Strategic Initiatives: Evolving Microsoft Access Applications to Microsoft SQL Server

Learn When and How to Upsize to SQL Server

Prepared by:

Dan Haught
Executive Vice President
FMS Professional Solutions Group
www.fmsinc.com/consulting
Toll Free (866) 367-7801
Local (703) 356-4700

Strategic IT Initiatives: Evolving Microsoft Access Applications to Microsoft SQL Server

Learn When and How to Upsize to SQL Server

Dan Haught, Executive Vice President of FMS, is globally recognized as a software development expert and brings extensive experience building database applications using state-of-the-art technology and program management to complex client engagements. Dan has a UNIX/C programming background, and is an expert in Access, VB, SQL, .NET and web development. He is the primary developer of several FMS products, a frequent speaker at industry conferences, and a contributing editor to several monthly magazines including Advisor Access/VB/SQL and ASP.NET PRO. Dan has co-authored several books on software development, including the Microsoft Jet Database Engine Programmer's Guide (Microsoft Press). He is responsible for new product origination, ensures that all FMS products and solutions employ standards-based best practices, and manages the daily operations of the development teams. Dan joined FMS in 1992.

Executive Summary

This whitepaper explores the issues related to upsizing Microsoft Access applications to take advantage of the performance, security, and reliability of Microsoft SQL Server. Topics discussed include:

- The Value of Access in Your Organization – a brief discussion of how Access provides power and agility to an organization's users
- Making the Decision: When to Upsize – an evaluation of the criteria to decide if an application has outgrown the capabilities of access
- Microsoft Access Data Architectures – a discussion of the type of data architectures that Access supports
- Types of Upsizing Projects – There are several approaches to upsizing. This section shows how to determine which one is best for you.
- Planning an Upsizing Project – careful planning results in a successful project. This section outlines what to plan for.
- FMS Expertise – FMS is a world-leader in both Access and SQL Server. We can deliver your upsizing project on time and on budget.

Recommended Reading

For more information on Access, SQL Server, and Upsizing, we recommend:

- Access in the Enterprise
  http://www.fmsinc.com/???
- When to upsize a Microsoft Access database to Microsoft SQL Server
  http://msdn.microsoft.com/library/default.asp?url=/library/en-us/off2000/html/acconWhenToUpsizeMDBtoSQLServer.asp

The Value of Access in Your Organization

Mid to large size organizations have hundreds to thousands of desktop computers in use on a daily basis. By design, each desktop has standard software that empowers staff to accomplish computing tasks without the intervention of the organization's IT department. This dynamic illustrates the central value tenet of desktop computing: *empowering users to increase productivity and lower costs through decentralizing computing*.

Microsoft Access is used in almost all organizations that rely on Microsoft Windows for desktop computing. Indeed, more desktop database data is stored in Access MDB files than in any other format. As users become more proficient in the operation of these applications, they begin to identify solutions to business tasks that they themselves can implement. The natural evolution of this process is that spreadsheets and databases are created and maintained by end-users to handle day to day tasks.

This dynamic allows both productivity and agility as users are empowered to solve business problems without the intervention of their organization's Information Technology infrastructure. Access fits perfectly into this space by providing a desktop database environment where end-users and power users can quickly develop database applications with tables, queries, forms and reports. Access is ideal for low-cost single user or workgroup database applications.

But with this power comes a price. As more users call on Microsoft Access to handle work issues, issues of data security, reliability, and management become acute. These issues show that some of your Access-based applications need to evolve—they need to move to a more robust environment. And when this evolution is identified, the need for a managed plan for addressing these issues becomes apparent.

The culmination of this evolution is *upsizing*—the process of moving data out of Access and into Microsoft SQL Server, and potentially rewriting the Access application in a more robust environment such as .NET.

This whitepaper shows you where Access fits within an organization and why it is successful there. It also outlines the issues related to the use of Access, information about Access data architectures, identifying when to upsize, and how FMS can help you successfully complete your next upsizing project.

Access and SQL Data Architectures

Microsoft Access is the premier desktop database product available for Microsoft Windows. Since its introduction in 1992, Access has provided a versatile platform for beginners and power users to create single-user and small workgroup database applications.

Microsoft Access has enjoyed great success because it pioneered the concept of stepping users through difficult task with the use of Wizards. This, along with an intuitive query designer, one of the best desktop reporting tools, and the inclusion of macros and a coding environment, all contribute to making Access the best choice for desktop database development.

