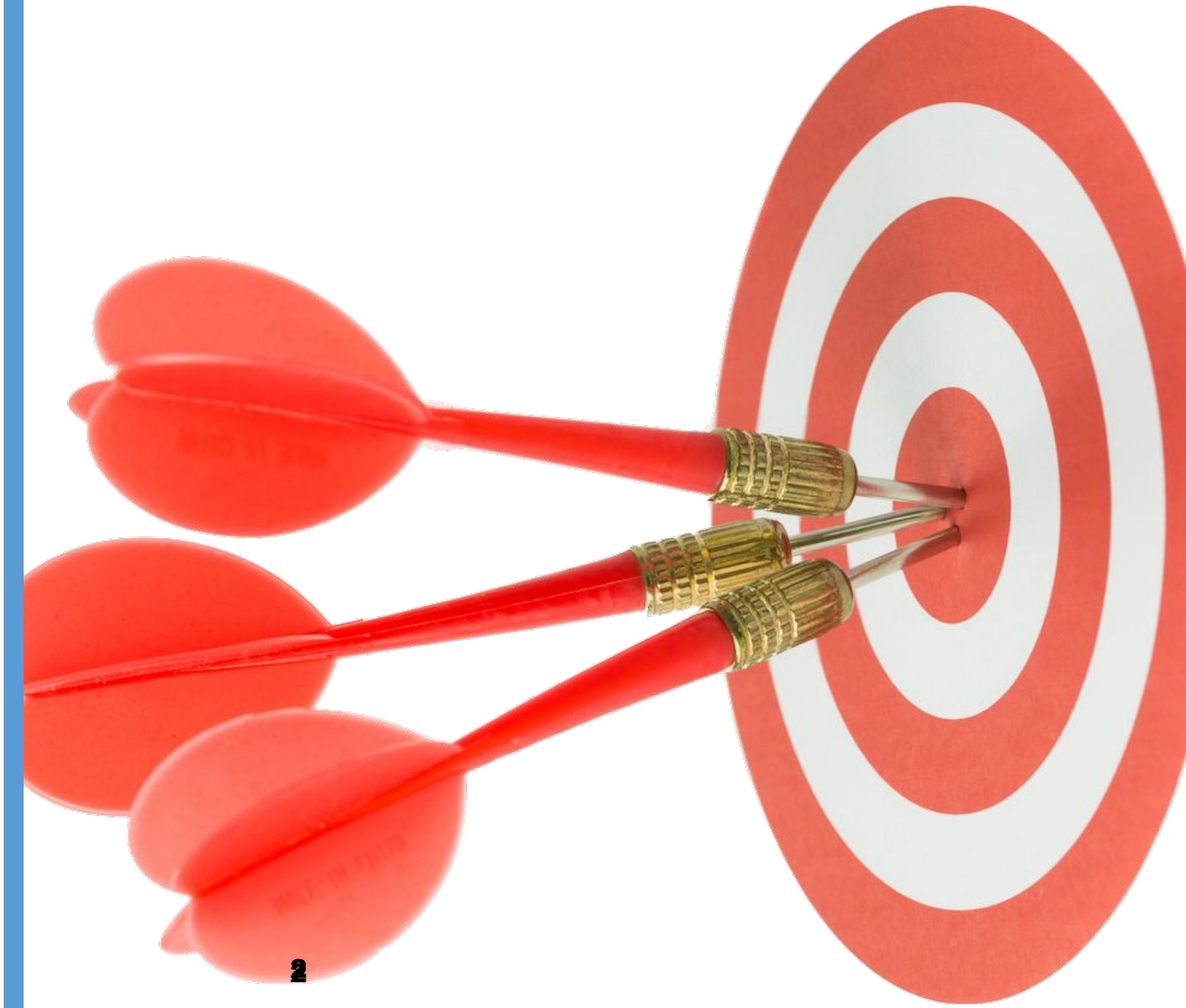
USING CQL

The Goal:

