Contents

Chapter 1. Glossary

agent..................................................................................................................7
adjacency list......................................................................................................7
adjacent vertex..................................................................................................7
anti-entropy........................................................................................................7
authentication.................................................................................................7
authorization.................................................................................................7
back pressure...............................................................................................7
Bloom filter....................................................................................................7
bootstrap..........................................................................................................8
cardinality.......................................................................................................8
cluster...............................................................................................................8
clustering.........................................................................................................8
clustering column..........................................................................................8
coalescing strategy........................................................................................8
column...............................................................................................................8
column family..................................................................................................8
commit log.......................................................................................................8
compaction.......................................................................................................9
converge..........................................................................................................9
composite partition key..................................................................................9
compound primary key..................................................................................9
consistency......................................................................................................9
consistency level............................................................................................9
coordinator node...........................................................................................9
cross-data center forwarding........................................................................9
datacenter.....................................................................................................10
DateTieredCompactionStrategy (DTCS).........................................................10
deep query......................................................................................................10
denormalization..............................................................................................10
directed graph...............................................................................................10
EBNF.................................................................................................................10
edge

element

eventual consistency
faceted search

garbage collector

global index

gossip

graph

graph degree

graph index

graph partitioning

graph traversal

HDD

HDFS

headroom

idempotent

immutable

incident edge

index

Job Tracker

Kafka Struct

keyspace

keytab

krb5.conf

Lifecycle Manager (LCM)

LeveledCompactionStrategy (LCS)

linearizable consistency

MapReduce

materialized view

memtable

meta-property

mixed workload

mutation

node

node repair
<table>
<thead>
<tr>
<th>Term</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>normalization</td>
<td>14</td>
</tr>
<tr>
<td>OLAP</td>
<td>14</td>
</tr>
<tr>
<td>OLTP</td>
<td>15</td>
</tr>
<tr>
<td>order</td>
<td>15</td>
</tr>
<tr>
<td>partition index</td>
<td>15</td>
</tr>
<tr>
<td>partition key</td>
<td>15</td>
</tr>
<tr>
<td>partition range</td>
<td>15</td>
</tr>
<tr>
<td>partition summary</td>
<td>15</td>
</tr>
<tr>
<td>partitioned vertex</td>
<td>15</td>
</tr>
<tr>
<td>Partitioner</td>
<td>15</td>
</tr>
<tr>
<td>primary key</td>
<td>15</td>
</tr>
<tr>
<td>property</td>
<td>16</td>
</tr>
<tr>
<td>range movement</td>
<td>16</td>
</tr>
<tr>
<td>RDD</td>
<td>16</td>
</tr>
<tr>
<td>read repair</td>
<td>16</td>
</tr>
<tr>
<td>replication factor (RF)</td>
<td>16</td>
</tr>
<tr>
<td>replication group</td>
<td>16</td>
</tr>
<tr>
<td>replica placement strategy</td>
<td>16</td>
</tr>
<tr>
<td>role</td>
<td>16</td>
</tr>
<tr>
<td>rolling restart</td>
<td>16</td>
</tr>
<tr>
<td>row</td>
<td>17</td>
</tr>
<tr>
<td>row cache</td>
<td>17</td>
</tr>
<tr>
<td>scan query</td>
<td>17</td>
</tr>
<tr>
<td>scheme</td>
<td>17</td>
</tr>
<tr>
<td>SearchAnalytics</td>
<td>17</td>
</tr>
<tr>
<td>search index</td>
<td>17</td>
</tr>
<tr>
<td>seed</td>
<td>17</td>
</tr>
<tr>
<td>segment</td>
<td>17</td>
</tr>
<tr>
<td>serializable consistency</td>
<td>18</td>
</tr>
<tr>
<td>SizeTieredCompactionStrategy (STCS)</td>
<td>18</td>
</tr>
<tr>
<td>slice</td>
<td>18</td>
</tr>
<tr>
<td>Snitch</td>
<td>18</td>
</tr>
<tr>
<td>SSD</td>
<td>18</td>
</tr>
<tr>
<td>SSTable</td>
<td>18</td>
</tr>
<tr>
<td>static column</td>
<td>18</td>
</tr>
</tbody>
</table>
streaming........................................................................................................................................... 18
strong consistency.................................................................................................................................. 18
superuser.................................................................................................................................................. 19
table.......................................................................................................................................................... 19
Task Tracker........................................................................................................................................... 19
TimeWindowCompactionStrategy (TWCS).............................................................................................. 19
token.......................................................................................................................................................... 19
tombstone.................................................................................................................................................. 19
traversal source....................................................................................................................................... 19
TTL.............................................................................................................................................................. 19
tunable consistency................................................................................................................................ 20
undirected graph.................................................................................................................................... 20
weak consistency...................................................................................................................................... 20
wide row.................................................................................................................................................... 20
upsert.......................................................................................................................................................... 20
vertex........................................................................................................................................................... 20
vertex-centric index................................................................................................................................ 20
vertex degree.......................................................................................................................................... 20
Vnode.......................................................................................................................................................... 20
zombie....................................................................................................................................................... 21
agent
DataStax agents must be installed on every managed node in a cluster and are necessary to perform most of the functionality within OpsCenter. See Lifecycle Manager (LCM).

