IBM's DB2 Spatial Extender: Managing Geo-Spatial Information Within The DBMS

Prepared for IBM Corporation
by Judith R. Davis
May, 1998

TABLE OF CONTENTS

EXECUTIVE SUMMARY

IBM is focused on extensible data management

IBM has focused research efforts on creating an extensible data-management infrastructure for more than a decade. The results of these efforts have become a major component of IBM's DB2 Universal Database (UDB), the company's flagship object-relational database management system (ORDBMS), and DB2 DataJoiner, IBM's ORDBMS for heterogeneous data access. A major goal of an ORDBMS is the ability to model complex data and objects—geo-spatial data, text, images, and other user-defined data types—directly in the DBMS. This gives users four key benefits:

- Enhances the business value of existing applications and data
- Improves business intelligence with integrated searching across all data types
- Facilitates the development of new applications and queries
- Improves overall application performance

DB2 Spatial Extender integrates GIS functionality into the DB2 ORDBMS

Traditionally, geo-spatial data have been managed by specialized geographic information systems (GISs) that cannot integrate spatial data with other business data stored in the RDBMS and other data sources. With the addition of object extensions to the RDBMS, GIS intelligence can now be incorporated directly into the ORDBMS. IBM has collaborated with partner Environmental Systems Research Institute (ESRI), a major developer of GIS systems, to accomplish this through DB2 Spatial Extender. IBM is initially delivering Spatial Extender on DB2 DataJoiner, and plans to merge these technologies with DB2 UDB in the next release of UDB.

Spatial Extender and DB2 are a strong combination

The integration of Spatial Extender with the DB2 ORDBMS has significant strengths:

- Full integration of geo-spatial intelligence with the DB2 ORDBMS and SQL, including indexing and cost-based query optimization. DB2 will have new extensions to support Spatial Extender, such as abstract data types and user-defined index structures.

- Compliance with emerging industry standards (SQL3, SQL/MM, and OGIS) and full support for existing GIS data in three industry formats (ESRI shape format, OGIS well-known text format, and OGIS well-known binary format).

- Full support for popular GIS tools plus built-in support for sophisticated rendering of map data for business-intelligence visualization, a geocoding function for U.S. addresses, and a sample set of world map data.

Spatial Extender has appeal regardless of database platform

Another key benefit is Spatial Extender's ability to exploit DB2 DataJoiner and apply spatial intelligence to data stored in heterogeneous data sources. Spatial Extender can, therefore, address geo-spatial business-intelligence requirements across both IBM and non-IBM data sources without having to physically move any data into DB2. This is an important differentiator for IBM.

By enabling any organization to enhance its understanding of its business, leverage the value of existing data, and build sophisticated new applications, DB2 Spatial

Extender will have broad appeal to a wide variety of database customers.

OVERVIEW OF DB2 SPATIAL EXTENDER

THE BUSINESS PROBLEM

Business intelligence is driving the need for data integration

Database users want to ask increasingly complex questions about their data in an effort to uncover new and valuable business intelligence. This is a driving force behind the dramatic growth in the development of decision-support and data-warehousing applications designed to support business-intelligence goals. Here are just a few examples:

- ***Retail site selection****:* "Where should we open our new stores?" A store or restaurant chain wants to expand and is evaluating possible new locations. In addition to typical business criteria, such as lease terms and the size of available buildings, the organization also wants to consider the following for each location: the demographics of the surrounding neighborhood (do the demographics fit our targeted customer base?), the crime rate in the area (a low crime rate is important for retail operations), proximity of the site to major highways (to attract customers from outside the immediate area), proximity of major competitors (a site with little competition will most likely mean higher sales), and proximity to any known problem areas that must be avoided (a restaurant obviously doesn't want to be close to a landfill).

- ***Insurance risk assessment****:* "Is this home location within our risk parameters? What price should we charge for insuring it?" Again, standard business considerations (age of the home, construction quality, size, etc.) are not enough in assessing candidates and cost for homeowners insurance. Other factors are the crime rate in the neighborhood, proximity to local emergency services, values of comparable properties in the area, and whether or not the house is within a known flood or earthquake zone.

  Another risk-assessment example: "What types of accidents happened within 500 feet of this intersection and resulted in total claims payments of more than $10,000? How many of these accidents involved a pedestrian and a compact car?" In this case, the insurance company is assessing its automobile claims experience in relation to a specific accident location in addition to other factors.

- ***Targeted marketing campaigns****:* "In which region should we test-market this new product?" A consumer-products or mail-order company wants to test-market a new product on a limited basis before rolling it out across the entire sales area. Selecting the appropriate location for the test involves finding the best match for the target customer profile among the demographics of each region. Other factors in the test-market decision could include availability or absence of similar competitive products, the popularity or name-brand recognition of various products, and relative shipping costs. The ability to graphically display all of this information on a map will help the company visualize how the criteria overlap and to perhaps identify unexpected criteria that may be important.

Queries such as these can only be answered if all of the data are available and the DBMS knows how to interpret the data. In the second insurance example, while the DBMS may have information about the location of accidents, it may not have the intelligence to figure out which ones were within a specified distance of the intersection, or which ones involved both a pedestrian and a particular type of car. Traditional DBMSs do not know how to handle spatial data or do complex text searches. They only understand typical business data expressed as numbers, characters, dates, etc.

The applications described above illustrate the need to integrate geo-spatial data with traditional business data, to provide advanced query-analysis functions for correlating the data, and to offer end-user tools that can visually display the data in a geo-spatial context. This is the precise set of requirements IBM is targeting with delivery of its new DB2 Spatial Extender.

EXTENDING THE DATA MANAGEMENT ENVIRONMENT

Extending relational to model complex business objects

The latest development in relational database management system (RDBMS) technology is extending the database server to understand new types of data, complex business relationships, and additional application semantics. IBM's DB2 Universal Database (DB2 UDB) Version 5 is a leading example of an extended-relational, or object-relational, DBMS (ORDBMS). DB2 UDB has evolved into a database server that can better represent real-world objects through the implementation of an extensible type system; user-defined functions; an extensible, cost-based optimizer; and object-oriented modeling techniques plus many other new features and functions.

An ORDBMS is a key component of a comprehensive and fully extensible data-management architecture. Other important capabilities include transparent access to heterogeneous data, client support for objects, and an extended file management system that allows the RDBMS to manage external files and data. IBM has already delivered most of these components and will continue to enhance its support for extensibility in its DB2 family of products as described in a paper titled *Creating An Extensible, Object-Relational Data Management Environment: IBM's DB2 Universal Database*. (This and other papers covering IBM's data-management strategy in more detail, such as *DataLinks: Managing External Data with DB2 Universal Database*, are available on IBM's web site; see page 21.)

One primary goal for an ORDBMS is to intelligently manage complex objects such as geo-spatial data, images, video and audio clips, time series, and ultimately, any user-defined data types needed to meet unique business requirements. Modeling complex data and objects directly in the database gives users four benefits.

Extending existing applications adds business value

New functions enhance the business value of existing applications and data.

The queries above demonstrate that users are discovering new ways to take advantage of existing data. In the second insurance example, why not translate the addresses of the intersection and each accident—which are already in the database as character strings—into "spatial data," or locations on a map, and compare them using spatial operators to see which ones meet the criteria? The ability to do a full-

text search on the contents of a column that describes accident details enables the user to get additional business value out of the data. By adding these capabilities to the ORDBMS, they are available to an existing application without reengineering.

Better business intelligence

Integrated searching across all data types improves business intelligence.

