statement that you are using (Section 13.2.6, “INSERT Statement”, Section 13.2.13, “UPDATE Statement”, and so forth).

You can get information about the number of rows actually inserted or updated with the `mysql_info()` C API function. You can also use the `SHOW WARNINGS` statement. See mysql_info(), and Section 13.7.7.42, “SHOW WARNINGS Statement”.

`InnoDB` and `NDB` tables support foreign keys. See Section 1.7.3.2, “FOREIGN KEY Constraints”.

1.7.3.2 FOREIGN KEY Constraints

Foreign keys let you cross-reference related data across tables, and foreign key constraints help keep this spread-out data consistent.

MySQL supports `ON UPDATE` and `ON DELETE` foreign key references in `CREATE TABLE` and `ALTER TABLE` statements. The available referential actions are `RESTRICT`, `CASCADE`, `SET NULL`, and `NO ACTION` (the default).

`SET DEFAULT` is also supported by the MySQL Server but is currently rejected as invalid by `InnoDB`. Since MySQL does not support deferred constraint checking, `NO ACTION` is treated as `RESTRICT`. For the exact syntax supported by MySQL for foreign keys, see Section 13.1.20.5, “FOREIGN KEY Constraints”.

`MATCH FULL`, `MATCH PARTIAL`, and `MATCH SIMPLE` are allowed, but their use should be avoided, as they cause the MySQL Server to ignore any `ON DELETE` or `ON UPDATE` clause used in the same statement. `MATCH` options do not have any other effect in MySQL, which in effect enforces `MATCH SIMPLE` semantics full-time.

MySQL requires that foreign key columns be indexed; if you create a table with a foreign key constraint but no index on a given column, an index is created.

You can obtain information about foreign keys from the `INFORMATION_SCHEMA.KEY_COLUMN_USAGE` table. An example of a query against this table is shown here:

```
mysql> SELECT TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME, CONSTRAINT_NAME
     > FROM INFORMATION_SCHEMA.KEY_COLUMN_USAGE
     > WHERE REFERENCED_TABLE_SCHEMA IS NOT NULL;
+--------------+---------------+-------------+-----------------+
| TABLE_SCHEMA | TABLE_NAME    | COLUMN_NAME | CONSTRAINT_NAME |
+--------------+---------------+-------------+-----------------+
| fk1          | myuser        | myuser_id   | f               |
| fk1          | product_order | customer_id | f2              |
| fk1          | product_order | product_id  | f1              |
+--------------+---------------+-------------+-----------------+
3 rows in set (0.01 sec)
```

Information about foreign keys on `InnoDB` tables can also be found in the `INNODB_FOREIGN` and `INNODB_FOREIGN_COLS` tables, in the `INFORMATION_SCHEMA` database.

`InnoDB` and `NDB` tables support foreign keys.

1.7.3.3 Enforced Constraints on Invalid Data

By default, MySQL 8.0 rejects invalid or improper data values and aborts the statement in which they occur. It is possible to alter this behavior to be more forgiving of invalid values, such that the server coerces them to valid ones for data entry, by disabling strict SQL mode (see Section 5.1.11, “Server SQL Modes”), but this is not recommended.

Older versions of MySQL employed the forgiving behavior by default; for a description of this behavior, see Constraints on Invalid Data.