ORACLE

Overview

Oracle7 Spatial Data Option™ Overview

Advances in Relational Database Technology for Spatial Data Management

April 1996

Part No. A43693–1

Oracle7 Spatial Data Option™ Overview, 7.3.2

Part No. A43693–1

This software was not developed for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It is the customer's responsibility to take all appropriate measures to ensure the safe use of such applications if the programs are used for such purposes.

ORACLE **Overview**

Oracle7 Spatial Data Option

Advances in Relational Database Technology for Spatial Data Management

Abstract

Oracle7 Spatial Data Option, a powerful extension to the Oracle® Universal Database™, breaks new ground in managing spatial data. By adding a new generic data type, extending SQL functionality, and providing an elegant methodology to operate on and maintain spatial data, Oracle7 Spatial Data Option solves two critical problems: how to efficiently and simply store, access, and manage different types of data in a single database, and how to improve performance for very large spatial databases holding gigabytes of spatial data.

Data Management and the Information Infrastructure

Over the last few years, enterprises of all kinds have radically shifted the focus of their information management technologies. For example:

- They are collecting data at an awesome and ever–accelerating rate, largely as a result of deploying increasingly powerful electronic acquisition equipment.
- They need to store and retrieve this data efficiently, and make it accessible to all users, enterprise–wide.
- They must distribute data across multiple databases and hardware platforms, in both multi–user and client/server networked environments.
- They must maintain data integrity and security.

Spatial Data Location

Spatial data is information that is described in more than one dimension. Location is inherently spatial information: location is a ”point” defined by the intersection of a latitude, longitude, and elevation. Location may have a time component as well; for example, if the information tracks the locations of delivery trucks or other moveables, such as emergency vehicles or police cars, whose physical location is time dependent.

Spatial data is location: latitude, longitude, elevation or depth, time.

It is not hard to imagine the difficulty of storing, accessing, managing, and manipulating location–based data if latitude, longitude, elevation, and time are stored as independent values. Operations that manipulate them would presume those values are not interrelated, when, of course, they are related to each other in an important way. If operations could only be done on the "points," on the locations themselves, or even on more complex spatial objects like areas, a great deal of efficiency and value can be achieved within your application.

You may automatically associate spatial data with the geographic information systems (GIS) community. The market for spatial information technology goes beyond the pure GIS market to allow geo–enabling any application. Almost any business application can benefit from being spatially–enabled. A spatially–enabled business application (SEBA) can, for example, convert address fields into geocodes and analyze the distribution of sales, customers, facilities, or suppliers. This is relevant for the retail market, manufacturing, real estate, banking, insurance, and mission–critical business applications. By turning location into a critical element in the decision making process, the Spatial Data Option is a key component of any information system.

Spatial Data: Beyond Location

Spatial data defines points in space or, more generally, locations and areas, and is an important category of information. Almost every application includes location information such as addresses. The Spatial Data Option is designed to bring the same benefits that it brings to location information to any set of data that is defined by more than one interrelated dimension. Some categories of information defined in more than one dimension are:

- demographic (age, income, number of children)
- physical (height, width, depth, weight)
- manufacturing (valve, pressure, flow rate)
- financial services (yield, price, maturity)
- stock trading (price, block size, date, time)
- retail (region, product, sales)
- consumer marketing (customer type, past purchase record, promotion)

Spatial data is any set of up to 32 interrelated values.

The benefits of spatial data are not limited to applications that rely on location–centric information. Any business application that stores and retrieves numeric data that is interrelated can be spatially organized; for example, the price, yield, and maturity of a government security. Queries to retrieve those government securities matching the required price, yield, and maturity characteristics are difficult to execute efficiently without the Spatial Data Option because you are operating on each value independently of the others when these values define a security at a point in time. With the Spatial Data Option, this query is a simple range retrieval. The Spatial Data Option can be applied to data warehouses both in its ability to encode related data and to effectively perform set operations on very large databases (VLDB).

Spatial Data: Before Oracle7 Spatial Data Option

Spatial data has previously been treated as a special category of data. It has been stored and managed in specially designed systems commonly known as geographic information systems (GIS). These systems provided the power to organize, query, and display location information visually on maps or geographic–centric interfaces. This treatment of spatial data has limited it to specialized applications and prevented it from being part of mainstream business applications. Before the availability of the Spatial Data Option, it was difficult to bridge the gap between spatial data and business data. When spatial data was only managed by specialized systems and business data was managed by RDBMS and associated application development tools, mainstream business applications were denied access to location–centric information. GIS systems were often without the data management facilities and utilities that are vital parts of mission–critical applications and VLDBs.

Oracle7 Spatial Data Option: Using the Relational Model to Store Spatial Data

Users require diverse data to be available in a single repository that can handle all types of data with the same query capabilities. They want to model spatial data in the same ways as attribute data. They want a flexible data structure, and a simple way to make changes to the data model. This data structure would not need to segregate spatial and attribute data based on differing requirements for indexing and access mechanisms.

Overview

The ideal solution is an enterprise–wide information infrastructure that includes a single database system for managing spatial data, with a data structure that is independent of the application. Furthermore, because existing datasets are immense and will certainly increase over time, the ideal solution also provides management techniques suitable for VLDBs.

Oracle Corporation has developed technology that overcomes the obstacles faced when using an RDBMS to store spatial data. Based on the powerful Oracle Universal Server, the Spatial Data Option allows you to manage all data within the relational model. This breakthrough technology employs an encoding technique which allows data to be organized spatially, opening up broad application opportunities for location–based and other spatial information.

The Spatial Data Option provides the complete integration of complex dimensional data into the world's most popular RDBMS, the Oracle Universal Server: a complete solution for managing any data – relational, spatial, decision support, text, image, video, and audio – in any application at any scale. This integration means that all the data integrity, security, and systems utilities that are part of Oracle7 are available to manage spatial data. This product brings complex spatial data into the mainstream of business application development without requiring specialized knowledge.

HHCODE Datatype

Oracle7 Spatial Data Option provides a new datatype, HHCODE, that allows application developers to encode multiple dimensions into a single unique value. This value is then stored in a single column in a spatial table. You can encode any data that is numeric and falls within a defined range.

In essence, HHCODE is a single value representing *n* dimensions that you can tailor to fit your needs–including defining dimensions that extend beyond the traditional space/time paradigm.

