v4.8 Getting Started Guide: Using SpatialWare with MapInfo Professional
Table of Contents

Chapter 1: Introduction .................................................. 5
   Overview ........................................................................ 5
   Key Features and Benefits .............................................. 5
   Intended Audience ......................................................... 6
   Specifications and System Requirements ............................ 6
   Definitions ........................................................................ 7
   Organization of This Guide .............................................. 9
   Documentation Resources .............................................. 9

Chapter 2: Using SQL Server Data in MapInfo Professional .......... 10
   Overview ........................................................................ 10
   Using MapInfo Professional ............................................ 10
   DBMS Tasks ..................................................................... 10
   Setting Up an ODBC Data Source .................................... 11
   Opening a DBMS Connection .......................................... 13
   Setting DBMS Connection Preferences ............................. 13
   Opening a DBMS Table .................................................. 14
   Saving a DBMS Table .................................................... 16
   Saving a Copy of a DBMS Table ...................................... 16
   Creating a New DBMS Table ........................................... 17
   Making a DBMS Table Mappable in MI Professional .......... 19
   Changing a DBMS Table Symbol .................................... 22
   Resolving Conflicts for Linked or Live Access Tables .......... 22
   Using Per-Row Styles ..................................................... 25
   Additional Resources .................................................... 25

Chapter 3: Uploading Data with EasyLoader ......................... 26
   Introduction ................................................................... 26
   Running EasyLoader ....................................................... 27
   EasyLoader Options ....................................................... 28
   Additional Resources ..................................................... 30
   MAPINFO_MAPCATALOG ................................................. 30

Chapter 4: Managing Spatial Tables .................................. 32
   Overview ........................................................................ 32
   Creating a New Database .............................................. 32
   Spatializing a Database ................................................ 34
   Connecting to a SpatialWare Instance .............................. 35
   Creating a Table ........................................................ 35
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spatializing a Table</td>
<td>36</td>
</tr>
<tr>
<td>Preparing an Existing Table</td>
<td>37</td>
</tr>
<tr>
<td>Importing Data</td>
<td>37</td>
</tr>
<tr>
<td>Dropping a Spatial Table</td>
<td>38</td>
</tr>
<tr>
<td>Additional Notes for SpatialWare Databases</td>
<td>38</td>
</tr>
<tr>
<td>Appendix A: Troubleshooting</td>
<td>40</td>
</tr>
<tr>
<td>Known Issues and Tips</td>
<td>40</td>
</tr>
<tr>
<td>Error Messages</td>
<td>41</td>
</tr>
<tr>
<td>Database Troubleshooting and Tips</td>
<td>42</td>
</tr>
<tr>
<td>Appendix B: Advanced SQL Server Users</td>
<td>44</td>
</tr>
<tr>
<td>Overview</td>
<td>44</td>
</tr>
<tr>
<td>Performing Expert Mode Queries</td>
<td>44</td>
</tr>
<tr>
<td>Coordinate Systems</td>
<td>45</td>
</tr>
<tr>
<td>Appendix C: Sample Data</td>
<td>48</td>
</tr>
<tr>
<td>Overview</td>
<td>48</td>
</tr>
</tbody>
</table>
Introduction

Overview

Welcome to Using SpatialWare for SQL Server with MapInfo Professional Getting Started Guide. This document provides instructions for getting the most out of your spatial data stored in MS SQL Server by displaying and analyzing it in MapInfo Professional. SpatialWare is the connection between these two products that makes it possible for spatial and non-spatial data alike to co-exist in a secure RDBMS such as MS SQL Server.

Spatial data is information that contains a geographic element so that it can be displayed on a map. By storing this data in SQL Server along with your business data, all the strengths of relational databases are available: security, multi-user access, storage efficiency and more.

This Getting Started Guide shows how easy it is to bring your data into MapInfo Professional. It opens up a wide range of functionality for displaying, editing, and analyzing your data so you can maximize your investment by making better business decisions. Thematic mapping, map object editing, redistricting, and geographic analysis are just a few of the features available in MapInfo Professional. This Getting Started Guide and the documentation with SpatialWare and MapInfo Professional offer a wealth of resources to help you manage business and spatial data effectively.

With MapInfo Professional and SpatialWare for SQL Server, spatial data may be a part of any SQL Server database insert, update or delete. You can query both spatial and non-spatial data within a single SQL Server query. Spatial data used in conjunction with non-spatial data is being used by companies to find their next customer, grow the number of products and services that are offered to customers, and serve customers better. Spatial data can help answer key questions such as: Where can I buy or service products? Can you get a service truck out to me today? Do you offer cellular service in my area? All these answers depend on knowing accurate location information. SpatialWare and MapInfo Professional let you capitalize on this information.

Key Features and Benefits

SpatialWare provides the following key features and benefits:

- A complete integration of spatial and attribute data
- A tight integration between Microsoft SQL Server and MapInfo Professional in a client/server environment
- Makes SQL Server an easy alternative location for storing and accessing TAB data
• Offers all the advantages of relational databases such as data security, multi-user access, transaction locking, and backups
• Scalability. The spatial data type enhancements are implemented entirely on the server side to exploit the scalability offered by Microsoft® SQL Server.

MapInfo Professional provides these features and benefits, among many others:
• A GUI-based query operation that can be as simple as picking functions and operators from a list to a full query engine for expert users.
• A comprehensive array of mapping display, editing and analytical features to see data trends, relationships and patterns that you might not have seen before in your data.
• ODBC connectivity for secure data retrieval and manipulation.

### Intended Audience

This document is intended for users of SpatialWare for SQL Server who wish to access spatial data for viewing and manipulating it via the MapInfo Professional interface.

### Specifications and System Requirements

SpatialWare has been thoroughly tested with MS SQL Server 2000 (Service Pack 3) on these platforms:
• Windows XP Professional (Service Pack 1)
• Windows NT 4.0 Workstation and NT 4.0 Server (both Service Pack 6A)

All of the mapping operations and features of MapInfo Professional are supported and available to SQL Server spatial databases. SpatialWare does not support three types of MapInfo Professional map objects: ellipse, arc, and rounded rectangle.

This document assumes that SpatialWare, MapInfo Professional, and MS SQL Server are installed and available for use. For installation instructions for these products please refer to the appropriate installation or user guide.
Definitions

This section defines a number of terms that are at the heart of using spatial data with SpatialWare for SQL Server and MapInfo Professional.

SpatialWare Terms

SpatialWare Instance
A SpatialWare instance is an instance of SQL Server on which SpatialWare has been installed. When SpatialWare is installed, a set of extended stored procedures is created in the master database of the instance. Also, a new data type and sets of stored procedures and functions are available to be created in a database of the instance. (They will be created in any previously spatialized databases from any earlier SpatialWare installation.)

Spatialized Database
A “spatialized” database is a SQL Server database in a SpatialWare instance. This database contains the new data type and sets of stored procedures and functions.

Spatial Data Type
A spatial data type called ST_Spatial is the name of the new data type which SpatialWare creates. It is this data type that allows spatial data to be stored and retrieved from SQL Server and other relational databases.

Spatial Table
A spatial table is a table in a spatialized database that contains a spatial column of type ST_Spatial (normally named SW_GEOMETRY) and a unique non-null integer key column (normally named SW_MEMBER). It also has a spatial index, built on the spatial column, and making use of the integer key column. A spatial table and its associated index are, roughly speaking, the equivalent in a SpatialWare database of the “TAB file” storage mechanism which MapInfo Professional uses natively.

SW_GEOMETRY is created automatically when you upload TAB files to SQL Server using EasyLoader, when you create new tables via MI Professional using File > Save Copy As or File > New Table, or run sw_spatialize_tab tool (see below).

The spatial index is maintained through database triggers when the geometries in the indexed column change as a result of an INSERT, UPDATE, or DELETE operation. A spatial index is required when searching spatial data. In the case of major data uploads, the spatial index can be dropped until the data is uploaded and re-established afterward.

