Datastax Enterprise Data Types

String Types

Type	Description	Constants Supported
ascii	US-ASCII character string	strings
text	UTF-8 encoded string	strings
varchar	UTF-8 encoded string (alias for text)	strings

Numeric Types

Type	Description	Constants Supported	Java Type Equivalent
tinyint	1-byte signed integer	integers	byte
smallint	2-byte signed integer	integers	short
int	32-bit signed integer	integers	int
bigint	64-bit signed long	integers	long
varint	Arbitrary-precision integer	integers	java.math.BigInteger
decimal	Variable-precision decimal	integers, floats	java.math.BigDecimal
float	32-bit IEEE-754 floating point	integers, floats	java.lang.Float
double	64-bit IEEE-754 floating point	integers, floats	java.lang.Double

Boolean Type:

Type	Description	Constants Supported
boolean	True or false value	booleans

Date and Time

Type	Description	Constants Supported
date	Date only (days since Unix epoch, Jan 1, 1970); format: yyyy-mm-dd	strings

Type	Description	Constants Supported
time	Time only (nanoseconds since midnight); format: HH:mm:ss[.fffffffff]	strings
timestamp	Date and time (millisecond precision since epoch); format: yyyy-mm-dd HH:mm:ss[.fffffffff]	integers, strings

UUID and Timebased UUID

Type	Description	Constants Supported
uuid	Universally unique identifier (standard UUID format)	uuids
timeuuid	Version 1 UUID (includes timestamp)	uuids

Ip Address Type:

Type	Description	Constants Supported
inet	IPv4 or IPv6 address	strings

Binary Data Type:

Type	Description	Constants Supported
blob	Arbitrary bytes (hexadecimal encoded)	blobs

Collections Type:

Type	Description	Syntax Example
list<T>	Ordered collection of 0+ elements (duplicates allowed)	[1, 2, 2]
set<T>	Unordered collection of 0+ unique elements	{1, 2}
map<K,V>	Key-value pairs (keys must be unique)	{'key1': 'value1', 'key2': 2}

Counter Type:

Type	Description	Constants Supported
counter	64-bit distributed counter (only for increment/decrement operations)	integers

Constraints:

Except Primary Key no other constraints are supported.

Indexes in Datastax Enterprise

Index Types in Datastax Enterprise:

Type	Description	Pros	Cons	Supported Versions
SAI (Storage-Attached Indexing)	Attaches indexes directly to memtables and SSTables using inverted indexes (for text) and kd-trees (for numerics). Default in DSE 6.8+.	- 40-86% higher write throughput - 200-670% lower write latency - 5-8x smaller disk footprint - Supports multiple indexes per table, vector search (AI/ML), AND/OR logic, ranges, CONTAINS on collections - Zero-copy streaming during node ops	- Slightly higher read latency in rare cases (e.g., SizeTiered compaction) - Requires Cassandra 4.0+	DSE 6.8+, HCD
2i (Original Secondary Indexing)	Stores indexes in hidden local tables per node.	- Simple for low-cardinality equality queries -	- High write amplification (2x+ latency) - Inefficient for high-cardinality or frequent	All DSE versions

Type	Description	Pros	Cons	Supported Versions
	Legacy default pre-4.0.	Works with partition keys for narrow scopes	updates/deletes - Larger disk use; hotspots on coordinators	
SASI (SSTable-Attached Secondary Indexing)	Per-SSTable index files for partial matches (e.g., LIKE) and sparse ranges.	- Good for text prefix/range queries - Better than 2i for sparse data (e.g., timestamps)	- Experimental/deprecated (disabled by default in 5.0+; enable via cassandra.yaml) - Higher overhead than SAI - Not for high-volume workloads	DSE 5.1-6.9 (experimental)
DSE Search	Solr-Lucene integration for full-text/tokenized search with analyzers and facets.	- Advanced text search (e.g., relevance scoring) - Faceted navigation	- Higher complexity and resource use - Not for simple filters; limited to text	DSE 6.8-6.9 (not in HCD)

Example of SAI Index:

```
CREATE CUSTOM INDEX user_email_idx ON keyspace.users (email) USING
'StorageAttachedIndex';
```

Example of 2i Index:

```
CREATE INDEX user_email_2i ON keyspace.users (email);
```

Example of SSA Index:

```
CREATE CUSTOM INDEX user_email_sasi ON keyspace.users (email) USING
'org.apache.cassandra.index.sasi.SASIIndex'
WITH OPTIONS = {'mode': 'PREFIX', 'tokenizer_class':
'org.apache.cassandra.index.sasi.analyzer.StandardAnalyzer'};
```

DSE Search Index:

DSE Search: Use OpsCenter or dsetool create_core for Solr configs; not pure CQL

When to Use:

- SAI: General filtering (e.g., numeric ranges, text equality, vector embeddings for AI). Ideal for query-driven models without denormalization.
- 2i/SASI: Legacy low-cardinality lookups (e.g., gender, status) or sparse timestamps; migrate to SAI.
- DSE Search: E-commerce faceting or log analysis with keywords.

Materialized Views (MVs)

Definition: MVs are automatically maintained tables derived from a base (source) table, with a different primary key structure to support new query patterns.

Introduced in Cassandra 3.0 adapted by DSE in version 6.0

Syntax:

```
CREATE MATERIALIZED VIEW keyspace.mv_table AS
SELECT pk_col, clustered_col, other_col FROM keyspace.base_table
WHERE pk_col IS NOT NULL AND other_col  IS NOT NULL
PRIMARY KEY (other_col, pk_col);
```

Materialized View must include pk_col of source table.

```
DESCRIBE MATERIALIZED VIEW cycling.cyclist_by_age;
```

Batch Operations (Execute Multi DML):

```
BEGIN BATCH

DML 1;
DML 2;
……..

APPLY BATCH;
```

Inserts, updates, or deletes to a single partition when atomicity and isolation is a requirement.

Multiple partition batching sends one message to the coordinator for all operations. The coordinator writes a batchlog that is replicated to other nodes to ensure that inconsistency will not occur if the coordinator fails.

Delete Operations in DSE:

Delete can used to delete a complete row from the table or set a column to null

Delete a Row:

Delete from <table> where <condition>;

Set a Column to Null:

Delete <column name> from emp where <condition>;

TTL in DataStax Enterprise

When configured TTL automatically deletes the data after specified time in seconds. TTL can be configured for at a table level or at column level:

Table Level:

```
ALTER TABLE cycling.cyclist_categories WITH default_time_to_live = 3600;

CREATE TABLE cycling.cyclist_names (
   id UUID PRIMARY KEY,
   lastname text,
   firstname text
) WITH default_time_to_live = 604800;
```

Column Level:

```
UPDATE cycling.cyclist_categories USING TTL 86400
SET category = 'Sprinter' WHERE id = 35c3d4d8-8b9a-11e2-9e96-08e0eb9c1c18;
```

A 0 value for TTL is considered as no expiry / diasabling TTL.