# INTRODUCTION TO CASSANDRA

## UNIT -2

### Chapter 7

Sowmya V
Assistant Professor
BMSCE, Bangalore

# Learning Objectives and Learning Outcomes

| Learning Objectives | Learning Outcomes |
|---|---|
| Introduction to Cassandra<br><br>1. To study the features of Cassandra.<br><br>2. To learn how to perform CRUD(Create, Read, Update and Delete) operations.<br><br>3. To learn about collections in Cassandra.<br><br>4. To import from and export to CSV format. | a) To comprehend the reasons behind the popularity of NoSQL database.<br><br>b) To be able to perform CRUD operations.<br><br>c) To distinguish between collections types such as SET, LIST and MAP.<br><br>d) To be able to successfully import from CSV.<br><br>e) To be able to successfully export to CSV. |

Big Data and Analytics by Seema Acharya and Subhashini Chellappan
Copyright 2015, WILEY INDIA PVT. LTD.

# Agenda

- Apache Cassandra – An Introduction
- Features of Cassandra
  - Peer-to-Peer Network
  - Writes in Cassandra
  - Hinted Handoffs
  - Tunable Consistency: Read Consistency and Write Consistency
- CQL Data Types
- CQLSH
- CRUD : Insert, Update, Delete and Select
- Collections : Set, List and Map
- Time To Live (TTL)
- Import and Export


Copyright 2015, WILEY INDIA PVT. LTD.

# NoSQL Database

- A NoSQL database (sometimes called as Not Only SQL) is a database that provides a mechanism to store and retrieve data
- These databases are schema-free
  - support easy replication,
  - have simple API,
  - eventually consistent,
  - and can handle huge amounts of data.
- The primary objective of a NoSQL database is to have
  - simplicity of design,
  - horizontal scaling, and
  - finer control over availability.

# NoSQL vs. Relational Database

| Relational Database | NoSql Database |
|---|---|
| Supports powerful query language. | Supports very simple query language. |
| It has a fixed schema. | No fixed schema. |
| Follows ACID (Atomicity, Consistency, Isolation, and Durability). | It is only “eventually consistent”. |
| Supports transactions. | Does not support transactions. |

# Apache Cassandra – An Introduction

- Cassandra was born at Facebook for inbox search.
- It was open-sourced by Facebook in July 2008.
- Cassandra became an Apache Incubator project in March 2009.
- It was made an Apache top-level project since February 2010.

- It is built on Amazon's dynamo and Google's BigTable.
- Its highly scalable, high performance distributed database.
- It is a column-oriented database designed to support peer-to-peer symmetric nodes instead of the master-slave architecture.