Spatial Functions
SpatialWare contains over 150 spatial functions and predicates. These are extensions to the SQL language, to retrieve data from tables with the ability to compute new geometric values or qualify data based upon geometric relationships. Advanced SQL Server users may benefit by using these. For MapInfo Professional users, we recommend that you query your SQL Server data via the Query > SQL Select dialog. A wide variety of functions and operators are provided. See the MapInfo Professional User Guide Chapter 8: Selecting and Querying Data for more information.
MapInfo Professional Terms

**SW_SPATIALIZE_TAB**
This MapBasic tool allows you to “spatialize” existing tables in SQL Server by automatically adding the spatial data type, SW_GEOMETRY column, key column SW_MEMBER and spatial index. This tool ships with MapInfo Professional. See Spatializing a SQL Server Table on page 20 for instructions.

**MAPINFO_MAPCATALOG**
The MAPINFO_MAPCATALOG is a registry table for databases that stores metadata about spatial tables in a database. It is used by MapInfo Professional to access map data from spatial databases, including SpatialWare for SQL Server. More information on the MACATALOG is provided on page 30.

**EasyLoader**
EasyLoader v 7.x is a Windows utility that allows you to upload MapInfo TAB files to a database, such as Microsoft SQL Server. The spatial information in the TAB files is maintained in the SQL Server database and is available for viewing and analyzing in MapInfo Professional.

During the upload process, EasyLoader creates a MAPINFO_MAPCATALOG for the spatial database (if one does not already exist), and inserts a row into the MACATALOG to represent the uploaded table. A separate entry is made for every uploaded table.

The instructions for running EasyLoader are provided in Uploading Data with EasyLoader in Chapter 3 on page 26.

**Make Table Mappable**
This is an operation in MapInfo Professional in which a table that contains spatial data is made viewable in a Map window. This is done by adding an entry for the table to the MAPINFO_MAPCATALOG. See Making the Table Mappable on page 21 for more information.

Tables that have been uploaded with EasyLoader or created directly in MapInfo Professional are already mappable and do not need to go through this process.

Other Terms

**Coordinate Systems**
The coordinate system of a table is defined by its entry in the MAPINFO_MAPCATALOG. Data loaded using EasyLoader will create this entry automatically, otherwise use the Make Table Mappable function in MapInfo Professional to register MapInfo tables in the catalog table.

Data extracted into MapInfo Professional is always connected to the coordinate system of the Map window into which it is selected. The coordinate system of an MapInfo Professional Map window is the coordinate system of the first table extracted into it.

**Microsoft SQL Server Enterprise Manager**
SQL Server Enterprise Manager is the primary administrative tool for Microsoft® SQL Server™ 2000 for managing SQL Server servers, databases, users, logins, permissions, etc.
Query Analyzer
SQL Query Analyzer is a graphical user interface in MS SQL Server for designing and testing Transact-SQL statements, batches, and scripts interactively. SQL Query Analyzer can be called from SQL Server Enterprise Manager.

Model Database
The model database is used as the template for all MS SQL Server databases created on a system. When a database is created, the first part of the database is created by copying in the contents of the model database, then the remainder of the new database is filled with empty pages. For more information, see the Microsoft SQL Server documentation.

Organization of This Guide
This guide is divided into several chapters and appendices.

Chapter 1: Introduction and Definitions of Terms
Chapter 2: Using SQL Server Data in MapInfo Professional
Chapter 3: Uploading Data with EasyLoader
Chapter 4: Managing Spatial Tables
Appendix A: Troubleshooting
Appendix B: Advanced SQL Server Users
Appendix C: Sample Data

Documentation Resources
To get the most out of SpatialWare for SQL Server and MapInfo Professional, refer to these resources.

- SpatialWare Installation Guide for installation instructions and getting started with SpatialWare.
- This Getting Started Guide, available in hard copy and as a PDF on the SpatialWare product CD. Provides instructions for accessing data via the MapInfo Professional interface and information on managing spatial databases in SQL Server.
- SpatialWare User Guide. This HTML document is installed on your system along with the server component of SpatialWare under the \doc subdirectory. It provides in-depth information for expert users who wish to construct complex spatial queries.
- Microsoft SQL Server 2000 documentation set.
Using SQL Server Data in MapInfo Professional

This chapter provides the instructions for tasks that allow you to access spatial data stored in SQL Server via the MapInfo Professional interface.

Overview

This chapter outlines how to go about bringing your spatial and business data into MapInfo Professional so that you can fully visualize and analyze the information. This chapter focuses on database access. Please refer to the MapInfo Professional documentation set for specifics on MapInfo mapping and analytical features.

Using MapInfo Professional

Data from SQL Server is brought into MapInfo Professional via an ODBC connection to the database. With MapInfo Professional integrated with SpatialWare for SQL Server, you simply make a connection from MapInfo Professional and specify the SQL Server tables you wish to access. Additional controls allow you to choose the portion of tables you want, as well as construct queries to extract the data that meets your criteria. When you are finished modifying the data, you simply save the table and it is stored securely in SQL Server.

DBMS Tasks

This chapter covers the following database tasks that you can do using MapInfo Professional and SpatialWare for SQL Server.

- Setting Up an ODBC Data Source
- Opening a DBMS Connection
- Setting DBMS Connection Preferences
- Opening a DBMS Table
- Saving a DBMS Table
- Saving a Copy of a DBMS Table
Setting Up an ODBC Data Source

Before you can access any data in SQL Server, you must create a connection to the data source.

1. From MI Professional, choose File > Open. At the Open dialog, click the DBMS Connection button on the upper right of the dialog.
2. At the Select Data Source dialog from the File Data Source or Machine Data Source tab, click the New button.
3. Select the type of data source to create ((User or System). Click Next.
4. From the list of available drivers, choose Microsoft SQL Server.
5. Continue through the Create Data Source wizard. Provide a name and description for your data source and the server to connect to.
6. Choose set the appropriate authentication setting.

7. Check the box “Use ANSI quoted identifiers” to turn them on.

8. Continue to the next dialog and make any setting changes necessary. Click Finish. At the ODBC Microsoft SQL Server Setup dialog, review the configuration. If you need to make changes, click Cancel and go back through the wizard dialogs.

9. Click the Test Data Source button. After a successful test, click OK to leave the Create Data Source wizard. If the test connection fails, make sure the server is available and try again.
Opening a DBMS Connection

To open SQL Server data in MapInfo Professional, you must make a connection to the DBMS via ODBC.

To open a DBMS connection:

1. Choose Open File > Open DBMS Connection. The Select Data Source dialog displays. Note: If both Oracle Spatial and ODBC are installed, the Open DBMS Connection dialog displays. Choose ODBC and click New to display the Select Data Source dialog.

2. Choose the appropriate data source or click the New button to set up a data source.

The Open DBMS Connection is also available as a button on the ODBC Toolbar, as well as on several dialogs accessible from the File menu: New Table, Open and Save Copy As. The illustration below shows the button in the upper right corner of the Open dialog.

The Open DBMS Connection button is also available when you choose Table > Maintenance > Make Table Mappable and Change DBMS Table Symbol.

Setting DBMS Connection Preferences

You can set MapInfo Professional to automatically open a DBMS connection when you start the program.

To set a DBMS connection on startup:

1. Choose Options > Preferences > Startup.

2. In the Default DBMS Connection box, click the Set button. The Select DBMS Connection dialog displays. Choose a connection from the list or click New to establish a new connection.
3. To clear a previously established connection, click the Clear button.
4. Click OK twice to leave the Preferences dialog.

Opening a DBMS Table

Once you have a connection established, you are ready to open a table from SQL Server.