The user wants to generate a single query that can access any data anywhere. The ORDBMS's ability to provide integrated access to data and perform complex analyses improves the organization's overall business intelligence. The organization can now use its data assets more effectively to make better business decisions.

The ability to handle a broader range of applications

Teaching the ORDBMS to understand additional types of data facilitates the development of new types of applications and queries.

In addition, users can now develop brand-new applications that were not previously possible. An example is the emerging use of advanced, precision-farming techniques. Here, a farmer uses detailed information about soil condition in various parts of a field to decide what type and amount of fertilizer to apply to each part. This information is combined with the geographic location of farm equipment as it moves across the field to actually apply the specified products in different amounts.

Improved performance

Native understanding of new types of data in the ORDBMS also enhances performance of the overall application.

Incorporating native understanding of new data types enables the ORDBMS to apply its well-known performance techniques across a broader set of application components. These techniques include cost-based query optimization to select the best access path for a given query, including the ability to combine access costs for new data types with access costs for traditional relational data in a single set of calculations to get globally-optimal access to all data; parallel query execution on multiprocessor platforms (SMP and MPP); the ability to leverage I/O performance optimization such as prefetching; and other mechanisms for enhancing performance such as new types of indexes.

DB2 Spatial Extender integrates GIS with the ORDBMS

This paper focuses on extending the ORDBMS to handle geo-spatial data. The concept of spatial data is universally understood and almost every organization can benefit from integrating spatial data into its business analysis. Until recently, spatial analysis has required the use of a specialized, proprietary geographic information system (GIS), a product designed to model and manage data in the form of geographic locations and geometry. A GIS system understands how to model geometric objects such as circles, polygons, lines, and points, and how to locate objects geographically. By introducing spatial extensions to the object-relational DBMS, DB2 Spatial Extender moves this GIS functionality into the ORDBMS, facilitating the integration of spatial data with other business data.

Spatially-enabling heterogeneous data is a key differentiator

In conjunction with DB2 DataJoiner, IBM's heterogeneous data-access solution, Spatial Extender also has the unique ability to "spatially enable" data across multiple, heterogeneous data sources. Users can now apply GIS capability to data stored in both DB2 and non-DB2 DBMSs, opening up significant opportunities to develop new applications and extend existing applications. Spatial Extender

technology is also applicable across all of the traditional applications that drove the development of GIS in the first place.

Joint effort with partner ESRI

DB2 Spatial Extender is a joint effort between IBM and a third-party partner, Environmental Systems Research Institute, Inc. (ESRI). As a market leader in GIS software, ESRI offers a broad portfolio of GIS products and tools. The company has invested 30 years of effort in research and development of spatial-data systems. ESRI and IBM collaborated on the development of the Spatial Extender, with each company contributing from its area of expertise. IBM added the necessary object extensions to the DB2 DataJoiner ORDBMS platform; ESRI employed these extensions to build Spatial Extender's ability to understand spatial data and functions.

This white paper describes the components of a GIS product, how the architecture is evolving, and the business advantages of integrating GIS with the RDBMS from the user's perspective. The paper also covers the architecture and functionality of IBM's DB2 Spatial Extender and assesses its impact on the marketplace.

THE EVOLUTION OF GEOGRAPHIC INFORMATION SYSTEMS (GIS)

A GIS understands geographic locations and proximity

A geographic information system (GIS) abstractly models spatial data by combining location and proximity intelligence with geographic mapping capabilities. It includes the actual geometry of objects (shape, size, location) stored as variable length records, related text attributes for each object for analysis purposes (for example, customer name and address for each location), specialized index structures, and an engine that interprets and manipulates spatial geometry. A GIS knows how to determine all locations in its database that are within a specified distance of another location, or whether two roads intersect. A GIS also provides robust data-visualization and analysis tools, enabling users to view their data in completely new ways. Instead of receiving a list of stores and their addresses, the user sees a map in color with store locations identified. The map may also display other information, such as major highways, to enhance the user's analytical context.

The Evolution of Spatial Queries

First-generation GIS:

Find all customers who live within 10 miles of the Alameda store.

Second-generation GIS:

Find all customers whose incomes are greater than $50,000 per year and who live within 10 miles of the Alameda store.

Third-generation GIS:

Find all customers whose incomes are greater than $50,000 per year and who live within 10 miles of the most profitable store in the eastern region.

Figure 1

Like document- and image-management systems and other specialized applications, GIS products are moving away from separate, proprietary software to full integration with the RDBMS. Figures 1 and 2 show the increasingly complex queries users want to ask and how the GIS's architecture is evolving to satisfy these requirements.

First-generation GIS is proprietary and only handles spatial data

In a first-generation GIS, the vendor's GIS software runs as an application on top of a file system. All GIS data and indexes are stored in flat files, and the GIS software is required to interpret and manipulate the data. These first-generation systems are designed to meet the needs of users at the project or departmental level where all required data is within the user organization's domain. Primary users of early GIS products have been local, state, and federal governments.

While many organizations use a first-generation GIS very successfully, these systems have several limitations. First, they can handle only spatial data, as illustrated in the first query in Figure 1. This is probably the most significant disadvantage from the end-user's perspective. A query that combines spatial data with other business data requires "glue code" as part of the application that runs on top of the GIS.

A first-generation GIS also uses a proprietary data model and interface; it is a separate, self-contained system for accessing and managing all spatial and related data. The GIS itself must manage the security, backup and recovery, and integrity of its data and indexes in the file system.

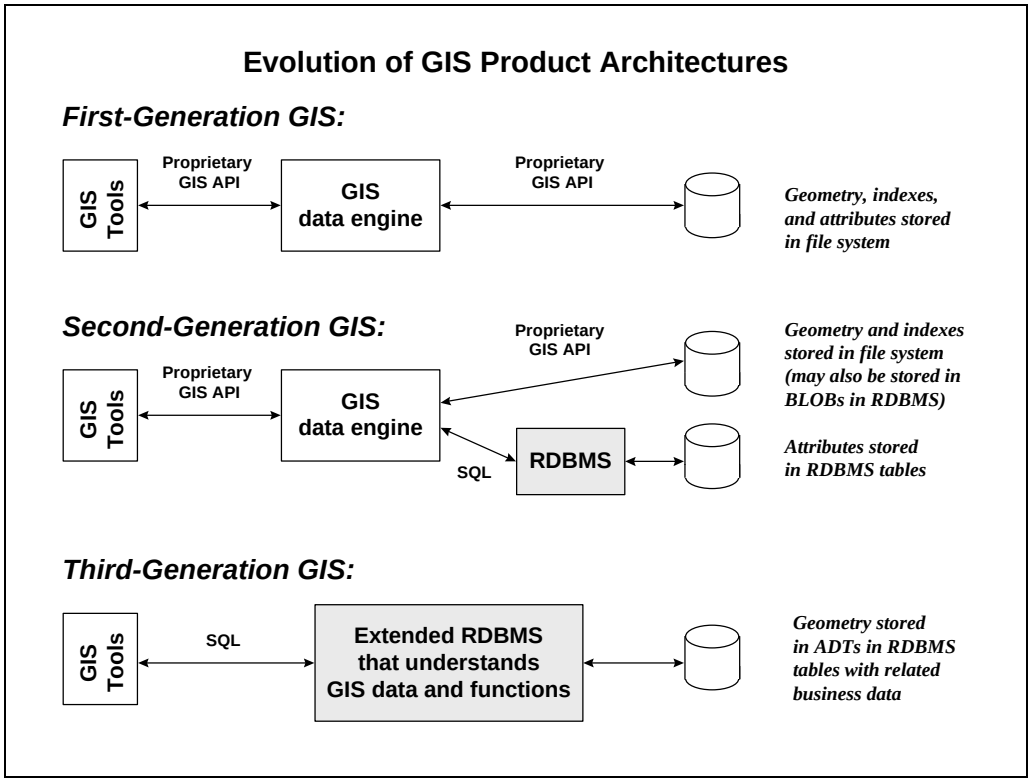

Figure 2

Second-generation GIS stores data in the RDBMS

Development of second-generation GIS was driven by the GIS vendor's desire to take advantage of the built-in data-management capabilities of the RDBMS. A second-generation GIS product can initiate a "handshake" with the RDBMS in order

to store GIS data and attributes in RDBMS tables. This makes life easier for the GIS vendor by transferring some of the data-management burden to the RDBMS. It also gives users access to additional non-spatial attributes and the ability to generate more complex queries involving spatial data. The second query in Figure 1 illustrates this. If customer income is now stored as an attribute in the RDBMS, the GIS issues an SQL query to the RDBMS to find all customers with incomes greater than $50,000 per year. The GIS then applies the spatial constraint to the customers that qualify. Most GIS solutions available today represent this architecture.