Once you encode your data, the Spatial Data Option groups it together in a predictable manner. It is dimensionally or spatially organized in the database, meaning that the data clusters based on its defined dimensions. This organization strategy provides rapid access to relevant data, because the Spatial Data Option immediately eliminates from consideration all data outside the dimensional boundaries of your query.

Fast, Efficient Data Access

Combining the values of all dimensions into one value significantly improves efficiency and performance over other relational methods. In the traditional RDBMS, storing spatial data in separate columns in the database limits both access and analysis capabilities.

Here is an example using two–dimensional data. Latitude and longitude of four North American cities are stored in an RDBMS table as follows:

City Name	Latitude	Longitude
Minneapolis	45 deg. lat N	93 deg. long W
Montreal	46 deg. lat N	73 deg. long W
Ottawa	45 deg. lat N	76 deg. long W
Philadelphia	40 deg. lat N	75 deg. long W

Table 1 Coordinate Values in a Relational Table

A two–dimensional proximity sort requires that the data in both the latitude and longitude columns are searched as a pair combination–not a feature of an RDBMS. In a spatial table, however, only the single column of HHCODE data (composed of latitude and longitude dimensions) needs to be searched to obtain an accurate result. Multiple–column storage limitations are even more obvious in range queries, such as the two–dimensional window query shown in Figure 1.

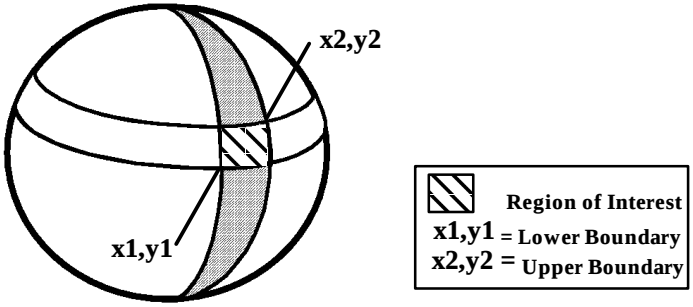

Figure 1 Window Query in an RDBMS

Even though you may be interested only in the window region defined by the two coordinate pairs, the entire range in both latitude and longitude must be searched.

In the relational model, it is not easy to represent or access more complex structures, such as two–dimensional polygons or three–dimensional areas. Because geometric operations must be performed on the complete dataset, increasing the number of dimensions to four or more compounds the search problem. HHCODE eliminates these issues.

Recursive Decomposition of Space

The HHCODE datatype is based on the principle of the recursive decomposition of space. For example, the Spatial Data Option decomposes a two–dimensional region by recursively subdividing its regions into four quadrants. This is known as a *quadtree* representation. Visualize the earth projected onto a flat surface, such as a map or a screen, and then subdivide, or grid, its surface. Figure 2 illustrates a two–dimensional quadtree decomposition of the earth; the two–dimensional surface is divided into four equal quadrants, then each of these is divided into four, and so on.

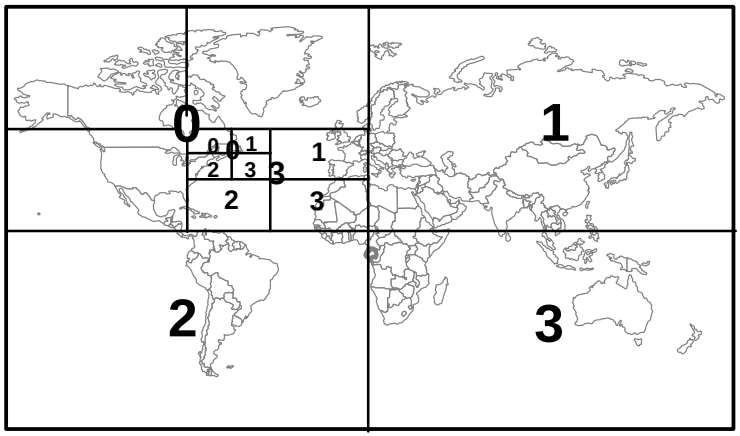

Figure 2 Decomposition of Space by Quadrants

Each quadrant is numbered to represent both its position (region of the earth) and level. Level 0 is the complete, non–decomposed region. Level 1 is the first level of decomposition, and so on. When data exists in only one of the quadrants, as in quadrant 0 of Figure 3, then only this quadrant will continue to subdivide.

When subdivision occurs, the number of the top–level, or *primary*, quadrant is appended to the number of the new quadrants created by the subdivision. Quadrant 0 in Figure 2 subdivides into quadrants 00, 01, 02, and 03, and continues to subdivide in the same manner. Figure 3 illustrates this numbering system, level by level, to four levels:

Figure 3 Levels of Decomposition

If you set the fourth level as the lowest level of subdivision, then all the data representing this region of the world will have one HHCODE value. If you want a greater level of resolution, the data will continue to subdivide until very small object spaces are represented. Each level of decomposition adds a digit to the linear string that specifies the region of space.

Data Resolution

The number of levels of subdivision is referred to as the *resolution* of the data. In our two–dimensional example, each level is a subdivision of the region by two. The top level of the earth's surface, with no decomposition, represents a 20,001.6 x 10,000.8 kilometer area. If this were represented by one HHCODE, the surface of the earth itself would be a single cell defined by these coordinates. Decomposition can continue indefinitely, providing greater and greater resolution.

Encoding Object Space Using HHCODE

The HHCODE datatype functions as an orderly breakdown of space represented as a linear string. The length of the string is determined by the resolution you define, which parallels the size of the cell, or *region*, that the HHCODE represents: longer strings encode smaller regions. This allows you to search on pattern matches (substrings of the HHCODE) and generate variable–sized cells by taking substring lengths of a single HHCODE.

The HHCODE is not a point; it is a bounded cell representing the object space in as many dimensions as you define. In two dimensions, you can visualize HHCODEs as variable–sized, interlocking four–point polygons. In three dimensions, you can visualize these polygons as cubes. Figure 4 illustrates points in two– and three–dimensional space.

