



AZTEC
COMPUTING

Guide to Upsizing from Access to SQL Server

An introduction to the issues involved in
upsizing an application from Microsoft
Access to SQL Server

January 2003

© Aztec Computing

Microsoft
CERTIFIED
Partner

1 Why Should I Consider Upsizing to SQL Server?

Microsoft Access is a great tool for quickly building easy to use database applications. However if your application has evolved to manage more and more data, or it is used by increasing numbers of users, then you may well have noticed deterioration in performance. You may also be experiencing periodic corruptions, leading to unwanted down time and increased maintenance costs that ultimately have a significant adverse effect on your business. You really want your application to manage your data reliably and efficiently, regardless of the volume of data or numbers of users, that is you want your application to be scalable.

So why isn't your Access application scalable? Well the problem lies with JET, Access' filesERVER database engine that performs Access' data management responsibilities. JET is a filesERVER database engine, and all filesERVER database engines suffer from the same problem. Each workstation using your application loads its own copy of JET, and all processing is performed on the workstation. When a user makes a request for data, the user's copy of JET has to fetch the data from the filesERVER, resulting in large amounts of data being sent across the network. As more and more users come on board the network quickly becomes your application's bottleneck. Worse still, data corruptions become more likely as each workstation's copy of JET has to cooperate to ensure data is locked correctly. A problem on any one workstation could result in corrupt data for everyone.

SQL Server is not a filesERVER database, but a dedicated database server, designed to manage large data volumes and to support a large number of concurrent users. Porting your application to SQL Server will provide improved performance and reliability. Your application's data management (and possibly some of its business logic) can be migrated to SQL Server, leaving the user interface largely unchanged in Access. The key to a successful migration is to combine a pragmatic approach (leaving as much unchanged in Access as possible) with an understanding of client server architecture (so you can identify where to focus your efforts to make the biggest improvements in performance). The next section provides more detailed suggestions for developing a successful strategy.

2 Developing a Migration Strategy

Moving your tables to SQL Server won't necessarily deliver the performance improvements you're seeking. In fact sometimes your application won't work at all without a little tweaking. Your overall strategy should be to:

- export the table structures to SQL Server
- get your application working
- modify your application to improve performance
- migrate the live data

2.1 Export the Table Structures

The first step is to move your tables from Access to SQL Server. Remember that whilst you are developing the new Access/SQL version of your application, users are still entering data into the existing version. Although it's useful to migrate the data at this stage, you will need to re-migrate the latest data set once you have tested your new version and are ready to go live.

2.2 Get Your Application Working

The second step is to get your application working, making as few changes as possible. Here you address the problems with your application caused by the small differences between the ways JET and SQL Server work. At the very least you will need to replace your existing Access tables (or links) with links to the new SQL Server versions of your tables.

2.3 Improve Performance

The third step is to modify your application in order for it to work efficiently with SQL Server, homing in on performance bottlenecks. Your core strategy should be to keep network traffic low, and to take advantage of the server's power wherever you can. Here are a few issues to consider:

Move Complex Queries to SQL Server

Even though you've moved your tables to SQL Server, it's likely your forms and reports are based upon queries. Your queries will still be processed by JET, which performs some processing itself and passes some on to SQL Server. Performance should be acceptable for the majority of your queries, but for the more complex queries you should consider getting SQL Server to do more of the work by making use of pass through queries or calling stored procedures (see below). SQL Server will do all the hard work on the server, returning only the final result set to Access.

Use Small Datasets

When your application makes requests from SQL Server for data, SQL Server performs a query and returns the results over the network to your application. Keeping network traffic light helps performance and scalability. Identify those forms (or other parts of your application) that retrieve large result sets (possibly whole tables) and try to modify them to encourage the user to retrieve only the data they need.

Batch updating

If you bind forms directly to SQL Server data (using linked tables or queries) remember that as users move from record to record their changes will be saved immediately, with a connection open all the time. Increasingly it is considered best practice to improve scalability by closing connections and reducing SQL Server resource usage as much as possible. Consider fetching data for manipulation into a temporary location and then breaking the links to SQL Server. Users can then edit the data “offline”, and only when they have finished making their changes are the updates sent back to SQL Server. On key data input forms, you could fetch data into unbound forms, or into temporary local tables to which your forms are bound. Add processing logic to the forms to update SQL Server with the changes once the user has finished editing the data.

Move Complex Business Logic to Stored Procedures

Stored procedures consist of SQL and programming statements that are stored in the SQL Server database. If your application includes complex processing that manipulates your data with little input from the user, moving the logic to a stored procedure will dramatically improve performance.