Limited query capabilities of GIS cannot exploit existing data

However, second-generation GISs still lack flexibility and do not directly integrate spatial data with other business data. The GIS controls the underlying RDBMS schema and the user must continue to use the proprietary GIS API to access and analyze spatial data and to create and maintain the attributes stored in the RDBMS. Therefore, data that already exist in the RDBMS (such as customer income) must be duplicated through the GIS and maintained in two places, unless the GIS includes a database integration tool to access the existing data.

The GIS continues to maintain its own indexes and cannot, on its own, exploit the performance benefits that integration with the RDBMS would offer. (These techniques, such as global cost-based query optimization and parallel processing, are described earlier in the paper). Second-generation systems also impose complexity on the database administrator, who must manage both systems and understand how they interact. This lack integration with existing RDBMS data has been a major obstacle for GIS vendors that want to provide services to large IT organizations.

A second-generation GIS solution is an appealing one for GIS-centric users who wish to continue using GIS tools and applications as the primary access method for their data, and do not need to combine GIS data with existing RDBMS data.

Third-generation GIS fully integrates spatial data into the ORDBMS

The ORDBMS, through its SQL extensions, provides an opportunity for the GIS to move to yet a third-generation architecture: full integration of spatial data within the ORDBMS. With support for abstract data types, user-defined functions, and user-defined indexes, the ORDBMS has the capacity to natively understand spatial data and operators, and to efficiently access spatial data. (Please refer to Appendix A for a glossary of object-relational terms.) Users can execute integrated queries across all types of data—spatial data and existing business data—using extended SQL. The user is no longer dependent on separate GIS software for dealing with spatial data. Instead, the user now gains the benefit of the GIS vendor's expertise within an integrated ORDBMS environment.

Third-generation GIS changes its orientation from *GIS-centric* to *DBMS-centric* in which spatial data is simply another data type within the ORDBMS environment. Users can now generate sophisticated queries that integrate spatial data with other ORDBMS data. (The third query in Figure 1 illustrates this.) The ORDBMS can apply its performance optimization techniques to GIS data as well as to traditional relational data. And the database administrator has a single set of system-management tools and procedures for managing the integrated environment. In summary, this more generalized solution to GIS functionality has the flexibility to meet a broader range of IT requirements and applications and to fit well into an existing data-management environment. Thus, it is an attractive alternative for many

organizations seeking to leverage their data assets to create valuable business intelligence.

IBM'S SPATIAL OFFERINGS

IBM supports both second- and third-generation GIS

IBM is clearly committed to providing customers with a fully-extensible data-management environment. Integrating spatial data with existing business data is an important part of this strategy, and IBM's efforts in this area are evolving in the same way as the GIS architectures described above. IBM's direction is to make both second- and third-generation GIS solutions available for DB2.

ESRI's SDE on DB2 UDB

ESRI's Spatial Data Engine (SDE) for DB2 UDB supports the second-generation GIS architecture and queries described in the previous section. It offers a valuable solution for organizations that wish to continue using ESRI's GIS API to manipulate spatial data. These users can also access other relational data through tools that ESRI provides.

DB2 Spatial Extender

As a third-generation GIS, the new DB2 Spatial Extender will provide the functionality of a GIS system within the DB2 ORDBMS through the addition of pre-defined spatial data types, spatial operators, the ability to generate spatial indexes, and optimization of spatial queries. All of this GIS functionality is integrated into the ORDBMS via SQL extensions. Thus, any SQL-based tool or application can take advantage of the extensions and users get integrated query access to all data types supported by the ORDBMS. With its tight integration of spatial functionality and optimized performance in handling spatial data, DB2 Spatial Extender is clearly positioned to address customers' business-intelligence requirements.

Support for both the Open GIS Consortium and SQL3 standards

IBM is committed to supporting the Open GIS Consortium standard for spatial data. Open GIS is a consortium of GIS vendors that has defined a set of geo-spatial data types and a set of functions to manipulate these types. The data types include, for example, a point (an x and y coordinate), a line (defined by its two end points), and a polygon (defined by its vertices). Sample functions include determining if two objects intersect, what objects are within a specified distance of another, and the area of a polygon. The Open GIS standard includes support for both a text and a binary representation of spatial data. ESRI, in building DB2 Spatial Extender, has incorporated support for both formats plus its own "shape" format.

IBM is also committed to the SQL3 standard. A component of this, SQL Multimedia (MM), defines standard SQL extensions for handling text, image, and spatial data. An effort is underway to align the SQL MM effort with the Open GIS standard so that both define the same functionality for spatial data (i.e., the functionality implemented in DB2 Spatial Extender).

DB2 DataJoiner applies spatial intelligence to heterogeneous data

IBM will initially deliver Spatial Extender on its DB2 DataJoiner ORDBMS platform, rather than on DB2 UDB, for one key reason. Business data can be stored anywhere and often is spread across multiple, heterogeneous data sources. Users want the ability to execute integrated queries on these data—for data-mining, OLAP, and other analytical applications—independent of where the business data are stored. The ideal solution, therefore, does not require the user to physically move or copy into the DB2 ORDBMS all of the business data that it wants to "spatially enable."

DB2 functionality, global catalog, global optimizer, SQL compensation

DB2 DataJoiner is IBM's solution for heterogeneous data access. (A white paper covering DB2 DataJoiner is available on IBM's Web site; see page 21.) DB2 DataJoiner provides transparent read/write access to all IBM RDBMS products plus VSAM, IMS, Oracle, Sybase, Sybase SQL Anywhere, Informix, Microsoft SQL Server, Oracle Rdb, and any database managers with ODBC- or X/Open-compliant call-level interfaces, such as Tandem Non-StopSQL (see Figure 3). DB2 DataJoiner is not just a simple gateway between DB2 and other database managers. It includes the following advanced features:

- Full functionality of the DB2 server—DB2 DataJoiner combines the DB2 ORDBMS with extensions for heterogeneous data access. Therefore, DB2 DataJoiner supports local DB2 tables, triggers, stored procedures, and other DB2 objects; and all DB2 management tools for backup/recovery and other administration operations.

