# Table of Contents

Mit Master/Detail Beziehungen werden Abhängigkeiten zwischen Tabellen bzw. Daten hergestellt. Der Master kann auch als Gruppierung angesehen werden und das Detail enthält die Einzelsätze aus der Gruppe.

Kurzum werden mit Master/Detail die 1:n und n:m Beziehungen von Datenbanken aufgelöst.

#### Anwendungsbeispiel

Unsere Applikation verwaltet Länder, Bundesländer und Bezirke. In der Datenbank werden die Tabellen in 3. Normalform erstelle. Nachfolgend ein Auszug aus der HSQLDB:DB:

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)
)
```

Wie anhand der Tabellendefinition zu sehen ist, sind Länder in Bundesländer eingeteilt und Bundesländer in Bezirke. In unserer Applikation erstellen wir eine Maske für die Bearbeitung der Länder, Bundesländer und Bezirke sowie deren Abhängigkeiten.

Wir erstellen zuerst unsere Server Objekte:

```
/**
 * 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;
}
```

Die View Definitionen für die Aufbereitung der Daten:

```
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
```

In unserer Maske müssen wir nun die Verbindung zu den Server Objekten herstellen und die Master/Detail Beziehungen definieren:

```
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();
```

Eine Master/Detail Beziehung wird beispielsweise hergestellt mit:

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

Die Master/Detail Beziehung zwischen Bundesländern und Ländern wird über den Fremdschlüssel (STATES.CTRY_ID) zum Primärschlüssel (COUNTRIES.ID) hergestellt. Das bedeutet daß bei der Selektion eines Landes alle Bundesländer angezeigt werden. Sämtliche Bundesländer die nicht dem gewählten “Master” Land zugeordnet wurden, werden nicht angezeigt.

Wenn ein neues Bundesland eingegeben wird, wird automatisch der richtige Fremdschlüssel verwendet. Das Bundesland wird somit automatisch dem richtigen Land zugeordnet.

Die Master/Detail Beziehung zwischen Bezirken und Bundesländern wird über den Fremdschlüssel (DISTRICTS.STAT_ID) zum Primärschüssel (STATES.ID) hergestellt. Das bedeutet daß bei der Selektion eines Bundeslandes alle Bezirke angezeigt werden. Sämtliche Bezirke die nicht dem gewählten “Master” Bundesland zugeordnet wurden, werden nicht angezeigt.

Wenn ein neuer Bezirk eingegeben wird, wird automatisch das richtige Bundesland verwendet. Der Bezirk wird somit automatisch dem richtigen Bundesland zugeordnet.

## Hinweis

Eine Master/Detail Beziehung setzt NICHT voraus das eine Fremdschlüsselbeziehung zwischen zwei Tabellen besteht. Es können beliebige Spalten verwendet werden.