CUSTOMER RELATIONSHIP MANAGEMENT AND LOCATION BASED SERVICES WITH ORACLE SPATIAL DATABASE

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Computer Science & Engineering

By:
B. Sridhar
(80732003)

Under the supervision of:
Parteek Bhatia
Sr. Lecturer, CSED

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

JUNE - 2009

Certificate

I hereby certify that the work which is being presented in the thesis entitled, **"Customer Relationship Management and Location Based Services with Oracle Spatial Database"**, in partial fulfillment of the requirements for the award of degree of Master of Engineering in Computer Science and Engineering submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Parteek Bhatia and refers other researcher's works which are duly listed in the reference section.

The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.

(B. SRIDHAR)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(PARTEEK BHATIA)
Sr.Lecturer,
Computer Science and Engineering Department,
Thapar University, Patiala.

Countersigned by

18/06/2009

(SEEMA BAWA)
Professor & Head,
Computer Science & Engineering Department,
Thapar University,
Patiala.

6/7/09

(R.K.SHARMA)
Dean (Academic Affairs),
Thapar University,
Patiala.

i

Acknowledgement

No volume of words is enough to express my gratitude towards my guide **Mr. Parteek Bhatia**, Department of Computer Science & Engineering, Thapar University, Patiala, who has been very concerned and has aided for all the materials essential for the preparation of this thesis report. He has helped me to explore this vast topic in an organized manner and provided me with all the ideas on how to work towards a research-oriented venture.

I am also thankful to **Dr. (Mrs.) Seema Bawa,** Head of Department, Computer Science & Engineering Department and **Mrs. Inderveer channa**, P.G. Coordinator, for the motivation and inspiration that triggered me for the thesis work.

I would also like to thank the staff members and my colleagues who were always there at the need of the hour and provided with all the help and facilities, which I required, for the completion of my thesis work.

Most importantly, I would like to thank my parents and the almightly for showing me the right direction out of the blue, to help me stay calm in the oddest of the times and keep moving even at times when there was no hope.

B. Sridhar

(80732003)

Abstract

A spatial database is a database that is optimized to store and query data related to objects in space, including points, lines and polygons. While typical databases can understand various numeric and character types of data, additional functionality needs to be added for databases to process spatial data types. We have the various applications for spatial database such as Mapmaking, Cartography, Emergency response planning, Urban development, Public transportation monitoring, Location-based services, Crime tracking and route planning.These applications can be solved by the GIS tools in the earlier days. Retrieving the information from GIS tools is very slow because no spatial indexing is used. In GIS tools are the loosely coupled system and have a poor integration between the third party tools and traditional DBMS.

We have used introduced the Oracle Spatial features of oracle 10g. It capable to handle all types of spatial data with efficient manner and above mentioned applications can be solved by using oracle spatial database. It supports GIS tools and uses the integrated system (Single Architecture), it is fully integrated with GIS tools and database applications. We can also use oracle spatial for location analysis without using the GIS tools.

In oracle spatial, the SDO_GEOMETRY object type is used to store the different geometric shapes, geographic data and related space. It uses the indexing technique to make the working more fast and efficient. To analyze the spatial data, it supports the spatial operators and spatial functions. The spatial operators are useful in proximity analysis, distance based analysis and intersection based analysis. The spatial functions are used to perform the calculations on geometries such as area of polygon and length of the geometry. It also supports the concepts of Network Modeling, Raster, Spatial Data Mining, Geocoder, 3D Types, Web Services and so on.

In thesis we identified problems in the areas of Customer Relationship Management (CRM) and Location Based Services (LBS) as application systems. The problems of CRM system like find area of sensitive customers which exist in those regions where our

competitors are also operating, find the closest customers from the location of a specific retail store are solved and implemented with oracle spatial database.

The problems of LBS system like find the marketing area to buy all different types of products in only place, find marketing areas of products from specified region within distance of 2 or 5 miles and find the marketing area of products having no competitors are solved and implemented with oracle spatial database.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Introduction

Location is an inherent part of business data, the organizations maintain customer address lists, own property, ship goods from and to warehouses, manage transport flows among their workforce, and perform many other activities. A majority of these activities entail managing locations of different types of entities, including customers, property, goods, and employees. Those locations need not be static. In fact, they may continually change over time. For instance, goods are manufactured, packaged and channelled to warehouses and retail/customer destinations. They may have different locations at various stages of the distribution network.

Let's consider an example of parcel services to illustrate how location is used. We have become increasingly accustomed to monitoring the status of parcel deliveries on the Web by locating our shipment within the distribution channel of our chosen service supplier. The simplicity and usefulness of this service is the result of a very complex underlying information system. The system relies on the ability to locate the parcel as it moves across different stages of the distribution network. Many information systems share location information in this process, which can be used to estimate, for instance, transit or delivery times. Systems such as RFID (Radio Frequency Identification, a technology to exchange data between tags and readers over a short range) are used to automatically record the movements of parcels along the distribution chain. Aircraft, trains, container ships, or trucks that move goods between distribution hubs use systems such as Global Positioning System (GPS) to locate their positions in real time. Even the "last mile" that is, the delivery of an individual parcel to the end customer is based on the geographical optimization of the delivery schedule as well as on the ability to locate the truck drivers in real time, to guide them to their destinations, and to estimate delivery times.

All of this location information is stored, analyzed, and exchanged between multiple systems and is the basis for making the entire operation cheaper, faster, and more reliable. Most of these systems are connected to each other through the Internet. The end user also uses the Internet to access the system and to query the current status of

his parcel. By analyzing the system in its entirety, we can recognize that the added value is the result of the integration of various systems, of their interoperability, and of the pervasive role of spatial information across the entire process. Spatial information plays a crucial role in enabling the systems and processes to run smoothly and efficiently.

This example illustrates the pervasiveness of location or spatial information in day-to-day business. In fact, market research estimates that the majority of the data handled by organizations perhaps as much as 80 percent of all data has a spatial dimension. The ability to properly manage the "where," or the spatial information, is key to the efficiency of organizations and could translate to substantial costs savings and commercial competitiveness. For instance, healthcare, telecommunications and local government organizations depend on spatial information to run their daily business. Other organizations in the fields of retail, distribution, and marketing use spatial information for strategic decision making. For example, choosing store locations, making investment decisions, examining market segmentation, and supporting clients.

At one point in time, the Internet seemed to have made location irrelevant. The Web emerged as a location less cloud, where we could contact anybody around the world instantly and shop anywhere without the usual constraints of geography. It seemed that the worlds of transport, logistics, and location received a critical blow. Of course, that thinking was naive. The Internet has made geography even more relevant and has bound digital and physical worlds closer than ever. It is now possible to do business over much farther distances, and tracking the locations of different components of a business and analyzing them have become all the more important.

1.2 Spatial Information Management with GIS Tools

Software tools for spatial information management have been traditionally known under the name of Geographical Information Systems (GIS).

The Geographic Information Systems (GIS) integrates hardware, software, and data for capturing, managing, analyzing, and displaying all forms of geographically referenced information. The evolution of GIS mainly from government then started development from commercial software vendors. The commercial software vendors of GIS are ESRI (Arcview/ArcInfo), MapInfo Corporation (MapInfo), Intergraph

(GeoMedia), Autodesk (Autodesk Map software/AutoCAD) and Laser-Scan (different packages). These tools are share and view the spatial data through internet and stores attributes, raster images, vector data which is shown in Figure 1.1. They have the capability to provide business logics such as relationships, domain values, networks, subtypes.

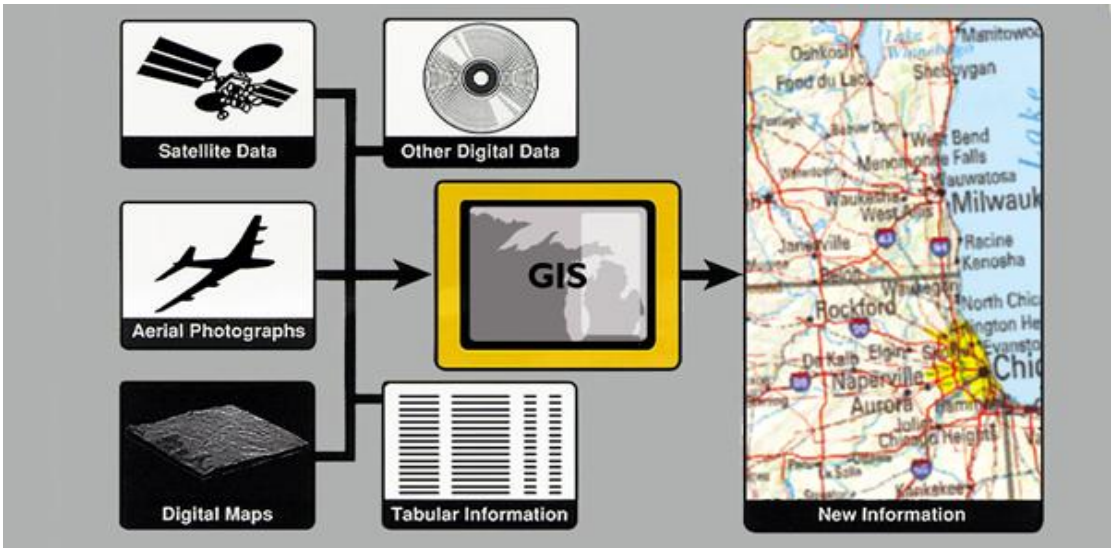


Figure 1.1: Geographical Information System (GIS) [20]

The GIS systems are specialized applications for storing, processing, analyzing, and displaying spatial data. They have been used in a variety of applications, such as land-use planning, geomarketing, logistics, distribution, network and utility management, and transportation [2]. However, until recently GIS have employed specific spatial data models and proprietary development languages, which held them separate from the main corporate databases. This has represented a barrier for the full deployment of the added value of spatial data in organizations.

As the use of GIS in enterprises and in the public sector has grown in popularity, some of the limitations of GIS have become apparent. Organizations often have to deal with multiple and incompatible standards for storing spatial data and they have to use different languages and interfaces to analyze the data. Furthermore, systems such as Customer Relationship Management (CRM) and Enterprise Resource Planning (ERP) or the systems used in logistics increasingly rely on the integration of spatial information with all other types of information. This has often been an operational and technical challenge that in some cases was solved by manually extracting

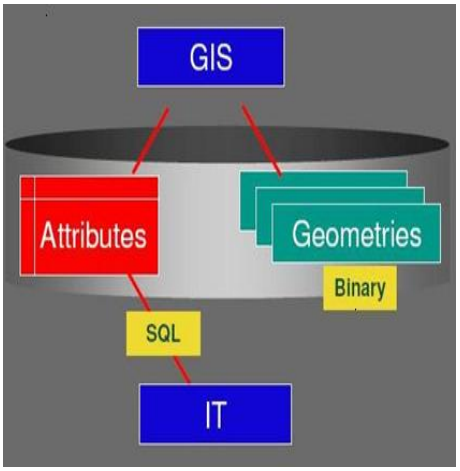
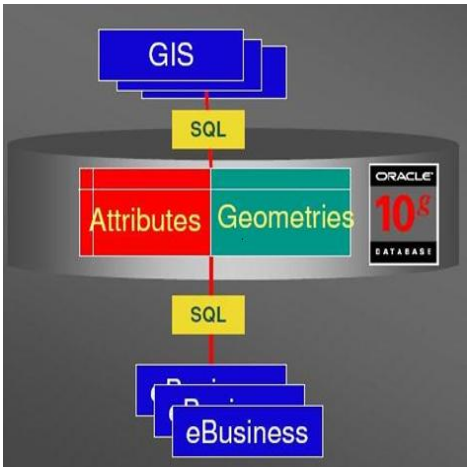
information from one system and loading it into another to perform the necessary spatial analysis.

1.3 Spatial Database Management System with Oracle Database

Oracle Spatial has an important role in changing the above situation. Once the spatial data is stored in an Oracle database, it can be processed, retrieved, and related to all the other data stored in the database such data that spatial information, or location, is just another attribute of a business object. This eliminates both the need for coordinating multiple data sources because of an application's dependence on special data structures and using different languages to query the data. Relevant features of Oracle Spatial are the ability to access spatial data through SQL statements, just like any other database content, and support for industry standards for spatial information (SQL and Open Geospatial). Above all, Oracle Spatial facilitates leveraging the full added value of spatial information, which becomes an integral part of the information assets of organizations.

1.4 GIS vs. Oracle Database

The following Table 1.1 shows the differences between the GIS and Oracle Database [14].

GIS	Oracle Database
1. GIS tools are loosely coupled system.	1. It is an integrated system.
2. Poor integration, It is a Dual Architecture. *i.e.*, 	2. Fully integrated, Single Architecture. *i.e.*, 

GIS	Oracle Database
3. Incompatible for e-business applications.	3. Compatible for all applications of GIS and e-business applications.
4. No indexing is used to retrieve the information.	4. Indexing is used to retrieve the information.
5. Separation between the relational data and spatial data.	5. No separation between the relational data and spatial data.

Table 1.1: GIS vs. Oracle Database

1.5 Evolution of DBMS Technology

The following Figure 1.2 shows the evolution of DBMS technology.

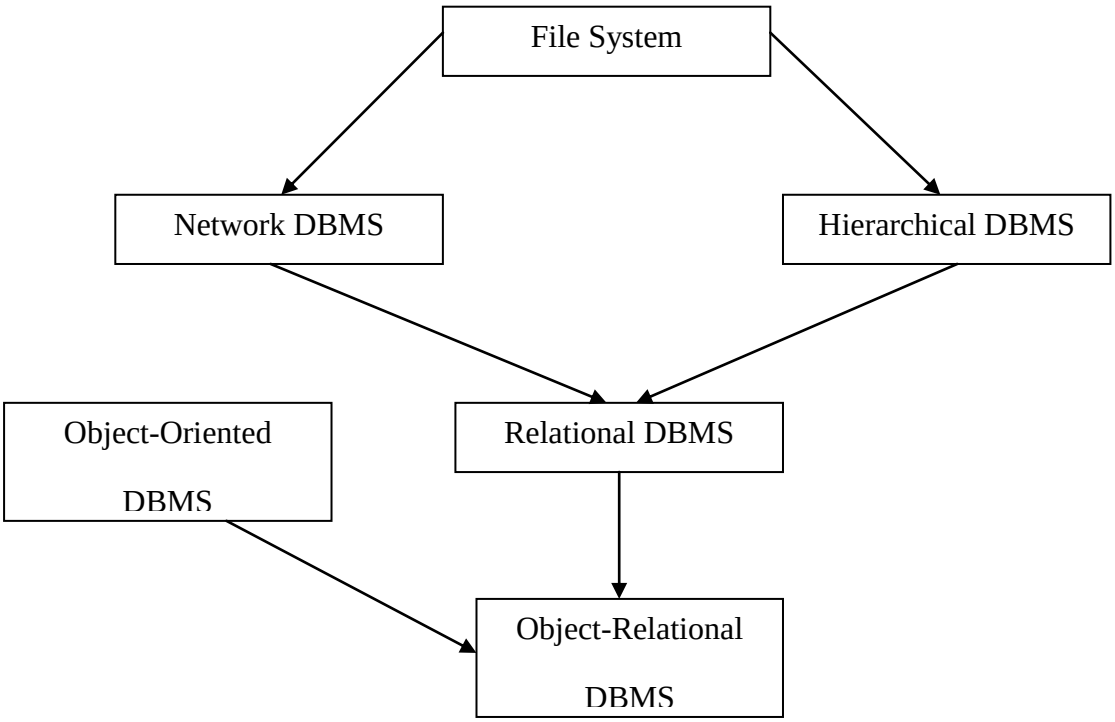


Figure1.2: Evolution of DBMS Technology

Spatial database is embedding in the object-Relational ORDBMS. [1]

1.5.1 History Sheet of Oracle Spatial Technologies

Spatial technology evolved between different versions of Oracle. Figure 1.3 shows the progression from Oracle 7.2 to Oracle 11g.

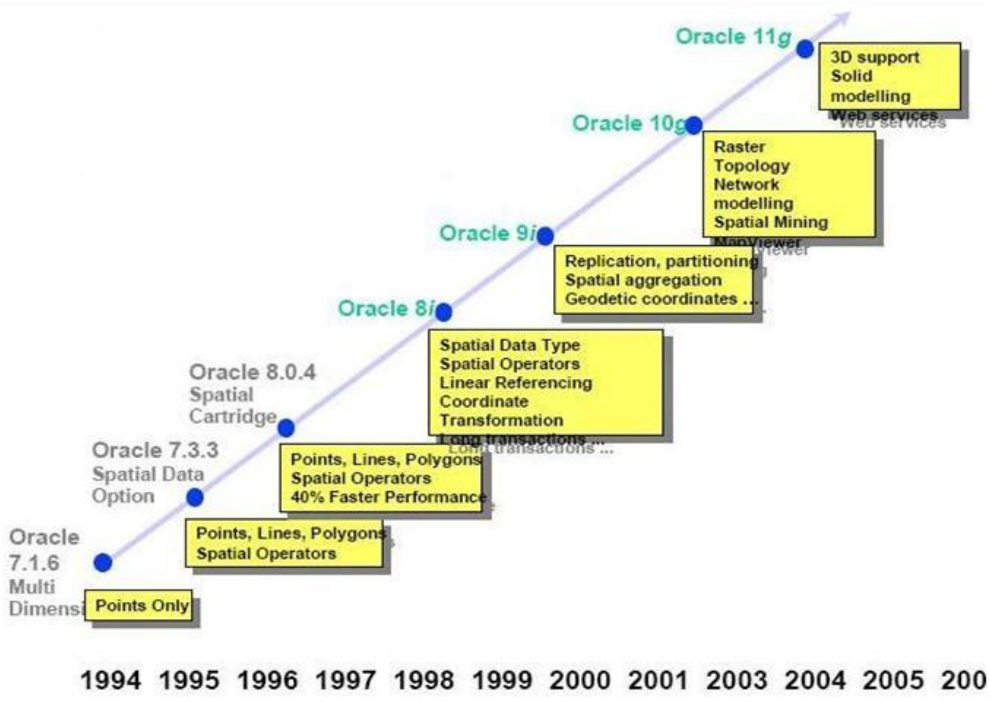


Figure 1.3: Evolution of Spatial Technology in Oracle [20]

Spatial technology was first introduced in Oracle 7.2 under the name Oracle MultiDimension (MD). Later, the product name changed to Oracle Spatial Data Option (SDO) and to Spatial Data Cartridge in Oracle 8. Since objects were not supported in these releases, the coordinates of a geometry were stored as multiple rows in an associated table. Managing spatial (geometry) data in these prior versions was inefficient and cumbersome. Starting with Oracle 8i, the SDO_GEOMETRY data type was introduced to store spatial data. Even in the latest versions (Oracle 11g, Oracle 10g, and Oracle 9i), the same SDO_GEOMETRY model is used to store spatial data in Oracle. In Oracle 9i (and Oracle 10g), the geometry data also included support for coordinate systems information specified using the SRID attribute in the SDO_GEOMETRY data type. In Oracle 10g, additional functionality (that exists in the Advanced Spatial Engine) such as the Network Data Model is introduced in the Spatial option of Oracle. In Oracle 10g Release 2, the EPSG (European Petroleum Survey Group) Coordinate Systems model was added to the Locator option. In Oracle 11g, several new features such as 3D geometry support and Spatial Web Services were introduced.