Since Access has been designed to be easy to use and approachable, it has never been intended as a platform for the most reliable and robust applications. In general, you are going to consider upsizing when these attributes become important for that application. Fortunately, the flexibility of Access allows you to upsize to SQL Server in a variety of ways, from a quick cost-effective, data-moving scenario to full application redesign.

Access provides a rich variety of data architectures that allow it to manage data in a variety of ways. When considering an upsizing project, it is important to understand the variety of ways in which Access can be configured to use its native Jet database format or SQL Server in both single and multi-user environments.

Access and the Jet Engine

The first important fact is that Access has its own database engine—the Microsoft Jet Database Engine. Jet is designed as a fileshare database that supports single and multi-user database applications with databases up to 2 GB in size.

But Access is more than a database engine—it is a development environment that allows users to design queries, create forms and reports, and write macros and Visual Basic code to automate the overall application. In its default configuration, Access uses Jet internally to store its design objects such as forms, reports, macros, and code, and also uses Jet to store all table data.

One of the key benefits of Access when it comes to upsizing is that you can redesign your application to continue to use the forms, reports, macros and code you have already designed in Access, and replace the Jet engine with SQL Server. This allows the best of both worlds: the ease of use of Access, with the reliability and security of SQL Server.

Access and SQL Server: A Quick Comparison

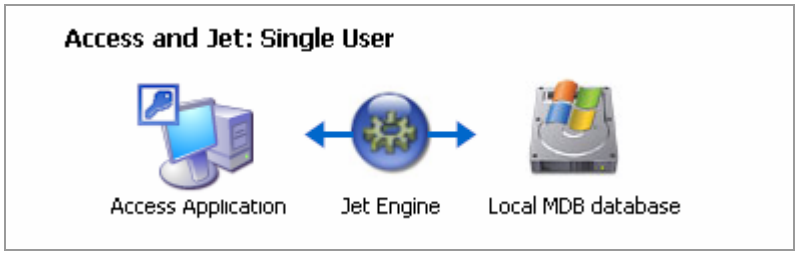
In order to understand some of the decision points in the upsizing decision process, take a look at the following comparison table.

	Access	**SQL Server**
Description	A database development environment that supports tables, queries, forms, reports, and programming logic	A scalable, reliable, and secure client/server database engine
Maximum Database Size	2 gigabytes	1 terabyte

Maximum Concurrent Users	5-15	Unlimited
Security	Basic desktop security	Robust enterprise level security
Performance	Limited by fileshare model	Limited only by hardware
Reliability	Fairly reliable	Very reliable

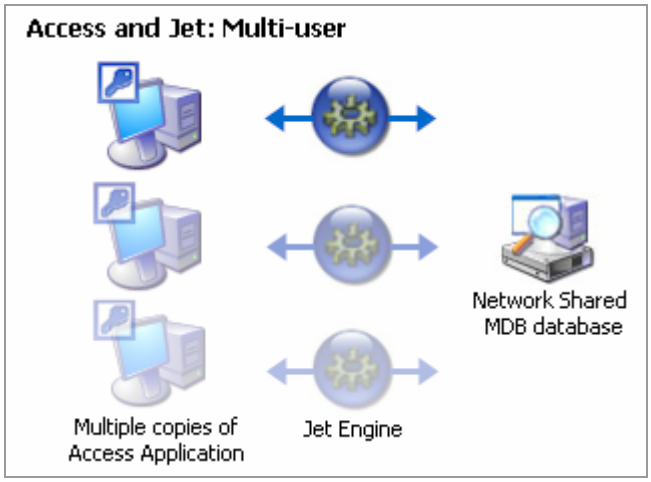
Access and Jet Single User

In its default configuration, Access uses the Microsoft Jet database engine to store both object definitions and table data. Access and Jet are run on the user's computer, and the database is stored on a local hard disk.



