Cassandra Query Language

Berlin Hackathon
June 10, 2011

Eric Evans
eevans@rackspace.com
@jericevans
http://blog.sym-link.com

Goals (mine anyway)

- Reasonable stability guarantees.
- Thinner clients
- Simpler, easier, to use
- Fewer external dependencies
- Confuse.
- Provoke.
- Troll.

http://caqel.deadcafe.org

It's SQL, except where it isn't

- Subset that makes sense
  - No joins
  - No subselects
- SELECT “projection”
  - Supports slicing (`SELECT a..b FROM...`)
  - … with limits
  - … and ordering
- Different semantics for `SELECT count()`
- `UPDATE vs INSERT semantics`
- `Etc, etc`

Terms

- Inferred from comparator / validator.
- Strings are valid identifiers, or any quoted value (e.g. foo, '3atme')
- Numbers as numeric literals, (or quoted).
- UUIDs in hex-notation (i.e. e38629da-aeaf-44a3-8f59-e0de19f69b3a).
- Binary as hexidecimal strings.

Types (comparator/validator)

- AsciiType → ascii
- BytesType → bytea
- IntegerType → varint
- LongType → int, bigint
- UTF8Type → text, varchar
- UUIDType → uuid

Coming Soon...

- `ALTER`
- `DESCRIBE`
- Compound columns (think supercolumns, but better).
- Prepared statements.
- Named keys.
- Custom protocol (No More Thrift).

Query

- `execute_cql_query(query, compression) → CqlResult`
- Query argument is bytes/binary.
- Compression argument is one of GZIP or NONE, (defaults to GZIP).
- Exactly one statement per request!
- Can raise...
  - InvalidRequestException
  - UnavailableException
  - TimedOutException
  - SchemaDisagreementException

Response

```
enum CqlResultType {
    ROWS = 1,
    VOID = 2,
    INT  = 3
}

struct CqlResult {
    1: required CqlResultType type,
    2: optional list<CqlRow> rows,
    3: optional i32 num
}

/** Row returned from a CQL query */
struct CqlRow {
    1: required binary key,
    2: required list<Column> columns
}
```

Drivers

- Minimal / low-level
- Do not leak any Thrift types!
- Be idiomatic
- client-dev@cassandra.apache.org
- APL 2.0 (w/ NOTICE, etc)
- Tests
- ...

The End