



Enterprise Mapping Deployments

Managing Spatial Data in a Relational
Database Management System

A M A P I N F O W H I T E P A P E R

knowingwhere
is just the beginning™

Contents

EXECUTIVE OVERVIEW	3
EVOLVING TECHNOLOGIES IN DATABASE ENVIRONMENTS.....	3
MEETING THE DEMAND FOR SPATIAL DATA.....	4
MAPINFO SPATIALWARE®	6
SPATIALWARE COMPONENTS	6
SPATIAL DATA TYPE	7
Point Objects	7
Line Objects	7
Area Objects.....	7
SPATIAL INDEXING.....	8
SPATIAL OPERATORS	10
Constructor Functions.....	11
General Functions	12
Measurement Functions.....	13
Observer Functions	13
Spatial Functions.....	14
Spatial Predicates.....	16
SQL SPATIAL QUERY EXAMPLES.....	17
COMPLEMENTARY MAPINFO PRODUCTS	20
MAPINFO PROFESSIONAL®	21
MAPINFO MAPX®	21
MAPINFO® MAPXTREME®	21
MAPINFO®	22
MAPMARKER® PLUS	22
MAPINFO® MAPXTEND™	22
MAPINFO® ROUTING J SERVER.....	22
DATA PRODUCTS	22
Streets and Boundaries.....	22
Demographics.....	22
Telecommunications.....	23
CONCLUSION	23

Executive Overview

Relational database management systems (RDBMS) have become the all-important, all-encompassing data repositories for corporate information assets. They contain a wealth of information about employees, customers, inventories, financial transactions, and a broad array of business information. In addition, the RDBMS has become the central information repository that today's applications are built upon.

Within the past few years, spatial information has become significantly important to businesses. Knowing "where" provides mission critical information to aid in business decision-making, and the need to widely deploy location-based solutions throughout an enterprise has grown rapidly. With this need has grown the demand to manage spatial data along with other data in the corporate RDBMS, where it can be easily accessed by any application.


Once relegated to the back office and proprietary systems due to complexity and cost, new technologies have enabled spatial information to come out of the closet and be integrated into mainstream RDBMSs, where it can serve an enterprise business need cost-effectively. MapInfo has contributed to this capability by offering our customers industry-leading technologies for storing, managing, accessing, and updating spatial data in the RDBMS.

MapInfo has more than 15 years development expertise and experience with spatial technology. We have a robust knowledge of customer requirements and a broad product portfolio of location-based technologies and spatial data that can make widely deployed mapping applications a successful reality for any business or organization.

Evolving Technologies in Database Environments

For a moment consider why database management systems are used to store corporate data. Of primary importance is the overall "environment" provided by these systems to organize, administer, protect, and distribute data as a corporate resource. The following is not a comprehensive list but provides an indication of the complexity of the tasks relegated to the RDBMS:

- Support for multiple users
- Support for various types of users
- Restrict unauthorized access
- Data sharing
- Data abstraction
- Multiple data views
- Meta data
- Concurrency
- Transaction processing
- Control information redundancy
- Enforce data integrity constraints
- Multiple system interfaces
- Backup and recovery



Prior to database management systems, these facilities were either absent or provided at the individual application level. The “file system” was the primary data storage facility available. This system provided only rudimentary capabilities to manage the file, but not the data within it (with the exception of some specialized operating systems such as PICK).

Early development of database systems focused on the ability to deliver “traditional” forms of data to the organization. This in itself proved to be a challenge. Applications and disciplines with requirements outside the typical data structures were basically ignored by mainstream database systems. As a result, Computer Aided Design (CAD), Geographic Information Systems (GIS), and many other scientific modeling systems were left to their own devices to solve data management issues. Many of these specialized systems continue to use traditional “file management” approaches even to this day.


New computing paradigms such as artificial intelligence and object programming were developed in the 1980s and 1990s. Object programming provides many advantages over traditional approaches and has become the dominant programming paradigm. New technologies such as object programming are the necessary stuff to prevent the collision between the demand for information and the ability for systems to provide it.

The Internet has provoked an almost alarming increase in demand for data, analysis, presentation, and representation, of which maps/spatial data is one component. As a result we have seen the introduction of a variety of technology solutions — for example, HTML, XML, ASP, JAVA, and COM — that help meet the demand.

The majority of database management projects have focused on moving legacy data systems into current data warehouse and data mart paradigms. The primary goal of these projects is to increase the accessibility and value of the data for business decision-makers. This in turn has resulted in new techniques for investigating and visualizing data, such as data cubes. This has also increased demand to support non-traditional data types such as multimedia. Within this classification resides spatial data, and its use in turn has resulted in new ways of looking at and analyzing data.

Meeting the Demand for Spatial Data

The value of WHERE has always been intuitive, but is now playing a major role in providing mission critical information about clients, products, physical assets, and other information that enhances the corporate bottom line. At first, the Internet seemed to deny the importance of where, but in truth it has greatly increased the importance of knowing where. As a consumer I may find the product on the Internet, but most often will still want it *now*. Knowing the closest outlet for the product can be critical to my buying decision.



From the supplier side I want to know:

- Where are my assets?
- Where are my markets?
- Where are my clients?
- Not only who is my next best client, but where are they?
- How far and how long will it take to ship the goods?
- What's the geographic relationship between clients, assets, products, etc.?

A communications company would certainly want answers to these questions. They might be particularly interested in finding all their customers contained within a particular service area so they can target market a new product or service available in that area.


