Migrating from Microsoft® Access to the MySQL database

By Paul DuBois for NuSphere Corporation

TABLE OF CONTENTS

Businesses that use Microsoft Access eventually begin to struggle against its limitations. Now they have an alternative: *Move data repositories to a system that provides better performance and reliability and that is more flexible in how it allows information to be used.* NuSphere has the technology you need to follow this course, Gemini the table type for the MySQL database. Together, Gemini and the MySQL database provide a business strength database server that runs under Windows, Linux, or Unix. Gemini provides performance and reliability enhancements such as row-level locking, transaction support, and automatic crash recovery. By using the MySQL database as a data management platform, you can continue to manipulate your data through Access as a front end if you wish. But you can also exploit your data in other ways, for example, by using MySQL to help you establish or strengthen your web presence. To do this, deploy a web server that handles requests from people who visit your site, and tie it to the MySQL database using a web programming language.

NuSphere Pro Advantage provides all the necessary tools in a single easy to install product. In addition to the Gemini table, it includes the Apache web server and the Perl and PHP programming languages. Perl and PHP both are extremely popular for web programming, and both interface easily with the MySQL database. Included with NuSphere Pro Advantage is NuSphere® PHPEd™, NuSphere's Integrated Development Environment for PHP-based web applications. The PHPEd IDE is a tool that speeds development time of web-based applications written in PHP. PHPEd is Windows and Linux-based.

Microsoft Access or the MySQL database?

Microsoft Access is a popular data management application that allows you to store information in tables that it manages directly from the local disk. You can also use Access as a front end, that is, as an interface to information that is located elsewhere and handled by another storage management system. In this case, Access acts as a client that connects to a server that provides the data. The MySQL database system is one such storage manager; if you install the MyODBC driver, Access can make ODBC connections to MySQL database servers over the network. You can still use the contents of your tables through Access, but the tables themselves are hosted by the MySQL server.

Access has its strengths, such as an easy to use interface. Access also has its limitations—it's generally used as a personal or single-user application, typically for managing limited amounts of data. (Access is not commonly used for databases hundreds of megabytes in size, for example.) Because of its storage management limitations, you may be considering how to retain the Access interface but migrate your information to a storage manager with greater capabilities. Or you may even be considering a move away from Access entirely. This article outlines some benefits that you stand to gain by using the MySQL database to manage your data, and provides some guidelines to help you migrate locally stored Access tables to MySQL. The final section of the article lists links to locations where you can find the tools discussed here.

Reasons to Migrate from Microsoft Access to the MySQL database

The use of the MySQL database as a storage manager for Access offers several benefits. One is that you can use your information in additional ways when it's not locked into Access. Other differences pertain more specifically to the case where you intend to continue using Access as the user interface to your information.

Deployment of information. When your information resides in MySQL, you're free to continue using it from Access if you wish, but a number of other possibilities open up as well. Any kind of MySQL database client can use the information, not just Access. This allows your data to be exploited more fully in more contexts, and by more people. For example, other people can use

the data through the standard MySQL client programs or from GUI-based applications. Your database also becomes more accessible over the Web. Microsoft Access now provides some capabilities for making a database available on the Web, but if MySQL manages the database, you have a wider range of options. The MySQL database integrates easily with Web servers like Apache through any of a number of languages, such as Perl, PHP, Python, Java, and Ruby. This allows you to provide a Web interface to your database with the language of your choice. In addition, the interface can be accessed by browsers on many types of machines, providing a platform-independent entryway to your information. All of these components can be obtained for free—the MySQL database, Apache web server, and the languages just mentioned have been released as Open Source. You can also obtain them in packages that include support.

Multiple-user access. Although Access provides some data sharing capabilities, that's not really its strength. It has the feel of a single-user data manager designed for local use. MySQL, on the other hand, easily handles many simultaneous users. It was designed from the ground up to run in a networked environment and to be a multiple-user system that is capable of servicing large numbers of clients.