To open a table from SQL Server:

1. Choose File > Open. The Open dialog displays.
2. If you haven’t established a connection to the database, click the Open DBMS Connection button in the upper right of the dialog. The Open DBMS Connection dialog displays. Click New and choose the appropriate data source for this connection.
3. From the Open dialog, choose the data source from the File of Type drop-down list to display available tables. If you wish, click the Filter Tables button to control the types of tables that are displayed in the list.
4. Choose the table(s) you wish to open, choose the preferred view and click Open. The Open DBMS Table Options dialog displays.

5. In the MapInfo TAB File location box, a default path and filename of the TAB to be created displays. Click the Browse... button if you wish to save it to another location and/or with another filename.
6. Choose either Standard mode (default) or Expert mode. For Standard mode, if you wish to choose only a portion of the table, click the Column Filter and/or Row Filter buttons to specify. In each case a Picker dialog displays.

For Expert mode, click the SQL View button. The Enter SQL Query dialog displays. Here you can edit the displayed query, load a previously saved query, or retrieve the last query used. For more information on Expert mode see Standard versus Expert Mode on page 16 as well as Appendix B: Performing Expert Mode Queries.

7. Choose if the table will be downloaded locally (Linked table) or accessed live. Linked tables are downloaded locally, yet maintain links to the database. Live access tables remain on the secure server. For Live Access, click the Cache box if you wish MapInfo to store the database objects from the initial or current mapview in memory. This speeds up subsequent operations. If Cache is not checked the objects are not stored in memory, thus saving memory; however subsequent operations must read the data from the database each time they are executed.

8. Click OK. If a local TAB file already exists for this table, click OK to overwrite it. The data is opened and displays in the appropriate view (Map window or Browser window) unless you chose No View at the Open dialog. You are now set to carry out any of the mapping and analytical features available in MapInfo Professional.
Standard versus Expert Mode

Most users will use the Standard Mode in which they will bring all or part of a table into MapInfo Professional. To create subsets of this data, use MI Pro’s SQL Select dialog that is accessible from the Query Menu. A wide variety of functions and operators are available to assist you with constructing even complex queries. See the MapInfo Professional documentation set for more information.

Use Expert mode when you need sophisticated spatial queries that cannot be created in the SQL Select dialog or when you want queries to be executed by SpatialWare on the server side. Queries can be executed in the Expert dialog using SpatialWare’s sp.spatial_query stored procedure. See the SpatialWare documentation set for more information.

Saving a DBMS Table

To save your table:

1. Choose File > Save Table. Choose the name of the table and click OK.
   You may be presented with the Resolve Conflicts Dialog if MapInfo has detected conflicts between the data in the database and the data saved in the database. See Resolving Conflicts for Linked or Live Access Tables on page 22 for more information.

2. For linked tables, the message "Would you like to refresh the Linked Table" displays. If you click OK, MapInfo Professional will download a fresh copy of the data from the server.

Saving a Copy of a DBMS Table

To save a copy of your DBMS table:

1. Choose File > Save Copy As. If the table is open, the Save Copy of Table As dialog displays. If more than one table is open, the Save Copy As dialog displays where you must select the appropriate table.
   At least one table must be open for Save Copy As to be available.

2. If necessary, establish a connection to the DBMS by clicking the Open DBMS Connection button in the upper right corner of the dialog.

3. Pick the name of the data source from the Save As Type drop-down list.

4. Specify a new table name for the copy.

5. Set the projection (optional) by clicking on the Projection button.

6. Click Save. The Save As dialog displays.

7. Choose the location and/or filename for the local copy of the TAB file that is created during this operation.

8. Click Save.
Creating a New DBMS Table

There are several ways to create a spatial table. In each method the necessary spatial columns and indices are automatically created. See Chapter 3: Introduction to Spatial Data, for more information about spatial tables and columns.

- **EasyLoader** utility creates the necessary indices and columns automatically from any TAB file and uploads it to SQL Server. See Chapter 5 for more on EasyLoader.
- From **File > Save Copy As**, you can turn an existing table into a spatial table and store it in SQL Server. See Saving a Copy of a DBMS Table on page 16.
- From **File > New Table**, create a table from scratch or pattern it after an existing table. The instructions are provided below.

**File > New Table**

To create a new table that you wish to store in SQL Server:

1. Choose File > New Table. The New Table dialog displays.
2. Choose the view for the new table: Open New Browser and/or Open New Mapper or neither.
3. Set the initial table structure by choosing to Create New or Using Table and choosing a table as the template. The table must be open for Using Table to be active.
4. Click Create. The New Table Structure dialog displays.
5. Add field(s) and provide name, type and width as needed. Reorder as necessary. Set the projection by clicking the Projection button.
   By default, the Table is Mappable checkbox is checked. All newly created tables are mappable, meaning that a spatial object column will be created and the table can be displayed in a Map window (even if the table contains no map objects).
   If you unchecked the Open New Mapper in step 2, then Table is Mappable is enabled. If you clear the Table is Mappable control, you will create a non-mappable table. For example, you may want to do this when you only want to upload data.
6. Click Create. The Create New Table dialog displays.
7. Choose the data source name from the Save as Type drop-down list or click the Open DBMS Connection button to establish a connection to a data source.
8. Give the new table a name and click Save. The New DBMS Table Options dialog displays.

![New DBMS Table Options dialog](image)

9. From the General tab, set the filename and location of the local TAB file that will be created for the new DBMS table.

10. To have a key column created for the table, keep the Key column checkbox checked. Type in a name for the column or accept the default SW_MEMBER. Without a key column, the DBMS table will only be opened and read as a Linked table. It will not be editable.

11. Choose either Download Data (Linked Table) or Live Access.
   For Live Access, keep the Cache checkbox checked to keep the attributes and objects that have been read into memory accessible. Clear it if you want all the data to be fetched from the database each time you zoom or make other changes. You will get the most up to date data, but the process will be less efficient.

12. On the Spatial tab, the default Index Type tables created for SpatialWare is SpatialWare.

![New DBMS Table Options Spatial tab](image)
13. Provide a name for the Index Column or accept the default SW_GEOMETRY. This is where the spatial information about a record is stored.

14. On the Styles tab, keep the Per Row Style checkbox checked if you wish to enable per row symbology whereby every object can have its own style. If unchecked the style of all objects is determined by the default object styles that are maintained in the MAPINFO_MAPCATALOG. For more information see Using Per-Row Styles on page 25.

15. Provide a name for the column which will be used to contain the style information, or accept the default name MI_STYLE.

16. Set the Default Object Styles by clicking on the appropriate button for Symbol, Line and/or Region. These styles are used if the Per Row Style setting is not turned on for the table (unchecked) or the style information for a particular object does not exist (style column for that row is empty). The information set here is entered into the MAPINFO_MAPCATALOG.

17. Click OK to leave the New DBMS Table Options dialog. The new table is created and displayed according to the settings you chose in step 2. Additionally, an entry for the table is automatically added to the MAPCATALOG.

Making a DBMS Table Mappable in MI Professional

If you have a table in your SQL Server database that cannot be displayed in a Map window, it is said to be non-mappable. You can make a non-mappable table mappable. This requires that the table structure be prepared to accept spatial information and an entry for the table added to the MAPINFO_MAPCATALOG.

Note: If you have created the table using MapInfo Professional or used EasyLoader to upload it, the table is already mappable. This section pertains to other databases that you wish to bring into MapInfo and cannot use MapInfo Professional or EasyLoader.

The procedure to follow depends on whether the table contains spatial data or not. If it does, you simply need to record the table in the MAPCATALOG. See Making the Table Mappable on page 21.
If your database does not yet contain spatial data, you will need to prepare the database for spatial data (spatialize), add the table(s) to the MAPCATALOG (make the table mappable), and then create the spatial objects for your data by geocoding, digitizing, or otherwise drawing the objects. For instructions on spatializing the table and making it mappable, see the next two sections. For creating spatial objects for your data, see the MapInfo Professional documentation set for more on geocoding, digitizing and drawing.