1.6 Introduction to Oracle Spatial Database

Oracle Spatial Database is designed to make the storage, retrieval, and manipulation

of spatial data easier and more natural to users. Once this data is stored in database, it can be easily and meaningfully manipulated and retrieved as it relates to all the other data stored in the database.

A spatial database is a collection of spatial data types,spatial analysis,spatial indexing. Spatial data types are to store the spatial data ,spatial analysis can be down by spatial operators and spatial functions, and spatial indexing is for fast accessing of spatial data [6].Spatial access through SQL is shown in following Figure 1.4.

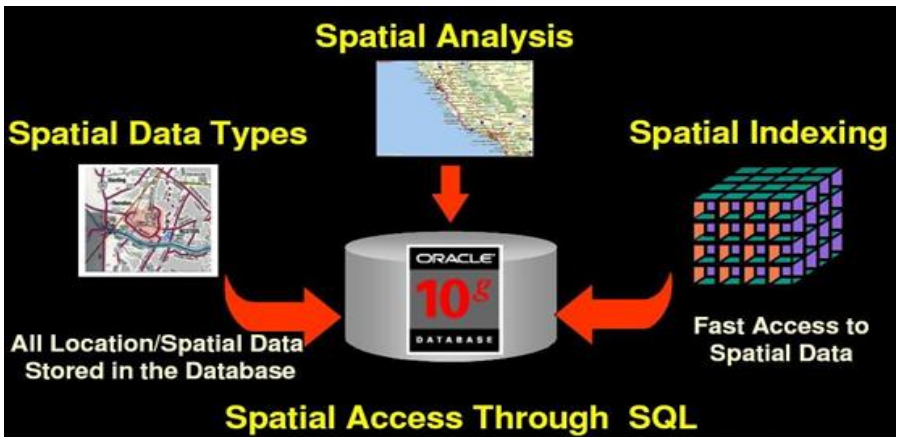


Figure 1.4: Spatial Database in Oracle [20]

The types of spatial data that can be stored using Oracle Spatial include geographic information system (GIS) data and data from computer-aided design (CAD) and computer-aided manufacturing (CAM) systems. Instead of operating on objects on a geographic scale, CAD and CAM systems work on a smaller scale such as an automobile engine, or a much smaller scale, such as printed circuit boards. The spatial data of GIS,CAD and CAM is in shown following Figure 1.5.

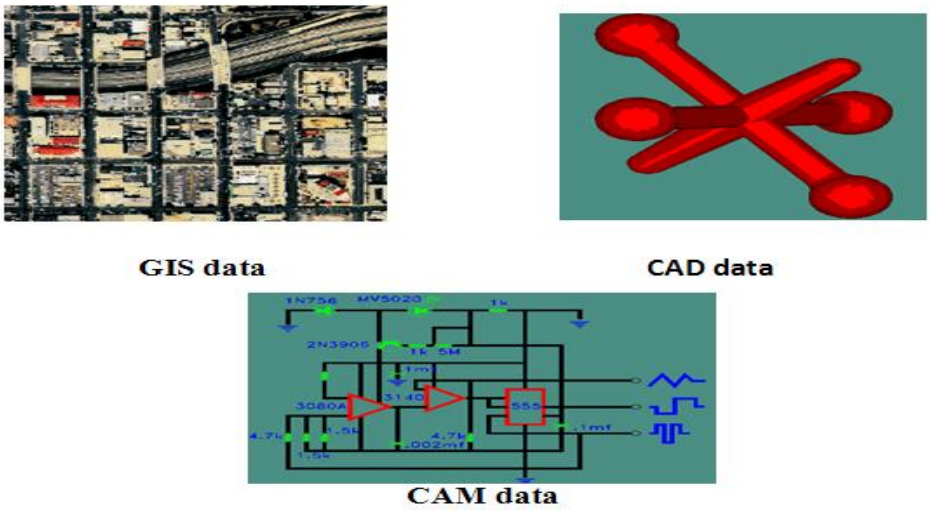


Figure 1.5: Spatial data in GIS,CAD and CAM [19]

1.6.1 Spatial Data in Various Industries

Spatial data in various industries [2,3] is as follows:

Banking and finance: These industries use location data for analysis of retail networks and for market intelligence. The customer database combined with demographics and wealth information helps banks define an optimal retail network and define the best product mix to offer at each branch.

Telecommunications: Location analysis helps telecom operators and carriers improve their competitive position. Spatial data is used for network planning, site location, maintenance organization, call center and customer support, marketing, and engineering.

Local and central government: Spatial information is heavily used by all government agencies, since they manage a multitude of assets distributed over large territories. Uses include natural resource management or land-use planning, road maintenance, housing stock maintenance, emergency management, and social services.

Law Enforcement: Spatial information helps officers in operational duties, as well as in crime analysis and prevention. Location information is used by field officers to locate places and other resources in the field in real time. Investigators use spatial data for crime analysis. Spatial patterns of crime are used to better locate police resources and improve prevention.

Real estate and property management: Geographic data and demographics are used to identify and assess locations for outlets, housing, or facilities. Land-use, transport, and utility networks are used to site industrial and production facilities.

Retail: Location data serves as a basis for operational and strategic decisions. It can be used to identify the profile of the best customers and help reach similar prospects. Spatial data can increase the relevance and focus of marketing campaigns and find the best layout of a distribution network for maximum profit.

Utilities: Many different utility systems can be found under almost every street. Utility companies use spatial information to design these underground systems, plan and monitor groundwork, and maintain their cable and pipe networks.

Communications, media, and advertising: Location data are frequently used for increasing the return of communications campaigns. Segmentation and location-based targeting help companies finesse the timing and appropriateness of marketing campaigns, thereby increasing their expectation of success.

Wireless data services: Wireless data services increasingly use location data to enrich the user experience and provide valuable services. Uses include personal navigation systems, friend finders, roadside emergency, location-based yellow page searches, and the like. Wireless location services are necessary for fast returns on investments made on third-generation telecom networks.

1.6.2 Advantages of using Oracle Spatial

Oracle spatial advantages are as follows:

- It eliminates the need for dual architectures, because all data can be stored in the same way. Unified data storage means that all types of data (text, maps, and multimedia) are stored together, instead of each type being stored separately.
- It uses SQL, a standard language for accessing relational databases, thus removing the need for specific languages to handle spatial data.
- It defines the SDO_GEOMETRY data type, which is essentially equivalent to the spatial types in the OGC and SQL/MM standards.
- It implements SQL/MM "well-known" formats for specifying spatial data. This implies that any solution that adheres to the SQL/MM specifications can easily store the data in Oracle Spatial, and vice versa, without the need for third-party converters.
- It is the de facto standard for storing and accessing data in Oracle and is fully supported by the world's leading geospatial data, tools, and applications vendors, including NAVTEQ, Tele Atlas, Digital Globe, 1Spatial, Autodesk, Bentley, eSpatial, ESRI, GE Energy/Smallworld, Intergraph,Leica Geosystems, Manifold, PCI Geomatics, Pitney/Bowes/MapInfo, Safe Software, Skyline, and many others.
- It provides scalability, integrity, security, recoverability, and advanced user management features for handling spatial data that are the norm in Oracle databases but are not necessarily so in other spatial management tools.
- It removes the need for separate organizations to maintain a spatial data

infrastructure (hardware, software, support, and so on), and it eliminates the need for specific tools and skills for operating spatial data.

- Through the application server, it allows almost any application to benefit from the availability of spatial information and intelligence, reducing the costs and complexity of spatial applications.

1.6.3 Operations on Spatial Data in Oracle

We can use following operations on spatial data in Oracle are as follows:

- Storage data model using the SDO_GEOMETRY data type
- Query and analysis using the Index Engine and Geometry Engine
- Location-enabling using the geocoder by converting address data into SDO_GEOMETRY data
- Visualization using MapViewer and Oracle Maps
- Advanced Spatial Engine functionality such as network analysis and routing

In the Lite edition of Oracle Database Server, none of the spatial functionality is included. In the Personal Edition, Standard Edition, Express Edition, and Enterprise Edition, Oracle Locator get freely in all those editions. But in the Personal Edition and the Enterprise Edition, the full functionality of Spatial technology is available as a priced option, called Oracle Spatial.

1.6.4 Difference between Oracle Locator and Oracle Spatial

The features of Oracle Locator and Oracle Spatial are described in the following Table 1.2.

Oracle Locator	Oracle Spatial
All geometric objects • points, lines, polygons • 2D, 3D, 4D. Indexing [17] Spatial queries Proximity queries Distances Projections	All features of Oracle included Geometric Transformations Spatial Aggregations Network Modeling Topology Raster Spatial Data Mining Geocoder,3D types Web Services

Table 1.2: Oracle Locator and Oracle Spatial Features

1.7 Managing and Analyzing Spatial Data

Storage of spatial data: The spatial database system could have a geometry type to store spatial information as points, lines, polygons, and other types of vector representations. The system may also have a network type for modeling road networks.

Spatial analysis on vector type of spatial data is as follows:

- *Within-distance:* This operation identifies all spatial data within a specified distance of a query location.
- *Contains:* This operation identifies all spatial data that contain a specified query location (geometry). Functions to detect other types of relationships may also be defined.
- *Nearest-neighbour:* This operation identifies all spatial data closest to a query location.
- *Distance:* This operation computes the distance between two spatial objects.
- *Buffer:* This operation constructs buffer zones around spatial data.
- *Overlay:* this operation overlays different layer of spatial data.
- *Visualization:* This operation presents spatial data using maps.

Spatial analysis on network type of spatial data is as follows:

Most spatial data such as road networks can also be represented as network data (in addition to vector data). We can perform the preceding analysis on such data using network proximity rather than spatial proximity.

1.8 Visualizing the Spatial Data in Oracle Spatial

We can visual the spatial data by using the MapViewer [7]. The MapViewer is a component of oracle application server.

The MapViewer is a server-side component that constructs maps by reading appropriate database views and tables and returns the maps to the client applications in the appropriate formats. Each map constructed is specified using one or more layers, or *themes*. Each theme represents a logical grouping of geographic spatial

features, such as roads, customer locations, rivers, and so on. These features are rendered with specific *styles.* The example of MapViewer is shown in Figure 1.6

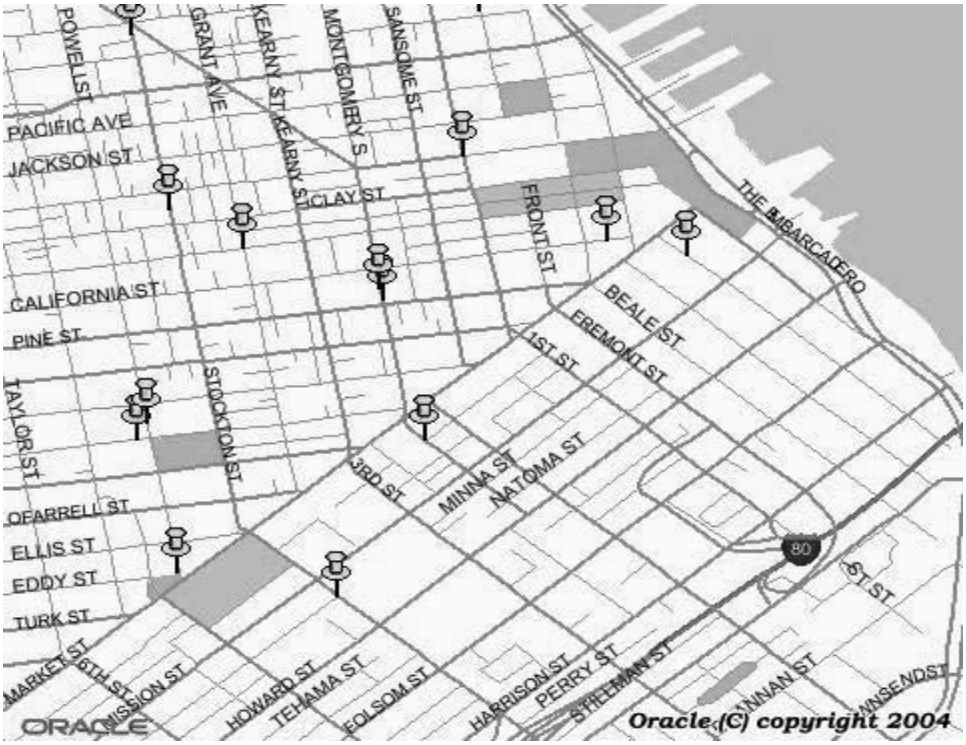


Figure 1.6: Map showing the positions of branches [3]

Chapter 2

Storage of Spatial Data Using SDO_GEOMETRY Object

We can store and model the different types of location information using the SDO_GEOMETRY [3]. To store the spatial data in the database we can use the SDO_GEOMETRY object in the table column. This object is provided by the Oracle software to store the spatial data in the database. By using the SDO_GEOMETRY object, we can store various types of simple geometries and complex geometries in the database. Simple geometries are point, line string, polygon *etc.*, and complex geometries are multipoint, multiline, multi polygon, composite solid, collection, *etc.*, Following are examples of the simple geometries and complex geometries.

- A point, which can be used to store the coordinate location of, for example, a customer site, a store location, a delivery address, and so on.
- A line string, which can be used to store the location and shape of a road segment.
- A polygon, which can be used to store city boundaries, business regions, and so on.
- Complex geometries, such as multiple polygons, which can be used to store boundaries for state such as Andhra Pradesh, Uttar Pradesh, Madhya Pradesh.

2.1 Use of Geometry Objects in Oracle

To store the simple geometries and complex geometries in the database, first we have to know the structure of SDO_GEOMETRY, including the different attributes and the values. We can create a table by using SDO_GEOMETRY object as data type as follows

```
SQL> CREATE TABLE geometry_examples

(name          VARCHAR2 (100),
description    VARCHAR2 (100),
geom           SDO_GEOMETRY );
```

The geometry_examples table contains a description of the name,a description of the geometry and the corresponding SDO_GEOMETRY object.

2.2 Types of Spatial Geometries in Oracle

There are two types of spatial geometries in oracle. These are categorized in 2D and 3D and are illustrated in the Figure 2.1.

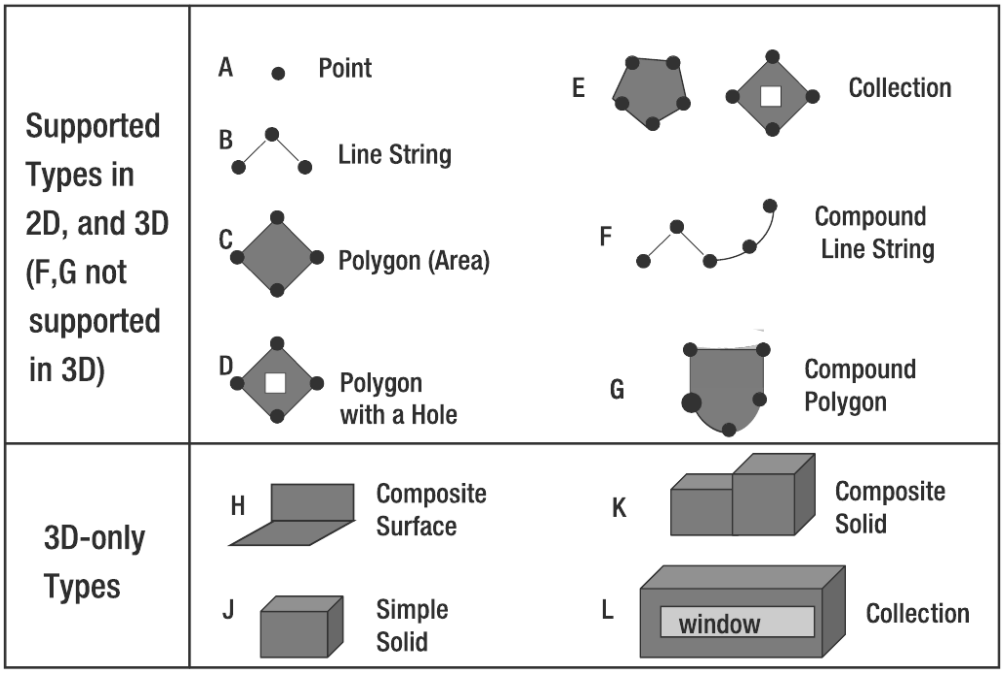


Figure 2.1: Examples of spatial data that SDO_GEOMETRY can represent [3]

2.3 SDO_GEOMETRY Type, Attributes, and Values

Oracle provided SDO_GEOMETRY to store the spatial data in the database. The structure of the SDO_GEOMETRY is as follows.

```
SQL> DESCRIBE SDO_GEOMETRY
Name                 Null?    Type
-------------------------- -------- --------------------
SDO_GTYPE                    NUMBER
SDO_SRID                     NUMBER
SDO_POINT                    SDO_POINT_TYPE
SDO_ELEM_INFO                SDO_ELEM_INFO_ARRAY
SDO_ORDINATES                SDO_ORDINATE_ARRAY
```

The purpose of each attribute served in the SDO_GEOMETRY is as follows:

- The SDO_GTYPE attribute specifies the type of shape (point, line, polygon, collection, multi- point, multiline, or multipolygon) that the geometry actually represents. Although the SDO_GTYPE attribute captures what type of geometry is being represented, it does not specify the actual coordinates.
- The SDO_SRID attribute specifies the ID of the spatial reference system (coordinate system) in which the location/shape of the geometry is specified.

