

Oracle® Spatial

Quadtree Indexing

10g Release 1 (10.1)

Part No. B12157-01

December 2003

Provides usage and reference information for the deprecated quadtree indexing capabilities of Oracle Spatial and Oracle Locator. Users are strongly encouraged to use R-tree indexes instead of quadtree indexes, as documented in *Oracle Spatial User's Guide and Reference*.

Oracle Spatial Quadtree Indexing, 10g Release 1 (10.1)

Part No. B12157-01

Copyright © 2003 Oracle Corporation. All rights reserved.

Primary Author: Chuck Murray

Contributors: Dan Abugov, Janet Blowney, Dan Geringer, Siva Ravada

The Programs (which include both the software and documentation) contain proprietary information of Oracle Corporation; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent and other intellectual and industrial property laws. Reverse engineering, disassembly or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. Oracle Corporation does not warrant that this document is error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Oracle Corporation.

If the Programs are delivered to the U.S. Government or anyone licensing or using the programs on behalf of the U.S. Government, the following notice is applicable:

Restricted Rights Notice Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication, and disclosure of the Programs, including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication, and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-19, Commercial Computer Software - Restricted Rights (June, 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and Oracle Corporation disclaims liability for any damages caused by such use of the Programs.

Oracle is a registered trademark, and PL/SQL is a trademark or registered trademark of Oracle Corporation. Other names may be trademarks of their respective owners.

Contents

Send Us Your Comments	ix
Preface	xi
Audience	xi
Documentation Accessibility	xii
Organization.....	xii
Related Documentation	xii
Conventions.....	xiii
1 Quadtree Indexing Concepts	
1.1 Quadtree Indexing Overview	1-2
1.1.1 Tessellation of a Layer During Indexing	1-3
1.1.2 Tiling with Quadtree Indexing.....	1-3
1.2 Quadtree Indexing Considerations and Guidelines	1-6
1.2.1 Determining Index Creation Behavior with Quadtree Indexes	1-7
1.2.2 Spatial Indexing with Fixed-Size Tiles.....	1-7
1.2.3 Using Partitioned Spatial Indexes	1-10
1.2.4 Exchanging Partitions Including Indexes.....	1-11
1.3 Quadtree Indexing Metadata.....	1-11
1.3.1 Metadata View Information for Quadtree Indexes.....	1-11
1.3.2 Spatial Index Table Definition for Quadtree Indexes	1-12
1.4 SQL Reference Information for Quadtree Indexing.....	1-12
1.5 Deprecated Tuning Subprograms for Quadtree Indexing	1-14

2 Tuning Subprograms for Quadtree Indexes

SDO_TUNE.ESTIMATE_INDEX_PERFORMANCE	2-2
SDO_TUNE.ESTIMATE_TILING_LEVEL.....	2-5
SDO_TUNE.ESTIMATE_TILING_TIME	2-7
SDO_TUNE.ESTIMATE_TOTAL_NUMTILES.....	2-10
SDO_TUNE.HISTOGRAM_ANALYSIS	2-13

Index

List of Examples

1-1	Creating a Fixed Index.....	1-10
-----	-----------------------------	------

List of Figures

1-1	Quadtree Decomposition and Morton Codes	1-3
1-2	Fixed-Size Tiling with Many Small Tiles	1-4
1-3	Fixed-Size Tiling with Fewer Large Tiles.....	1-5
1-4	Tessellated Multielement Geometry.....	1-6
1-5	Sample Domain.....	1-8
1-6	Fixed-Size Tiling at Level 1	1-8
1-7	Fixed-Size Tiling at Level 2	1-9

List of Tables

1-1	Choosing R-Tree or Quadtree Indexing.....	1-1
1-2	Columns in the xxx_SDO_INDEX_METADATA Views for Quadtree Indexes	1-11
1-3	Columns in a Quadtree Spatial Index Data Table	1-12
1-4	SQL Features for Quadtree Indexes.....	1-13
1-5	SDO_LEVEL Values.....	1-14
2-1	Tuning Subprograms for Quadtree Indexes.....	2-1

Send Us Your Comments

Oracle Spatial Quadtree Indexing, 10g Release 1 (10.1)

Part No. B12157-01

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 title and part number of the documentation and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Electronic mail: nedc-doc_us@oracle.com
- FAX: 603.897.3825 Attn: Spatial Documentation
- Postal service:
Oracle Corporation
Oracle Spatial Documentation
One Oracle Drive
Nashua, NH 03062-2804
USA

If you would like a reply, please give your name and contact information.

