**CASSANDRA BACKUPS**

**WHAT ARE BACKUPS :**

Immutable SSTable files are used by Apache Cassandra to store data. The backup copies of the database data that are kept as SSTable files in the Apache Cassandra database are called backups. There are various uses for backups, some of which are as follows:

* To preserve a copy of the data
* To be able to restore a table in the event that a node, partition, or network failure causes the loss of table data
* For portability, to be able to move the SSTable files to another computer

**TYPES OF BACKUPS IN APACHE CASSANDRA :**

A snapshot is a hard link-created copy of the SSTable files for a table at a specific moment in time. Also kept is the DDL needed to build the table. Users can create snapshots, or they can be created automatically. If snapshots are made prior to each compaction, it is determined by the setting snapshot\_before\_compaction in the cassandra.yaml file. Snapshot\_before\_compaction is initially set to false. Auto\_snapshot can be set to true (default) in cassandra.yaml to automatically create snapshots prior to keyspace truncation or table deletion. The auto snapshots may cause truncates to be delayed, and there is another setting in cassandra.yaml that controls the length of time the coordinator should wait for truncates to finish. Cassandra waits 60 seconds by default for auto snapshots to finish.

When memtables are flushed to disk as SSTables, an incremental backup is a duplicate of the table's SSTable files made by a hard link. In order to shorten backup times and save disk space, incremental backups are often combined with snapshots. It is need to explicitly enable incremental backups using nodetool or the incremental\_backups parameter in cassandra.yaml as they are not enabled by default. When enabled, Cassandra builds a hard link in the backups/ subdirectory of the keyspace data for each SSTable that has been flushed or streamed locally. Additionally, system table incremental backups are produced.

**EXAMPLES FOR BACKUPS AND SNAPSHOTS :**

We will generate some sample data in this part that may be used to illustrate incremental backups and snapshots. A Cassandra cluster with three nodes has been employed. The keyspaces are first made. Next, table data is inserted and tables are created inside of a keyspace. Two keyspaces, cqlkeyspace and catalogkeyspace, each containing two tables, have been employed.

Create the keyspace cqlkeyspace:

CREATE KEYSPACE cqlkeyspace

WITH replication = {'class': 'SimpleStrategy', 'replication\_factor' : 3};

Create two tables t and t2 in the cqlkeyspace keyspace.

USE cqlkeyspace;

CREATE TABLE t (

id int,

k int,

v text,

PRIMARY KEY (id)

);

CREATE TABLE t2 (

id int,

k int,

v text,

PRIMARY KEY (id)

);

Add data to the tables:

INSERT INTO t (id, k, v) VALUES (0, 0, 'val0');

INSERT INTO t (id, k, v) VALUES (1, 1, 'val1');

INSERT INTO t2 (id, k, v) VALUES (0, 0, 'val0');

INSERT INTO t2 (id, k, v) VALUES (1, 1, 'val1');

INSERT INTO t2 (id, k, v) VALUES (2, 2, 'val2');

Query the table to list the data:

SELECT \* FROM t;

SELECT \* FROM t2;

results in

id | k | v

----+---+------

1 | 1 | val1

0 | 0 | val0

(2 rows)

id | k | v

----+---+------

1 | 1 | val1

0 | 0 | val0

2 | 2 | val2

(3 rows)

Create a second keyspace catalogkeyspace:

CREATE KEYSPACE catalogkeyspace

WITH replication = {'class': 'SimpleStrategy', 'replication\_factor' : 3};

Create two tables journal and magazine in catalogkeyspace:

USE catalogkeyspace;

CREATE TABLE journal (

id int,

name text,

publisher text,

PRIMARY KEY (id)

);

CREATE TABLE magazine (

id int,

name text,

publisher text,

PRIMARY KEY (id)

);

Add data to the tables:

INSERT INTO journal (id, name, publisher) VALUES (0, 'Apache Cassandra Magazine', 'Apache Cassandra');

INSERT INTO journal (id, name, publisher) VALUES (1, 'Couchbase Magazine', 'Couchbase');

INSERT INTO magazine (id, name, publisher) VALUES (0, 'Apache Cassandra Magazine', 'Apache Cassandra');