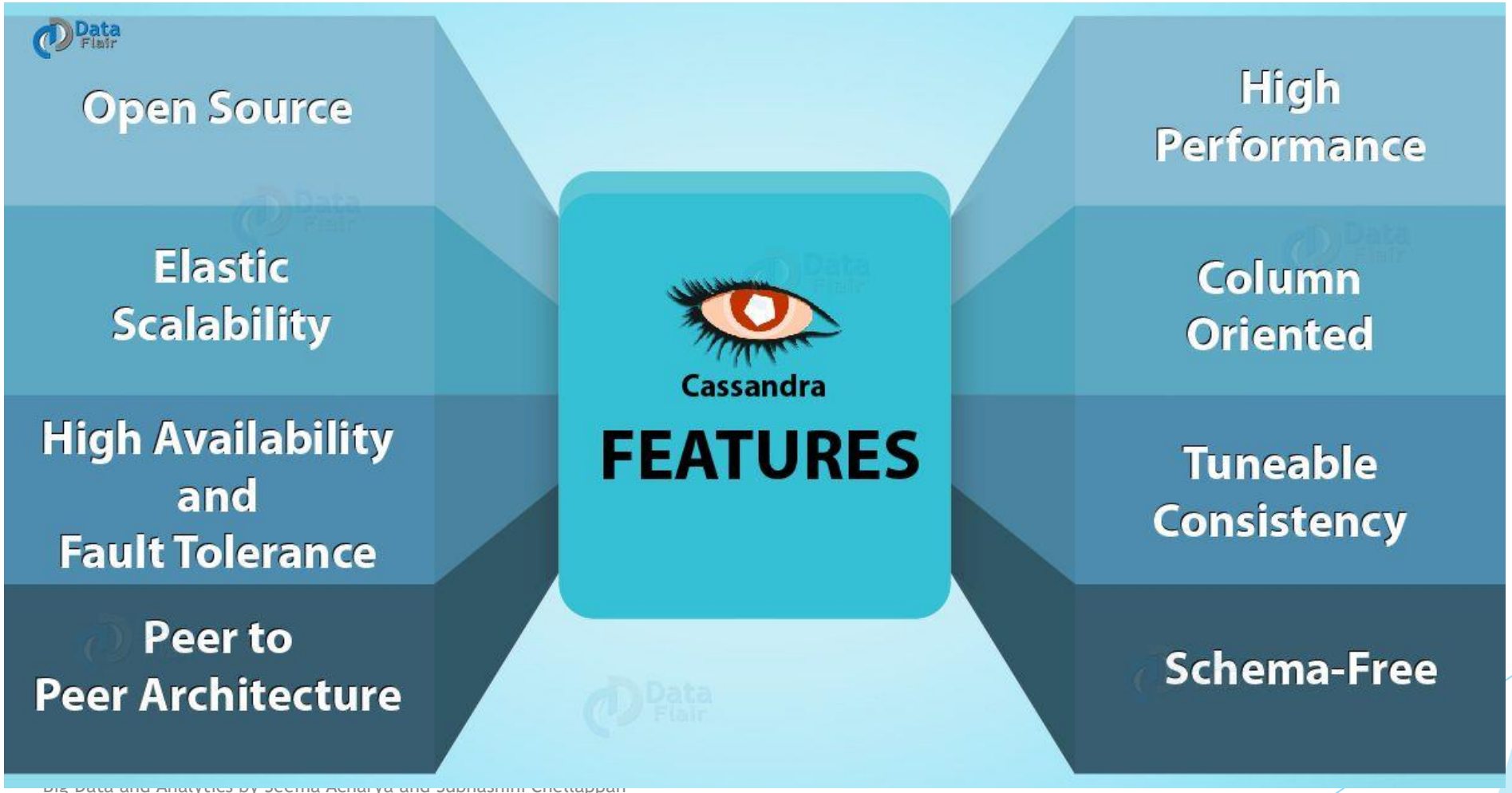

Copyright 2015, WILEY INDIA PVT. LTD.

# What is Apache Cassandra?

- Apache Cassandra is an open source, distributed and decentralized/distributed storage system (database), for managing very large amounts of structured data spread out across the world.

- It provides highly available service with no single point of failure.

# Features of Cassandra

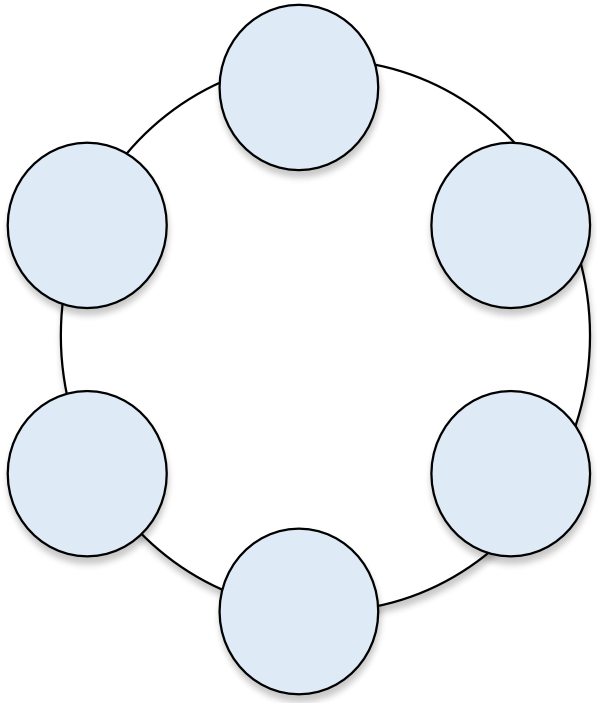



Big Data and Analytics by Seema Acharya and Subhashini Chellappan
Copyright 2015, WILEY INDIA PVT. LTD.

# Peer to Peer Network

- The aim of Cassandra's design is overall system availability and ease of scaling
- Does NOT have a Master Slave Architecture
- This design also makes it easier to scale Cassandra by adding new nodes

**Sample Cassandra Cluster**



# Gossip and Failure Detection

- Cassandra uses a gossip protocol for intra-ring communication so that each node can have state information about other nodes
- The gossiper runs every second on a timer
- Because Cassandra gossip is used for failure detection, the Gossiper class maintains a list of nodes that are alive and dead.

- Failure detection is implemented in Cassandra by the org.apache.cassandra.gms
- For repairing unread data, Cassandra uses anti-entropy version of gossip protocol.
- Anti-entropy, compares all replicas of each piece of data and update each replica to the newest version.

Big Data and Analytics by Seema Acharya and Subhashini Chellappan
Copyright 2015, WILEY INDIA PVT. LTD.