One might say, "I don't really need spatial awareness, I can just list the service area as an attribute of the client record." This is true. But consider what happens if the service area boundary changes. You will be required to determine which customers remain in that service area and which ones are changed to a new service area. You will have to visit each customer record, then determine the new service area, and you will probably do so by using a map of the service areas. With the spatially-enabled system you would simply change the service area boundaries, perform a new query with the "contain" or "overlap" function, and let the system do the work for you.

In the end, everyone who needs consistent and accurate information can have it. Each discipline (experts) can maintain the data they are responsible for while allowing other departments access to the information. For example, Marketing can determine the impact or opportunities of the new service areas, and Engineering can determine how many customers they have affected.

Now that the demand exists for spatial data, the database companies and others, such as MapInfo, are filling the technology void. Some database companies are adding limited spatial capabilities to the core of their database environments. MapInfo enters the problem space with a different perspective than the database developers, offering a solution called MapInfo SpatialWare® that extends RDBMS capabilities. The primary reason for MapInfo's approach is its history. MapInfo has been providing solutions within the spatial domain for more than 15 years. The knowledge base behind MapInfo technology contains several characteristics missing from the majority of database vendors. They are:

- Expertise, experience, and heritage in spatial technology
- Robust understanding of customer requirements
- Experience with the development and deployment of spatial technology; we are into our third generation of the technology

In addition to *SpatialWare*, MapInfo provides a broad range of products designed to meet a variety of end user requirements in performing spatial analysis and mapping tasks, and conforming to corporate computing environments. MapInfo solutions can be deployed over the Internet or an Intranet using a three-tier architecture, in a two-tier client/server architecture, or in standalone desktop environments.



Applications can serve many departments, such as Marketing, Sales, Finance, Customer Support, Billing, and Engineering. With the spatial data stored in the database along with the other data components, each department/application has access to the same data source. All the information is consistent, up-to-date, and available throughout the enterprise.

MapInfo SpatialWare®

SpatialWare extends an Informix®, Microsoft® SQL Server, or IBM® DB2 database to handle spatial data — points, lines, and areas (polygons) — just as it handles the traditional non-spatial data types like character and integers. *SpatialWare* uses a standards-based approach conforming to the ISO Multimedia/Spatial standards for handling spatial data. MapInfo actively participates in the spatial standards process, dedicating resources to both the Open GIS Consortium, and the International Standards Organization (ISO) committees that deal with spatial data.

SpatialWare is implemented on these databases in the following ways:

- On SQL Server through the Extended Stored Procedure mechanism
- On IBM DB2 as an Extender
- On Informix as a DataBlade

SpatialWare extends database capabilities without using a middleware architecture. All functionality is contained within the RDBMS environment. This methodology creates a tightly integrated solution with the database allowing the user access to spatial extensions within the normal database environment and its tools. Thus users and administrators can perform the majority of database tasks from within the normal tools provided by the database vendor.

SpatialWare Components

To spatially enable any database requires the provision of three component parts:

1. Spatial Data Type defining the data structure and storage mechanism
2. Spatial Indexing providing custom index structure to handle spatial data
3. Spatial Operators extending the SQL interface to the data

Each of these components will be discussed in the sections below.

Spatial Data Type

Traditional data types such as character, integer, float, and date do not provide the necessary data structure to properly represent spatial information. Spatial data requires a more complex data type to represent two dimensional objects that may be composed of a variety of simple or complex geometric primitives. The Spatial Data Type, `ST_SPATIAL`, provides for the efficient storage of spatial data within the database. This data type can manage a variety of geometry types: points, lines, and areas (surfaces) (See Figure 1). *SpatialWare* provides the necessary data structures to store this geometry information in the database.

Three basic geometry types — Point, Line, and Area (also referred to as polygon or surface) — provide the basic classifications for identifying spatial objects. Each classification can be composed of a composite set of geographic primitives such as: point, line, polyline, curve, arc, circle, and polygon.

These basic classifications are used to represent features, or objects, in the real world. Common data models represent features such as buildings and equipment locations (towers) as point features; roads and power lines as linear features; and political areas, census zones, and lakes as area (polygon, or surface) features. The logical data models may employ different classifications to represent the same type of feature depending upon the application requirements. For example, a local government may require a house to be represented as a polygon that shows the footprint of the building as it sits on the ground. A bank or insurance company may only require that the house be represented as a point depicting the general location of the structure. It is typically the application requirements and cost of obtaining the data that dictate how the features will be modeled.

Point Objects

The representation of features such as a client location, house, fire hydrant, or tower typically use single points. In most cases the location of the point is sufficient to represent the user's concept of the feature even though a house is not really a point. Point is the simplest of the classifications; it is defined as a single set of x/y coordinates, and optionally Z.

Line Objects

The representation of features such as roads, rivers, and power lines are linear constructs and can be composed of several types of primitive geometry that include simple two point lines, multi-point lines (polyline), curves, and arcs. A linear feature can be represented by a grouping of these elements such as a simple two-point line, followed by an arc, followed by a polyline. The line primitives are themselves made up of collections of points.

Area Objects

Area (polygon or surface) features are represented by a bounding area such as a lake, flood zone, statistical area, state, or county. They can be a complex collection of closed polygons that can also contain islands within polygons. Area features are themselves defined by linear components, which are ultimately defined by points. A

key distinction of an area is that the lines defining the boundary of the area do not cross themselves and are closed (the last point is also the first point).