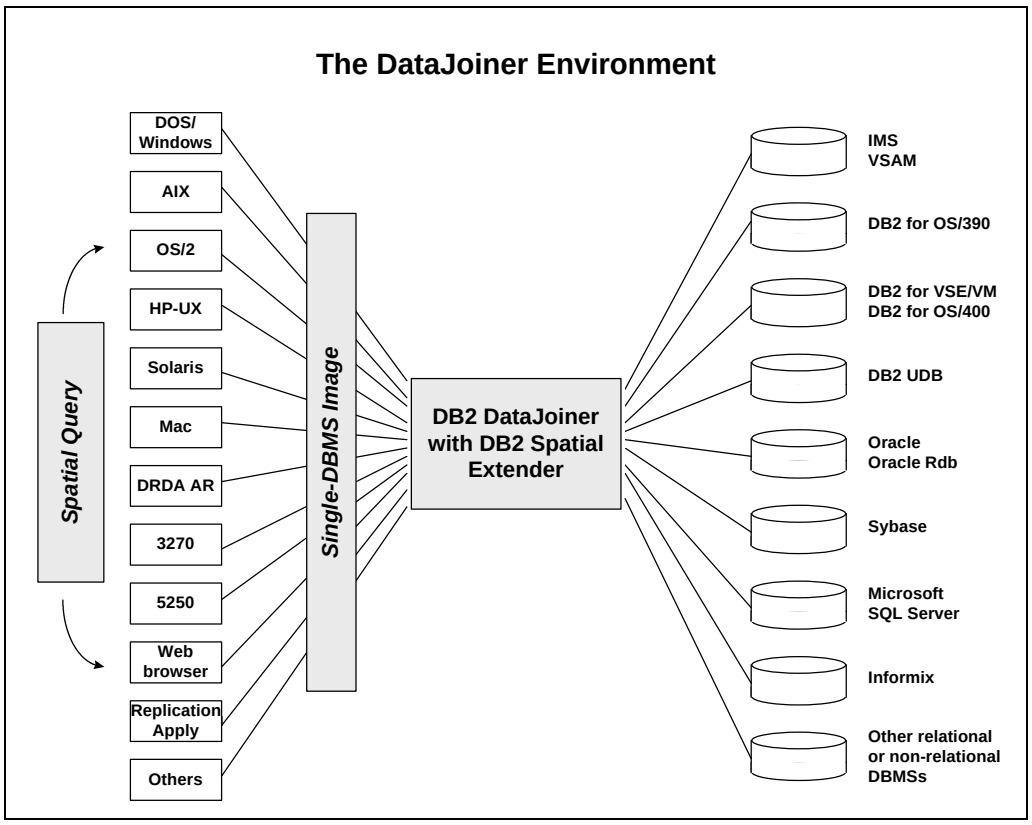

Figure 3

- Global catalog—DB2 DataJoiner's global catalog provides location transparency for heterogeneous data. It includes information about the type, location, metadata, and statistics of data stored in remote data sources.

- Global optimizer—The global optimizer is a key feature with its extensive knowledge about the various data managers supported and its distributed query capabilities. A join can be executed across any number of data sources, and the optimizer considers many factors in deciding how to do each join: volume and location of data, the capabilities of each participating data manager, network

costs, and others. DataJoiner can execute joins locally or on remote servers with the intelligence to move data among servers if that results in better performance.

- SQL compensation capabilities—Another impressive feature of DB2 DataJoiner is its ability to handle SQL compensation requirements. If a back-end server doesn't support a DB2 feature, DataJoiner may compensate for that rather than restrict the user's ability to take advantage of the DB2 functionality.

New object-relational extensions are the foundation

In summary, DB2 DataJoiner takes advantage of DB2's object-relational extensions, including the ability to simulate these capabilities, where possible, in non-DB2 data managers. In addition, DataJoiner v. 2.1.1 has new SQL3 extensions to support DB2 Spatial Extender. These extensions provide a generic infrastructure on which ESRI has defined the spatial data types, spatial functions, and methods for efficient access to spatial data that comprise the Spatial Extender itself. (These same extensions are also available to other third parties.) Thus, DB2 DataJoiner can deliver a SQL3-based, object-relational API for accessing data maintained in any supported data manager.

Spatial capabilities available on non-DB2 data sources

When combined with the heterogeneous access capabilities of DB2 DataJoiner, DB2 Spatial Extender allows the user to leave data where it is, but to make use of it immediately in a spatial context. Thus, DB2 DataJoiner could execute the third query in Figure 1 even if (1) customer data are in a DB2 database and stores data are in a non-DB2 data source or (2) both customers and stores are in a non-DB2 data source.

IBM will merge DataJoiner, Spatial Extender into next release of DB2 UDB

IBM will be merging DB2 DataJoiner's heterogeneous data-access capability and DB2 Spatial Extender with DB2 UDB in the next major release of UDB. This will result in a versatile ORDBMS engine that encompasses leading-edge database functionality in all areas of extensibility. DB2 UDB will then have all of the SQL3 extensions that are part of DB2 DataJoiner and of UDB v. 5, plus heterogeneous data-access functionality, plus new object extensions.

BUSINESS APPLICATIONS FOR DB2 SPATIAL EXTENDER

INTRODUCTION

Spatial intelligence is applicable across a broad range of industries

Traditional GIS users have been governments, utilities, and telecommunications companies. Today, many other commercial organizations have recognized the need to spatially enable their existing data in order to learn new things about their businesses and customers. Developments driving this increased interest in geo-spatial analysis are the enhanced GIS and mapping capabilities of third-party software; the commercial availability of geographic databases, such as a U.S. streets database or a database on toxic waste locations; and the ORDBMS's ability to integrate spatial data with other types of business data. Descriptions of how some industries can gain value from incorporating GIS functionality into the RDBMS have already been covered above.

In addition, two early beta customers of DB2 Spatial Extender, an independent software vendor (ISV) and an end-user organization, provided perspective on the need to incorporate support for spatial data within their business processes. Both clearly appreciate the advantages of a third-generation GIS solution and agree that integrating spatial-data intelligence into the RDBMS is the right approach.

PRECISION FARMING

Taking advantage of existing spatial data

The end-user organization is a large manufacturer of farm, construction, and grounds-care equipment. This company sees a pervasive need for GIS capability and is evaluating the benefits of implementing spatial-data intelligence at many levels. One new agricultural application is called site-specific agriculture or precision farming. Even within a single field there can be significant variability in soil conditions and crop productivity. To optimize the overall yield, the farmer would like to know within a few meters what the different conditions are and use this information to apply the right chemicals in exactly the right place at the right time. This can result in both productivity and environmental benefits.

Advanced electronics plus ORDBMS extensions

The convergence of two technologies is making this type of application a reality. The first is advanced electronic instrumentation techniques, such as extremely accurate global positioning systems (GPS) and enhanced wireless communication systems. The second is delivery of technology like DB2 Spatial Extender that incorporates geo-spatial and other complex data types as first-class citizens in the database. With spatial-data intelligence in the ORDBMS, the user can overlay with precision different views of geographical data. An example is the topology of a wheat field overlaid with DBMS statistics about soil conditions and yield information within that field. Support for images, video, and audio data types in the ORDBMS mean that spatial data can be enhanced by the collection of other data, such as videos of field operations or recordings that describe the condition of a crop in a particular location.

Spatial Extender offers a standard API for spatial data

The company has been experimenting with ESRI’s SDE and is now looking at the additional benefits offered by DB2 Spatial Extender. This organization is already familiar with DataJoiner, having used it successfully in several areas. By adding native intelligence about geo-spatial data to DataJoiner, Spatial Extender eliminates the need to manage and maintain two separate server architectures—the SDE GIS and the RDBMS—for spatial capabilities. The Spatial Extender also supports more sophisticated queries through its spatial operators and by enabling users to develop custom operations to interpret geographic data. Another advantage of Spatial Extender is the use of a standard query language for geo-spatial data (SQL3).

