A Guide for Migrating From Microsoft Access to MySQL

A MySQL/Sun Technical White Paper

July 2009

Table of Contents

1 Introduction

With the rapid growth of Open Source and MySQL in the database market, many corporations, government agencies, educational institutions, and others have begun to migrate away from traditional and expensive desktop and large-scale proprietary databases to more cost effective solutions. Of course, a migration from any database is not something to be taken lightly; this being the case, many organizations are educating themselves as to the benefits and true effort/costs of moving to an alternative database management system such as MySQL.

In particular, many MySQL customers are migrating from Microsoft Access because they have reached the conclusion that the combination of enhanced scalability (both in terms of concurrent userload and overall data volume), cost-savings, platform freedom, and feature set of MySQL make for a compelling business case to offload some or all their Access applications to the MySQL database server. In fact, for the past three years, users and customers who have participated in MySQL's global end of year survey have indicated that Microsoft Access is one of the top platforms they intend to remove and replace with MySQL.

This paper provides insight into what is needed for considering a move from Access to MySQL and presents a number of options that help make for an easy transition. Both the business and technical sides of migrating to MySQL will be dealt with, so whether you are a manager or a seasoned technical developer, business analyst, or DBA, you will find the needed answers to questions that revolve around migrating to the world's most popular open source database - MySQL.

2 Why Migrating from Access makes Good Business Sense

Before making any commitment to a technology, modern enterprises must look beyond the technical promises software vendors make and first consider the business side of acquiring a technology. While each business follows its own methodology for approving the deployment of a given piece of software, the core set of factors that normally govern acceptance are calculating the total cost of ownership and validating the viability of the software vendor.

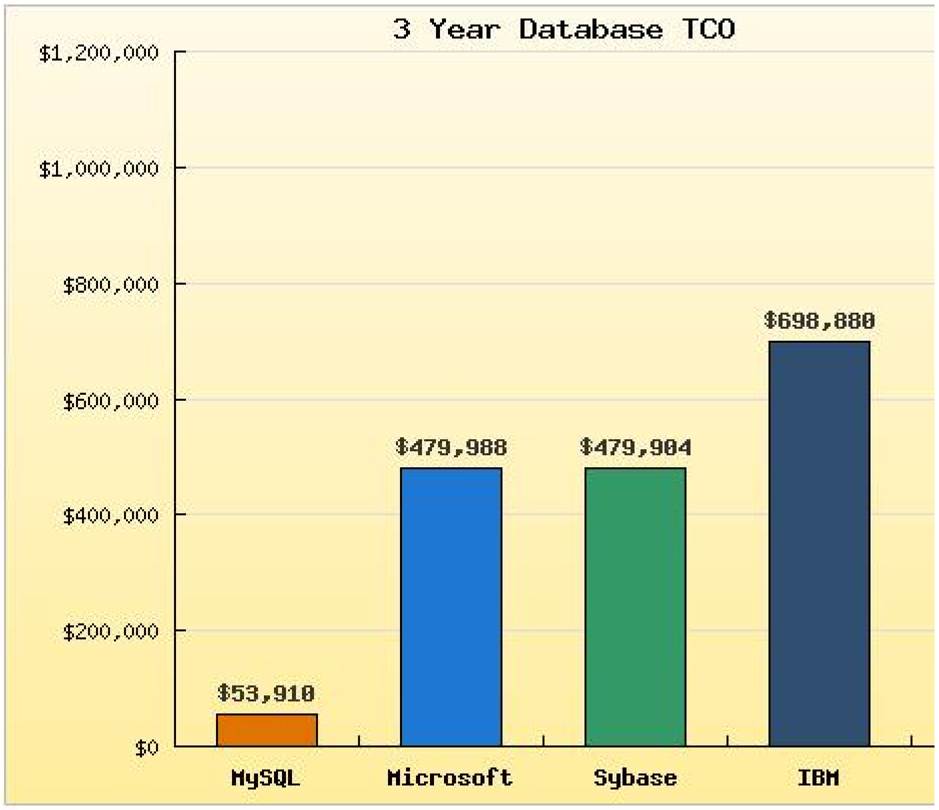
Let's take a more detailed look at each of the two factors above and see how MySQL measures up when it comes to calculating the total cost of ownership of the MySQL database server, as well as the current health of the company that stands behind that database.

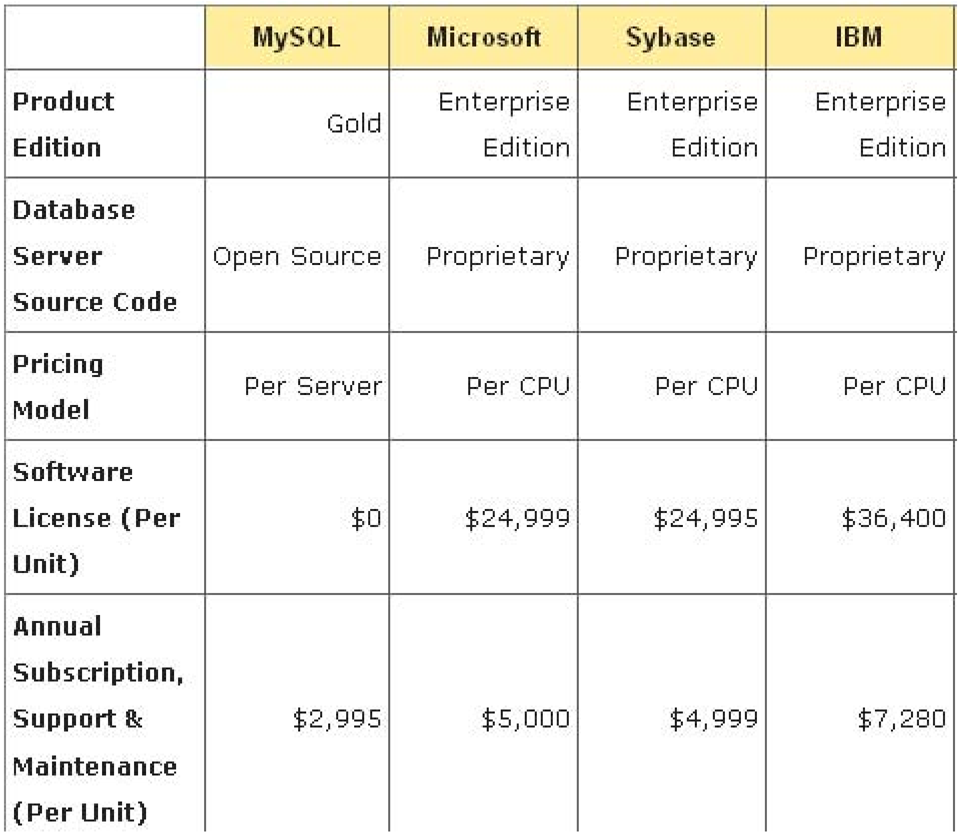
2.1 Measuring the Costs of using Microsoft Access vs. MySQL

One of the driving factors that can be attributed to the huge popularity and adoption of open source software in business today is the dramatic cost savings that accompanies owning such software. Given that many large Global 2000 enterprises spend between $500,000 and $10,000,000 in annual costs for new licenses and existing maintenance for proprietary database software, MySQL becomes an incredibly attractive alternative as most companies find they can slash their costs some 80-90% by using MySQL for new application projects as well as upgrading existing systems to use MySQL in place of its more expensive and proprietary alternatives.

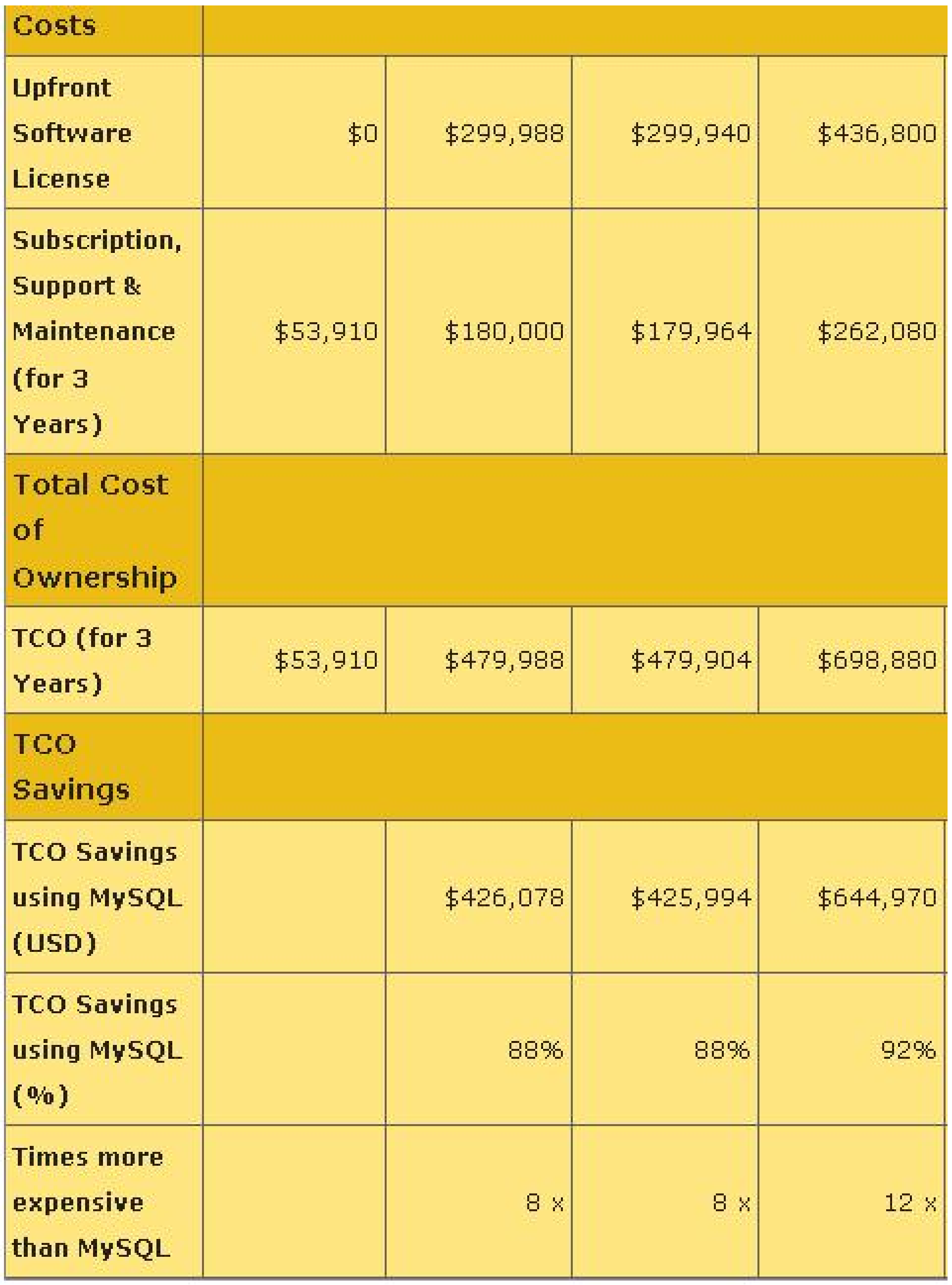
Given that Microsoft Access (as it is a desktop database solution) is usually viewed favorably in terms of cost savings when compared to other large-scale database vendors like IBM, it might be surprising to learn that migrating to MySQL from Access can still result in impressive cost savings. In the white paper,"Microsoft Access or Microsoft SQL Server: What's Right in Your Organization?" (http://www.microsoft.com/sql/solutions/migration/access/compare-access.mspx), Microsoft's own numbers show that an Access deployment can move into the $10,000 cost range for just a single user deployment and $50,000 or more for a departmental or workgroup implementation. And should an upsizing operation to SQL Server be done, the costs jump much higher.