**Spatializing a SQL Server Table**

Preparing a SQL Server table to accept spatial information is known as spatializing the table. The process involves creating the necessary columns to hold the spatial information. MapInfo Professional provides a MapBasic tool to carry out this operation. The Spatialize Table tool can be found in the MapInfo Professional Tools directory after installing SpatialWare. It is called SW_SPATIALIZE_TAB.MBX.

You must have ALTER privileges for the table you want to spatialize. The table must reside in a spatially enabled database. Check if the database has been prepared for spatial information. You can check whether the database has been prepared for spatial information by executing the following query from Query Analyzer:

```sql
select dbo.hg_version()
```

If the database is NOT properly prepared, you will get an 'Invalid object name' or 'Incorrect syntax' error message. Otherwise, you will get the current SpatialWare version string.

If you have admin privileges, you can spatialize the database by following the procedure in [Spatializing a Database on page 34](#).

Note: The Spatialize Table tool does not put in any spatial data; it is only preparing the table to accept spatial data. To populate the table with spatial data, you would use the features in MapInfo Professional, such as geocoding to add coordinates to point records, or the drawing tools to create other spatial objects.

To run the Spatialize Table tool:

1. From the MapInfo Professional Tools menu, choose Run MapBasic Program and navigate to the Tools directory and run SW_SPATIALIZE_TAB.MBX. This tool is also listed on the Tool Manager under Spatialize SQL Server Table.
2. Make a connection to the SQL Server database where the table you wish to spatialize is stored. At the Connection Manager dialog, select an existing connection or click New to create one. When you have a connection, the Spatializing the Table dialog displays.

   ![Spatializing the Table](image)

3. Choose the Table from the Tables drop-down list.
4. To create the spatial column, click the Add Spatial Column button. A column of type ST_SPATIAL is created and listed in the drop-down list. The default column name is SW_GEOMETRY.

5. To set the Key column, choose the appropriate column and click the Add Key Column button. The chosen column will be renamed with the default Key column name SW_MEMBER. The column will not be renamed if the existing column already has a Key defined for it. It must be of type integer and be unique, not null.

6. As a final step to spatialize the table, click the Spatialize button. The spatialize operation is carried out. The table is now ready to accept spatial information. Proceed to the next section to make the table mappable so it can be viewed in MI Pro.

Making the Table Mappable

This process will add an entry for the table in the MAPINFO_MAPCATALOG, a registry table for databases that stores metadata about spatial tables. In order for a table to be viewable in MapInfo Professional, it must have an entry in the catalog. This is a one-time process for a table.

To make a table mappable:

1. Choose Table > Maintenance > Make DBMS Table Mappable or click the Make Table Mappable button from the DBMS Toolbar. The Select DBMS Table dialog displays the available non-mappable tables.
   If a connection to the database has not been established the Open DBMS Connection dialog displays first. See page 13 for more on how to make a connection.

2. From the Select DBMS Table dialog choose the appropriate table and click Open. The Make Table Mappable dialog displays.

   ![Make Table Mappable dialog](image)

3. Choose the Index Type called SpatialWare. This is the default for SpatialWare tables.

4. Choose the index column for the table. The default column is SW_GEOMETRY.

5. To set the object type that will be stored in this table, choose from All, Points, Lines or Regions. All means that all three object types may be contained in the table.

6. Keep the Per Row Style checkbox checked if you wish to enable per row symbology whereby every object can have its own style. If unchecked the style of all objects is determined by the default object styles (maintained in the MAPINFO_MAPCATALOG).
7. Set the column to be used to contain the style information. By default, the column is called MI_STYLE.

8. Set the default Object Styles by clicking on the appropriate button for Symbol, Line and/or Region. These styles are used if the Per Row Style setting is not turned on for the table (unchecked) or the style information for a particular object does not exist (style column for that row is empty). The information set here is entered into the MAPINFO_MAPCATALOG.

9. Set the Projection if necessary. The projection of this table must match the projection used by the corresponding database table.

10. Click OK. The table is now mappable. To view the table in a Map window, you must re-open it.

Changing a DBMS Table Symbol

To change the style for objects in a mappable DBMS table:

1. Choose Table > Maintenance > Change DBMS Table Symbol or click the Change Table Symbol button from the DBMS Toolbar. The Open dialog displays.

2. From the Open dialog choose the appropriate table and click Open. The Change Table Object Style dialog displays.

3. Set the new styles for the table by clicking on the appropriate button for Symbol, Line and/or Region. The appropriate style picker dialog displays for the map object.

4. Click OK. Close the table and re-open it in order for the changes to take effect.

Resolving Conflicts for Linked or Live Access Tables

Because the records in a table are from a DBMS, it is possible that other users may have changed or deleted them on the database since the table was downloaded into MapInfo. As a result, conflicts may exist between the data residing on the database and the new data that you want to save to the database.

When saving a table, if MapInfo finds conflicts between the data on the database and any records in the remote table since it was downloaded, the Resolve Conflicts dialog automatically appears. Use this dialog to resolve conflicts as explained below.

This process will be invoked whenever an attempt to save a remote table detects a conflict in an update. The dialog allows the user to choose which fields from the conflicting records will be used to update the database. Three instances of the record being updated must be considered:

- The original server state of the record (the record as it appeared when initially extracted from the database).
- The current local state of the record (the record as it appears in the session of MapInfo making the update, possibly after editing by the user).
• The current server state of the record (the record as it appears in the database at the time of the update).

A conflict exists when the original state of the record does not match the server state, implying that another user has updated the database since it was extracted by MI Pro.

The conflict resolution dialog will appear once for each conflicting record. At any point in this process, the user may choose to leave this interactive mode and have the rest of the conflicts resolved automatically. The user may choose to use all local values or all server values.

For each conflict found during a commit, the user will be presented with a modal dialog box. This box will display enough information for the user to decide which data values to use to update the row in question.

<table>
<thead>
<tr>
<th>Type of conflict</th>
<th>Default resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value changed on server, local value unchanged</td>
<td>Use server value</td>
</tr>
<tr>
<td>Value changed on server, local value changed</td>
<td>Use local value</td>
</tr>
</tbody>
</table>

Column Shows the name of the column in the record that has data which is in conflict and needs to be resolved.

If the column name is too long and does not fit into the list box, a truncated representation will be shown instead. Its full value is displayed in the Column field, below.
| Original DBMS | Shows the original data as it appeared when extracted from the database.  
| If the original data is too long and does not fit into the list box, a truncated representation will be shown instead. Its full value can be displayed in the Original field, below. |
| Current MapInfo | Shows the data as it appears in the MapInfo linked table you are trying to save.  
| The MapInfo field will be blank if the record has been deleted from the MapInfo database.  
| If the data from the MapInfo linked table is too long and does not fit into the list box, a truncated representation will be shown instead. Its full value can be displayed in the MapInfo field, below. |
| Current DBMS | Shows data as it appears in the remote database at the time of the update. (This data might have been changed by another user since it was downloaded into a MapInfo table.)  
| The Current field will be blank if the record has been deleted on the remote database, after being downloaded into a MapInfo linked table.  
| If the current data from the remote database is too long and does not fit into the list box, a truncated representation will be shown instead. Its full value can be displayed in the Current field, below. |
| Current MapInfo | Check this box to update the remote database with the value from the MapInfo linked table.  
| If the current record on the database or the MapInfo record was deleted, then this check box is not available. Instead, you must use the MapInfo or Current button, below. |
| Current DBMS | Check this box to retain the current value on the remote database.  
| If the current record on the database or the MapInfo record was deleted, then this check box is not available. Instead, you must use the MapInfo or Current button, described below. |
| MapInfo button | This button selects all the MapInfo values. If the record you are trying to resolve was deleted from the MapInfo table, selecting the MapInfo button will delete the record from the remote database.  
| If the record you are trying to resolve was deleted from the remote database, selecting the MapInfo button will insert the new record into the remote database. |
| Current button | This button selects all the current DBMS values. If the record you are trying to resolve was deleted from the MapInfo table, selecting the Current button will ignore the deletion and retain the current record in the remote database.  
| If the record you are trying to resolve was deleted from the remote database, selecting the Current button will insert the new record into the remote database. |
Using Per-Row Styles

Per-row styles allows the use of different object styles on a remote database. You can modify the styles for individual or groups of objects and save the styles to a DBMS table.