Comprehensive integration through partnership with ESRI

A key strength of IBM’s solution is the integration of spatial intelligence deep in the DB2 relational architecture through ADTs, index extensions, and the ability to take advantage of the DB2 query optimizer. (This is discussed in more detail in the section below on “How DB2 Spatial Extender Works.”) Supporting spatial data through BLOBs and spatial functions is not sufficient from this user’s point of view. The partnership with ESRI is also important for IBM. Working closely with ESRI is a proactive approach that enables IBM to deliver comprehensive, integrated GIS capability to customers more quickly than would be possible otherwise.

MANAGING COMPLEX NETWORKS

Networks have special modeling requirements

The second beta customer is an ISV that provides automated mapping/facilities management (AM/FM) solutions for managing complex networks. AM/FM is a subset of GIS that focuses on managing infrastructures that extend over large geographic areas. Networks in particular have special characteristics that must be modeled in the database, such as the connections, or relationships, between various components in the network. Typical customers include electric, gas, cable TV, and telecommunications companies, among others. The company's software allows customers to manage valuable resources by defining their locations and connectivity, and tracking the current status of each resource.

Moving to a 3rd-generation GIS product

This ISV organization has both a proprietary, first-generation product and a second-generation product that takes advantage of the RDBMS. The second-generation version continues to store the geometry outside the database in a proprietary format while non-spatial information is moved into RDBMS tables. Reference keys tie the proprietary data to the RDBMS data. The company is now moving toward a third-generation system to unify the management and storage of all data within the RDBMS. DB2 is one of the major ORDBMSs supported, and the company is participating in the beta program to understand the capabilities and features of the Spatial Extender and IBM's directions in spatial-data management.

The need to integrate and model complex data using ADTs

This company is also very positive about the benefits of using an object-relational approach for modeling spatial data in the DBMS. This approach enables the user to integrate spatial data with other business data and with other complex data types such as audio and video. "Giving the RDBMS more intelligence and data-modeling capabilities opens up a whole new world in terms of building much more robust applications. For example, the user can bring up a map on the screen, click on a manhole, and up pops a video on the manhole. This solves our need to model network connectivity better than a pure relational solution. We also view ADTs as the ideal way to implement spatial data types." (Alternatives to ADTs are discussed in more detail in the section below on "How DB2 Spatial Extender Works.")

Open GIS support is very important

The ISV intends to provide an Open GIS-compliant interface so its products can take advantage of any RDBMS that supports this standard. Therefore, Spatial Extender's support for Open GIS is an important advantage for IBM.

HOW DB2 SPATIAL EXTENDER WORKS

ARCHITECTURE

New object extensions

To support DB2 Spatial Extender, IBM is implementing object extensions beyond those already available in DB2 UDB. New extensions include abstract data types (ADTs) with sub-typing and inheritance, user-defined predicates, user-defined index structures, and the ability for the RDBMS to efficiently store and retrieve data that

can vary greatly in size. These extensions comprise the infrastructure required to enable the DBMS to understand user-defined types (UDTs), and ESRI has built DB2 Spatial Extender using these generic extensions.

ADTs use attributes to encapsulate internal structure

ABSTRACT DATA TYPES. ADTs are user-defined types that describe complex column structures to the RDBMS. The new type has an encapsulated internal structure represented as a set of attributes. ADTs allow the user to represent new entities in the database, such as images, text, and spatial data. An example of a text ADT is a "document" data type with attributes such as text (a LOB that contains the actual text), language (character), and format (character). A sample spatial ADT is a "polygon" data type with attributes such as geometry (a LOB containing the points representing the polygon), area (integer), and number of points (integer). ESRI has defined specific spatial data types as ADTs in the Spatial Extender (see Figure 4).

Users who need to define complex spatial types not yet supported by Spatial Extender, such as arcs, splines, ellipses, and circles, can define these as new ADTs or as sub-types of ESRI's pre-defined ADTs.

DB2 Spatial Extender Data Types

Data Type	Category	Sample Applications
Geometry	Super class	All geometries have the following attributes or properties: interior, boundary, exterior; simple or non-simple; empty or not empty; number of points; envelope; dimension; Z coordinates; measures; spatial reference system
Point Linestring Polygon	Base geometry sub-classes	Store or customer location Road, river, power line Water body, flood plain
Multipoint Multilinestring Multipolygon	Homogeneous collections sub-classes	Incidents of epidemic outbreak Network of roads, earthquake fault line Non-contiguous parcel of land such as an island chain

Figure 4

Multiple sub-types can be stored in the same column

ADTs also support "value substitutability." This is the ability to store different sub-types in the same column on a row-by-row basis. A column of type "geometry" can contain a point in one row and a polygon in another row if point and polygon are both sub-types of geometry. Without value substitutability, the data must be normalized so instances of each sub-type are stored in separate tables. This means, for example, that a query asking whether a point is contained within a particular polygon requires a join among three tables. In contrast, storing a variety of sub-types in one column avoids the necessity for joins when comparing spatial entities of different sub-types. This has two significant benefits. It enhances the developer's flexibility in designing the database structure and improves performance.

Flexibility in storing ADTs

Because the contents of ADTs can vary dramatically in size, the DBMS can determine whether data should be stored inline in a column in the table, or stored

separately from the rest of the row in a manner similar to the way LOBs are handled. A point, for example, is stored inline as a single set of x,y coordinates. A polygon representing the state of California (many thousands of points) is stored separately. When defining an ADT, the user can specify a threshold, or number of bytes, that the DBMS stores inline. The DBMS relies on this threshold to move data between inline and separate storage on a row-by-row basis. For example, the user can choose to keep attributes inline for fast reference. If an ADT value is very small (e.g, a point), it will be stored inline. If the value is very large, it may be stored partially inline and partially in separate storage or all separately.

Sub-types and inheritance support code reuse

TYPE HIERARCHIES. Another important feature of ADTs is support for sub-types and inheritance. By defining hierarchies of ADTs, the developer can reuse both attributes and methods. Figure 5 illustrates the type hierarchy in DB2 Spatial Extender and the fact that this hierarchy can be extended to incorporate new user-defined types, indicated by dotted lines.