But, consider the following three-year total cost of ownership comparison between using MySQL and SQL Server, Sybase, or IBM DB2 – the MySQL subscription not only costs much less than these options, but also costs less than a Workgroup Access deployment even after three years:





	MySQL	Microsoft	Sybase	IBM
Product Edition	Gold	Enterprise Edition	Enterprise Edition	Enterprise Edition
Database Server Source Code	Open Source	Proprietary	Proprietary	Proprietary
Pricing Model	Per Server	Per CPU	Per CPU	Per CPU
Software License (Per Unit)	$0	$24,999	$24,995	$36,400
Annual Subscription, Support & Maintenance (Per Unit)	$2,995	$5,000	$4,999	$7,280



Costs				
Upfront Software License	$0	$299,988	$299,940	$436,800
Subscription, Support & Maintenance (for 3 Years)	$53,910	$180,000	$179,964	$262,080
Total Cost of Ownership				
TCO (for 3 Years)	$53,910	$479,988	$479,904	$698,880
TCO Savings				
TCO Savings using MySQL (USD)		$426,078	$425,994	$644,970
TCO Savings using MySQL (%)		88%	88%	92%
Times more expensive than MySQL		8 x	8 x	12 x

Sample Hardware Configuration for a small Online Retailer:

- Number of Server Machines: 6 (commodity x86 machines)
- Number of CPUs (Per Machine): 2 (Dual CPU)
- Number of Cores (Per CPU): 2 (Dual Core)
- Total CPUs in deployment: 12
- Total Cores in deployment: 24

2.2 A Quick Look at MySQL

Migrating to a new database platform generally means making a long term commitment, so modern enterprises considering the move to MySQL want to know the history behind the database as well as things like who's using it and how.

Many who first happen upon MySQL are surprised to learn that MySQL has been in business for over fourteen years. With operations in over 26 countries and eleven million installations, MySQL continues to expand and grow as the world's most popular open source database. Every day, 50-70,000 people across the world download MySQL software to power their business applications, a trend that has not gone unnoticed by the major IT analyst groups.

In 2008, Gartner Group found a 50% increase from 2007 to 2008 in the usage of open source databases in production.[1] Forrester Research found that ""MySQL has the highest adoption and growth. MySQL continues to have the largest mindshare in the open source database market and has the highest number of paying customers for product support."[2]

Because of such popularity, and to support an ever-increasing enterprise customer base, MySQL offers MySQL Enterprise for customers who desire top-notch enterprise grade software, production support, and other services. For enterprise customers, MySQL Enterprise provides enterprise certified software, around-the-clock production support, advanced monitoring of all MySQL Servers regardless of geographic location, regular software updates and hot fixes, and much more. World-class IT system and solution vendors have recognized the popularity of MySQL Enterprise and have now partnered with MySQL to resell MySQL Enterprise to their large customer base. Such MySQL partner resellers now include HP, Novell, Dell, and others.

For those wishing references and validation that MySQL can be depended upon for critical application needs, one need only look at MySQL's rapidly growing customer base, which reads like a Who's-Who in the modern business community. Modern enterprises such as Yahoo, Google, Bank of America, Weather Channel, Facebook, Sage Software, Amazon, Ticketmaster, Alcatel, and many others across all business industries can be found in MySQL's customer rolodex. For more information, see http://www.mysql.com/customers/.

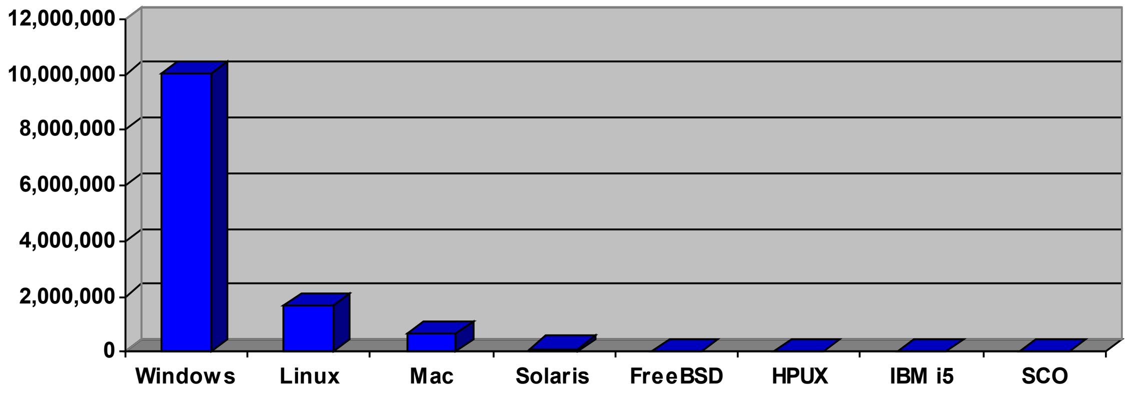
[1] The Growing Maturity of Open Source Database Management Systems, November 2008.
[2] Forrester Open Source Update, 2008.

3 The Technical Case for Migrating from Access to MySQL

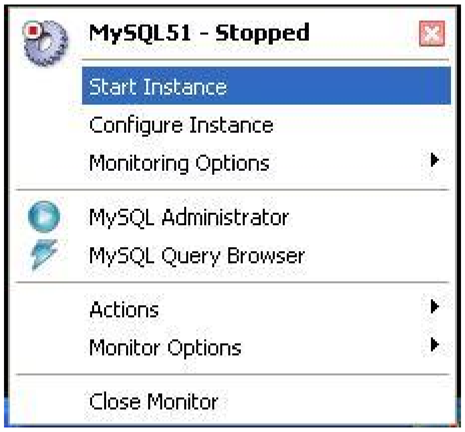
While MySQL passes the business case litmus test with flying colors, the next hurdle to overcome with respect to the question of whether to migrate to MySQL from Access is the technical review of the database feature set and capabilities. Just exactly how good can an open source database be?

3.1 MySQL on Windows? Absolutely!

While MySQL has a rich heritage of working on open source operating systems like Linux, it may be surprising to learn how popular MySQL is among those using Microsoft Windows for their database server platform and desktop database applications. Windows far exceeds any other platform for the MySQL in terms of overall downloads. For example, here are some download statistics from April 2008 to April 2009 (which includes all server versions from 5.0 on up):



production platform people use for their MySQL database, and for MySQL's enterprise paying customers, 54% said they used MySQL/Windows for development purposes, but 32% said they deploy production MySQL databases on the Microsoft platform – the 2nd most popular platform (RHEL was #1). Perhaps more surprising, the MySQL Community stated they use Windows for development 65% of the time and deploy MySQL production databases on Windows in 44% of their roll-outs, making it the number one platform for both development and production.



Why is MySQL so popular among Windows users? First, MySQL is incredibly easy to install and run on Windows. A graphical install and configuration of MySQL takes less than five minutes to install on Windows, which is faster and easier than Access. A full MySQL 5.1 GA install for Windows is only a 150MB download.

It is also easy to install multiple instances of MySQL on the same Windows machine as each instance runs as its own separate service and accepts incoming connections on its own distinct port. It is extremely easy to start/stop MySQL instances on a Windows server through either the Windows Services utility or the MySQL supplied systray System Manager.

Drivers for MySQL are also easily installed and configured on any Windows server or developer's workstation, with ODBC and .NET drivers being available. Freely supplied graphical management, development, data modeling, and migration tools are also provided from MySQL so a developer or

DBA can easily point-and-click their way through creating MySQL databases, tables, indexes, stored procedures, and other such objects.

The bottom line is that it is very easy to use MySQL on the Microsoft Windows platform. It should also be pointed out that, unlike Access, MySQL can be used on over two-dozen other platforms as well so platform lock-in never becomes an issue when it comes to using MySQL.

3.2 All the Right Features

The MySQL Server offers many features that the file-based Access database simply doesn't have. Whether you need a deeper implementation of user-based security, robust data encryption, increased levels of user concurrency (MVCC-based, row-level locking, etc.), high-availability/24x7 features, stored procedures, triggers, or more, MySQL has the things that Microsoft Access (using the Jet database engine) simply doesn't supply.

In terms of core features, DBAs and CIO's alike will be pleased to find that MySQL contains all the necessary capabilities to run the vast majority of their application needs:

Database Feature	**Available in MySQL**
Open Source	√
Available on two-dozen platforms (32 and 64 bit) including Windows: (RedHat, SuSE, Fedora, Solaris, HPUS, AIX, SCO, FreeBSD, Mac OS, Windows)	√
Pluggable Storage Engine Architecture (MyISAM, InnoDB, Merge, Memory, Archive, Cluster)	√
High-Availability Clustered Database	√
ANSI SQL, Subqueries, Joins, Cursors, Prepared Statements	√
Stored Procedures, Triggers, SQL and User-Defined Functions	√
Updateable Views	√
ACID Transactions with Commit, Rollback	√
Distributed Transactions	√
Row-level Locking	√
Snapshot/Consistent Repeatable Reads (readers don't block writers and vice-versa)	√
Server-enforced Referential Integrity	√
Strong Data type support (Numeric, VARCHAR, BLOB, etc)	√
High-Precision Numeric Data types	√
Robust Indexing (clustered, b-tree, hash, full-text)	√
Dynamic Memory Caches	√
Unique Query Cache	√
Cost-Based Optimizer	√
Unicode, UTF-8	√
XML, XPath	√
Geospatial support	√
Replication (Row-based and Statement-based)	√
Partitioning (Range, List, Hash, Key, Composite)	√
VLDB (terabytes) capable	√
High-speed, parallel data load utility	√
Online Backup with Point-in-Time Recovery	√
Automatic Restart/Crash Recovery	√
Automatic Storage Management (auto-expansion, undo management)	√
Compressed and Archive Tables	√
Information Schema/Data Dictionary	√
Robust Security (SSL, fine grained object privileges)	√
Built-in data encryption and decryption	√
Built-in Job/Task Scheduler	√
Drivers (ODBC, JDBC, .NET, PHP, etc)	√
GUI Tools (Workbench, Administrator, Query Browser, Migration Toolkit)	√

Table 1 – Core feature set of the MySQL database server

As shown above, MySQL contains a very strong feature set, and exceeds Access in many areas.