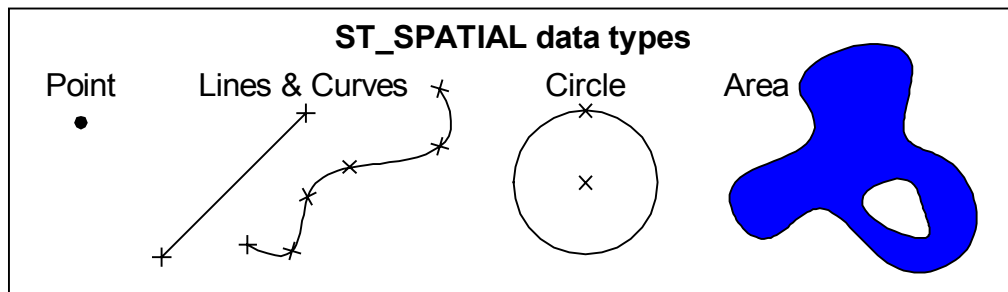


Figure 1: Spatial Data Types

Spatial Indexing

Spatial indexing is required to ensure the performance of spatial queries to the database. Otherwise, spatial queries would be prohibitively slow. *SpatialWare* implements Range Tree (R-Tree) indexing and has been an industry leader in the use of this technique. Without an index, the query (database) engine must sequentially evaluate the predicate for every record in a table, requiring excessive I/O. If the table is small — only a few hundred records — the absence of an index may not be an issue. As the table size increases, the need for indexes also increases. To sequentially search through an unordered table with millions of records could take an unreasonable amount of time, rendering the query useless. If an index exists over the attribute, the engine may use it to minimize the number of I/Os necessary to satisfy the predicate.

Unlike indexes for standard character and integer data types, a spatial index must accommodate two-dimensional objects. However, the principal is similar: the objective is to find the desired records (those that meet the qualification) while inspecting the fewest number of records possible (least number of I/Os).

The R-Tree index organizes records by the geographic extent of each object. Objects can be grouped together and described as being contained within a larger rectangular space. The size of each space is dependent upon the number of objects in that space. As the number of objects increases, the space can be split into two areas to reduce the number of objects in each new area. The process continues so the entire geographic extent of the data is covered, and the number of objects related to each area is approximately equal. The R-Tree index is a height balanced tree similar to a B-Tree, making it both highly concurrent and recoverable, which are necessary attributes for modern database systems.

When a spatial qualifier is used in a query it performs a two-pass process. First, the index can "traverse" to that part of the index that exhibits some spatial relationship (common area). The first pass compares the overall physical extent of an object with that of the qualifier. If there is no intersection between the object envelopes,

the record can be ignored. If the envelopes intersect, then the second pass is invoked that compares the actual detail of each geometry to determine if the record meets the qualification.

The example below is a simple illustration showing how the R-Tree index is organized (see Figure 2). The extent of the data can be variable; in this example it covers a single state, New Mexico. The points represent some other set of data, say client locations, within the database. The light colored shape at the bottom left might represent a desired search area.

The query objective is to find all points inside the target search area. The spatial extent of the search area can be used to traverse the index. The extent of the target area can skip the majority of the database and focus on the portion of the R-Tree Node 1, where there is a common spatial extent. Only the records in Node 1 need to be inspected further to determine if the node geometry falls inside or outside the target polygon. In this example, four records need to be tested further. The point at the lower left is quickly eliminated, as the envelope of the point does not intersect the envelope of the target polygon, a very fast test to perform. The other points pass the first test, since the point envelope and the polygon envelope overlap. These points then go through a second pass to determine if the point geometry is contained within the bounds of the target polygon.