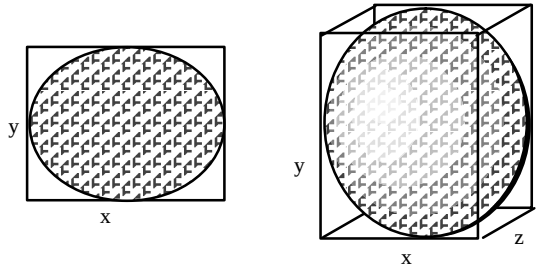

Figure 4 Points in Two– and Three–Dimensional Space

In *n*–dimensions, you can visualize an HHCODE as a multidimensional cell. Note, however, that the *x* and *y* values for a two–dimensional point will only create an equal–sided square if the data ranges for *x* and *y* are the same. In three dimensions, you can most easily visualize an HHCODE as a cube, but it is in fact a hexahedron with sides that are not necessarily equal.

When the HHCODEs are sorted, the data clusters dimensionally or spatially, and the values representing neighboring points match to a substring level. Thus, you can use HHCODEs to represent variable–sized object spaces, depending on your specific requirements for data precision, analysis, and visual display. HHCODEs can be aggregated dynamically to vary resolution. The hierarchical organization of the HHCODE allows you to vary resolution with the complexity of the object you are representing; shorter codes represent larger areas of space.

Variable–Scale Representation

When you first define an HHCODE, you set the resolution (level of detail) by specifying the scale of the data you are encoding. This precision can either reflect data–collection accuracy, or it can be scaled. The hierarchical structure of the HHCODE allows both dynamic variable–scale representations and data *generalization*.

You can obtain multiple–scale representations through data *aggregation* and *disaggregation*, which is easily accomplished by varying the length of the HHCODEs in a process called *truncation*.

Figure 5 illustrates the aggregation process using HHCODE truncation. You might want to change the scale for a variety of reasons, including:

- removing detail to reveal structural components and general shapes (generalization)
- revealing detail that might be masked by generalization
- creating homogeneous regions, such as cities, from data collected and stored at a more detailed level, such as street addresses
- fuzzy matching of data that is proximal in one dimension or in a subset of the encoded dimensions

Level 5 Resolution

HHCODEs are represented at their encoded resolution; there is no truncation. The SD*SQL function HHSUBSTR can be used to match HHCODEs to the substring level.

Level 4 Resolution

Truncation of HHCODEs to a Level 4 substring match reduces the number of HHCODEs by merging each four cell group (quadrant) into a single HHCODE.

Level 3 Resolution

This is the result of truncation of HHCODEs to a Level 3 substring match.

Level 2 Resolution

This is the result of truncation of HHCODEs to a Level 2 substring match.

Level 1 Resolution

This is the result of truncation of HHCODEs to a Level 1 substring match.

Figure 5 Multiple–Scale Representation in Two Dimensions

Primitive Data Types

It is easy to represent the primitive data types–points, lines, and regions–with HHCODE.

Points

Although the HHCODE is not a point in the sense of a zero–dimensional range representation, it can represent points very effectively; the length of the HHCODE and its level of resolution determine the ”size” of the point it represents.

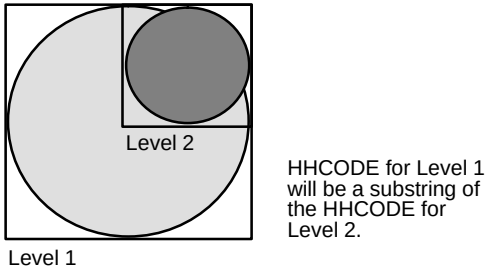

Figure 6 HHCODE of a Point in Two Dimensions

Lines

Lines in two–dimensional space are represented in an HHCODE by two end points, each with two coordinate values, as illustrated in Figure 7.

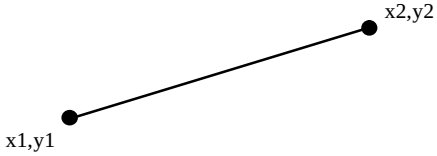

Figure 7 HHCODE Line Representation

Each of the coordinates for the start and end points of the line is encoded as a single dimension value. The line is an inferred vector between these points. With this type of line representation, you can infer several pieces of information from the HHCODE:

- the length of the line
- the line’s direction in cartographic space
- the line’s direction in a network of lines that make up a *polygon*
- polygons associated with the line

You can infer direction in a *network* based on the order of the coordinates in the HHCODE. Conceptually, a single line can occupy the same object space and belong to more than one network, but it is represented by two HHCODEs. For example, a line segment may belong to both polygon A and polygon B as illustrated by Figure 8, even though the line directions are different.

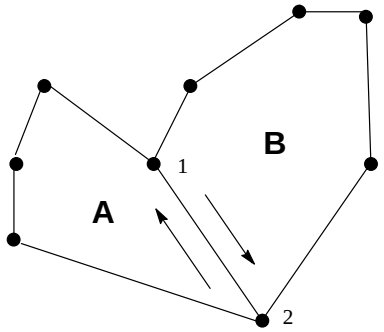

Figure 8 Line Direction in Networks and Polygons

The order of the start and end points within the HHCODE implies direction. Thus, if you wanted to define two directions for the single line shown in Figure 9, two HHCODEs would result.

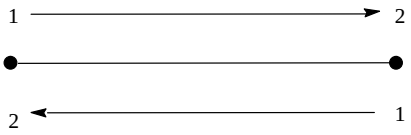

Figure 9 Bidirectional Lines in a Network

Regions

The HHCODE represents an area or region primitive in multiple dimensions. Consequently, regions are easily expressed as a single HHCODE in the same way that a point is represented. It is unlikely, however, that a single HHCODE will realistically represent a region, because the data is not likely to decompose into a single multidimensional cell. Regions, or polygons, are composed of multiple HHCODEs decomposed to the scale you specify. This can be useful for the generalization of object space. Figure 10 is a two–dimensional illustration of a polygon representation or region.

Region Represented by Vectors

Region Represented by Variable–Length HHCODEs

Figure 10 Two–Dimensional Encoding of a Region

The process of creating a polygon using HHCODEs is called *tessellating*, because smaller and smaller tiles (cells) are created to account for all the data. The polygon then consists of the set of these cells (HHCODEs).

Tessellation is useful for more than just polygon representation; the structure of both the HHCODE and spatial tables allows the dynamic grouping of HHCODEs to extract subsets of data. You can then use these subsets for independent data analysis or transfer them to another database.