3.3 Examining MySQL Performance and Scalability

One obvious reason to switch from Microsoft Access to MySQL is for greater performance, better concurrent user management, and higher scalability. A hallmark of MySQL has been outstanding performance in all areas of database activity, whether it's in the area of transaction processing, data warehousing, or high-traffic Web sites. As has already been mentioned, many modern enterprises are realizing incredible performance gains by utilizing a scale-out architecture design coupled with MySQL, where data is replicated across many MySQL commodity servers and load balanced so even the largest end user load can be served with tremendously fast response times.

One example of MySQL performance in action is the whereivebeen.com web site, which was re-launched in April 2009. Where I've Been is a community/portal and offers a local and global platform with the ability to research and plan travel, share videos, photos, stories, as well as post and receive reviews and travel advice, all within the context of each member's preferred networks. Craig Ulliott, Where I've Been Founder and CTO commented about MySQL: "We were particularly brutal to that server, at one point we were doing well over 7000 queries per second. It is impressive how well one machine handled that much load." Using MySQL in a cloud environment proved to be not only impressive in terms of performance, but also in terms of reliability and uptime, with the Where I've Been IT staff reporting that MySQL served 48 billion queries without a single reboot and no downtime whatsoever.

3.4 MySQL – Always Up, Always On

In Microsoft's white paper on comparing Access and SQL Server (cited above), the following statement is made about Access: "File server databases using Jet may become corrupt and require regular maintenance to maintain optimal results. Even with maintenance, the chance of failure is much higher than with SQL Server." Most IT professionals admit that reliability and contribution to system uptime are two key technical requirements for any piece of software, but especially for database software. MySQL doesn't disappoint in this area and offers many features that help ensure continuous database operation.

A standalone MySQL installation is highly reliable with many MySQL installations experiencing little or no downtime over several years. As was previously discussed, many large-scale enterprises use MySQL replication to implement scale-out architectures that provide very high levels of redundancy, with the end result being completely uninterrupted operations over dozens and even hundreds of servers.

For those needing the highest possible levels of uptime, MySQL Cluster can be utilized to provide "5 9's" high availability solutions. Using a shared-nothing architecture, MySQL Cluster is used by customers who simply cannot afford for their systems to be offline at any time. MySQL Cluster is particularly well suited to applications such as telecommunications and ecommerce transactional and catalog systems. MySQL Cluster runs on Linux and UNIX platforms, with support for Microsoft Windows also just recently being added.

3.5 MySQL for Embedded, Desktop, and Departmental Applications

Embedded relational database management systems (RDBMS's) are databases that ISVs / OEMs include with their software products. These embedded databases are normally "invisible" to the end user of the software with standard database management functions handled automatically either by the database itself or through an application interface exposed through the OEM's application.

The MySQL Server has been named a leader in the overall and embedded database management market by Forrester and other analyst groups for a number of years. The reasons for this include the benefits that the open source paradigm brings to OEM's and ISV's, numerous technical advantages of the MySQL database, strong ease-of use, complete technical support, a dual-licensing model, and (as already mentioned) a very low total cost of ownership.

Sun provides a targeted offering – the MySQL Embedded Server – that is designed exactly for OEM solution deployments. The MySQL Embedded Server supplies these benefits for OEM and ISV's looking to bundle a database with their application or software-as-a-service offering:

- **High-performance** – Excellent response times and high performance for all data management operations are experienced with the MySQL Embedded Server.
- **Zero administration** – The MySQL Embedded Server requires little, if any, intervention with respect to operating within bundled software.
- **Small footprint** – Very little storage space and computing resources (memory, CPU, etc.) are required making it ideal for OEM solutions.
- **Platform independence** – Over 20 of the most popular operating systems and hardware platforms are supported.
- **Standards compliance** – Adherence to common database standards with respect to programming interfaces and language constructs is present within the MySQL Embedded Server.
- **Product support** – The Sun/MySQL worldwide support organization stands behind the MySQL Embedded Server, 24x7x365, delivering quick resolutions to inquiries and issues.
- **Ease of use** – The MySQL Embedded Server is extremely easy to use, configure, develop against, and maintain.
- **Dual licensing –** Sun/MySQL provides a dual-licensing model so there is no need for a commercial vendor to worry about their application falling under open source guidelines. Those wishing to embed or package MySQL with their application can use a commercial license of the MySQL Embedded Server that legally functions as any other commercial piece of software does.

The small footprint of MySQL and its ability to consume few resources on low-end hardware and individual PC workstations makes it also ideal for desktop database applications like those serviced by Microsoft Access. For extreme low footprints, MySQL offers an embedded library that can scale down to only 3-4MB and link directly with an application to deliver outstanding performance.

In the area of departmental applications, MySQL is ideal as a high-performant, network-efficient, full-featured database that supports all forms of Web, traditional GUI-based interfaces, and Access front ends. What oftentimes happens with Microsoft Access is, because it's simple to use, Access databases and applications begin to mushroom very quickly across a single department or multiple business units. Before long, a number of these applications become mission critical to the areas they serve, which triggers a number of problems that the original designers hadn't considered (e.g. backup/recovery of critical information). Moreover, the front ends designed in Access become more widely used with the extra workload degrading what used to be acceptable response times from the file-based Access database. At this point a tug-of-war begins with the IT/datacenter staff recognizing the importance of protecting and enabling the key information used by these various departmental applications and the originating business analysts who built the Access applications wanting to maintain control over their systems to enhance and customize it.

This is where a hybrid Access-MySQL solution can come into play. The information from all the various Access databases can be migrated under one or more MySQL server umbrellas so that the IT group can assume control of the actual data and overall database performance aspects of the situation, and the business analysts can keep their Access front end (forms, reports, etc.), manage them as they see fit, but use MySQL as their back end data management system. Such a solution becomes a win-win for both parties.

3.6 MySQL – No Limits

A number of years ago, vendors such as Microsoft came out with free "lite" databases with the original premise being that developers needed a smaller version of their enterprise databases to work with. But the move was also seen by some industry watchers as an attempt to take away some of the momentum of Open Source databases such as MySQL, which can be downloaded and used free of charge in many cases.

Although not labeled a 'lite' database, Microsoft's Access is generally targeted for low-end applications and as such, it carries with it a number of restrictions and limitations that MySQL does not suffer from. The following is a list of a few of the more important weaknesses of Microsoft Access to consider:

- Access can at most support a 2GB database, with Microsoft's documents stating that a database over 300MB may have issues; MySQL can scale up to multi-terabytes.
- Access is not able to handle many concurrent users, with Microsoft stating that 25 concurrent users is their recommended limit (see white paper referenced above) and anything above that should be migrated to a different solution; MySQL can handle thousands of concurrent users.
- File databases such as Access do not take advantage of modern hardware with many CPU's or cores; MySQL makes use of the advances in today's hardware to deliver a highly-performant database server.
- In terms of data protection, if an Access database is open and/or users are accessing it, it cannot be backed up. MySQL offers many forms of backup options while users are connected to and using the database.

While Access is typically deployed in a simple desktop environment, oftentimes the database and/or supporting application will grow and when it does, it is likely it will hit the limits mentioned above. Rather than start out with Access and then be forced to switch to another DBMS as time goes by, it is oftentimes smarter to begin with a database like MySQL that future-proof's your application.

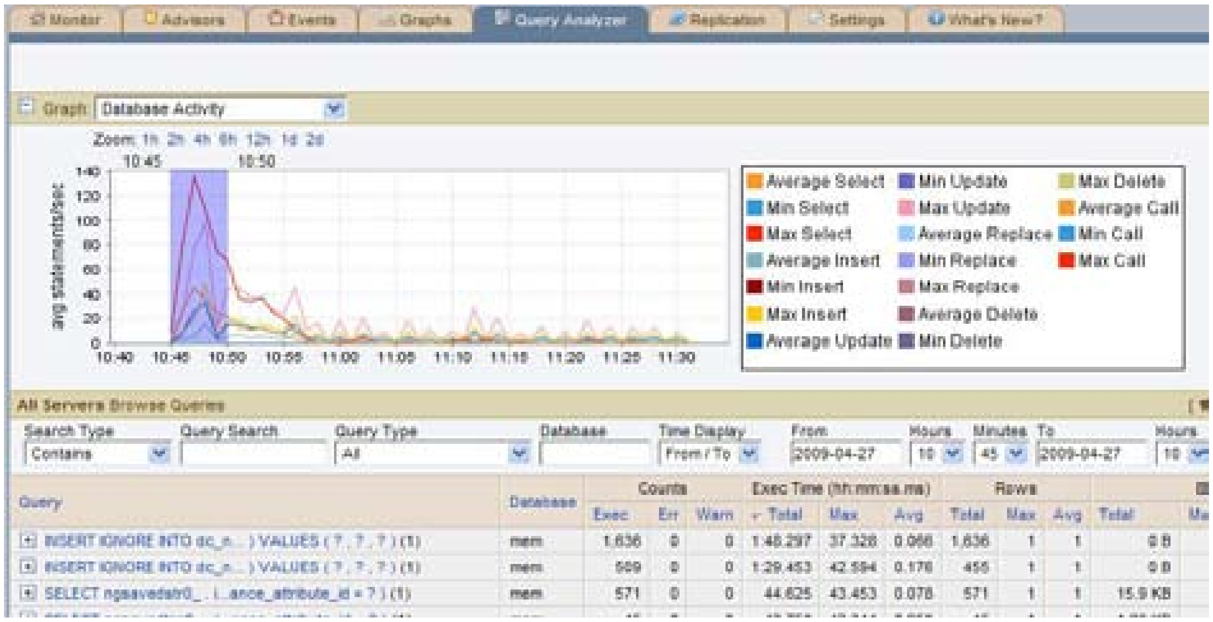
3.7 MySQL Enterprise for Production Applications

For production deployments of MySQL, it is recommended that a company subscribe to MySQL Enterprise. MySQL Enterprise contains the software and services necessary to support MySQL in mission-critical environments where a business is relying on their database-driven systems to drive their key applications. The MySQL Enterprise subscription is comprised of the following three components:

The MySQL Enterprise Server – is the most reliable, secure and up-to-date version of MySQL. MySQL Enterprise provides the added value of the update services wrapped around the MySQL Enterprise server in the form of:

- Monthly Rapid Updates
- Quarterly Service Packs
- Hot Fix Build Program
- Extended End-of-Life Program

The MySQL Enterprise Monitor with Query Analyzer – is a distributed web application that you deploy within the safety of your corporate firewall. The Monitor continually monitors all of your MySQL Servers and proactively alerts you to potential problems and tuning opportunities before they become costly outages. It also provides you with MySQL expert advice on the issues it has found so you know where to spend your time in optimizing your MySQL systems. Further, it has the ability to trace and bubble to the top all of the worst running queries across your MySQL servers so you know exactly what SQL code is slowing down your systems.