Management of large databases. The MySQL database can manage hundreds of megabytes of data, and more. Care to try that with Access?

Security. When Access tables are stored locally, anyone can walk up to your Windows machine, launch Access, and gain access to your tables. It's possible to assign a database a password, but many people routinely neglect to do so. When your tables are stored in a MySQL database, the MySQL server manages security. Anyone attempting to access your data must know the proper user name and password for connecting to the MySQL database.

Backup management. If you work in an organization that supports many Access users, migrating data to the MySQL database provides a benefit for backups and data integrity. With Access databases centralized in MySQL, they're all backed up using the regular MySQL backup procedures that already exist at your site. If individual Access users each store their data locally, backup can be more complicated: 50 users means 50 database backups. While some sites address this problem through the use of network backups, others deal with it by making backups

the responsibility of individual machine owners—which unfortunately sometimes means no backups at all.

Local disk storage requirements. Local Access database files become smaller, because the contents of tables are not stored internally, they're stored as links to the MySQL server where the tables reside. This results in reduced local disk usage. And, should you wish to distribute a database, less information need be copied. (Of course, anyone you distribute the database to also must have access to the MySQL database server.)

Cost. MySQL can be obtained for free. Access cannot. Providing other means of using your database (such as through a Web interface) can reduce your dependence on proprietary software and lower your software acquisition and licensing costs.

Hardware choices. MySQL runs on several platforms; Microsoft Access is a single-platform application. If you want to use Access, your choice of hardware is determined for you.

Migration Strategies

Should you wish to migrate from Access to the MySQL database, you can do so either partially or completely. It's not without reason that Microsoft Access is popular—it provides an interface that many people are comfortable working with. If you're such a user, you can continue to use the interface by migrating partially: Transfer locally stored Access tables to MySQL, then set up links in the Access database that point to the tables managed by the MySQL database server. This way you continue to enjoy the familiarity of the Access interface (the tool with which you're conversant), but also take advantage of the strengths of MySQL for data storage, management, and security.

If you're less tied to the user interface, you can migrate completely away from Access. Transfer your Access tables to MySQL, then use your information with tools intended for working with the MySQL database.

Methods of Transferring Databases from Microsoft Access to the MySQL database

In general, to migrate information from Microsoft Access to the MySQL database, you first copy the contents of your tables from an Access database to the MySQL server. (To perform the operation of transferring the tables to MySQL, you can choose from several methods, described below.) If you plan to continue using Access for the interface to your data, the next step after transferring the tables is to replace them with links: Delete the tables stored in your Access database, establish an ODBC connection from Access to the MySQL server, and recreate the tables as links to the MySQL tables. (Naturally, before you delete anything, it's prudent to make a backup first, just in case something goes wrong.) If you don't plan to continue using Access, you need not create any links.

Some transfer methods require making an ODBC connection to the MySQL database server. NuSphere products include MyODBC, the MySQL-specific ODBC driver, and can set it up for you during the NuSphere installation procedure.

Telling Microsoft Access to Export Its Own Tables

One approach to migrating data from Access to MySQL is to use the export feature provided by Access itself to write out the contents of each table as a text file. Each file then can be loaded into MySQL using a `LOAD DATA` statement or the **mysqlimport** command-line utility. Suppose you export a table `mytable` into a file `mytable.txt` using CSV (comma separated values) format, and you want to import it into a table named `mytable` in a MySQL database named `mydb`. You can invoke the **mysql** program, then issue a `LOAD DATA` statement to import the file like this:

```
C:\> mysql mydb
mysql> LOAD DATA LOCAL INFILE 'mytable.txt'
    -> INTO TABLE mytable
```

```
-> FIELDS TERMINATED BY ',' ENCLOSED BY '"'
    -> LINES TERMINATED BY '\r\n';
```

Alternatively, use **mysqlimport** from the command line (type the command all on one line):

```
C:\> mysqlimport --local --fields-terminated-by=,
        --fields-enclosed-by='"'
        --lines-terminated-by='\r\n'
        mydb mytable.txt
```