- Familiarize yourself with CQL Basics
  - Basic CRUD commands
    - aka Data Manipulation Language (DML)
  - Schema management commands
    - aka Data Definition Language (DDL)



CQL is Easy – It looks like SQL

- CRUD commands include:
  - CREATE – to create a keyspace or table
  - INSERT – to insert a row
  - UPDATE – to update columns for a row
  - DELETE – to delete columns or a complete row
  - SELECT – Retrieve rows

- Additional commands:
  - DESCRIBE – to inspect keyspaces, tables, etc.
  - TRUNCATE – to delete table contents
  - DROP – to remove keyspaces, tables, etc.

Basic CQL Terminology

- **Database** = a cluster of machines for hosting data storage
- **Keyspace** = container of tables (like a relational database)
- **Table** = a container of row/columns (like a relational table)
- **Row** = a set of column values that share the same primary key
- **Column** = a named value within a row
- **Primary key** = one or more columns that uniquely identify a row
- **Partition** = a set of rows – the atomic level of physical access
- **Partition key** = the piece of the primary key used to retrieve partition

Expected Common CQL Data Types

- **TEXT** – UTF-8 encoded string
- **INT** – 32-bit signed integer
- **FLOAT** – 32-bit floating point
- **UUID** – 128-bit number, like a GUID
- **TIMESTAMP** – 64-bit milliseconds since January 1, 1970
- **BOOLEAN, BLOB & many more** – check the docs

Creating a Keyspace

```
CREATE KEYSPACE killrvideo
    WITH REPLICATION = {
        'class' : 'NetworkTopologyStrategy',
        'DC1' : 3
    };
```

Creating a Keyspace

Keyspace Name

```
CREATE KEYSPACE killrvideo
    WITH REPLICATION = {
        'class' : 'NetworkTopologyStrategy',
        'DC1' : 3
    };
```

Creating a Keyspace

```
CREATE KEYSPACE killrvideo
    WITH REPLICATION = {
        'class' : 'NetworkTopologyStrategy',
        'DC1' : 3
    };
```

Replication Strategy

Creating a Keyspace

```
CREATE KEYSPACE killrvideo
    WITH REPLICATION = {
        'class' : 'NetworkTopologyStrategy',
        'DC1' : 3
    };
```

Replication Factor

Creating a Table - Syntax

```
CREATE TABLE killrvideo.user_videos (
    userid              uuid,
    added_date          timestamp,
    videoid             uuid,
    name                text,
    preview_image_location text,
    PRIMARY KEY ((userid), added_date, videoid)
);
```

Creating a Table - Syntax

Keyspace Name

```
CREATE TABLE killrvideo.user_videos (
    userid              uuid,
    added_date          timestamp,
    videoid             uuid,
    name                text,
    preview_image_location text,
    PRIMARY KEY ((userid), added_date, videoid)
);
```

Creating a Table - Syntax

Table Name

```
CREATE TABLE killrvideo.user_videos (
    userid              uuid,
    added_date          timestamp,
    videoid             uuid,
    name                text,
    preview_image_location text,
    PRIMARY KEY ((userid), added_date, videoid)
);
```

Creating a Table - Syntax

```
CREATE TABLE killrvideo.user_videos (
    userid                uuid,
    added_date            timestamp,
    videoid               uuid,
    name                  text,
    preview_image_location text,
    PRIMARY KEY ((userid), added_date, videoid)
);
```

Column Names

Creating a Table - Syntax

```
CREATE TABLE killrvideo.user_videos (
    userid                uuid,
    added_date            timestamp,
    videoid               uuid,
    name                  text,
    preview_image_location text,
    PRIMARY KEY ((userid), added_date, videoid)
);
```

Column Data Types

Creating a Table - Syntax

```
CREATE TABLE killrvideo.user_videos (
    userid                uuid,
    added_date            timestamp,
    videoid               uuid,
    name                  text,
    preview_image_location text,
    PRIMARY KEY ((userid), added_date, videoid)
);
```

