Apache Cassandra Database

Software Engineering Branch

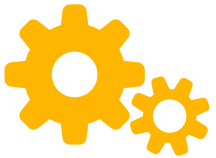
Database Administration class

BENATHMANE Lalia

Charfaoui Younes & Bourbai Ismail

Hello!

Today we're going to present the ins and outs of Cassandra database.



Our process is easy

first

Introduction

Basic of NoSQL, cassandra and the instalation process

second

Key Principles

The Different aspects of the Cassandra database

third

Demo Example

An demo illustrating basics of Cassandra query language

last

Debate

Strenghs and Weaknesses, and some questions

1

Introduction

Let's start with some definitions.

I don't always use Cassandra,
But when I do, I denormalize
-Meme.

NoSQL Databases

A NoSQL database (Not Only SQL) is a database that provides a mechanism to store and retrieve data other than the tabular relations used in relational databases. These databases are schema-free, support easy replication, have simple API, eventually consistent, and can handle huge amounts of data.

NoSQL Databases

In general, they share the following features:

- Schema-free databases
- Easy replication support
- Simple API
- Distributed
- Open Source
- BASE (instead of ACID)
- Huge amount of data
- Horizontally scalable

Apache Cassandra

A distributed NoSQL database system for managing large amounts of structured data across many commodity servers, while providing highly available service and no single point of failure.

Caracteristics

- Data security
- Data sharing
- Physical independence
- Speed of access
- Verification of integrity
- Manipulability
- Limitation of the roundness

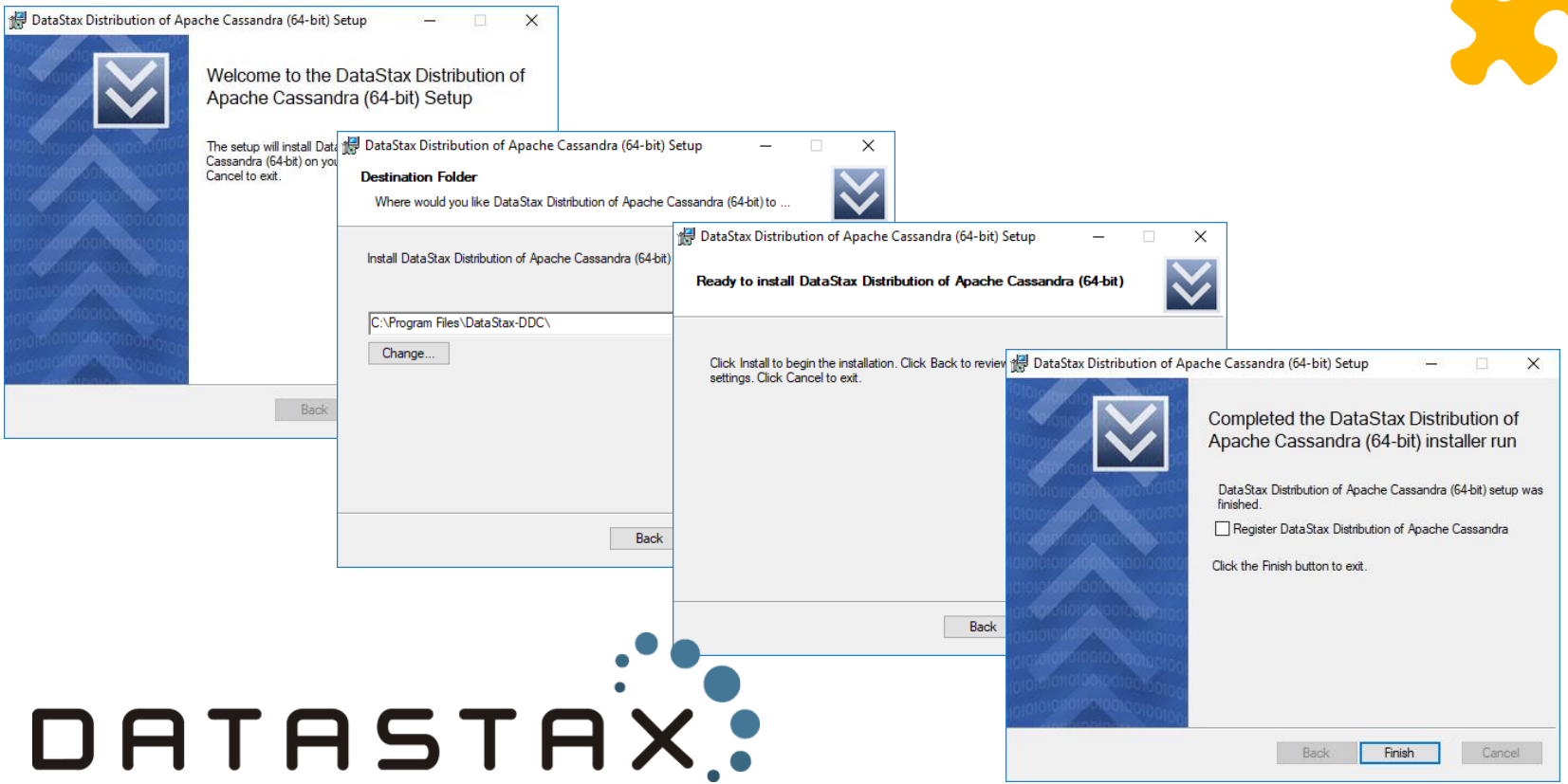
Cassandra support most of the General DBMS characteristics