adjacency list
A collection of unordered lists used to represent a finite graph. Each list describes the set of neighbors of a vertex in the graph.

adjacent vertex
A vertex directly attached to another vertex by an edge in a graph.

anti-entropy
The synchronization of replica data on nodes to ensure that the data is fresh.

authentication
Process of establishing the identity of a user, DSE tool, or application.

authorization
Process of establishing permissions to database resources through roles.

back pressure
Pausing or blocking the buffering of incoming requests after reaching the threshold until the internal processing of buffered requests catches up.

Bloom filter
An off-heap structure associated with each SSTable that checks if any data for the requested row exists in the SSTable before doing any disk I/O.
**bootstrap**
The process by which new nodes join the cluster transparently gathering the data needed from existing nodes.

**cardinality**
The number of unique values in a column. For example, a column of employee ID numbers, unique for each employee, would have high cardinality; a column of employee ZIP codes would have low cardinality.

An index on a column with low cardinality can boost read performance, since the index is significantly smaller than the column. An index for a high-cardinality column may reduce performance. If your application requires a search on a high-cardinality column, a materialized view might be a better choice.

**cluster**
Two or more database instances that exchange messages using the gossip protocol.

**clustering**
The storage engine process that creates an index and keeps data in order based on the index.

**clustering column**
In the table definition, a clustering column is a column that is part of the compound primary key definition, but not the first column, which is the position reserved for the partition key. Columns are clustered in multiple rows within a single partition. The clustering order is determined by the position of columns in the compound primary key definition.

**coalescing strategy**
Strategy to combine multiple network messages into a single packet for outbound TCP connections to nodes in the same data center (intra-DC) or to nodes in a different data center (inter-DC). A coalescing strategy is provided with a blocking queue of pending messages and an output collection for messages to send.

**column**
The smallest increment of data, which contains a name, a value, and a timestamp.

**column family**
A container for rows, similar to the table in a relational system. Called table in CQL 3.

**commit log**
A file to which the database appends changed data for recovery in the event of a hardware failure.
**compaction**
The process of consolidating SSTables, discarding tombstones, and regenerating the SSTable index. The available compaction strategies are:

- **DateTieredCompactionStrategy (DTCS)** (deprecated)
- **LeveledCompactionStrategy (LCS)**
- **SizeTieredCompactionStrategy (STCS)**
- **TimeWindowCompactionStrategy (TWCS)**

**converge**
The real-world state of the node, datacenter, or cluster has been aligned with the desired state by successfully executing a Lifecycle Manager job.

**composite partition key**
A partition key consisting of multiple columns.

**compound primary key**
A primary key consisting of the partition key, which determines on which node data is stored, and one or more additional columns that determine clustering.

**consistency**
The synchronization of data on replicas in a cluster. Consistency is categorized as weak or strong.

**consistency level**
A setting that defines a successful write or read by the number of cluster replicas that acknowledge the write or respond to the read request, respectively.

