

Bases de données spécialisées

A CQL(SH) primer



2024-2025

Giovanni Bernardi, gioXYZirif.fr

<http://www.irif.fr/~gio/index.xhtml>

Teams

Cassy
AMIRA_AMIRA_DJEDOU
Big_Dark_Dolphins_members
group_GoF
JONS_group_members
Senku_Group
SexyMembres
ZYAN_group
AIR
limonad
cogito

Warm-up

Why weak consistency ?

Why weak consistency ?

Theorem (CAP)

It is impossible in the asynchronous network model to implement a read/write data object that guarantees the following properties:

- ▶ *Availability*
- ▶ *Atomic Consistency*

in all fair executions (including those in which messages are lost).

published first at PODC 2002

- ▶ China's secret grip on undersea cables . . . 27 Nov 2024, Regtech Times
- ▶ The 'Western underbelly' leaving Britain exposed to enemy states 30 Nov 2024, The Telegraph
- ▶ What happens when undersea internet cables snap The NY Times, 30 Nov 2024
- ▶ Submarine cable resilience board announced on same day maybe-cut-by-China Baltic cable repaired The Register, 2 Dec 2024

Warm-up



is designed to offer which properties out of

Strong Consistency, Availability, Partition tolerance

?

Warm-up



is designed to offer which properties out of

Strong Consistency, **Availability**, **Partition tolerance**

Unique snitch and partitioner per cluster

```
user@host: $ nodetool describcluster
```

```
Cluster Information:
```

```
Name: Punk_is_not_dead
```

```
Snitch: org.apache.cassandra.locator.GossipingPropertyFileSnitch
```

```
DynamicEndPointSnitch: enabled
```

```
Partitioner: org.apache.cassandra.dht.Murmur3Partitioner
```

```
Schema versions:
```

```
5b547ad6-544b-3208-b432-9a9778c9b9da: [127.0.0.1]
```

Unique snitch and partitioner per cluster

```
user@host: $ nodetool describcluster
```

```
Cluster Information:
```

```
Name: Punk_is_not_dead
```

```
Snitch: org.apache.cassandra.locator.GossipingPropertyFileSnitch
```

```
DynamicEndPointSnitch: enabled
```

```
Partitioner: org.apache.cassandra.dht.Murmur3Partitioner
```

```
Schema versions:
```

```
5b547ad6-544b-3208-b432-9a9778c9b9da: [127.0.0.1]
```

What is the purpose of the snitch ?

Unique snitch and partitioner per cluster

```
user@host: $ nodetool describcluster
Cluster Information:
Name: Punk_is_not_dead
Snitch: org.apache.cassandra.locator.GossipingPropertyFileSnitch
DynamicEndPointSnitch: enabled
Partitioner: org.apache.cassandra.dht.Murmur3Partitioner
Schema versions:
    5b547ad6-544b-3208-b432-9a9778c9b9da: [127.0.0.1]
```

What is the purpose of the snitch ?

- ▶ Keeps track of network topology to route requests efficiently
- ▶ Spread replicas to avoid “correlated failures”

Thus far

- ▶ CAP theorem
- ▶ Anomalies and weak consistency
 - Write Skew example in PostgreSQL
- ▶ Cassandra
 - outline of the distributed system, code snippets
 - installation, configuration
 - sketch of node management

Questions we will address

Server side

How to set-up and run a cassandra node ?

Last week

Client side

How to interact with the storage system ?

Server side

Client side



postgresql



psql



cassandra



cqlsh



nodetool

How to connect to a Cassandra server ?

cqlsh

synopsis simplified:

```
$ cqlsh [addr_ip] [port] [-u username] [-p password]  
        [-e query]
```

basic usage to connect to localhost

```
$ cqlsh
```

options can be given in ~/.cassandra/cqlshrc

SHOW

Directive to make `cqlsh` display information about

VERSION: version of `cqlsh`, Cassandra server, CQL specification, and native protocol

Usually displayed as first line at connection time

HOST: the node details for host of the `cqlsh` session

SESSION `tr_session_id`: activity details for a query identified by `tr_session_id`. Session ids are store in the table `system_traces.sessions`.

TRACING ON | OFF

Directive to make `cqlsh` (en|dis)able storage of metadata for all CQL statements in the current session.