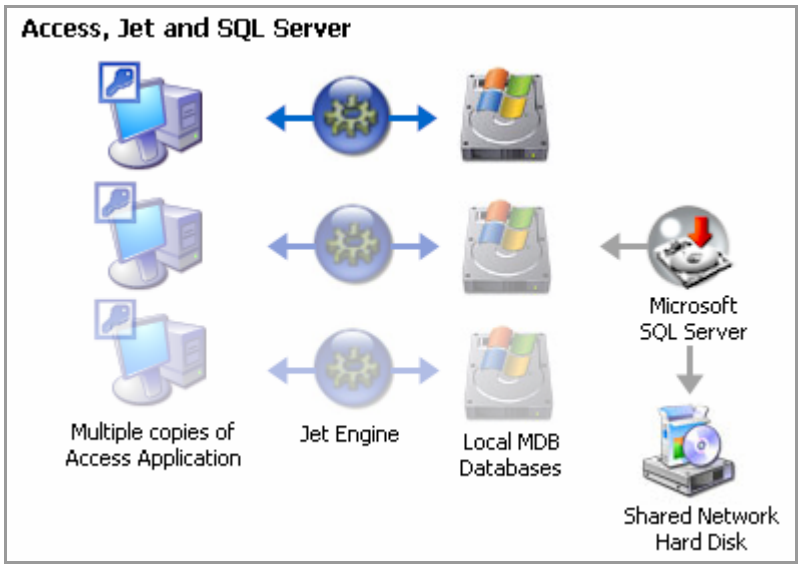
Access and Jet Multi-User

Access and the Jet engine allow multi-user access. In this scenario, each user runs a local copy of Access and Jet, and points to a shared database on a network drive.



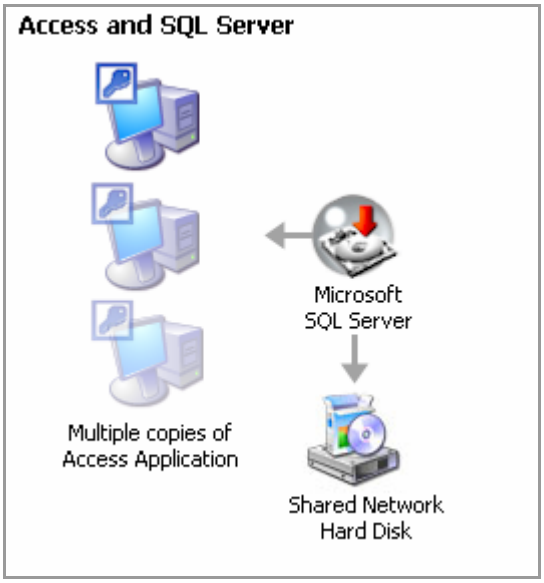
Access, Jet and SQL Server

Access also allows you to point to SQL Server for your data storage. In this scenario, Access still uses Jet to run queries, store object definitions, manage temporary tables, and hold security settings. However, all table data is stored in SQL Server.



Using Access and SQL Server without Jet

In this scenario, the Jet engine is bypassed completely. Access 2000 and later have the ability to directly connect to SQL Server without the need for the Jet engine.



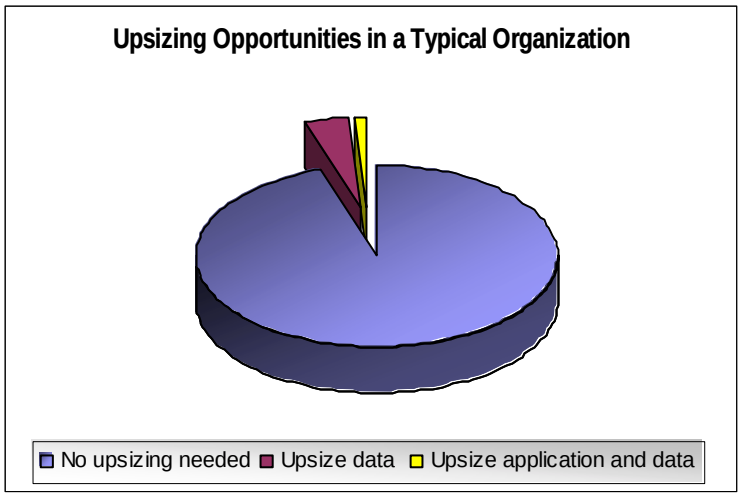
Making the Decision: When to Upsize

Now that you have seen the various architectures and database engine options available, you'll want to explore the decision points and parameters for making the upsizing decision.

The most important part in this process is understanding that not all Access databases need to be upsized. In fact, a majority of Access applications should

not be upsized—the cost and disruption to business is simply not a cost-effective use of your resources. These databases work fine on a day to day basis and do not need attributes such as scalability, security, and 100% reliability. Of all the Access databases in your organization, only a few are candidates for upsizing.