# Partitioner

- Distribution of data on various nodes in cluster
- Partitioner is a Hash function which computes tokens.
- Partition key helps to identify a row uniquely

# Replication Factor

- Determines number of copies of data that will be stored across nodes in a cluster
- Two Strategies
  - Simple Strategy
  - Network Topology Strategy

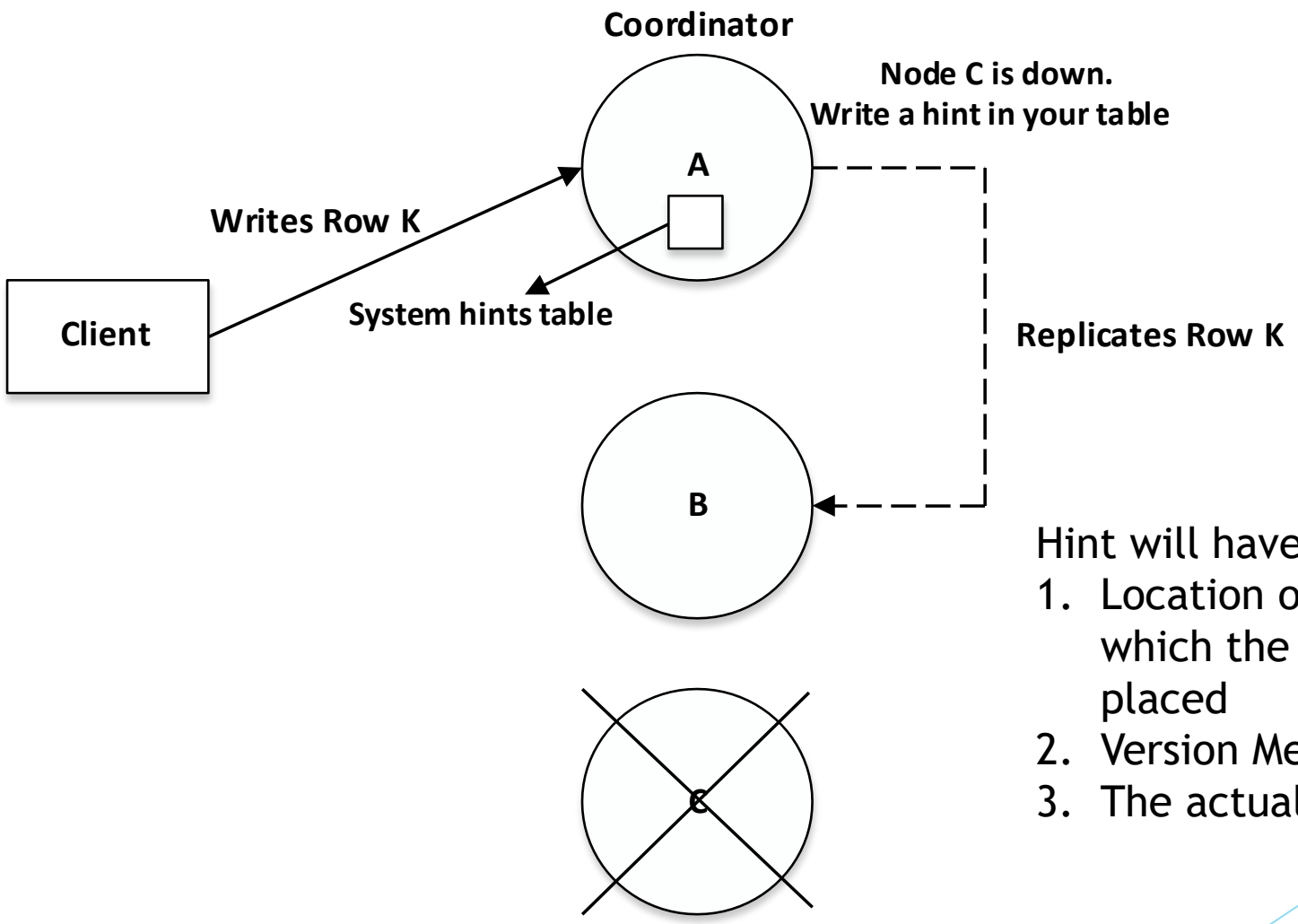
# Writes in Cassandra

- A client that initiates a write request.
- It is first written to the commit log. A write is taken as successful only if it is written to the commit log.
- The next step is to push the write to a memory resident data structure called Memtable. A threshold value is defined in the Memtable.
- When the number of objects stored in the Memtable reaches a threshold, the contents of Memtable are flushed to the disk in a file called SSTable (Stored string Table). Flushing is a non-blocking operation.
- It is possible to have multiple Memtables for a single column family. One out of them is current and the rest are waiting to be flushed.


Copyright 2015, WILEY INDIA PVT. LTD.

# Hinted Handoffs

Why Cassandra is all for availability?




Hint will have:

1. Location of the node on which the replica is to be placed
2. Version Metadata
3. The actual data

# Tunable Consistency


Copyright 2015, WILEY INDIA PVT. LTD.

# Read Consistency

| | |
|---|---|
| ONE | Returns a response from the closest node (replica) holding the data. |
| QUORUM | Returns a result from a quorum of servers with the most recent timestamp for the data. |
| LOCAL_QUORUM | Returns a result from a quorum of servers with the most recent timestamp for the data in the same data center as the coordinator node. |
| EACH_QUORUM | Returns a result from a quorum of servers with the most recent timestamp in all data centers. |
| ALL | This provides the highest level of consistency of all levels and the lowest level of availability of all levels. It responds to a read request from a client after all the replica nodes have responded. |


Copyright 2015, WILEY INDIA PVT. LTD.