Encoding Dimensions

Before your HHCODE is defined in a spatial table, you must determine the following:

- the number and types of dimensions you want to encode
- the coordinate system you will use for each dimension
- the range extent, as defined by upper and lower boundaries for each dimension
- the scale for each dimension

One–Dimensional HHCODEs

One–dimensional HHCODEs can take advantage of fast data loading, table partitioning, archiving, and space reduction. For example, zip codes can be represented as shown in Table 2.

Dimension Number	Lower Boundary	Upper Boundary	Scale
1. Zip Code	00000	99999	0

Table 2 One–Dimensional HHCODE

Two–Dimensional HHCODEs

Two–dimensional HHCODE representations can include *x* and *y* in cartographic dimensions. For example, you could define the dimensions as shown in Table 3.

Dimension Number	Lower Boundary	Upper Boundary	Scale
1. Latitude	–90 degrees	+90 degrees	7
2. Longitude	–180 degrees	+180 degrees	7

Table 3 Two–Dimensional HHCODE

Three–Dimensional HHCODEs

Figure 11 illustrates three–dimensional HHCODEs in Cartesian coordinates as *x*, *y*, and *z*.

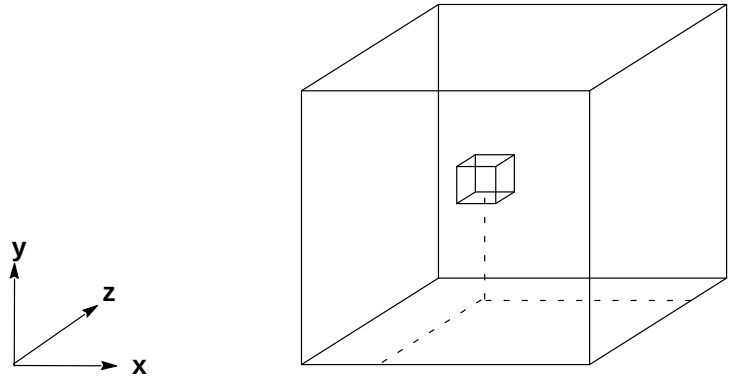

Figure 11 Three–Dimensional HHCODE

Although three–dimensional cells are often represented as cubes, or *cubic octrees*, remember that the volume created may not be equilateral on all sides, depending on the range values of each dimension. The volume is actually a hexahedron with six faces.

Using Time as a Dimension

In Figure 12, time is the third dimension in a three–dimensional object space. This creates a three–dimensional temporal key, which can represent the successive positions over time of a two–dimensional object. The HHCODE is thus a space / time key which is totally unique, because no one object can be in two places at the same time, nor can any two objects occupy the same space at the same time.

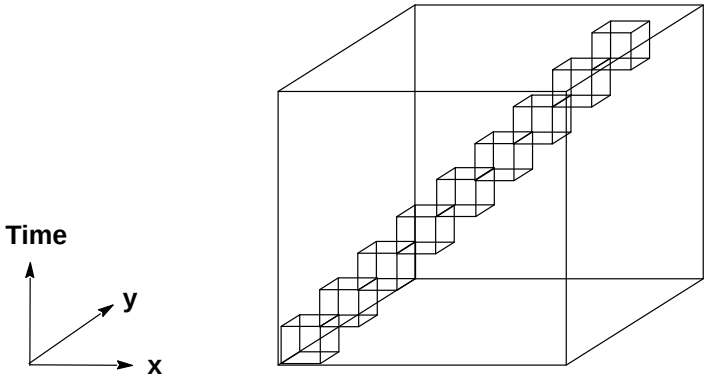

Figure 12 Three–Dimensional Spatiotemporal HHCODE

Each of the object data spaces is a unique space / time cell with its own user–defined time range. You encode time in the same way as any other dimension: by defining the time range as lower and upper boundary values. Although the time range never varies, changing the frequency of the data collection can provide very tiny or very large time slices. These can be useful for displaying animation or expressing minute changes or long–term trends by truncating the HHCODE data based on the time dimension. In this way, you can use time for both generalization and tessellating, as shown with the *x–y* data illustrated previously.

Using Multiple User–Defined Dimensions

You can use virtually any kind of data as spatial data; any numerical data that falls within a defined range, and therefore defines a ”range data space,” can be encoded into an HHCODE. Even data that is normally considered attribute data can be used as a dimension. This provides analysis capabilities that are easier to use than if the data is stored conventionally.

When deciding whether to encode a certain type of data as a dimension rather than an attribute, consider if it is useful to create a ”multidimensional data space” that includes the data as part of its definition.

For example, in a two–dimensional database, latitude and longitude are considered dimensions. Depth, when included, is treated as an attribute. As a result, the data space is defined by only its latitude and longitude values, and the depth attribute value can be used only as a filter in a query. When all three dimensions are encoded into an HHCODE, however, all three

define the object data space. The following example illustrates how this can be useful.

When data about weather is stored as HHCODE data, each HHCODE is essentially a multidimensional topology that exists between constraints. The topology is defined by its dimensions, and its constraints are defined by the data ranges for all dimensions.

Dimension	Dimension Range
1. Latitude	–90 – +90 degrees
2. Longitude	–180 – +180 degrees
3. Altitude	0 – 100,000 feet
4. Time	0 – 25,000 years
5. Temperature	0 – 273 Kelvin
6. Pressure	0 – 500 Atmospheres
7. Electricity	0 – 1,000 coulombs
8. Water Content	0 – 65,000 millibars
9. Dust	0 – 100,000 parts per million

Table 4 Multiple Dimensions Define a Weather Space

The nine dimensions in Table 4 form a weather space. When the values of each dimension are in particular combinations of ranges, certain weather features occur, such as a snow storm over the Canadian Rockies or a hurricane over the Caribbean. The use of multiple dimensions to define an entity or condition can be generalized for use by almost any type of data. The Spatial Data Option ensures that *n*–dimensional HHCODES are clustered spatially to mirror logical real–world relationships.