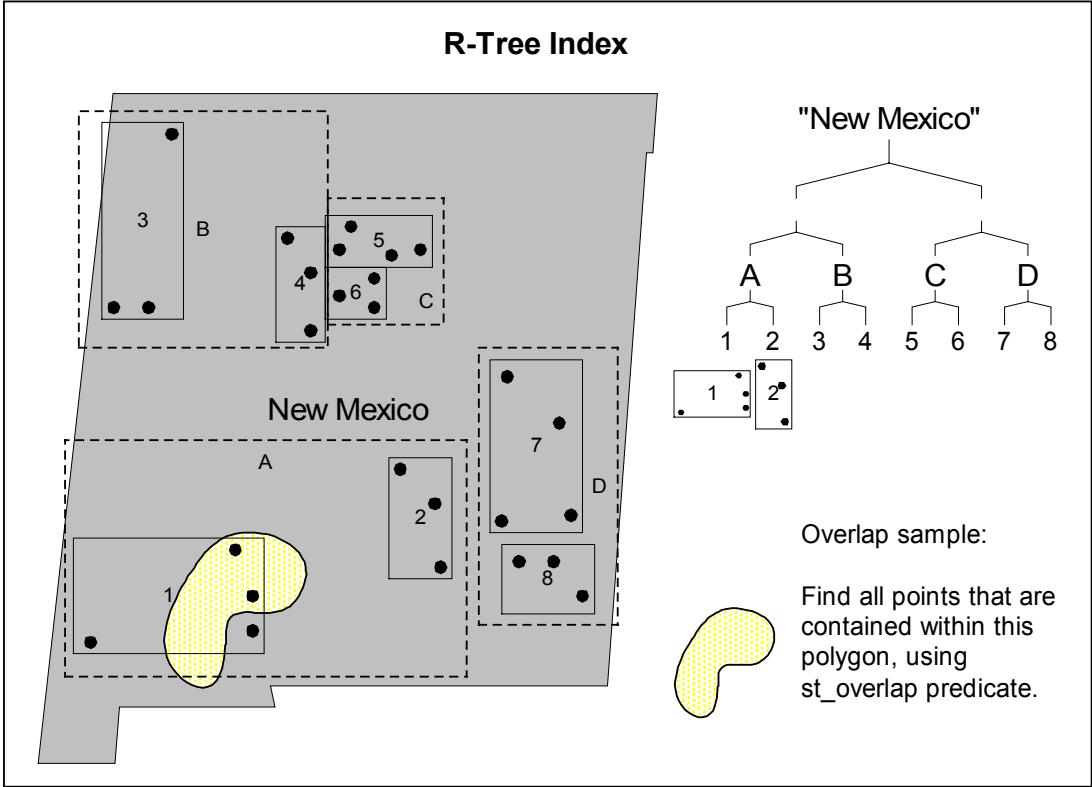


Figure 2: Organization of an R-Tree Index

Spatial Operators

SQL is the language used to query information from tables in the database; it provides the interface between a requestor and the database system. The "requestor" can submit a query from an SQL interpreter, or an application using an ODBC or JDBC connection. The requestor forms a query statement used by the database engine to query records from tables and return the specified information to the requestor.

Standard operators perform operations such as add, subtract, sum, string operations, and time between dates. To make use of the spatial data it is essential to provide extensions to SQL that allow operations to be performed on the spatial data type. Common spatial questions include:

- Which and how many customers live inside a service area?
- Which customers are within 1 mile of my branch office?
- What utility services are available at my location?

Just like you might qualify a query statement to search for clients with an age greater than thirty, spatial functions allow you to ask questions that evaluate the geometry data such as asking for all clients that are inside a particular area, such as flood zones.

Find all clients who are over thirty years of age:

```
SELECT client.name
FROM client
WHERE age > 30;
```

Find all clients who live in 100-year flood zones:

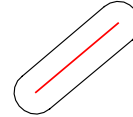
```
SELECT client.name
FROM client, floodzone
WHERE ST_OVERLAPS( client.sw_geometry, floodzone.sw_geometry)
AND floodzone.type = '100';
```

The functions are designed to be "object" oriented making it unnecessary for the user to know about the underlying structure of the geometry. For example overlap, buffers, and other functions operate on types: point, line, and polygon. The user does not need to know the type of geometry prior to using the operator. The following queries work equally well to create a 500-unit buffer around objects in tables with different types of geometry.

```
SELECT ST_BUFFER(client.sw_geometry, 500)
FROM client
WHERE client.age > 30;
```



```
SELECT ST_BUFFER(road.sw_geometry, 500)
FROM road
WHERE road.name = 'MAIN ST';
```



```
SELECT ST_BUFFER(lake.sw_geometry, 500)
FROM lake
WHERE lake.name = 'BLUE LAKE';
```



In these examples, the user doesn't know or specify that clients are points and streets are lines. The operator understands the geometry type and knows how to place a buffer around Points, Lines, or Polygons.

SpatialWare provides over 150 spatial operators. The operators can be broadly classified into the following categories:

- Constructor
- General
- Measurement
- Observer
- Spatial
- Spatial Predicate

The following will present a sampling of the functions available with *SpatialWare*. Functions with an "ST_" prefix are functions defined by the ISO standard, functions with the "HG_" prefix are extensions to the standard provided by MapInfo. The functions that are extensions conform to the same syntax rules as the standard functions.

Constructor Functions

Constructor functions are used to create geometry objects. In the majority of cases the user will won't have to use constructor functions. Utilities are provided to load data with geometry and perform this operation for the user. There are times, however, when the user may want, or need, to specify geometry manually. In most cases this technique will be used to specify simple features such as points.

For example, an application may obtain or determine the location from another source, or from existing values in a database. If the user knows the location value (Longitude, Latitude) it can be specified in the query statement using the appropriate constructor. These functions are most often used within insert and update statements and may also be used within a predicate phrase.

The following specifies a "point" location
`ST_SPATIAL (ST_POINT (-109.342156, 42.876123))`

`ST_SPATIAL` identifies that this is a spatial object and is optional. This constructor works equally well:
`ST_POINT (-109.342156, 42.876123)`

This construct can be used to define a geometry anywhere an `st_spatial` data type is expected. For example to insert a geometry into a table the following SQL statement would be performed:

```
INSERT INTO client(sw_member, name, sw_geometry)
values(3, 'Albert Newman', 'ST_Spatial(ST_Point(-111.11,44.44))')
```

General Functions

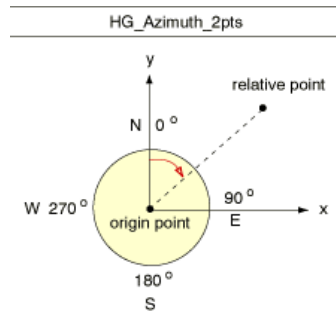
This category is for general functions and those specific to a database. The general functions provide the ability to perform conversions and define constants. Database specific functions in this category are typically for establishing parameter settings and/or performing conversion (casting) operations to convert the internal geometry data structure of the database into a client application format or human readable format.

<code>HG_GetString</code>	Converts the binary data from DB2 or SQL Server into human readable form.
<code>HG_Morph_Out</code>	Converts the internal binary into a MapInfo client (TAB) format.
<code>HG_Pi</code>	Returns the constant Pi value of the system.
<code>HG_Radians</code>	Returns the radian equivalent of a degree value.
<code>HG_Degrees</code>	Returns the degree equivalent of a radian value.