Additionally, from the list of candidates for upsizing, a majority can be upsized using a cost-effective process where only the data is moved to SQL Server. All of the application's functionality in terms of forms and reports is kept in Access. And only the smallest percentage of upsizing projects involve rewriting the Access application in a new environment such as .NET.



The following section examines each of the key areas involved in database planning, and discusses how Access performs in each area.

Scalability

Scalability is defined as the ability of an application to operate in an acceptable manner as the number of users or processes calling the application increases. Access/Jet is not a scalable solution, and scalability is often the primary motivation for upsizing.

Maximum Database Size

Access can support up to 2 gigabytes of data. However, in many cases, this limit is theoretical rather than practical.

- Access uses the file share-based Jet database engine. Unlike client/server solutions such as Microsoft SQL Server, file share databases are not optimized for large datasets. For example, an Access query that needs to provide a total of 10,000 orders needs to pull all 10,000 orders across the network, do the computations locally, and then provide the total. In the client/server model, the same query is handled directly by the server, and only the result is returned to the client application. With larger database sizes, the file share architecture is not capable of handling data loads.
- Jet is not designed for optimum or reliable performance with large database sizes. Many installations will see data corruption because of poor network connectivity or incorrectly designed applications. This corruption occurs most frequently when Access databases begin to exceed 100 MB in size.

Number of Concurrent Users

Microsoft Access can technically allow 255 connections per database. However, this is a theoretical limit and cannot be attained in the real world. In reality, the number of connections/users that an Access database can support is dictated by how well the application was designed and implemented.

Put another way, a professionally designed and well tuned Access application can easily support ten to twenty users with amazingly good performance. On the other hand, a poorly executed Access application can run at a crawl with only two users.

Unfortunately, very few Access databases are actually well designed and implemented with best practices. This is because most Access databases are created by beginners or power users who simply do not have the experience of knowledge to create professional applications. They are built over time and new features and data models are “tacked” on as the need arises. The result is an overall solution that can never reliably support more than a few users.

Architectural Issues

Because Access uses the Jet engine for database management, it cannot scale well by definition. Jet is limited to run on a single CPU, whereas client/server solutions such as Microsoft SQL Server can support multiple CPUs. Additionally, Jet queries always run on the client computer, which eliminates the centralized query/data optimization necessary for a scalable solution.

Reliability and Availability

Reliability is one the key benchmarks to examine when considering upsizing. Indeed, for many mission critical applications, reliability is the single most important consideration. Microsoft Access is not intended as an inherently reliable solution for several reasons.

Database Corruption

When Microsoft Access/Jet databases encounter an error or connection problem, they become corrupt. A corrupt database generally locks out all users of the database, and generally results in data loss and business disruption.

Microsoft Access/Jet databases are prone to corruption for a number of reasons. Since Access/Jet uses a file share model, all users are concurrently holding active connections to data. If any one of those users unexpectedly loses the connection, especially during a data update process, the database can become corrupt. This can happen if the user’s network connection is intermittent, driver versions are not up to date, or if multiple versions of the Jet DLLs are used to read the same database file.

Microsoft Access includes a Compact/Repair utility, but data corruption is usually not fixed by this utility. Third party repair services are available, but this requires sending the affected database off to another location, paying a fee, and waiting for it to be returned. In a best case scenario, 90% of the last changes will be intact, leaving another 10% permanently lost.

Backup and Maintenance Issues

Because Access uses the file share model, the entire database is locked at the file level as soon as it is accessed by the first user. This means that there are no reliable mechanisms for performing backups of the database file unless all users are disconnected.

In a multi-user environment, it is often difficult to coordinate the process of ensuring all users log off of an Access application before making a backup. Typical scenarios involve users leaving their computers on when they leave the office for the day. This leaves the database open, and backup software will not be able to reliably copy the database file. Often, this is only detected after the backup fails, leaving the system administrator to track down the problem and hope it is resolved before the next backup runs.

Additionally, Microsoft Access is not self-tuning. It does not automatically reclaim lost database space, or optimize indexes and queries. This maintenance can be performed by running the repair/compact utility, but this also requires that all users be logged out of the database.

Since many Access databases are stored locally on user machines, they are often not included in any type of backup or maintenance plan.