VLDB Performance through Data Partitioning

One widely recognized method of optimizing VLDB performance is to divide data among large numbers of tables. The goal is to limit the number of tables a query must consider when searching for the requested data, and to minimize the amount of data to be scanned once a table has been identified. Until now, database designers have had to know the approximate quantity of data, the number of tables that will be required, and how to distribute this data effectively. This method is problematic for the following reasons:

- In many cases, database designers cannot know beforehand how much data will be accumulated.
- Data growth over time cannot be accurately predicted, and may require restructuring and reloading of the database numerous times.

With the Spatial Data Option's unique dynamic data partitioning technique, VLDB designers do not need to know or estimate how much data there will be, how to divide it, or where to put it for optimal retrieval performance.

This data partitioning technique takes advantage of the HHCODE structure to store spatial data in multiple tables, called *partitions*, that subdivide dynamically–and automatically–when data is loaded. Each partition represents a specific region of multidimensional space. Partitioning does not adversely affect efficiency, because data is not reorganized; it is simply divided into multiple child partitions when a table containing data for a specific region becomes too dense to maintain fast access time. The spatial organization of data is always maintained. This is the key to fast access.

During partitioning, data is loaded until the partitions are filled to the maximum level that you define, and then they subdivide. This process is automatic. The database designer must only estimate a comfortable partition size for a particular application or collection of data. In other words, the system, rather than the database designer, takes on the burdensome task of partitioning and clustering for best performance.

Oracle7 Spatial Data Option Table Architecture

The Spatial Data Option supports both partitioned and non–partitioned spatial tables. You should determine which to use based on how much data you expect to store.

- If the amount of data is manageable as a single table (taking future growth into account), then use a single, non–partitioned table structure.
- If you anticipate a very large amount of data, or the amount of data cannot be predicted, use a partitioned table structure.

Non–Partitioned Table Storage

Non–partitioned spatial tables have the same characteristics as standard Oracle7 tables, except that they contain an HHCODE column. When you create a non–partitioned spatial table all data is stored in this single table. The spatial organization of the data is maintained in the HHCODE for an individual record. However, to determine the spatial organization of a record in relation to another record within this single table, you can perform an explicit ORDER BY of the HHCODE column.

Partitioned Table Storage

In most cases, a single table cannot store, access, or manage massive amounts of data effectively. In this case, you would label the table as "partitionable." The HHCODE column provides the *partitioning key*.

Data is initially loaded in up to 2^n tables, called partitions, where *n* equals the number of dimensions encoded in the HHCODE. Each partition is identical in structure to the parent table, and contains the same number of columns, column data type and size specifications, and HHCODE dimension definitions. The number of partitions that ultimately exist depends on the quantity and density of data. Conceptually, however, the partitioned table is still a single virtual table. All of the information required to keep track of the partitions belonging to a partitioned table is kept in the Spatial Data Option data dictionary.

You set the maximum partition size when you create a table by setting a *high water mark*, which is the maximum number of records allowed in a partition before it subdivides.

Table Partitioning Process

You can visualize partitioning using the following sample dataset. Create a partitioned table and set a high water mark of 10,000, which limits the amount of data in a given partition to 10,000 records before it subdivides into another set of partitions.

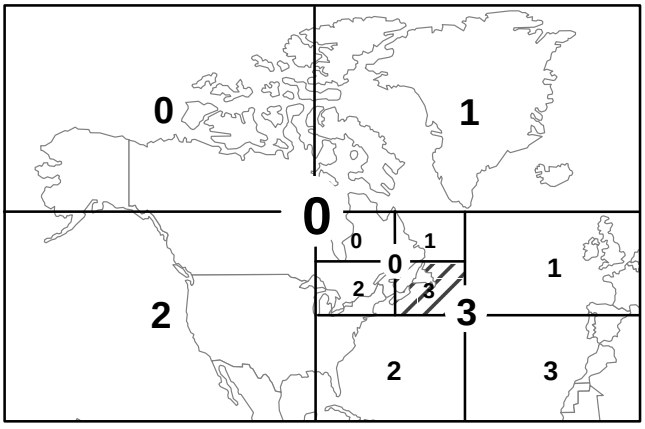

Figure 13 Map Representation of Table Partitioning

When data is loaded and the high water mark is exceeded, the parent table subdivides into child tables, or partitions, representing the next level of decomposition. The initial table is now empty, so it is dropped. Each of the new partitions may contain up to 10,000 records and subdivides if the number of records exceeds 10,000. This partitioning process continues until all records are loaded.

A partition represents a spatial region with boundaries determined by both the high water mark and the density of the HHCODE data for that region. Because the data in one region may be more dense than in another, partitioning may continue to lower levels for a specific quadrant, but not others. Two–dimensional partitioning can be illustrated as a quadtree diagram as shown in Figure 14.

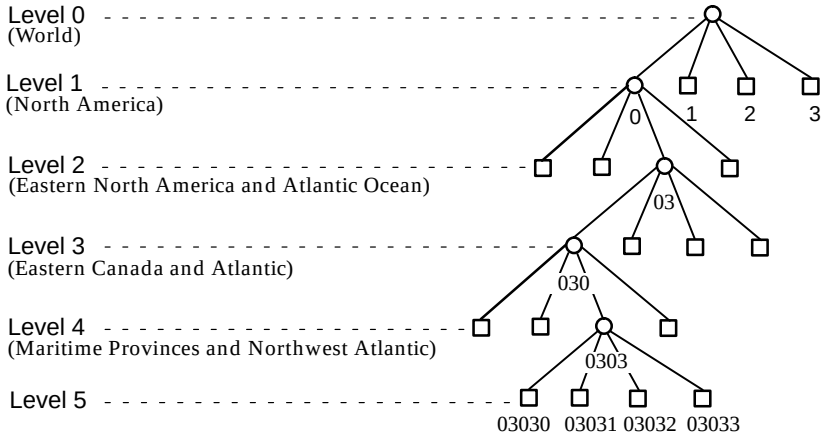

Figure 14 Quadtree Representation of Table Partitioning

In this particular example, none of the primary quadrants (1, 2, and 3), that represent the world has very dense data; when the data is loaded, the partitions do not subdivide again. Because the dataset for the quadrant 0 partition, representing North America, is large, it does continue to subdivide in the quadrants where population of points is greatest.

Level 4 is the last level where subdivision takes place. Only partition 0303, which represents the shaded quadrant in Figure 13, contains sufficient data to subdivide. The four partitions created from partition 0303 all have fewer records than the high water mark, so subdivision stops.