Measurement Functions

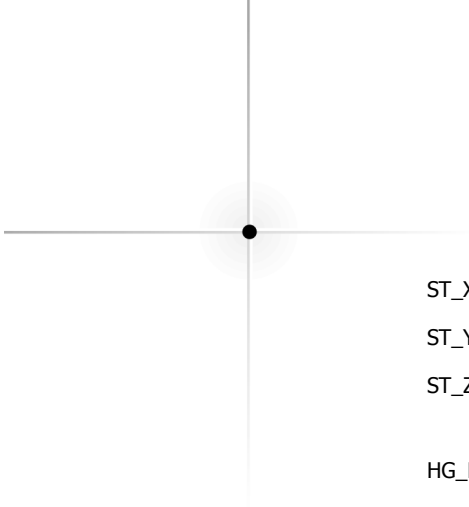
Spatial measurement functions perform calculations on geometry's to find a measurable characteristic, such as length, slope, area, or height. *SpatialWare* provides measurement functions defined by the ISO standard as well as functions that extend the standard. These functions are used within the select phrase of a query statement.

ST_AREA	Computes the area of an area (polygon) geometry. If the area contains islands that area is excluded.
ST_LENGTH	Computes the 2-dimensional length of a linear geometry
ST_PERIMETER	Computes the 2-dimensional perimeter of a polygon
ST_LENGTH_3D	Computes the 3-dimensional length of a linear geometry. The object must contain Z coordinate values.
ST_PERIMETER_3D	Computes the 3-dimensional perimeter of a polygon. The object must contain Z coordinate values.
HG_DISTANCE	Computes the distance between any two points
HG_SLOPE	Computes the Slope of a linear object. The object must have Z coordinate values.
HG_AZIMUTH	Computes the Azimuth, or compass direction, between the beginning and ending point of a linear feature.
HG_AZIMUTH_2PTS	Computes the Azimuth between any two points.



Observer Functions

Spatial Observer functions return numbers, objects, or conditions (attributes) of a geometry. The primary purpose of these functions is to interrogate a geometry object to disclose information about its internal structure. Some functions return a float or integer value, such as the number of points in a geometry or the X, Y, Z values. Others return ST_SPATIAL geometry that can be used within other functions. These functions are most often used within the select portion of a query statement, but may also be used within a predicate. Many of the functions provided are extensions to the ISO standard.

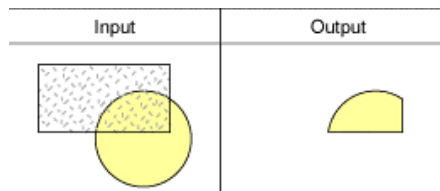


ST_X	Returns the X coordinate value of a point.
ST_Y	Returns the Y coordinate value of a point.
ST_Z	Returns the Z coordinate value of point, or null if a Z value does not exist.
HG_NCOORDS	Returns the total number of points in an ST_Spatial, counting points embedded in other geometry constructors (ST_GeometricPrimitives).
HG_BEGIN_POINT	Returns the beginning point of a linear feature
HG_END_POINT	Returns the ending point of a linear feature

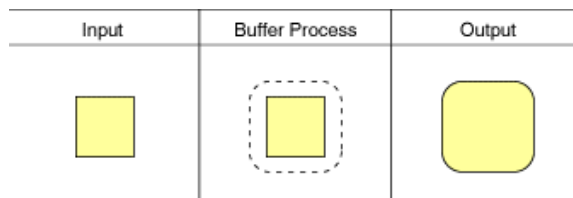
Spatial Functions

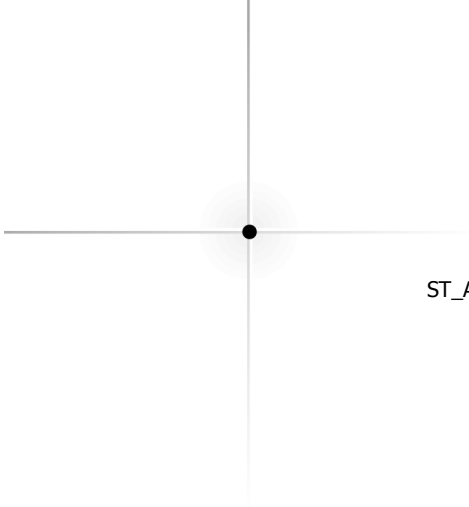
Spatial functions perform operations on geometries for the purpose of creating a new geometry, such as the earlier examples that created buffers. These functions may be used within the predicate or select phrase. Many are extensions to the ISO standard.

ST_OVERLAP	Returns all of the overlapping elements between two spatial objects. Two elements are overlapping when they share common points.
------------	--



ST_BUFFER	Takes as inputs a geometry, buffer distance, and filter tolerance. The filter parameter specifies a filter tolerance on the buffer creation process. Returns a new "expanded" geometry.
-----------	---





ST_ADJACENT

Returns an ST_SPATIAL made up of the points of intersection and common line segments between two spatial objects.

Input	Adjacent Process	Output

HG_CENTROID

Returns the centroid of a given geometry. The centroid is a point representing the weighted center of a shape, sometimes referred to as the "visual center" of the feature.

Input	Output
 object	 Point representing the weighted center.

HG_CONVEX_HULL

Returns the minimum boundary around a spatial object without the boundary being concave. The result is the spatial object representing the convex hull of the perimeter.