**coordinator node**
The node that determines which nodes in the ring should get the request based on the cluster configured snitch.

**cross-data center forwarding**
A technique for optimizing replication across data centers that sends data from one data center to a node in another data center, and that node forwards the data to other nodes in its data center.
**datacenter**
A group of related nodes that are configured together within a cluster for replication and workload segregation purposes. Not necessarily a separate location or physical data center. Datacenter names are case sensitive and cannot be changed.

**DateTieredCompactionStrategy (DTCS)**
DateTieredCompactionStrategy (DTCS) is deprecated starting in Apache Cassandra 3.8 and in DataStax Enterprise. This strategy is particularly useful for time series data. It stores data written within a certain period of time in the same SSTable. For example, Apache Cassandra can store your last hour of data in one SSTable time window, and the next 4 hours of data in another time window, and so on. The most common queries for time series workloads retrieve the last hour/day/month of data.

**deep query**
Graph queries that traverse a dense graph (a large number of connected vertices) or a graph with a high branching factor.

**denormalization**
Denormalization refers to the process of optimizing the read performance of a database by adding redundant data or by grouping data. This process is accomplished by duplicating data in multiple tables, grouping data for queries.

**directed graph**
A set of vertices and a set of arcs (ordered pairs of vertices). In DSE Graph, the terminology “arcs” is not used, and edges are directional.

**EBNF**
EBNF (Extended Backus-Naur Form) syntax expresses a context-free grammar that formally describes a language. EBNF extends its precursor BNF (Backus-Naur Form) with additional operators allowed in expansions. Syntax (railroad) diagrams graphically depict EBNF grammars.

**edge**
A connection between graph vertices. Edges can be unordered (no directional orientation) or ordered (directional). An edge can also be described as an object that has a vertex at its tail and head.

**element**
A graph element is a vertex, edge, or property.
**eventual consistency**

The database maximizes availability and partition tolerance. It ensures eventual consistency by updating all replicas during read operations and periodically checking and updating any replicas not directly accessed. This ensures that any query always returns the most recent version of the result set, and that all replicas of any given row will eventually become completely consistent with each other.

**faceted search**

Faceted search is the dynamic clustering of items or search results into categories that uses any value in any field to drill into search results, or even skip searching entirely.

**garbage collector**

Garbage collector is a Java background process that frees heap memory when it is no longer in use by the program. The main algorithms in Java to allocate and clean up memory are Continuous Mark Sweep (CMS) and Garbage-First (G1). By default, DataStax Enterprise 5.1 and higher use the G1 garbage collector.

**global index**

An index structure over the entire graph.

**gossip**

A peer-to-peer communication protocol for exchanging location and state information between nodes.

**graph**

A collection of vertices and edges.

**graph degree**

The largest vertex degree of the graph.

**graph index**

An graph index is a data structure that allows for the fast retrieval of elements by a particular key-value pair.

**graph partitioning**

A process that consists of dividing a graph into components, such that the components are of about the same size and there are few connections between the components.
graph traversal
An algorithmic walk across the elements of a graph according to the referential structure explicit within the graph data structure.

HDD
A hard disk drive (HDD) or spinning disk is a data storage device used for storing and retrieving digital information using one or more rigid rapidly rotating disks. Compare to SSD.

HDFS
Hadoop Distributed File System (HDFS) stores data on nodes to improve performance. HDFS is a necessary component in addition to MapReduce in a Hadoop distribution.

headroom
The amount of disk space required by a process (such as compaction) in addition to the space occupied by the data being processed.

idempotent
An operation that can occur multiple times without changing the result, such as performing the same update multiple times without affecting the outcome.

immutable
Data on disk that cannot be overwritten.

incident edge
In a graph, an edge incident to a particular vertex, meaning that the edge and vertex touch.

index
A native capability for finding a column in the database that does not involve using the primary key.