We can specify the coordinates of the elements in one of the following ways:

- If the geometry is a point (for example, the location of customers), then we can store the coordinates in the SDO_POINT attribute of SDO_GEOMETRY.
- If the geometry is an arbitrary shape (for example, a street network or city boundaries), then we can store the coordinates using the SDO_ORDINATES and SDO_ELEM_INFO array attributes.
  - The SDO_ORDINATES attribute stores the coordinates of all elements of the geometry.
  - The SDO_ELEM_INFO attribute specifies where in the SDO_ORDINATES array a new element starts, how it is connected (by straight lines or arcs) and whether it is a point, a line, or a polygon.

2.3.1 SDO_GTYPE Attribute

This attribute describes the type of geometric shape modeled in the object. It has a distinct value to indicate whether the geometry is a point, a line string, a polygon, a multipoint, a multipolygon, a multiline, or an arbitrary collection. The geometry object may itself be a combination of multiple elements, each of a different shape. But this attribute specifies the general type for the entire object (with all elements it is composed of). The SDO_GTYPE attribute is a four-digit number *i.e.,* D00T. The first and the last digits take different values based on the dimensionality and shape of the geometry, as described in Table 2.1. The second and third digits are always set to 0.

Digit	Values
D (dimension of the geometry)	2=Two-dimensional, 3=Three dimensional, 4 = Four-dimensional
T (shape/type of the geometry)	0 = Uninterpreted type, 1 = Point, 2 = Line, 3 = Polygon/surface, 4 = Collection, 5 = Multipoint, 6 = Multiline, 7 = Multipolygon/multisurface, 8 = Solid, 9 = Multisolid

Table 2.1: SDO_GTYPE values for different Geometries

The D in the D00T representation of the SDO_GTYPE is used to store the dimensionality of (each vertex in the shape of) the geometry object. Spatial can work with two to four dimensional geometries. If the geometry is 2-dimensional, then it has two ordinates for each vertex in the geometric shape. If the geometry is 3-dimensional, then each vertex has three ordinates, and so on. These ordinates for vertices of the geometry are stored in the SDO_ORDINATES (or SDO_POINT) attribute.

The value of T is 2, if the geometry represents a line string. This line could be a simple line connecting any number of points by straight lines or arcs. Alternatively, this line could be a combination of multiple parts specifying straight-line segments and arc segments. Note that the line is still contiguous. If the geometry consists of multiple line segments that are not connected, then the type is 6 (multiline). For objects B and F in Figure 2.1, the value of T is 2, and SDO_GTYPE is 2002.

The type T is 3, if the geometry represents an area bounded by a closed line string (also referred to as ring) of edges. The boundary may be connected by lines, arcs, or a combination of both. The polygon can contain one or more inner rings called voids. In such cases, the area of the polygon is computed by subtracting the areas of the voids.

The area covered by a geometry that has T equal to 3 should be contiguous. Objects C, D, and G are examples. Note that object D has one outer ring and one inner ring (rectangle), but there is still only one single "contiguous" area shown by the shaded region. So, this is considered a single polygon with type T set to 3.

If there is more than one (nonvoid) polygon in the geometry (that is, if the area of the geometry is not contiguous), then it is a multipolygon geometry and the type is 7. Object E in Figure 2.1 is an example of this. If the geometry is a collection of points, lines, and/or polygons, the geometry is collection geometry. The value of T for this geometry is 4. For object E, which has two polygons (one with a void), we can set the type to 7, a multipolygon. Alternatively, we can set it to the more generic description of a collection. The type T in this case will be 4.

For a linear-referenced geometry, SDO_GTYPE is structured as DL0T. The second digit L, in that case refers to the dimension number (3 or 4) to use for the measure values in a linear-referenced geometry. In some applications, the third or fourth dimension holds additional information that can be stored with each vertex of the geometry. This additional dimension may not pertain to the shape of the geometry but may specify a "measure value" that is application-related. For example, the third dimension could model the height of each vertex point in the geometry. In transportation applications, the third ordinate for each vertex in a road segment is used to store the mile marker. To denote that a dimension as a measure dimension, we can use the L digit in SDO_GTYPE. If L is set to 3, then the third dimension is the measure dimension; if L is set to 4, then the fourth dimension is treated as the measure dimension. So, Oracle Spatial does not interpret the measure dimension values by default. Oracle Spatial does provide some functions to operate on the measure dimension for specific applications.

2.3.2 SDO_SRID Attribute

This attribute specifies the coordinate system in which the data in the spatial layer is stored. The coordinate system could be one of the following:

- Geodetic: Angular coordinates, expressed in terms of "longitude, latitude" with respect to the earth's surface.

- Projected: Cartesian coordinates that result from performing a mathematical mapping from an area on the earth's surface to a plane.
- Local: Cartesian coordinate systems with no link to the earth are surface and sometimes specific to an application. These are used in CAD/CAM and other applications where the spatial data does not pertain to locations on the earth.

Different geodetic and projected coordinate systems are devised to maximize the accuracy (of distances and other spatial relationship calculations) for different parts/regions of the world. In the case of geodetic coordinate systems, we can consult the CS_SRS table for possible values by selecting rows where the WKTEXT column starts with a prefix of 'GEOGCS'.

```
SQL>--Selecting SRIDs of Geodetic Coordinate Systems
SQL> SELECT SRID FROM MDSYS.CS_SRS WHERE WKTEXT LIKE
'GEOGCS%';
```

We can select the SRIDs for the projected coordinate system from the MDSYS.CS _SRS table by searching for rows where the WKTEXT column starts with 'PROJCS'. Analogously, we can find the SRIDs for local coordinate systems by searching for the prefix 'LOCAL_CS' in the WKTEXT column of the MDSYS.CS_SRS table.

```
SQL>-- Selecting SRIDs of Projected Coordinate Systems
SQL> SELECT SRID FROM MDSYS.CS_SRS WHERE WKTEXT LIKE
'PROJCS%';
```

In most cases, we don't have to choose the coordinate system. Instead, we obtain the geometry data from a third-party vendor, and the SRID is already populated in these geometries.

2.3.3 SDO_POINT Attribute

This attribute specifies the location of point geometry, such as the location of a customer. This attribute is of type SDO_POINT_TYPE, which is another object type. The structure of SDO_POINT object is as follows:

```
SQL>--SDO_POINT_TYPE Data Type
SQL> DESCRIBE SDO_POINT_TYPE
Name                  Null?    Type
--------------------- -------- -------------
 X                              NUMBER
```

Y	NUMBER
Z	NUMBER

The SDO_GTYPE for point geometry is set to D001. Consider point A in Figure 2.2, identified by coordinates X_A and Y_A representing a customer location.

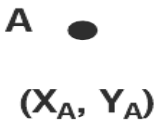


Figure 2.2: Example of a point at coordinates X_A and Y_A

Representation of point data in geometry_examples table is as follows:

```
SQL> INSERT INTO geometry_examples (name, description, geom) VALUES
(‘POINT’, '2-dimensional Point at coordinates (-79, 37) with srid set to 8307',
SDO_GEOMETRY (
2001, -- SDO_GTYPE format: D00T. Set to 2001 for a 2-dimensional point
8307, -- SDO_SRID (geodetic)
 SDO_POINT_TYPE (
-79, -- ordinate value for Longitude
37, -- ordinate value Latitude NULL -- no third dimension (only 2 dimensions)),
NULL,
NULL));
```

One row is inserted into the geometry_examples table. This point geometry is stored in the database which is two dimensional point geometry. In the insert statement, we are used the geodetic coordinate systems. Note that while insert into geodetic coordinates first we should pass the longitude then latitude ordinates.

2.3.4 SDO_ORDINATES Attribute

This attribute stores the ordinates in all dimensions of all elements of geometry. The SDO_ORDINATES attribute is of type SDO_ORDINATE_ARRAY, which we can see in the following snippet, is a collection of type VARRAY (variable-length array) of numbers. The VARRAY is useful for storing the points that describe a geometric shape in the proper order so that no explicit processing is needed when fetching that shape. If the data dimensionality is D, then every consecutive D number in the SDO_ORDINATES specifies the coordinates of a vertex. For example, if you want to model a line connecting point A that has coordinates (X_a, Y_a) with point B that has

coordinates (X_b, Y_b), then the SDO_ORDINATES will contain the numbers X_a, Y_a, X_b and Y_b, in that order. The size of this array attribute is set to 1048576. This large size limit provides enough room to store the vertices of large and complex geometries.

SQL> DESCRIBE SDO_ORDINATE_ARRAY

SDO_ORDINATE_ARRAY VARRAY (1048576) OF NUMBER

If the SDO_ORDINATES attribute specifies the ordinates (in all dimensions) of all elements of a geometry object, how are these ordinates interpreted and separated to represent different elements that make up the geometry? The information that is needed to interpret and separate the ordinates into elements is specified in the SDO_ELEM_INFO attribute.

2.3.5 SDO_ELEM_INFO Attribute

The SDO_ELEM_INFO attribute is of type SDO_ELEM_INFO_ARRAY, which is also a VARRAY of numbers with a maximum size of 1,048,576 numbers. Every three consecutive numbers in the SDO_ELEM_INFO are grouped into a descriptor triplet, describing an element or a part of an element. So, logically, the SDO_ELEM_INFO attribute is an array of triplets (three numbers). This means the size of this array attribute is always a multiple of 3. Each descriptor triplet is associated with an element of the geometry. The triplet is of the form <offset, element-type, interpretation>. The offset specifies the starting index in the SDO_ORDINATES array where the ordinates of the element are stored. The other two numbers, element-type (etype) and interpretation, take different values depending on whether the associated element represents a point, a line, or a polygon and whether the boundaries are connected by straight lines, arcs, or both. For the simple geometries contain only one element where as in the complex geometries may contain more than one element, so in such cases this attribute representation changes. Descriptor triplet can be prepared by using the following tables for the all geometries of type simple (Table 2.2) or complex (Table 2.3).

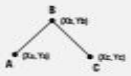
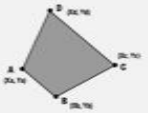
Name	Element-Type (Etype)	Interpretation	SDO_ELEM_INFO: (1, Etype, Interpretation)	SDO_ORDINATES	Illustration
Point (for example, customer location)	1	*N*, where *N* is the number of points. 1 is for a single point; >1 is for a point cluster.	(1, 1, 1)	(Xa, Ya)	
Line string (for example, streets, highways)	2	1 = Connected by straight lines 2 = Connected by arcs	(1, 2, 1) (1, 2, 2)	(Xa, Ya, Xb, Yb, Xc, Yc) (Xa, Ya, Xb, Yb, Xc, Yc)	
Polygon (for example, city boundary, buffer zone)	1003	1 = Polygon boundary connected by straight lines	(1,1003, 1)	(Xa, Ya, Xb,Yb, Xc, Yc, Xd, Yd, Xa, Ya)	
		3 = Rectangle polygon (only specify lower-left and upper-right corners)	(1, 1003, 3)	(Xa, Ya, Xc, Yc)	
		4 = Circle polygon (specify three points on boundary of circle)	(1, 1003, 4)	(Xa, Ya, Xb, Yb, Xc, Yc)	

Table 2.2: Values for SDO_ELEM_INFO (and SDO_ORDINATES) for Simple Geometries [3]

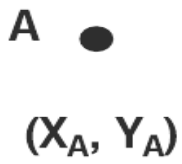
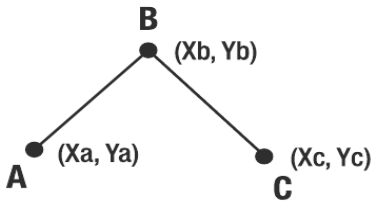
Name	Element-Type (Etype)	Interpretation
Voided polygon	1003 = Outer polygon 2003 = Interior polygon (hole)	1 = Polygon boundary connected by straight lines. 2 = Polygon boundary connected by circular arcs. 3 = Rectangle polygon. The `lower_left` and `upper_right` corner vertices of the rectangle are specified in the `SDO_ORDINATES` array. 4 = Circular polygon. Any three vertices on the boundary of the circle are specified in the `SDO_ORDINATES` array.
Compound line string	4	*N* = Specifies the number of subelements that constitute the compound line string. The *N* triplets for these *N* subelements follow the current (header) triplet.
Compound polygon	1005 = Outer polygon 2005 = Interior polygon	*N* = Specifies the number of straight-line and circular-arc subelements that constitute the polygon boundary. The *N* triplets for these *N* subelements follow this triplet.

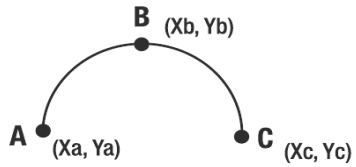
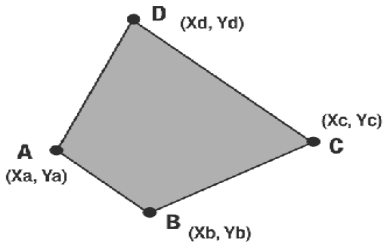
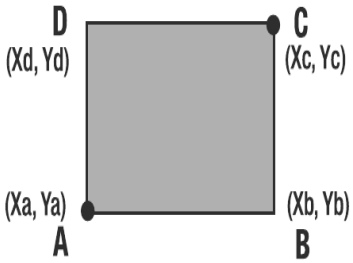
Table 2.3: Values for <Element-Type, Interpretation> in an Element Descriptor Triplet for Complex Geometries [3]

2.4 Simple Two Dimensional Geometry Examples

The simple two dimensional geometries are point, Line String Connected by Straight Lines), Line String (Connected by Arcs), Polygon(ring (boundary) Connected by Straight Lines, Polygon(ring (boundary) Connected by Arcs, Rectangle Polygon, Circle Polygon and so on.

In the following Table 2.4 shows that, for every simple two dimensional geometry there is a respective SQL insert statement.