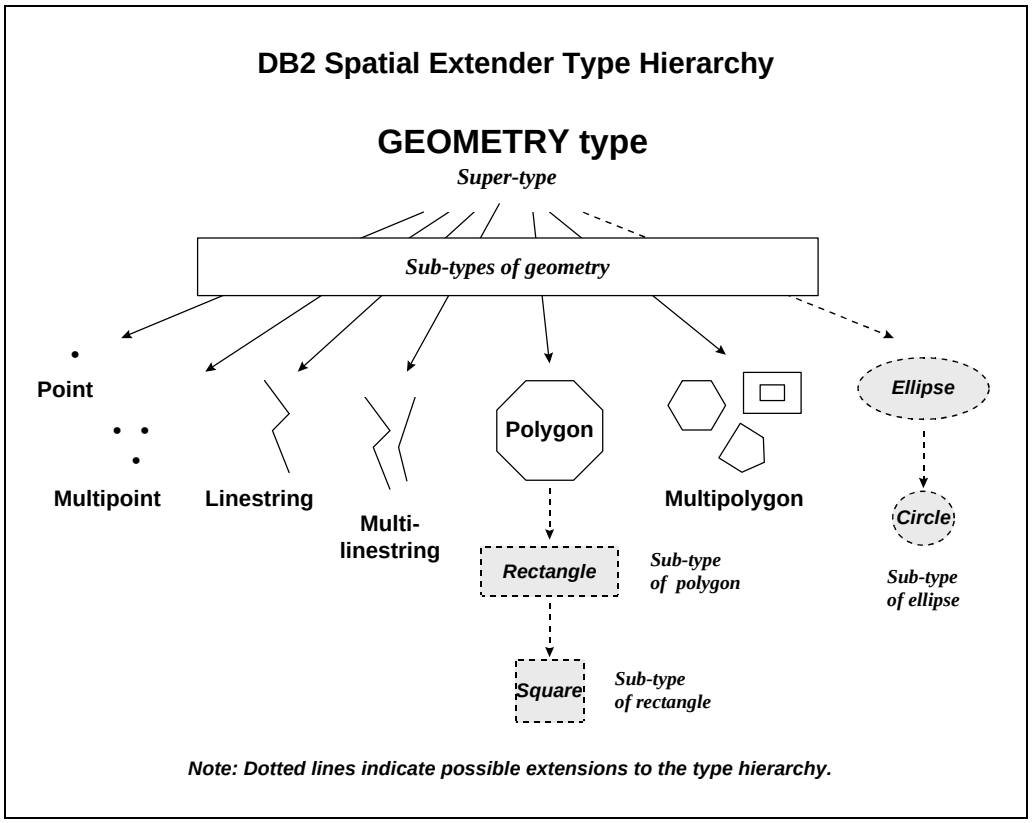

Figure 5

Each sub-type inherits the structure (attributes) and the behavior (methods or functions) of its super-type; additional attributes and methods can be defined for a sub-type and inherited methods can be modified as appropriate to differentiate the subtype. An example that applies to spatial geometry is a hierarchy of a shape, a polygon (a two-dimensional shape with three or more vertices), a rectangle (a polygon that has exactly four sides and four right angles), and square (a rectangle in which all four sides are of equal length). The shape is the super-type. The polygon is a sub-type of shape. Another sub-type of shape could be an ellipse, which has

different characteristics than a polygon. The rectangle is a sub-type of a polygon and the square is a sub-type of the rectangle. In the case of the rectangle and the square, the function to calculate "area" is the same, eliminating the need to define a separate method for a square. The square simply inherits the area function from the rectangle.

UDFs implement the behavior of new data types

USER-DEFINED PREDICATES. DB2 already supports UDFs, which implement operators, or methods, that can be applied to UDTs. DB2 Spatial Extender includes pre-defined functions for operations such as spatial calculations (for example, the distance between two points), comparisons (the customers located within a five-mile radius of a store), data exchange, and others. ESRI has developed all of the UDFs for the Spatial Extender (see Figure 6). To achieve better performance, Spatial Extender functions run in an "unfenced" mode in the same address space as the database server itself.

As predicates, UDFs trigger the use of indexes

To support Spatial Extender, IBM has also extended the notion of UDFs to allow them to be used as user-defined predicates. When a query includes, in a "where" clause, a UDF that is also defined as a predicate, the optimizer knows to look for an index on the column that would help in evaluating the predicate. User-defined predicates are used, for example, to compare two spatial objects, such as two polygons or a point and a polygon, to see if they overlap or if one is contained within another.

IBM Spatial Extender Functions

Type of Function	Sample functions
Comparison functions	contains, cross, disjoint, equals, intersects, overlap, touch, within, envelopes intersect
Relationship functions	common point, embedded point, line cross, area intersect, interior intersect, etc.
Combination functions	difference, symmetric difference, intersection, overlay, union, etc.
Calculation functions	area, boundary, centroid, distance, endpoint, length, minimum distance, etc.
Data-exchange functions	astext (for Open GIS well-known text representation), asbinary (for Open GIS well-known binary representation), asbinaryshape (for ESRI shape representation), etc.
Transformation functions	buffer, locatealong, locatebetween, convexhull, etc.

Figure 6

New index structures for fast access to complex data

USER-DEFINED INDEX STRUCTURES. B-trees, as typical RDBMS index structures, are designed to index types that can be ordered according to one dimension (such as integers—1, 2, 5, 7). Because most spatial data types are at least two-dimensional, B-trees cannot handle them. It is necessary to extend the RDBMS engine to support spatial indexes. Therefore, IBM opened up the DB2 index manager so that new index types—called index extensions—can be defined. This includes the type of index, which data types it works with, how index entries are generated from column values, and how the index can be exploited to evaluate queries efficiently. All of these index extensions are implemented through SQL extensions; this high-level API makes it relatively easy for the developer to integrate complex, user-

defined index functionality. ESRI used these extensions to implement support for spatial indexes, called grids, in the Spatial Extender.

New optimization techniques are a key extension

QUERY OPTIMIZER. The DBMS is aware of the existence of spatial operators and indexes and the query optimizer takes advantage of these when accessing and manipulating spatial data. In addition to specifying the cost of using a UDF, the developer can indicate whether the UDF can be used as a predicate in the "where" clause and can be evaluated using an index. This information helps the DBMS efficiently search spatial data.

Cheap algorithms for filtering data speed up performance

Spatial UDFs, such as comparing two polygons to see if they overlap, can be extremely expensive to execute in terms of both time and memory requirements. Therefore, optimization techniques to minimize execution cost are very important. One option is to provide "cheap" algorithms for filtering out spatial data that don't fit query criteria without actually executing an expensive UDF. ESRI, for example, uses an algorithm called "envelope" to accomplish this. Envelope, stored as an attribute of a spatial data type, is a set of points that defines the minimum rectangle that completely contains the geometry of the type. (The envelope of a point and a line is the geometry itself.) Any polygon whose envelope doesn't fall within the space identified in the search criteria can be easily eliminated from the search without reconstructing the actual geometry of the polygon. Polygons that do qualify in this initial pass are then further evaluated using the more expensive UDF.

The DBMS can take advantage of cheap, filtering algorithms in a couple of ways:

- If an index on a spatial column exists, the DBMS identifies which rows qualify by doing an index lookup. Because the index is built on the "envelope" algorithm described above, these rows need further evaluation.

- DB2 extensions also support "sargable" predicates. These are predicates that can be pushed down into and evaluated by the lowest-level component of the DBMS. In other words, as the DBMS is fetching data from disk, it applies the cheap algorithm to each row. If a row doesn't qualify, the DBMS doesn't even transfer the row for further evaluation within the engine.

The final step is to evaluate qualifying rows using the expensive UDF to see which ones really satisfy the user's request.

ALTERNATIVES TO ABSTRACT DATA TYPES FOR SPATIAL DATA

Alternatives do not give RDBMS spatial intelligence

The ability to define spatial data types as ADTs with attributes has advantages over two other alternatives. The key difference is that neither of these alternatives to ADTs gives the DBMS native intelligence about spatial data.

Using unstructured, opaque types for geometry

One alternative is to store geometry in unstructured "opaque types," or LOBs, along with attributes that are encoded as part of the LOB. Because the DBMS doesn't understand the content of the LOB, the user can only access attributes by calling UDFs that understand how to decode the contents of the opaque type to find the attribute value. With ADTs, the engine doesn't need these "decoder" UDFs because it understands the layout of an ADT and where the attributes are stored. The

attributes can be requested directly in a SQL statement, so the query using ADTs will most likely execute faster. For example, if "area" is an attribute for a polygon defined as an ADT, the DBMS engine knows how to find the attribute very fast. If the polygon is defined as an opaque type, the DBMS must take the whole LOB off the disk and pass it to the specified UDF which then locates or calculates the area. The ability to index on the output of a function would improve query performance in the case of opaque types, but would degrade update performance because the index must also be updated.