As mentioned previously, the number of actual partitions created depends not only on the high water mark and the amount of data, but also on the number of dimensions specified in the HHCODE. In Figure 14 and Figure 13, only two dimensions were specified in the HHCODE, so each partition subdivides into a set of, at maximum, four new partitions (2^2=4).

Figure 15 illustrates this process for a three–dimensional HHCODE. In this example, the data is loaded into eight or fewer partitions at Level 1. Partitions with no data in them are not actually created, but inferred. Information about inferred

partitions is kept in the Spatial Data Option data dictionary. If more data is loaded later that falls within the data space represented by one of these inferred partitions, then the partition is created physically. Figure 15 illustrates the partitioning process.

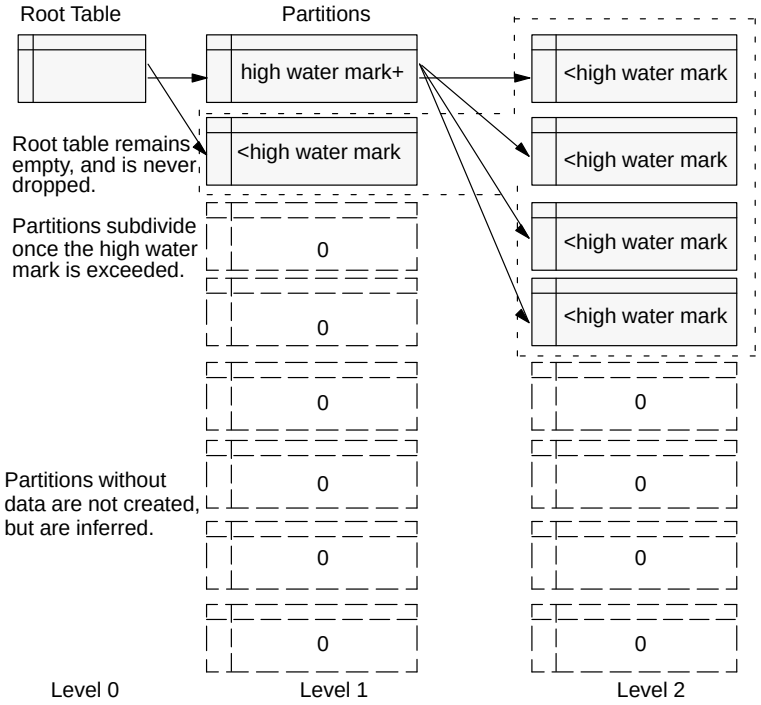

Figure 15 Three–Dimensional Table Partitioning Process

As data is loaded and the high water mark is exceeded for a partition, it subdivides into another set of up to eight partitions. Remember that only the partitions that actually hold data are created. The final set of partitions is made up of only the tables that still contain data. The partitions that exist after loading are indicated by the shading in Figure 15, and are also shown in Figure 16.

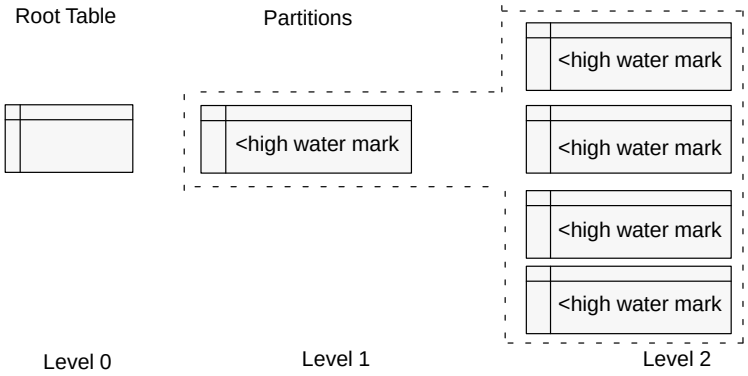

Figure 16 Partitions Remaining after Data Loading

Because they are spatially related, the HHCODEs in each partition are identical to a specific sublevel, represented by a common substring. The real–world relationships between objects are maintained in the database and in the data representing them.

The Spatial Data Option data dictionary keeps track of all the partitions for a partitioned spatial table and the substring of the HHCODE common to all records in a partition. This metadata accelerates query processing by identifying the partitions that need to be scanned.

Data Access and Manipulation

Data Access Methods

Rather than performing traditional RDBMS searches of the database or database indexes, the Spatial Data Option uses partitioning and the Spatial Data Option data dictionary to query only relevant tables. Also, unlike hierarchical spatial index structures, Spatial Data Option accesses both spatial and attribute data with a common query language, SQL, and its extensions.

As already outlined, partitions are created when data is loaded. The number of partitions is determined by the number of dimensions defined, as well as the high water mark you set. For each partition, one entry is maintained in the Spatial Data Option data dictionary.

Users typically access *n*–dimensional ”windows” of data, but the whole volume of data does not have to be searched to access these defined windows. When a user requests data, the following occurs:

- The user identifies an area of interest represented by an *n*–dimensional window.
- The Spatial Data Option data dictionary is scanned, and the partitions that overlap the window are identified.
- The identified partitions are scanned, and the set of rows meeting the query criteria is extracted and passed to the application.

This access method considerably reduces overall storage requirements compared with spatial index structures or traditional RDBMS indexes as shown in Figure 17.

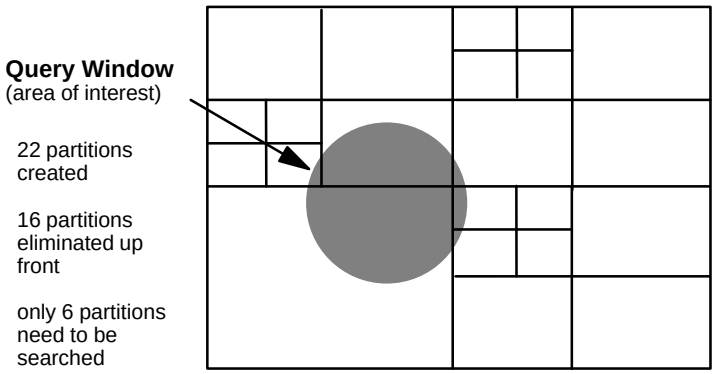

Figure 17 Area of Interest Limits the Number of Partitions that Must Be Scanned

Extracting Windows of Data

There are two methods for extracting windows of data:

- the custom method, which allows users to extract data using a set of four HHCODE functions, and provides full selection optimization
- the package method, which uses a set of packaged PL/SQL procedures that create either a table or a view of the window of data

Both methods allow users to extract windows of data in any number and combination of dimensions. In addition, windows can be defined in three formats:

- as a range search, which is rectangular in two dimensions
- as a proximity search, which is circular in two dimensions
- as a polygon search, which is a multi–sided region in two dimensions

Manipulating Spatial Data

In addition to data access operations, the Spatial Data Option provides SQL extensions for manipulating, querying, and analyzing spatial data. These procedures and functions allow end users to formulate ad hoc queries about spatial data, and provide a basis for application developers to build complex applications that access all the data in the RDBMS. The SQL extensions include:

- Data Definition Language (DDL) procedures and functions for creating, altering, locking, and dropping spatial tables
- Data Manipulation Language (DML) procedures and functions for selecting, deleting, inserting, and updating spatial tables, and for partition maintenance operations
- data extraction procedures and functions for selecting subsets of data from a spatial table
- HHCODE functions as described later in this section

Collectively these procedures and functions provide a complete set of tools for creating, analyzing, and manipulating spatial data in an Oracle7 database. You can also use them in conjunction with other SQL commands and functions to integrate spatial data seamlessly into the Oracle7 environment.

HHCODE Functions

The Spatial Data Option functions fall into several general categories:

- **Creating and Decoding HHCODEs**

  This set of functions allows you to create (or encode) an HHCODE from spatial data values, decode an HHCODE to return the original spatial values, and compose or collapse dimensions out of an HHCODE to create a new HHCODE of fewer dimensions. These functions also give you piecewise access to the individual dimensions of the HHCODE.

- **Obtaining HHCODE Metadata**

  This set of functions provides metadata about an HHCODE. Some of these functions provide information that can be queried from the Spatial Data Option data dictionary. Other functions provide information about the internal structure and real world representation of an HHCODE. This is helpful in designing the most effective HHCODEs, especially for partitioning.

- **Finding HHCODE Commonalities**

  These functions allow you to find the common code and number of matching levels for two HHCODEs.

- **Sorting and Grouping HHCODEs**

  These functions allow you to sort and group spatial data.

- **Using Dates and Times in HHCODEs**

  These functions allow you to convert character string data values into numeric, decimal Julian values, and vice versa. They are used when using a time or date as an HHCODE dimension.

- **Performing Calculations on HHCODEs**

  These functions perform substring and distance calculations on HHCODEs.

- **Window Extraction**

  These functions allow you to identify the partitions and records that fall within a spatial region and are used in a SQL query command.

Summary

Oracle7 Spatial Data Option provides solutions to fundamental problems in managing spatial data.

- By using HHCODEs to represent spatial data, you can store both spatial and attribute data in a single RDBMS table structure–and enjoy the benefits of Oracle7, including security, data integrity, and data independence.
- The HHCODE datatype stores data in a single column in a spatial table, and can represent multiple dimensions. The number and types of dimensions represented by the HHCODE are entirely up to you, so you can even use attribute information as dimensions.
- The unique structure of HHCODE stores data in a spatial table to reflect its spatial organization. The data aggregates according to its defined dimensions, acting as its own index and eliminating the need for spatial or traditional RDBMS indexes.
- Table partitioning ensures fast and consistent query performance in extremely large databases. Only partitions that represent a chosen area of interest are queried for the requested data. Thus, query performance is linear, based primarily on the size of the dataset requested, not the size of the database.

Appendix A: Oracle7 Spatial Data Option Product Modules

The Spatial Data Option is an integrated set of functions, applications, and utilities that enables spatial data to be stored, accessed, and analyzed quickly and efficiently in an Oracle7 database. It is based on HHCODE, which is a generic RDBMS datatype for encoding *n* dimensions into a one–dimensional value.

Figure A – 1 illustrates how the product modules relate to each other, to the Oracle7 database, and to the stored data.