The Instalation

To strat using cassandra we need to set a workplace for it first.

Requirements:

- The latest version of Java 8
- The latest version of Python 2.7 or 3.6
- Download the Software (DataStax Community Edition for Apache Cassandra™)

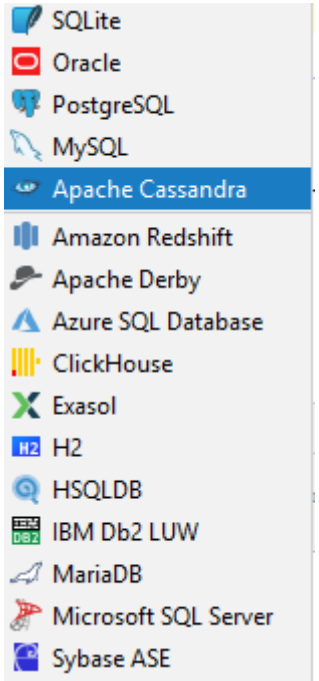

DataStax Distribution of Apache Cassandra (64-bit) Setup
Welcome to the DataStax Distribution of Apache Cassandra (64-bit) Setup
The setup will install Data
Cassandra (64-bit) on you
Cancel to exit.
Back
DataStax Distribution of Apache Cassandra (64-bit) Setup
Destination Folder
Where would you like DataStax Distribution of Apache Cassandra (64-bit) to ...
Install DataStax Distribution of Apache Cassandra (64-bit)
C:\Program Files\DataStax-DDC\
Change...
Back
DataStax Distribution of Apache Cassandra (64-bit) Setup
Ready to install DataStax Distribution of Apache Cassandra (64-bit)
Click Install to begin the installation. Click Back to review
settings. Click Cancel to exit.
Back
DataStax Distribution of Apache Cassandra (64-bit) Setup
Completed the DataStax Distribution of Apache Cassandra (64-bit) installer run
DataStax Distribution of Apache Cassandra (64-bit) setup was finished.
Register DataStax Distribution of Apache Cassandra
Click the Finish button to exit.
Back
Finish
Cancel