Input	Convex Hull Operation	Output
 Object containing multiple elements.		 convex hull

HG_LL_CIRCLE

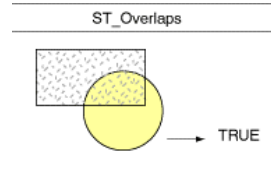
Accepts the Longitude/Latitude of a center point and a radial distance in meters, and creates a geometry representing a search area which is a good approximation of a circle in Longitude/Latitude.

Spatial Predicates

Spatial Predicates analyze geometry for specific spatial relationships. These functions return true (1) or false (0) values and are used within a WHERE clause. In addition to the ISO standard functions, there are a number of functions that extend the standard.

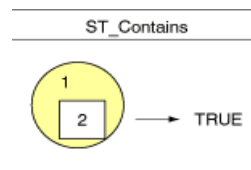
ST_OVERLAPS

Returns TRUE if elements of two geometry's overlap. Two elements are overlapping when they share common points.



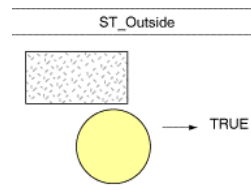
ST_CONTAINS

Returns TRUE if geometry1 entirely contains geometry2, and FALSE if it does not. Boundaries can touch, but the inner object cannot have any points outside the containing boundary.



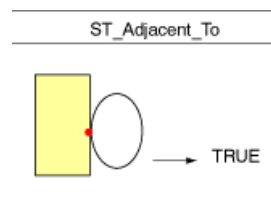
ST_OUTSIDE

Returns TRUE only if there are no common points between the two objects.



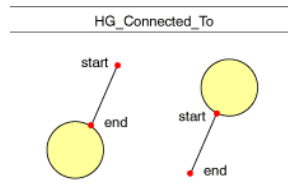
ST_ADJACENT_TO

Returns TRUE if two objects touch. They touch if they have one or more common boundary points, but no common interior points.



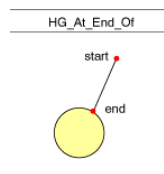
HG_CONNECTED_TO

Returns TRUE if the end point or start point of a line matches a point in the object.



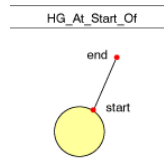
HG_AT_END_OF

Returns TRUE if the end point of the line matches a point in the ST_Spatial's boundary.



HG_AT_START_OF

Returns TRUE if the start point of a line matches a point in the boundary of a object.



SQL Spatial Query Examples

The following sample queries illustrate how some of the spatial functions are actually used to solve problems. As with any SQL statement, the user should consider how the query is formed as some syntax can be optimized to be more efficient than others. In addition, it is sometimes valuable to "dissect" a problem and perform multiple query statements storing intermediate results in temporary tables.

- 1) A real estate application may want to provide information about available services such as electric, gas, and phone service. We will presume there is a table that contains the service areas of all utility companies. The Application may use a geocoding engine, such as MapInfo® MapMarker®, to determine the Longitude and Latitude of the person's address. The application would submit the following query to get a list of utility company names and other vital information for the desired location.

```

SELECT utility_co.name, utility_co.phone
FROM utility_co
WHERE ST_OVERLAPS ( ST_POINT (-109.342156, 42.876123),
utility_co.sw_geometry );

```

A wireless phone company could use a similar query to determine if a user is within their coverage area.

- 2) Using the above example, if the utility table also contained the type of utility they could specify a particular kind of utility: "Find the name of the electric utility."

```

SELECT utility_co.name, utility_co.phone
FROM utility_co
WHERE ST_OVERLAPS ( ST_POINT (-109.342156, 42.876123),
utility_co.sw_geometry ) AND
utility_co.type = 'ELECTRIC';

```

- 3) A bank, retail store, or other facility might want to know how many or who their clients are that live within walking distance (500 meters) of a particular facility. Note that this query uses a nested buffer function inside the overlap function. The result of the buffer, a new geometry, becomes the second input geometry of the overlap function. In effect we put a buffer around store with id "11" and find all of the client points that fall inside the buffer.

```

SELECT client.name
FROM client, store
WHERE ST_OVERLAPS
(client.sw_geometry, ST_BUFFER( store.sw_geometry, 500.0 )) AND
Store.id = 11;

```

- 4) Using MapInfo client tools such as MapInfo Professional®, MapInfo MapX®, or MapInfo® MapXtreme®, map data can be displayed to the clients. The following map illustrates the results of an investigation to determine property values possibly affected by a 100 year flood event (see Figure 3). The following queries are used to generate the results shown on the map:

Select all parcels that OVERLAP with the 100-year flood zone, and calculate the area of the overlap.

```

SELECT a.sw_member,
ST_Area( ST_Overlap( b.sw_geometry,a.sw_geometry ) ) as area,
HG_morph_out( a.sw_geometry )
FROM parcel a, flood100 b
WHERE ST_Overlaps( a.sw_geometry, b.sw_geometry )

```

Select buildings contained WITHIN the 100 flood zone, with the building identifier and flood zone identifier attributes:

```
SELECT a.sw_member, b.sw_member as flood_cm,  
       HG_morph_out( a.sw_geometry )  
FROM privbldg a, flood100 b  
WHERE ST_within( a.sw_geometry, b.sw_geometry )
```

Select Buildings touching only the EDGE of the flood zone:

```
SELECT a.sw_member, b.sw_member as flood_cm,  
       HG_morph_out( a.sw_geometry )  
FROM privbldg a, flood100 b  
WHERE ST_overlaps( a.sw_geometry, HG_as_paths( b.sw_geometry ) )
```