# Write Consistency

| | |
|---|---|
| ALL | This is the highest level of consistency of all levels as it necessitates that a write must be written to the commit log and Memtable on all replica nodes in the cluster. |
| EACH_QUORUM | A write must be written to the commit log and Memtable on a quorum of replica nodes in all data centers. |
| QUORUM | A write must be written to the commit log and Memtable on a quorum of replica nodes. |
| LOCAL_QUORUM | A write must be written to the commit log and Memtable on a quorum of replica nodes in the same data center as the coordinator node. This is to avoid latency of inter-data center communication. |
| ONE | A write must be written to the commit log and Memtable of at least one replica node. |
| TWO | A write must be written to the commit log and Memtable of at least two replica nodes. |
| THREE | A write must be written to the commit log and Memtable of at least three replica nodes. |
| LOCAL_ONE | A write must be sent to, and successfully acknowledged by, at least one replica node in the local data center. |

Copyright 2015, WILEY INDIA PVT. LTD.

# CQL Data types


Copyright 2015, WILEY INDIA PVT. LTD.

# CQL Data types

| Int | 32 bit signed integer |
|---|---|
| Bigint | 64 bit signed long |
| Double | 64-bit IEEE-754 floating point |
| Float | 32-bit IEEE-754 floating point |
| Boolean | True or false |
| Blob | Arbitrary bytes, expressed in hexadecimal |
| Counter | Distributed counter value |
| Decimal | Variable – precision integer |
| List | A collection of one or more ordered elements |
| Map | A JSON style array of elements |
| Set | A collection of one or more elements |
| Timestamp | Date plus time |
| Varchar | UTF 8 encoded string |
| Varint | Arbitrary-precision integers |
| Text | UTF 8 encoded string |


Copyright 2015, WILEY INDIA PVT. LTD.

# CQLSH

## Logging to cqlsh

| | |
|---|---|
| ***Objective:*** | What is it that we are trying to achieve here? |
| ***Input (optional):*** | What is the input that has been given to us to act upon? |
| ***Act:*** | The actual statement /command to accomplish the task at hand. |
| ***Outcome:*** | The result/output as a consequence of executing the statement. |

Big Data and Analytics by Seema Acharya and Subhashini Chellappan
Copyright 2015, WILEY INDIA PVT. LTD.

# Example:

**Objective:** To get help with CQL.

**Act:**

**Help**

**Outcome:**

```
C:\windows\system32\cmd.exe - Python cqlsh

d:\apache-cassandra-2.0.0\apache-cassandra-2.0.0\apache-cassandra-2.0.0\bin>Python cqlsh
Connected to Test Cluster at localhost:9160.
[cqlsh 4.0.0 | Cassandra 2.0.0 | CQL spec 3.1.0 | Thrift protocol 19.37.0]
Use HELP for help.
cqlsh> help

Documented shell commands:
==========================
CAPTURE       COPY  DESCRIBE  EXPAND  SHOW    TRACING
CONSISTENCY   DESC  EXIT      HELP    SOURCE

CQL help topics:
================
ALTER                          CREATE_TABLE_OPTIONS   REVOKE
ALTER_ADD                      CREATE_TABLE_TYPES     SELECT
ALTER_ALTER                    CREATE_USER            SELECT_COLUMNFAMILY
ALTER_DROP                     DELETE                 SELECT_EXPR
ALTER_RENAME                   DELETE_COLUMNS         SELECT_LIMIT
ALTER_USER                     DELETE_USING           SELECT_TABLE
ALTER_WITH                     DELETE_WHERE           SELECT_WHERE
APPLY                          DROP                   TEXT_OUTPUT
ASCII_OUTPUT                   DROP_COLUMNFAMILY      TIMESTAMP_INPUT
BEGIN                          DROP_INDEX             TIMESTAMP_OUTPUT
BLOB_INPUT                     DROP_KEYSPACE          TRUNCATE
BOOLEAN_INPUT                  DROP_TABLE             TYPES
CREATE                         DROP_USER              UPDATE
CREATE_COLUMNFAMILY            GRANT                  UPDATE_COUNTERS
CREATE_COLUMNFAMILY_OPTIONS    INSERT                 UPDATE_SET
CREATE_COLUMNFAMILY_TYPES      LIST                   UPDATE_USING
CREATE_INDEX                   LIST_PERMISSIONS       UPDATE_WHERE
CREATE_KEYSPACE                LIST_USERS             USE
CREATE_TABLE                   PERMISSIONS            UUID_INPUT

cqlsh>
```