All Servers Browse Queries

Search Type	Query Search	Query Type	Database	Time Display	From	Hours	Minutes	To	Hours
Contains		All		From / To	2009-04-27	10	45	2009-04-27	10

Query	Database	Counts			Exec Time (hh:mm:ss.ms)			Rows			
		Exec	Err	Warn	Total	Max	Avg	Total	Max	Avg	Total
INSERT IGNORE INTO dc_n...) VALUES (? , ? , ?) (1)	mem	1,636	0	0	1:46.297	37.328	0.066	1,636	1	1	0 B
INSERT IGNORE INTO dc_n...) VALUES (? , ? , ?) (1)	mem	509	0	0	1:29.453	42.594	0.176	455	1	1	0 B
SELECT ngsavedatr0_ . i...ance_attribute_id = ?) (1)	mem	571	0	0	44.625	43.453	0.078	571	1	1	15.9 KB

MySQL Production Support Services – MySQL Enterprise includes 24 X 7 X 365 production support for your MySQL Servers to help ensure your business critical applications are continuously available and running at their peak. MySQL Production Support Services include:

- Online Self-Help Support – The knowledge base is a self-help tool that provides you with access to 2,000+ technical articles on MySQL specific topics that help quickly answer questions and solve problems.
- Problem Resolution Support – Allows you to work directly with the MySQL Production Support team via phone, email or an online for quick resolution of technical problems.
- Consultative Support – Allows you to work with MySQL Engineers on the proper installation, configuration and deployment of MySQL and its advanced feature set and on best practices around the design and tuning of schemas, queries and application specific code.
- Advanced Support for MySQL High Availability and Scalability Solutions – MySQL Enterprise includes full production support for additional advanced MySQL features and third-party solutions to scale the availability and performance of your online applications.

4 Practical Suggestions for Easy Migration from Access to MySQL

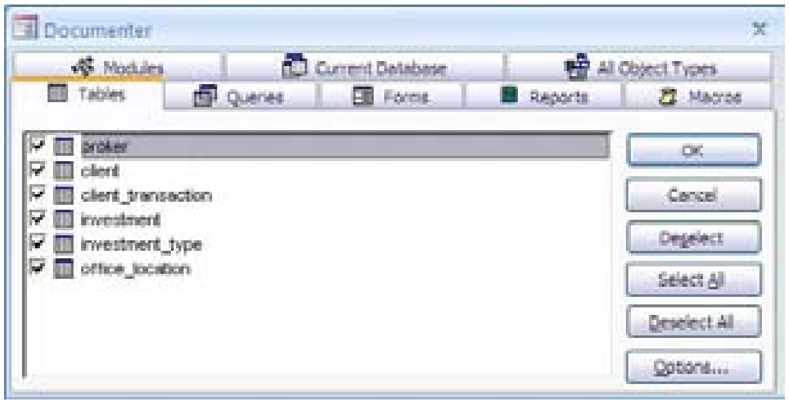
This section contains practical advice for migrating from Microsoft Access to MySQL.

4.1 All or Nothing?

Before working through each migration step, it is helpful to first point out that one very successful MySQL deployment strategy that customers have used is to move forward with MySQL for new development, or migrate existing Access applications where the Jet database is encountering performance issues or drawing close to reaching some of the limitations identified earlier. As Table 1 in this paper has demonstrated, MySQL offers the perfect balance of ease-of-use and a strong enterprise feature set for handling the vast majority of any application's database requirements. But MySQL can easily coexist in any IT infrastructure alongside Access or any other database platform because its tremendous ease of use means little to no additional management overhead.

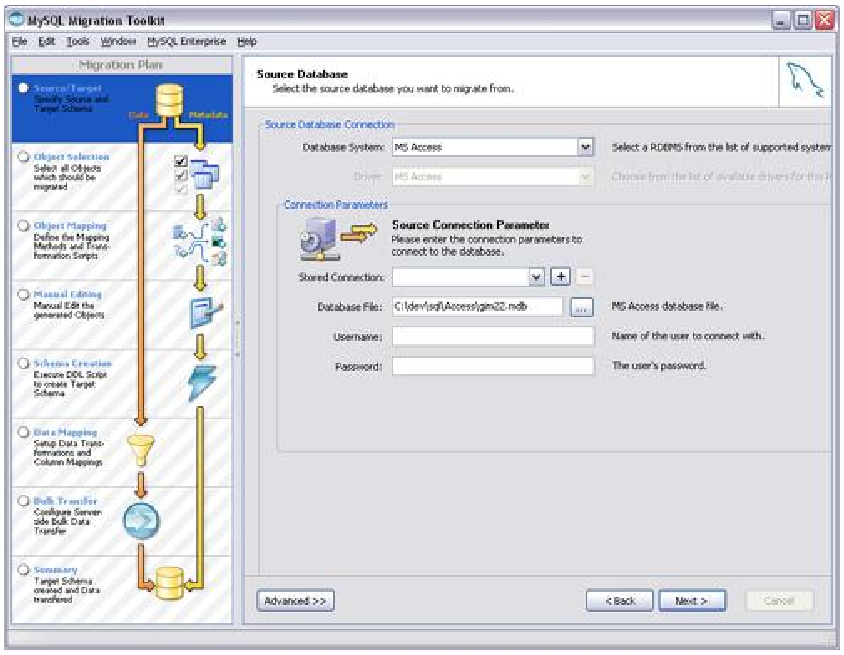
So what are the steps from moving an Access database to MySQL? Let's work through those now.

4.2 Step 1 – Document Access Sources



broker
client
client_transaction
investment
investment_type
office_location

Documenting an existing Access database is a very easy thing to do. Within Access 2007 itself, you can use the Database Documenter feature that interrogates your Access database and creates a nicely structured report detailing all your database objects and associated metadata, any queries, forms, or reports you've created, and then exports the set of documentation out to whatever format you choose (e.g. text file, MS Word RFT file).


MySQL Migration Toolkit
File
Edit
Tools
Window
MySQL Enterprise
Help
Migration Plan
Source/Target
Specify Source and Target Schema
Data
Metadata
Object Selection
Select all Objects which should be migrated
Object Mapping
Define the Mapping Methods and Transformation Scripts
Manual Editing
Manual Edit the generated Objects
Schema Creation
Execute DDL Script to create Target Schema
Data Mapping
Setup Data Transformations and Column Mappings
Bulk Transfer
Configure Server side Bulk Data Transfer
Summary
Target Schema created and Data transferred
Advanced >>
< Back
Next >
Cancel

Source Database
Select the source database you want to migrate from.

Source Database Connection

Database System: MS Access — Select a RDBMS from the list of supported systems

Driver: MS Access — Choose from the list of available drivers for this RDBMS

Connection Parameters

Source Connection Parameter
Please enter the connection parameters to connect to the database.

Stored Connection:

Database File: C:\dev\sql\Access\gim22.mdb — MS Access database file.

Username: — Name of the user to connect with.

Password: — The user's password.

Another approach is to use a good third party data modeling tool for understanding the Access source, with there being a number of good products on the market, such as Sybase's/Quest's PowerDesigner and Embarcadero's ER/Studio, that support the reverse engineering of multiple datasources such as Access.

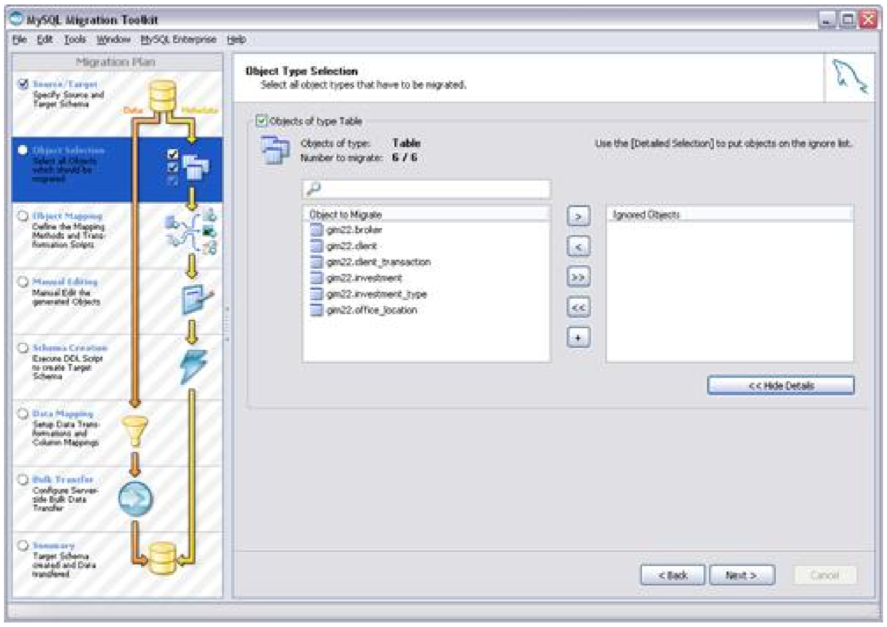
Moving data and index structures over to MySQL isn't typically a challenging task as MySQL supports all the important datatypes, table designs, and index structures. To help migrate data objects from Access to MySQL, MySQL supplies a freely downloadable graphical Migration Toolkit. The tool supplies a reverse engineering component exactly like those found in the best data modeling tools. A user first supplies the connection details to the source database system through an easy to use interface and then connects to and reverse engineers the Access databases targeted for migration.

With the MySQL Migration Toolkit, obtaining the necessary source database metadata is a very easy process and helps quickly jump start any migration effort.

4.3 Step 2 – Design MySQL Targets

Once the Access source metadata has been obtained and digested, the next step is to design the MySQL target database. This basically involves translating the source objects and their properties (such as column datatypes) to MySQL complements. As one can imagine, this step can be extremely time consuming and error-prone if attempted by hand as some Access databases may have many objects that must be converted. Appendix A and B in this paper contain datatype and function conversion suggestions when moving from Access to MySQL so be sure to reference those if you find yourself needing to do some or all of your migration by hand.

Note that many data modeling tools have the ability to convert an Access schema to a MySQL schema with only a few mouse clicks. Naturally, the models can be tweaked if need be. The automatic conversion performed by modeling tools for Access to MySQL data objects is a huge time saver and can result in lots of productivity gains for a database migration team.