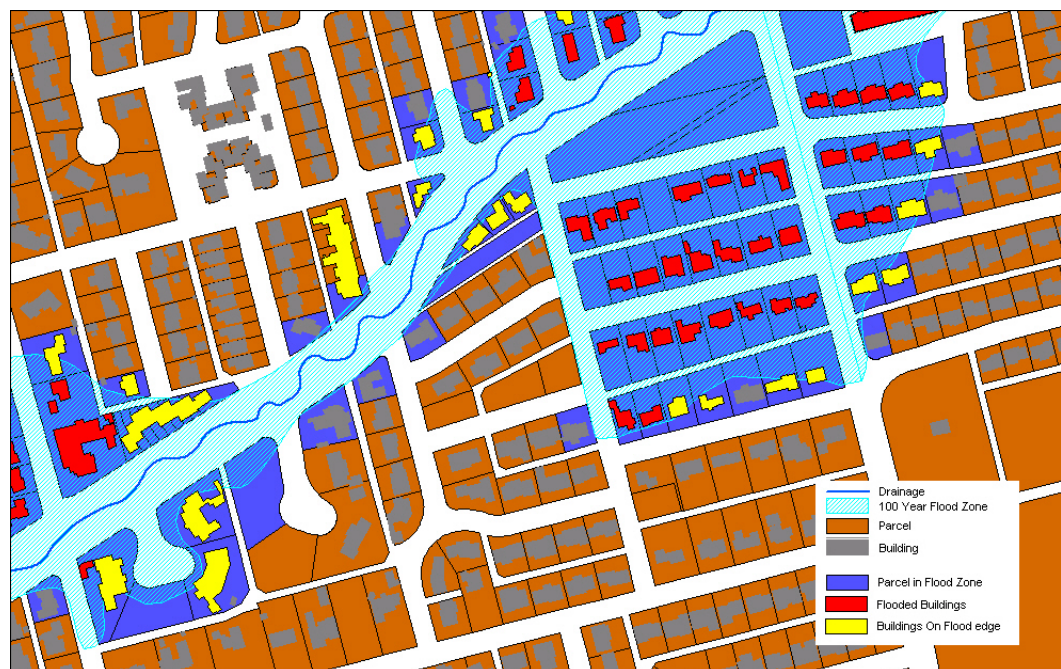


Figure 3: Flood Map

Complementary MapInfo Products

SpatialWare provides one component of the total spatial enabling solution. *SpatialWare* enables the database environment as discussed in this document. It can be used by itself to return the solutions to spatial questions in textual format for “traditional” types of applications. In some cases this is suitable; for example, an application may only need to know the names and contact information of clients that are near a facility, without the presentation of a map. In this case, a list of names and phone numbers may be sufficient to complete the task. In many cases however, it is the presentation of the resultant information on a map that not only provides the result details but also geographically illustrates the conditions.

Visualization, further analytical capabilities, and deployment within different environments is provided by additional offerings in the MapInfo product line (see Figure 4). The integration of MapInfo technologies and data allows organizations to capitalize on spatial information systems. These products are described below.

MapInfo Product Overview

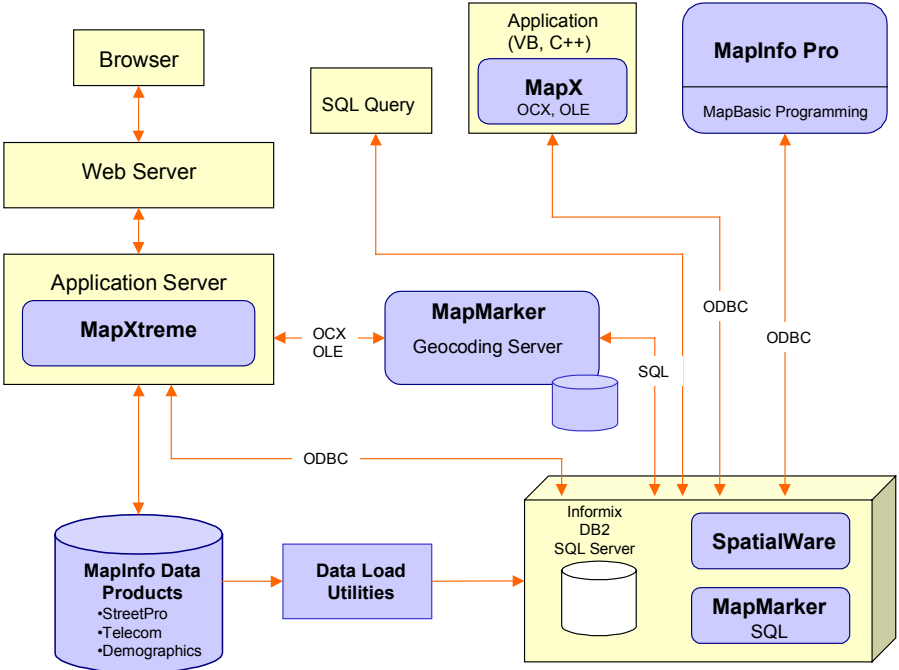


Figure 4: MapInfo product line in an enterprise deployment

MapInfo Professional®

MapInfo Professional sets the standard for desktop mapping, visualization, and geographic analysis. Direct remote database read/write capabilities allow you to access your corporate data through ODBC connections, and keep it current. MapInfo users throughout the organization can connect to a central *SpatialWare* database to manage and share information between different departments. Everyone can have access to the vital spatial information used in decision making.