To use per-row styles, the DBMS table must be set up correctly:

- The MAPINFO_MAPCATALOG table for the database must contain columns, which control the use of styles. They are RENDITIONTYPE, RENDITIONCOLUMN, and RENDITIONTABLE.
- The map table itself must have a character column wide enough to store the complete style string. Style strings vary in width. We recommend a minimum of 50 characters be provided. For custom symbols, which use much longer strings, allow 200 characters.
- The entry for the table in the MAPCATALOG must be set correctly. This means that the RENDITIONTYPE is 1, and the RENDITIONCOLUMN contains the name of the column that will contain the style string (default is MI_STYLE). This entry is set by EasyLoader or it may be set using the Make Table Mappable function.

Additional Resources

Now that you have successfully brought your data into MapInfo Professional, you are ready to take advantage of all the mapping display and analytical features available to Pro users. Refer to the MapInfo Professional User Guide and Online Help for information.
Uploading Data with EasyLoader

This chapter explains how to use EasyLoader, a MapInfo utility that uploads TAB files to a DBMS database.

Introduction

EasyLoader v 7.x is a Windows utility available from MapInfo Corporation that allows you to upload MapInfo TAB files to a remote database, such as Microsoft SQL Server. The spatial information in the TAB files is maintained in the SQL Server database and is available for viewing and analyzing in MapInfo Professional.

EasyLoader can upload tables to SQL Server containing points, lines, polygons and text objects. Ellipses, arcs and rounded rectangles are not supported.

EasyLoader is installed into the Tools directory during the MapInfo Professional installation process. It is also available as a download from the MapInfo website.

The Microsoft SQL Server driver should be used when connecting to remote databases through EasyLoader. This driver is provided when MapInfo Professional is installed.

The EasyLoader upload process will:

- create spatial columns and indices that are required for SpatialWare and MapInfo Professional.
- add an entry to the MAPCATALOG for each uploaded table (required).
- provide options for how you wish the table to be processed.

EasyLoader can be run from within MapInfo Professional or as a standalone program using its executable file easyloader.exe. It can also be run as a command line utility by using its command line arguments. This is useful for automating loading or replacing data on the server.

Prerequisites for EasyLoader

The following are necessary prior to running Easyloader:

- The database to which you want to upload data must be spatialized first. See page 34.
- You must have privileges to create and update tables on the database in which the new table will be created. You also need permission to update the MAPINFO_MAPCATALOG, a registry table containing metadata about spatial data.
• If you are using EasyLoader to initially create the MAPCATALOG, you will need permission to create a new user, MAPINFO, which is the user that will own the MAPCATALOG. More on the MAPCATALOG is found on page 30.

The following section describes the steps and options to upload a TAB file. This information is also accessible from the EasyLoader dialog Help button.

Running EasyLoader

To upload MapInfo TAB files using EasyLoader:

1. Run the EasyLoader from the MapInfo Professional Tools menu. If necessary, run Tool Manager to load and/or autoload EasyLoader onto the Tools menu. The main EasyLoader dialog displays.

2. Under Connection Information, click the ODBC button to connect your SQL Server database. Provide the necessary connection information (e.g., data source name or User ID, password and server name). Click OK to return to the EasyLoader dialog box.

3. Click the Source Tables button to display a list of MapInfo tables from a single directory. When tables are selected for uploading, the names will display in the MapInfo Tables list box.

4. Choose the tables and select the appropriate Server Table processing task (Create new table, Append to existing table, Replace existing Table). Additional options are available in the Options section beginning on page 28.

   Note: The Upload button is not available until table(s) are chosen.

5. To create local TAB files, provide a directory or browse to its location. By default, EasyLoader will not generate these files. The file naming convention for these tables is yourServerTableName_srv.tab.
6. To set options for the upload process, click the Options button. The Options dialog will display. See Options section for an explanation of available options. Click OK to return to the main EasyLoader dialog.

![EasyLoader Options Dialog]

7. Click the Upload button to start the upload process. Close EasyLoader once the upload process is finished.

If you haven’t already created the Spatial Index during the upload process, do so now by either executing a create index statement or re-uploading the table, making sure this time to create the Spatial Index and replace the table (see steps 1-3).

**EasyLoader Options**

The options available in EasyLoader are described below.

**Table Processing Options (Main Dialog)**

**Create New Table**

A server table will be created with the name that you specify. If this option is chosen and a table with the same name already exists on the server, an error message will display, making you aware of this problem. You will need to use a different name or choose the option: Replace Existing Table, in order to upload the table.

**Replace Existing Table**

If a server table of the same name already exists, it will be dropped and a new table will be created to match the MapInfo table being uploaded when this option is selected.

**Append to Existing Table**

The MapInfo table will be appended to the server table if the server table exists and the structure of the two tables match. Otherwise, you will get an error and the table will not be uploaded.
Append All to One Table

All MapInfo tables listed are uploaded to a single server table. The server table name is the one visible in the Server Table box. This feature is meant to be used to upload tables with the same structure and projection to one table. For example, instead of creating a new table for each street layer, check the Append All to One box, and only one table will be created. All of the tables will then be appended to this table. Note: It is possible that some tables will not be appended if their structure differs.

When this option is used with the Replace Table option, the server table will be dropped, a new table will be created, and all tables listed will be appended to that one. When used with the Create Table options, the server table will be created, and all tables listed will be appended to that one.

Spatial Object Type

Choose from MapInfo SpatialWare, MICODE (XY with Key), and XY. The default for loading spatial data is MapInfo SpatialWare.

For SpatialWare users, the spatial object type is MapInfo SpatialWare.

MapInfo SpatialWare

To select this option, the server must have SpatialWare installed. Tables will be uploaded as spatial data.

XY and XY with MapInfo key (MICODE)

Use this option if the server does not have SpatialWare installed. The data will be stored as xy coordinates on the server. Therefore, the server table will be created as a point table. If the MapInfo table to be uploaded is not a point table and this option is chosen, the centroid will be extracted and stored on the server table if you instruct it to do so. The difference between XY and MICODE is that the MICODE will provide a MapInfo key as the spatial index, hence its performance is superior to XY.

Table Processing Options (Options Dialog)

Grant Public Access to Table

PUBLIC is granted all access to the server table.

Exclusive Use of Table

You can speed up load time on large tables significantly if you know that you will be the only one attempting to update/upload the table. By NOT checking this option, the loader will verify after each commit that no other updates are made to the table while it is being uploaded. Checking this option will prevent this check from occurring which can make a significant change to the run time for large tables.
Create Unique Index

A unique index is created on the column `sw_member` for SpatialWare, `mi_prinx` for Oracle, or `mi_sql_rec_num` for XY and MICODE. These columns are sequential numbers that are generated by the loader. These columns are always created, but do not have to be indexed.

Create Spatial Index

For SpatialWare tables the index is created on the geometry column. A spatial index is created and Update Statistics is executed after an r-tree index is created. You may also build your own spatial index to suit your specific needs. If you choose to do this, clear this check box to save time in loading.

Create IDENTITY Column (SQL Server only)

Check this box if you wish to create the key column (sw_member) with IDENTITY properties. When this feature is in use, the unique key column values will be generated automatically by SQL Server. You do not need to fill in the key manually when a new row is inserted.

To upload the table without IDENTITY, select the Options button and uncheck Create IDENTITY Column.

Style Column

This allows you to specify that per-row symbology is to be loaded with the data. The symbology is loaded as a text string in the column specified. The name of the column to be used may be specified in the edit text box. The name is initialized to the default column name, which is `MI_STYLE`.