INSERT INTO magazine (id, name, publisher) VALUES (1, 'Couchbase Magazine', 'Couchbase');

Query the tables to list the data:

SELECT \* FROM catalogkeyspace.journal;

SELECT \* FROM catalogkeyspace.magazine;

results in

id | name | publisher

----+---------------------------+------------------

1 | Couchbase Magazine | Couchbase

0 | Apache Cassandra Magazine | Apache Cassandra

(2 rows)

id | name | publisher

----+---------------------------+------------------

1 | Couchbase Magazine | Couchbase

0 | Apache Cassandra Magazine | Apache Cassandra

(2 rows)

Snapshots

In this section, we demonstrate creating snapshots. The command used to create a snapshot is nodetool snapshot with the usage:

$ nodetool help snapshot

results in

NAME

nodetool snapshot - Take a snapshot of specified keyspaces or a snapshot

of the specified table

SYNOPSIS

nodetool [(-h <host> | --host <host>)] [(-p <port> | --port <port>)]

[(-pp | --print-port)] [(-pw <password> | --password <password>)]

[(-pwf <passwordFilePath> | --password-file <passwordFilePath>)]

[(-u <username> | --username <username>)] snapshot

[(-cf <table> | --column-family <table> | --table <table>)]

[(-kt <ktlist> | --kt-list <ktlist> | -kc <ktlist> | --kc.list <ktlist>)]

[(-sf | --skip-flush)] [(-t <tag> | --tag <tag>)] [--] [<keyspaces...>]

OPTIONS

-cf <table>, --column-family <table>, --table <table>

The table name (you must specify one and only one keyspace for using

this option)

-h <host>, --host <host>

Node hostname or ip address

-kt <ktlist>, --kt-list <ktlist>, -kc <ktlist>, --kc.list <ktlist>

The list of Keyspace.table to take snapshot.(you must not specify

only keyspace)

-p <port>, --port <port>

Remote jmx agent port number

-pp, --print-port

Operate in 4.0 mode with hosts disambiguated by port number

-pw <password>, --password <password>

Remote jmx agent password

-pwf <passwordFilePath>, --password-file <passwordFilePath>

Path to the JMX password file

-sf, --skip-flush

Do not flush memtables before snapshotting (snapshot will not

contain unflushed data)

-t <tag>, --tag <tag>

The name of the snapshot

-u <username>, --username <username>

Remote jmx agent username

--

This option can be used to separate command-line options from the

list of argument, (useful when arguments might be mistaken for

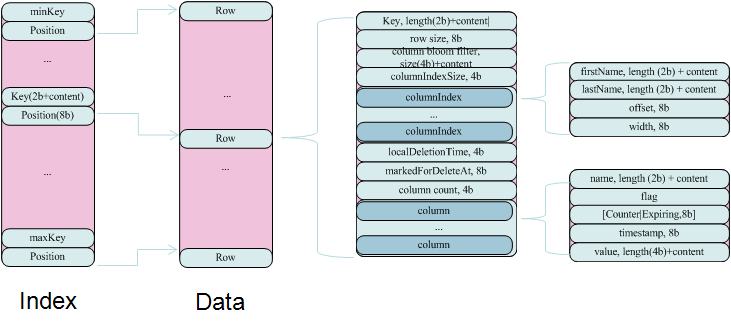
command-line options

[<keyspaces...>]

List of keyspaces. By default, all keyspaces

**Data Directory Structure :**

The keyspace and table folders, along with the data files contained therein, make up the directory structure of Cassandra data. The table directory also contains backup and snapshot directories, which are used to store backups and snapshots for a certain table, respectively.



Sequential writes to Cassandra nodes first impact the CommitLog. (After that, Cassandra stores values in in-memory data structures called Memtables that are particular to a column family. Whenever a preset threshold is surpassed, the Memtables are flushed to disk. (1, memtable datasize). 2, the number of items reaches a predetermined limit, and 3, a memtable's lifetime ends.))

Every keyspace has a subdirectory in the data folder. Three types of files are contained in each subfolder:

Information files: An file containing key-value string pairs that have been sorted by keys is known as an SSTable (a term taken from Google's nomenclature).

File index: Pairs (key, offset) that point to data files

Bloom filter: every key within the data set