- Built-in conflict management helps you manage discrepancies in data when writing to the server.
- Create new map objects from polygon intersections, merges or splits, and perform data calculations on the new areas.
- Perform detailed geographic searches with area selection tools and buffer zones. Build SQL queries that access and integrate data from multiple tables.
- Integrate geographic criteria into database queries (contains, intersects, within, etc.).
- Perform thematic mapping, buffers, districting, geocoding and other spatial operations.

MapInfo MapX®

MapInfo MapX is the premier ActiveX component for mapping developers to quickly integrate mapping into client side business applications using standard visual programming tools such as Visual Basic. *MapInfo MapX* offers true Object Linking and Embedding (OLE) control, allowing you to integrate mapping into new and pre-existing business applications. Organizations can realize significant benefits through an enhanced understanding of tabular data, which can result in higher valued quality analysis and increased productivity to improve a company's bottom line.

- Powerful, layered object model supports many objects, events, and hundreds of methods and properties.
- Spatial server access allows developers to query and edit live data stored in spatial servers like the Oracle8i® Spatial server and *SpatialWare* running on Informix®, Microsoft® SQL Server, and IBM® DB2 databases.

MapInfo® MapXtreme®

MapXtreme is a tool set for creating a web-based mapping application that can be deployed on an organization's Intranet or over the Internet. In a *MapXtreme* implementation, the application and data reside on a centralized server so performing maintenance and upgrades on the application become much easier. End users interact with it via any standard web browser.

- Mapping functions include thematic mapping, buffering, object (map) editing, draw layer, find, map display, layer control, spatial selections, geocoding, and extensive database binding.
- Spatial server access allows developers to query and edit live data stored in spatial servers like the Oracle8i Spatial server and *SpatialWare* running on Informix, SQL Server, and IBM DB2 databases.

MapInfo® MapMarker® Plus

MapMarker Plus is a geocoding engine that turns ordinary data records containing address information into geographic objects that can be displayed on a map. This helps users to better see relationships in their data. *MapMarker Plus* can geocode files of millions of records with street-level precision in one-pass batch mode processing for the entire United States. It is also available for other countries.

- Incorporates sophisticated matching algorithms with enhanced street data from GDT, Inc., to yield the highest geocoding rates and ensure that analysis incorporates the best available data.
- Available for a variety of deployment strategies: single user, NT and UNIX server, Java, and database (Oracle, Informix, and SQL Server) implementations.

MapInfo® MapXtend™

MapXtend is a developer tool for creating location-based applications running on wireless handheld devices. *MapXtend* seamlessly integrates with other MapInfo server and client technologies. Applications created with *MapXtend* are designed to give mobile field staff live access to the most updated corporate data on equipment and customers, helping increase efficiency and improve service.

MapInfo® Routing J Server

MapInfo Routing J Server is an engine for creating web-based applications for routing people, products, and resources. It lets you add turn-by-turn directions to calculate either the shortest distance or quickest route between any two points.

Data Products

All MapInfo software applications make use of external data sets for enhanced analysis and business intelligence. MapInfo offers comprehensive, accurate, and up-to-date worldwide data products including boundaries, demographics, streets, and industry specific information. All are optimized for use in MapInfo software and solutions, for both individual analysts and broad deployments.

Streets and Boundaries

MapInfo's high-quality, regularly-updated street and boundary maps are available for key markets around the world. These maps are designed for use in MapInfo applications for routing, drive-time studies, background information, and analysis and visualization. Boundary maps are available for postal, political and industry-specific areas.

Demographics

MapInfo has a wide selection of worldwide demographic data products containing information such as population, income, expenditure, retail activity, employment, consumer trends, business summary data, and lifestyle segmentation data.



Telecommunications

MapInfo offers a comprehensive portfolio of telecommunications industry data to support infrastructure, wireless, and convergence applications. Products include industry-specific boundaries, infrastructure information, and statistics, as well as data processing and analysis engines.

Conclusion

In today's competitive business environment, the notion of "where" has become increasingly important to decision making. This has created a situation where spatial information has become a "must have" for organizations to be more competitive and make more informed decisions. Additional emphasis has been placed on storage and analysis of spatial data within the core database environment, allowing the integration of traditional data analysis with spatial analysis.

SpatialWare allows users to store, query, and analyze spatial data in the same way as other data, providing all the benefits and power an RDBMS has to offer. For a minimal investment, businesses can unleash the full power of their database: find where customers and potential customers are, grow a customer base, and serve existing customers better.

SpatialWare and a spatially-enabled RDBMS are at the core of a spatially-enabled IT backbone. Combined with MapInfo's broad array of leading products and technology, *SpatialWare* helps maximize the value of data. MapInfo offers complete solutions for any technology environment. Whether you need to support ten or ten thousand users, *SpatialWare* deployed in an enterprise environment can help you to serve them all better.



CORPORATE/AMERICAS

HEADQUARTERS

One Global View
Troy, NY
12180-8399 USA
518.285.6000 TEL
1.800.327.8627
518.285.6060 FAX
sales@mapinfo.com E-MAIL
www.mapinfo.com

EUROPEAN/UNITED KINGDOM

HEADQUARTERS

44.1753.848200 TEL
44.1753.621140 FAX
europe@mapinfo.com E-MAIL

ASIA-PACIFIC/AUSTRALIA

HEADQUARTERS

61.2.9437.6255 TEL
61.2.9439.1773 FAX
australia@mapinfo.com E-MAIL



© 2001 MapInfo Corporation. All rights reserved. MapInfo, MapInfo Professional, MapMarker, MapXtreme, MapInfo MapX, MapXtend, and SpatialWare are trademarks of MapInfo Corporation. All other marks are the property of their respective holders.