2.4 Migrate Your Production Data

Once you've made all your modifications to the application (and tested it thoroughly) you'll need to migrate the data from the old version. First make sure nobody can make any further changes to the old data. Take a copy of the file for future reference. Migrate the data overnight or at a weekend so that users don't lose productive time. There are a number of ways you can migrate data detailed below, but some form of script is best since it can be rerun if necessary.

3 Technical Detail

This section provides further information about the issues to consider when upsizing. It is not prescriptive, but provides pointers for further investigation.

3.1 Data Structures

Data Types

SQL Server's data types are similar to, but not identical to Access'. Make sure you choose the correct data type. The table below shows how to map the more common types. Note that for linked tables Access displays the equivalent Access type, not the underlying SQL type.

Access	SQL Server	Comment
Text (n)	varchar (n)	Use nvarchar for Unicode characters.
Byte	TinyInt	
Integer	Smallint	
Long Integer	Int	
Single	Real	
Double	Float	
Currency	Money	
Yes/No	Bit	Access uses -1 for true, SQL Server uses 1. Access treats null as false, but SQL Server doesn't, so consider assigning false as default value.
Date/Time	Datetime	
Memo	Text	

Primary Keys

SQL Server implements primary keys in a similar way to Access, although they are known in SQL Server as primary key constraints. A unique index is created to support the primary key constraint. You will usually, but not always, want to specify the index to be a clustered index (see indexes below). Like Access, any column participating in the primary key cannot contain a null value.

Unique Constraint

If in addition to the primary key you have a field or combination of fields that you want to restrict to unique values, add a Unique Constraint (equivalent to Access' unique index). Note that like Access, you can include columns that contain null values. However whereas Access allows multiple null values and will still enforce the uniqueness of the non-null values, SQL Server only allows one record with a null value in the column. SQL Server 2000 can mimic Access' behaviour using an indexed view on the non-null records. However SQL Server 7 does not support indexed views, so you will have to resort to the use of triggers if you want to enforce uniqueness of non-null values.

Foreign Keys

These are similar to Access' foreign keys, but known in SQL Server as Foreign Key Constraints. The constraint must reference a Primary Key or Unique Key constraint of the referenced table. There are two important considerations when migrating from Access.

Firstly, although a foreign key column may contain null values, you will not be able to implement cascading updates or deletes if the column contains null values. Secondly, unlike Access, SQL Server does not create an index to support a foreign key constraint. You will usually want to create your own indexes on foreign keys to improve performance.

Relationships

SQL Server 7 and SQL Server 2000 both support referential integrity. However only SQL Server 2000 supports cascading updates and deletes (although it doesn't allow null values in Foreign Key fields). If you need to implement cascading updates and deletes in SQL Server 7 you will need to use triggers.

Mandatory Fields

Whereas in Access you specify whether a field is "required", in SQL Server you specify whether or not it is allowed to accept Null values. Be careful because some of SQL Server's front-end tools specify Null by default, whereas others specify Not Null.

Validation Rules

Access allows validation rules to be defined at the field and table level. The majority of these rules can be implemented in SQL Server using Check Constraints. If for some reason your rule cannot be implemented as a constraint, you could use a Trigger.

Defaults

Default values are implemented much the same way in Access and SQL Server.

Indexing

SQL Server supports two types of indexes, clustered and non-clustered (actually it also supports a third type known as free text indexes used for searching within Text fields). A clustered index forces the data in a table to be physically ordered according to the index's key. Consequently a table can have only one clustered index. A clustered index is usually more efficient than a non-clustered index, so you'll want to create the clustered index on the field (or field combination) likely to be accessed most often. This may not always be the table's primary key.

3.2 Migrating Data

To migrate your data you should consider

- using the Upsizing Wizard
- exporting the tables from Access
- importing the tables from Access
- using Data Transformation Services (DTS)
- linking your Access database to SQL Server

Whilst the Upsizing Wizard will probably do a good job of migrating table structures and data, it cannot be used to rerun the data transfer once your new application is ready to enter production. The same is true if you simply decide to export/import the data. DTS allows you to save the transfer as a package, which you can rerun at any time, with

plenty of scope to customise the data transformation. You can also link your Access database to SQL Server and execute SQL Server stored procedures against your Access tables to migrate the data.

3.3 Connecting to SQL Server

Your application must know how to connect to SQL Server. Most commonly you will create an ODBC DSN that defines how your application will connect to SQL Server.