# A Simple Introduction


- *Figure - A list of values*

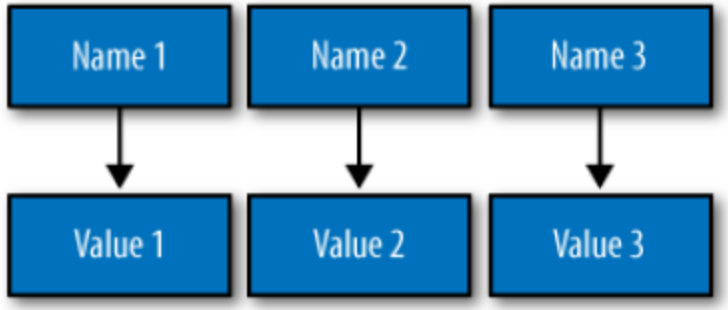



*Figure - A map of name/value pairs*

Example: Column names maybe firstName, lastName, phone, email, and so on

# Data Model

- Columns are grouped into Column Families(CF):
  - CFs have to be defined in advance → structured storage system
  - The number of CFs is not limited per table
- Types of Column Families:
  - Simple
  - Super (nested Column Families)
- Column
  - Has (Name, Value, Timestamp) and Can be ordered by timestamps or name
- Row
  - Can have  a different number of columns

# Column Family

Cassandra defines a *column family* to be a logical division that associates similar data. For example, we might have a User column family, a Hotel column family, an AddressBook column family, and so on

