Keeping two different databases in sync

A little example of how to keep two tables in sync between two different databases (that may run on different platforms/maachines/sites) with MsAccess.

Imagine you have this situation, an application at your office that keeps track of your inventory, sales, accountring … based on a full fledged, transactional, database (Firebird 1.5 in my case) as the source, and a web site, on a remote computer, powered by a fast database (MySQL 4.0.17) as the destination, that shows all the items available in your catalog to visitors.

The challenge is to keep those databases aligned without much hassle, and here I'll show you how:

First of all create a fictional "items" table in both databases, with similar data structures (I created those tables in the standard example databases provided, EMPLOYEE.FDB for Firebird and TEST for MySQL):

Firebird:

```
CEATE TABLE "ITEMS"
(
  "ITEM_CODE"        VARCHAR(50) NOT NULL,
  "ITEM_NAME"        VARCHAR(255),
  "PRICE"       FLOAT,
 PRIMARY KEY ("ITEM_CODE")
);
```

MySQL:

```
CREATE TABLE `items` (
        `ITEM_CODE` varchar(50) NOT NULL default '',
        `ITEM_NAME` varchar(255) default NULL,
        `PRICE` float default NULL,
        PRIMARY KEY  (`ITEM_CODE`)
        TYPE=MyISAM
```

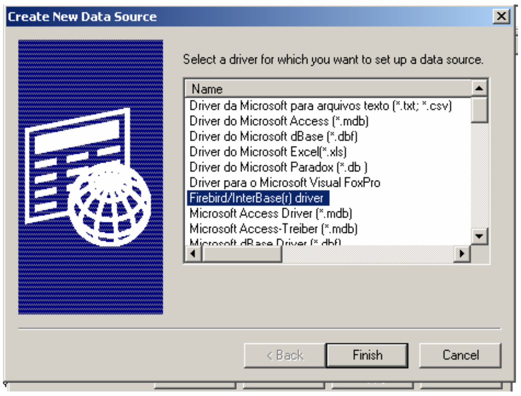
Then add some data to Firebird:

```
INSERT INTO items (item_code, item_name, price) VALUES ('ABZ01', 'First widget', 12,5);

INSERT INTO items (item_code, item_name, price) VALUES ('ABZ02', 'Second widget', 20);
```

Now set up two ODBC connections:

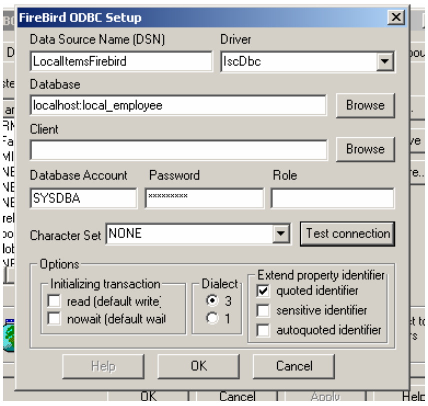
One to Firebird:



Create New Data Source

Select a driver for which you want to set up a data source.

Name
Driver da Microsoft para arquivos texto (*.txt; *.csv)
Driver do Microsoft Access (*.mdb)
Driver do Microsoft dBase (*.dbf)
Driver do Microsoft Excel(*.xls)
Driver do Microsoft Paradox (*.db)
Driver para o Microsoft Visual FoxPro
Firebird/InterBase(r) driver
Microsoft Access Driver (*.mdb)
Microsoft Access-Treiber (*.mdb)
Microsoft dBase Driver (*.dbf)

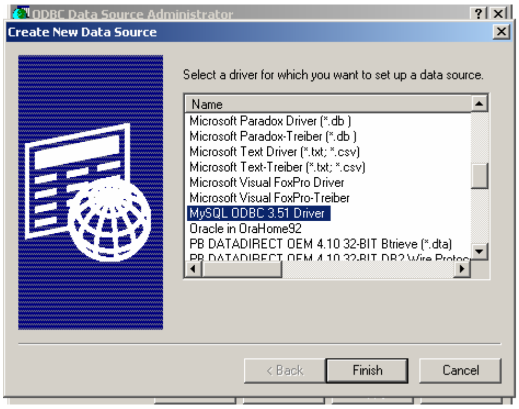


FireBird ODBC Setup

Data Source Name (DSN): LocalItemsFirebird
Driver: IscDbc
Database: localhost:local_employee
Client:
Database Account: SYSDBA
Password: *********
Role:
Character Set: NONE

Options
Initializing transaction: □ read (default write) □ nowait (default wait)
Dialect: ● 3 ○ 1
Extend property identifier: ☑ quoted identifier □ sensitive identifier □ autoquoted identifier

And another one to MySQL:

