columns are listed as the *first* columns in the same order. Such an index is created on the referencing table automatically if it does not exist. This index might be silently dropped later if you create another index that can be used to enforce the foreign key constraint. *`index_name`*, if given, is used as described previously.

- `InnoDB` permits a foreign key to reference any index column or group of columns. However, in the referenced table, there must be an index where the referenced columns are the *first* columns in the same order. Hidden columns that `InnoDB` adds to an index are also considered (see Section 15.6.2.1, "Clustered and Secondary Indexes").

  `NDB` requires an explicit unique key (or primary key) on any column referenced as a foreign key. `InnoDB` does not, which is an extension of standard SQL.

- Index prefixes on foreign key columns are not supported. Consequently, `BLOB` and `TEXT` columns cannot be included in a foreign key because indexes on those columns must always include a prefix length.

- `InnoDB` does not currently support foreign keys for tables with user-defined partitioning. This includes both parent and child tables.

  This restriction does not apply for `NDB` tables that are partitioned by `KEY` or `LINEAR KEY` (the only user partitioning types supported by the `NDB` storage engine); these may have foreign key references or be the targets of such references.

- A table in a foreign key relationship cannot be altered to use another storage engine. To change the storage engine, you must drop any foreign key constraints first.

- A foreign key constraint cannot reference a virtual generated column.

For information about how the MySQL implementation of foreign key constraints differs from the SQL standard, see Section 1.7.2.3, "FOREIGN KEY Constraint Differences".

Referential Actions

When an `UPDATE` or `DELETE` operation affects a key value in the parent table that has matching rows in the child table, the result depends on the *referential action* specified by `ON UPDATE` and `ON DELETE` subclauses of the `FOREIGN KEY` clause. Referential actions include:

- `CASCADE`: Delete or update the row from the parent table and automatically delete or update the matching rows in the child table. Both `ON DELETE CASCADE` and `ON UPDATE CASCADE` are supported. Between two tables, do not define several `ON UPDATE CASCADE` clauses that act on the same column in the parent table or in the child table.

  If a `FOREIGN KEY` clause is defined on both tables in a foreign key relationship, making both tables a parent and child, an `ON UPDATE CASCADE` or `ON DELETE CASCADE` subclause defined for one `FOREIGN KEY` clause must be defined for the other in order for cascading operations to succeed. If an `ON UPDATE CASCADE` or `ON DELETE CASCADE` subclause is only defined for one `FOREIGN KEY` clause, cascading operations fail with an error.

  

  Note

  Cascaded foreign key actions do not activate triggers.

- `SET NULL`: Delete or update the row from the parent table and set the foreign key column or columns in the child table to `NULL`. Both `ON DELETE SET NULL` and `ON UPDATE SET NULL` clauses are supported.