Apache Cassandra™ Query Language (CQL) SPECIFICATION 3.2.0

REFERENCE GUIDE - P.1

DATASTAX

ALTER KEYSPACE

```
ALTER ( KEYSPACE | SCHEMA ) keyspace_name
WITH REPLICATION = map
| ( WITH DURABLE_WRITES = ( true | false ))
AND ( DURABLE_WRITES = ( true | false ))
```

map is a map collection, a JSON-style array of literals, such as 'class' : 'SimpleStrategy' or 'NetworkTopologyStrategy' in this format:
`{ literal : literal, literal : literal ... }`

ALTER TABLE

```
ALTER TABLE keyspace_name.table_name instruction
```

instruction is:

```
  ALTER column_name TYPE cql_type
| ( ADD column_name cql_type )
| ( DROP column_name )
| ( RENAME column_name TO column_name )
| ( WITH property AND property ... )
```

cql_type is compatible with the original type and is a CQL type other than a collection or counter. Exceptions: ADD supports a collection type and also, if the table is a counter, a counter type.

property is a CQL table property (p. 4) and value, such as read_repair_chance = .5.

ALTER TYPE

```
ALTER TYPE name instruction
```

name is an identifier of a user-defined type.
field_name is an arbitrary identifier for the field.
new_type is an identifier other than the reserved type names.

ALTER USER

```
ALTER USER user_name WITH PASSWORD 'password'
( NOSUPERUSER | SUPERUSER )
```

BATCH

```
BEGIN ( UNLOGGED ) BATCH
  USING TIMESTAMP timestamp
  dml_statement;
  dml_statement;
  ...
APPLY BATCH;
```

dml_statement is:

```
INSERT
UPDATE
DELETE
```

CREATE INDEX

```
CREATE CUSTOM INDEX IF NOT EXISTS index_name
ON keyspace_name.table_name ( KEYS (column_name) )
( USING class_name ) ( WITH OPTIONS = map )
```

Restrictions:
USING class_name is allowed only if CUSTOM is used and class_name is a string literal containing a java class name.

index_name is an identifier, enclosed or not enclosed in double quotation marks, excluding reserved words.

map is described in ALTER KEYSPACE.

CREATE KEYSPACE

```
CREATE ( KEYSPACE | SCHEMA ) IF NOT EXISTS keyspace_name
WITH REPLICATION = map
AND DURABLE_WRITES = ( true | false )
```

map is described in ALTER KEYSPACE.

CREATE TABLE

```
CREATE TABLE IF NOT EXISTS keyspace_name.table_name
( column_definition, column_definition, ...)
WITH property AND property ...
```

column_definition is:

```
column_name cql_type STATIC PRIMARY KEY
| column_name frozen<tuple<tuple_type>
   tuple<tuple_type> ... > PRIMARY KEY
| column_name frozen<user-defined_type> PRIMARY KEY
| ( PRIMARY KEY ( partition_key ) )
```

Restrictions:
- There should always be exactly one primary key definition.
- cql_type of the primary key must be a CQL type or user-defined type.
- cql_type of a collection uses this syntax:

```
 LIST<cql_type>
 | ( SET<cql_type> ) | ( MAP<cql_type, cql_type> )
```

PRIMARY KEY is:

```
column_name
| ( column_name1, column_name2, column_name3 ...)
| ((column_name4, column_name5), column_name6, column_name7 ...)
```

column_name1 is the partition key.
column_name2, column_name3 ... are clustering columns.
column_name4', column_name5 are partitioning keys.
column_name6, column_name7 ... are clustering columns.

(continued)

Apache Cassandra™ Query Language (CQL) SPECIFICATION 3.2.0

REFERENCE GUIDE - P.2

DATASTAX

CREATE TABLE (continued)

`property` is a CQL table storage property or one of these directives:

```
COMPACT STORAGE
| ( CLUSTERING ORDER BY (clustering_column
  ( ASC | DESC ), ...)  )
```

CREATE TRIGGER

```
CREATE TRIGGER trigger_name ON table_name
USING 'java_class'
```

CREATE TYPE

```
CREATE TYPE IF NOT EXISTS keyspace.type_name
(field, field, . . . )
```

`type_name` is a type identifier other than reserved type names.
`field` is: `field_name type`
`field_name` is an arbitrary identifier for the field.
`type` is a CQL collection or non-collection type other than a counter type.

CREATE USER

```
CREATE USER IF NOT EXISTS user_name
WITH PASSWORD 'password'
NOSUPERUSER | SUPERUSER
```

DELETE

```
DELETE column_name, ... |  ( column_name term )
FROM keyspace_name.table_name
USING TIMESTAMP integer
WHERE row_specification
( IF ( EXISTS | ( condition ( AND condition ) ... ) ) )
```

term is:

```
[ list_position ] | key_value
```

row_specification is one of:

- `primary_key_name = key_value`
- `primary_key_name IN ( key_value, key_value, ...)`

condition is:

```
| column_name [ list_position ] = key_value
column_name = key_value
```

DROP INDEX

```
DROP INDEX IF EXISTS index_name
```

DROP KEYSPACE

```
DROP ( KEYSPACE | SCHEMA ) IF EXISTS keyspace_name
```

DROP TABLE

```
DROP TABLE IF EXISTS keyspace_name.table_name
```

DROP TRIGGER

```
DROP TRIGGER trigger_name ON table_name
```

DROP TYPE

```
DROP TYPE IF EXISTS type_name
```

`type_name` is the name of a user-defined type.