 Point	INSERT INTO geometry_examples (name, description, geom) VALUES ('POINT', '2-dimensional Point at coordinates (-79, 37) with srid set to 8307', SDO_GEOMETRY (2001, -- SDO_GTYPE format: D00T. Set to 2001 for a 2-dimensional point 8307, -- SDO_SRID (geodetic) SDO_POINT_TYPE (-79, -- ordinate value for Longitude 37, -- ordinate value Latitude NULL -- no third dimension (only 2 dimensions)), NULL, NULL));
 Line String (Connected by Straight Lines)	INSERT INTO geometry_examples VALUES ('LINE STRING', '2-D line string connecting A(X_a=>1,Y_a=>1),B(X_b=>2, Y_b=>2), C(X_c=>2,Y_c=>1)', SDO_GEOMETRY (2002, -- SDO_GTYPE: D00T. Set to 2002 as it is a 2-dimensional line string 32774, -- SDO_SRID NULL, -- SDO_POINT_TYPE is null SDO_ELEM_INFO_ARRAY -- SDO_ELEM_INFO attribute (1, -- Offset is 1 2, -- Element-type is 2 for a LINE STRING 1 -- Interpretation is 1 if line string is connected by straight lines),SDO_ORDINATE_ARRAY -- SDO_ORDINATES attribute(1,1, -- X_a, Y_a values 2,2, -- X_b, Y_b values 2,1 -- X_c, Y_c values)));

 Line String (Connected by Arcs)	INSERT INTO geometry_examples VALUES (‘Arc STRING', '2-D Arc string connecting A(X_a=>1,Y_a=>1),B(X_b=>2, Y_b=>2), C(X_c=>2,Y_c=>1)', SDO_GEOMETRY (2002, -- SDO_GTYPE: D00T. Set to 2002 as it is a 2-dimensional line string 32774, -- SDO_SRID NULL, -- SDO_POINT_TYPE is null SDO_ELEM_INFO_ARRAY -- SDO_ELEM_INFO attribute (1, -- Offset is 1 2, -- Element-type is 2 for a LINE STRING 2 -- Interpretation is 1 if line string is connected by arcs),SDO_ORDINATE_ARRAY -- SDO_ORDINATES attribute(1,1, -- X_a, Y_a values 2,2, -- X_b, Y_b values 2,1 -- X_c, Y_c values)));
 Polygon(ring (boundary) Connected by Straight Lines	INSERT INTO geometry_examples VALUES ('POLYGON','2-D polygon connecting A(X_a, Y_a), B(X_b, Y_b), C(X_c, Y_c), D(X_d, Y_d)', SDO_GEOMETRY (2003, -- SDO_GTYPE: D00T. Set to 2003 as it is a 2-dimensional polygon 32774, -- SDO_SRID NULL, SDO_POINT_TYPE is null SDO_ELEM_INFO_ARRAY(1, -- Offset is 1 1003, -- Element-type is 1003 for an outer POLYGON element 1 -- Interpretation is 1 if boundary is connected by straight lines.), SDO_ORDINATE_ARRAY(1,1, -- X_a, Y_a values 2,-1, -- X_b, Y_b values 3,1, -- X_c, Y_c values 2,2, -- X_d, Y_d values 1,1 -- X_a , Y_a values : Repeat first vertex to close the ring)));
Polygon(ring (boundary) Connected by Arcs	We have to replace the interpretation parameter by 2 in the SDO_ELEM_INFO_ARRAY attribute of Polygon (ring (boundary) Connected by Straight Lines.
 Rectangle Polygon	INSERT INTO geometry_examples VALUES ('RECTANGLE POLYGON','2-D rectangle polygon with corner points A(X_a, Y_a), C (X_c, Y_c)',SDO_GEOMETRY(2003, -- SDO_GTYPE: D00T. Set to 2003 as it is a 2-dimensional polygon 32774, -- SDO_SRID null, -- SDO_POINT_TYPE is null SDO_ELEM_INFO_ARRAY(1, -- Offset is 1 1003, -- Element-type is 1003 for (an outer) POLYGON 3 -- Interpretation is 3 if polygon is a RECTANGLE),SDO_ORDINATE_ARRAY(1,1,-- X_a, Y_a values 2,2 -- X_c, Y_c values)));

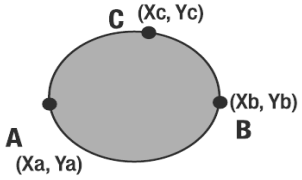
 Circle Polygon	INSERT INTO geometry_examples VALUES ('CIRCLE POLYGON','2-D circle polygon with 3 boundary points A(X_a,Y_a), B(X_b,Y_b), C(X_c,Y_c)',SDO_GEOMETRY(2003, -- SDO_GTYPE: D00T. Set to 2003 as it is a 2 dimensional polygon 32774, -- SDO_SRID NULL, -- SDO_POINT_TYPE is null SDO_ELEM_INFO_ARRAY -- SDO_ELEM_INFO(1, -- Offset is 1 1003, -- Element-type is 1003 for (an outer) POLYGON 4 -- Interpretation is 4 if polygon is a CIRCLE),SDO_ORDINATE_ARRAY(1,1, -- X_a, Y_a values 3,1, -- X_b, Y_b values 2,2 -- X_c, Yc values)));

Table 2.4: Simple two dimensional geometry examples

2.5 Complex Two Dimensional Geometry Examples

Simple geometries are composed of a simple element-an element with just one descriptor triplet. In contrast, complex geometries have more than one element descriptor triplet for an element. A complex geometry can be any of the following:

- *A compound line string or a compound polygon*: In such geometry, the boundary is connected by both straight lines and circular arcs. For instance, streets that have both straight-line segments and arcs (to denote connecting roads) can be stored as compound line string geometry. Objects F and G in Figure 2.1 are examples of such a compound line string element and compound polygon geometry, respectively.
- *A voided polygon*: This geometry has an outer ring and one or more inner rings. The outer and inner ring polygon elements are specified as simple polygon elements. Object D in Figure 2.1 is an example of voided polygon geometry. Lakes and other bodies of water that have islands can be stored as voided polygons. Note that the area of the interior rings is not considered part of these geometries.
- *A collection*: This geometry is a collection of multiple elements such as points, lines, and/or polygons. Object E in Figure 2.1, is an example of such a collection.

2.5.1 SDO_ELEM_INFO for Compound Elements

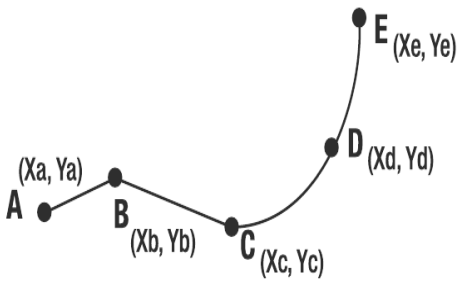
If the compound element has N sub elements, then there will be N + 1 descriptor triplets, one header triplet specifying that it is a compound element, followed by N

triplets, one for each sub element. The N sub elements have to be simple elements, and their descriptor triplets will be constructed for simple elements (above in the SDO_ELEM_INFO_ARRAY attribute). The header triplet has the following form:

- The offset specifies the starting offset for the compound element in the SDO_ORDINATES array.
- The element-type specifies one of the following:
  - A compound line string (element-type = 4).
  - A compound polygon (element-type = 1005 or 2005). The element-type will be 1005 if the compound element is used an outer polygon ring, and it will be 2005 if it is used an inner ring (void).
- The interpretation for the header triplet specifies the number of sub elements that make up this compound element.

Examples for the complex two dimensional geometries are compound Line String, Compound Polygon (Polygon with Void), Collections and so on.

The following Table 2.5 shows that for every complex two dimensional geometry there is a respective SQL insert statement.



Compound Line String

```
INSERT INTO geometry_examples
VALUES('COMPOUND LINE STRING', '2-D
Compound Line String connecting A,B, C by a line
and C, D, E by an arc', SDO_GEOMETRY(2002,
-- SDO_GTYPE: D00T. Set to 2002 as it is a 2-
dimensional Line String 32774,  -- SDO_SRID
NULL, -- SDO_POINT_TYPE is null
SDO_ELEM_INFO_ARRAY(1,-- Offset is 1 4,  --
Element-type is 4 for Compound Line String 2, --
Interpretation is 2 representing number of sub
elements 1, 2, 1,  -- Triplet for first sub element
connected by line 5, 2, 2   -- Triplet for second sub
element connected by arc; offset is 5),
SDO_ORDINATE_ARRAY( 1,1, -- Xa, Ya values
for vertex A  2,3,      -- Xb, Yb values for vertex B
3,1,      -- Xc, Yc values for vertex C 4,2,      -- Xd,
Yd values for vertex D 5,3       -- Xe, Ye values for
vertex E)));
```

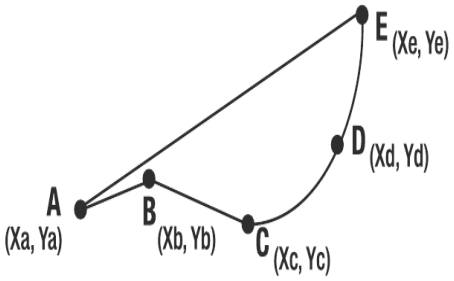
 Compound Polygon	INSERT INTO geometry_examples VALUES('COMPOUND POLYGON', '2-D Compound Polygon connecting A,B, C by a line and C, D, E by an arc(vertex E connects to A)', SDO_GEOMETRY(2003,-- SDO_GTYPE: D00T. Set to 2003 as it is a 2-dimensional polygon 32774, -- SDO_SRID NULL, -- SDO_POINT_TYPE is null SDO_ELEM_INFO_ARRAY(1,-- Offset is 1 1005, -- Element-type is 1005 for Compound polygon 3, -- Interpretation is 3 representing number of sub elements 1, 2, 1, -- Triplet for first sub element connected by line 5, 2, 2 -- Triplet for second sub element connected by arc; offset is 5 9,2,1—Triplet for third sub element; offset is 9), SDO_ORDINATE_ARRAY(1,1, -- X_a, Y_a values for vertex A 2,3, -- X_b, Y_b values for vertex B 3,1, -- X_c, Y_c values for vertex C 4,2, -- X_d, Y_d values for vertex D 5,3 -- X_e, Y_e values for vertex E 1,1 –because it is closed ring)));

Table 2.5: Complex two dimensional geometry examples

2.5.2 SDO_ELEM_INFO for Voided Polygon Element

If the voided polygon has N void (inner ring) sub elements and one outer ring sub element, then there will be at least N + 1 descriptor triplets. The first triplet will specify the descriptor triplet for the outer ring. This will be followed by descriptor triplets for each of the N void sub elements. If all the sub elements are simple elements, then there will be exactly N + 1 descriptor triplets. Otherwise, the size will reflect the descriptors for any compound sub elements. For example, the voided-polygon object D in Figure 2.1, has two descriptor triplets. The first triplet represents the outer polygon ring and has an element-type of 1003. The second triplet represents the rectangular void and has an element-type of 2003.

Polygon with a Void

The lake with island is an example for polygon with void [5] which is in shown in Figure2.3. Lake is represented by polygon and its coordinates are A, B, C and D.

Island is represented by rectangle polygon and its co-ordinates are E and F.

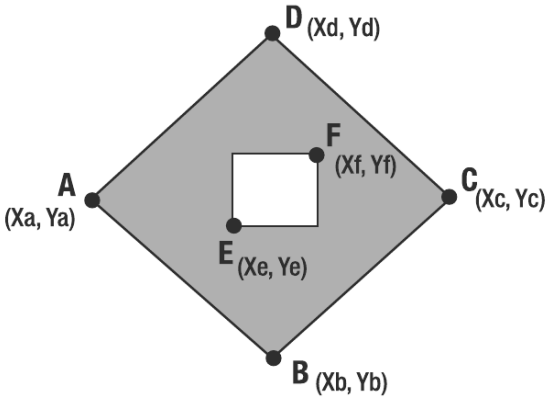


Figure 2.3: Polygon with Void

Polygon ABCD (without the void) is a simple polygon whose boundary is connected by straight lines. The constructor looks like this:

SDO_GEOMETRY (2003, 32774, NULL, SDO_ELEM_INFO (1, 1003, 1), SDO _ORDINATE_ARRAY (X_a, Y_a, X_b, Y_b, X_c, Y_c, X_d, Y_d, X_a, Y_a))

The rectangular polygon EF is not inside ABCD, the constructor looks as follows:

SDO_GEOMETRY (2003, 32774, NULL, SDO_ELEM_INFO (1, 1003, 3), SDO_ ORDINATE_ARRAY (X_e, Y_e, X_f, Y_f))

Using these two constructors, we can combine the two polygons to represent a polygon with a void as shown in Figure 2.4. In this figure, the outer element descriptor describes the outer polygon, and the inner element descriptor describes the inner polygon.

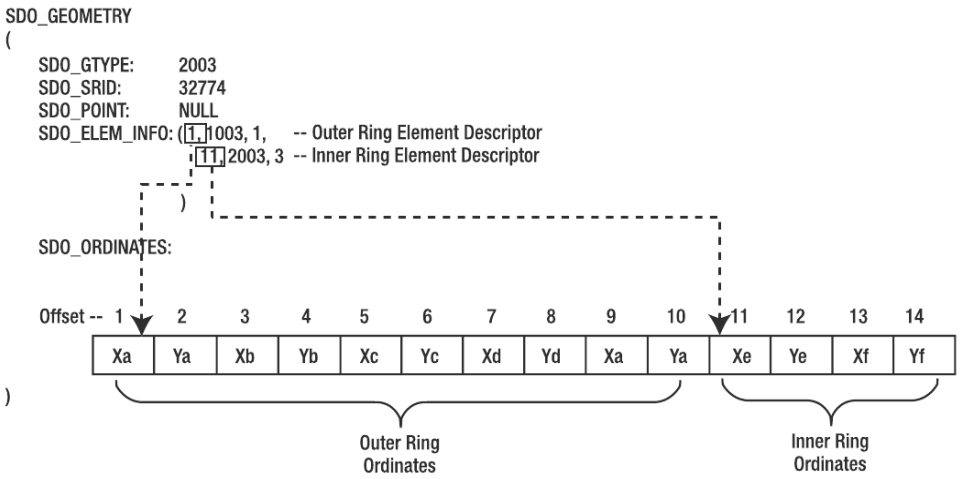


Figure 2.4: Storing Polygon with a Void as an SDO_GEOMETRY

2.5.3 Collections

Collections can be 'homogeneous', as in a multipoint, multiline, multipolygon collection. Or they can be 'heterogeneous', containing a combination of point, line, and/or polygon geometries. In Table 2.1, we saw that multipoint, multiline, multipolygon, and heterogeneous collections each have a different SDO_GTYPE. Now we will see how to represent these geometries using the SDO_GEOEMTRY data type.

Multipoint Collection Example

We had been seen the how to model a single point using the SDO_POINT attribute in the SDO_GEOMETRY type. Here we can model multiple points as a single collection geometry that is, we can store all three points A, B, and C in Figure 2.5 as sub elements of a single multipoint geometry.

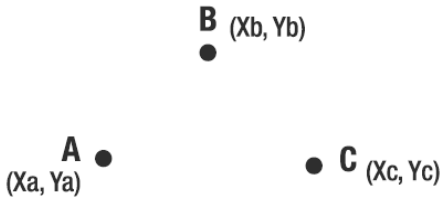


Figure 2.5: Multipoint Collection

We can represent the three points as a single element. The element will have a descriptor triplet of the form (1, 1, *N*) where *N* represents the number of points (if *N* = 1, then the element has just one point). For the Multipoint SDO_GTYPE is 2005.The corresponding constructor is as follows:

SDO_GEOMETRY (2005, 32774, NULL, SDO_ELEM_INFO_ARRAY (
1, 1, 3 -- "Point cluster" element with 3 points),SDO_ORDINATE_ARRAY(X_a, Y_a -- coordinates of first point X_b,Y_b, -coordinates of second point X_c, Y_c -- coordinates of third point)).

2.6 Metadata for Spatial Tables

Spatial treats all the objects in a single SDO_GEOMETRY column of a table as a spatial layer. For instance, the geometry objects stored in the geom column of the geometry_examples table are treated as a spatial layer. To perform validation, index

creation and querying with respect to each spatial layer (in other words, all the geometry objects in a specific SDO_GEOMETRY column of a table), we need to specify the appropriate metadata for each layer. This will include the following information.

- The number of dimensions
- The bounds for each dimension
- The tolerance for each dimension
- The coordinate system

This information for each spatial layer is populated in the USER_SDO_GEOM_ METADATA dictionary view. Structure of the USER_SDO_GEOM_METADATA is as follows:

SQL> DESCRIBE USER_SDO_GEOM_METADATA;

Name	Null?	Type
TABLE_NAME	NOT NULL	VARCHAR2 (32)
COLUMN_NAME	NOT NULL	VARCHAR2 (1024)
DIMINFO		MDSYS.SDO_DIM_ARRAY
SRID		NUMBER

USER_SDO_GEOM_METADATA [5] table describes the coordinate system and tolerance for each spatial column that a user owns. The table takes four values such as the table name that has a spatial column, the column name for the spatial data, an array describing the minimum, maximum values and tolerance value for each dimension, and a number stating the coordinate system. The null value for the coordinate system tells Oracle Spatial to use the default Cartesian coordinate system.

Chapter 3

Spatial Operators, Spatial Relationships and its Usage

3.1 Spatial Operators

The spatial operators can be used on spatial object data types. The operators can take the advantage of spatial index [17, 18] on spatial data types and require the spatial index on first geometry specified in the operator. The general syntax for spatial operator is as follows

```
<spatial_operator>
(table_geometry IN SDO_GEOMETRY,
  query_geometry IN SDO_GEOMETRY
[,parameter_string IN VARCHAR2
[, tag IN NUMBER]]
) ='TRUE';
```

- 'table_geometry' is the SDO_GEOMETRY column of the table on which the operator is applied.
- 'query_geometry' is the query location, this could be a SDO_GEOMETRY column of another table or bind variable. The query_geometry column may not require the spatial index,
- 'parameter_string' specifies the parameters specific to the spatial operators
- 'tag' specifies a number used only in specific spatial operators.

3.2 Evolution of Spatial Operators

Spatial operators [3] are evaluated in a two stage filtering mechanism involving the spatial index. As shown in Figure 3.1, a spatial operator is first evaluated using the spatial index. This evaluation using the index is referred to as the primary filter. Here, the approximations in the index (the MBRs (Minimum Bounding Rectangle) stored in the spatial index table) are used to identify a candidate set of rows that satisfies the operator relationship with respect to the query location. The identified rows are then passed through the Geometry Engine, referred to as the secondary filter, to return the correct set of rows for the specified operator. Note that all of this processing is transparent to the user, just specifying the operator in the WHERE clause of a SQL

statement will internally invoke the appropriate index (primary filter) and the Geometry Engine (secondary filter) functionality to identify the correct set of rows.

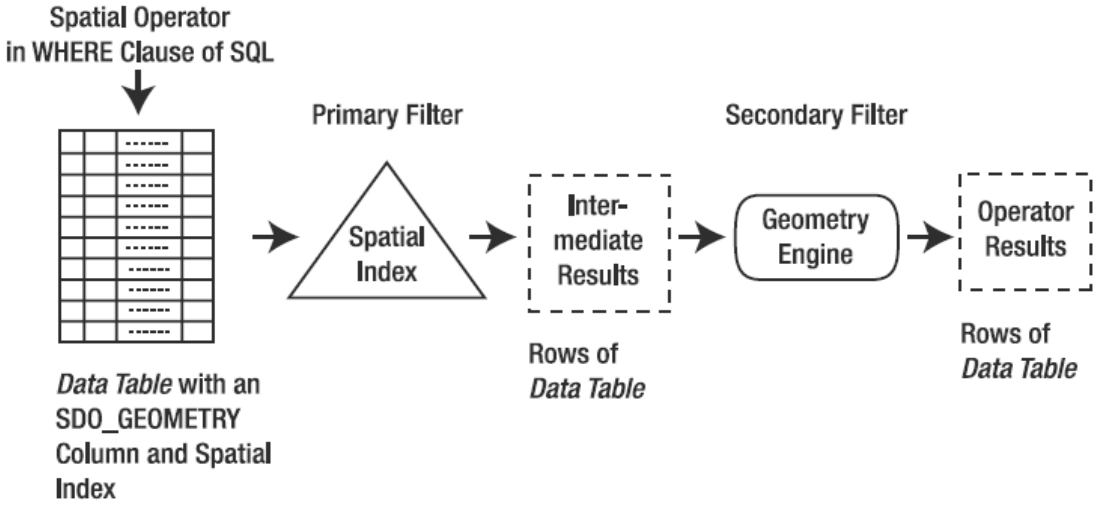


Figure 3.1: Spatial Operator evaluation using an associated spatial index [3]

The main spatial operators [4] are shown in the Table 3.1.