If you need to provide connection parameters such as the hostname, user name, or password, list them on the **mysql** or **mysqlimport** command line before the database name:

```
C:\> mysqlimport --local --fields-terminated-by=,
        --fields-enclosed-by='"'
        --lines-terminated-by='\r\n'
        -h some_host -p -u some_user
        mydb mytable.txt
```

The advantage of this approach is that it requires no special conversion tools. It can be used to produce data files even on machines that have no MySQL database support. (If you don't have the MySQL database client programs installed on your Access machine, create the data files, then copy them to another machine where the MySQL programs are installed and load the files into MySQL from there.) The disadvantage is that the MySQL database tables must already exist before you can load data into them, so you must issue the appropriate CREATE TABLE statements yourself. For the example just shown, that means you must already have created the table mytable in the mydb database before using either LOAD DATA or **mysqlimport**.

Converters that Generate Intermediate Files

A second approach to data transfer is to use a converter that reads an Access table and produces from it one or more files containing SQL statements that create the table for you and load data into it. Then you execute the intermediate SQL file or files using the **mysql** program. Several free converters that work like this are available, each of which takes the form of an Access module:

- `exportsql.txt`

  Works with Access95, Access97, Access2000. Exports all tables in a database, producing one file containing `DROP TABLE` statements (in case you want to remove MySQL tables created during an earlier data transfer exercise) and another file containing `CREATE TABLE` and `INSERT` statements for creating and loading the tables. The files are written to the `C:\TEMP` directory.

- `access_to_mysql.txt`

  Exports all tables in a database into a file `C:\TEMP\mysqldump.txt` containing `DROP TABLE`, `CREATE TABLE`, and `INSERT` statements to drop any existing MySQL tables and recreate them. Less sophisticated than `exportsql.txt` in terms of type conversion and handling of special characters.

- `mdb2sql.bas`

  Access97 only. Exports selected tables to files in a directory of your choosing. Writes a data file for each selected table, plus one SQL script containing `CREATE TABLE` statements for creating the tables and `LOAD DATA` statements for importing the data files into them.

Near the beginning of the source code for each of these converters, you'll find instructions that you should read, because the details of the process for generating the SQL and data files are converter-specific. Also, be sure to note any prerequisites that must be satisfied before using the converters. These include the following:

- Both `exportsql.txt` and `access_to_mysql.txt` expect to write files to the `C:\TEMP` directory, so you must create that directory if it doesn't exist:

  ```
  C:\> mkdir C:\TEMP
  ```

- Alternatively, you can modify the module source so that it writes files to another existing directory.

- If you want to use `exportsql.txt` to convert Access2000 tables, you need to enable support for the DAO (Data Access Objects) interface. From Access, go into the Visual Basic editor, select the Tools >> References menu option, then enable the "Microsoft DAO 3.6 Object Library" option in the window that comes up.

- `mdb2sql.bas` requires that you have Advanced Wizards installed, because it uses the Documenter function included in that Wizard set.

After following the export procedure for a converter that generates intermediate SQL files from Access tables, you'll end up with one or more files that need to be executed with the **mysql** program, as follows. Assuming that you want to create tables in a database named `mydb`, you can execute a SQL file `file.sql` like this:

```
C:\> mysql mydb < file.sql
```

If you need to provide connection parameters, list them on the command line before the database name:

```
C:\> mysql -h some_host -p -u some_user mydb < file.sql
```

Converters That Perform Direct Data Transfer

Some conversion tools can transfer data directly from an Access database into MySQL. That is, they create the MySQL tables for you and load the information into them as well. This avoids the need for any intermediate files. On the other hand, such tools require that you be able to connect to the MySQL server from the machine on which your Access information is stored. (This requirement is easily satisfied if you install MySQL on your Access machine.)

Tools that can perform direct data transfer are:

- MyAccess

  $30 shareware. (Non-registered copies are fully functional, but an annoyance dialog that must be dismissed pops up every five minutes.) Works with Access97, Access2000.