Attached Tables

Your new attached SQL tables contain information on how Access is to connect to SQL Server. Common practice is to create an ODBC data source (DSN), and when you attach your tables, this information will be embedded with the table definition. When attaching tables, don't select the option that allows the SQL Server password to be stored with the linked table definition or you will compromise your security.

When you attach a SQL Server table, Access names the linked table based upon the owner of the table and the table name. This is because in SQL Server it is possible to have tables with the same name owned by multiple users. When you attach a table, you will most probably see Access prefix the table name with "dbo_". If you are sure that your database will only contain one table with this name, then rename the attachment back to its original Access name by removing the prefix. Any queries, forms, reports or code will then work automatically with the new SQL table.

Integrated security

SQL Server allows the developer to use either SQL Server security or Integrated Security. With SQL Server security the developer defines users and groups within SQL Server, and assigns permissions to them. Integrated Security on the other hand allows the developer to assign permissions directly to NT logins and groups. When connecting to SQL Server using integrated security, you don't need to supply a username or password. You can even use a combination of both methods. Where NT groups are well defined, use integrated security as less maintenance is required.

Preconnecting

If you are using SQL Server's built in security, then ODBC will prompt you to complete any missing information when you first access any SQL Server resource. "Preconnecting" allows you to provide this information up front when your application starts. Use the OpenDatabase function to preconnect passing it an ODBC connection string.

Moving from development to production

If your application uses a DSN then each workstation that uses the application must have a copy of the DSN. Alternatively you can create a File DSN on a shared drive, or make your connections "DSN-less". To use DSN-less connections you must specify the ODBC driver and server name in your connection strings.

Note that if you use different servers for development and production, then even if you use the same DSN you will need to ensure that you refresh the links to your linked tables and pass through queries, as Access seems to cache the name of the last server used.

ADO Connections

ADO/OLEDB is a newer technology for data access, and completely different from ODBC. ADO is discussed in more detail below, but if you do use ADO you will have to make

separate ADO connections as it cannot use any ODBC connections you have already established.

3.4 Viewing & Editing Data

Use attached tables to display data

As long as Access can uniquely identify records in a table, then you can edit data displayed in a datasheet or bound to a form. If Access can't find a unique index, then it will ask you to specify a unique identifier when you attach the table.

Note that changes to a record are saved back to SQL Server as soon as you finish editing a record. ODBC will use optimistic locking, that is assume no one else has changed the record since you read it. Before you save your updates, SQL Server checks to make sure the record hasn't changed by comparing the content of each field. You can speed up this process by adding a timestamp field to the table. SQL Server automatically updates a timestamp field every time a record is changed, so it only needs to check this one field to see if anyone has updated your record.

Autonumbers

Access assigns a new value to an autonumber field as soon as a user starts to create a new record. SQL Server on the other hand doesn't allocate the new number for an identity field until the record is saved. Make sure your application doesn't contain logic that assumes the value is assigned as soon as the user starts to create a new record.

If you open any recordsets based upon tables containing identity fields, you must add the `dbSeeChanges` option to the `OpenRecordset` method.

Note that SQL Server's identity is actually an attribute of a field, rather than a field type. You can even tell SQL Server to stop providing new numbers temporarily (perhaps when you are transferring data) using the `SET IDENTITY_INSERT` command.

Error handling

When an ODBC operation fails, your application will be provided with a series of error messages, some providing more detail than others. The errors will be placed in an errors collection of the `dbengine` object which you should loop through to examine the error in detail. If an ADO operation fails, error messages will be placed in the errors collection of the current ADO Connection object.

3.5 Improve Performance

Your core strategy should be to keep network traffic low, and to take advantage of the database server's power where you can.

Indexing

You can help SQL Server to retrieve information more quickly by providing suitable indexes. As mentioned in the section on table structures, try and make a good choice for your table's clustered index. Remember also that SQL Server does not automatically index foreign keys, so you should index these yourself. Whilst indexing is a key factor in retrieving information, creating excessive indexes can have an adverse impact on the performance of updates.

Small Recordsets

The most important factor for achieving good performance is to use small recordsets. Not only does this keep network traffic low, but also ODBC only requires one SQL Server connection for recordsets of less than 100 rows. Larger recordsets require two connections, one for the keyset and one for the data. Make sure your forms aren't bound to large recordsets. If they are, try and amend them to allow the user to specify criteria that will restrict the rows.