MySQL Migration Toolkit
File Edit Tools Window MySQL Enterprise Help
Migration Plan
Source/Target
Specify Source and Target Schema
Data
Metadata
Object Selection
Select all Objects which should be migrated
Object Mapping
Define the Mapping Methods and Transformation Scripts
Manual Editing
Manual Edit the generated Objects
Schema Creation
Execute DDL Script to create Target Schema
Data Mapping
Setup Data Transformations and Column Mappings
Bulk Transfer
Configure Server-side Bulk Data Transfer
Summary
Target Schema created and Data transfered
< Back
Next >
Cancel
<< Hide Details

Object Type Selection
Select all object types that have to be migrated.

Objects of type Table

Objects of type: Table
Number to migrate: 6 / 6

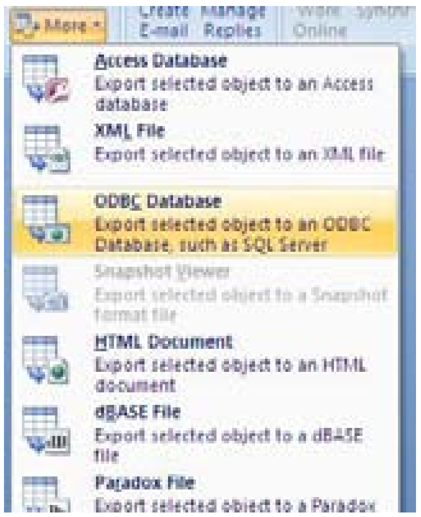
Use the [Detailed Selection] to put objects on the ignore list.

Object to Migrate	Ignored Objects
gim22.broker	
gim22.client	
gim22.client_transaction	
gim22.investment	
gim22.investment_type	
gim22.office_location	

Be aware that the MySQL Migration Toolkit automatically converts any source database reverse engineered through the tool to a MySQL counterpart, complete with all datatype translations and other like conversions. A user has complete control over exactly what objects are migrated as well as how MySQL specific designations – such as what underlying storage engine to use – are done.

As already noted, this particular step can be extremely challenging, and if not done properly, a DBA can find themselves spinning their wheels in redesigning objects or troubleshooting data transfer failures that occur because of invalid datatype mapping. However, when one uses the MySQL Migration Toolkit, such issues are a thing of the past as the tool does all the work right the first time.

4.4 Step 3 – Run Migration to MySQL


More
Create E-mail
Manage Replies
Work Online
Access Database
Export selected object to an Access database
XML File
Export selected object to an XML file
ODBC Database
Export selected object to an ODBC Database, such as SQL Server
Snapshot Viewer
Export selected object to a Snapshot format file
HTML Document
Export selected object to an HTML document
dBASE File
Export selected object to a dBASE file
Paradox File
Export selected object to a Paradox

Once the source Access metadata has been understood and the MySQL target database designed, the next step is to run the actual data migration process. The extract, transform, and load (ETL) stage can be quite elaborate depending on what one wants to accomplish, and in addition to the MySQL Migration Toolkit, there are many heavy-duty third party ETL tools on the market that offer extreme flexibility in just how to move, aggregate, map, and transform data from Access to MySQL databases. Further, Open Source ETL tools such as Pentaho and Talend have free versions of their products that can handle most data movement/transformation use cases.

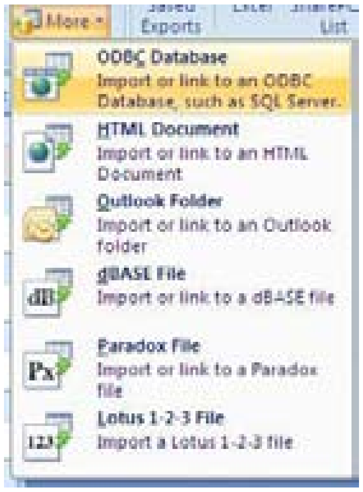
For simple and uncomplicated migrations to MySQL, the export function of Access can be utilized. You can export each table to an ODBC MySQL datasource (that you define ahead of time), and Access will take care of exporting both the table structure and data out to a MySQL database. Note, however, that if many objects are involved, it can be a tedious process to individually export each table out to MySQL. Also the export function does not export indexes defined on the Access table out to MySQL.

The MySQL Migration toolkit can come to the rescue as a more efficient option over the Access export utility as it can migrate many objects at once (including indexes and data), offers a number of options for those needing to customize the migration operation, and the price can't be beat as the Migration Toolkit is free and published under the open source GPL license. See Appendix C for a step-by-step sample migration of Access objects and data to MySQL.

For migrations needing heavy customizations, another data migration strategy is to migrate all the existing Access objects and data from the source database to a MySQL staging database. Once safely in the MySQL database server, a DBA or developer can create stored procedures or other migration code that selectively moves and manipulates data from the staging database into another MySQL database that will be used for further development or production.

An additional option is to use the scripting capabilities of the MySQL Migration Toolkit to massage the Access data as its being transferred from the Access source to the MySQL target database. Although most will use the graphical interface of Migration Toolkit to perform database migrations in point-and-click style, advanced users can exploit the underlying architecture to script out more complex ETL processes and schedule them if need be.

4.5 Using an Access Front End and MySQL Back End


More
Saved Exports
Excel
SharePoint List
ODBC Database
Import or link to an ODBC Database, such as SQL Server.
HTML Document
Import or link to an HTML Document
Outlook Folder
Import or link to an Outlook folder
dBASE File
Import or link to a dBASE file
Paradox File
Import or link to a Paradox file
Lotus 1-2-3 File
Import a Lotus 1-2-3 file

If you are using Access forms or reports and want to replace the Access/Jet database with MySQL as the back end database, the simplest option is to migrate the existing Access database to MySQL as described above and then use the Access Linked Tables option to point to the new MySQL database. Access allows you to use its forms, reports, queries, etc., to reference external datasources, and MySQL is an option available to you via the Import/ODBC option within the Access framework.

When you choose the Import option, Access will display an ODBC dialog box that allows you to select a previously created MySQL ODBC datasource, and it also gives you the option to create a new one that points to your MySQL server should one not already exist. Once connected to MySQL, you can select one or more tables to link to your Access IDE. Access will then create a link with MySQL and display the new tables in the Access tables explorer. You can view and modify the underlying table data, but you won't be able to alter the table's design (add/change columns, etc.) For that, you will have to use standard MySQL tools and/or commands.

If you do not have a MySQL ODBC connector/driver on your workstation, you can download the latest from the MySQL web site at: http://dev.mysql.com/downloads/connector/.

One disadvantage of this approach is that, with Access front ends, queries are still processed locally and there is the potential for a large amount of network traffic to be generated.

5 Conclusion

From both a business and technical perspective, the question of whether a migration to MySQL from Access makes sense is easily answered. Countless modern enterprises are enjoying huge cost savings with MySQL while at the same time powering their desktop, departmental, and enterprise database-driven systems with ease.

A summary of why to consider a move to MySQL from Microsoft Access (other than cost) includes the following:

- MySQL runs great on the Microsoft OS platform, is extremely popular as evidenced by many developing and running production MySQL databases on Windows, but MySQL can also be ported to other operating systems if desired, whereas Access cannot.
- Regarding installation and configuration, MySQL installs faster, has a smaller software footprint, but is capable of scaling much higher than Access.
- There are no database size, concurrent user, or other such limits with MySQL as those found in Access.
- MySQL storage engines provide more flexibility and offer more performance and custom application options over Access's file-based paradigm. Plus, the growing storage engine ecosystem gives MySQL great opportunity to quickly develop and innovate.
- MySQL's feature set far exceeds Access.
- If high availability is desired, MySQL has a number of proven solutions including replication, SANs, DRBD, and MySQL Cluster.
- MySQL's performance easily exceeds the Access/Jet database engine' abilities.

- The ubiquity of MySQL, the Open Source nature of the product, and its great Community provide many benefits including a great developer and DBA network of everyone working together to help ensure a high-quality product and each other's success.

Following the simple data migration steps outlined in this paper helps ensure a smooth transition from Access and ultimate success in the end. To assist with migrations from Access, MySQL offers the freely downloadable Migration Toolkit, which greatly lessens the amount of time it takes to perform database migrations and drastically reduces the amount of errors that normally result when such complex operations are attempted by hand.

Making the switch from Access to MySQL – whether done in full or in a partial format where both MySQL and Access are used for the proper application situation – can make great sense, both from a financial and a technology perspective. By following the guidelines and steps in this paper, you can be assured that you will succeed in your implementation of MySQL no matter whether you are just testing the waters with open source or have made the full commitment to make it your preferred deployment platform.

6 Resources

6.1 MySQL Migration Aids

MySQL Migration Toolkit
http://www.mysql.com/products/tools/migration-toolkit/
Graphically migrate any existing Microsoft Access, Microsoft Access, or Oracle database. Any database with a JDBC Driver can be migrated.

MySQL Migration Central
http://www.mysql.com/why-mysql/migration/
Learn how to quickly and easily migrate your existing database to MySQL.

Migration Forums
http://forums.mysql.com/
Interact with other MySQL professionals who are involved with current database migrations.

Migration Migration Jumpstart
http://www.mysql.com/consulting/packaged/migration.html
When you need expert assistance from MySQL Certified Consultants, choose the MySQL Migration Jumpstart. A dedicated, on-site MySQL certified consultant will work closely with your team to jumpstart your migration to MySQL from Oracle, Sybase, Microsoft Access, Microsoft Access, DB2, Informix, and other proprietary databases.

6.2 Other White Papers

A Guide to Lower Database TCO, MySQL
http://www.mysql.com/tco
A Computerworld article, "MySQL Breaks Into the Data Center" revealed how MySQL has become the world's most popular open source database and why corporations intent on lowering their cost of operations are using it to further commoditize their IT infrastructure. In this white paper we'll show you how. You'll also learn how organizations such as Cox Communications, NASA, Facebook and Yahoo! have improved database reliability, performance and TCO using MySQL.

A Guide to Cost Effective Scale-Out using MySQL, MySQL
http://www.mysql.com/why-mysql/white-papers/mysql_wp_scaleout.php
Countless modern enterprises are using MySQL in conjunction with commodity hardware to realize incredible cost savings and nearly unlimited application scalability. In this white paper, we’ll cover the techniques that MySQL customers are using today to power and protect their most important business applications.

7 Appendix A – Access to MySQL Datatype Conversion Suggestions

This appendix supplies general suggestions as to what datatypes in Microsoft Access may map best to MySQL datatypes.

Access Datatype	MySQL Suggested Datatype
BINARY(SIZE)	BINARY
BYTE	TINYINT
CURRENCY	NUMERIC OR DECIMAL
DATE	DATE OR DATETIME
DECIMAL	DECIMAL
DOUBLE	FLOAT
GUID	IDENTITY COLUMN
INTEGER	SMALLINT
LONGBINARY	VARBINARY
LONG INTEGER	INT OR BIGINT
MEMO	VARCHAR(SIZE) OR TEXT
SINGLE	REAL
TEXT	VARCHAR(SIZE)
YESNO	TINYINT

8 Appendix B – Access to MySQL Function Conversion Suggestions

This section provides suggestions on converting Access SQL functions to MySQL functions.