If you have problems with the software, please contact your local Oracle Support Services.

Preface

Oracle Spatial Quadtree Indexing provides usage and reference information for the deprecated quadtree indexing capabilities of Oracle Spatial and Oracle Locator. You are strongly encouraged not to use quadtree indexing for spatial applications, but to use R-tree indexes instead, as documented in *Oracle Spatial User's Guide and Reference*.

Note: To use this document, you must understand the main concepts, data types, techniques, operators, procedures, and functions of Oracle Spatial, which are documented in *Oracle Spatial User's Guide and Reference*.

This preface contains these topics:

- [Audience](#)
- [Documentation Accessibility](#)
- [Organization](#)
- [Related Documentation](#)
- [Conventions](#)

Audience

This guide is intended for anyone who needs to use spatial quadtree indexes.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For additional information, visit the Oracle Accessibility Program Web site at

<http://www.oracle.com/accessibility>

Accessibility of Code Examples in Documentation JAWS, a Windows screen reader, may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, JAWS may not always read a line of text that consists solely of a bracket or brace.

Organization

This guide has the following elements:

Chapter 1, "Quadtree Indexing Concepts"

Explains concepts and techniques that are specific to spatial quadtree indexing.

Chapter 2, "Tuning Subprograms for Quadtree Indexes"

Provides reference information for deprecated tuning subprograms that apply only to spatial quadtree indexes.

Related Documentation

For more information, see *Oracle Spatial User's Guide and Reference* in the Oracle Database documentation set.

Oracle error message documentation is only available in HTML. If you only have access to the Oracle Documentation CD, you can browse the error messages by range. Once you find the specific range, use your browser's "find in page" feature to locate the specific message. When connected to the Internet, you can search for a

specific error message using the error message search feature of the Oracle online documentation.

This document (*Oracle Spatial Quadtree Indexing*) is available only through the Oracle Technology Network (OTN). Go to the OTN Spatial page at

<http://otn.oracle.com/products/spatial/>

To download free release notes, installation documentation, white papers, or other collateral, go to OTN. You must register online before using OTN; registration is free and can be done at

<http://otn.oracle.com/membership>

If you already have a username and password for OTN, then you can go directly to the OTN Spatial page.

Conventions

The following conventions are used in this guide:

Convention	Meaning
.	Vertical ellipsis points in an example mean that information not directly related to the example has been omitted.
...	Horizontal ellipsis points in statements or commands mean that parts of the statement or command not directly related to the example have been omitted.
boldface text	Boldface text indicates a term defined in the text.
monospace text	Monospace text is used for the names of parameters, files, and directory paths. It is also used for SQL and PL/SQL code examples.
<i>italic text</i>	Italic text is used for book titles, emphasis, and some special terms.
< >	Angle brackets enclose user-supplied names.
[]	Brackets enclose optional clauses from which you can choose one or none.

Quadtree Indexing Concepts

This chapter describes the concepts and techniques specific to the deprecated quadtree indexing capabilities of Oracle Spatial. You are strongly encouraged not to use quadtree indexing for spatial applications, but to use R-tree indexes instead, as documented in *Oracle Spatial User's Guide and Reference*, which no longer contains information about quadtree indexing.

Oracle Spatial lets you use R-tree indexing (the default) or quadtree indexing, or both. The use of spatial quadtree indexes is discouraged. You are strongly encouraged to use R-tree indexing for spatial indexes, unless you need to continue using quadtree indexes for special situations. Significant performance improvements have been made to spatial R-tree indexing for this release. In choosing whether to use an R-tree or quadtree index for a spatial application, consider the items in [Table 1-1](#).