Different Versions of Access and Jet

Microsoft Access has strong dependencies on specific versions of the Jet Engine, and other related data access components. For example, if you create a database with Access 97, it can be opened with Access 2000 which will then make it unavailable to Access 97 users. Additionally, if new versions of standard Microsoft data access components such as DAO and ADO are installed, they may make existing Access applications fail. This is especially true in multi-user situations.

This lack of backward compatibility between driver versions often precludes organizations to upgrading to newer versions of Office, since a new version of Access can likely cause existing Access applications to stop working.

Security

Microsoft Access offers three different security mechanisms.

1. Database Passwords: You can assign a password to a database. Only users who know the password can open the MDB file.
2. Jet Workgroup Security: Users, groups, and object permissions can be defined and shared across multiple workgroups.
3. File Encryption: contents of the database can be encrypted at the file level.

Unfortunately, these mechanisms are neither robust nor reliable. Database passwords use a very simple encryption mechanism. In fact, removing an Access database password is simple matter given that free and commercial password “removers” are easily available on the web. While Access users may not be concerned about such lapses, IT managers certainly should be.

And while Jet Workgroup Security is more robust, it still leaves the contents of the entire MDB database open from the file system. Since all table data and code is stored in plain view, it is a trivial matter to open an MDB file in a string-compatible editor and view code, passwords, and table data.

Finally, because Access requires full read permissions for all users to the actual database file, anyone who can see a shared network drive can walk off with the database on a disk or CDR, or email it outside of your organization.

Upsizing Scenarios

When contemplating an Access upsizing project, it is important to understand that there are a variety of upsizing methodologies. These range from simple data

moving, to complete re-architecture and redesign. In order to choose the correct path for your upsizing project, you should be familiar with the three types of data architecture that Access supports.

This section outlines the upsizing scenarios and provides details about the benefits and drawbacks of each approach. The following scenarios are examined:

Scenario	Description	% Of Existing Databases
Already Right-sized	Many Access databases do not need to be upsized.	90%
Upsize Data Only	Leave application design and logic in Access, and move data to SQL Server.	7%
Upsize Application and Data using Access	Rewrite the application in Access using an Access Data Project, remove the need for Jet entirely, and move data to SQL Server.	2%
Upsizing Application and Data using .NET technologies	Rewrite the application using Visual Studio .NET for Windows and/or web access, and move data to SQL Server	1%

Scenario 1: Already Right-sized

If you were to inventory the use of Access in your organization, you would likely find hundreds to thousands of MDB databases scattered across computers and network drives. These databases run the gamut from simple lists built by staff members, to workgroup-level multi-user applications.

With database counts that run into the hundreds, and given the potential cost and disruption to business that upsizing may involve, it is obvious that only a small subset of the total should be candidates for upsizing.

The first rule of upsizing is that the large majority of your databases should not be upsized: the cost is prohibitive. And even if you had the resources to upsize a majority of your Access databases, there would be no real gain. Simple lists or reports used by a single person typically do not fall into the realm of mission critical applications. Indeed, these types of applications are what Access is designed for, and are well within its capabilities.

Finally, many of the databases you would find in a typical inventory process may not have been used for 6 months to a year. These obsolete databases are no longer important to your organization, and other than for archival purposes, are not candidates for upsizing.

The key advantage to this scenario is that you don't have to do anything; no cost no business disruption. The disadvantage is that Access/Jet based solutions cannot scale and do not enjoy the reliability and security of SQL Server. But that is typically not an issue for the majority of your Access databases.

Advantages	Disadvantages
▪ Cost: no additional software is needed, since Jet is included with Access ▪ Ease of use: no SQL Server knowledge required	▪ Limited scalability ▪ Limited security ▪ Limited number of users

- Lowest development costs

- Limited reliability
- Jet databases prone to failure if new versions of Office, Access, Jet, or data access components are installed

Scenario 2: Upsize Data Only

Because Access has the ability to link to SQL Server for table data, migrating only the data is one of the best balances between cost and advantages. In this scenario, you move all table data to SQL Server and leave all forms, reports, queries, macros and logic in the existing Access database.

The key benefit of this scenario is that it is the quickest and most cost effective because it has the least impact on existing application logic. Very few, if any parts of the existing application need to be changed. With a small investment, you can gain the reliability and maintenance benefits of SQL Server and keep your Access investment in place.

The largest disadvantage of this approach is that all access to SQL Server happens through the Jet engine. This is because the Jet engine must translate every query and data access operation from its native format to SQL compliant commands. This translation adds overhead in performance, and uses additional SQL connections which can lead to additional SQL license purchases.