Another disadvantage of using opaque data types is the lack of support for sub-typing and inheritance. Therefore, instead of allowing the DBMS to decide which version of the "area" function to call based on the data type involved, there must be an "area" function defined for each type and the user must invoke the correct function for a particular type.

Using existing relational data types for geometry

The second alternative is a "pure relational" approach in which geometry is stored as a normalized set of fixed-length fields. For example, a point is stored as an x- and y-coordinate (each a column value). A line is stored as a reference to two points (its endpoints), etc. Reconstructing any geometry other than a point, therefore, requires joining data from multiple rows and a layer of specialized software on the application side that knows how to assemble the geometry. As with opaque types, the DBMS doesn't natively understand the meaning of the data as geometry; it can only use its generic access methods for standard data types such as integers and characters. Performance is obviously an issue with this approach.

GIS vendors and users are clear on the benefits of ADTs in enabling the RDBMS to understand and manage new types of data. ESRI and both of the beta users of Spatial Extender reached this conclusion based on their experiences with GIS applications.

TAKING ADVANTAGE OF DB2 SPATIAL EXTENDER

Spatial Extender provides the foundation

Once the user has installed DB2 Spatial Extender, spatial data types, operators, and grid indexes are available as built-in functionality. Here are the steps necessary to take full advantage of Spatial Extender's capabilities.

Enable the database, create spatial columns, load the geometry

The first step is to specify which database(s) should be enabled for spatial searches. The next step is to create new tables with columns defined as spatial types or add new spatial columns to existing tables within these databases. To illustrate this process, let's assume an existing database with tables containing information about customers and stores with the following structures:

customers:	
name	varchar (30)
id	int
income	float
address	varchar (30)
state	char (2)

stores:	
id	int
name	varchar (30)
address	varchar (50)
revenue	float
state	char (2)

With Spatial Extender, the existing "customers" table, for example, can now have a column for the geographic location of each customer, defined as a point, and the "stores" table can have a similar column for location, as follows:

```
ALTER TABLE customers ADD COLUMN location point
ALTER TABLE stores ADD COLUMN location point
```

The next step is to insert the geometry data and attributes into the new columns. One option is to use a "geocoder" utility. This takes an existing address and constructs the geometry as x,y coordinates. However, the ability to geocode addresses is not uniformly or universally supported; it varies from country to country and requires a structured streets network against which to map the addresses. DB2 Spatial Extender ships with a geocoding facility and a streets database for major U.S. cities. Other user-defined geocoders can be built for applications such as precision farming where the usual "street address" technique does not apply.

Create spatial indexes using index extensions

Another step is to create spatial indexes over the spatial columns to improve performance on spatial queries. There is a level of tuning required to create an optimal spatial index from a performance standpoint (that is, defining the best grid size, which depends on the size and the size variation of the geo-spatial object in the index). The grid index extension itself is already defined by ESRI as part of Spatial Extender; here are the SQL statements for creating spatial indexes on the customers and stores tables:

```
CREATE INDEX CustLoc ON customers (location)
        USING spatial_index (120000.0, 0, 0)
CREATE INDEX StorLoc ON stores (location
        USING spatial_index (120000.0, 0, 0)
```

Now the user can include spatial constraints in queries such as the following. This query uses both indexes created above.

Find the name and address of all California customers whose income is greater than $50,000 per year and who do not live within 10 miles of any store in California.

```
SELECT c.id, c.name, c.address
        FROM customers c, stores s
        WHERE c.state = 'CA' AND
                c.income > 50000 AND
                s.state = 'CA' AND
                NOT EXISTS
                        (SELECT id FROM stores s WHERE
                        distance (c.location, s.location) <= 10)
```

Add map data with MapLoader

Now, to generate queries that require geographic information about entities outside the database, such as rivers or city limits, one more step is necessary: The relevant map data must be loaded into relational tables. DB2 Spatial Extender includes a MapLoader tool that can take commercially-available map data in ESRI's "shape" format and load it into relational tables. (ESRI has published this format for use by others. There are also transformation functions in Spatial Extender that can translate other map formats into the ESRI format.) Spatial Extender also includes sample

geographic feature data. With map data in the database, users can generate spatial queries such as this one (the index on customer location is again used to speed up the query):

Find all customers who live within the San Jose city limits.

```
SELECT c.id, c.name, c.address
        FROM customers c, usCities u
        WHERE u.name = 'San Jose' AND
                within (c.location, u.boundary) = 1
```

Tools to visually display spatial data are an important component

While these spatial queries can be generated directly using SQL, data visualization is particularly valuable for spatial data. So visual GIS tools are an important component in exploiting Spatial Extender functionality. Such tools provide two benefits. First, they can visually display the results of a query. For example, the query above results in a map of San Jose with dots showing the location of each customer. Second, the user can visually generate queries with the tool. Viewing customers as they are distributed throughout San Jose may prompt the user to add store locations to the map for analysis purposes. Then the user might draw a two-mile circle around a particular store in San Jose and ask for customers who have purchased at that store to see the distribution in relation to the circle.

DB2 Spatial Extender includes ESRI's ARCview tool for visualization of spatial data. ARCview also includes the ability to correlate different layers of spatial data by putting one on top of another. For example, the user could display a map of California as the bottom layer and overlay it with store locations and then with customer locations. Queries can be generated to highlight specific data of interest, such as the one above that identifies all customers with incomes over $50,000 that don't live within 10 miles of any store.

DB2 Spatial Extender customers can also use ESRI's other GIS applications, such as ARC/Info and MapObjects, against any data source supported by DB2 DataJoiner. Another important set of tools for customers are third-party tools and applications that are compatible with ESRI's GIS products. A wide variety of such applications are available off-the-shelf to enable customers to make use of spatial data. If a vendor has built a crime-analysis system in ARCview, a Spatial Extender user can take advantage of this pre-defined application. Other vendors' GIS tools can also be used instead of those from ESRI if the tools are compatible with ESRI's SDE API.

DB2 DataJoiner can apply spatial functionality to foreign data

And finally, all of this functionality is available even if the data are outside the local DB2 engine. Let's assume stores data are in an Oracle database and customer data are in IMS files (see Figure 7). (Either could be stored in any other backend that DB2 DataJoiner supports.) Because spatially-oriented data, such as addresses, are in another data source, the ADTs representing the geographic locations of stores and customers and the necessary index structures to speed up spatial queries are created in a local DB2 database. The geocoder software is then applied to each foreign address character string and the resulting geometry stored in the local ADT. Replication capabilities are used to receive notification of any changes to this address data so that the local ADT can be updated as well. For the stores data in

Oracle, for example, DB2 DataJoiner automatically creates the necessary triggers in Oracle to trap address changes that DataJoiner needs to know about.