DATASTAX

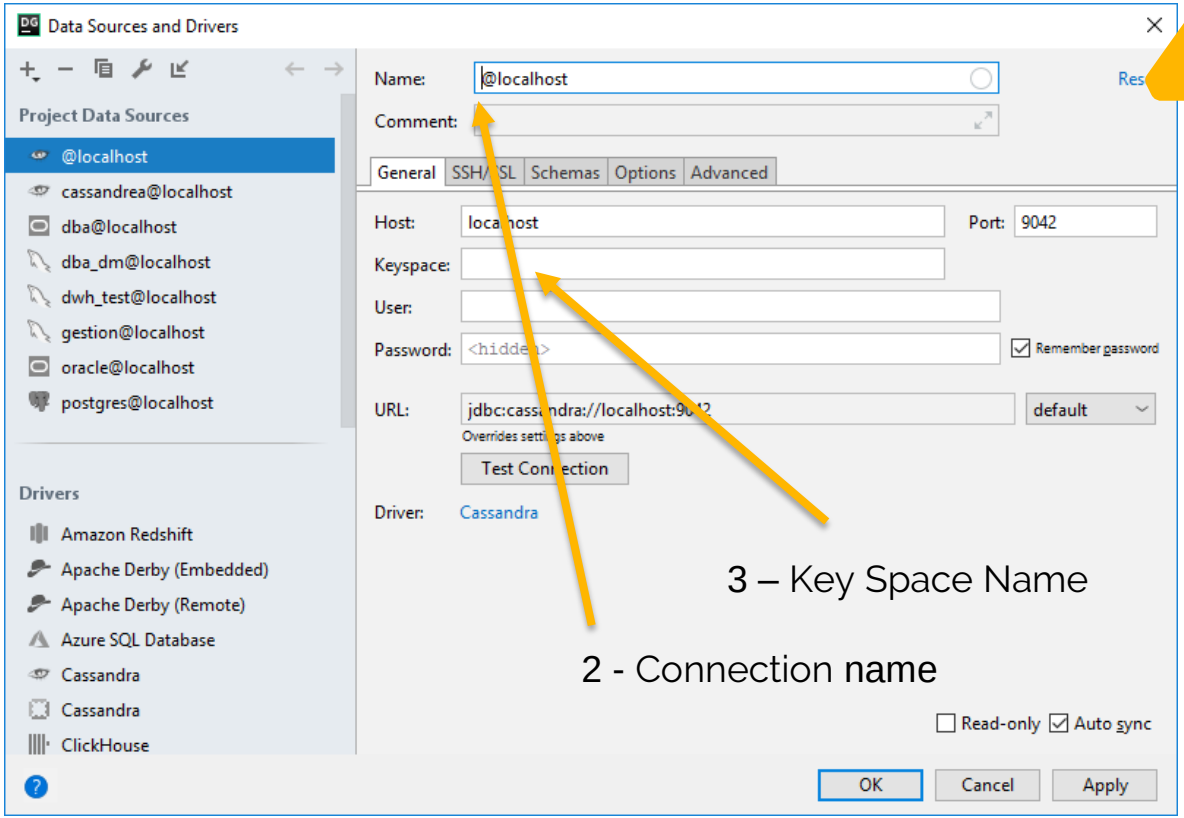
Additional Tool:

You can use DataGrip for interacting with the database instead of the CQLSH, but it does require a license key for using it.

