection restaurants --out restaurants.json

Import into MySQL

util. importJson (("restaurants.{ collection : "restaurants",convertBsonOid : true }


Questions are the path to learning