Table 1-1 *Choosing R-Tree or Quadtree Indexing*

R-Tree Indexing	Quadtree Indexing
The approximation of geometries cannot be fine-tuned. (Spatial uses the minimum bounding rectangles.)	The approximation of geometries can be fine-tuned by setting the tiling level and number of tiles.
Index creation and tuning are easier.	Tuning is more complex, and setting the appropriate tuning parameter values can affect performance significantly.
Less storage is required.	More storage is required.
If your application workload includes nearest-neighbor queries (SDO_NN operator), R-tree indexes are faster, and you can use the <code>sdo_batch_size</code> keyword.	If your application workload includes nearest-neighbor queries (SDO_NN operator), quadtree indexes are slower, and you cannot use the <code>sdo_batch_size</code> keyword.

Table 1–1 (Cont.) Choosing R-Tree or Quadtree Indexing

R-Tree Indexing	Quadtree Indexing
Heavy update activity to the spatial column may decrease the R-tree index performance until the index is rebuilt.	Heavy update activity does not affect the performance of a quadtree index.
You can index up to four dimensions.	You can index only two dimensions. If LRS (linear referencing system) data is indexed using a spatial quadtree index, only the first two dimensions are indexed; the measure dimension and its values are not indexed.
An R-tree index is recommended for indexing geodetic data if SDO_WITHIN_DISTANCE queries will be used on it.	A quadtree index is not recommended for indexing geodetic data if SDO_WITHIN_DISTANCE queries will be used on it.
An R-tree index is required for a whole-Earth index.	A quadtree index cannot be used for a whole-Earth index.

This chapter contains the following major sections:

- [Section 1.1, "Quadtree Indexing Overview"](#)
- [Section 1.2, "Quadtree Indexing Considerations and Guidelines"](#)
- [Section 1.3, "Quadtree Indexing Metadata"](#)
- [Section 1.4, "SQL Reference Information for Quadtree Indexing"](#)
- [Section 1.5, "Deprecated Tuning Subprograms for Quadtree Indexing"](#)

1.1 Quadtree Indexing Overview

In the linear quadtree indexing scheme, the coordinate space (for the layer where all geometric objects are located) is subjected to a process called **tessellation**, which defines exclusive and exhaustive cover tiles for every stored geometry. Tessellation is done by decomposing the coordinate space in a regular hierarchical manner. The range of coordinates, the coordinate space, is viewed as a rectangle. At the first level of decomposition, the rectangle is divided into halves along each coordinate dimension, generating four tiles. Each tile that interacts with the geometry being tessellated is further decomposed into four tiles. This process continues until some termination criteria, such as size of the tiles or the maximum number of tiles to cover the geometry, is met. The results of the tessellation process on a geometry are stored in a table, referred to as the SDOINDEX table.

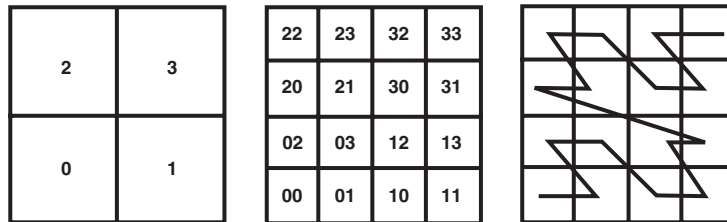
Spatial quadtree indexing uses fixed-size tiles to cover a geometry. Fixed-size tiles are controlled by tile resolution. If the resolution is the sole controlling factor, then tessellation terminates when the coordinate space has been decomposed a specific number of times. Therefore, each tile is of a fixed size and shape.

Fixed-size tile resolution is controlled by a user-selectable keyword named `sdo_level`. Smaller fixed-size tiles provide better geometry approximations. The smaller the number of tiles, the coarser are the approximations.