https://www.jetbrains.com/datagrip/



1 – Choose Cassandra



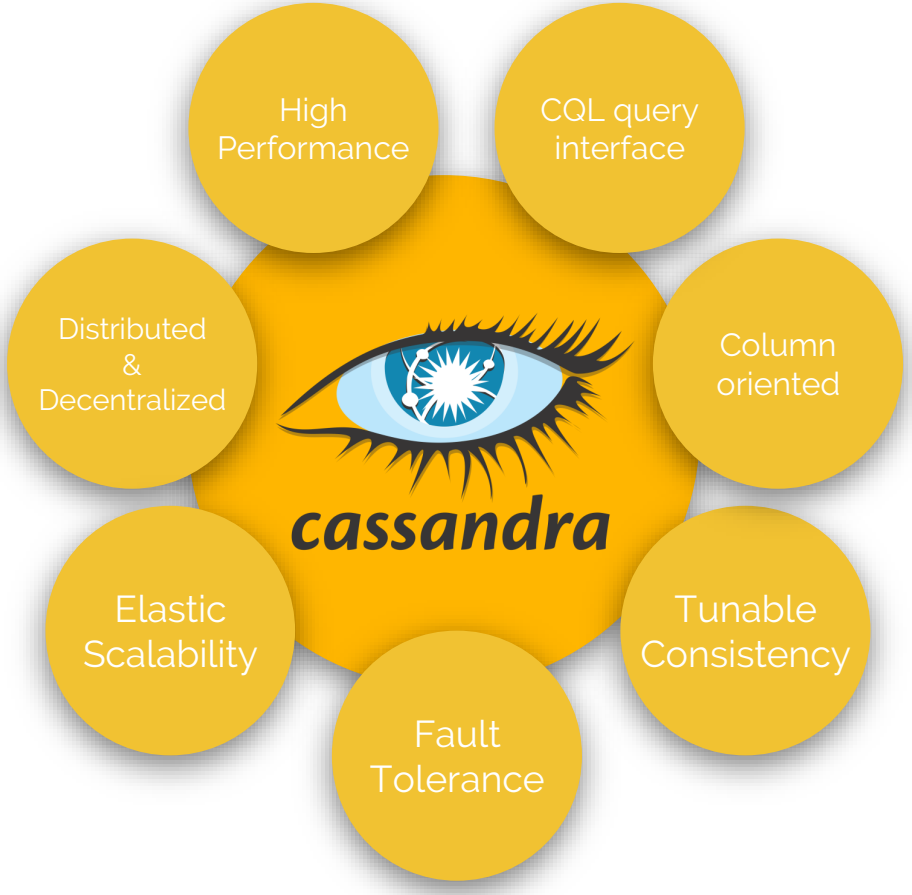
2

Key Principles

The “Must” Be understood of the cassandra

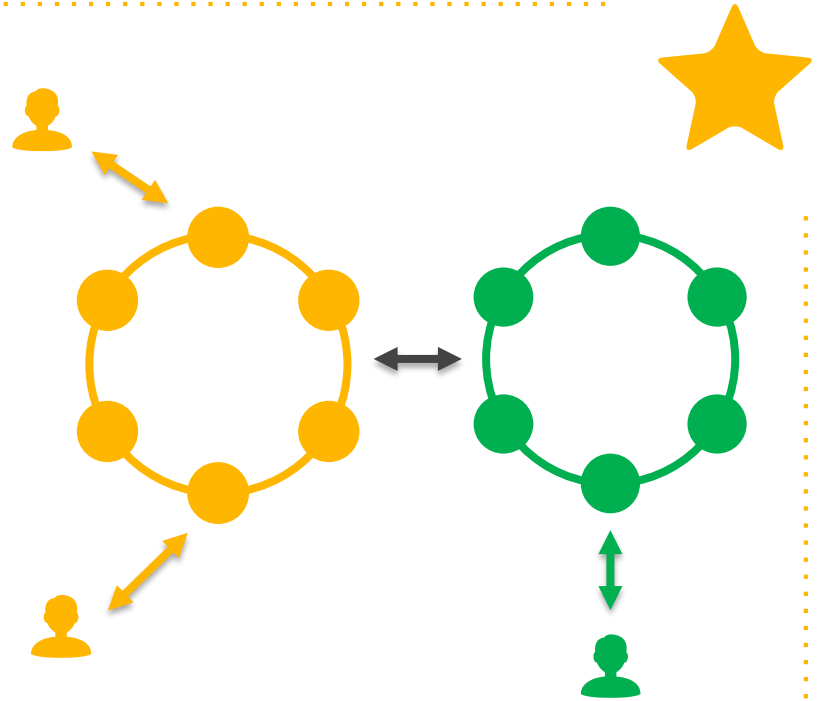
Key Features

This features makes the Cassandra Empire !



Distributed & Decentralized

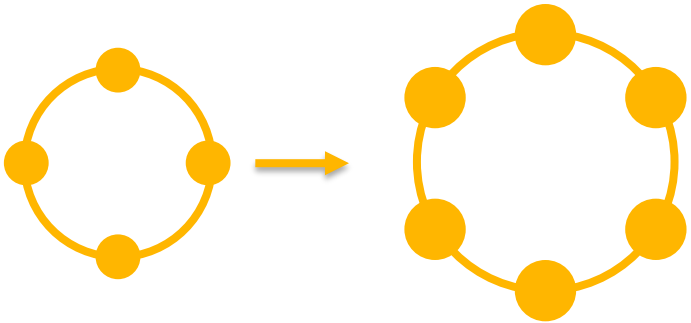
- Distributed: Capable of running on multiple machines
- Decentralized: No single point of failure
- No master-slave issues due to peer-to-peer architecture (protocol "gossip")



Read- and write-requests to any node

Elastic Scalability

- Cassandra scales horizontally, adding more machines that have all or some of the data on
- Adding of nodes increase performance throughput linearly
- Decreasing and increasing the node count happen seamlessly