Try executing the following queries

```
tracing off
```

```
select * from keyspace_name.table_name;
```

```
tracing on
```

```
select * from keyspace_name.table_name;
```

where `keyspace_name`, `table_name` are names of entities that exist in the cluster.

```
try running: select * from system_traces.sessions;
```

- ▶ Useful to troubleshoot performance issues
- ▶ Documentation

PAGING ON | OFF

Makes `cqlsh` (en|dis)able displaying results in chunks of 100 line.

Try executing the following queries

```
paging off
select * from system.compaction_history;
paging on
select * from system.compaction_history;
```

► Documentation

CLEAR

CTRL + I

USE `keyspace_name`

Makes the session “connect” to the given keyspace.

`keyspace_name`: This parameter is the name of the keyspace to connect to. Use quotes (“`keyspace_name`”) for case-sensitive syntax.

It is the analogous of the instruction `\c dbname` in `psql`.

DESCRIBE

Makes `cqlsh` display information about

TABLES: Every keyspace name, and for each keyspace the names of all the tables it contains.

TABLE `ks_name.table_name`: The schema of the table with name `table_name` in the keyspace with name `ks_name`

Try executing the following queries

```
describe tables
```

```
describe system_traces.sessions
```

DESCRIBE

Directive to make `cqlsh` display information about

CLUSTER: the cluster name, partitioner, and token ranges.

SCHEMA: the schema (i.e. the definitions) of all non-system keyspaces.

FULL SCHEMA: also the definitions of the system keyspaces.
`system`, `system.system_traces`, `system_distributed`, ...

CONSISTENCY

Directive to make `cqlsh` show or modify the consistency level *within one session!*

`no options`: Display the current consistency level

`consistency_level` : Set the consistency level to `consistency_level1`. As expected, the consistency level determines data availability versus data accuracy for transactions during a session.

Handy manner to use cqlsh

```
$ cqlsh -e "your_query"
```

Useful to write scripts that query the storage system.

Example

```
$cqlsh -e "describe keyspaces"
```

Of course the point of writing scripts is to manipulate the output of queries

```
cqlsh -e "select * from system_schema.keyspaces;" | grep SimpleStrategy
```

Python driver

```
pip3 install cassandra-driver
```


Cassandra Query Language

looks similar to SQL

but it behaves differently !



KEYSPACE

Operations: CREATE, **USE** *not sql*, ALERT, DROP

Example

```
CREATE KEYSPACE [IF NOT EXISTS] demo
WITH replication = { 'class': 'NetworkTopologyStrategy',
                    'datacenter1' : 3
                    'datacenter2' : 2 };
```

► Official documentation

Replication class

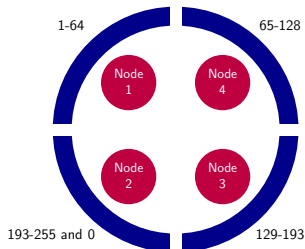
SimpleStrategy

- ▶ [...] returns the RF nodes that lie right next to each other on the ring.
- ▶ “generally not a wise choice for production because it [...] does not respect datacenter layouts and can lead to wildly varying query latency.”

NetworkTopologyStrategy

- ▶ “allows to set the replication factor independently for each data-center.”
- ▶ `cassandra-rackdc.properties`

SimpleStrategy

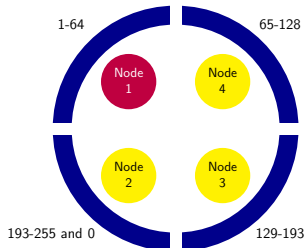


A client asks Cassandra to write the row (69000, Lyon) in a keyspace with replication factor 3.

The server

1. computes the token, $hash(69000) = 67$
2. finds the node responsible for token 67, Node 4
3. writes (69000, Lyon) in Node 4
4. and in Node 3 and Node 2

SimpleStrategy

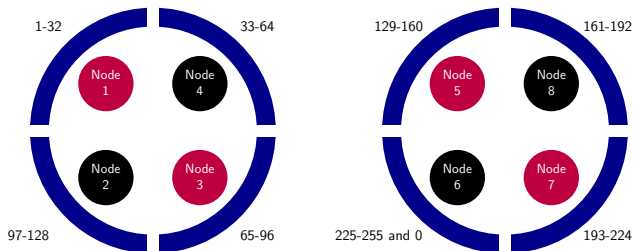


A client asks Cassandra to write the row (69000, Lyon) in a keyspace with replication factor 3.