1.1.1 Tessellation of a Layer During Indexing

The tiles at a particular level can be linearly sorted by systematically visiting tiles in an order determined by a space-filling curve, as shown in Figure 1–1. The tiles can also be assigned unique numeric identifiers, known as Morton codes or z-values. The terms tile and tile code are used interchangeably in this and other sections related to spatial indexing.

Figure 1–1 Quadtree Decomposition and Morton Codes



1.1.2 Tiling with Quadtree Indexing

Spatial quadtree indexing uses tiles of equal size to cover a geometry. Because all tiles are the same size, they all have codes of the same length, and the standard SQL equality operator (=) can be used to compare tiles during a join operation. This results in excellent performance characteristics.

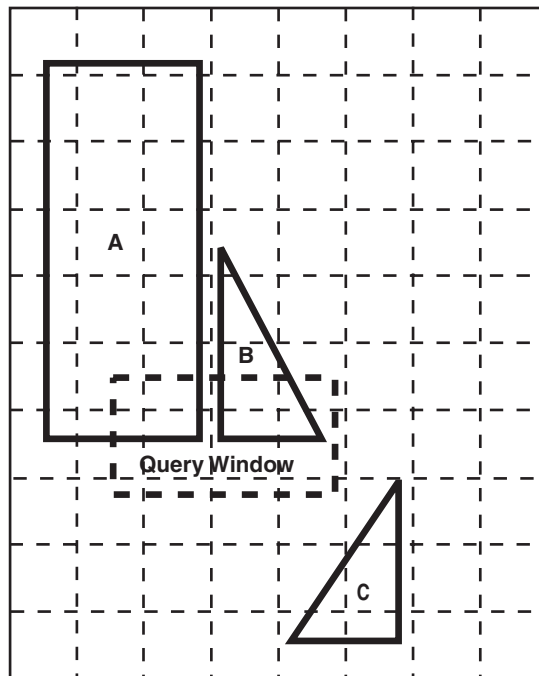
Two geometries are likely to interact, and hence pass the primary filter stage, if they share one or more tiles. The SQL statement for the primary filter stage is:

```
SELECT DISTINCT <select_list for geometry identifiers>
FROM table1_sdoindex A, table2_sdoindex B
WHERE A.sdo_code = B.sdo_code
```

The effectiveness and efficiency of this indexing method depend on the tiling level and the variation in size of the geometries in the layer. If you select a small fixed-size tile to cover small geometries and then try to use the same size tile to cover a very large geometry, a large number of tiles would be required. However, if the chosen tile size is large, so that fewer tiles are generated for a large geometry, then the index selectivity suffers because the large tiles do not approximate the small geometries very well. [Figure 1-2](#) and [Figure 1-3](#) illustrate the relationships between tile size, selectivity, and the number of cover tiles.

With a small fixed-size tile, as shown in [Figure 1-2](#), selectivity is good, but a large number of tiles is needed to cover large geometries. A query that uses the query window in [Figure 1-2](#) would identify geometries A and B, but would correctly reject C.

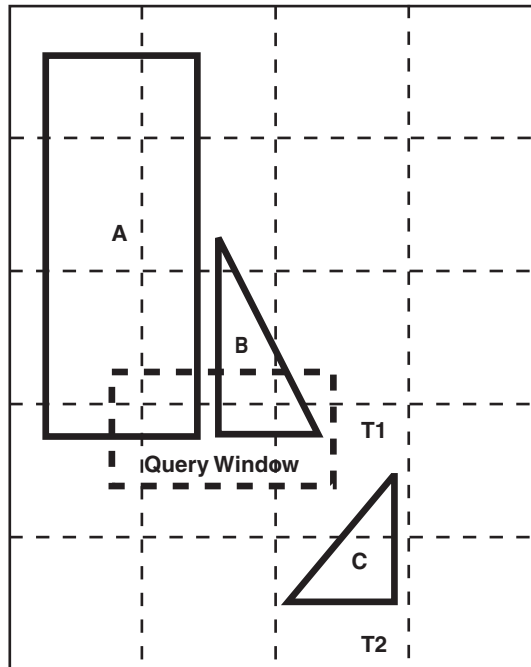
Figure 1-2 Fixed-Size Tiling with Many Small Tiles



With a large fixed-size tile as shown in [Figure 1-3](#), fewer tiles are needed to cover the geometries, but the selectivity is not as good. The same window query as in

Figure 1–2 would probably pick up all three geometries. Any object that shares tile T1 or T2 would identify object C as a candidate, even though the objects may be far apart, such as objects B and C are in Figure 1–3.