Scenario 2 is best for Access applications with a small number of users and small database sizes.

Advantages	Disadvantages
▪ Lowest cost upsizing project ▪ Data is located in SQL Server offering security, scalability, and reliability	▪ Using Jet as a layer on top of SQL Server may provide slow performance ▪ Multiple copies of local MDB require synchronization ▪ Local MDB databases offer limited security, scalability, and reliability ▪ Since Jet is still used, local databases susceptible to failure if new versions of Access, Jet, or data access components are installed

Scenario 3: Upsize Application and Data using Access

If you are planning to upsize an Access application and performance and scalability are as important as the other benefits (reliability, security, maintenance, etc), you may want to consider not only moving to SQL Server for your data, but rewriting your Access application to remove the Jet engine from the equation.

You can do this by using the Access Data Project format that is available in Access 2000 and later. Access Data Projects allow the same use for forms, reports, macros, code, and intuitive design tools available for standard Access databases. However, they directly connect to SQL Server for data access.

There are two situations where you would consider this scenario:

1. An existing Access application is using Jet only and needs to be upsized to SQL Server in a way that offers optimum performance and scalability.
2. An existing Access application is connected to SQL Server through Jet and needs better performance and fewer SQL Server connections.

The major benefit of this scenario is that it results in an Access application that provides the best performance and scalability, on top of all the other positive attributes of SQL Server.

The major drawback to this approach is that it requires more development effort because Access objects such as forms, reports, queries, and code need to be redesigned to work directly with SQL Server.

Advantages	Disadvantages
▪ Enjoy all the benefits of SQL Server, including performance and scalability. ▪ Data is located in SQL Server offering security and reliability	▪ Cost of redesigning application logic and retesting. Of course, this is offset by the improvements gained.

Scenario 4: Upsizing Application and Data using .NET technologies

When an existing Access application outgrows Access, it can do so in a major way. Access is no longer able to keep up with your organization's needs for data capacity and performance. Or you may need to migrate all or part of an application to the web. There are a small percentage of Access upsizing projects that can only be successfully completed by migrating out of Access altogether.

In this scenario, the Access application is used as the beginning roadmap for a completely new design. Additional technologies such as Visual Basic, Active Server Pages, and/or Visual Studio .NET are employed to rewrite the application from the ground up. As a natural matter of course, the data moves to SQL Server. Additionally, you can consider migrating other data sources, such as Oracle and DB2 into SQL Server for a centrally managed solution.

The key advantage of this approach is flexibility. You can create an application that can target Windows desktops and the Web with minimum changes. A more professional development environment such as Visual Studio .NET offers advantages such as team based management, source code control, and professional tools and components available from a rich array of third party vendors. With this scenario, you end up with a reliable, scalable, and manageable application that can move from the business unit to the enterprise level.

The key disadvantage of this approach is cost. Since you are ultimately discarding the Access application and its database, you are creating a new application with a new design, development and implementation project. Fortunately, only a small majority of Access applications require this level of effort.

Advantages	Disadvantages
▪ Flexibility: application can target Windows, Web, and more ▪ Scalability and reliability: using .NET development technologies with SQL Server offer the best mid-business and enterprise level	▪ Highest cost ▪ Retraining of staff ▪ New application testing ▪ Additional developer expertise

return on investment

- Ease of Management: Versions of Access no longer play any role in the application's ability (or inability) to be used across the enterprise.

Inventorying Access Databases in Your Organization

One of the biggest challenges your organization may face is identifying how many Access databases you have, and which ones should be upsized. The problem is where to start—how do you efficiently inventory your Access databases just to get an initial handle on the problem? Even with conservative estimates, an organization with 500 deployed desktops may potentially have 10,000 Access databases.

There are several strategies for solving this problem. The simplest route is to communicate with desktop users, usually through an email message, and ask for basic feedback on each user's database inventory.

- How many Access databases do you currently use?
- How many tables are in these databases?
- Do you share this database with other users?
- Do you link to, or use import/export on, corporate data?
- Are your databases being backed up?

A well defined (and brief) set of questions will help you identify which databases may be at risk:

For larger organizations, a better managed approach is to implement a system that can automatically inventory and report on Access databases. Using a desktop agent that regularly inventories local and network hard drives, in combination with a centralized server reporting application, can provide tangible benefits for an organization's need to manage desktop data, and schedule upsizing projects.