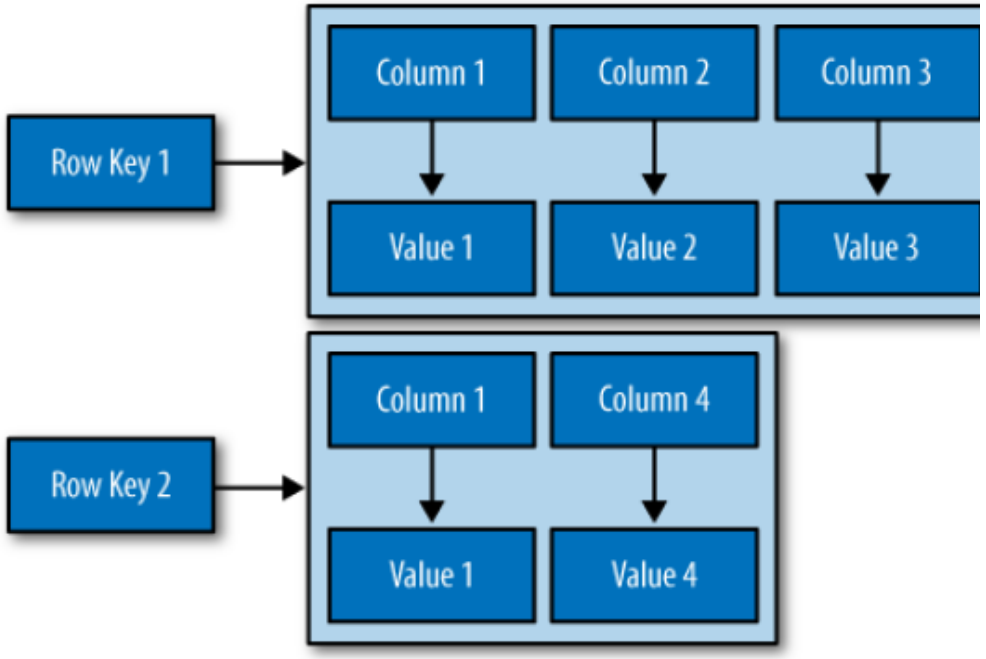



*Figure - A column family*

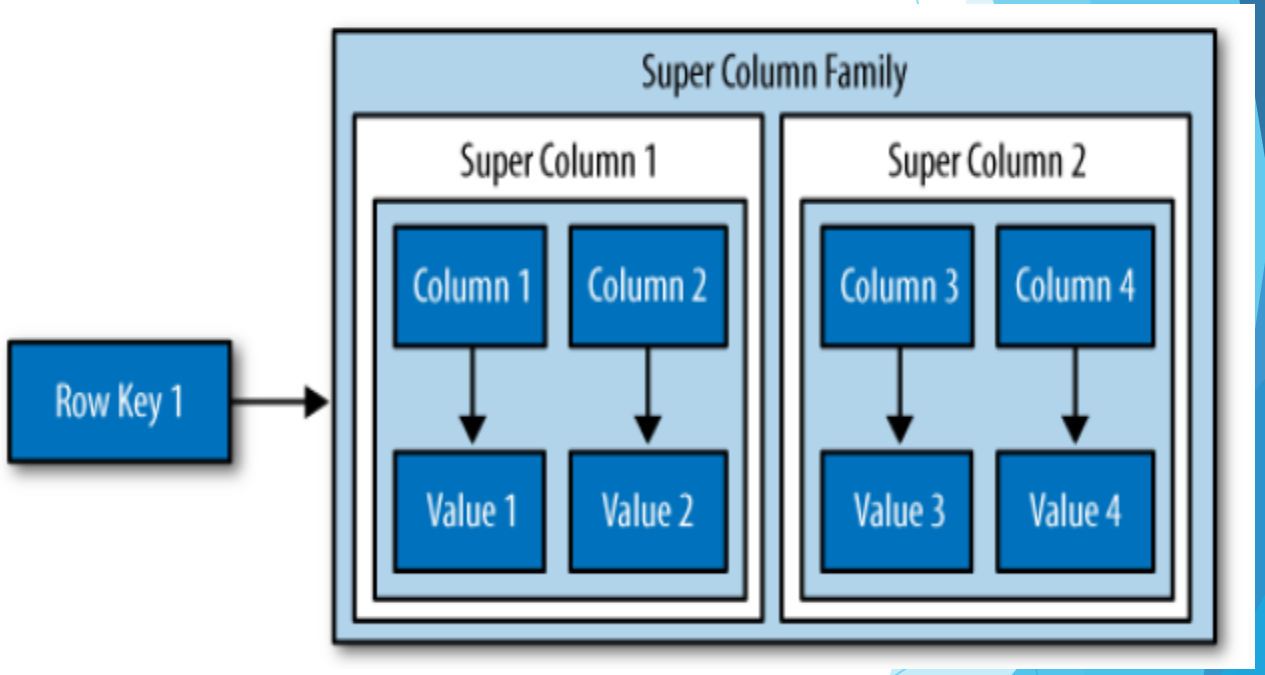



*Figure - A super column family*

# Cluster

- Cassandra database is distributed over several machines that operate together.
- Outermost container is known as the Cluster.
- Cassandra arranges the nodes in a cluster, in a ring format, and assigns data to them.

# Keyspace

- A cluster is a container for keyspaces
- A *keyspace* is the outermost container for data in Cassandra
- A keyspace has a name and a set of attributes that define keyspace-wide behaviour.
- For example, if your application is called Twitter, you would probably have a cluster called Twitter-Cluster and a keyspace called Twitter.

# CRUD (CREATE, READ, UPDATE, AND DELETE) OPERATIONS

Objective: To create a keyspace by the name “Students”

Act:

```
CREATE KEYSPACE Students WITH REPLICATION = {
        'class':'SimpleStrategy', 'replication_factor':1
 };
```

**Outcome:**

```
C:\windows\system32\cmd.exe - Python cqlsh
cqlsh> CREATE KEYSPACE Students WITH REPLICATION = {
   ...                 'class':'$impleStrategy',
   ...                 'replication_factor':1
   ...   };
cqlsh>
```

Big Data and Analytics by Seema Acharya and Subhashini Chellappan
Copyright 2015, WILEY INDIA PVT. LTD.

**Objective:** To describe all the existing keyspaces.

**Act:**

**DESCRIBE KEYSPACES;**

**Outcome:**

```
C:\windows\system32\cmd.exe - python cqlsh

d:\apache-cassandra-2.0.0\apache-cassandra-2.0.0\apache-cassandra-2.0.0\bin>python cq
Connected to Test Cluster at localhost:9160.
[cqlsh 4.0.0 | Cassandra 2.0.0 | CQL spec 3.1.0 | Thrift protocol 19.37.0]
Use HELP for help.
cqlsh> describe keyspaces;

system  students  system_traces

cqlsh>
```

**Objective:** To get more details on the existing keyspaces such as keyspace name, durable writes, strategy class, strategy options, etc.

**Act:**

```
SELECT *
        FROM system.schema_keyspaces;
```

**Outcome:**

```
cqlsh> SELECT * FROM system.schema_keyspaces;

 keyspace_name | durable_writes | strategy_class                                    | strategy_options
---------------+----------------+---------------------------------------------------+----------------------------
       demo_con |           True | org.apache.cassandra.locator.SimpleStrategy       | {"replication_factor":"3"}
         system |           True |  org.apache.cassandra.locator.LocalStrategy       |                         {}
 system_traces |           True | org.apache.cassandra.locator.SimpleStrategy       | {"replication_factor":"1"}
       students |           True | org.apache.cassandra.locator.SimpleStrategy       | {"replication_factor":"1"}

(4 rows)
```

***Note:*** Cassandra converted the Students keyspace to lowercase as quotation marks were not used.

**Objective:** To use the keyspace "Students", use the following command:

Use keyspace_name

Use connects the client session to the specified keyspace.

**Act:**

**USE Students;**

**Outcome:**

```
C:\windows\system32\cmd.exe - python cqlsh
cqlsh> use Students;
cqlsh:students>
```

# CRUD – Create Table

To create a column family or table by the name “student_info”.

```
CREATE TABLE Student_Info (
  RollNo int PRIMARY KEY,
  StudName text, DateofJoining
  timestamp, LastExamPercent
  double
  );
```

**Outcome:**

```
cqlsh> use Students;
cqlsh:students> CREATE TABLE Student_Info (
            ...   RollNo int PRIMARY Key,
            ...   StudName text,
            ...   DateofJoining timestamp,
            ...   LastExamPercent double
            ...   );
```

The table “student_info” gets created in the keyspace “students”.

Bi
Copyright 2015, WILEY INDIA PVT. LTD.

**Objective:** To lookup the names of all tables in the current keyspace, or in all the keyspaces if there is no current keyspace.