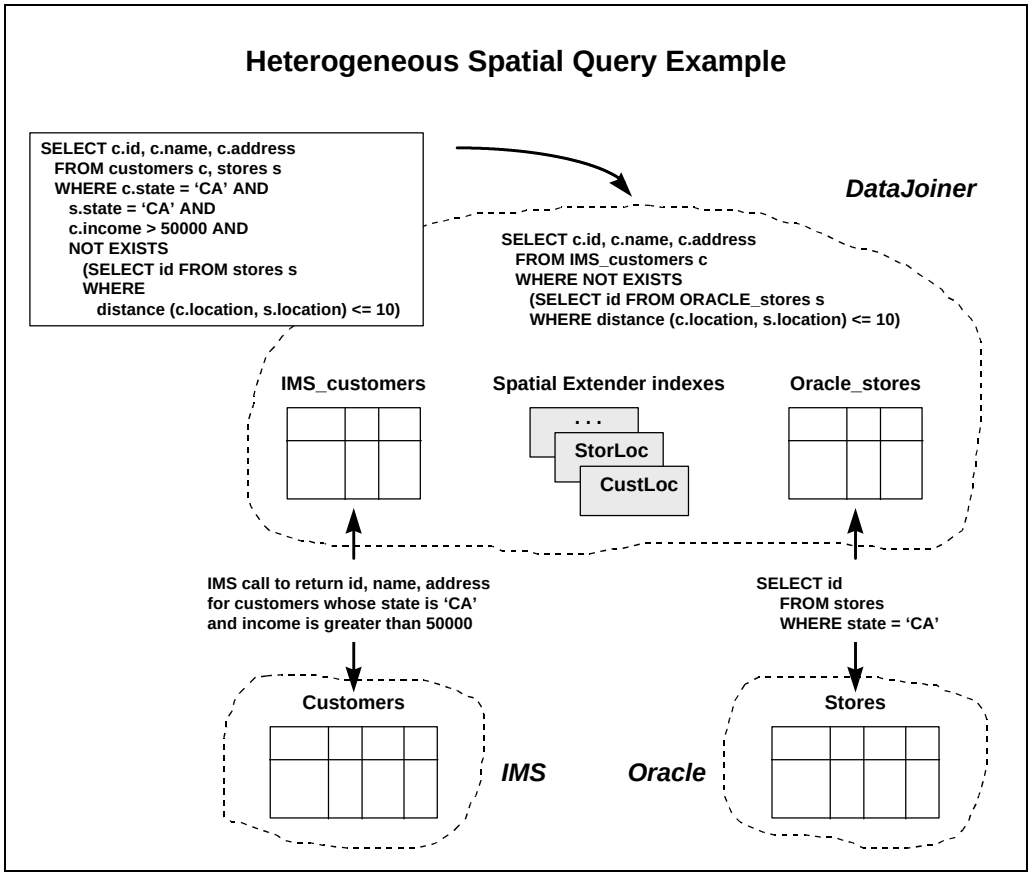

Figure 7

In the case of the first query above, DB2 DataJoiner pushes down the selects to the back-end data manager: Oracle selects stores in California and IMS selects customers with incomes greater than $50,000 and California addresses. DB2 DataJoiner then combines the results extracted from the foreign data sources and evaluates the spatial component of the query locally. All of this is physically transparent to the user; the query is written as if all the data are stored in local DB2 tables (as illustrated in the syntax of the original query).

CONCLUSIONS

DB2 Spatial Extender offers impressive benefits to customers

DB2 Spatial Extender illustrates the considerable strengths of the advanced object extensions IBM is adding to DB2. These provide a solid foundation on which ESRI has implemented GIS intelligence and full integration of geo-spatial data within the ORDBMS environment. DB2 Spatial Extender, when combined with DB2 and DB2 DataJoiner extensions, provides significant benefits to users:

- Full integration of geo-spatial intelligence with the DB2 ORDBMS and SQL, including indexing and cost-based query optimization for improved performance.
- Full support for popular GIS tools with built-in support for sophisticated rendering of map data for business-intelligence visualization, a geocoding function for U.S. addresses, and a sample set of world map data.
- Full support for existing GIS data in three industry formats (ESRI shape format, OGIS well-known text format, and OGIS well-known binary format).
- Compliance with emerging industry standards (SQL3, SQL/MM, and OGIS).

Applying spatial intelligence across heterogeneous data is key

With DB2 DataJoiner's federated data-access capabilities, Spatial Extender can also apply spatial intelligence to data stored in heterogeneous data sources. Introducing Spatial Extender on DB2 DataJoiner demonstrates the important role transparent, heterogeneous data access plays in allowing customers to apply extended DBMS functionality across a wide array of data sources. This also gives IBM the opportunity to attract both DB2 and non-DB2 customers who want integrated geo-spatial functionality and the ability to leverage existing data assets. This is an important differentiator for IBM.

The partnership with ESRI has enabled IBM to quickly deliver comprehensive, integrated spatial functionality and a wide variety of GIS tools. IBM's third-generation architecture also means that other third-party vendors can create their own extenders or add new functionality on top of DB2 Spatial Extender. Encouraging other vendors to take advantage of IBM's extensions will ensure choice and breadth of functionality for customers.

IBM and ESRI need to educate the IT/business community on the benefits of adding spatial information to the ORDBMS and the new types of applications and analysis that are now possible. IBM clearly plans to focus on the business-intelligence requirements of users in addition to those who have been traditional users of GISs.

IBM is well on its way to fulfilling its promise to offer customers a fully-extensible data-management environment. The combined set of capabilities that DB2 UDB ultimately represents—object extensions plus parallel execution of database operations, DataLinks, DB2 DataJoiner's heterogeneous data access, and all of the DB2 Extenders, including DB2 Spatial Extender—provides a formidable platform on which IBM, third parties, and customers can integrate new functionality and applications to meet unique and ever-changing business requirements.

IBM white papers are available on the following IBM web site:
http://www.software.ibm.com/data/pubs/papers

Information about ESRI's products are available on ESRI's web site:
http://www.esri.com

APPENDIX A. OBJECT-RELATIONAL GLOSSARY

Abstract data type (ADT)

Abstract data type (ADT) commonly refers to structured types that are used to define columns in tables. See "structured type."

Distinct type

Used to extend an existing base data type for a column or to encapsulate arbitrarily complex internal structures not expressed in SQL (examples are C or Java data structures). In the latter case, a host language such as C is used to define structure and attributes; the DBMS sees the type as a large object, varchar, integer, etc. User-defined functions are used to access structure and attributes. A distinct type is represented as a built-in data type and is sometimes called an "opaque" type. Distinct types do not support sub-types or inheritance.

Extensible indexing system

The ability to define and use index structures other than traditional B-tree indexes.

Type hierarchies

Type hierarchies use inheritance to enable the developer to reuse the attributes and methods of a super-type and modify them as appropriate to differentiate a sub-type. An example is a hierarchy of a polygon, rectangle, and square. See "sub-types."

Large objects (LOBs)

Large objects (LOBs) are data types that allow the RDBMS to store a large amount of data (up to 2GB) in a single column.

Opaque type

See "distinct type."

Row type

Row type commonly refers to structured types that are used to define rows in a table. See "structured type."

Structured type

Used to describe an entire row or a set of nested attributes in a column of a table as a single user-defined type (UDT). The DBMS is aware of the attributes and internal structure of the type. A structured type is represented as a list of attributes or columns. Structured types support sub-types and inheritance and can be used either as abstract data types (ADTs) or row types.

Sub-types

A sub-type inherits the structure (attributes) and the behavior (methods or functions) of its super-type; additional attributes and methods can be defined for a sub-type and inherited methods can be modified as appropriate to differentiate the subtype.

Unstructured type

See "distinct type."

User-defined function (UDF)

Used to define the methods by which applications create, manipulate, and access data stored in user-defined types (UDTs).

User-defined type (UDT)

User-defined types (UDTs) enable the user/developer to define new data types, model business objects, and represent complex data or business relationships. See "distinct type" and "structured type."

ABOUT THE AUTHOR

Judith R. Davis is an independent consultant with over 15 years of experience in the business application of information technology. Her primary interest is database management systems and related technology. She is a respected consultant and analyst in this area and has worked extensively with users and major vendors. She has also authored many articles for IT publications and magazines. Most recently, she has focused on the technology and business requirements driving the development of object extensions to the relational DBMS, and the role that the object-relational DBMS plays in creating a fully-extensible data-management environment.