- The index type for a non-SPATIAL index depends on the storage engine. Currently, B-tree is used.
- Permitted for a column that can have NULL values only for InnoDB, MyISAM, and MEMORY tables.

Index Options

Following the key part list, index options can be given. An *index_option* value can be any of the following:

- KEY_BLOCK_SIZE [=] *value*

  For MyISAM tables, KEY_BLOCK_SIZE optionally specifies the size in bytes to use for index key blocks. The value is treated as a hint; a different size could be used if necessary. A KEY_BLOCK_SIZE value specified for an individual index definition overrides a table-level KEY_BLOCK_SIZE value.

  KEY_BLOCK_SIZE is not supported at the index level for InnoDB tables. See Section 13.1.20, “CREATE TABLE Statement”.

- *index_type*

  Some storage engines permit you to specify an index type when creating an index. For example:

```
CREATE TABLE lookup (id INT) ENGINE = MEMORY;
CREATE INDEX id_index ON lookup (id) USING BTREE;
```

  Table 13.1, “Index Types Per Storage Engine” shows the permissible index type values supported by different storage engines. Where multiple index types are listed, the first one is the default when no index type specifier is given. Storage engines not listed in the table do not support an *index_type* clause in index definitions.

Table 13.1 Index Types Per Storage Engine

Storage Engine	Permissible Index Types
InnoDB	BTREE
MyISAM	BTREE
MEMORY/HEAP	HASH, BTREE
NDB	HASH, BTREE (see note in text)

The *index_type* clause cannot be used for FULLTEXT INDEX or (prior to MySQL 8.0.12) SPATIAL INDEX specifications. Full-text index implementation is storage engine dependent. Spatial indexes are implemented as R-tree indexes.

If you specify an index type that is not valid for a given storage engine, but another index type is available that the engine can use without affecting query results, the engine uses the available type. The parser recognizes RTREE as a type name. As of MySQL 8.0.12, this is permitted only for SPATIAL indexes. Prior to 8.0.12, RTREE cannot be specified for any storage engine.

BTREE indexes are implemented by the NDB storage engine as T-tree indexes.

Note

For indexes on NDB table columns, the USING option can be specified only for a unique index or primary key. USING HASH prevents the creation of an ordered index; otherwise, creating a unique index or primary key on an NDB table automatically results in the creation of both an ordered index and a hash index, each of which indexes the same set of columns.