relatively small cost to write performance. Beginning with NDB 8.0.19, 1 is the default for READ_BACKUP, and the default for ndb_read_backup is ON (previously, read from any replica was disabled by default).

You can set READ_BACKUP for an existing table online, using an ALTER TABLE statement similar to one of those shown here:

```
ALTER TABLE ... ALGORITHM=INPLACE, COMMENT="NDB_TABLE=READ_BACKUP=1";

ALTER TABLE ... ALGORITHM=INPLACE, COMMENT="NDB_TABLE=READ_BACKUP=0";
```

For more information about the ALGORITHM option for ALTER TABLE, see Section 23.5.11, “Online Operations with ALTER TABLE in NDB Cluster”.

PARTITION_BALANCE: Provides additional control over assignment and placement of partitions. The following four schemes are supported:

1. FOR_RP_BY_NODE: One partition per node.

   Only one LDM on each node stores a primary partition. Each partition is stored in the same LDM (same ID) on all nodes.

2. FOR_RA_BY_NODE: One partition per node group.

   Each node stores a single partition, which can be either a primary replica or a backup replica. Each partition is stored in the same LDM on all nodes.

3. FOR_RP_BY_LDM: One partition for each LDM on each node; the default.

   This is the setting used if READ_BACKUP is set to 1.

4. FOR_RA_BY_LDM: One partition per LDM in each node group.

   These partitions can be primary or backup partitions.

5. FOR_RA_BY_LDM_X_2: Two partitions per LDM in each node group.

   These partitions can be primary or backup partitions.

6. FOR_RA_BY_LDM_X_3: Three partitions per LDM in each node group.

   These partitions can be primary or backup partitions.

7. FOR_RA_BY_LDM_X_4: Four partitions per LDM in each node group.

   These partitions can be primary or backup partitions.

PARTITION_BALANCE is the preferred interface for setting the number of partitions per table. Using MAX_ROWS to force the number of partitions is deprecated but continues to be supported for backward compatibility; it is subject to removal in a future release of MySQL NDB Cluster. (Bug #81759, Bug #23544301)

FULLY_REPLICATED controls whether the table is fully replicated, that is, whether each data node has a complete copy of the table. To enable full replication of the table, use FULLY_REPLICATED=1.

This setting can also be controlled using the ndb_fully_replicated system variable. Setting it to ON enables the option by default for all new NDB tables; the default is OFF. The ndb_data_node_neighbour system variable is also used for fully replicated tables, to ensure that when a fully replicated table is accessed, we access the data node which is local to this MySQL Server.