Operator	Description
SDO_FILTER	Specifies which geometries may interact with a given geometry.
SDO_RELATE	Determines whether or not two geometries interact in a specified way
SDO_NN	Determines the nearest neighbour geometries to a geometry
SDO_NN_DISTANCE	Returns the distance of an object returned by the SDO_NN operator
SDO_JOIN	Performs a spatial join based on one or more topological relationships.
SDO_WITHIN_DISTANCE	Determines if two geometries are within a specified distance from one another.

Table 3.1: Spatial Operators

3.3 Spatial Relationships and Filtering

Spatial uses secondary filters to determine the spatial relationship between entities in the database. The spatial relationship is based on geometry locations. The most common spatial relationships are based on topology and distance. In the four-intersection model, we will consider the boundary and interior of the geometric objects. In this model, we get the 2x2 resultant matrix. In the nine-intersection model, we will consider the boundary, interior and exterior of the geometric objects, we get the 3x3 resultant matrix [9, 10]. To determine spatial relationships, Spatial has several secondary filter methods. These are as follows:

- The SDO_RELATE operator evaluates topological criteria.
- The SDO_WITHIN_DISTANCE operator determines if two spatial objects are within a specified distance of each other.
- The SDO_NN operator identifies the nearest neighbours for a spatial object.

3.3.1 Topological Relationships

We can identify the various topological relationships [1, 19] on spatial objects. The topological relationships are as follows:

- DISJOINT -- The boundaries and interiors do not intersect.
- TOUCH -- The boundaries intersect but the interiors do not intersect.
- OVERLAPBDYDISJOINT -- The interior of one object intersects the boundary and interior of the other object, but the two boundaries do not intersect. This relationship occurs, for example, when a line originates outside a polygon and ends inside that polygon.
- OVERLAPBDYINTERSECT -- The boundaries and interiors of the two objects intersect.
- EQUAL -- The two objects have the same boundary and interior.
- CONTAINS -- The interior and boundary of one object is completely contained in the interior of the other object.
- COVERS -- The interior of one object is completely contained in the interior or the boundary of the other object and their boundaries intersect.
- INSIDE -- The opposite of CONTAINS, A INSIDE B implies B CONTAINS A.

- COVEREDBY -- The opposite of COVERS. A COVEREDBY B implies B COVERS A.
- ON -- The interior and boundary of one object is on the boundary of the other object (and the second object covers the first object). This relationship occurs, for example, when a line is on the boundary of a polygon.
- ANYINTERACT -- The objects are non-disjoint.

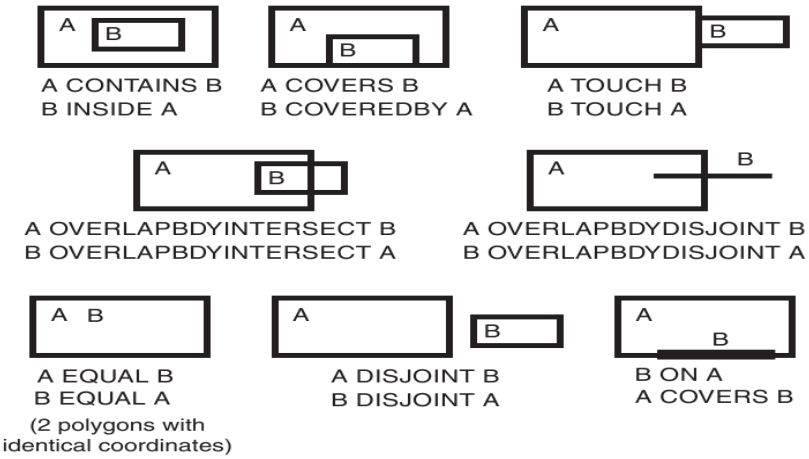


Figure 3.2: Topological Relationships of spatial objects A and B

We can use these relations in mask parameter or its combination in the spatial operator of SDO_FILTER [18].

3.4 SDO_FILTER

In a spatial R-tree index [18], each geometry is represented by its Minimum Bounding Rectangle (MBR). Consider the following layer containing several objects in Figure 3.3. Each object is labelled with its geometry name (geom_1 for the line string, geom_2 for the four-sided polygon, geom_3 for the triangular polygon, and geom_4 for the ellipse), and the MBR around each object is represented by a dashed line.

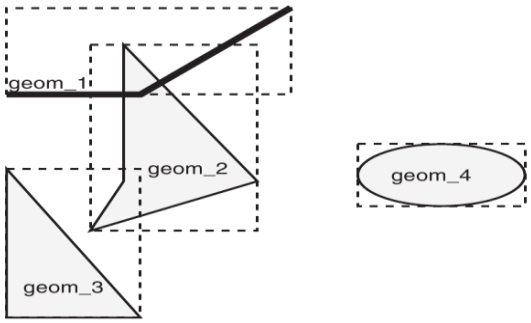


Figure 3.3: Geometries with MBRs

A typical spatial query is to request all objects that lie within a query window, that is, a defined fence or window. A dynamic query window refers to a rectangular area that is not defined in the database, but that must be defined before it is used. Figure 3.4, shows the same geometries as in Figure 3.3, but adds a query window represented by the heavy dotted-line box.

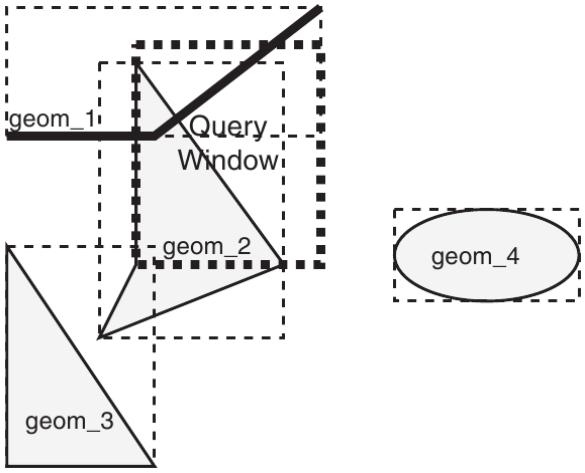


Figure 3.4: Layers with a Query Window

In Figure 3.4, the query window covers parts of geometries geom_1 and geom_2, actually geom_3 which is not covered the query window but MBR of geom_3 is covered when we mention the mask parameter is 'ANYINTERSECT' in such case we will not get the geom_3 as result. The query window does not cover any part of the geom_4 geometry or its query window.

3.4.1 Primary Filter Operator

The SDO_FILTER operator implements the primary filter portion of the two-step process involved in the Oracle Spatial query processing model. The primary filter uses the index data to determine only if a set of candidate object pairs may interact. Specifically, the primary filter checks to see if the MBRs of the candidate objects interact, not whether the objects themselves interact. The SDO_FILTER operator syntax is as follows:

SDO_FILTER
(geometry1 SDO_GEOMETRY,
geometry2 SDO_GEOMETRY)

- 'geometry1' is a column of type SDO_GEOMETRY in a table. This column must be spatially indexed.

- ‘geometry2’ is an object of type SDO_GEOMETRY. This object may or may not come from a table. If it comes from a table, it may or may not be spatially indexed.

3.4.1.1 Usage of Primary Filter

The primary filter uses geometry approximations (or index tiles) to reduce computational complexity and is considered a lower-cost filter. The primary filter can be used in three following ways.

- **Primary Filter with a Temporary Query Window**

It performs a primary filter operation without inserting the query window into a table. The window will be indexed in memory and performance will be very good. The SQL statement for Primary Filter with a Temporary Query Window is as follows:

SELECT a.Feature_ID FROM TARGET a WHERE SDO_FILTER (A.shape, SDO_GEOMETRY (2003, NULL, NULL, SDO_ELEM_INFO_ARRAY (1, 1003, 3), SDO_ORDINATE_ARRAY(x1, y1, x2, y2))) = 'TRUE';

Here, (x1, y1) and (x2, y2) are the lower-left and upper-right corners of the query window. The dashed lines of query window is represented in the Figure 3.4, if we take such a query window in this SQL statement then we get the result is geom_1, geom_2 and geom_3.

- **Primary Filter with a Transient Instance of the Query Window**

The above SQL select statement can also be represented by using of bind variable. It is represented by the following SQL select statement.

SELECT a.Feature_ID FROM TARGET a WHERE SDO_FILTER (A. shape, :theWindow) ='TRUE';

- **Primary Filter with a Stored Query Window**

Here we assumed that the query window is inserted into a table called WINDOWS, with an ID of WINS_1.The usage of the Primary Filter with stored query window is as follows:

SELECT A.Feature_ID FROM TARGET A, WINDOWS B WHERE B.ID = 'WINS_1' AND SDO_FILTER (A.shape, B.shape) = 'TRUE';

3.4.2 Secondary Filter Operator

The secondary filter applies exact computational geometry to the result set of the primary filter. These exact computations yield the exact answer to a query. The secondary filter operations are computationally more expensive, but they are applied only to the relatively small result set returned from the primary filter. This operator can be used only if a spatial index has been created on two dimensions of data.General syntax for SDO_RELATE is as follows:

SDO_RELATE
(<geometry-1>,
<geometry-2>,
'MASK=<mask>
QUERYTYPE=<querytype>
[other optional parameters]')

- 'geometry-1' is a column of type SDO_GEOMETRY in a table. The column must be spatially indexed.
- 'geometry-2' is an object of type SDO_GEOMETRY. This object may or may not from the table. If it comes from a table. It may or may not spatially indexed
- 'mask' is a parameter in this, we can use topological relationships and their combination.
- the 'querytype' parameter we can use JOIN or WINDOW which is an optional

3.4.2.1 Usage of Secondary Filter

It performs both primary and Secondary Filter operations. The secondary filter can be used in following two ways:

- **Secondary Filter Using a Temporary Query Window**

It performs both primary and secondary filter operations without inserting the query window into a table. The window will be indexed in memory and performance will be very good. The SQL statement for Secondary Filter Using a Temporary Query Window is as follows:

SELECT A.Feature_ID FROM TARGET A WHERE SDO_RELATE (A.shape, SDO_GEOMETRY (2003, NULL, NULL, SDO_ELEM_INFO_ARRAY (1, 1003, 3),SDO_ORDINATE_ARRAY(x1,y1,x2,y2)),'mask=anyinteractquerytype=WINDOW') = 'TRUE';

Here, (x1, y1) and (x2, y2) are the lower-left and upper-right corners of the query window. The dashed line of query window is represented in the Figure 3.4 if we take such a query window in this SQL statement then we get the result is geom_1, geom_2.

- **Secondary Filter Using a Stored Query Window**

Here, we assumed that the query window is inserted into a table called WINDOWS, with an ID of WINS_1.The usage of the Primary Filter with stored query window is as follows:

SELECT A.Feature_ID FROM TARGET A, WINDOWS B WHERE B.ID = 'WINS_1' AND SDO_RELATE (A.shape, B.shape, 'mask=anyinteract') = 'TRUE';

3.5 SDO_WITHIN_DISTANCE Operator

This operator performs the proximity analysis (types of proximity analysis are customer analysis and sales region analysis) in the business application domain [11]. For example, we can identify the customers within a quarter-mile radius of store site.

Given a set of locations, the SDO_WITHIN_DISTANCE operator returns all locations that are within a specified distance from a query location. Figure 3.5, shows an example of the SDO_WITHIN_DISTANCE operator specifies a distance *'d'* from the query location '*Q*'. The spatial index will retrieve the objects A, B and C that are within this specified distance *'d'*, objects D and E are eliminated, because they are farther than distances d from query location *Q*.

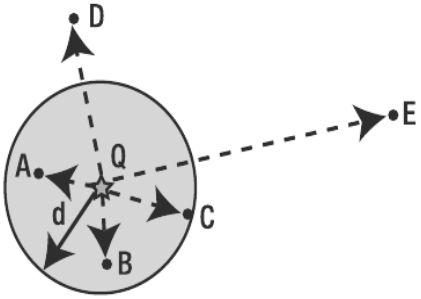


Figure 3.5: SDO_WITHIN_DISTANCE operator specifies a maximum distance'*d*'.

Syntax for the SDO_WITHIN_DISTANCE operator is as follows:

```
SDO_WITHIN_DISTANCE (
table_geom IN SDO_GEOMETRY,
query_geom        IN SDO_GEOMETRY,
 parameter_string IN VARCHAR2 ) ='TRUE'
```

- ‘table_geom’ is the SDO_GEOMETRY column of the table that is searched.
- ‘query_geom’ is the SDO_GEOMETRY specifying the query location. This could be a column of another table, a bind variable, or a dynamically constructed object.
- ‘parameter_string’ specifies the parameter distance and optionally the parameter unit (for the distance specified). The string will be of the form DISTANCE= <numericvalue> [UNIT=<string>] [min_resolution=a] [max_resolution=b]'.

3.5.1 Usage of SDO_WITHIN_DISTANCE Operator

It determines if two geometries are within a specified Euclidean distance from one another. The use of SDO_WITHIN_DISTANCE operator is illustrated with following example. *i.e.*,

To retrieving all customers within a quarter-mile radius of a competitor store, for this we have to use the following SQL select statement:

SQL> SELECT ct.id, ct.name FROM competitors comp, customers ct WHERE comp.id=1 AND SDO_WITHIN_DISTANCE (ct.location, comp.location, 'DISTANCE=0.25 UNIT=MILE ‘) ='TRUE';

3.6 SDO_NN (Nearest Neighbour Spatial Operator)

The SDO_WITHIN_DISTANCE operator is not appropriate when you need to obtain a specific number of neighbours, no matter how far they are from the query location. For these cases, the SDO_NN operator [12] is appropriate.

Figure 3.6, shows an example. A, B, C, D, and E are locations in a table that is spatially indexed. Q is a query location. The SDO_NN operator orders the items A, B, C, D, and E based on their distance to Q and returns them in the order of distance. If only one neighbour is requested, then A is returned. If two neighbours are requested, A and B are returned.

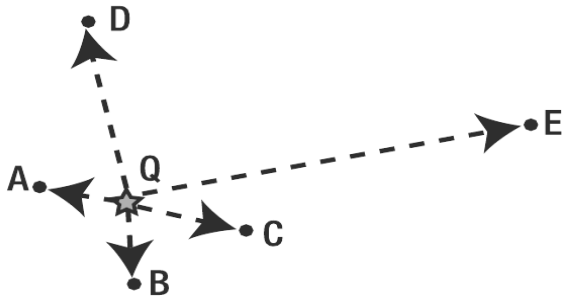


Figure 3.6: SDO_NN on five locations: A, B, C, D, and E

General syntax for the SDO_NN operator is as follows

```
SDO_NN (
table_geometry       IN SDO_GEOMETRY,
query_geometry       IN SDO_GEOMETRY
[, parameter_string IN VARCHAR2
[, tag               IN NUMBER]] ) ='TRUE'
```

- ‘table_geom’ specifies the SDO_GEOMETRY column of the table whose spatial index is to be used.
- ‘query_geom’ specifies the SDO_GEOMETRY for the query location. This could be a column of another table or a bind variable.
- ‘parameter_string’ is an optional argument, specifies one of two tuning parameters, SDO_BATCH_ SIZE or SDO_NUM_RES. We discuss these parameters in the next sections.
- ‘tag’ is another optional argument, allows the SDO_NN operator to be bound to an ancillary distance operator. We discuss it in the later part of this section. Note that this tag can be specified only if parameter_string is specified.

3.6.1 Usage of SDO_NN operator

It determines the nearest neighbour geometries to geometry [16]. The use of SDO_NN operator is illustrated with following example. *i.e.,*

To retrieving the five nearest customers to a specific competitor, for that we have to use the following SQL select statement:

```
SQL> SELECT ct.id, ct.name, ct.customer_grade FROM competitors comp, customers ct WHERE comp.id=1 AND SDO_NN (ct.location, comp.location) ='TRUE' AND ROWNUM<=5 ORDER BY ct.id;
```

Here we selecting the only the five nearest customers to competitor id is 1. Another way to writing the following SQL select statement by using the parameter SDO_NUM_RES.

```
SQL> SELECT ct.id, ct.name, ct.customer_grade FROM competitors comp, customers ct WHERE comp.id=1 AND SDO_NN (ct.location,  comp.location, 'SDO_NUM_RES=5')='TRUE' ;
```

3.7 SDO_NN_DISTANCE

Returns the distance of an object returned by the SDO_NN operator. Valid only within a call to the SDO_NN operator. Syntax for the SDO_NN_DISTANCE is as follows:

SDO_NN_DISTANCE (number)

SDO_NN_DISTANCE [11] is an ancillary operator to the SDO_NN operator. It returns the distance between the specified geometry and a nearest neighbour object. This distance is passed as ancillary data to the SDO_NN operator. We can choose any arbitrary number for the number parameter. The only requirement is that it must match the last parameter in the call to the SDO_NN operator. Use a bind variable to store and operate on the distance value.

The following example finds the two objects from the SHAPE column in the COLA_ MARKETS table that are nearest to a specified point (10,7), and it finds the distance between each object and the point.

