

Using the Derby Database

Posted At : June 8, 2007 1:37 PM | Posted By : Steve

Related Categories: ColdFusion, SQL, DataMgr

When I read Ben Forta's [announcement](#) (and [follow up](#)) that ColdFusion 8 would ship with support for the [Derby database](#), my first thought was "I should add support for that in [DataMgr](#)". This proved to be a good introduction to Derby.

I decided that I would get my local copy of my [demonstration site](#) running on Derby.

Set Up

The steps [Ben Forta gives for setting up Derby](#) are:

1. Go to the "Data Sources" page of the ColdFusion Admin
2. Enter a name for your data source and select "Apache Derby Embedded" as the driver
3. Click the "Add" button
4. Copy the name of your data source into the "Database Folder" field (no need for a full path)
5. Click the "Show Advanced Settings" button
6. Type "create=true" (without the quotes) into the "Connection String" field (this step will be replaced by checking a box in the full release)
7. Click the "Submit" button

This creates both the data source and the database.

Creating Tables

Ben notes that since Derby doesn't come with a GUI, you will have to create tables with CREATE statements. A nice aspect of DataMgr is that it will do that for you if you want.

Which meant that I had to figure out the syntax for creating tables in Derby. In theory, create syntax should be the same regardless of database. In practice, this often isn't the case.

Here is an example of create syntax in Derby (creating the table used by the "Events" portion of the site):

```
CREATE TABLE evtEvents (
  EventID int NOT NULL GENERATED ALWAYS AS IDENTITY,
  EventDate timestamp,
  Title varchar(50) DEFAULT 'Steve',
  LongDescription timestamp,
  DateEntered timestamp NOT NULL,
  PRIMARY KEY (EventID)
)
```

The default is in just for demonstration purposes. Notice that "NULL" never appears by itself. If you try to tell Derby that a field should allow nulls, it throws an error. It assumes a field will allow nulls so you only need to tell it if it can't.

The syntax for an identity field is a bit unusual. I imagine that this allows for more flexibility, but I haven't really looked into it yet.

Getting Meta Data

As DataMgr uses database introspection, I needed to find out what tables and columns are in the database from Derby. According to the [documentation](#), Derby will retrieve this information using "SHOW TABLES" and "SHOW COLUMNS" respectively. No luck. Another entry says that "DESCRIBE TABLES" and "DESCRIBE COLUMNS" should work. Still no luck.

This failing could have been because of the version of Derby that is installed with ColdFusion, but I wasn't able to locate that information so I can't say for sure.

Fortunately, ColdFusion 8 comes with the `cfdbinfo` tag which (as [Brian Rinaldi explains](#)) provides the information that I need.

Escaping

So far so good. My create table example doesn't pass all of my requirements for DataMgr, however.

DataMgr allows for invalid table and column names, so my create statement must as well. For example, a column must be able to be a reserved word or have spaces. This is a simple matter of finding out how use escaping in Derby:

```
CREATE TABLE "evtEvents" (
  "EventID" int NOT NULL GENERATED ALWAYS AS IDENTITY,
  "EventDate" timestamp,
  "Title" varchar(50) DEFAULT 'Steve',
  "LongDescription" CLOB,
  "DateEntered" timestamp NOT NULL,
  PRIMARY KEY ("EventID")
)
```

That solves the problem.

Unfortunately, Derby proves to be case-sensitive (in the sort of odd way that Oracle is). When I didn't escape my table and field names, it converted them to upper case. When I escaped them, it didn't. This would be fine except that it caused `cfdbinfo` to fail.

It seems that `cfdbinfo` converts the value of the "table" attribute for `type="columns"` to upper case and I couldn't figure out a way to convince it not to.

I could use Derby's sys tables to get the information that `cfdbinfo` provides, but it seemed easier to just capitulate and use upper case. This turns out to work well for another principle which is that SQL that I write by hand still works as I expect.

So, my escape method for Derby now converts to upper case, resulting in the following create statement.

```
CREATE TABLE "EVTEVENTS" (
  "EVENTID" int NOT NULL GENERATED ALWAYS AS IDENTITY,
  "EVENTDATE" timestamp,
  "TITLE" varchar(50) DEFAULT 'Steve',
  "LONGDESCRIPTION" CLOB,
  "DATEENTERED" timestamp NOT NULL,
  PRIMARY KEY ("EVENTID")
)
```

Syntax Oddities

Now everything should work fine, right? Well, almost. It turns out that I put a ";" at the end of my insert statement in DataMgr. No other database complains, but Derby did. So, out it goes.

That got the "Events" example working, but not the examples of using Relation Fields.

It turns out that Derby doesn't have a way *in sql* to limit the number of rows returned by a query. This may not seem like a major issue as you could always use the `maxrows` attribute of `cfquery` (made easier by the [new `attributeCollection` attribute](#)). It is a big deal, however, if you are using subqueries.

In my case, I typically have an ORDER BY statement in my subqueries (just part of how DataMgr works). Derby, however, throws a fit on this. Other databases don't, because I limit the number of rows in combination with the ORDER BY statement which makes it acceptable. It wasn't too difficult to remove the ORDER BY from the subqueries, but it is a disappointing omission from the database.

Casting

Another thing to note is that you can't cast from a numeric data type to a varchar, but you can cast from a numeric data type to a char (though you do have to specify a length):

CAST(myfield AS char(50))

Conclusion

I am really glad that ColdFusion comes with the option of a Derby database. It will be nice to know that it will be available for any ColdFusion application running on CF8.

The Derby database does have some oddities (and is missing any way to limit the number of records returned in SQL), but they are relatively easy to work around.

Hopefully my experience will help you out if and when you decide to try out Apache Derby.