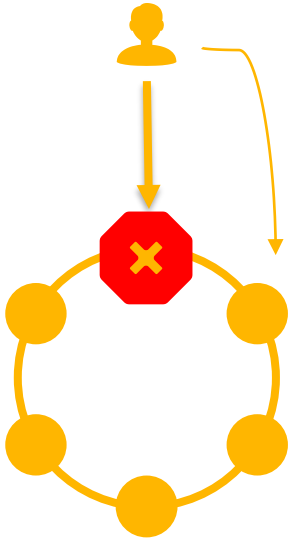


Linearly scales to terabytes and petabytes of data

High Availability & Fault Tolerance

High Availability?

- Multiple networked computers operating in a cluster
- Facility for recognizing node failures
- Forward failing over requests to another part of the system



No single point of failure due to the peer-to-peer architecture

Column oriented
Key-Value Store

- Data is stored in sparse multidimensional hash tables
- A row can have multiple columns not necessarily the same amount of columns for each row
- Each row has a unique key, which also determines partitioning

R1	C1 Key	C2 Key	C3Key	
	C1 Value	C3 Value	C3 Value	.....
R2	C4 Key	C5 Key		
	C4 Value	C5 Value		
.....				

No relations!

Cassandra Query Language

- "*CQL 3 is the default and primary interface into the Cassandra DBMS*"
- Familiar SQL-like syntax that maps to Cassandras storage engine and simplifies data modelling

"SQL-like" but NOT relational SQL

Cassandra Query Language

```
CRETE TABLE songs (
        Id uuid PRIMARY KEY, title text,
        Album text, Artist text,
        data blob );
```

```
SELECT * FROM songs
WHERE id = 'a3e64f8f...';
```

```
SELECT *  FROM songs ;
```

```
INSERT INTO songs (id, title, album, artist)
VALUES( 'a3e64f8f...', 'Hazim ra3d', 'Spacetoon', 'Tarkan' );
```

Cassandra Query Language

```
INSERT INTO songs (id, title)
VALUES( 'a3e64f8f...', 'Al Kanas');
```

This is Possible With Cassandra

Cassandra Query Language

The resulting table in RDMBS is this:

id	title	artist	album	data
a3e64f8f…	Hazim Ra3d	Tarkan	Spacetoon	*null*
g617Dd23…	Al Kanas	*null*	*null*	*null*

Cassandra Query Language

The resulting table in Cassandra is this:

id	title	artist	album	data
a3e64f8f…	Hazim Ra3d	Tarkan	Spacetoon	
g617Dd23…	Al Kanas			

MySQL Comparision:

Statistics based on 50 GB Data

	Cassandra	MySQL
Average Write	**0.12 ms**	**~300 ms**
Average Read	**15 ms**	**~350 ms**

Stats provided by Authors using Facebook data.

And Much More...

The Data Model

How the Database is Organized ?

Data Model

Cluster:

Cassandra database is distributed over several machines that operate together. The outermost container is known as the Cluster. For failure handling, every node contains a replica, and in case of a failure, the replica takes charge. Cassandra arranges the nodes in a cluster, in a ring format, and assigns data to them.

Data Model

Keyspace

Outermost container for data (one or more column families), like database in RDBMS.