SELECT /*+ INDEX (c cola_spatial_idx) */ c.mkt_id, c.name, SDO_NN_DISTANCE(1) dist FROM cola_markets c WHERE SDO_NN(c.shape, SDO_GEOMETRY(2001,NULL,SDO_POINT_TYPE(10,7,NULL), NULL, NULL), 'SDO_NUM_RES=2', 1) = 'TRUE' ORDER BY dist;

3.8 SDO_JOIN

Performs a spatial join [15] based on one or more topological relationships. Syntax for the SDO_JOIN is as follows:

SDO_JOIN (table_name1, column_name1, table_name2, column_name2,
params, preserve_join_order)

It returns SDO_ROWIDSET.

SDO_JOIN returns an object of SDO_ROWIDSET, which consists of a table of objects of SDO_ROWIDPAIR. Oracle Spatial defines the type SDO_ROWIDSET is as follows:

CREATE TYPE sdo_rowidset as TABLE OF sdo_rowidpair;

Oracle Spatial defines the object type SDO_ROWIDPAIR as:

```
CREATE TYPE sdo_rowidpair AS OBJECT
(rowid1 VARCHAR2 (24),     rowid2 VARCHAR2 (24));
```

In the SDO_ROWIDPAIR definition, rowid1 refers to a rowid from table_name1, and rowid2 refers to a rowid from table_name3. SDO_JOIN is technically not an operator, but a table function, note that the geometries in column_name1 and column_name2 must have the same SRID (coordinate system) value and the same number of dimensions. The following example joins the COLA_MARKETS table with itself to find, for each geometry, all other geometries that have any spatial interaction with it. In following example, rowid1 and rowid2 correspond to the names of the attributes in the SDO_ROWIDPAIR.

```
SELECT a.name, b.name FROM cola_markets a, cola_markets b,
TABLE(SDO_JOIN('COLA_MARKETS', 'SHAPE', 'COLA_MARKETS', 'SHAPE',
'mask=ANYINTERACT')) c WHERE c.rowid1 = a.rowid AND c.rowid2 = b.rowid
ORDER BY a.name;
```

Chapter 4

Spatial Functions and its Usages

Geometric processing functions are also called as spatial functions. Oracle Spatial provides functions that perform calculations on geometries, such as area of a polygon and length or perimeter of geometry. These functions can be used, for example, to determine the total area of all counties around Asia Continent, length of an interstate highway, or length of a state border. Oracle Spatial functions can also generate new geometries such as buffers, unions, intersections, and more. They can be used, for example, to define sales regions by creating a 5 mile buffer around all sales offices, to find the geometry representing the union of two sales regions, or to find the intersection between two sales regions.

The difference between spatial operators and spatial functions are that, for spatial operators we require the spatial index on SDO_GEOMETRY objects, spatial operators execution process is more time consuming than the spatial functions and spatial operators which are used in the where clause of select statement itself (SQL statement). But the spatial functions are used in select statement and where clause of select statement also. A spatial function does not require the spatial index on spatial objects. Even though the spatial functions, we can perform complex analysis than spatial operators. Some applications may require the analysis based on combination of spatial operators and spatial functions

4.1 Categories of Spatial Functions

The spatial functions are categorized as buffering functions, relationship analysis functions, geometry combination functions and geometric analysis functions.

4.1.1 Buffering Functions

The SDO_BUFFER function [13] creates a buffer around an existing SDO_GEOMETRY object. The object can be of any type point, line, polygon, or collection. In the following Figure 4.1, shows original geometry and its buffered geometry.

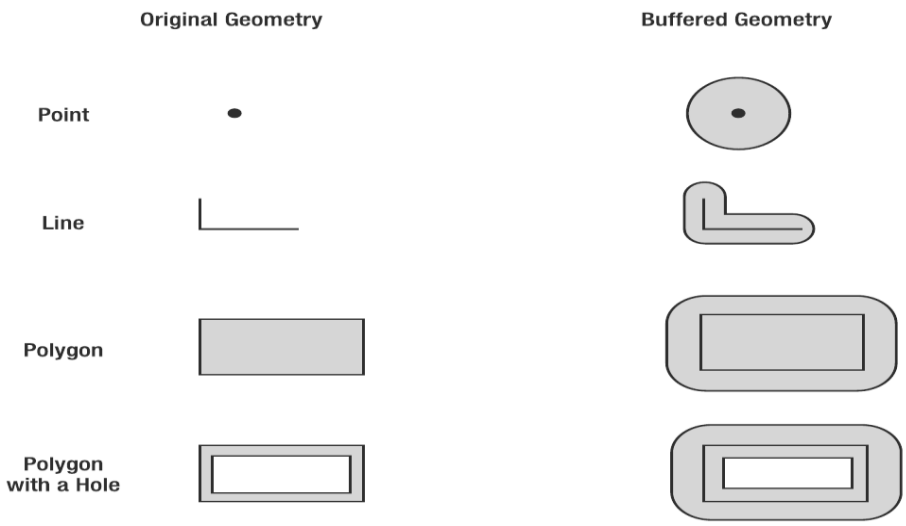


Figure 4.1: Geometric objects and buffered geometries for some simple types

Construct the buffers around the geometries by using the SDO_BUFFER spatial function. General syntax for the SDO_BUFFER is

```
SDO_BUFFER (
geometry     IN SDO_GEOMETRY,
distance     IN NUMBER,
 tolerance    IN NUMBER
[, params    IN VARCHAR2])
```

It returns a SDO_GEOMETRY.

- 'geometry' is a parameter that specifies an SDO_GEOMETRY object to be buffered.
- 'distance' is a parameter that specifies a numerical distance to buffer the input geometry.
- 'tolerance' is a parameter that specifies the tolerance.
- params is the optional fourth argument that specifies two parameters, unit=<value_string> and arc_tolerance=<value_number>.

The unit=<value_string> parameter specifies the unit in which the distance is specified. We can obtain possible values for the units by consulting the MDSYS.SDO_DIST_UNITS table. The arc_tolerance=<value_number> parameter is required if the geometry is geodetic (SDO_SRID is 8307 or 8265 for geodetic co-ordinate system). In geodetic space, arcs are not permitted. Instead, they are represented using straight-line approximations. The arc_tolerance parameter specifies the maximum distance between an arc and its straight-line approximation. Note that

arc_tolerence is always has to be greater than the tolerance for the geometry Figure 4.2, shows this arc tolerance.

Figure 4.2: Arc Tolerance

Using the SDO_BUFFER we can construct a quarter-mile buffer around the each branch location in the branches table, the corresponding SQL statement is as follows:

SQL> CREATE TABLE sales_regions AS SELECT id,
SDO_GEOM.SDO_BUFFER (b.location, 0.25, 0.5, 'arc_tolerance=0.005 unit=mile')
geom FROM branches b;

The first parameter is the geometry to be buffered. The second parameter specifies the buffer distance as 0.25. The third parameter specifies the tolerance to be 0.5 meters which is the tolerance unit for geodetic geometries. The parameter_string parameter in the fourth argument specifies the units for the buffer distance (of 0.25). In this case, the units are miles. The buffer distance, then, is 0.25 miles. Additionally, the parameter_string parameter also specifies an arc tolerance of 0.005.

We can create Buffers around Competitor Locations as follows:

SQL> CREATE TABLE COMPETITORS_SALES_REGIONS AS SELECT id,
SDO_GEOM.SDO_BUFFER (cmp.location, 0.25, 0.5, 'unit=mile
arc_tolerance=0.005') geom FROM competitors cmp;

4.1.2 Relationship Analysis Functions

These functions determine the relationships between two SDO_GEOMETRY objects [11]. For example, using these functions, we can compute the distance between a potential customer and a branch (business) location (then we can know whether the customer is within a quarter-mile from the branch location or not). Alternatively, we can determine whether a customer or a supplier is inside a specified buffer zone around a branch location or not. The first function is SDO_DISTANCE which can apply on three dimensional objects. This function determines how far apart two

geometries are. The second function is RELATE. This function determines whether two geometries interact in any specified manner or not.

SDO_DISTANCE: The SDO_DISTANCE function [3] computes the minimum distance between any two points on the two geometries. Figure 4.3, shows some examples. This distance computation takes into account both vertices and the interpolated curves of each geometry. In the line geometry example, one of the vertices of the line is closest to the second (point) geometry. In the polygon example, one of the curves is closest to the second (point) geometry.

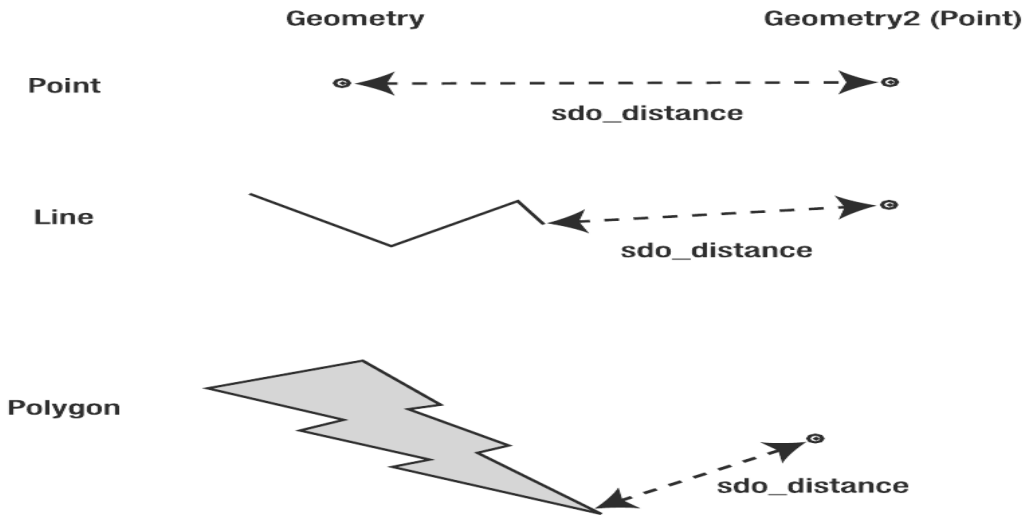


Figure 4.3: The SDO_DISTANCE function for different pairs of geometric objects [3]

The SDO_DISTANCE function has the following syntax:

```
SDO_DISTANCE (
geometry1       IN SDO_GEOMETRY,
geometry2       IN SDO_GEOMETRY,
 tolerance      IN NUMBER
[, params       IN VARCHAR2])
```

It returns a number.

- ‘geometry1’ and ‘geometry2’ are the first two arguments, and they specify SDO_GEOMETRY objects.
- ‘tolerance’ specifies the tolerance for the dataset.
- params is the optional fourth parameter in a string of the form 'unit=<value_string>'.

This specifies the units in which the distance should be returned. We can obtain possible values for the units by consulting the MDSYS.SDO_DIST_UNITS table. This function returns the minimum distance between geometry1 and geometry2 in the units specified. If no unit is specified, the default unit for the coordinate system is used (this can be determined by inspecting the SDO_SRID attribute in the SDO_GEOMETRY objects and looking at the WKTEXT attribute in the MDSYS.CS_SRS table for that SRID).

We can use the following queries by using SDO_DISTANCE

For example, to Identifying Customers within a Quarter-Mile of a Competitor Location. We have to use the following SQL select statement:
SQL> SELECT ct.id, ct.name FROM competitors comp, customers ct WHERE comp.id=1 AND SDO_GEOM.SDO_DISTANCE (ct.location, comp.location, 0.5, 'unit=mile') < 0.25 ORDER BY ct.id;

The above query can also be writing by using the SDO_WITHIN_DISTANCE spatial operator. The SDO_WITHIN_DISTANCE operator used the spatial index and the SDO_GEOM.SDO_DISTANCE function does not need to create the spatial index.

RELATE

The RELATE function determines whether two geometries interact in a specified manner or not. RELATE function is in SDO_GEOM package. For RELATE function spatial index is required but for SDO_RELATE operator, it requires spatial index. The RELATE function has the following syntax:

```
RELATE (
geometry_A     IN SDO_GEOMETRY,
 mask,              IN VARCHAR2,
geometry_Q,    IN SDO_GEOMETRY,
tolerance         IN NUMBER)
```

it returns a relationship of type VARCHAR2 datatype.

Where, ‘geometry_A’ and ‘geometry_Q’ are arguments that specify geometric objects. The mask argument can take one of the following values:

- DETERMINE, which determines the relationship or interaction, geometry_A has with geometry_Q

- The relationships are INSIDE, COVEREDBY, COVERS, CONTAINS, EQUAL, OVERLAPBDYDISJOINT, OVERLAPBDYINTERSECT, ON and TOUCH.
- ANYINTERACT if any of the preceding relationships holds
- DISJOINT if none of the preceding relationships holds

This RELATE function returns the following:

- 'TRUE' if the geometries intersect and the ANYINTERACT mask is specified
- The value of mask if geometry_A satisfies the specified mask-type relationship with geometry_Q
- 'FALSE' if the relationship between the geometries does not match the relationship specified in the second argument, mask
- The type of relationship, if the mask is set to 'DETERMINE'

We can use the following query by using RELATE function:

For example, to determine all the relationships of ‘cola_b’ with all other sales regions of colas.

SQL>SELECT c.name, SDO_GEOM.RELATE (c.shape, 'determine', c_b.shape, 0.005) relationship FROM cola_markets c, cola_markets c_b WHERE c_b.name = 'cola_b';

4.1.3 Geometry Combination Functions

These functions perform intersection, union, and other geometry combination functions on pairs of geometries [3]. We can use these functions to identify pairs of sales regions that intersect (or overlap) and find the intersection areas.

In mathematics, two sets of items, A and B, can be combined using different set-theory operations such as A minus B, A union B and A intersection B. Here, we cover similar functions that act on a pair of geometries instead of a pair of sets. If A and B are two geometries, the semantics of each of the geometry combination functions are illustrated in Figure 4.4.

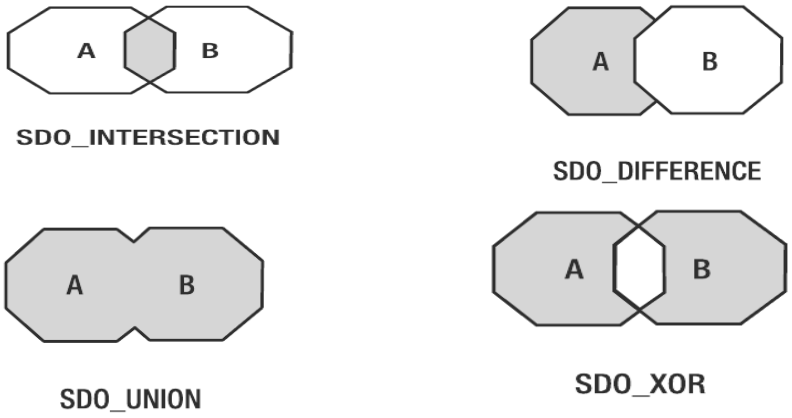


Figure 4.4: Semantics of geometry combination functions for octagon-shaped polygon geometries A and B

The values returned by each function described in Figure 4.4, are as follows:

- A SDO_INTERSECTION B: Returns the region of A that is also shared by B.
- A SDO_UNION B: Returns the region covered by A or B.
- A SDO_DIFFERENCE B: Returns the region covered by A that is not also covered by B.
- A SDO_XOR B: Returns the region of A and B that is not shared by both.

The general syntax for geometry combination functions is follows:

SDO_<set_theory_fn>
(geometry_A IN SDO_GEOMETRY,
geometry_B IN SDO_GEOMETRY,
Tolerance IN NUMBER)

It returns the SDO_GEOMETRY

- geometry_A and geometry_B are SDO_GEOMETRY objects (with the same SRIDs).
- Tolerance is the tolerance value for the geometric objects.

The function returns an SDO_GEOMETRY that computes the appropriate geometry combination function for geometry_A with respect to geometry_B.

We can use the following query by using geometry combination functions:

For example, find the intersection region of cola_a and cola_c in cola_markets table
SQL> SELECT SDO_GEOM.SDO_INTERSECTION (c_a.shape, c_c.shape, 0.005) FROM cola_markets c_a, cola_markets c_c WHERE c_a.name = 'cola_a' AND c_c.name = 'cola_c';

It returns intersection of two geometries of cola_a and cola_b. Likewise we can calculate union, difference and XOR of geometries.

4.1.4 Geometric Analysis Functions

In the geometric combination functions we examined how to construct geometries that represent the intersection, union, or difference of a pair of geometries. In the geometric analysis functions [3, 11], we describe how to perform further analysis on individual geometries. These individual geometries can be columns of existing tables, or they can be the result of other operations such as unions and intersections. For example, we can compute the area of the intersection region for each pair of overlapping sales regions. Next, we can identify the pair (of sales regions) that has the maximum area for the overlap. The pair can then be marked as a potential candidate for merging associated business units.

By using the geometric analysis functions we can calculate the area, length, or volume of an input SDO_GEOMETRY object with functions area, length and volume respectively. We can use these functions on a two-dimensional geometry or a three-dimensional geometry. These functions have the following generic syntax in PL/SQL.

```
Function_name (
geometry        IN SDO_GEOMETRY,
 tolerance      IN NUMBER
[, units_params IN VARCHAR2])
```

It returns number.

- ‘geometry’ specifies the geometry object to be analyzed.
- ‘tolerance’ specifies the tolerance to be used in this analysis.
- units_params is an optional third argument that specifies the units in which the area/length is to be returned. This argument is of the form 'unit=<value_string>'. To mention the different units for the length functions and area functions we have to consult the tables called MDSYS.SDO_DIST_UNITS and MDSYS.SDO_

AREA_UNITS respectively. There is no such table defined for volume functions (in other words, no unit conversion is performed).