FMS provides products and services that allow such automated management functions. For more information, visit www.fmsinc.com/????.

Planning an Upsizing Project

To avoid unnecessary costs, ensure application availability, and minimize risks, it is important to carefully plan your Access upsizing project. The amount of planning is directly related to the type of upsizing project you envision. For example, a simple migration of data to SQL Server requires less planning that a complete rewrite of the application and data migration. This section provides guidelines and best practices for planning your upsizing project.

Phase 1: Design and Planning

Choose Your Upsizing Scenario

Your level of planning and overall effort is directly related to which upsizing scenario you choose. For example, upsizing data to SQL server while leaving the Access front-end in place requires less effort, but yields fewer benefits. Once you have chosen your plan, be sure to clearly state goals, timeline, and budget.

Identify a SQL Server Installation

Once you choose to upsize an Access application, you will need to either identify an existing SQL Server installation to use, or plan to create one. SQL Server comes in a variety of editions, from the freely available Microsoft Database Engine (MSDE) to SQL Server Enterprise edition. In general, all editions of SQL Server, including MSDE, are capable of handling small workgroup applications involving 1-20 users. If you are upsizing both the application and the database, and your needs call for the greatest scalability, functionality, and reliability, consider using the Enterprise Edition.

Administration

Before your upsizing project is deployed, you should have an administrative plan in place for your new SQL Server data. Planning for this before the rollout is key. Installing SQL Server and creating objects is only part of the equation. You should define backups schedules, fault tolerance parameters (as needed), and administrative staff who are responsible for the database component.

Development Plan

Create a development plan that covers each aspect of the Access application that must be changed. If you are only planning to upsize the data to SQL Server, there are still parts of the Access front-end that may need to change. For example, the Jet database engine uses different data types, and a different SQL grammar than does SQL Server. Plan to identify any areas of incompatibility and change Access objects as needed. If your scenario calls for a complete rewrite of the Access application in a different environment, such as .NET, you need to approach the project as full lifecycle software development effort and plan accordingly. Finally, be sure to identify risk areas such as data destabilization or loss that could potentially occur, and have a proactive plan in place to address them.

Evaluate the Microsoft Upsizing Wizard

Microsoft provides an upsizing wizard that allows semi-automatic upsizing of Access to SQL Server. Unfortunately, this wizard is quite limited in its ability to create usable SQL Server-based applications. When you are contemplating and upsizing project, you can certainly plan to use the Microsoft upsizing wizard as a starting point. However, for all but the most simple (i.e. Scenario 2) upsizing projects, the upsizing wizard will only accomplish about 40% of the work. This section describes the limitations you can encounter with the Microsoft Upsizing Wizard.

Issue	Description
Non-standard table/field names	Jet and SQL use different naming standards. The upsizing wizard can find some, but not all. And those that it does find and rename will not work in any existing code.
Differences in SQL	Access/Jet uses its own dialect of SQL that is different from the ANSI SQL supported by SQL Server. Many Jet-based queries cannot run on SQL Server without rewriting.
Data type conversion issues	Access/Jet has its own standards for data types that are different in some cases from SQL Server. The upsizing wizard can make some choices for you in terms of converting data types, but changes require developer review.
Architectural Issues	The Microsoft upsizing wizards cannot rewrite your application to work correctly with the SQL Server client/server model. Almost all Access/Jet applications

	are designed to work with the fileshare model of Jet. These designs do not lend themselves well to the client/server model and can result in poor performance.
Code Not Converted	The upsizing wizard does not convert any of the VBA code in your application. This can result in serious errors as parts of your application point to SQL Server while your code still points to an Access/Jet database.
Items not Upsized	The Microsoft upsizing wizard does not convert any of the following objects: hidden objects, security settings, Format and InputMask properties, Table/Field caption properties, table lookup fields, cross-tab queries, action queries that take parameters, many query properties, macros, and module code

In general, consider using the Microsoft Upsizing Wizard as a starting point, or for proof of concept phases. However, it cannot be relied on to actually upsize an application in the correct way.

Phase 2: Implementation

Configure SQL Server

Use the data diagram that is part of your development plan to implement the first version of SQL Server objects, such as tables, views, and stored procedures. Implement users, groups, and roles as needed. It is important to have these objects in place before development starts—developers can't work against a SQL Server backend that isn't there. Don't worry about performance optimization yet, that happens later.