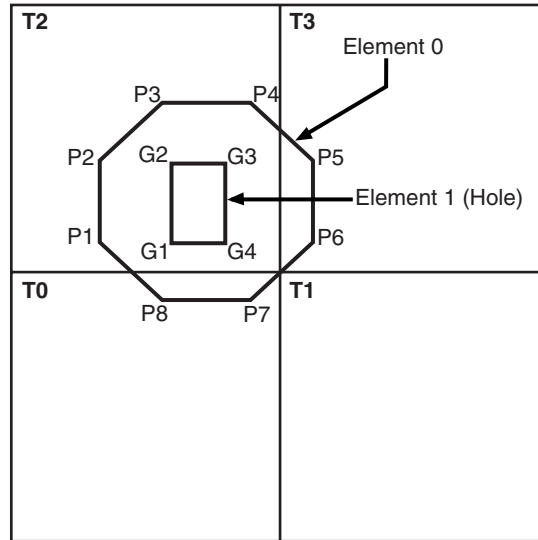
Figure 1–3 Fixed-Size Tiling with Fewer Large Tiles



You can use the [SDO_TUNE.ESTIMATE_TILING_LEVEL](#) function (described in [Chapter 2](#)) or the tiling wizard of the Spatial Index Advisor tool in Oracle Enterprise Manager to help determine an appropriate tiling level for your data set.

Figure 1–4 illustrates a multielement geometry, in this case a polygon with a hole, tessellated to three fixed-sized tiles (T0, T2, and T3) at level 1. The codes for these cover tiles are then stored in an SDOINDEX table.

Figure 1–4 Tessellated Multielement Geometry



Only three of the four tiles generated by the first tessellation interact with the geometry. Only those tiles that interact with the geometry are stored in the index table.

All elements in a geometry are tessellated. In a multielement geometry such as the one in [Figure 1–4](#), Element 1 (the hole) is already covered by tile T2 from the tessellation of Element 0 (the exterior polygon ring). If, however, the specified tiling resolution was such that tile T2 was further subdivided and one of these smaller tiles was completely contained in Element 1, then that tile would be excluded because it would not interact with the geometry.

1.2 Quadtree Indexing Considerations and Guidelines

This section contains considerations and guidelines for creating and using spatial quadtree indexes. It supplements the considerations and guidelines in *Oracle Spatial User's Guide and Reference*. This section contains the following subsections:

- [Section 1.2.1, "Determining Index Creation Behavior with Quadtree Indexes"](#)
- [Section 1.2.2, "Spatial Indexing with Fixed-Size Tiles"](#)
- [Section 1.2.3, "Using Partitioned Spatial Indexes"](#)

- [Section 1.2.4, "Exchanging Partitions Including Indexes"](#)

1.2.1 Determining Index Creation Behavior with Quadtree Indexes

With a quadtree index, the tessellation algorithm used by the CREATE INDEX statement and by index maintenance routines on insert or update operations is determined by the SDO_LEVEL value, which is specified with the `sdo_level` keyword in the PARAMETERS clause of the CREATE INDEX statement. The SDO_LEVEL value is interpreted as follows:

SDO_LEVEL	Action
Not specified or 0	R-tree index
>= 1	Quadtree index (indexing with fixed-size tiles)

An explicit commit operation is executed after the tessellation of all geometries in a geometry column.