MyAccess is an Access add-in that allows direct transfer when you connect from Access to MySQL over an ODBC connection.

- DBTools

  Free. Works with Access97, Access2000. DBTools actually is intended primarily as an application for administering MySQL, but it includes data import capabilities that can be used to read Access databases for transfer to MySQL. (It can also read data from other sources such as Excel spreadsheets, making it particularly useful for transferring to MySQL information that is stored in a variety of formats.) Because DBTools reads Access databases directly, you can use it to migrate Access tables even if you don't have Access installed locally, as long as you have the database files containing the tables to be transferred. DBTools does not require ODBC.

- MySQLFront.

  Free. MySQLFront is similar in many ways to DBTools. It can read Access97 and Access2000 files directly. If ODBC is installed, MySQLFront can import information into MySQL from ODBC data sources over the network.

As an example how one of these tools works, here's how you'd use DBTools to perform data transfer from Access to MySQL. Begin by visiting the DBTools download page at `http://dbtools.vila.bol.com.br/`, transferring the installer (a program named `setup.exe`), and running it. This will install DBTools on your machine.

If you want to transfer Access2000 databases, you need to enable DAO. (If you don't, DBTools will crash whenever you try to open an Access2000 database.) To turn on DAO, launch DBTools (it will tell you there is no server profile; that's normal), select the Options >> Preferences menu item, and select the DAO 3.6 option. Then quit and relaunch DBTools, because DAO isn't actually activated until the next launch after you enable it.

With DBTools running, establish a connection to your MySQL server. (Click the Server icon in the toolbar or use the Server >> Add Server menu item to define a profile for the MySQL server

you want to connect to.) You must be connected to the server before you can transfer information; many of the menu items and icons in the tool bar are disabled until you establish a connection, including those related to importing data.

After connecting to MySQL, use the Import Data Wizard to select the Access database file containing the tables you want to transfer. One of the dialogs presented during this process asks you to select the file type for the kind of database you want to use. Select the Access97 type for either Access97 or Access2000 databases.

If you intend to continue using Access after transferring the tables, open the database from Access, delete the tables that you just transferred to MySQL, connect to the MySQL server, and set up links to the tables.

Conclusion

This article lays out some of the reasons you may find it advantageous to switch from storing and managing data using Access, and to use MySQL instead. These include multiple-user availability, security, and ability to manage large amounts of information. The article also describes some converters you can use to migrate your existing Access data into MySQL. If you're ready to move beyond the limitations of Access, NuSphere provides the tools you need.

Links

Both MySQL and MyODBC, the MySQL driver for ODBC, are available as part of NuSphere Pro Advantage and NuSphere Technology Platform. You can download the NuSphere Technology Platform for free, after registering, at: `http://www.nusphere.com/`.

Information about Microsoft Access may be obtained at:
http://www.microsoft.com/office/access/

Links for the conversion tools discussed in this article are available at the Contributed Software page of the MySQL Web site (`http://www.mysql.com/downloads/contrib.html`). Some of the

converters have their own pages as well. These may provide versions that are more up-to-date than those available on the MySQL Web site:

- DBTools: http://dbtools.vila.bol.com.br/
- exportsql.txt: http://www.cynergi.net/exportsql/
- MyAccess: http://www.accessmysql.com/
- MySQLFront: http://www.mysqlfront.de/

About NuSphere Corporation

NuSphere delivers the first Internet Application Platform (IAP) based on open source components, providing an integrated foundation that allows companies to deploy reliable, cost-effective, enterprise-class applications across Windows, UNIX and Linux environments. NuSphere® Advantage is an integrated software suite that pairs the reliability and cost-effectiveness of PHP, Apache, Perl and open source databases with new technology for building business-critical web applications and web services. Based in Bedford, Mass., the company's commercial software services include technical support, consulting and training. For more information, visit www.nusphere.com or call +1-781-280-4600.