*Note:* To load per-row symbology, the MAPINFO_MAPCATALOG for the database must contain the following columns `RENDITIONTYPE`, `RENDITIONCOLUMN`, `REDITIONTABLE`, and `NUMBER_ROWS`. See the section MAPINFO_MAPCATALOG for more information.

Additional Resources

For more information on using EasyLoader, please view the online help provided with this utility, "EasyLoader.HLP".

MAPINFO_MAPCATALOG

The MAPINFO_MAPCATALOG is a registry table for databases that stores metadata about geometry tables in the database. Using the tablename and ownername as the key, the MAPINFO_MAPCATALOG identifies the geometry column, geometry type, projection, projection
bounds, and table and feature level rendition information. The MAPINFO_MAPCATALOG is used by a number of MapInfo products that access map data from databases, such as MapInfo Professional.

If a MAPCATALOG does not exist, it can be created during the upload process when running EasyLoader.

When using ODBC, EasyLoader will not issue public grants, which must be done by other means. If you do not have adequate permissions, creation will not succeed and the table will not be uploaded. Required permissions are to create a new user, MAPINFO, and create a new table.

After the table is uploaded, an entry is made in the MAPINFO_MAPCATALOG to represent that table. A separate entry is made for every uploaded table.

If the table is made up of a single type of object, then the server object type is restricted to that type, otherwise the type is ALL. Also, the symbol clause generated is based on the server type. For example: After uploading the table ‘States.tab’ the server type will be X.2 (polygons), where X is a number that represents either the DB2, IUS, SQL Server, or Oracle Spatial, and the symbol clause will have only the information for a polygon.

**MAPINFO_MAPCATALOG Format**

The MAPINFO_MAPCATALOG has the following table structure:

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPATIALTYPE</td>
<td>FLOAT</td>
</tr>
<tr>
<td>TABLENAME</td>
<td>CHAR(32)</td>
</tr>
<tr>
<td>OWNERNAME</td>
<td>CHAR(32)</td>
</tr>
<tr>
<td>SPATIALCOLUMN</td>
<td>CHAR(32)</td>
</tr>
<tr>
<td>DB_X_LL</td>
<td>FLOAT</td>
</tr>
<tr>
<td>DB_Y_LL</td>
<td>FLOAT</td>
</tr>
<tr>
<td>DB_X_UR</td>
<td>FLOAT</td>
</tr>
<tr>
<td>DB_Y_UR</td>
<td>FLOAT</td>
</tr>
<tr>
<td>COORDINATESYSTEM</td>
<td>CHAR(254)</td>
</tr>
<tr>
<td>SYMBOL</td>
<td>CHAR(254)</td>
</tr>
<tr>
<td>XCOLUMNNAME</td>
<td>CHAR(32)</td>
</tr>
<tr>
<td>YCOLUMNNAME</td>
<td>CHAR(32)</td>
</tr>
<tr>
<td>RENDETIONTYPE</td>
<td>INTEGER</td>
</tr>
<tr>
<td>RENDETIONCOLUMN</td>
<td>CHAR(32)</td>
</tr>
<tr>
<td>RENDETIONTABLE</td>
<td>CHAR(32)</td>
</tr>
<tr>
<td>NUMBER_ROWS</td>
<td>INTEGER</td>
</tr>
</tbody>
</table>
Managing Spatial Tables

This chapter covers some database tasks for preparing and using spatial databases.

Overview

This chapter covers a number of database management tasks when working with spatial data in MapInfo Professional. In general, this chapter shows the manual way to perform the task through SQL statements and stored procedures. Some of these tasks can be done via MapInfo Professional or by using MapBasic applications, as described in the previous chapter.

The following tasks are presented in this chapter:

- Creating a database
- Spatializing a database
- Creating a table
- Spatializing a table
- Preparing an existing table
- Importing data
- Creating an R-tree index
- Performing checks on a database
- Dropping a table
- Uninstalling

Creating a New Database

You create a new database using SpatialWare for SQL Server the same way you would create any SQL Server database.

The Enterprise Manager provides a Create Database wizard to walk you through the process.

The following are conditions that are specific to databases that SpatialWare will access.

- "Recursive triggers" setting turned OFF.
  This can be managed through Enterprise Manager, by selecting a database and following Properties > Options > Settings.
• Set QUOTED_IDENTIFIER to ON (recommended).
   Both the client and server side should have the same settings. SpatialWare stored
   procedures are created by default with QUOTED_IDENTIFIER set to ON, but SQL Server
   executes the spatial queries in the current QUOTED_IDENTIFIER environment (which
   could be set to OFF). Problems may arise if the SQL Server QUOTED_IDENTIFIER
   settings are not the same.

There are two ways to set quoted identifiers on: set for the entire server, or set for individual
databases.

To set for the entire server, highlight the server name and right click, then choose Properties. The
SQL Server Properties dialog displays. Select the Connections tab and select the Quoted
Identifiers check box. For example:

![SQL Server Properties](image)

To set for an individual database, highlight the database name and right click, then choose
Properties. The database Properties dialog will display. Select the Connections tab and select the
Quoted Identifiers check box.

SpatialWare for SQL Server does not attempt to determine the current QUOTED_IDENTIFIER
environment. Its stored procedures are created with this setting ON. It also parses spatial queries
as though this setting is ON. However, whatever is passed to SQL Server will execute in the
current QUOTED_IDENTIFIER environment.
Spatializing a Database

Spatializing a database is a process by which a database is prepared to accept spatial data and perform spatial data operations. In addition a MAPINFO_MAPCATALOG is created which is a registry table for databases that stores metadata about spatial tables. MapInfo Professional uses the information in the MAPCATALOG when accessing spatial data, in particular to retrieve coordinate system information.

Note: A database must be spatialized to be accessible from SpatialWare for SQL Server.

Two spatialized databases, known as Georgetown and World, are installed with SpatialWare and attached and ready for use. See Appendix C: Sample Data for details.

To spatialize additional databases, run a stored procedure on it via Query Analyzer or via MI Professional. See below.

Via Query Analyzer

To spatialize additional databases, you can execute a stored procedure using SQL Server’s Query Analyzer. Execute the following statement:

exec sp_spatialize_db

You must do this for every existing database that you plan to use with SpatialWare and any new database you create.

Via MapInfo Professional

The stored procedure above can also be run via MapInfo Professional.

1. In MapInfo Professional choose File > Open. To connect to the database, click the Open DBMS Connection button in the upper right corner of the Open dialog. Choose an existing connection to the data source or create a new one.

2. In the Files of Type list, select the data source name for the database.

3. Select a table name.
   Choose a table from the Open Table dialog (this is necessary to continue with the spatializing procedure). Click Open.

4. In the Open DBMS Table Options dialog, check the Expert Mode button, then click SQL View button. The Enter SQL Query dialog displays.

5. Input the following statement in the edit area provided.

   exec sp_spatialize_db
   
   Click OK. The selected database is then spatialized.
Connecting to a SpatialWare Instance

To connect to an instance of SpatialWare use the Microsoft SQL Server utility Query Analyzer which will execute the SpatialWare statements.

Creating a Table

A spatial table must contain a column of type ST_Spatial. This is the column where the spatial geometry is stored. By default, this column is called SW_GEOMETRY, but it can be any legal name.

A spatial table must also have a key column that is of type integer, not null, and unique (preferably primary key). This column is used for maintaining spatialized columns, and for R-tree indexing. By default, this column is called SW_MEMBER, but it can be any legal name.

For tables to be accessible via MapInfo Professional, the table name must be limited to 32 characters in length.

Creating a New Table

There are several ways to create a spatial table. In each method the necessary spatial columns and indices are automatically created.