Partition Key

Creating a Table - Syntax

```
CREATE TABLE killrvideo.user_videos (
    userid              uuid,
    added_date          timestamp,
    videoid             uuid,
    name                text,
    preview_image_location text,
    PRIMARY KEY ((userid), added_date, videoid)
);
```

Clustering Columns

Inserting a Row - Syntax

```
INSERT INTO killrvideo.user_videos
    (userid, added_date, videoid, name,
    preview_image_location )
    VALUES(
        uuid(),
        toTimestamp(now()),
        ef70b72f-7b11-4339-87dc-54fd5568afde,
        'Cat videos',
        'youtube.com/watch?v=WXZtCP64Yr8'
    );
```

Keyspace Name

```
INSERT INTO killrvideo.user_videos
    (userid, added_date, videoid, name,
    preview_image_location )
    VALUES(
        uuid(),
        toTimestamp(now()),
        ef70b72f-7b11-4339-87dc-54fd5568afde,
        'Cat videos',
        'youtube.com/watch?v=WXZtCP64Yr8'
    );
```

Table Name

```
INSERT INTO killrvideo.user_videos
    (userid, added_date, videoid, name,
    preview_image_location )
    VALUES(
        uuid(),
        toTimestamp(now()),
        ef70b72f-7b11-4339-87dc-54fd5568afde,
        'Cat videos',
        'youtube.com/watch?v=WXZtCP64Yr8'
    );
```

Column Names

```
INSERT INTO killrvideo.user_videos
    (userid, added_date, videoid, name,
    preview_image_location )
    VALUES(
        uuid(),
        toTimestamp(now()),
        ef70b72f-7b11-4339-87dc-54fd5568afde,
        'Cat videos',
        'youtube.com/watch?v=WXZtCP64Yr8'
    );
```

```
INSERT INTO killrvideo.user_videos
    (userid, added_date, videoid, name,
    preview_image_location )
    VALUES(
        uuid(),
        toTimestamp(now()),
        ef70b72f-7b11-4339-87dc-54fd5568afde,
        'Cat videos',
        'youtube.com/watch?v=WXZtCP64Yr8'
    );
```

Associated Column Values

Selecting a Row - Syntax

```
SELECT name, preview_image_location FROM
    killrvideo.user_videos
    WHERE userid = ef70b72f-7b11-4339-87dc-54fd5568afde;
```

Selecting a Row - Syntax

Column Name(s)

```
SELECT name, preview_image_location FROM
    killrvideo.user_videos
    WHERE userid = ef70b72f-7b11-4339-87dc-54fd5568afde;
```

Selecting a Row - Syntax

```
SELECT name, preview_image_location FROM
    killrvideo.user_videos
    WHERE userid = ef70b72f-7b11-4339-87dc-54fd5568afde;
```

Keyspace Name

Selecting a Row - Syntax

```
SELECT name, preview_image_location FROM
    killrvideo.user_videos
    WHERE userid = ef70b72f-7b11-4339-87dc-54fd5568afde;
```

Table Name

Selecting a Row - Syntax

```
SELECT name, preview_image_location FROM
    killrvideo.user_videos
    WHERE userid = ef70b72f-7b11-4339-87dc-54fd5568afde;
```

Primary Key Column

Selecting a Row - Syntax

```
SELECT name, preview_image_location FROM
    killrvideo.user_videos
    WHERE userid = ef70b72f-7b11-4339-87dc-54fd5568afde;
```

Primary Key Column Value(s)

Mutating a Row - Syntax

```
UPDATE killrvideo.videos
    SET description = 'Cute cat videos'
    WHERE videoid = ef70b72f-7b11-4339-87dc-54fd5568afde;
```