DROP USER

```
DROP USER IF EXISTS user_name
```

GRANT

```
GRANT permission_name PERMISSION
| ( GRANT ALL PERMISSIONS ) ON resource TO user_name
```

`permission_name` is one of:

- ALTER
- AUTHORIZE
- CREATE
- DROP
- MODIFY
- SELECT

`resource` is one of:

- ALL KEYSPACES
- KEYSPACE keyspace_name
- TABLE *keyspace_name*.table_name

INSERT

```
INSERT INTO keyspace_name.table_name
( column_name, column_name ... )
VALUES ( value, value, ... ) IF NOT EXISTS
USING option AND option
```

`value` is one of:

- a literal
- a set

```
{ literal, literal, ... }
```

- a list

```
[ literal, literal, ... ]
```

- a map collection, described in ALTER KEYSPACE

`option` is one of:

- TIMESTAMP microseconds
- TTL seconds

LIST PERMISSIONS

```
LIST permission_name PERMISSION
| ( LIST ALL PERMISSIONS )
  ON resource OF user_name NORECURSIVE
```

`permission_name` and *resource* are shown in GRANT.

LIST USERS

```
LIST USERS
```

REVOKE

```
REVOKE ( permission_name PERMISSION )
| ( REVOKE ALL PERMISSIONS )
ON resource FROM user_name
```

permission_name and *resource* are shown in GRANT.

SELECT

```
SELECT select_expression
FROM keyspace_name.table_name
WHERE relation AND relation ...
ORDER BY (clustering_column ( ASC | DESC ), ... )
LIMIT n
ALLOW FILTERING
```

select_expression is:

```
selection_list | ( COUNT ( * | 1 ) )
```

selection_list is:

```
  selector AS alias, selector AS alias, ... | *
```

alias is an alias for a column.

selector is:

```
column_name
| ( WRITETIME (column_name) )
| ( TTL (column_name) )
| ( function (selector, selector, ...) )
```

function is a timeuuid function, a token function, or a blob conversion function.

relation is:

```
column_name op term
| ( column_name, column_name, ... ) op term-tuple
| column_name IN ( term, ( term, ... ) )
| column_name, column_name, ... ) IN ( term-tuple,
   ( term-tuple ... ) )
| TOKEN ( column_name, ... ) op ( term )
```

op is:

```
= | < | > | <= | > | = | CONTAINS | CONTAINS KEY
```

term-tuple is:

```
( term, term, ... )
```

term is a constant, such as a true or false, a bind marker (?), or a set, list, or map.

TRUNCATE

```
TRUNCATE keyspace_name.table_name
```

UPDATE

```
UPDATE keyspace_name.table_name
USING option AND option
SET assignment, assignment ...
WHERE row_specification
IF column_name = literal
  AND column_name = literal ...
```

option is one of:

- TIMESTAMP microseconds
- TTL seconds

assignment is one of:

- column_name = value
- set_or_list_item = set_or_list_item (+ | -) ...
- map_name = map_name (+ | -) ...
- column_name [term] = value
- counter_column_name = counter_column_name (+ | -) integer

set, list, map are defined in INSERT.

term is:

```
[ list_position ] | key_value
```

row_specification is one of:

- primary_key_name = key_value
- primary_key_name IN (*key_value* ,...)

USE

```
USE keyspace_name
```

CQL Table Properties

bloom_filter_fp_chance
Desired false-positive probability for SSTable Bloom filters. Default 0.01 for SizeTieredCompactionStrategy, 0.1 for LeveledCompactionStrategy

caching
Cache memory settings. Values: For keys, ALL or NONE; Default All. For rows_per_partition, number of CQL rows, NONE, or ALL; Default NONE.

comment
A human readable comment describing the table.

compaction
Options for SSTable compaction:
• bucket_high
• bucket_low
• cold_reads_to_omit
• enabled
• max_threshold
• min_threshold
• min_sstable_size
• sstable_size_in_mb
• tombstone_compaction_interval
• tombstone_threshold
Default SizeTieredCompaction

compression
The compression algorithm. Values: LZ4Compressor, SnappyCompressor, and DeflateCompressor. Default LZ4Compressor. Subproperties for the table:
• sstable_compression
• chunk_length_kb
• crc_check_chance

dclocal_read_repair_chance
The probability of read repairs being invoked over all replicas in the current data center. Default 0.1

default_time_to_live
The default expiration time in seconds for a table. Used in MapReduce/Hive scenarios in which you have no control of TTL. Default 0 seconds.

gc_grace_seconds
The time to wait before garbage collecting tombstones (deletion markers). Default 864000 seconds (10 days).