The geometric analysis functions are as follows:

SDO_AREA

This function computes the area of SDO_GEOMETRY object. For example, if a rectangle object has a length of 10 units and a width of 20 units, the area would be 10 * 20 = 200 square units. Likewise, for arbitrary geometric objects, this function returns the area covered by them. The area function makes sense only for a polygon, surface, or solid (or collection) geometry. For a point or a line string, the area will always be 0. For solids, we can use the area function to calculate the surface area of the solid.

We can use the following query by using SDO_AREA function

For Example, find areas of each of the sales regions (colas) of cola_markets.

SQL>SELECT name, SDO_GEOM.SDO_AREA (shape, 0.005) FROM cola_markets;

SDO_LENGTH

This function computes the length of SDO_GEOMETRY object. For example, if a rectangle object has a length of 10 units and a width of 20 units, the length would be 2(10 + 20) = 60 units.

We can use the following query by using SDO_LENGTH function

For Example, find lengths of each of the sales regions (colas) of cola_markets.

SQL>SELECT name, SDO_GEOM.SDO_LENGTH (shape, 0.005) FROM cola_markets;

SDO_LENGTH is function one of the function in the package SDO_GEOM. By above SQL select statement, we get the length or perimeter of each geometry objects in the cola_markets but the point geometry length is zero.

SDO_VOLUME

This function takes geometry and a tolerance value and returns the volume if the input geometry is a three-dimensional solid or a multisolid geometry. For all other types of geometries, this function returns 0.

We can use the following query by using SDO_LENGTH function

For Example, to find the volume of all geometries in the city_buildings
SQL>SELECT SDO_GEOM.SDO_VOLUME (GEOM, 0.05) FROM city_buildings WHERE id=1;

The above query computes the volume (in default units of cubic feet) for building id equal to 1 in the city_buildings table. This building has dimensions of 200 feet by 200 feet by 400 feet, and hence the volume is 200 * 200 * 400 =16000000 cubic feet.

Chapter 5

Problem Statement

A Traditional Database Management System is very well able to solve those problems where the data is stored in the form of strings, numbers, and dates. But, it cannot manage the spatial data because it is more complex as compared to traditional (business) data. It has the complexity regarding the modeling of space and data types, spatial relationships and inadequacy of traditional access paths.

There are number of applications like General Location-Based Service, Customer Relationship Management, Supply Chain Management, Geographic Information Systems (GIS), Utilities, Infrastructure Energy Exploration and Distribution In-vehicle Telematics [8], which involves the use of spatial data. These applications cannot be solved by the traditional database. There is a need of database that can handle and process the spatial data and Oracle Spatial database 10g is an answer to these problems.

We have identified two problem areas for our thesis and provide the solution to these with the help of Oracle Spatial Database 10g.

The identified problem areas are as follows:

5.1 Customer Relationship Management (CRM)

The CRM is an important component in modern business and the company's growth depends on it. In any CRM database the major entities are customers, products, supplier and delivery. There are some problems in managing the CRM application with traditional databases. The modern CRM system should possess the following characteristics:

1. It should able to find area of sensitive customers which exist in those regions where our competitors are also operating. There is a need to give them special discounts and promotions to those customers, so that we can retain those customers.

2. In case of home delivery system, it should able to find out the shortest path or optimum path depending on the traffic data from delivery site to customer site, so that can fulfill the customer requirement in minimum time.
3. It should find out the closest customers from the location of a specific retail store.
4. There is a need to identify the routes from delivery sites to customer locations and cluster goods, in such a way that, the same delivery can serve multiple customers.

A traditional DBMS system cannot achieve these characteristics, because it would not be possible to store and process space, region or geographic information in it. It requires the use of a database which can handle and process spatial database.

5.2 Location Based Services (LBS)

It enables companies to extend their customer base and to analyze coverage area of their products. Consider the case of, the companies like the soft drink manufacturers where a companies are producing various types of cola products namely cola_a, cola_b, cola_c, cola_d and cola_e. Some colas products may be produced by the more than one company. To satisfy the customer, the company should maintain the quality of the products and should introduce new schemes to customers. There should have such a system which can perform the following analytical tasks:

1. The retail store wants to buy all different types of products like colas in only place. By this analysis, customers can save the money if wanted to buy the different products at one place.
2. It should able to find marketing areas of products from specified region within distance of 2 or 5 miles. If the availability of products is more in the short distance then we get chance to buy more number of products so we can save the time to buy the products and save the money to travel for that distance. If the availability of products is more in the long distance in such a case customer have to spend more money to travel long distance. We can open branch office to make it availability more in the short distance.
3. It should able to determine the relationship among marketing areas of several products with mrkt_id is equal to 1. If determined relationship is disjoint then open new branch in that region. While opening the new branch we have to

check whether it is useful or not. From that region to competitors region having the distance is more then it useful to open the new branch.

4. It should to able to find the marketing area of products such as cola_a and cola_b having no competitors.

To perform these analyses, the system will require the support of geographical area processing of marketing interest for several products like colas. Even though the geographical location is known, there is another problem of handling of the location information about the customers and marketing geographical areas/regions of the products.

Again the traditional DBMS cannot provide the solution to the above problem.

To achieve this objective, there should be some systems which can analyze the customer's location information to give better services to the customers.

5.3 Methodology

To solve the problems of location based services and space related applications, Oracle Spatial database 10g is used. It provides a SQL schema "MDSYS", where "MD" stands for "Multi Dimensional" and functions that facilitate the storage and manipulation of spatial data. It has a support of number of operators, functions, and procedures for performing area-of-interest queries, spatial join queries, and other spatial analysis operations for problems related to space and geometry.

Chapter 6

Implementation and Results

We have identified two problems related with spatial database as mentioned in the chapter 5 for our thesis work. These are as follows:

1. Customer relationship management (CRM)
2. Location based services(LBS)

6.1 Customer Relationship Management (CRM) Application

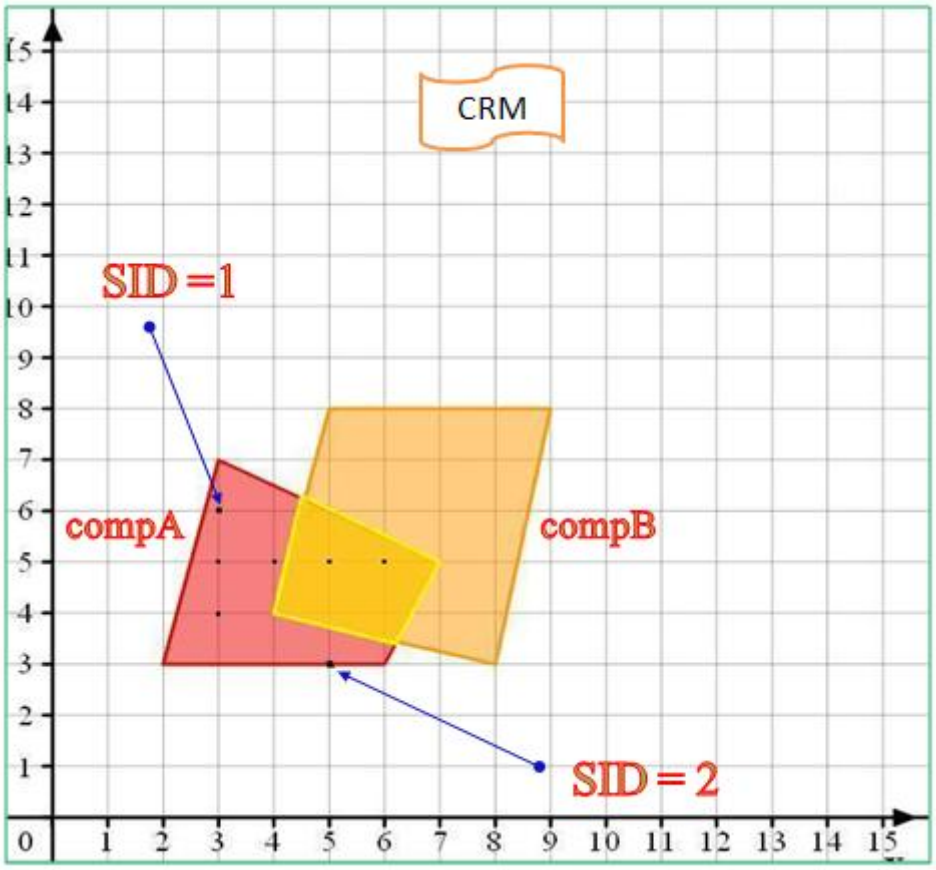


Figure 6.1: Graph representation for CRM system

As mentioned in chapter 5, (CRM problem), we consider the three entities such as COMPANIES, CUSTOMERS and STORES. Here,

1. COMPANIES having attributes like comp_name, address and region.

We consider the two companies namely compA and compB as shown in Figure 6.1. The address attribute indicates the location name of the company while region attributes shows geographical region of the company's location.

2. CUSTOMERS entity having the attributes like cid, cname, address, cust_geo_locaton, comp_name(comp_name attribute references to comp_name column in the companies attribute).

Here, we considered five customers, three of them are located in Patiala geographical area (non overlap body intersect region of companies). Two of them are located in the intersection area of both companies. Customers are represented in the shape of point. The point location shows location of the geographical coordinates of the customers (Longitude and Latitude values). The all five customers are belonging to the compA.

3. STORES entity having the attributes like sid, address, store_geo_name, comp_name (comp_name attribute references to comp_name column in the companies attribute).

Here, we consider the two retail stores. These are in the geographical area of Patiala and are represented in the shape of point. Retail stores are having address, where ever this address is located, such address has taken in the form of geographical coordinates. These retail stores are belonging to compA.

NOTE: For implementation of CRM system, we did not consider the actual geographical areas of companies, geographical locations of customers and stores are not considered. We assumed the user defined locations of customers and stores, and also assumed the user defined areas of companies.

6.1.1 Implementation of CRM System using Oracle Spatial 10g

As described in chapter 5, we have identified the following problems of CRM system. There are as follows:

1. It should be able to find area of sensitive customers which exist in those regions where our competitors are also operating.
2. In case of home delivery system, it should able to find out the shortest path or optimum path depending on the traffic data from delivery site to customer site.
3. It should be able to find the closest customers from the location of a specific retail store.

4. There is a need to identify the routes from delivery sites to customer locations and cluster goods, in such a way that, the same delivery can serve multiple customers.

In order to provide the solutions to above problems, we have created the following database:

```
SQL> create table companies
  2  (comp_name varchar2(20) primary key,
  3  address varchar2(30),
  4  region sdo_geometry);

Table created.

SQL> create table customers
  2  (cid number(4) primary key,
  3  cname varchar2(20),
  4  address varchar2(30),
  5  cust_geo_location sdo_geometry,
  6  comp_name varchar2(20) references companies(comp_name));

Table created.

SQL> create table stores
  2  (sid number primary key,
  3  address varchar2(20),
  4  store_geo_name sdo_geometry,
  5  comp_name varchar2(20) references companies(comp_name));

Table created.
```

Figure 6.2: SQL statements for creation of COMPANIES, CUSTOMERS and STORES

To store the various geometries shapes, we used SDO_GEOMETRY object. By using this object, we can store the company's geographical region, customer's location and retail store's location.

To store the data of the companies, customers and stores in the database we used the following SQL statements as shown in the Figure 6.3.

```
SQL> insert into companies values('compA','PATIALA',
  2  mdsys.sdo_geometry(2003,null,null,sdo_elem_info_array(1,1003,1),
  3  sdo_ordinate_array(2,3,6,3,7,5,3,7,2,3)));

1 row created.

SQL> insert into companies values('compB','RAJPURA',
  2  mdsys.sdo_geometry(2003,null,null,sdo_elem_info_array(1,1003,1),
  3  sdo_ordinate_array(4,4,8,3,9,8,5,8,4,4)));

1 row created.

SQL> insert into customers values(1000,'jasmeet singh','leela bhavan,patiala
',mdsys.sdo_geometry(2001,null,sdo_point_type(3,4,null),null,null),'compA');


1 row created.

SQL> insert into customers values(1001,'gurpal singh','sherawali gate,patial
a',mdsys.sdo_geometry(2001,null,sdo_point_type(3,5,null),null,null),'compA')
;

1 row created.

SQL> insert into customers values(1002,'sridhar','a/c market,patiala',mdsys.
sdo_geometry(2001,null,sdo_point_type(4,5,null),null,null),'compA');

1 row created.

SQL> insert into customers values(1003,'pavan','anardana chowk,patiala',mdsy
s.sdo_geometry(2001,null,sdo_point_type(5,5,null),null,null),'compA');

1 row created.

SQL> insert into customers values(1004,'sudesh','thripuri market,patiala',md
sys.sdo_geometry(2001,null,sdo_point_type(6,5,null),null,null),'compA');

1 row created.

SQL> insert into stores values(1,'thapar,patiala',mdsys.sdo_geometry(2001,nu
ll,sdo_point_type(3,6,null),null,null),'compA');

1 row created.

SQL> insert into stores values(2,'bus stand,patiala',mdsys.sdo_geometry(2001
,null,sdo_point_type(5,3,null),null,null),'compA');

1 row created.
```

Figure 6.3: SQL insert statements of COMPANIES, CUSTOMERS and STORES tables

In Figure 6.3, the top two SQL insert statements for companies, we had used SDO_GEOMETRY object to store geographical region of company “compA” and “compB”. The compA and compB are in shape of polygon which are represented in the Figure 6.1.The description of SDO_GEOMETRY object values for such companies are as follows:

SDO_GTYPE –2003, 2 for two dimensional and 3 for polygon shape of geometry (D00T)

SDO_SRID—spatial reference identifier is NULL (because we taken the local co-ordinates)

SDO_POINT—this value is NULL (because the shape of compA is polygon)

SDO_ELEM_INFO_ARRAY— in this value, 1 for offset, 1003 for exterior polygon and 1 for interpretation

SDO_ORDINATE_ARRAY—shape is polygon so first coordinate repeated in last also.

In case of insert in CUSTOMERS and STORES tables we have used point object, these are represented in the Figure 6.1.Description of SDO_GEOMETRY for CUSTOMERS and STORES as follows:

SDO_GTYPE –2001, 2 for two dimensional and 1 for point shape of geometry (D00T)
SDO_SRID—spatial reference identifier is NULL (because we taken the local co-ordinates)
SDO_POINT—here we have taken two dimensional co-ordinates. So, third ordinate is NULL
SDO_ELEM_INFO_ARRAY— is NULL
SDO_ORDINATE_ARRAY—is NULL

Next we have to add the information of table_name and column_name into USER_SDO_GEOM_META table it is necessary when we are using the spatial data only.USER_SDO_GEOM_METADATA describes ranges of X-axis and Y-axis from lower bound to upper bound, tolerance value and spatial reference identifier. The Figure 6.4 shows SQL insert statements of USER_SDO_GEOM_METADATA for the tables of COMPANIES, CUSTOMERS and STORES.

```
SQL> insert into user_sdo_geom_metadata values('companies','region',sdo_dim_
array(sdo_dim_element('X',0,15,0.005),sdo_dim_element('Y',0,15,0.005)),null)
;

1 row created.

SQL> insert into user_sdo_geom_metadata values('customers','cust_geo_locatio
n',sdo_dim_array(sdo_dim_element('X',0,15,0.005),sdo_dim_element('Y',0,15,0.
005)),null);

1 row created.

SQL> insert into user_sdo_geom_metadata values('stores','store_geo_name',sdo
_dim_array(sdo_dim_element('X',0,15,0.005),sdo_dim_element('Y',0,15,0.005)),
null);

1 row created.
```

Figure 6.4: SQL insert statements of USER_SDO_GEOM_METADATA

Next last step is to create the index on the spatial data. We should create the spatial index on the tables in order to perform the spatial analysis.

```
SQL> create index companies_spatial_idx on companies(region)
  2  indextype is mdsys.spatial_index;

Index created.

SQL> create index customers_spatial_idx on customers(cust_geo_location)
  2  indextype is mdsys.spatial_index;

Index created.

SQL> create index stores_spatial_idx on stores(store_geo_name)
  2  indextype is mdsys.spatial_index;

Index created.
```

Figure 6.5: Spatial index on COMPANIES, CUSTOMERS and STORES tables

6.1.2 Solutions of the CRM System

- In order to find the region of the sensitive customers which are existing in those where our competitors are operating (intersection area of compA and compB, this is area is shown with light orange color in the Figure 6.1). We have used the following SQL statement on designated database.

```
SQL> select sdo_geom.sdo_intersection(a.region,b.region,0.005)
  2  intersection_area from companies a,companies b where
  3  a.comp_name='compA'and b.comp_name='compB';

INTERSECTION_AREA(SDO_GTYPE, SDO_SRID, SDO_POINT(X, Y, Z), SDO_ELEM_INFO, SDO_OR
--------------------------------------------------------------------------------
SDO_GEOMETRY(2003, NULL, NULL, SDO_ELEM_INFO_ARRAY(1, 1003, 1), SDO_ORDINATE_ARR
AY(4.55555556, 6.22222222, 4, 4, 6.22222222, 3.44444444, 7, 5, 4.55555556, 6.222
22222))
```

Through this SQL statement, we analyzed that company had decided to give special discounts and promotions to such area of sensitive customers because to retain these customers.

- In order to find the 2 closest customers from the location of retail store ID is equal to 1. We have used the following SQL statement on designated database.

```
SQL> SELECT c.cid,c.cname,c.address FROM stores s,customers c WHERE
  2  s.sid = 1 and sdo_nn(c.cust_geo_location, s.store_geo_name,
  3  'sdo_num_res=2')='TRUE';

       CID CNAME                ADDRESS
---------- -------------------- ------------------------------
      1001 gurpal singh         sherawali gate,patiala
      1002 sridhar              a/c market,patiala
```

Through this SQL statement, we can analyze that, if retail stores are so far away from customer's location then in such a case provides delivery facilities to the customers.