**Act:**

**DESCRIBE TABLES;**

**Outcome:**

```
cqlsh:students> describe tables;

student_info

cqlsh:students>
```

**Objective:** To describe the table "student_info" use the below command.

**Act:**

**DESCRIBE TABLE student_info;**

**Outcome:**

C:\windows\system32\cmd.exe - python cqlsh

```
cqlsh:students> describe table student_info;

CREATE TABLE student_info (
  rollno int,
  dateofjoining timestamp,
  lastexampercent double,
  studname text,
  PRIMARY KEY (rollno)
) WITH
  bloom_filter_fp_chance=0.010000 AND
  caching='KEYS_ONLY' AND
  comment='' AND
  dclocal_read_repair_chance=0.000000 AND
  gc_grace_seconds=864000 AND
  index_interval=128 AND
  read_repair_chance=0.100000 AND
  replicate_on_write='true' AND
  populate_io_cache_on_flush='false' AND
  default_time_to_live=0 AND
  speculative_retry='NONE' AND
  memtable_flush_period_in_ms=0 AND
  compaction={'class': 'SizeTieredCompactionStrategy'} AND
  compression={'sstable_compression': 'LZ4Compressor'};
```

# CRUD - Insert

```
BEGIN BATCH
INSERT INTO student_info (RollNo,StudName,DateofJoining,LastExamPercent)
VALUES (1,'Michael Storm','2012-03-29', 69.6)
INSERT INTO student_info (RollNo,StudName,DateofJoining,LastExamPercent)
VALUES (2,'Stephen Fox','2013-02-27', 72.5)
INSERT INTO student_info (RollNo,StudName,DateofJoining,LastExamPercent)
VALUES (3,'David Flemming','2014-04-12', 81.7)
INSERT INTO student_info (RollNo,StudName,DateofJoining,LastExamPercent)
VALUES (4,'Ian String','2012-05-11', 73.4)
APPLY BATCH;
```

## Outcome:

```
cqlsh:students> BEGIN BATCH
           ...   INSERT INTO student_info (RollNo,StudName,DateofJoining,LastExamPercent)
           ... VALUES (1,'Michael Storm','2012-03-29',69.6)
           ...     INSERT INTO student_info (RollNo,StudName,DateofJoining,LastExamPercent)
           ... VALUES (2,'Stephen Fox','2013-02-27',72.5)
           ...   INSERT INTO student_info (RollNo,StudName,DateofJoining,LastExamPercent)
           ...  VALUES (3,'David Flemming ','2014-04-12',81.7)
           ...   INSERT INTO student_info (RollNo,StudName,DateofJoining,LastExamPercent)
           ...  VALUES (4,'Ian String','2012-05-11',73.4)
           ... APPLY BATCH;
cqlsh:students>
```

# CRUD - Select

To view the data from the table “student_info”.

```
SELECT *
        FROM student_info;
```

The above select statement retrieves data from the “student_info” table.

**Outcome:**

```
cqlsh:students> select * from student_info;

 rollno | dateofjoining                          | lastexampercent | studname
--------+----------------------------------------+-----------------+-----------------
      1 | 2012-03-29 00:00:00India Standard Time |            69.6 |   Michael Storm
      2 | 2013-02-27 00:00:00India Standard Time |            72.5 |     Stephen Fox
      4 | 2012-05-11 00:00:00India Standard Time |            73.4 |      Ian String
      3 | 2014-04-12 00:00:00India Standard Time |            81.7 | David Flemming

(4 rows)

cqlsh:students>
```

**Objective:** To view only those records where the RollNo column either has a value 1 or 2 or 3.

**Act:**

```
SELECT *
        FROM student_info
                WHERE RollNo IN(1,2,3);
```

## Outcome:

```
cqlsh:students> Select * from student_info where RollNo IN(1,2,3);

 rollno | dateofjoining                          | lastexampercent | studname
--------+----------------------------------------+-----------------+-----------------
      1 | 2012-03-29 00:00:00India Standard Time |            69.6 |   Michael Storm
      2 | 2013-02-27 00:00:00India Standard Time |            72.5 |     Stephen Fox
      3 | 2014-04-12 00:00:00India Standard Time |            81.7 | David Flemming

(3 rows)

cqlsh:students>
```


Copyright 2015, WILEY INDIA PVT. LTD.

# CRUD – Create Index

To create an index on the “studname” column of the “student_info” column family use the following statement

**CREATE INDEX ON student_info(studname);**

To execute the query using the index defined on “studname” column use

```
SELECT *
        FROM student_info
                WHERE studname='Stephen Fox' ;
```

**Outcome:**

```
cqlsh:students> create index on student_info(studname);
cqlsh:students> select * from student_info where studname='Stephen Fox' ;

 rollno | dateofjoining                          | lastexampercent | studname
--------+----------------------------------------+-----------------+----------
      2 | 2013-02-27 00:00:00India Standard Time |            72.5 | Stephen Fox

(1 rows)

cqlsh:students>
```