CQL Table Properties (continued)

min_index_interval, max_index_interval
Configures the sample frequency of the partition summary to control the sampling of entries from the partition index. Default 128 and 2048, respectively.

memtable_flush_period_in_ms
Forces flushing of the memtable after the number of specified milliseconds elapses. Default 0

read_repair_chance
Specifies the probability for invoking read repairs on non-quorum reads. Default 0.0

speculative_retry
Overrides normal read timeout when read_repair is not 1.0, sending another request to read. Options:
• ALWAYS — Retry reads of all replicas.
• Xpercentile — Retry reads based on the effect on throughput and latency.
• Yms — Retry reads after specified milliseconds
• NONE — Do not retry reads.
Default 99percentile.

Functions

Blob conversion
Converts native types into binary data (blob).
• typeAsBlob() takes a native type and returns it as a blob
• bigintAsBlob(3) returns 0x0000000000000003
• blobAsType takes a 64-bit blob argument and converts it to a bigint value
• blobAsBigint(0x0000000000000003) returns 3

dateOf()
Used in a SELECT clause to extract the timestamp of a timeuuid column in a resultset. Returns the extracted timestamp as a date.

minTimeuuid() and maxTimeuuid()
Returns a UUID-like result given a conditional time component as an argument.
Example:

```
SELECT * FROM myTable
  WHERE t > maxTimeuuid('2013-01-01 00:05+0000')
  AND t < minTimeuuid('2013-02-02 10:00+0000')
```

now()
Generates a new unique timeuuid, useful for inserting values. Returns a unique value.

TTL()
Returns the remaining time-to-live for a column.

unixTimestampOf()
Used in a SELECT clause to extract the timestamp of a timeuuid column in a resultset. Returns a raw, 64-bit integer timestamp.

WRITETIME()
Returns date/time in microseconds that the column was written to the database.

Syntax elements

Generally, the elements used in the command syntax have the following definitions. A few elements have a slightly different meaning when used with a particular command and are redefined in the synopsis of the command.

clustering_column
A column that, in addition to the partition key, determines clustering.

column_name
Alphanumeric column name, case-insensitive unless enclosed in double quotation marks. No reserved keywords. Unreserved keywords enclosed in quotation marks are ok. Enclose names having unparseable characters in double quotation marks.

constant
A string, integer, float, boolean, UUID, or blob.

counter_column_name
A column_name of a column of type counter.

keyspace_name
A keyspace name, starting with an alpha character, consisting of 32 or fewer alpha-numeric characters and underscores. Case-insensitive unless enclosed in double quotation marks.

key_value
The value of a primary key.

literal
• Data that is of a supported data type
• Float constant in E notation
• Numeric constant
• A letter, followed by any sequence of letters, digits, or the underscore
• A string, characters enclosed in single quotation marks
• Whitespace that separates of terms, otherwise ignored

partition_key
The column that determines on which node data is stored.

property
A CQL storage property, such as speculative_retry = '10ms'.

table_name
Valid table names are strings of alphanumeric characters and underscores, which begin with a letter.

timestamp
Microseconds representing the standard base time since epoch: January 1 1970 at 00:00:00 GMT.

variable
A bind variable, such as ?, used with a prepared statement.

Apache Cassandra™ Query Language (CQL)

SPECIFICATION 3.2.0

REFERENCE GUIDE - SUPPORTED DATA TYPES

DATASTAX

CQL Type	Description
ASCII	US-ASCII character string
BIGINT	64-bit signed long
BLOB	Arbitrary bytes (no validation), expressed as hexadecimal
BOOLEAN	true or false
COUNTER	Distributed counter value (64-bit long)
DECIMAL	Variable-precision decimal
DOUBLE	64-bit IEEE-754 floating point
FLOAT	32-bit IEEE-754 floating point
INET	IP address string in IPv4 or IPv6 form
INT	32-bit signed integer
LIST	A collection of one or more ordered elements
MAP	A JSON-style array of literals: { literal : literal, literal : literal ... }
SET	A collection of one or more elements
TEXT	UTF-8 encoded string
TIMESTAMP	Date plus time, encoded as 8 bytes since epoch
TUPLE	A group of two or three fields
UUID	A UUID in standard UUID format
TIMEUUID	Type 1 UUID only
VARCHAR	UTF-8 encoded string
VARINT	Arbitrary-precision integer

Office Locations

DATASTAX HQ - SF BAY AREA
3975 Freedom Circle
Santa Clara, CA 95054
650-389-6000

DATASTAX TX
902 East 5th St. #202
Austin, TX 78702
512-537-7809

LEGEND: * Uppercase means literal * Lowercase means not literal * Italics mean optional * The pipe (|) symbol means OR or AND/OR * Ellipsis (...) means repeatable * orange ( ) indicate scope, not literal