The server

1. computes the token, $hash(69000) = 67$
2. finds the node responsible for token 67, Node 4
3. writes (69000, Lyon) in Node 4
4. and in Node 3 and Node 2

NetworkTopologyStrategy

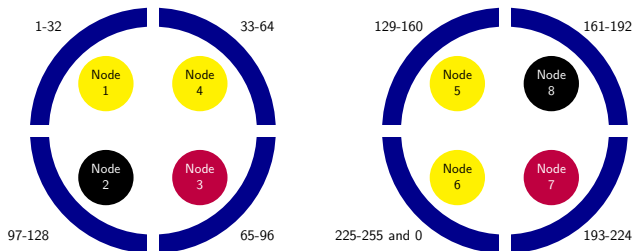


A client asks Cassandra to write the row (69000, Lyon) in a keyspace with replication factor 4 (2 + 2).

The server

1. computes the token, $hash(69000) = 67$
2. finds the node responsible for token 67, Node 4
3. writes (69000, Lyon) in Node 4
4. and in enough nodes in different dc/racks (example: 1, 5, 6)

NetworkTopologyStrategy



A client asks Cassandra to write the row (69000, Lyon) in a keyspace with replication factor 4 (2 + 2).

The server

1. computes the token, $hash(69000) = 67$
2. finds the node responsible for token 67, Node 4
3. writes (69000, Lyon) in Node 4
4. and in enough nodes in different dc/racks (example: 1, 5, 6)

A first keyspace, and troubleshooting

or how to avoid the error

`cassandra.cluster.NoHostAvailable:`

`ex0-dummy-cql.py`

TABLE

Operations: CREATE, ALTER, **TRUNCATE** *not sql*, DROP

Example

```
CREATE TABLE [IF NOT EXISTS] mytable (  
    cname1 type1,  
    :  
    cnameN typeN,  
    PRIMARY KEY (cnameA, ..., cnameM))  
                partitionK  
WITH comment='2 buffalo buffalo buffalo buffalo'
```

- ▶ First column in PK identifies **partition**
- ▶ In the primary key it is crucial to have columns to find a row within a partition *clustering columns*
- ▶ Official documentation

TABLE

```
CREATE TABLE simplepk (  
    id int,  
    title text,  
    band text,  
    PRIMARY KEY (id, title, band))
```

```
CREATE TABLE compositepk (  
    id int,  
    title text,  
    band text,  
    PRIMARY KEY ((id, title), band))
```

Different syntax, different meaning:

- ▶ **id** partition key; clustering columns: title, band
- ▶ **(id, title)** composite partition key; clustering column: band

TABLE

Example

```
TRUNCATE TABLE dummy;
```

- ▶ TRUNCATE deletes all the data in a table, but not the table
- ▶ Alternative to DROP TABLE

INSERT

```
insert into  
tablename(cname1,cname2,cname3, ...)  
values (v1,v2,v3, ...);
```

Example

```
insert into dummy(id,num) values (0,10);  
insert into dummy(id) values (0);
```

- ▶ requires the entire primary key, but not the other values
- ▶ missing values are set to null
- ▶ **it is not SQL**: if the same primary key is used in two inserts the row is overwritten!
- ▶ Official documentation

Semantics of SQL and CQL inserts are different!

`ex0-dummy-sql.sh`

`ex0-dummy-cql.py`

UPDATE

Example

```
update dummy set num=54 where id=9;
```

```
update dummy set num=1 where id=33;
```

UPSERT

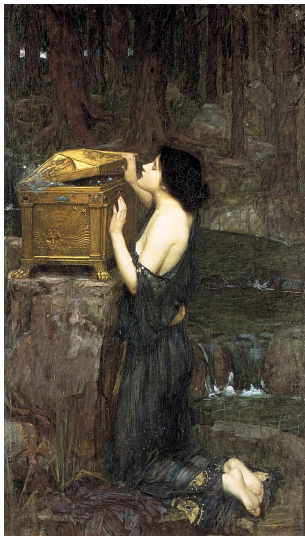
no fundamental difference between insert and update operations

- ▶ update a row and the primary key does not exist
⇒ Cassandra creates the row with exceptions
- ▶ insert a row w/ duplicate primary key
⇒ Cassandra replaces the row
- ▶ Official documentation

ex1-update.py

Questions ?