Job Tracker
Job Trackers are used for analytics nodes that analyze data using Hadoop, including DSE Hadoop and external Hadoop systems. Within a data center, the Job Tracker monitors the execution and status of distributed tasks that comprise a MapReduce job.
Kafka Struct

A Kafka Struct is an Apache Kafka structured record containing a set of named fields with values, each field using an independent Schema. DataStax Apache Kafka Connector supports structs, including advanced types such as Avro format backed by the schema registry.

keyspace

A namespace container that defines how data is replicated on nodes in each datacenter.

keytab

File that contains a pair of Kerberos principals and encrypted keys. Use a keytab file to authenticate with a Kerberos enabled DataStax database without entering a password.

krb5.conf

File that contains the Kerberos configuration used by clients for connection and ticket generation. See MIT Kerberos krb5.conf documentation. The default location is /etc. If krb5.conf is in another location, override the default location by setting the environment variable KRB5_CONFIG. To use multiple configuration files, set a colon-separated filename list in KRB5_CONFIG; all files are read.

Lifecycle Manager (LCM)

Lifecycle Manager (LCM) is a provisioning and configuration management system for easily managing DataStax Enterprise (DSE) clusters. It is a web interface that enables efficient installation and configuration of DataStax Enterprise nodes. It completely defines the cluster configuration including datacenter and node topology and integrates deeply with the full spectrum of DataStax Enterprise settings.

LeveledCompactionStrategy (LCS)

This compaction strategy creates SSTables of a fixed, relatively small size that are grouped into levels. Within each level, SSTables are guaranteed to be non-overlapping. Each level (L0, L1, L2 and so on) is 10 times as large as the previous. Disk I/O is more uniform and predictable on higher than on lower levels as SSTables are continuously being compacted into progressively larger levels. At each level, row keys are merged into non-overlapping SSTables in the next level. This process can improve performance for reads, because the database can determine which SSTables in each level to check for the existence of row key data.

linearizable consistency

Also called serializable consistency. The restriction that one operation cannot be executed unless and until another operation has completed.

To ensure linearizable consistency in writes, the database supports Lightweight transactions. The first phase of a Lightweight transaction works at SERIAL consistency and follows the Paxos protocol to ensure that the required operation succeeds. If this phase succeeds, the write is performed at the consistency level specified for the operation. Reads performed at the SERIAL consistency level are executed without database's built-in read repair operations.
**MapReduce**
Hadoop’s parallel processing engine that can process large data sets relatively quickly. A necessary component in addition to MapReduce in a Hadoop distribution.

**materialized view**
A materialized view is a table with data that is automatically inserted and updated from another base table. The materialized view has a primary key that differs from the base table, so that different queries can be implemented.

**memtable**
A database table-specific, in-memory data structure that resembles a write-back cache.

**meta-property**
In a graph, a property that describes some attribute of another property.

**mixed workload**
A single cluster can run transactional, search, and analytics nodes.

**mutation**
A mutation is either an insertion or a deletion.

**node**
A Java virtual machine (a platform-independent execution environment that converts Java bytecode into machine language and executes it) that runs an instance of the Licensed Software.

**node repair**
A process that makes all data on a replica consistent.

**normalization**
Normalization refers to a series of steps used to eliminate redundancy and reduce the chances of data inconsistency in a database’s schema. In DataStax Enterprise, this process is inefficient because joining data in multiple tables for queries requires accessing more nodes.

**OLAP**
Online Analytical Processing (OLAP) performs multidimensional analysis of business data and provides the capability for complex calculations, trend analysis, and sophisticated data modeling. Compare to OLTP.
OLTP
Online transaction processing (OLTP) is characterized by a large number of short on-line transactions for data entry and retrieval. Compare to OLAP.

order
The magnitude of the number of edges to the number of vertices.

partition index
A list of primary keys and the start position of data.

partition key
The first column declared in the PRIMARY KEY definition, or in the case of a compound key, multiple columns can declare those columns that form the primary key.

partition range
The limits of the partition that differ depending on the configured partitioner. Murmur3Partitioner (default) range is $-2^{63}$ to $+2^{63}$ and RandomPartitioner range is $0$ to $2^{127}$-1.

partition summary
A subset of the partition index. By default, 1 partition key out of every 128 is sampled.

partitioned vertex
Used for vertices that have a very large number of edges, a partitioned vertex consists of a portion of a vertex's data that results from dividing the vertex into smaller components for graph database storage.