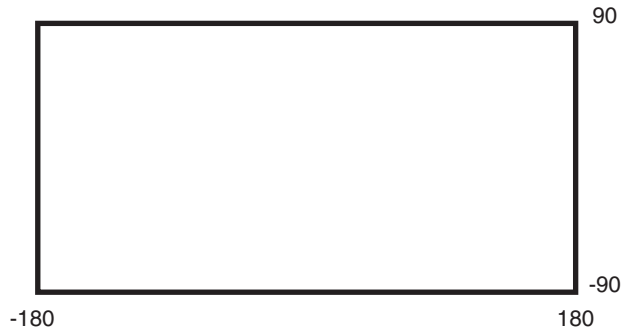
By default, spatial index creation requires a sizable amount of rollback space. To reduce the amount of rollback space required, you can specify the `sdo_commit_interval` keyword in the PARAMETERS clause of the CREATE INDEX statement. This will perform a database commit after every *n* geometries are indexed, where *n* is a user-defined value.

1.2.2 Spatial Indexing with Fixed-Size Tiles

The fixed-size tile algorithm for quadtree indexing is expressed as a level referring to the number of tessellations performed. To use quadtree indexing, set the SDO_LEVEL value to the desired tiling level. The relationship between the tiling level and the resulting size of the tiles depends on the domain of the layer.

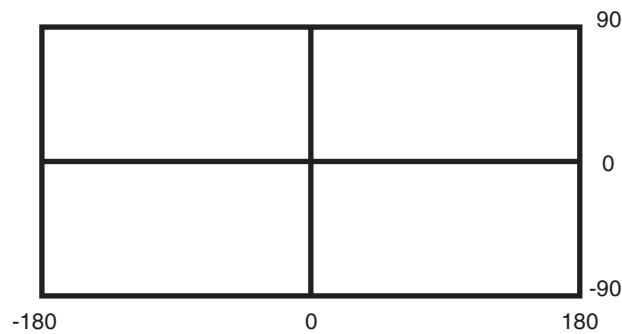
The domain used for indexing is defined by the upper and lower boundaries of each dimension stored in the DIMINFO column of the USER_SDO_GEOM_METADATA view, which contains an entry for the table and geometry column to spatially index. A typical domain could be -180 to 180 degrees for longitude, and -90 to 90 degrees for latitude, as represented by the rectangle in [Figure 1-5](#). (The transference of the domain onto a sphere or other projection is left up to an application, unless a coordinate system is specified.)

Figure 1–5 Sample Domain



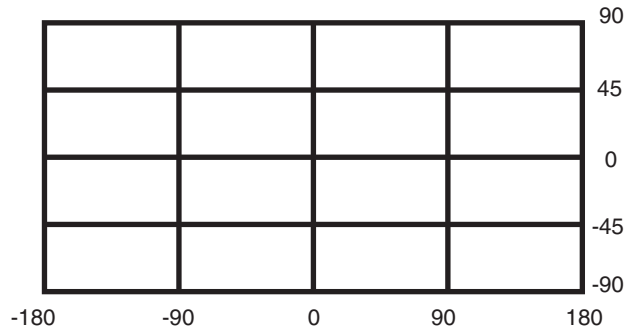
If the SDO_LEVEL column is set to 1, the tiles created by the indexing mechanism are the same size as tiles at the first level of tessellation. Each tile would be 180 degrees by 90 degrees, as shown in [Figure 1–6](#), where the original domain is divided into four rectangles (tiles) of equal size.

Figure 1–6 Fixed-Size Tiling at Level 1



The formula for the number of fixed-size tiles in a domain is 4^n where n is the number of tessellations stored in the SDO_LEVEL column. In reality, tiles are only generated where geometries exist, and not for the whole domain. [Figure 1–7](#) shows fixed-size tiling at level 2. In this figure, each tile is 90 degrees by 45 degrees, and the original domain is divided into 16 rectangles (tiles) of equal size.