**Objective:** Let us create another index on the "LastExamPercent" column of the "student_info" column family.

## Act:

```
CREATE INDEX ON student_info(LastExamPercent);
```

## Outcome:

```
cqlsh:students> create index on student_info(Last
cqlsh:students> select * from student_info where LastExamPercent = 81.7;

 rollno | dateofjoining                          | lastexampercent | studname
--------+----------------------------------------+-----------------+-----------------
      3 | 2014-04-12 00:00:00India Standard Time |            81.7 | David Flemming

(1 rows)
```

**Objective:** To specify the number of rows returned in the output using limit.

**Act:**

```
SELECT rollno, hobbies, language, lastexampercent
        FROM student_info LIMIT 2;
```

**Outcome:**

```
cqlsh:students> select rollno, hobbies, language, lastexampercent from student_info limit 2;

 rollno | hobbies                                | language           | lastexampercent
--------+----------------------------------------+--------------------+-----------------
      1 |                 {'Chess, Table Tennis'} | ['Hindi, English'] |            69.6
      4 | {'Lawn Tennis, Table Tennis, Golf'} | ['Hindi, English'] |            73.4

(2 rows)
```

**Objective:** To use column alias for the column "language" in the "student_info" table. We would like the column heading to be "knows language".

## Act:

```
SELECT rollno, language AS "knows language"
        FROM student_info;
```

## Outcome:

```
cqlsh:students> select rollno, language as "knows language" from student_info;

 rollno | knows language
--------+----------------------------
      1 |          ['Hindi, English']
      4 |          ['Hindi, English']
      3 | ['Hindi,English,French']

(3 rows)
```

# CRUD – Update

Objective: To update the value held in the “StudName” column of the “student_info” column family to “David Sheen” for the record where the RollNo column has value = 2.

***Note:*** An update updates one or more column values for a given row to the Cassandra table. It does not return anything.

**UPDATE student_info  SET StudName = 'David Sheen'  WHERE RollNo = 2;**

**Outcome:**

```
cqlsh:students> UPDATE student_info  SET StudName = 'David Sheen'  WHERE RollNo = 2;
cqlsh:students> select * from student_info where rollno = 2;

 rollno | dateofjoining                          | lastexampercent | studname
--------+----------------------------------------+-----------------+-----------
      2 | 2013-02-27 00:00:00India Standard Time |            72.5 | David Sheen

(1 rows)

cqlsh:students>
```

Big Data and Analytics by Seema Acharya and Subhashini Chellappan
Copyright 2015, WILEY INDIA PVT. LTD.

**Objective:** Let us try updating the value of a primary key column.

**Act:**

**UPDATE student_info SET rollno=6 WHERE rollno=3;**

**Outcome:**

```
cqlsh:students> update student_info set rollno=6 where rollno=3;
Bad Request: PRIMARY KEY part rollno found in SET part
cqlsh:students>
```

***Note:*** It does not allow update to a primary key column.

**Objective:** Updating more than one column of a row of Cassandra table.

**Step 1**: Before the update

```
cqlsh:students> select rollno, studname, lastexampercent from student_info where rollno=3;

 rollno | studname        | lastexampercent
--------+-----------------+-----------------
      3 | David Flemming  |            81.7
```

**Step 2:** Applying the update

```
cqlsh:students> select rollno, studname, lastexampercent from student_info where rollno=3;

 rollno | studname | lastexampercent
--------+----------+-----------------
      3 |  Samaira |              85
```

**Step 3:** After the update

```
cqlsh:students> DELETE LastExamPercent FROM student_info where RollNo=2;
cqlsh:students> select * from student_info where rollno = 2;

 rollno | dateofjoining                          | lastexampercent | studname
--------+----------------------------------------+-----------------+-----------
      2 | 2013-02-27 00:00:00India Standard Time |            null | David Sheen

(1 rows)

cqlsh:students>
```

# CRUD – Delete

To delete the column “LastExamPercent” from the “student_info” table for the record where the RollNo = 2.

***Note:*** Delete statement removes one or more columns from one or more rows of a Cassandra table or removes entire rows if no columns are specified.

**DELETE LastExamPercent FROM student_info WHERE RollNo=2;**

**Outcome:**

```
cqlsh:students> DELETE LastExamPercent FROM student_info where RollNo=2;
cqlsh:students> select * from student_info where rollno = 2;

 rollno | dateofjoining                          | lastexampercent | studname
--------+----------------------------------------+-----------------+-------------
      2 | 2013-02-27 00:00:00India Standard Time |            null | David Sheen

(1 rows)

cqlsh:students>
```

Big Data and Analytics by Seema Acharya and Subhashini Chellappan
Copyright 2015, WILEY INDIA PVT. LTD.

**Objective:** To delete a row (where RollNo = 2) from the table "student_info".

**Act:**

**DELETE FROM student_info WHERE RollNo=2;**

**Outcome:**

```
cqlsh:students> DELETE  FROM student_info where RollNo=2;
cqlsh:students> select * from student_info where rollno=2;

(0 rows)

cqlsh:students>
```

# Assignment

- Create a table “Project_details” with primary key as (project_id, project_name)
- Perform CRUD operations.