Partitioner
Distributes the data across the cluster. The types of partitioners are Murmur3Partitioner (default), RandomPartitioner, and OrderPreservingPartitioner.

primary key
The partition key. One or more columns that uniquely identify a row in a table.
property
A key-value pair that describes some attribute of either a vertex or an edge. Property key is used to describe the key in the key-value pair. All properties are global in DSE Graph, meaning that a property can be used for any vertices. For example, “name” can be used for all vertices in a graph.

range movement
A change in the expanse of tokens assigned to a node.

RDD
Resilient Distributed Datasets (RDD) is a fundamental data structure of Spark. It is an immutable distributed collection of objects. RDDs have actions, which return values, and transformations, which return pointers to new RDDs.

read repair
A process that updates database replicas with the most recent version of frequently-read data.

replication factor (RF)
The total number of replicas across the cluster is referred to as the replication factor (RF). A replication factor of 1 means that there is only one copy of each row in the cluster. If the node containing the row goes down, the row cannot be retrieved. A replication factor of 2 means two copies of each row, where each copy is on a different node. All replicas are equally important; there is no primary or master replica.

replication group
See data center data center.

replica placement strategy
A specification that determines the replicas for each row of data.

role
Set of permissions assigned to users that limit access to database resources. When using internal authentication, roles can also have passwords and represent a single user, DSE client tool, or application.

rolling restart
A procedure that is performed during upgrading nodes in a cluster for zero downtime. Nodes are upgraded and restarted one at a time, while other nodes continue to operate online.
**row**

1) Columns that have the same primary key. 2) A collection of cells per combination of columns in the storage engine.

**row cache**

A database component for improving performance of read-intensive operations. The row cache, in off-heap memory, holds rows most recently read from the local SSTables. Each local read operation stores its result set in the row cache and sends it to the coordinator node. The next read first checks the row cache. If the required data is there, the database returns it immediately. This initial read can save further seeks in the Bloom filter, partition key cache, partition summary, partition index, and SSTables.

The database uses LRU (least-recently-used) eviction to ensure that the row cache is refreshed with the most frequently accessed rows. The size of the row cache can be configured in the cassandra.yaml file.

**scan query**

A graph query that traverses an entire graph or large sections of the graph.

**scheme**

1) Authentication: Defines a service used for authentication and/or role assignment, such as Kerberos or LDAP. 2) Database: Describes all database resources.

**SearchAnalytics**

Nodes started as stand-alone processes or services in SearchAnalytics mode allow you to create analytics queries that use search indexes.

**search index**

In DataStax Enterprise, a search index is an Apache Solr™ core. Each DSE Search index uses an internally stored index configuration pair that is automatically generated when the index is created. See Search index config 6.0 | 6.7 and Search index schema 6.0 | 6.7.

**seed**

A seed, or seed node, is used to bootstrap the gossip process for new nodes joining a cluster. A seed node provides no other function and is not a single point of failure for a cluster.

**segment**

A segment is a small token range of a table used by NodeSync to validate and repair data across replicas. NodeSync uses the size of the entire table to determine how many segments (depth) to divide the table into so that segments are ~200MB.
serializable consistency
See linearizable consistency.

SizeTieredCompactionStrategy (STCS)
The default compaction strategy. This strategy triggers a minor compaction when there are a number of similar sized SSTables on disk as configured by the table subproperty, min_threshold. A minor compaction does not involve all the tables in a keyspace. Also see STCS compaction subproperties in the relevant CQL documentation.

slice
A set of clustered columns in a partition that you query as a set using, for example, a conditional WHERE clause.

Snitch
The mapping from the IP addresses of nodes to physical and virtual locations, such as racks and data centers. There are several types of snitches. The type of snitch affects the request routing mechanism.