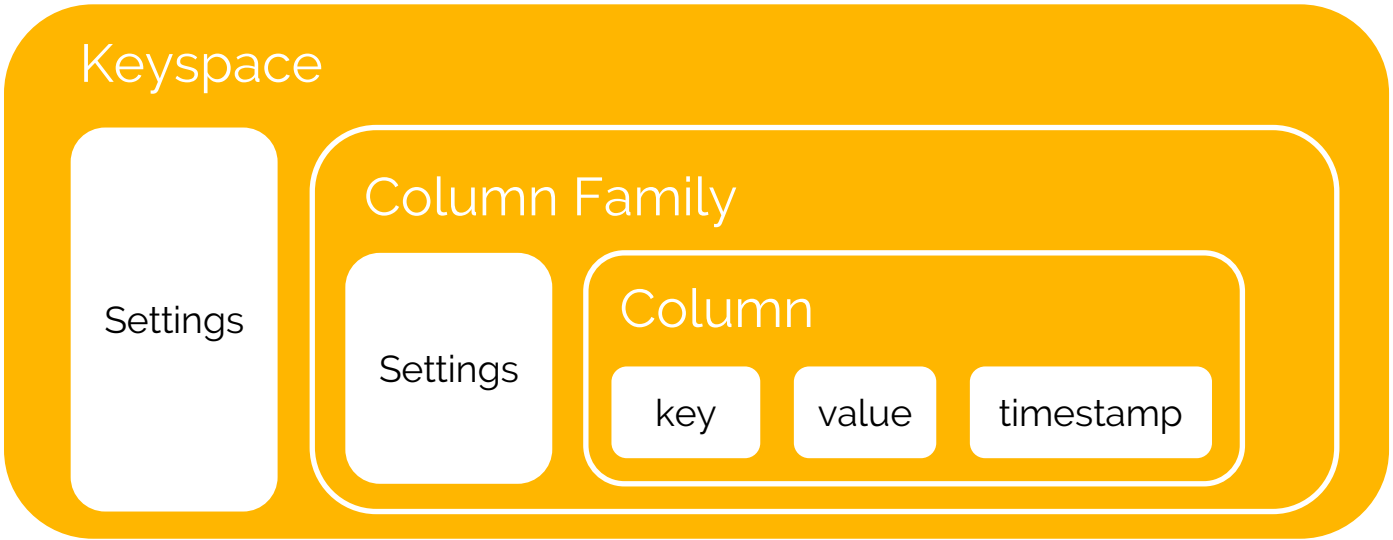
Column family

Contains Super columns or Columns (but not both).

Column

Basic data structures with: key, value, timestamp

Data Model


Keyspace
Settings
Column Family
Settings
Column
key
value
timestamp

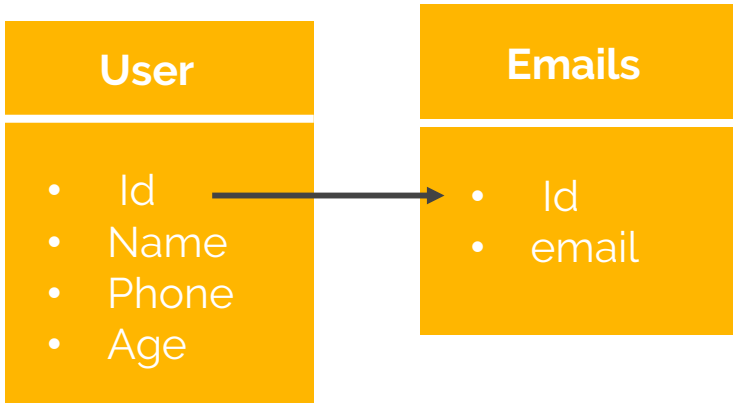
3

Demo

Example illustrating different part of CQL

Examples Using CQL

The Following Slides will demonstrate different cases with different CQL interfaces like DDL, DML etc..



Interface DDL

Same as SQL, but with keyspaces and types option added.

DROP

- Type
- Keyspace , Table
- Index , Trigger

CREATE

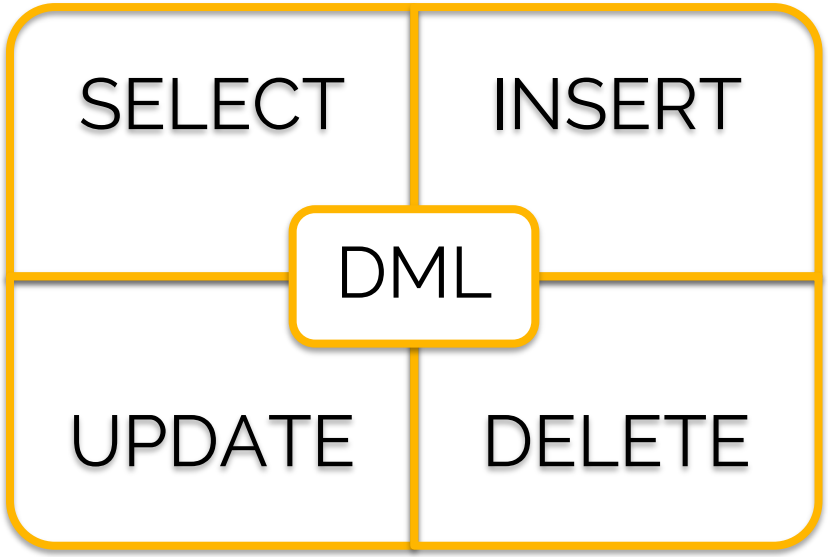
- Type
- Keyspace , Table
- Index , Trigger