- EasyLoader utility uploads TAB files to SQL Server. See Chapter 4 for more on EasyLoader.
- In MapInfo Professional, using File > Save Copy As, you can turn an existing table into a spatial table and store it in SQL Server. See Saving a Copy of a Table in Chapter 4.
- In MapInfo Professional, choose File > New Table and create a table from scratch or pattern it after an existing table. See Creating a Table in Chapter 4.
- Using Query Analyzer, SpatialWare for SQL Server can be employed directly. See examples below.

CREATE TABLE Example

The necessary columns can be included easily when you create a new table. The following example creates a new table called testcon with a key column called SW_MEMBER and a geometry column called SW_GEOMETRY. The key column is set to IDENTITY so SW_MEMBER is updated automatically. This helps in multi-user environments:

```sql
create table testcon (  
    sw_member integer not null IDENTITY Primary Key,  
    sw_geometry st_spatial
)
```

Although SQL Server allows table names of up to 128 characters, your spatial table names must be no more than 32 characters in length in order to use them with MapInfo client software.
Spatializing a Table

After creating your table, you need to spatialize it. Use either the MapBasic tool provided, or for a more explicit method execute the Create R-tree procedure (sp_sw_create-rtree) in Query Analyzer. It is really a two-step process beginning with sp_sw_spatialize_column, but sp_sw_create-rtree calls sp_sw_spatialize_column internally.

Note: You do not need to spatialize a table if you uploaded it with EasyLoader or used MI Professional’s New Table or Save Copy As features to create it. The spatialize procedures have already been performed.

MapBasic Tool

A MapBasic tool called Spatialize Table is included with MapInfo Professional to spatialize your table. SW_SPATIALIZE_TAB.MBX is located in the MapInfo Professional Tools directory. For instructions on how to use it see page 20.

Spatialize Column Procedure

For a more detailed method of spatializing a table, use the Spatialize Column procedure sp_sw_spatialize_column. This procedure does two things: it creates an auxiliary table and triggers on the base table for maintaining spatial data.

Note: If you are also going to create an r-tree index, this step is not necessary, as it is called internally from sp_sw_create-rtree (see below).

To spatialize a column, execute the Spatialize Column procedure in the Query Analyzer. The syntax is as follows:

```sql
exec sp_sw_spatialize_column '<owner>', '<table_name>', '<spatial_column>', '<key_column>' [, 'filegroup']
```

For example:

```sql
exec sp_sw_spatialize_column 'dbo', 'testcon', 'sw_geometry', 'sw_member'
```

If filegroup is not specified, then the default filegroup is used.

Creating an R-Tree Index for a Table

If a table contains a non-null and unique key column of type integer, a spatial (R-tree) index can be created on its spatial column. An R-Tree index is a spatial index that is used to improve the speed of queries issued against a particular table. It is an alternative access method to quickly determine whether a particular feature qualifies against a spatial predicate.

An R-tree index is automatically created on a table when you spatialize it using the Spatialize Table MapBasic tool (see page 20).

To generate an R-tree table for your table, run the Create R-Tree procedure on the table. Its syntax is as follows:
exec sp_sw_create_rtree 'owner', 'table_name', 'spatial_column', 'key_column', [ file_group ], [ prefetch_size ], [ cache_size ]

For example:

exec sp_sw_create_rtree 'dbo', 'testcon', 'sw Geometry', 'sw member', null, 200, 10000

The last three items, file group, prefetch size, and cache size, are optional. They can be added to fine-tune processing time. Their default values are null, 100, and 10000 respectively. Prefetch size defines how many rows will be read at a time, and cache size defines a size of an internal buffer in K bytes.

To estimate the amount of cache size that the R-tree creation process can make use of for a table, multiply the number of rows in the table by 7%. For example, 7% x 1 million rows = 70000. Although you may choose any value larger than 100 for cache size, increasing the cache size relative to the size of your table will improve performance.

Prepared an Existing Table

To prepare an existing table, columns must be added. You do not need to add a key column if your table already has one, but it must contain unique values and it must be defined as not null, of type integer.

1. Alter the table to add a key column called SW_MEMBER and a column of type ST_Spatial, called SW_GEOMETRY. Set the key column to IDENTITY.

   alter table table1 add
   sw_member integer not null IDENTITY Primary Key,
   sw_geometry st_spatial

   The values of sw_member are set automatically for the existing records.

2. You must now spatialize your table in order to use it. See page 20.

Note: Rather than alter your existing table, you can also create a table containing the unique key column and ST_Spatial type column and join it to your existing table in spatial queries.

Importing Data

Data can be imported into SQL Server in two ways: via EasyLoader or by saving a copy of an existing table from MapInfo Professional.

MapInfo EasyLoader

MapInfo provides an upload utility called MapInfo EasyLoader. It imports TAB tables containing attribute and geometry (map object) data from MapInfo Professional, MapX, or MapMarker into a SpatialWare database.
EasyLoader is a stand-alone, client-side upload utility specific to MapInfo data sets. An ODBC link is used on the client side for data transfer. EasyLoader is presented in Chapter 5: Uploading Data using EasyLoader.

**Save Copy As Method**

You can import data and automatically create the necessary spatial indices and columns through MapInfo Professional's Save Copy As feature. See Chapter 4 for instructions.

**Dropping a Spatial Table**

To drop a spatial table, use the following statements:

1. To drop the table's R-tree index. Type:
   ```sql
   exec sp_sw_drop_rtree '<owner>', '<table_name>', '<spatial_column>', '<key_column>'
   ```
2. Despatialize the spatial column (the column of type ST_Spatial). Type:
   ```sql
   exec sp_sw_despatialize_column '<owner>', '<table_name>', '<spatial_column>', '<key_column>'
   ```
3. Drop the table.
   ```sql
   drop table <table_name>
   ```

Steps 1 and 2 can also be accomplished by using the "despatialize" capability of SW_SPATIALIZE_TAB.MBX (see Spatializing a SQL Server Table on page 20).

**Additional Notes for SpatialWare Databases**

**Naming Objects**

Although SQL Server allows object names of up to 128 characters, your spatial table and column names must be no more than 32 characters in length in order to use them with SpatialWare.

**Geometry Sizes**

Links between client applications and the server database have been tested with single geometries up to 250,000 nodes in size.

**Unique Identifier Column**

SpatialWare tables to be edited must have a uniquely constrained column. Without it, edited rows cannot be properly identified when being updated in the server database. So, such tables are marked as non-editable within a client.
The name of the key column can be different from SW_MEMBER, but it must be integer, not null, and defined unique or with a unique index. The geometry column can be any legal name. By default it is called SW_GEOMETRY.

**Automatically Setting SW_MEMBER**

Using an IDENTITY attribute on the column automatically updates the SW_MEMBER column each time a new row is inserted. This is an ease of use feature that frees you from having to assign a unique key value for each new row. SQL Server will automatically assign a unique value.
Known Issues and Tips

There are a number of issues or tips that you should be aware of when using MapInfo Professional with your spatialized SQL Server database. This section discusses the following topics (listed in alphabetical order):

• Datetime Functionality
• Non-Updateable Column Types
• Unsupported Map Object Types

Datetime Functionality

MapBasic

The MapBasic command server_columninfo (used in MapBasic script writing) will return an incorrect type for a datetime column in a SQL Server table (the type float is returned instead).

Column Restrictions

MapInfo clients may only display the Date portion of a SQL Server DateTime field. The time portion (if present) is not displayed. An update to a DateTime field from a client application will blank the time portion of the field, so care should be used.

Non-Updateable Column Types

The following column types cannot be updated from client applications into a SQL Server database:

• VarBinary
• Uniqueidentifier
• DateTime (This can be updated, but the time information is lost if present.)
• Ntext
• binary
• Sql_variant
• TimeStamp
Unsupported Map Object Types

The following MapInfo map object types are not supported in SpatialWare:

- ellipses
- arcs
- rounded rectangles

Error Messages

This section lists error messages and possible solutions for them.

"A cursor with the name ... already exists" – Recursive Triggers

The "Recursive triggers" setting for a spatial database must be turned OFF.