SSD
A solid-state drive (SSD) is a solid-state storage device that uses integrated circuits to persistently store data. Compare to HDD.

SSTable
A sorted string table (SSTable) is an immutable data file to which the database writes memtables periodically. SSTables are stored on disk sequentially and maintained for each database table.

static column
A special column that is shared by all rows of a partition.

streaming
A component that handles data exchange among nodes in the cluster. It is part of SSTable file. Examples include:

- When bootstrapping a new node, the new node gets data from existing nodes using streaming.
- When running nodetool repair, nodes exchange out-of-sync data using streaming.
- When bulkloading data from backup, sstableloader uses streaming to complete task.

strong consistency
When reading data, the database performs read repair before returning results.
**superuser**

Superuser is a role attribute that provides root database access. Superusers have all permissions on all objects. By default, DataStax Enterprise database includes a superuser role `cassandra` with password `cassandra`. This account runs queries, including logins, with a consistency level of `QUORUM`. DataStax recommends, creating a superuser for your deployment and removing the `cassandra` role.

**table**

A collection of ordered (by name) columns fetched by row. A row consists of columns and have a primary key. The first part of the key is a column name. Subsequent parts of a compound key are other column names that define the order of columns in the table.

**Task Tracker**

One Task Tracker service per node handles the Hadoop MapReduce tasks that are scheduled for a Hadoop-enabled node.

**TimeWindowCompactionStrategy (TWCS)**

This compaction strategy compacts SSTables based on series of time windows. During the current time window, the SSTables are compacted into one or more SSTables. At the end of the current time window, all SSTables are compacted into a single larger SSTable. Then the next time window starts and the compaction process repeats. Each TWCS time window contains data within a specified range and contains varying amounts of data.

**token**

An element on the ring that depends on the partitioner. A token determines the node's position on the ring and the portion of data it is responsible for. The range for the Murmur3Partitioner (default) is $-2^{63}$ to $+2^{63}$. The range for the RandomPartitioner is $0$ to $2^{127} - 1$.

**tombstone**

A marker in a row that indicates a column was deleted. During compaction, marked columns are deleted.

**traversal source**

A domain specific language (DSL) that specifies the traversal methods used by a graph traversal.

**TTL**

Time-to-live (TTL) is an optional expiration date for values that are inserted into a column. Also see *Expiring columns* in the relevant CQL documentation.
**tunable consistency**
The database ensures that all replicas of any given row eventually become completely consistent. For situations requiring immediate and complete consistency, the database can be tuned to provide 100% consistency for specified operations, datacenters or clusters. The database cannot be tuned to complete consistency for all data and operations.

**undirected graph**
A set of vertices and a set of edges (unordered pairs of vertices).

**weak consistency**
When reading data, the database performs read repair after returning results.

**wide row**
A data partition, which CQL 3 transposes into familiar row-based resultsets.

**upsert**
A change in the database that updates a specified column in a row if the column exists or inserts the column if it does not exist.

**vertex**
A vertex is the fundamental unit of which graphs are formed. A vertex can also be described as an object that has incoming and outgoing edges.

**vertex-centric index**
A local index structure built per vertex in a graph.

**vertex degree**
The number of edges incident to a vertex in a graph.

**Vnode**
Vnode is a virtual node. Normally, nodes are responsible for a single partitioning range in the full token range of a cluster. If vnodes are enabled, each node is responsible for several virtual nodes, effectively spreading a partitioning range across more nodes in the cluster. Doing so can reduce the risk of hotpotting, or straining one node in the cluster.
zombie

A row or cell that reappears in a database table after deletion. This can happen if a node goes down for a long period of time and is then restored without being repaired.

Deleted data are not erased from database tables, but marked with tombstones until compaction. The tombstones created on one node must be propagated to the nodes containing the deleted data. If one of these nodes goes down before this happens, the node may not receive the most up-to-date tombstones. If the node is not repaired before it comes back online, the database finds the non-tombstoned items and propagates them to other nodes as new data.

To avoid this problem, run `nodetool repair` on any restored node before rejoining it to its cluster.