To reduce interaction with SQL Server still further on data capture forms, you can bind the forms to temporary local (JET) tables. To display data, fetch the requested data from SQL and insert it into your temporary tables. Let the user edit the data in the temporary tables "offline", and only write the changes back to SQL Server when the editing session is complete. Batching updates like this improves the speed of update and improves multi-user concurrency and is the paradigm behind new, scalable technologies (e.g. disconnected recordsets in ADO.Net). However this approach can be a bit more involved, as you have to duplicate your table structures locally, and ensure you perform validation before attempting your batch updates. Also the longer your data is offline, the more likely you are to encounter an update conflict caused by someone changing the data since you read it.

Download static data

To reduce network traffic and improve performance of combo boxes and list boxes, consider downloading relatively static data, such as lookup tables, into local copies of the tables at application startup.

Queries

Once you've attached your SQL tables to your application and ensured they have the same name as the original Access table, your queries should continue to work. However, you should identify the poorer performing queries and modify them to try and get SQL Server to perform more of the work (SQL Server's Query Analyser and Profiler tools can help you with this task).

When Access runs a query based upon an attached SQL table, it calls upon JET to translate the query into syntax SQL Server can understand. Sometimes JET fires off queries to SQL Server, but still has to perform some local processing. Pass Through queries allow you to construct queries that bypass JET and are executed by SQL Server directly. Pass through queries are written using native SQL Server syntax. Be careful as there are some subtle differences between SQL Server and JET syntax (for example SQL Server uses "%" as the wildcard character, rather than "*"). Since there isn't a way of telling Access which of the fields requested in your query should be used as the primary key, Access cannot uniquely identify each record. It therefore cannot allow users to edit the data and consequently all recordsets based upon pass through queries are read only.

In general you will find that it is more efficient to replace batch updates on attached tables with SQL statements that act directly on SQL Server by using pass through queries or executing stored procedures (discussed below).

Move Logic to Stored Procedures

At the simplest level, stored procedures allow you to save compiled queries (with parameters) on the server for later execution, a bit like Access' saved queries. However stored procedures can also be used to process complex business logic. Stored procedures are written using Transact SQL, which supports input and output parameters,

return values, variables, conditional processing and loop structures. Complex operations that work on batches of records will usually benefit from markedly improved performance if the logic is moved from Access' VBA to a SQL Server stored procedure.

Triggers are special types of stored procedures that are automatically invoked whenever data in a table is modified (inserted, updated or deleted). Triggers can be useful for performing complex validation, populating redundant / de-normalised fields and maintaining audit trails.

Download data for reports

When running reports, Access often retrieves data multiple times. If a report's performance is giving cause for concern, try downloading the base data for the report from SQL Server into a temporary Access table and run the report off the Access table.

ADO v ODBC

ADO/OLEDB is the successor data connectivity technology to ODBC. Whilst ODBC will be sufficient for most of your application's requirements, passing parameters to and from stored procedures is much easier using ADO. The developer also has much finer control over your application's connections when using ADO.

3.6 Assigning Permissions

You should use SQL Server's security functionality to specify permissions on your SQL Server objects (tables, views, stored procedures). As with Access, it's probably better to assign permissions to groups of users rather than to individuals. The most flexible way is to create database roles. Any user, or group of users (including NT groups) who have been given access to your database can be added to a role.

3.7 Access Data Projects

Microsoft introduced Access Data Projects (ADPs) in Access 2000. An ADP allows the developer to see and work directly with SQL Server objects from within Access. The developer can therefore create and maintain SQL Server tables, views, diagrams and stored procedures. Since ADPs don't use JET, you cannot create local tables, attach ODBC tables, or create traditional Access queries. Also you will not be able to use DAO code to manipulate your tables and queries. Therefore whilst ADP architecture is an interesting innovation and may be considered for use on new projects, upsizing an existing application to an ADP requires a lot more effort.

3.8 Backups

SQL Server backups can be made while the server is on line and the database is in use. It also keeps a log of all changes made to the database in a transaction log. Full backups and transaction log backups together make it possible to recover a database right up to the point at which it failed. Once your application enters production, ensure you have a backup strategy in place.

4 Further Information

Over the past 15 years Aztec Computing has helped organisations to design and implement custom database applications using Microsoft technologies. We have built networked and client server systems, browser applications, mobile solutions and web services for a wide range of organisations including the London Fire Brigade and France Telecom. Aztec is an authorised Microsoft Certified Partner and plays an active part in several industry forums including the UK Access User Group, for which we are the London regional coordinators.

If you would like assistance to upsize your applications or have any comments on this article, please contact us at:

Aztec Computing
2 Viscount House
6a Love Lane
Pinner
Middlesex
HA5 3EF

Tel: (020) 88 66 55 77
Fax: (020) 88 66 61 16
Email: info@azteccomputing.com
Web: www.azteccomputing.com