This can be managed through Enterprise Manager, by selecting a database and following Properties > Options > Settings. If this setting is ON, you can expect some insert and update operations to fail with the message

"A cursor with the name ... already exists."

or

"Maximum stored procedure nesting level exceeded...

The definition for column 'm' must include a data type

If you use the "select ... into ..." construct in the Enter SQL Query dialog, and you have a literal in the select list, it will fail. For example, the following statement:

exec sp_spatial_query 'select 101 as m into tab2 from tab1'

Returns the error:

Server: Msg 173, Level 11, State 1, Line 0
The definition for column 'm' must include a data type.

An alternative is to create the tab2 table in advance and then do:

insert into tab2 exec sp_spatial_query 'select 101 as m from tab1'

SQL Server Column not found: OBJECT

If you get the following error message on a query submitted via the MapInfo Enter SQL Query dialog or MapBasic script, then try using the fully qualified SW_GEOMETRY reference in place of the OBJECT reference.

Error: SQLServer Column not found: OBJECT
For example, the fully qualified form would be:

```sql
select WORLD.SW_GEOMETRY, WORLD.SW_MEMBER
from WORLD, WORLDCAP
where WORLD.COUNTRY = WORLDCAP.COUNTRY
```

Instead of using the general spatial column identifier 'OBJECT' without any additional qualifications:

```sql
select OBJECT, WORLD.SW_MEMBER
from WORLD, WORLDCAP
where WORLD.COUNTRY = WORLDCAP.COUNTRY
```

## Database Troubleshooting and Tips

### Size and Quantity Limits

You need to be aware of the following maximum sizes and limitations:

- Maximum size of geometry binary or string values is 2 GB.
- Maximum size of the query passed into `sp_spatial_query` is 8000 characters.
- Maximum number of tables in a join statement is ten.

### Updating and Server Security

Normally, update permissions are determined when a table is opened. However, in some cases, column permission checks from the client applications are done at commit time (when data is saved back to the server database). This happens only when SpatialWare functions are in the initial select statement used to open the table. If a column does not have the permissions for a particular user to update it, the following message is displayed:

```
'Update permission denied on column 'col_name' of object 'TABLENAME',
database 'DB_NAME' owner 'owner_name' Operation cancelled.
```

### Inserting Spatial Data – Unterminated string

Please make sure that spatial strings (i.e., parameters to `ST_Spatial()`) do not include new line characters. This can happen inadvertently when using Query Analyzer. If this occurs, then an "Unterminated string" error will result.
Overview

This appendix provides helpful information for advanced users of SpatialWare for SQL Server who wish to construct expert queries. The following topics are covered:

- Performing Expert Mode Queries
- Troubleshooting
- Coordinate Systems

Performing Expert Mode Queries

Advanced users may want to submit their own spatial queries. To do this, you should be familiar with the SpatialWare SQL syntax and should study the list of spatial SQL functions that the SpatialWare server supports.

Documentation has been provided on the SpatialWare server for users executing expert queries. This documentation is distributed in an online (HTML) format and is installed with the server component. To view this documentation, point your Windows Internet Explorer (4.0 or higher) or Netscape Communicator (4.5 or higher) browser to the index.htm file in the doc subdirectory where the server component files are installed (i.e., file:///C:/sw_4.8_s/doc/index.htm).

Additional Rules

Queries submitted via the MapInfo Enter SQL Query dialog box or a MapBasic script should either use the column name SW_GEOMETRY in fully qualified form:

```sql
select WORLD.SW_GEOMETRY,WORLD.SW_MEMBER
from WORLD,WORLDCAP
where WORLD.COUNTRY=WORLDCAP.COUNTRY
```

Or, the MapInfo Professional general spatial column identifier 'OBJECT' without any additional qualifications:

```sql
select OBJECT,WORLD.SW_MEMBER
from WORLD,WORLDCAP
where WORLD.COUNTRY=WORLDCAP.COUNTRY
```

Note: Aggregate functions cannot be used in expert queries. Stored procedures, such as sp_nearest, can also not be used in this context.
Performing Spatial Database Queries

The following are a few tips to keep in mind when performing spatial database queries using either the Enter SQL Query dialog in MapInfo Professional or a custom MapBasic program:

- The Enter SQL Query dialog adds the sp_spatial_query prefix if needed and encloses the query in quotes.
- If you have strings within the query, you don’t have to escape the string using two single quote (‘’) characters instead of one. The Enter SQL Query dialog performs this automatically. For example:
  ```sql
  select sw_member from rdpaved where rdname like 'GLENN%
  ```
  Note that the string in the where clause is ‘GLENN%’ and not “GLENN%”.

Use of square brackets in the Enter SQL Query dialog

MapInfo Professional will not allow the use of square brackets in the Enter SQL Query Dialog. However, if QUOTED_IDENTIFIER is set to ON for the database (as is recommended), you can use double quotes instead.

You need to delimit identifiers if they do not comply with the format of regular SQL Server identifiers.

Spatial Function Results and Live Access

When using SpatialWare, the results of selects using spatial functions can be edited and saved back to the original table. For example, the results of the following statement, which will be a set of buffer areas around the lakes, could be edited and the modified results saved:

```sql
exec sp_spatial_query ' 
  select ST_buffer LAKE.SW_GEOMETRY, 66.0, 1.0) 
  from LAKE 
',
```

The edited buffer of a particular lake would then replace the original lake geometry. This behavior is different from SpatialWare implementations on other RDBMSs where buffer results cannot be edited and saved. This difference is noted here for clarity.

Coordinate Systems

The coordinate system of a table is defined by the entry in the MAPINFO_MAPCATALOG. Data loaded using EasyLoader or via MI Professional’s Save Copy As function will create this entry automatically, otherwise use the Make Table Mappable function in MapInfo Professional to register MapInfo tables in the catalog table.

Spatial queries take spatial data as arguments in matching coordinate systems. For example to do a buffer-overlap, the data in both tables must be in the same coordinate system.
Data extracted into MapInfo Professional is always transformed automatically by MapInfo Professional into the coordinate system of the Map window into which it is selected. The coordinate system of an MapInfo Professional Map window is the coordinate system of the first table extracted into it.

Results from queries are returned in the same coordinate system as the data used. If you need a result to be returned in another coordinate system, then use the HG_CSTransform function to transform the spatial data to another coordinate system.
Sample Data

Overview

Sample data is provided to allow you to explore and experiment with a spatial data set. Georgetown and World databases and scripts are automatically installed into your SQL Server directory (in the Data subdirectory where SQL Server is installed). Georgetown is also referred to by the examples in the online SpatialWare product documentation.

Georgetown Database Description

This is a database of artificial data, as follows:

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>flood100</td>
<td>Flood-prone regions</td>
</tr>
<tr>
<td>lake</td>
<td>Lakes</td>
</tr>
<tr>
<td>parcel</td>
<td>Land parcels</td>
</tr>
<tr>
<td>privbldg</td>
<td>Private buildings</td>
</tr>
<tr>
<td>pubbldg</td>
<td>Public buildings</td>
</tr>
<tr>
<td>rdpaved</td>
<td>Paved roads</td>
</tr>
<tr>
<td>spotelev</td>
<td>Elevation points</td>
</tr>
</tbody>
</table>

World Database Description

Description of the World database.

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>worldcap</td>
<td>World capitals</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>world</td>
<td>Generalized world boundaries</td>
<td>MapInfo Corporation from National Imagery and Mapping Agency (NIMA)</td>
</tr>
<tr>
<td>ocean</td>
<td>Ocean</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>grid15</td>
<td>Grid of latitude and longitude at 15 degree increments</td>
<td>MapInfo Corporation</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
<td>Source</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------</td>
<td>------------------------------------------------</td>
</tr>
<tr>
<td>gazmaj</td>
<td>Major cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>gazmin</td>
<td>Other cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
</tbody>
</table>