JDBC: DatabaseMetaData

Last update: 2014-06-23

Through the `java.sql.DatabaseMetaData` interface you can obtain meta data about the database you have connected to. For instance, you can see what tables are defined in the database, and what columns each table has, whether given features are supported etc.

The `DatabaseMetaData` interface contains a lot of methods, and not all of them will be covered in this tutorial. You should check out the JavaDoc's. This text will just cover enough to give you a feeling for what you can do with it.

Obtaining a DatabaseMetaData Instance

You obtain the `DatabaseMetaData` object from a `Connection`, like this:

```
DatabaseMetaData databaseMetaData = connection.getMetaData();
```

Once you have obtained this `DatabaseMetaData` instance, you can call methods on it to obtain the meta data about the database.

Database Product Name and Version

You can obtain the database product name and version, like this:

```
int    majorVersion   = databaseMetaData.getDatabaseMajorVersion();
int    minorVersion   = databaseMetaData.getDatabaseMinorVersion();

String productName    = databaseMetaData.getDatabaseProductName();
String productVersion = databaseMetaData.getDatabaseProductVersion();
```

If you already know exactly what database your application is running against, you may not need this. But, if you are developing a product that needs to be able to run against many different database products, this information can be quite handy in determining what database specific features it supports, SQL it supports etc.

Database Driver Version

You can obtain the driver version of the JDBC driver used, like this:

```
int driverMajorVersion = databaseMetaData.getDriverMajorVersion();
int driverMinorVersion = databaseMetaData.getDriverMinorVersion();
```

Again, if your application runs against a very specific database, this may not really be so informative. However, for applications that need to be able to run against many different database products and versions, knowing the exact version of the used driver may be an advantage. For instance, a certain driver version may contain a bug that the application need to work around. Or, the driver may be missing a feature that the application then needs to work around.

Listing Tables

You can obtain a list of the defined tables in your database, via the `DatabaseMetaData`. Here is how that is done:

```
String   catalog          = null;
String   schemaPattern    = null;
String   tableNamePattern = null;
String[] types            = null;

ResultSet result = databaseMetaData.getTables(
    catalog, schemaPattern, tableNamePattern, types );

while(result.next()) {
    String tableName = result.getString(3);
}
```

First you call the `getTables()` method, passing it 4 parameters which are all null. The parameters can help limit the number of tables that are returned in the `ResultSet`. However, since I want all tables returned, I passed null in all of these parameters. See the JavaDoc for more specific details about the parameters.

The `ResultSet` returned from the `getTables()` method contains a list of table names matching the 4 given parameters (which were all null). This `ResultSet` contains 10 columns, which each contain information about the given table. The column with index 3 contains the table name itself. Check the JavaDoc for more details about the rest of the columns.

Listing Columns in a Table

You can obtain the columns of a table via the `DatabaseMetaData` too. Here is how:

```
String   catalog           = null;
String   schemaPattern     = null;
String   tableNamePattern  = "my_table";
String   columnNamePattern = null;


ResultSet result = databaseMetaData.getColumns(
    catalog, schemaPattern,  tableNamePattern, columnNamePattern);

while(result.next()){
    String columnName = result.getString(4);
    int    columnType = result.getInt(5);
}
```

First you call the `getColumns()` method, passing 4 parameters. Of these, only the `tableNamePattern` is set to a non-null value. Set it to the name of the table you want to obtain the columns of.

The `ResultSet` returned by the `getColumns()` method contains a list of columns for the given table. The column with index 4 contains the column name, and the column with index 5 contains the column type. The column type is an integer matching one of the type constants found in `java.sql.Types`

To get more details about obtaining column information for tables, check out the JavaDoc.

Primary Key for Table

It is also possible to obtain the primary key of a table. You do so, like this:

```
String   catalog   = null;
String   schema    = null;
String   tableName = "my_table";

ResultSet result = databaseMetaData.getPrimaryKeys(
    catalog, schema, tableName);
```

```
while(result.next()){
    String columnName = result.getString(4);
}
```

First you call the `getPrimaryKeys()` method, passing 3 parameters to it. Only the `tableName` is non-null in this example.

The `ResultSet` returned by the `getPrimaryKeys()` method contains a list of columns which make up the primary key of the given table. The column with index 4 contains the column name.

A primary key may consist of multiple columns. Such a key is called a compound key. If your tables contains compound keys, the `ResultSet` will contain multiple rows. One row for each column in the compound key.

Supported Features

The `DatabaseMetaData` object also contains information about the features the JDBC driver and the database supports. Many of these features are represented by a method you can call, which will return true or false depending on whether the given feature is supported or not.

I will not cover all the feature support related methods here. I will just give you a few examples. Consult the JavaDoc for a full list of feature support methods, and their meaning.

```
databaseMetaData.supportsGetGeneratedKeys();

databaseMetaData.supportsGroupBy();

databaseMetaData.supportsOuterJoins();
```