- **Creating NDB tables with user-defined partitioning.** Support for user-defined partitioning in NDB Cluster is restricted to [LINEAR] KEY partitioning. Using any other partitioning type with ENGINE=NDB or ENGINE=NDBCLUSTER in a CREATE TABLE statement results in an error.

  It is possible to override this restriction, but doing so is not supported for use in production settings. For details, see User-defined partitioning and the NDB storage engine (NDB Cluster).

  Default partitioning scheme. All NDB Cluster tables are by default partitioned by KEY using the table's primary key as the partitioning key. If no primary key is explicitly set for the table, the "hidden" primary key automatically created by the NDB storage engine is used instead. For additional discussion of these and related issues, see Section 24.2.5, "KEY Partitioning".

  CREATE TABLE and ALTER TABLE statements that would cause a user-partitioned NDBCLUSTER table not to meet either or both of the following two requirements are not permitted, and fail with an error:

  1. The table must have an explicit primary key.

  2. All columns listed in the table's partitioning expression must be part of the primary key.

  Exception. If a user-partitioned NDBCLUSTER table is created using an empty column-list (that is, using PARTITION BY [LINEAR] KEY()), then no explicit primary key is required.

  Maximum number of partitions for NDBCLUSTER tables. The maximum number of partitions that can defined for a NDBCLUSTER table when employing user-defined partitioning is 8 per node group. (See Section 23.1.2, "NDB Cluster Nodes, Node Groups, Fragment Replicas, and Partitions", for more information about NDB Cluster node groups.

  DROP PARTITION not supported. It is not possible to drop partitions from NDB tables using ALTER TABLE ... DROP PARTITION. The other partitioning extensions to ALTER TABLE—ADD PARTITION, REORGANIZE PARTITION, and COALESCE PARTITION—are supported for NDB tables, but use copying and so are not optimized. See Section 24.3.1, "Management of RANGE and LIST Partitions" and Section 13.1.9, "ALTER TABLE Statement".

- **JSON data type.** The MySQL JSON data type is supported for NDB tables in the mysqld supplied with NDB 8.0.

  An NDB table can have a maximum of 3 JSON columns.

  The NDB API has no special provision for working with JSON data, which it views simply as BLOB data. Handling data as JSON must be performed by the application.

23.1.7.2 Limits and Differences of NDB Cluster from Standard MySQL Limits

In this section, we list limits found in NDB Cluster that either differ from limits found in, or that are not found in, standard MySQL.

Memory usage and recovery. Memory consumed when data is inserted into an NDB table is not automatically recovered when deleted, as it is with other storage engines. Instead, the following rules hold true:

- A DELETE statement on an NDB table makes the memory formerly used by the deleted rows available for re-use by inserts on the same table only. However, this memory can be made available for general re-use by performing OPTIMIZE TABLE.

  A rolling restart of the cluster also frees any memory used by deleted rows. See Section 23.5.5, "Performing a Rolling Restart of an NDB Cluster".