ALTER

- Type
- Keyspace , Table
- Index , Trigger

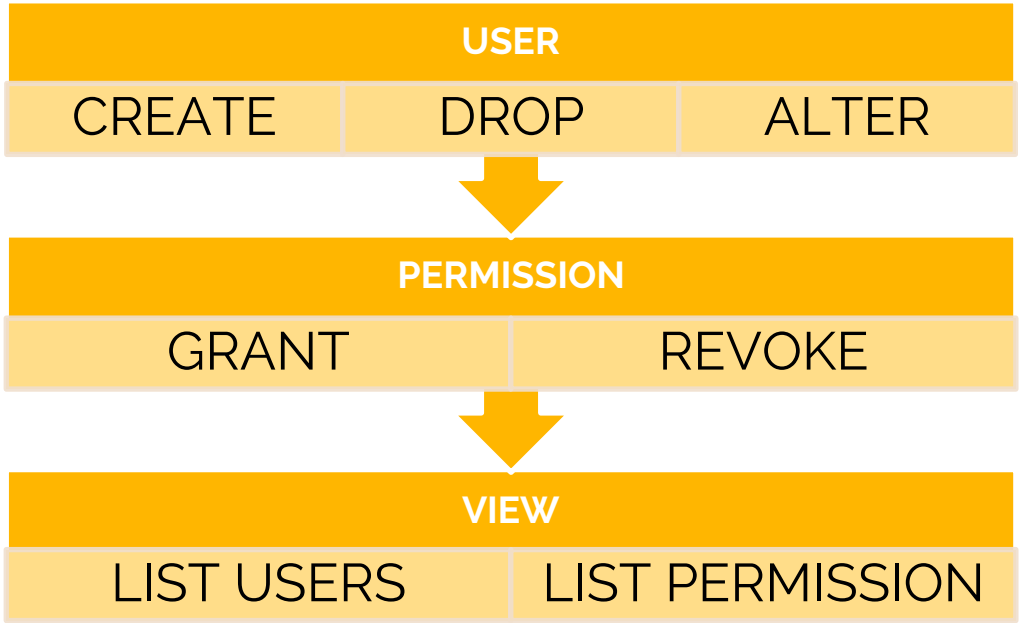
Interface DML

The DML Interface is the Same With Normal SQL DML



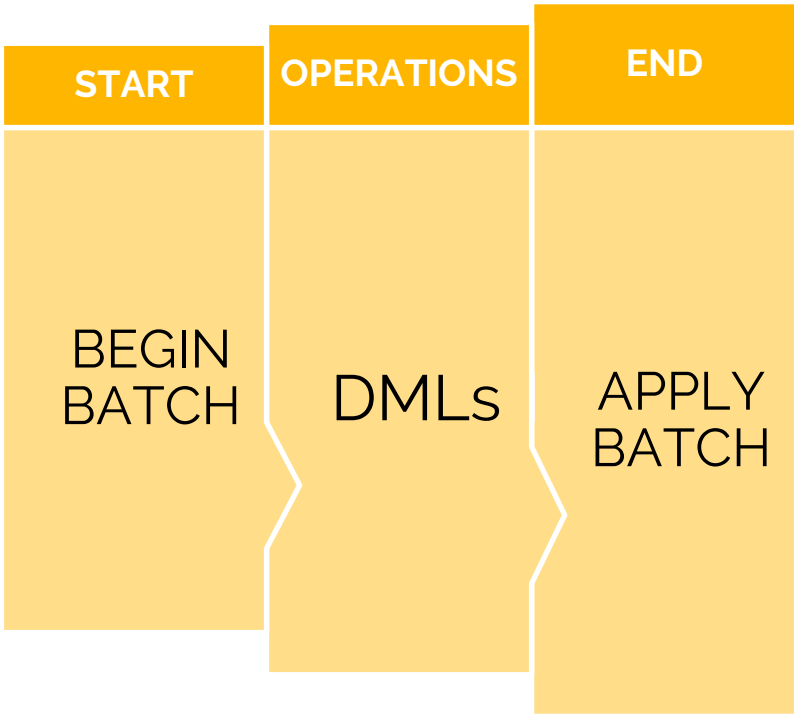
Interface DCL

Create users (Roles), give them permission, and start using them.