Access Function	Suggested MySQL Function
Asc	ascii
ccur(	convert(decimal,
cdbl(	convert(float,
chr	Char
Chr$	Char
Cint(	convert(smallint,
Clng(	convert(int,
csng(	convert(real,
Cstr(	convert(varchar,
cvdate(	convert(datetime/date,
Date()	Convert or cast
Day(	day
Hour(	hour
int	floor
lcase	lower
lcase$	lower
len	length
Ltrim$	Ltrim
Mid	Substr or substring
Mid$	Substr or substring
month(	month
Now()	Now()
minute(	minute
Rtrim$	Rtrim
Right$	Right
Sgn	Sign
second(	second
space$	space

Access Function	Suggested MySQL Function
time()	time
str$	strcmp
ucase	upper
ucase$	upper
weekday(	dayofweek
year(	year

9 Appendix C – Sample Migration

This section provides a simple example of moving an Access database schema over to MySQL using the MySQL Migration Toolkit.

9.1 Sample Access Schema

The Access 2007 schema is a small database used for tracking financial investments in a small brokerage company. The Access schema layout is as follows:

Table: broker

Name	Type	Size
broker_id	Long Integer	4
office_location_id	Long Integer	4
broker_last_name	Text	40
broker_first_name	Text	20
broker_middle_initial	Text	1
manager_id	Long Integer	4
years_with_firm	Text	255

Table: client

Name	Type	Size
client_id	Long Integer	4
client_first_name	Text	20
client_last_name	Text	40
client_gender	Text	1
client_year_of_birth	Long Integer	4
client_marital_status	Text	20
client_street_address	Text	40
client_postal_code	Text	10
client_city	Text	30
client_state_province	Text	40
client_phone_number	Text	25
client_household_income	Text	255
client_country	Text	40
broker_id	Long Integer	4

Table: client_transaction

Name	Type	Size
client_transaction_id	Long Integer	4
client_id	Long Integer	4
investment_id	Long Integer	4
action	Text	10
price	Text	255
number_of_units	Long Integer	4
transaction_status	Text	10
transaction_sub_timestamp	Date/Time	8
transaction_comp_timestamp	Date/Time	8
description	Text	200
broker_id	Long Integer	4
broker_commission	Text	255

Table: investment

Name	Type	Size
investment_id	Long Integer	4
investment_type_id	Long Integer	4
investment_vendor	Text	30
investment_name	Text	200
investment_unit	Text	20
investment_duration	Text	10

Table: investment_type

Name	Type	Size
investment_type_id	Long Integer	4
investment_type_name	Text	30

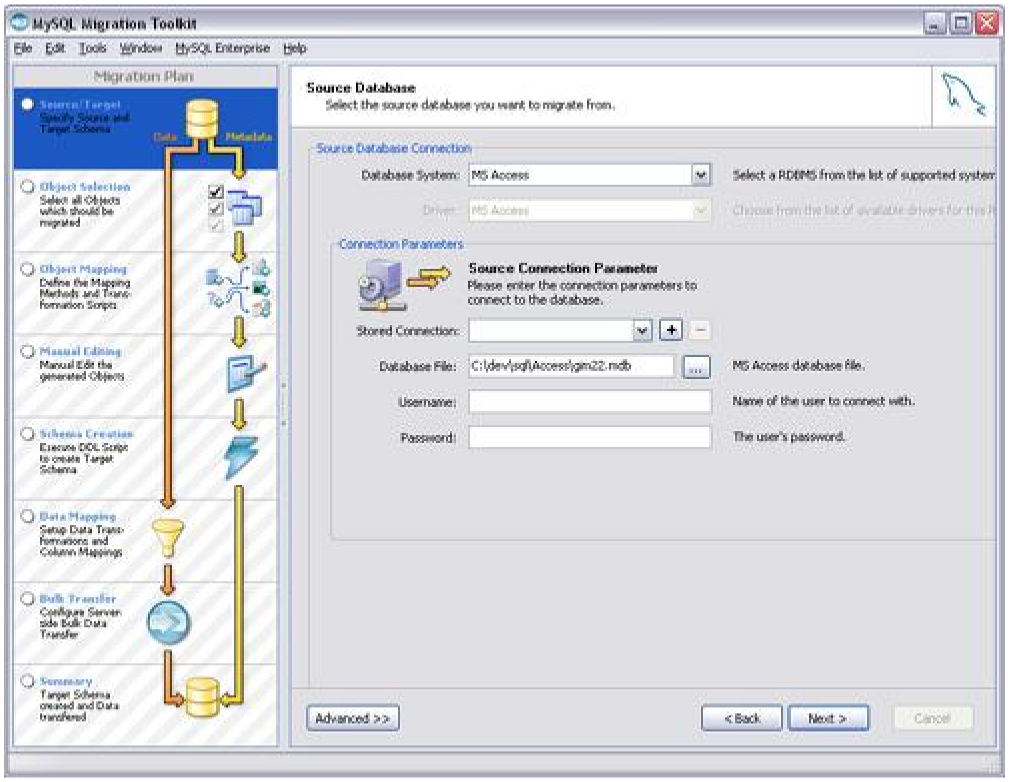
Table: office_location

Name	Type	Size
office_location_id	Long Integer	4
office_name	Text	20
office_address	Text	50
office_city	Text	30
office_state_province	Text	40
office_postal_code	Text	10
office_country	Text	40

9.2 Moving the Access Database to MySQL with the MySQL Migration Toolkit

The following steps are used to move the Access 2007 database schema (with data) over to a MySQL 5.1 database. Note that the current version of Migration Toolkit requires that you save an Access 2007 to an Access 2003 format (.mdb file) before starting the migration.

Step 1 – Connect to Access Source/database:


MySQL Migration Toolkit
File Edit Tools Window MySQL Enterprise Help
Migration Plan
Source/Target
Specify Source and Target Schema
Data
Metadata
Object Selection
Select all Objects which should be migrated
Object Mapping
Define the Mapping Methods and Transformation Scripts
Manual Editing
Manual Edit the generated Objects
Schema Creation
Execute DDL Script to create Target Schema
Data Mapping
Setup Data Transformations and Column Mappings
Bulk Transfer
Configure Server-side Bulk Data Transfer
Summary
Target Schema created and Data transfered
Advanced >>
< Back
Next >
Cancel

Source Database
Select the source database you want to migrate from.

Source Database Connection

Database System: MS Access — Select a RDBMS from the list of supported systen

Driver: MS Access — Choose from the list of available drivers for this

Connection Parameters

Source Connection Parameter
Please enter the connection parameters to connect to the database.

Stored Connection:

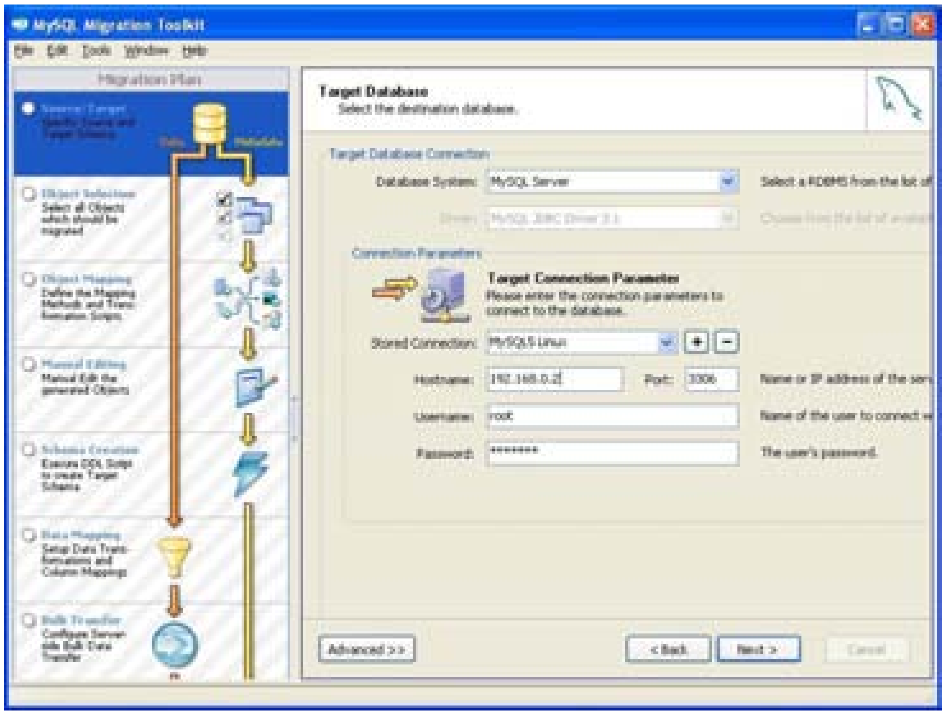
Database File: C:\dev\sql\Access\gim22.mdb — MS Access database file.

Username: — Name of the user to connect with.

Password: — The user's password.

Supply the path and filename of the Access database you wish to migrate and any security information (usually not needed).

Step 2 – Connect to MySQL Target Server:



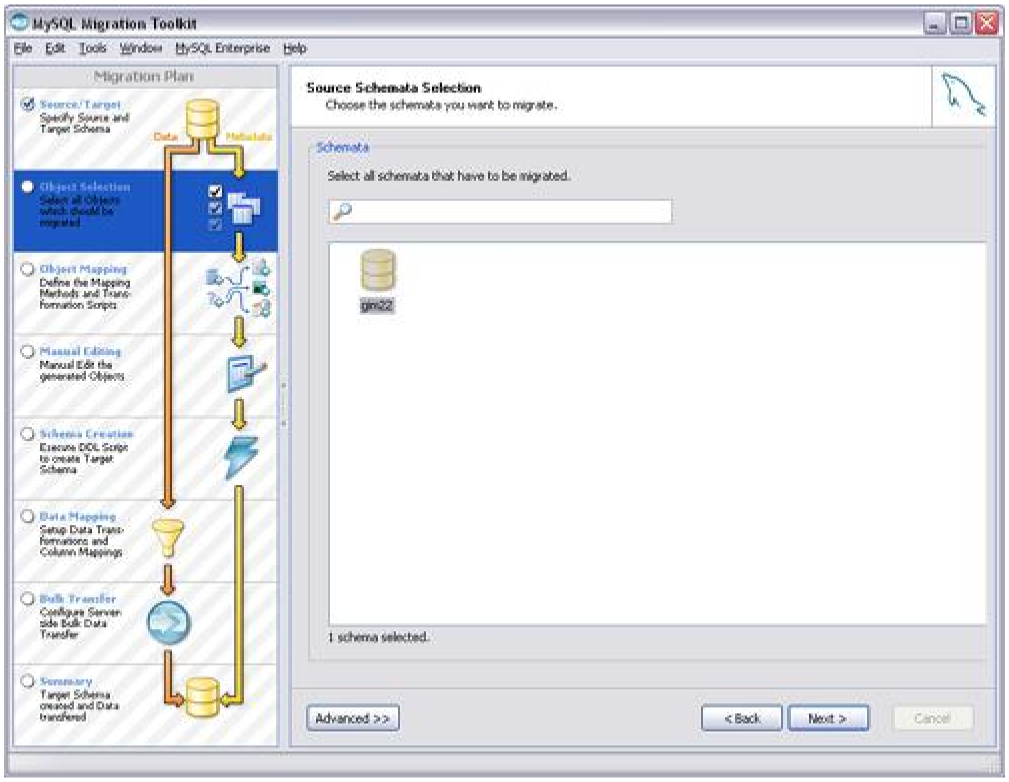
Target Database
Select the destination database.

