Table of Contents

Master/detail relationships are used to create relationships between tables or other datasets. The master can also be viewed as a group, whereas the detail contains the individual sets of the group.

In short, master/detail is used to show the 1:n and n:m relationships of a database.

Example

Our application handles the administration of countries, states, and districts. Tables in the database are created in 3rd NF. The following is an excerpt from the HSQLDB:

createdb.sql

```
CREATE TABLE COUNTRIES
(
  ID      INTEGER IDENTITY,
  COUNTRY VARCHAR(200) NOT NULL,
  EU      CHAR(1) DEFAULT 'N' NOT NULL,
  CONSTRAINT CTRY_UK UNIQUE (COUNTRY)
)

CREATE TABLE STATES
(
  ID      INTEGER IDENTITY,
  CTRY_ID INTEGER NOT NULL,
  STATE   VARCHAR(200) NOT NULL,
  CONSTRAINT STAT_UK UNIQUE (STATE),
  CONSTRAINT STAT_CTRY_ID_FK FOREIGN KEY (CTRY_ID) REFERENCES COUNTRIES
(ID)
)

CREATE TABLE DISTRICTS
(
  ID       INTEGER IDENTITY,
  STAT_ID  INTEGER NOT NULL,
  DISTRICT VARCHAR(200),
  CONSTRAINT DIST_UK UNIQUE (DISTRICT),
  CONSTRAINT DIST_STAT_ID_FK FOREIGN KEY (STAT_ID) REFERENCES STATES
(ID)
)
```

As the table definition shows, countries consist of individual states, and states are made up of districts. In our application, we want to create a form that allows the editing of countries, states and districts, as well as the relationships between them.

We first create our server objects:

```
/**
 * Returns the countries storage.
 *
```

```
 * @return the Countries storage
 * @throws Exception if the initialization throws an error
 */
public DBStorage getCountries() throws Exception
{
   DBStorage dbsCountries = (DBStorage)get("countries");

   if (dbsCountries == null)
   {
      dbsCountries = new DBStorage();
      dbsCountries.setDBAccess(getDBAccess());
      dbsCountries.setFromClause("V_COUNTRIES");
      dbsCountries.setWritebackTable("COUNTRIES");
      dbsCountries.open();

      put("countries", dbsCountries);
   }

   return dbsCountries;
}

/**
 * Returns the districts storage.
 *
 * @return the Districts storage
 * @throws Exception if the initialization throws an error
 */
public DBStorage getDistricts() throws Exception
{
   DBStorage dbsDistricts = (DBStorage)get("districts");

   if (dbsDistricts == null)
   {
      dbsDistricts = new DBStorage();
      dbsDistricts.setDBAccess(getDBAccess());
      dbsDistricts.setFromClause("V_DISTRICTS");
      dbsDistricts.setWritebackTable("DISTRICTS");
      dbsDistricts.open();

      put("districts", dbsDistricts);
   }

   return dbsDistricts;
}

/**
 * Returns the states storage.
 *
 * @return the States storage
 * @throws Exception if the initialization throws an error
 */
```

```
public DBStorage getStates() throws Exception
{
   DBStorage dbsCountries = (DBStorage)get("states");

   if (dbsCountries == null)
   {
      dbsCountries = new DBStorage();
      dbsCountries.setDBAccess(getDBAccess());
      dbsCountries.setFromClause("V_STATES");
      dbsCountries.setWritebackTable("STATES");
      dbsCountries.open();

      put("states", dbsCountries);
   }

   return dbsCountries;
}
```

The view definition for processing the data:

```
CREATE VIEW V_COUNTRIES AS
SELECT c.ID
      ,c.COUNTRY
      ,c.EU
  FROM COUNTRIES c
 ORDER BY c.COUNTRY

CREATE VIEW V_STATES AS
SELECT s.ID
      ,s.CTRY_ID
      ,s.STATE
  FROM STATES s
 ORDER BY s.STATE

CREATE VIEW V_DISTRICTS AS
SELECT d.ID
      ,d.STAT_ID
      ,d.DISTRICT
  FROM DISTRICTS d
 ORDER BY d.DISTRICT
```

We now have to create the connection to the server objects and define the master/detail relationships:

```
rdbCountries.setDataSource(dataSource);
rdbCountries.setName("countries");
rdbCountries.open();

rdbStates.setDataSource(dataSource);
rdbStates.setName("states");
rdbStates.setMasterReference(new ReferenceDefinition(new String[]
```

```
{"CTRY_ID"},
                                    rdbCountries, new String[] {"ID"}));
rdbStates.open();

rdbDistricts.setDataSource(dataSource);
rdbDistricts.setName("districts");
rdbDistricts.setMasterReference(new ReferenceDefinition(new String[]
{"STAT_ID"},
                                       rdbStates, new String[] {"ID"}));
rdbDistricts.open();
```

A master/detail relationship can be created as follows:

```
rdbDistricts.setMasterReference(new ReferenceDefinition(new String[]
{"STAT_ID"},
                                       rdbStates, new String[] {"ID"}));
```

The master/detail relationship between countries and states is created via the foreign key (STATES.CTRY_ID) to the primary key (COUNTRIES.ID). This means that when a country is selected, all associated states are displayed. States that are not assigned to the selected “master” country are not shown.

When a new state is entered, the correct foreign key is used automatically. The state is, therefore, assigned to the correct country.

The master/detail relationship between districts and states is created between the foreign key (DISTRICTS.STAT_ID) and the primary key (STATES.ID). This means that when a state is selected, all associated districts are displayed. Districts that are not assigned to the selected “master” state are not shown.

When a new district is entered, the correct state is used. The district is, therefore, automatically assigned to the correct state.

Note

A master/detail relationship does NOT require that a foreign key relationship exist between two tables. Any columns can be used.