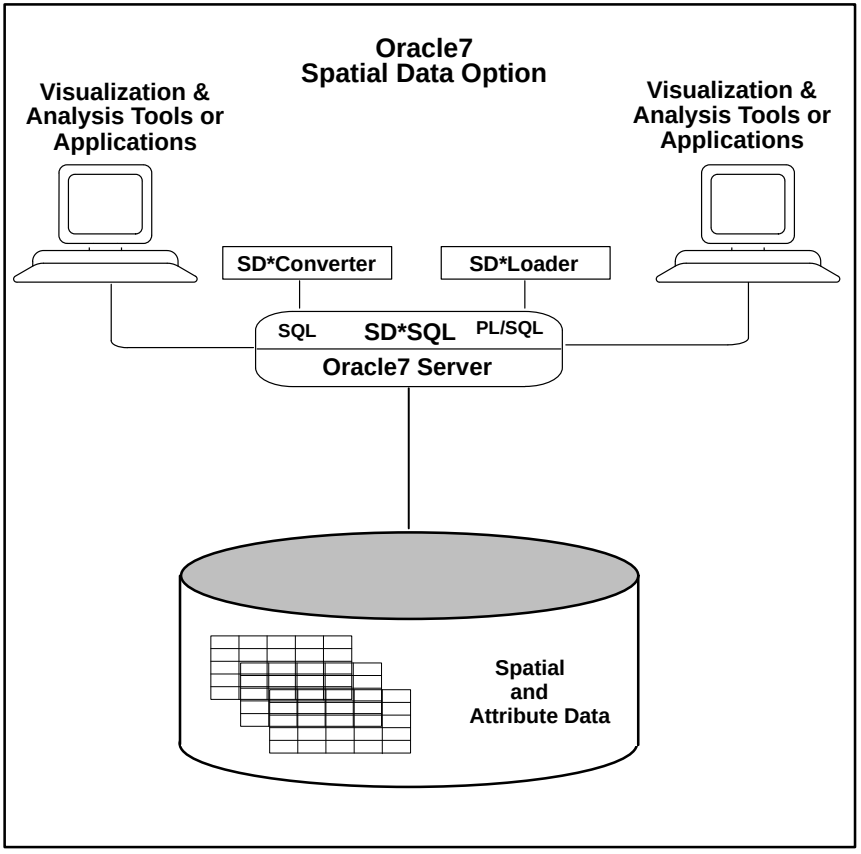

Figure A – 1 Oracle7 Spatial Data Option Product Architecture

	The suite of Spatial Data Option product modules consists of extensions to Oracle7 and a set of utilities for managing very large spatial datasets. These utilities include conversion and loading modules for bringing data into spatial tables, and modules for efficiently managing and manipulating them.
SD*SQL	A SQL and procedural extension to Oracle7 for the management and retrieval of spatial data. SD*SQL provides the functionality required by application developers to build sophisticated graphical user interfaces (GUIs), applications, and products that access, display, and analyze this data.
SD*Converter	A utility that reads external datafiles and converts them into Spatial Load Format (SLF). SLF is the format required to bulk load data into spatial tables.
User–Defined SLF Converter	A set of C language library routines for custom data conversion programs.
SD*Loader	A utility that sorts, loads, and partitions SLF files into spatial tables.

Appendix B: Glossary

This section defines some of the terminology used throughout this document to describe Oracle7 Spatial Data Option concepts and processes.

Aggregation	A process of grouping distinct data (attributes and / or spatial data). The aggregated dataset has a smaller number of data elements than the input dataset.
Attribute Data	In traditional GIS systems, attribute data is the descriptive information characterizing a specific two– or three–dimensional area. For example, the attribute data associated with a stand of forest could include class of forest, property owner's name, major type of tree, and soil type. The Spatial Data Option allows attribute data to be treated as dimensions.
Generalization	A process of simplifying the thematic or geometric content of a map.
GIS (Geographic Information System)	A computer software system (often including hardware) with which spatial information can be captured, stored, analyzed, displayed, and retrieved.
HHCODE	A generic RDBMS datatype for encoding *n* dimensions into a single value.
High Water Mark	A high water mark is a user–defined maximum number of records allowed in a partition before it subdivides.
Multidimensional Cell	In *n*–dimensions, an HHCODE can be visualized as a multidimensional cell or *n*–dimensional area. In two dimensions a cell is referred to as a quadrant. Note, however, that *x* and *y* values for a two–dimensional point will only create an equal–sided point if the data ranges for *x* and *y* are the same. In three dimensions, an HHCODE is most conveniently illustrated as a cube, but is in fact a hexahedron, whose sides are not necessarily equal. This is true for all *n*–dimensional objects.
Network	A series of lines that define the extents of an *n*–point polygon.

Partition	A partition is a physical Oracle7 table that contains data for a particular spatial region. It is logically part of a single, virtual partitioned spatial table. A partition subdivides automatically into child partitions if the amount of data loaded into it exceeds the user–defined high water mark.
Polygon	A class of spatial objects having an area and perimeter, and representing a closed boundary region of uniform characteristics.
Recursive Decomposition	The continuous subdivision of a defined dataset until a specified condition is met. With the Spatial Data Option, this condition is that the number of records in a partition is less than the defined high water mark.
Region	A continuous area with some degree of uniformity in selected characteristics. (See also *Polygon*.)
Resolution	A minimum distance between two objects that can be distinguished by a sensor.
Spatial Data	Spatial data is data that is referred to by a location in *n*–dimensional space and whose position is described by multiple values or components. For example, a point's location in two dimensions can be described by its latitude and longitude values, and in three dimensions by the addition of elevation. In addition to the three Euclidean dimensions and time, it is possible with the Spatial Data Option to define and use data as spatial that is not traditionally treated as such.
Spatial Database	A database containing information indexed by location.
Spatiotemporal Data	Data defined with reference to both space and time.
Temporal Database	Databases designed for the storage and management of spatiotemporal data.
Topology	1. The relationship in space between objects in a spatial database. 2. Explicit coding of spatial relationships between data elements.

Truncation

Multiple–scale representations can be obtained through the process of data aggregation and disaggregation, which is easily accomplished by varying the length of the HHCODEs in a process called truncation.

References

Batty, P. "Why Use a Single DBMS for GIS?", 1991–1992 *International GIS Sourcebook*, Fort Collins, CO.: GIS–World, Inc., 1992, p. 395–398.

Bradley, M. and G. Gall. "New Dimensions in Transparent Data Sharing: Integrating GIS and MIS", *The Canadian Conference on GIS: Proceedings*, Ottawa, Canada: 1994, Vol. 2, p. 1177–1184.

Eganhofer, M.J. "Why Not SQL?", *International Journal of Geographical Information Systems*, 1992, Vol. 6, No. 2, p. 71–85.

Langran, Gail. "A Review of Temporal Database Research and Its Use in GIS Applications", *International Journal of Geographical Information Systems*, 1989, Vol. 3, No. 3, p. 215–232.

Langran, Gail. "Accessing Spatiotemporal Data in a Temporal GIS", *Proceedings of Auto–Carlo 9*, Falls Church, VA: ACSM, 1989, p. 191–198.

Samet, H. *Applications of Spatial Data Structures: Computer Graphics, Image Processing, and GIS*, New York: Addison–Wesley, 1990.

Varma, H., H. Boudreau, and W. Prime. "A Data Structure for Spatio–Temporal Databases", *International Hydrographic Review*, Monaco, LXVII(1), 1990, p. 71–91.

Varma, H., et al. "A New Tool for Managing Very Large Volumes of Hydrographic Data", *Proceedings of the U.S. Hydrographic Conference '94*. Rockville, Maryland: The Hydrographic Society of America, 1994, p. 50–54.

GIS World Sourcebook 1995. GIS World, Inc., 1995. pp. 561–571.

Authors: Gwendolyn Gall, Catharine Kristian, Mark Bradley, and Jim Rawlings

Reader's Comment Form

Oracle7 Spatial Data Option™ Overview
Part No. A43693–1

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the topic, chapter, and page number below:

_____

_____

_____

_____

_____

_____

_____

_____

Please send your comments to:

Government Products Division Documentation Manager
Oracle Corporation
500 Oracle Parkway MS 659204
Redwood City, CA 94065
Phone: (415) 506–2503 Fax: (415) 506–7408

If you would like a reply, please give your name, address, and telephone number below:

_____

_____

_____

Thank you for helping us improve our documentation.