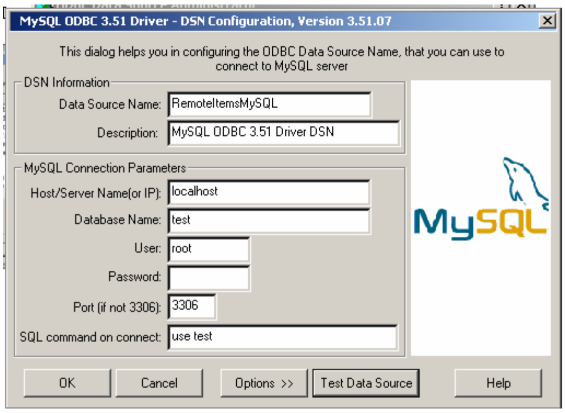


ODBC Data Source Administrator

Create New Data Source

Select a driver for which you want to set up a data source.

Name
Microsoft Paradox Driver (*.db)
Microsoft Paradox-Treiber (*.db)
Microsoft Text Driver (*.txt; *.csv)
Microsoft Text-Treiber (*.txt; *.csv)
Microsoft Visual FoxPro Driver
Microsoft Visual FoxPro-Treiber
MySQL ODBC 3.51 Driver
Oracle in OraHome92
PB DATADIRECT OEM 4.10 32-BIT Btrieve (*.dta)
PB DATADIRECT OEM 4.10 32-BIT DB2 Wire Protoc



MySQL ODBC 3.51 Driver - DSN Configuration, Version 3.51.07

This dialog helps you in configuring the ODBC Data Source Name, that you can use to connect to MySQL server

DSN Information
Data Source Name: RemoteItemsMySQL
Description: MySQL ODBC 3.51 Driver DSN

MySQL Connection Parameters
Host/Server Name(or IP): localhost
Database Name: test
User: root
Password:
Port (if not 3306): 3306
SQL command on connect: use test

Ok, test both datasources to check that all parameters are correct!!

If everything is up and running you can now fire MsAccess and create a new database, I’ll call it “sync_firebird_mysql.mdb”.

Now create two linked tables from ODBC datasources, one Firebird_ITEMS and the other one MySQL_ITEMS.
With the two tables created browse them, to check that all is working fine, Firebird_ITEMS should contain data, while MySQL_ITEMS should be empty.
Now we have to create a query to detect differencies between the two tables, the simples query can be generated with the usual MsAccess’s wizard, we want a query to find non matched records, otherwise we can write it down by hand:

SELECT Firebird_ITEMS.ITEM_CODE, Firebird_ITEMS.ITEM_NAME, Firebird_ITEMS.PRICE

```
FROM Firebird_ITEMS LEFT JOIN MySQL_ITEMS ON
Firebird_ITEMS.ITEM_CODE = MySQL_ITEMS.ITEM_CODE
WHERE (((MySQL_ITEMS.ITEM_CODE) Is Null));
```

And save it as “Firebird_ITEMS_not_matched_in_MySQL_ITEMS”

Running it shows us the full content of the Firebird_ITEMS table, as MySQL_ITEMS is empty, those are the records to be transferred.

Now build a second query, that will pick up all records found by the first one and insert them into MySQL_ITEMS, again this query can be generated with the wizard or written by hand, the syntax is:

```
        INSERT INTO MySQL_ITEMS ( ITEM_CODE, ITEM_NAME, PRICE )
SELECT Firebird_ITEMS_not_matched_in_MySQL_ITEMS.ITEM_CODE,
Firebird_ITEMS_not_matched_in_MySQL_ITEMS.ITEM_NAME,
Firebird_ITEMS_not_matched_in_MySQL_ITEMS.PRICE
FROM Firebird_ITEMS_not_matched_in_MySQL_ITEMS;
```

And save it as “Update_MySQL_ITEMS”.

Now run it!! You’ll get a warning about rows being added to table, now check MySQL_ITEMS, you’ll find that it is filled with data.

Now add a row to the ITEMS table stored in Firebird and commit transaction:

```
        INSERT INTO items (item_code, item_name, price) VALUES ('AZ03', 'Third
widget', 13);
```

Ok, I expect the “Firebird_ITEMS_not_matched_in_MySQL_ITEMS” query to show only this new item added, let’s run it!! And yes, it contains just one row!!

To realign the two databases I’ll run “Update_MySQL_ITEMS”, it warns me that data will be added and opening MySQL_ITEMS shows that the new item has been added.

Note that this is only a partial implementation of a syncing tool as it is unable to detect and update changed rows, you can easily verify this by running the two statements below on Firebird, however it can detect correctly new and deleted items.

```
update items set item_name = 'Third widget modified' where item_code = 'AZ03'

update items set price = 20 where item_code = 'ABZ02'
```

Of course this is only a starting point, you can find out which records are new and which are only changed with another view and then write some code to update each row.