Mutating a Row - Syntax

Keyspace Name

```
UPDATE killrvideo.videos
    SET description = 'Cute cat videos'
    WHERE videoid = ef70b72f-7b11-4339-87dc-54fd5568afde;
```

Mutating a Row - Syntax

Table Name

```
UPDATE killrvideo.videos
    SET description = 'Cute cat videos'
    WHERE videoid = ef70b72f-7b11-4339-87dc-54fd5568afde;
```

Mutating a Row - Syntax

```
UPDATE killrvideo.videos
    SET description = 'Cute cat videos'
    WHERE videoid = ef70b72f-7b11-4339-87dc-54fd5568afde;
```

Column to Update

Mutating a Row - Syntax

```
UPDATE killrvideo.videos
    SET description = 'Cute cat videos'
    WHERE videoid = ef70b72f-7b11-4339-87dc-54fd5568afde;
```

New Value

Mutating a Row - Syntax

```
UPDATE killrvideo.videos
    SET description = 'Cute cat videos'
    WHERE videoid = ef70b72f-7b11-4339-87dc-54fd5568afde;
```

Primary Key Column Name(s)

Note: WHERE clause requires all columns/values of partition key

Mutating a Row - Syntax

```
UPDATE killrvideo.videos
    SET description = 'Cute cat videos'
    WHERE videoid = ef70b72f-7b11-4339-87dc-54fd5568afde;
```

Primary Key Column Value(s)

Deleting Columns - Syntax

```
DELETE name, description FROM
 killrvideo.videos
    WHERE videoid = ef70b72f-7b11-4339-87dc-54fd5568afde;
```

Deleting Columns - Syntax

Column Name(s)

```
DELETE name, description FROM
 killrvideo.videos
    WHERE videoid = ef70b72f-7b11-4339-87dc-54fd5568afde;
```

Deleting Columns - Syntax

Keyspace Name

```
DELETE name, description FROM
 killrvideo.videos
    WHERE videoid = ef70b72f-7b11-4339-87dc-54fd5568afde;
```

Deleting Columns - Syntax

```
DELETE name, description FROM
 killrvideo.videos
    WHERE videoid = ef70b72f-7b11-4339-87dc-54fd5568afde;
```

Table Name

Deleting Columns - Syntax

```
DELETE name, description FROM
 killrvideo.videos
    WHERE videoid = ef70b72f-7b11-4339-87dc-54fd5568afde;
```

Primary Key Column Name(s)

Deleting Columns - Syntax

```
DELETE name, description FROM
 killrvideo.videos
    WHERE videoid = ef70b72f-7b11-4339-87dc-54fd5568afde;
```

Primary Key Column Value(s)

cqlsh

Cassandra provides a prompt Cassandra query language shell (cqlsh) that allows users to communicate with it. Using this shell, you can execute Cassandra Query Language (CQL).

Using cqlsh, you can

- define a schema,
- insert data, and
- execute a query.

Options	Usage
cqlsh --help	Shows help topics about the options of **cqlsh** commands.
cqlsh --version	Provides the version of the cqlsh you are using.
cqlsh --color	Directs the shell to use colored output.
cqlsh --debug	Shows additional debugging information.
cqlsh --execute cql_statement	Directs the shell to accept and execute a CQL command.
cqlsh --file= **"file name"**	If you use this option, Cassandra executes the command in the given file and exits.
cqlsh --no-color	Directs Cassandra not to use colored output.
cqlsh -u **"user name"**	Using this option, you can authenticate a user. The default user name is: cassandra.
cqlsh-p **"pass word"**	Using this option, you can authenticate a user with a password. The defa password is: cassandra.