- The following SQL statement shows distances from the store id is equal to 1.

```
SQL> SELECT c.cid,c.cname,c.address,sdo_nn_distance(1) distance FROM
  2  stores s,customers c WHERE s.sid = 1 and
  3  sdo_nn(c.cust_geo_location, s.store_geo_name,
  4  'sdo_num_res=3',1)='TRUE' order by distance;

       CID CNAME                ADDRESS                          DISTANCE
---------- -------------------- ------------------------------ ----------
      1001 gurpal singh         sherawali gate,patiala                  1
      1002 sridhar              a/c market,patiala             1.41421356
      1000 jasmeet singh        leela bhavan,patiala                    2
```

6.1.3 Conclusion of CRM System

With the help of oracle spatial database, we solved two in out of four problems of CRM system. To solve another two problems we have to know the concepts of "Network Modeling of Geographic Objects". Those problems are solving in future.

6.2 Location Based Services (LBS) Application

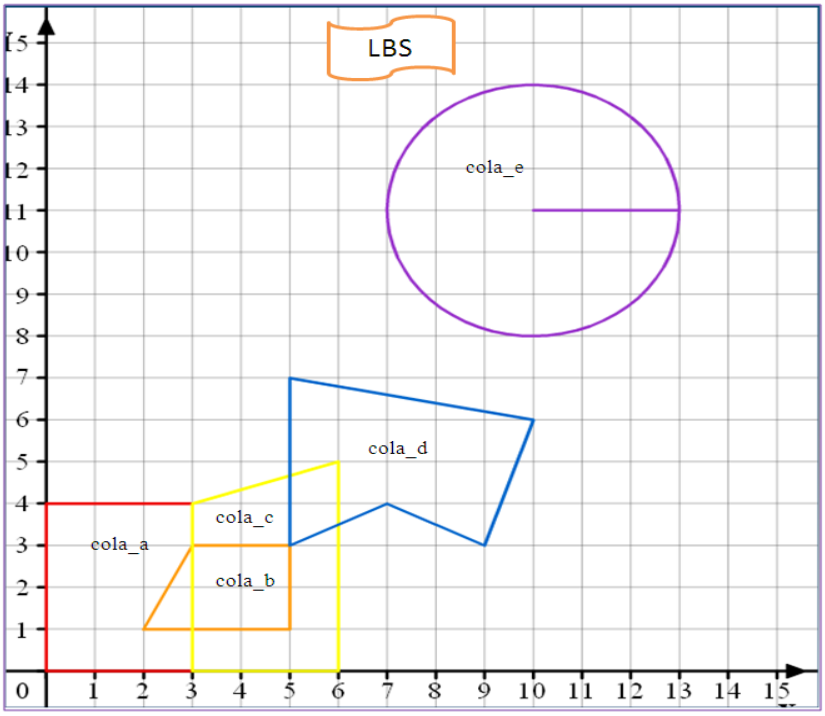


Figure 6.6: Graph representation for LBS

In LBS Application, we consider the entity is COLA_MRKTS and it has attributes like mrkt_id,geo_name and shape. Companies are producing various types of cola products namely cola_a, cola_b, cola_c, cola_d and cola_e. The cola_a is in shape of rectangle polygon,cola_e is in shape of circle polygon and remaining products are in the shape of polygon. These shapes are represented as marketing areas of colas. In

such areas, their marketing is spread. The shapes of all cola products are represented in the Figure 6.6.

6.2.1 Implementation of LBS Application using Oracle Spatial 10g

As described in chapter 5, we have identified the following problems of LBS system. There are as follows

1. The retail store wants to buy all different types of products like colas in only place.
2. It should able to find marketing areas of products from specified region within distance of 2 or 5 miles.
3. It should able to determine the relationship among marketing areas of several products with mrkt_id is equal to 1.
4. It should to able to find the marketing area of products such as cola_a and cola_b having no competitors.

To create the table COLA_MRKTS, we should use the following SQL create statement

```
SQL> create table cola_mrkts
  2  (mrkt_id number primary key,
  3  geo_name varchar2(32),
  4  shape mdsys.sdo_geometry);
Table created.
```

To store the various types of geometries we have to use SDO_GEOMETRY object. So, COLA_MRKTS table uses SDO_GEOMETRY object type to store the geometries.

The marketing area of cola_a is in the shape of rectangle polygon so, we should take lower and upper corners of the rectangle in SDO_ORDINATE_ARRAY of SDO_GEOMETRY object. The corresponding SQL insert statement is as follows:

```
SQL> insert into cola_mrkts values(1,'cola_a',
  2  mdsys.sdo_geometry(2003,--sdo_gtype 2-d polygon
  3  null,--sdo_srid is null
  4  null,--sdo_point attribute is null
  5  sdo_elem_info_array(1,1003,3),--offest 1 etype 1003 interpretation 3 for rectangle
  6  sdo_ordinate_array(0,0,3,4)))
  7  ;

1 row created.
```

The marketing area of the cola_b,cola_c and cola_d are in the shape of polygon. The corresponding SQL insert statements for the marketing area of cola_b, cola_c and cola_d as follows:

```
SQL> insert into cola_mrkts values(2,'cola_b',
  2  mdsys.sdo_geometry(2003,--sdo_gtype 2-d polygon
  3  null,--sdo_srid is null
  4  null,--sdo_point attribute is null
  5  sdo_elem_info_array(1,1003,1),--offest 1 etype 1003 interpretation 1 if connected
  6  --lines
  7  sdo_ordinate_array(2,1,5,1,5,3,3,3,2,1)));

1 row created.
```

```
SQL> insert into cola_mrkts values(3,'cola_c',
  2  mdsys.sdo_geometry(2003,--sdo_gtype 2-d ploygon
  3  null,--sdo_srid attribute is null
  4  null,--sdo_point attribute is null
  5  sdo_elem_info_array(1,1003,1),
  6  sdo_ordinate_array(3,0,6,0,6,5,3,4,3,0)));

1 row created.
```

```
SQL> insert into cola_mrkts values(4,'cola_d',
  2  mdsys.sdo_geometry(2003,--sdo_gtype 2-d ploygon
  3  null,--sdo_srid attribute is null
  4  null,--sdo_point attribute is null
  5  sdo_elem_info_array(1,1003,1),
  6  sdo_ordinate_array(5,3,7,4,9,3,10,6,5,7,5,3)));

1 row created.
```

The marketing area of cola_e is in the shape of circle polygon. So, we should take three co-ordinates of the circle. We should insert these co-ordinates in SDO_ORDINATE_ARRAY (counter clock direction). The corresponding SQL insert statement is as follows:

```
SQL> insert into cola_mrkts values(5,'cola_e',mdsys.sdo_geometry(2003,null,n
ull,sdo_elem_info_array(1,1003,4),--interpretation 4 for cicle polygon
  2  sdo_ordinate_array(10,8,13,11,10,14)));

1 row created.
```

We have to add the information of table_name and column_name into USER_SDO_GEOM_METADATA table it is necessary when we are using the spatial data only. USER_SDO_GEOM_METADATA describes ranges of X-axis and Y-axis from lower bound to upper bound, tolerance value and spatial reference identifier. The SQL insert statement of USER_SDO_GEOM_METADATA for the COLA_MRKTS table is as follows:

```
SQL> insert into user_sdo_geom_metadata values('cola_mrkts','shape',
  2  sdo_dim_array(sdo_dim_element('X',0,15,0.005),sdo_dim_element('Y',0,15,0.005)),
  3  null);

1 row created.
```

The next step is to create the index on the spatial data. We should create the spatial index on the tables in order to perform the spatial analysis. The creation of spatial index on COLA_MRKTS is as follows:

```
SQL> create index cola_spatial_idx on cola_mrkts(shape)
  2  indextype is mdsys.spatial_index;

Index created.
```

6.2.2 Solutions of LBS Application

- The retail store wants to buy all different types of products like colas in only one place. For this, we have to use overlapbdyintersect or touch spatial relationships on marketing areas of the colas. The corresponding SQL statement is as follows

```
SQL> select count(c.mrkt_id) as res,c.geo_name from cola_mrkts c,cola_mrk
ts_d where sdo_geom.relate(c.shape,'determine',d.shape,0.005) like
  2  'OVERLAPBDYINTERSECT' or sdo_geom.relate(c.shape,'determine',d.shape
,0.005)like 'TOUCH'
  3  group by c.mrkt_id,c.geo_name
  4  order by res desc;

       RES GEO_NAME
---------- --------------------------------
         3 cola_b
         3 cola_c
         2 cola_a
         2 cola_d
```

By this SQL statement, we can analyze that, we can buy the different types of products in the marketing area of cola_b and cola_c.

- The marketing areas of products from specified region within distance of 2 or 5 miles. If the availability of products is more in the short distance then we get chance to buy more number of products and we can save the time to buy the products. If the availability of products is more in the long distance in such as case we can open new branch office to make it availability more in the short distance. These can analyze by the following SQL select statements:

```
SQL> select c.mrkt_id,c.geo_name from cola_mrkts c where sdo_within_distance
(c.shape,mdsys.sdo_geometry(2003,null,null,sdo_elem_info_array(1,2003,3),sdo
_ordinate_array(7,2,9,5)),'distance=5')='TRUE';

   MRKT_ID GEO_NAME
---------- --------------------------------
         3 cola_c
         1 cola_a
         2 cola_b
         4 cola_d
         5 cola_e

SQL> select c.mrkt_id,c.geo_name from cola_mrkts c where sdo_within_distance
(c.shape,mdsys.sdo_geometry(2003,null,null,sdo_elem_info_array(1,2003,3),sdo
_ordinate_array(7,2,9,5)),'distance=2')='TRUE';

   MRKT_ID GEO_NAME
---------- --------------------------------
         3 cola_c
         2 cola_b
         4 cola_d
```

- The following SQL select statement determines the relationship among the several products with mrkt_id is equal to 1 by this result we can analyze whether open new the branch or not.

```
SQL> select c.mrkt_id,c.geo_name ,d.geo_name, sdo_geom.relate(c.shape,'deter
mine',d.shape,0.005)relationship from cola_mrkts c,cola_mrkts d where c.mrkt
_id=1'
  2  /

   MRKT_ID GEO_NAME                         GEO_NAME
---------- -------------------------------- --------------------------------

RELATIONSHIP
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------
         1 cola_a                           cola_a
EQUAL

         1 cola_a                           cola_b
OVERLAPBDYINTERSECT

         1 cola_a                           cola_c
TOUCH

         1 cola_a                           cola_d
DISJOINT

         1 cola_a                           cola_e
DISJOINT
```

- The following result shows the marketing area of products of cola_a and cola_b having no competitors. The corresponding SQL statement is as follows:

```
SQL> select sdo_geom.sdo_xor(c.shape,d.shape,0.005) XOR_area_of_colaa_col
ab from cola_mrkts c,cola_mrkts d where c.mrkt_id=1 and d.mrkt_id=2;

XOR_AREA_OF_COLAA_COLAB(SDO_GTYPE, SDO_SRID, SDO_POINT(X, Y, Z), SDO_ELEM
_INFO,
--------------------------------------------------------------------------
-------
SDO_GEOMETRY(2007, NULL, NULL, SDO_ELEM_INFO_ARRAY(1, 1003, 1, 11, 1003,
1), SDO
_ORDINATE_ARRAY(5, 3, 3, 3, 3, 1, 5, 1, 5, 3, 0, 4, 0, 0, 3, 0, 3, 1, 2,
1, 3, 3
, 3, 4, 0, 4))
```

6.2.3 Conclusion of LBS Application

With the help of oracle spatial database, we solved the problems of LBS application. By above spatial analysis, company's growth may increases and we can provide best services to customers.

Chapter 7

Conclusion and Future Scope

7.1 Conclusion

The traditional DBMS cannot handle the spatial data so that we are going for the spatial database. The Spatial database very much useful for manage and analyze spatial data. We can manage the spatial data with oracle spatial 10g.The various applications of spatial database are Urban development, Public transportation monitoring, Location-based services and Route planning. These applications can be solved by the GIS tools in the earlier days. Retrieving the information from GIS tools is very slow because no spatial indexing is used. In GIS tools are the loosely coupled system and have a poor integration between the third party tools and traditional DBMS.

In oracle spatial database have spatial operators and spatial functions features. These are used for spatial analysis. The spatial operators are useful in proximity analysis, distance based analysis and intersection based analysis. The spatial functions are used to perform the calculations on geometries such as area of polygon and length of the geometry.

In case of Customer Relationship Management (CRM) application, based on the problem statement of CRM application we can conclude the following points:

- Sensitive customers region/area can be calculated by using the spatial function, in this area of customers are applicable to special discounts and promotions of company.
- We can able to find the closest customers location from specific location so the retail store can interact with such customers.

In case of Location Based Services (LBS) application, based on the problem statement of LBS application we can conclude the following points:

- We can buy all products of companies at one place.
- If no marketing areas of products are not available from specified distance (customer's location) then we open the branch office to make it available to the customers.

- We determined the topological relationships among the spatial objects.
- We found as a sales region having no competitors between different marketing areas of products.

7.2 Future Scope

This thesis work can be extended in future on following aspects:

- We solved two in out of four problems of CRM application. To solve another two problems, we have to know the concepts of "Network Modeling of Geographic Objects".
- To visualize the spatial objects we can use MapViewer component of oracle application server 10g. In future, CRM and LBS applications can be embed the MapViewer.
- Map Builder component can be used to build own maps. In future, build maps for CRM and LBS applications.
- GUI tool can be build for CRM and LBS applications for easy usability.

References

[1] Shashi Shekar and Sanjay Chawala, "Spatial Database A Tour" Book, Prentice Hall, 2003.

[2] P. Rigaux, M. Scholl and A. Voisard," Spatial Databases with applications to GIS", Morgan Kaufmann Publishers, 2002.

[3] Ravi Kothuri, Albert Godfrind and Euro Beinat,"Pro Oracle Spatial for Oracle Database 11g", Apress publishers, 2007.

[4] "Oracle spatial users' guide and reference 10g" Release 1 (10.1), Oracle Corporation Press. http://download.oracle.com/docs/pdf/B10826_01.pdf

[5] Matt Bauer," Mapping Geometric Data with Oracle Spatial" Article. http://www.oreillynet.com/pub/a/network/2003/11/10/oracle_spatial.html

[6] Guting R.H., "An Introduction to Spatial Database Systems", Special Issue on Spatial Database Systems of the VLDB Journal, Vol. 3, No. 4, October 1994.www.cise.ufl.edu/~mschneid/Research/thesis_papers/Gue94VLDBJ.pdf

[7] Liujian (LJ) Qian, "Developing spatial applications using Oracle Spatial and MapViewer",SpatialGroup,Oracle.
www.ucgis.org/visualization/whitepapers/qia2.pdf

[8] B.Sridhar, Parteek Bhatia," Spatial Database in Oracle", National Conference, ETSNT-2009.

[9] Max J.Egenhifer,Jayant Sharma and David Mark,"A Critical Comparison of the 4- intersection and 9-intersection models for spatial relations:Formal Analysis".

[10] Jun Chen,Zhilin LI,Chengming LI,C.M. Gold,"Describing Topological Relations with Voronoi-Based 9-Intersecton Model".
www.ifp.uni-stuttgart.de/publications/commIV/chen33neu.pdf

[11] T. L. Wang and D. Shasha. Query processing for distance met- rics. Proceeding of 16th VLDB Conference, pp. 602–613, Brisbane, Australia, August 1990.

[12] N. Roussopoulos, S. Kelley, and F. Vincent. Nearest neighbor queries. Proceeding of ACM SIGMOD, pp. 71–79, San Jose, CA, May 1995.

[13] "Spatial Relationships in Spatial Analysis"
http://www.geog.ubc.ca/courses/klink/gis.notes/ncgia/u15.html

[14] “GIS and Oracle Spatial Difference” www.spatialdbadvisor.com/file_download/35/BADSIG_November_11th.pdf

[15] S. Shekhar, C. Lu, S. Chawla, and S. Ravada. “Efficient join index based join processing: a clustering approach”. IEEE Transactions on Knowledge and Data Engineering.

[16] S. Berchtold, D. A. Keim, H.-P. Kriegel, and T. Seidl. “A new technique for nearest neighbor search in high-dimensional space”. IEEE Trans. on Knowledge and Data Engineering, Vol. 12, No. 1, pp. 45–57, 2000.

[17] Ravi Kanth V Kothuri, Siva Ravada, and Daniel Abugov. Quadtree and r-trees in oracle spatial: A comparison using gis data. In Proceeding ACM SIGMOD International Conference on Management of Data, 2002.

[18] D. Papadis, T. Sellis, Y. Theodoridis, and M. Egenhofer. “Topological relations in the world of minimum bounding rectangles: a study with r-trees”. In Proceeding ACM SIGMOD International Conference on Management of Data, pp. 92–103, 1995.

[19] "Oracle Spatial10g" Ravi Kothuri, www.nyoug.org/Presentations/2003/10gspatial.pdf

[20] "Oracle Location Based Services" www.people.cis.ksu.edu/~hankley/d764/Slides08/764_23_Gudelli_OracleSpatial.ppt

List of Publications

[1] B.Sridhar, Parteek Bhatia, "Spatial Database in Oracle", National Conference, ETSNT'09, Amity University, Noida, Uttar Pradesh. April 17 and 18, 2009.

[2] B.Sridhar, Parteek Bhatia, "Spatio-Temporal Database for Handling of Moving Objects", National Conference on Impact of IT on Society Emerging Trends and Issues, DAV College, Amritsar, Punjab,2009.