Development

Based on your development plan, staff your development team and provide the resources necessary. Make the existing Access application available to the team for use a benchmark or prototype resource. Keep an eye on the milestones and risk areas defined in your planning process.

Testing

Before the first test deployment of the new application, basic developer-based testing should occur. Use the existing Access application as a model to reduce the amount of time needed for the initial testing effort. Compare each functional area in the original Access application against the new code base. If you are completely rewriting the Access front end application as well as moving the data, you should plan to involve dedicated quality assurance/testing staff to find critical errors.

Documentation

Most Access applications are created by end users, and as such, lack documentation. Since you are investing in the process of upsizing, now is a good time to spend some time documenting the new application. At a minimum, create a configuration and troubleshooting document that outlines where the application's component parts reside, desktop and network settings, and basic troubleshooting techniques based on the results of your testing plan. If you have the resources, you may want to consider more complete documentation in the form of data diagrams, flowcharts, code listings, etc.

Training

When you take an existing in-production application and change or rewrite it, you must plan to ensure that the application's users are on board. Depending on the scope of the changes involved in the upsizing project, training for the application's users may involve a few hours of walkthroughs, to a full formal training regimen with the associated training guides and documentation. Good training is crucial if you want to get the buy-in of the application's users.

Rollout

Your first rollout of the application is typically deployed to a subset of the entire user population. Select a small group of users and employ them as the beta-testers. The obvious goal is to verify the planning and development work—does the new application work correctly? Beyond that, user feedback may help identify any last minute issues not addressed in the planning and implementation process. Users can also provide invaluable information regarding usability.

Once you have been through initial testing, and made any necessary changes or fixes, roll the application out to the entire user base. Depending on the number of users in the application, and the importance and currency of the data, you may want to consider running the old Access-based system in tandem with the new system for a period of time. This provides an extra degree of security should the new application experience problems.

Phase 3: Stabilization

Once the new application is in production use for all users, the project enters the stabilization period. Defects are identified by users and fixes are planned. Users will also see opportunities for new functionality (as is the case with any application) and these should be duly noted by management. Ongoing support to users is important since an upsizing project often results in application attributes that are no longer under the control of the end user (i.e. SQL Server).

During this period, you should also monitor performance, not only in terms of what users may be reporting as slow, but active monitoring of SQL Server using tools such as the query analyzer and performance counters.

FMS Expertise

When Microsoft decided to engage a certified partner to lend technical expertise and support for its Access to SQL Server 2000 Upsizing campaign, FMS was at the top of the list. We have focused on the Microsoft technology platform throughout our history, and remain one of the most trusted advisors to several Microsoft development teams today. Our reputation is one of consistent, thorough, and significant involvement in all aspects of the software development community, and our products have won some 40 industry awards thus far.

Since the first release of MS Access in 1992, FMS has been providing award winning Access development software tools to the programming community. Now known around the world as the preeminent supplier of third party Access software, FMS has continually been on the forefront of Access innovation since the beginning. In fact, Luke Chung, our president and founder, is listed on Microsoft's website as a 'Top Ten Access Hero' for his support and early adoption of the best selling database program (http://www.microsoft.com/office/access/10years/chung.asp).

Equally important, we have been developing reliable high performance database systems using Microsoft SQL Server Since version 4.2 (in 1993). Our team has extensive experience in data normalization techniques, best practices for database design, and a deep internal knowledge of how SQL Server works. We have implemented everything from small-scale workgroup solutions, to highly scalable ecommerce business sites using SQL Server. Additionally, our Developer Tools Group has two developer products specifically for SQL Server, award-winning Total SQL Analyzer Pro and Total SQL Statistics. You can also read about FMS on the Microsoft SQL Server Site.

Conclusion

Database evolution should be expected and it is a normal course in any company's usual business development. The importance, size, and / or user accessibility (growth) of a particular application often can exceed its original concept or development platform. While most Access applications can spend their entire useful life functioning perfectly well, some should be migrated to more secure and robust platforms. Knowing which Access databases are candidates for upsizing and exactly how to perform the migration can prove challenging to even the most experienced network manager.

By keeping the Access application and data within the Microsoft family of products (e.g., Access to SQL Server 2000), and engaging an experienced technology partner like FMS, the process can be quite manageable, and cost effective.