Interface TCL

For multiple operations use the BATCH command



Metadata & Logging

How to see metadata and make logging in Cassandra database ?

Metadata Using Describe

keyspace

```
Describe keyspace name
```

Table

```
Describe table keyspace_name .table_name
```

Others

```
Describe keyspaces, tables, schema
```

Metadata Keyspace

Query the defined key spaces using the SELECT statement.

```
SELECT * FROM
system_schema.keyspaces
```

keyspace_name	durable_writes	replication
test	True	{'class': 'org.apache'}....
.....	.....	.....

Metadata Tables

Getting information about tables in the test keyspace.

```
SELECT * FROM system_schema.tables
WHERE keyspace_name = test';
```

keyspace_name	table_name	…….
test	users	……..
……	……	……

Metadata Columns

Getting information about columns in the users tables.

```
SELECT * FROM system_schema.columns
WHERE keyspace_name = test' AND table_name = 'users';
```

table_name	column_name	kind	type	…….
users	age	regular	int	……..
……	……	……	……	……

Logging with System.log

To see what is happening in the database, you can use the system.log file in the Cassandra home to directory to track creational query.

```
{CASSANDRA HOME}/utils/cassandra.logdirISUNDEFINED/
```

Here is an Example

```
{CASSANDRA HOME}/utils/cassandra.logdirISUNDEFINED/
```

Logging with System.log

Here is an Example

```
INFO  [main] 2018-11-08 23:48:36,960
MigrationManager.java:302 - Create new Keyspace:
KeyspaceMetadata {name=system_traces,
params=KeyspaceParams {durable_writes=true,
replication=ReplicationParams
{class=org.apache.cassandra.locator.SimpleStrategy,
replication_factor=2 }
```

Logging with Tracing

It's an option to activate in the Cassandra database

```
TRACING [ ON | OFF]
```

The result will be on different keyspace called system_traces. In a table called events

```
USE system_traces;
SELECT * FROM events;
```

Logging with Tracing

Example:

```
INSERT INTO product(id , name) VALUES (UUID(), 'Hello');
```

Result:

```
Execute CQL3 query
Parsing insert into product(id , name) values(UUID(), 'Hello');
Preparing statement
.....
```

4

Debate

Strength and weakness of Cassandra.

Strengths (1)

- Linear scale performance

The ability to add nodes without failures leads to predictable increases In performance

- Supports multiple languages

Python, C#/.NET, C++, Ruby, Java, Go, and many more…

- Operational and developmental simplicity

There are no complex software tiers to be managed, so administration duties are greatly simplified.

Strengths (2)

- Ability to deploy across data centers

Cassandra can be deployed across multiple, geographically dispersed data centers

- Cloud availability

Installations in cloud environments

- Peer to peer architecture

Cassandra follows a peer-to-peer architecture, instead of master-slave architecture

Strengths (3)

- Flexible data model

Supports modern data types with fast writes and reads

- Fault tolerance

Nodes that fail can easily be restored or replaced

- High Performance

Cassandra has demonstrated brilliant performance under large sets of data

Strengths (4)

- Schema-free/Schema-less

In Cassandra, columns can be created at your will within the rows. Cassandra data model is also famously known as a schema-optional data model

- AP-CAP

Cassandra is typically classified as an AP system, meaning that availability and partition tolerance are generally considered to be more important than consistency in Cassandra

Weaknesses (1)

Use Cases where is better to avoid using Cassandra

- If there are too many joins required to retrieve the data
- To store configuration data
- During compaction, things slow down and throughput degrades
- Basic things like aggregation operators are not supported
- Range queries on partition key are not supported

Weaknesses (2)

Use Cases where is better to avoid using Cassandra

- If there are transactional data which require 100% consistency
- Cassandra can update and delete data but it is not designed to do so

Thanks!

Any questions?