Target Database Connection
Database System: MySQL Server — Select a RDBMS from the list of
Driver: MySQL JDBC Driver 3.1 — Choose from the list of available

Connection Parameters
Target Connection Parameter
Please enter the connection parameters to connect to the database.
Stored Connection: MySQL5 Linux
Hostname: 192.168.0.2 Port: 3306 — Name or IP address of the serv
Username: root — Name of the user to connect w
Password: ******** — The user's password.

Enter MySQL connection information for the target server and press Next. The Migration Toolkit connects to both the source and target servers.

Step 3 – Select Target Database to Migrate:



Source Schemata Selection
Choose the schemata you want to migrate.

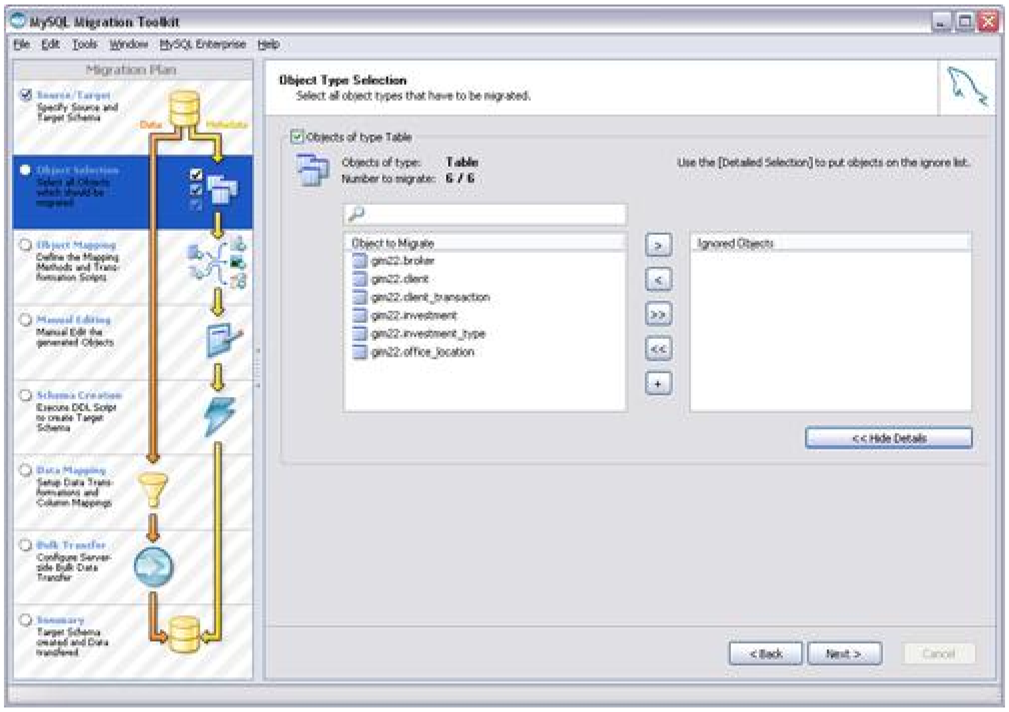
Schemata
Select all schemata that have to be migrated.

gm22

1 schema selected.

For this example, ‘gim22’ is the Access source database. The Migration Toolkit will reverse engineer the source database schema along with all datatypes, etc.

Step 4 (Optional) – Choose Objects to Migrate:



Specify Source and Target Schema

Select all Objects which should be migrated

Define the Mapping Methods and Transformation Scripts

Manual Edit the generated Objects

Execute DDL Script to create Target Schema

Setup Data Transformations and Column Mappings

Configure Server-side Bulk Data Transfer

Target Schema created and Data transfered

Object Type Selection

Select all object types that have to be migrated.

☑ Objects of type Table

Objects of type: **Table**

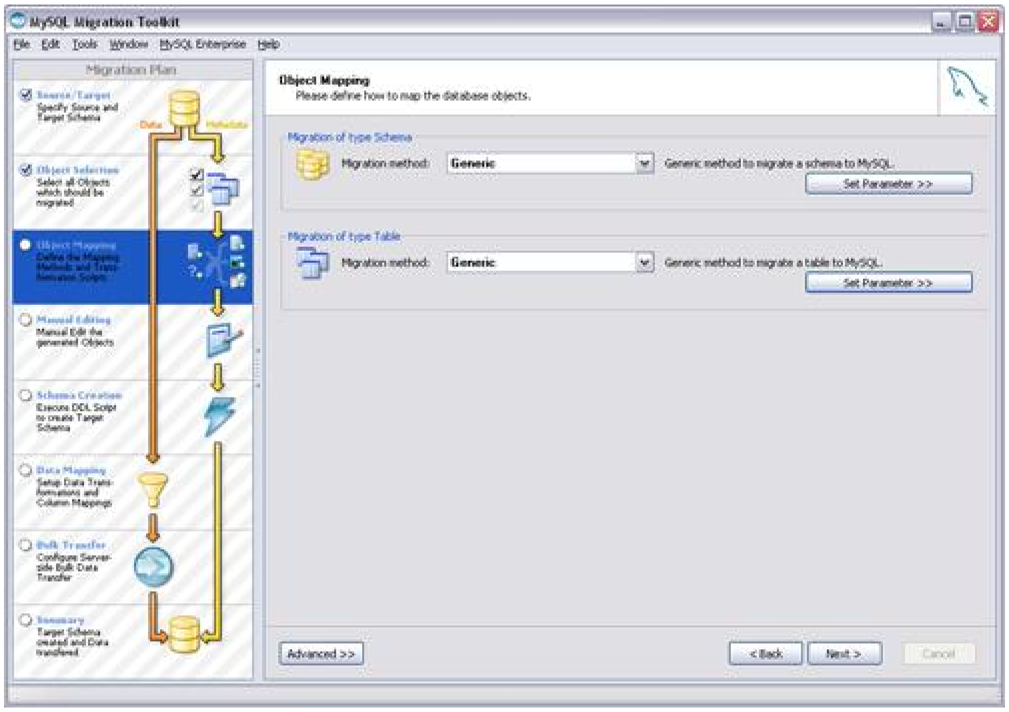
Number to migrate: **6 / 6**

Use the [Detailed Selection] to put objects on the ignore list.

Object to Migrate	Ignored Objects
gim22.broker	
gim22.client	
gim22.client_transaction	
gim22.investment	
gim22.investment_type	
gim22.office_location	

You can optionally select what objects to migrate to MySQL. The default is all data objects in the schema. You can view the objects by pressing the ‘Detailed selection’ button on the wizard screen.

Step 5 – Tweak MySQL Defaults:


MySQL Migration Toolkit
File Edit Tools Window MySQL Enterprise Help
Migration Plan
Source/Target
Specify Source and Target Schema
Data
Metadata
Object Selection
Select all Objects which should be migrated
Object Mapping
Define the Mapping Methods and Transformation Scripts
Manual Editing
Manual Edit the generated Objects
Schema Creation
Execute DDL Script to create Target Schema
Data Mapping
Setup Data Transformations and Column Mappings
Bulk Transfer
Configure Server-side Bulk Data Transfer
Summary
Target Schema created and Data transfered
Set Parameter >>
Advanced >>
< Back
Next >
Cancel

Object Mapping
Please define how to map the database objects.

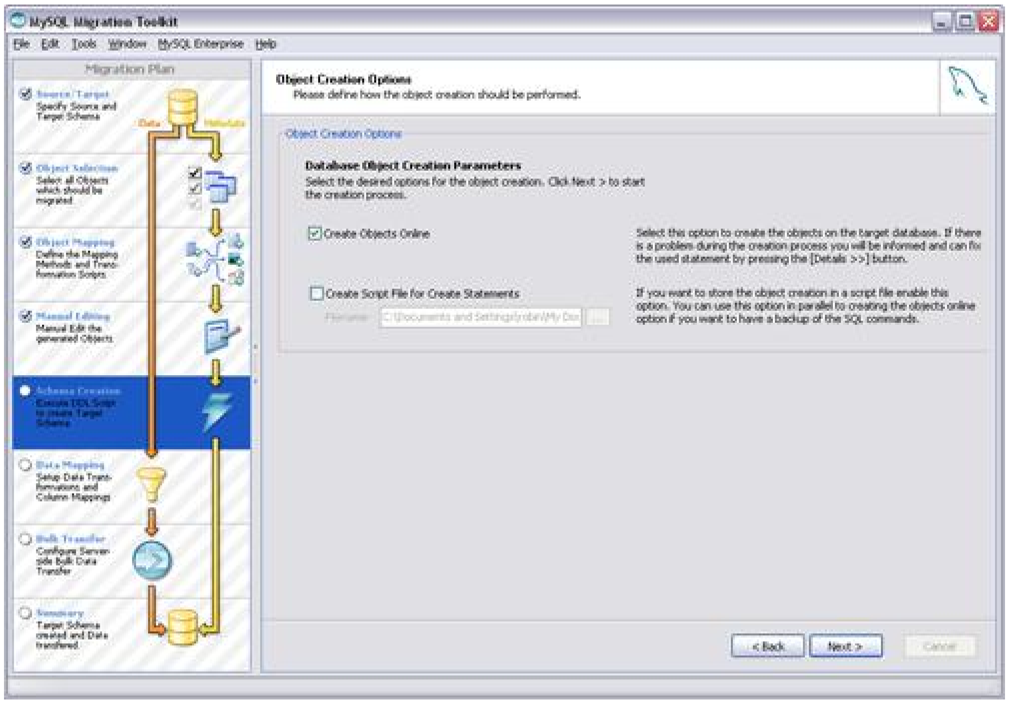
Migration of type Schema
Migration method: Generic — Generic method to migrate a schema to MySQL.

Migration of type Table
Migration method: Generic — Generic method to migrate a table to MySQL.

This step allows you to customize your MySQL settings like target engine, etc. Intelligent defaults are provided (such as InnoDB storage engine for Access databases, using autoincrement columns, etc.) The Migration Toolkit will then proceed and inform you if any mapping problems are found and generate the SQL necessary to create the MySQL target database.