Questions ?



J. W. Waterhouse, 1896

Consider the following table schema

```
create table  
dots(id int, color text, primary key (id, color))  
with  
clustering order by (color desc)
```

- ▶ The partition key is id
- ▶ **Clustering order:** The order of the rows within each partition depends on color

SELECT

Example

```
select table_name,comment from system_schema.tables;
```



```
SELECT * FROM dots;
```



```
SELECT * FROM dots ORDER BY id DESC;
```



```
SELECT * FROM dots WHERE color='black';
```



► [Official documentation on SELECT](#)

Again, SQL vs CQL semantics

`ex2-orderby-sql.sh`

`ex2-orderby-cql.py`

`ex2-orderby-goodschema.py`

SELECT

Order by

- ▶ The order in which partitioned rows are returned depends on the order of the tokens and not on the key values themselves. (post)
- ▶ Clustering column(s) determine the data's sort order only **within a partition.**
- ▶ ORDER BY not necessary if order is in table definition.

ORDER BY can be used only

- ▶ on clustering columns (i.e. not the partition key)
- ▶ if the partition key is restricted via IN or EQ
but check paging ...

ex3-filtering-cql.py

SELECT

Where

A WHERE clause can use only the columns that

- ▶ are part of primary key, or
- ▶ have a secondary index on them not discussed so far...

Rationale: if a WHERE clause

- ▶ contains a partition key, Cassandra knows which node to contact: response time is constant wrt the number of nodes.
- ▶ does not contain a partition key, Cassandra contacts each node, and it scans the complete dataset
 - execution time proportional to the total amount of data stored in the referenced table,
 - performance is likely to get worse as you add nodes to the cluster.

in general

ALLOW FILTERING signals poor data modelling

take home message

Rethink your data modelling !

ex4-index.py

DELETE

Example

```
delete name from students_good where sid=1164220;
```



```
delete name from students_good  
      where uni='P7' and sid=1164220;
```



```
delete name from students_good where uni='P7';
```



```
delete name from students_good;
```



- ▶ all deletions within the same partition are applied atomically and in isolation
- ▶ DELETE can be conditional through an IF clause, but internally Paxos will be used, worsening the performances.
- ▶ Official documentation on DELETE

BATCH

Way to group together multiple upserts and deletes.

Example

```
BEGIN BATCH  
modif1  
modif2  
⋮  
modifn  
APPLY BATCH;
```

What for ?

- ▶ Saves network round-trips if contains multiple UPDATE
- ▶ Updates in a batch performed in isolation within partitions
- ▶ Eventually all the modifs will take place, or none will
- ▶ Official documentation on BATCH

Batches are *not* analogues of SQL transactions.

Transactions ?

Cassandra supports a form of lightweight transactions.

One such transaction

- ▶ might block other lightweight transactions from occurring
- ▶ does not stop normal read and write operations from occurring
- ▶ if mixed with normal operations might generate errors

due to the usage of different timestamp systems

How to create a lightweight transaction ?

Using IF conditions in INSERT, UPDATE, DELETE

- ▶ Implementation via Paxos out of scope
- ▶ Guaranteed consistency: linearisability out of scope
- ▶ Further details here and here

Generalized Consensus and Paxos

Paxos made simple

SwiftPaxos: Fast Geo-Replicated State Machines

Other differences with SQL

- ▶ JOIN:

JOINS must be computed by the application.

- ▶ FOREIGN KEYS:

Referential integrity must be guaranteed by the application.

Keep in mind you can BATCH updates.

- ▶ tokens

Tokens

`ex5-token.py`

How to find which machine is in charge of a token ?

```
$ nodetool [-p port] [-h host] getendpoints keyspace  
table id
```

```
user@host: $ nodetool getendpoints demo dummy 73  
192.168.0.101  
192.168.0.104  
192.168.0.103
```

ex6-fk-join.py

ex6-music.py

End of Part III

Geo-replicated databases
and
weak consistency

Weak consistency in Neo4j

- ▶ **read-committed** isolation level
- ▶ Neo4j allows **lost updates**

To ensure deterministic behaviour [...], it is necessary to explicitly acquire a write lock on the node in question. In Cypher there is no explicit support for this, but it is possible to work around this limitation by writing to a temporary property.

That's the story.
Thank you for the attention



Questions?