Commands Once Inside of CQLSH

- HELP − Displays help topics for all cqlsh commands.
- CAPTURE − Captures the output of a command and adds it to a file.
- CONSISTENCY − Shows the current consistency level, or sets a new consistency level.
- COPY − Copies data to and from Cassandra.
- DESCRIBE − Describes the current cluster of Cassandra and its objects.
- EXPAND − Expands the output of a query vertically.
- EXIT − Using this command, you can terminate cqlsh.
- PAGING − Enables or disables query paging.
- SHOW − Displays the details of current cqlsh session such as Cassandra version, host, or data type assumptions.
- SOURCE − Executes a file that contains CQL statements.
- TRACING − Enables or disables request tracing.

CQL Data Definition Commands

- CREATE KEYSPACE − Creates a KeySpace in Cassandra.
- USE − Connects to a created KeySpace.
- ALTER KEYSPACE − Changes the properties of a KeySpace.
- DROP KEYSPACE − Removes a KeySpace
- CREATE TABLE − Creates a table in a KeySpace.
- ALTER TABLE − Modifies the column properties of a table.
- DROP TABLE − Removes a table.
- TRUNCATE − Removes all the data from a table.
- CREATE INDEX − Defines a new index on a single column of a table.
- DROP INDEX − Deletes a named index.

CQL Data Manipulation Commands

- INSERT − Adds columns for a row in a table.
- UPDATE − Updates a column of a row.
- DELETE − Deletes data from a table.
- BATCH − Executes multiple DML statements at once.

CQL Clauses

- SELECT − This clause reads data from a table
- WHERE − The where clause is used along with select to read a specific data.
- ORDERBY − The orderby clause is used along with select to read a specific data in a specific order.

Help (Inside the Shell)

```
cqlsh> help

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

CQL help topics:
================
ALTER           CREATE_TABLE_OPTIONS       SELECT
ALTER_ADD       CREATE_TABLE_TYPES         SELECT_COLUMNFAMILY
ALTER_ALTER     CREATE_USER                SELECT_EXPR
ALTER_DROP      DELETE                     SELECT_LIMIT
ALTER_RENAME    DELETE_COLUMNS             SELECT_TABLE
```

Capture

Capture

This command captures the output of a command and adds it to a file. For example, take a look at the following code that captures the output to a file named Outputfile.

```
cqlsh> CAPTURE '/home/ernesto/CassandraProgs/Outputfile'
```

When we type any command in the terminal, the output will be captured by the file given. Given below is the command used and the snapshot of the output file.

```
cqlsh:tutorialspoint> select * from emp;
```

COPY

This command copies data to and from Cassandra to a file. Given below is an example to copy the table named emp to the file myfile.

```
cqlsh:tutorialspoint> COPY emp (emp_id, emp_city, emp_name,
emp phone,emp sal) TO ‘myfile’;
4 rows exported in 0.034 seconds.
```

Describe

Describe

This command describes the current cluster of Cassandra and its objects. The variants of this command are explained below.

Describe cluster − This command provides information about the cluster.

```
cqlsh:tutorialspoint> describe cluster;

Cluster: Test Cluster
Partitioner: Murmur3Partitioner

Range ownership:
                    -6583809122496445 57 [127.0.0.1]
                    -2833890865268921414 [127.0.0.1]
                    -6792159006375935836 [127.0.0.1]
```

Describe

Describe Keyspaces − This command lists all the keyspaces in a cluster. Given below is the usage of this command.

```
cqlsh:tutorialspoint> describe keyspaces;
```

```
system_traces system tp tutorialspoint
```

Describe tables − This command lists all the tables in a keyspace. Given below is the usage of this command.

```
cqlsh:tutorialspoint> describe tables;
emp
```

Using CQL

Exercise

Here's What We Just Did

- Used basic DML commands
  - **DESCRIBE KESPACES, KEYSPACE, TABLE**
  - **CREATE TABLE**
  - **TRUNCATE/DROP TABLE**
- Used basic CRUD commands
  - **INSERT**
  - **SELECT**
  - **UPDATE**
  - **DELETE columns, rows**

Now You Know …

- CQL Basics
  - **Basic CRUD commands (DML)**
  - **As well as schema commands (DDL)**