Step 6 – Create MySQL Target Objects:

This step allows you to create the new MySQL database and object schema (with no data) online or save the SQL statements to a file. If you choose to proceed with the default online option, the Migration Toolkit will create the MySQL database and schema objects (without data):


MySQL Migration Toolkit
File Edit Tools Window MySQL Enterprise Help
Migration Plan
Source/Target
Specify Source and Target Schema
Data
Metadata
Object Selection
Select all Objects which should be migrated
Object Mapping
Define the Mapping Methods and Transformation Scripts
Manual Editing
Manual Edit the generated Objects
Schema Creation
Execute DDL Script to create Target Schema
Data Mapping
Setup Data Transformations and Column Mappings
Bulk Transfer
Configure Server-side Bulk Data Transfer
Summary
Target Schema created and Data transferred
< Back
Next >
Cancel

Object Creation Options
Please define how the object creation should be performed.

Object Creation Options

Database Object Creation Parameters
Select the desired options for the object creation. Click Next > to start the creation process.

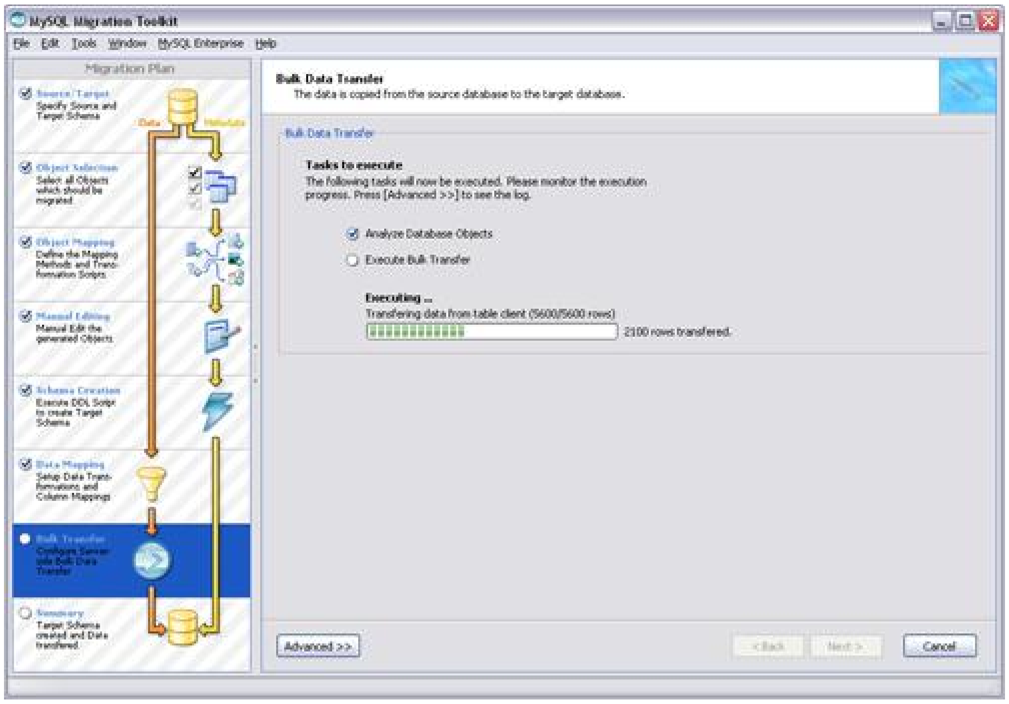
[✓] Create Objects Online — Select this option to create the objects on the target database. If there is a problem during the creation process you will be informed and can fix the used statement by pressing the [Details >>] button.

[ ] Create Script File for Create Statements — If you want to store the object creation in a script file enable this option. You can use this option in parallel to creating the objects online option if you want to have a backup of the SQL commands.

Filename: C:\Documents and Settings\[illegible]\My Doc

Step 7 – Migrate Existing Access Data to MySQL:

This step allows you to migrate the existing Access data to the new MySQL database schema. You can optionally choose to generate INSERT statements into a file for later execution. If you choose to proceed, the migration toolkit will move the data in bulk from Access to MySQL:



Bulk Data Transfer
The data is copied from the source database to the target database.

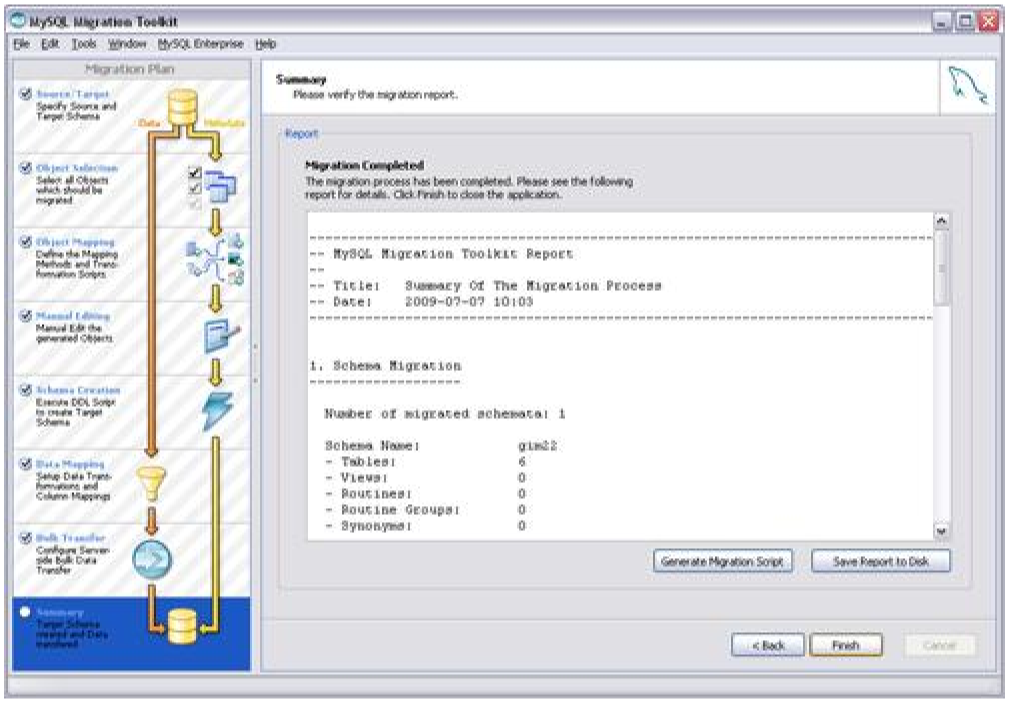
Bulk Data Transfer

Tasks to execute
The following tasks will now be executed. Please monitor the execution progress. Press [Advanced >>] to see the log.

✓ Analyze Database Objects
○ Execute Bulk Transfer

Executing ...
Transfering data from table client (5600/5600 rows)
2100 rows transfered.

Step 8 – Review Migration



Summary
Please verify the migration report.

Report

Migration Completed
The migration process has been completed. Please see the following report for details. Click Finish to close the application.

```
-------------------------------------------------------------------------------
-- MySQL Migration Toolkit Report
--
-- Title:   Summary Of The Migration Process
-- Date:    2009-07-07 10:03
-------------------------------------------------------------------------------


1. Schema Migration
-------------------

  Number of migrated schemata: 1

  Schema Name:            gim22
  - Tables:               6
  - Views:                0
  - Routines:             0
  - Routine Groups:       0
  - Synonyms:             0
```

The final step is to review the migration process and ensure the desired results were obtained.

9.3 Sample MySQL Schema Generated from Microsoft Access

The following MySQL database was generated from the sample Access schema:

```
DROP TABLE IF EXISTS `gim22`.`broker`;
CREATE TABLE `gim22`.`broker` (
 `broker_id` int(10) DEFAULT NULL,
 `office_location_id` int(10) DEFAULT NULL,
 `broker_last_name` varchar(40) DEFAULT NULL,
 `broker_first_name` varchar(20) DEFAULT NULL,
 `broker_middle_initial` varchar(1) DEFAULT NULL,
 `manager_id` int(10) DEFAULT NULL,
 `years_with_firm` varchar(255) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `gim22`.`client`;
CREATE TABLE `gim22`.`client` (
 `client_id` int(10) DEFAULT NULL,
 `client_first_name` varchar(20) DEFAULT NULL,
 `client_last_name` varchar(40) DEFAULT NULL,
 `client_gender` varchar(1) DEFAULT NULL,
 `client_year_of_birth` int(10) DEFAULT NULL,
 `client_marital_status` varchar(20) DEFAULT NULL,
 `client_street_address` varchar(40) DEFAULT NULL,
 `client_postal_code` varchar(10) DEFAULT NULL,
 `client_city` varchar(30) DEFAULT NULL,
 `client_state_province` varchar(40) DEFAULT NULL,
 `client_phone_number` varchar(25) DEFAULT NULL,
 `client_household_income` varchar(255) DEFAULT NULL,
 `client_country` varchar(40) DEFAULT NULL,
 `broker_id` int(10) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `gim22`.`client_transaction`;
CREATE TABLE `gim22`.`client_transaction` (
 `client_transaction_id` int(10) DEFAULT NULL,
 `client_id` int(10) DEFAULT NULL,
 `investment_id` int(10) DEFAULT NULL,
 `action` varchar(10) DEFAULT NULL,
 `price` varchar(255) DEFAULT NULL,
 `number_of_units` int(10) DEFAULT NULL,
 `transaction_status` varchar(10) DEFAULT NULL,
 `transaction_sub_timestamp` datetime DEFAULT NULL,
 `transaction_comp_timestamp` datetime DEFAULT NULL,
 `description` varchar(200) DEFAULT NULL,
 `broker_id` int(10) DEFAULT NULL,
 `broker_commission` varchar(255) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `gim22`.`investment`;
CREATE TABLE `gim22`.`investment` (
 `investment_id` int(10) DEFAULT NULL,
 `investment_type_id` int(10) DEFAULT NULL,
 `investment_vendor` varchar(30) DEFAULT NULL,
 `investment_name` varchar(200) DEFAULT NULL,
 `investment_unit` varchar(20) DEFAULT NULL,
 `investment_duration` varchar(10) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `gim22`.`investment_type`;
CREATE TABLE `gim22`.`investment_type` (
 `investment_type_id` int(10) DEFAULT NULL,
 `investment_type_name` varchar(30) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `gim22`.`office_location`;
CREATE TABLE `gim22`.`office_location` (
 `office_location_id` int(10) DEFAULT NULL,
 `office_name` varchar(20) DEFAULT NULL,
 `office_address` varchar(50) DEFAULT NULL,
 `office_city` varchar(30) DEFAULT NULL,
 `office_state_province` varchar(40) DEFAULT NULL,
 `office_postal_code` varchar(10) DEFAULT NULL,
 `office_country` varchar(40) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```