Figure 1-7 Fixed-Size Tiling at Level 2



The size of a tile can be determined by applying the following formula to each dimension:

$$\text{length} = (\text{upper_bound} - \text{lower_bound}) / 2^{\text{sdo_level}}$$

The length refers to the length of the tile along the specified dimension. Applying this formula to the tiling shown in [Figure 1-7](#) yields the following sizes:

$$\begin{aligned} \text{length for dimension X} &= (180 - (-180)) / 2^2 \\ &= (360) / 4 \\ &= 90 \end{aligned}$$

$$\begin{aligned} \text{length for dimension Y} &= (90 - (-90)) / 2^2 \\ &= (180) / 4 \\ &= 45 \end{aligned}$$

At level 2, the tiles are 90 degrees by 45 degrees in size. As the number of levels increases, the tiles become smaller and smaller. Smaller tiles provide a more precise fit of the tiles over the geometry being indexed. However, because the number of tiles generated is unbounded, you must take into account the performance implications of using higher levels.

Note: The Spatial Index Advisor component of Oracle Enterprise Manager can be used to determine an appropriate level for indexing with fixed-size tiles. The [SDO_TUNE.ESTIMATE_TILING_LEVEL](#) function, described in [Chapter 2](#), can also be used for this purpose; however, this function performs less analysis than the Spatial Index Advisor.

Besides the performance aspects related to selecting a fixed-size tile, tessellating the geometry into fixed-size tiles might have benefits related to the type of data being stored, such as using tiles sized to represent 1-acre farm plots, city blocks, or individual pixels on a display. Data modeling, an important part of any database design, is essential in a spatial database where the data often represents actual physical locations.

In [Example 1-1](#), assume that data has been loaded into a table called `ROADS`, and the `USER_SDO_GEOM_METADATA` view has an entry for `ROADS.SHAPE`. You can use the following SQL statement to create a fixed index named `ROADS_FIXED`.

Example 1-1 Creating a Fixed Index

```
CREATE INDEX ROADS_FIXED ON ROADS(SHAPE) INDEXTYPE IS MDSYS.SPATIAL_INDEX
  PARAMETERS ('SDO_LEVEL=8');
```

The `SDO_LEVEL` value is used while tessellating objects. Increasing the level results in smaller tiles and better geometry approximations.

1.2.3 Using Partitioned Spatial Indexes

You can create a partitioned spatial index on a partitioned table. This section describes usage considerations specific to quadtree indexing. It supplements the information about partitioned spatial indexes in *Oracle Spatial User's Guide and Reference*.

If you specify parameters for individual partitions, the following considerations apply:

- All index partitions must have either R-tree or quadtree indexes. Mixed R-tree and quadtree indexes are not allowed.
- For quadtree indexes, the same `sdo_level` value must be used for each partition.

In the following example, if the `COUNTIES` table has two partitions, `P1` and `P2`, you can create a quadtree index as follows:

```
CREATE INDEX counties_idx ON counties(geometry)
  INDEXTYPE IS MDSYS.SPATIAL_INDEX
  PARAMETERS ('sdo_level=6 tablespace=def_tbs')
LOCAL
(PARTITION ip1 PARAMETERS ('sdo_level=6 tablespace=local_tbs1'),
 PARTITION ip2 PARAMETERS ('sdo_level=6 tablespace=local_tbs2'),
 PARTITION ip3);
```

In the preceding example:

- IP1 is the index partition that corresponds to partition P1 of the COUNTIES table, and IP2 is the index partition that corresponds to partition P2.
- The tablespace parameters are specified for each of the local index partitions as LOCAL_TBS1 and LOCAL_TBS2, respectively.
- If you omit the PARTITION ip2 PARAMETERS . . . clause (as is done for partition IP3), the default parameters specified before LOCAL are used. Specifically, the DEF_TBS tablespace is used for storing the index partition (which will have the same name as the second partition of the COUNTIES table, that is, P2).

1.2.4 Exchanging Partitions Including Indexes

You can use the ALTER TABLE statement with the EXCHANGE PARTITION ... INCLUDING INDEXES clause to exchange a spatial table partition and its index partition with a corresponding table and its index. In addition to the information about exchanging partitions including indexes in *Oracle Spatial User's Guide and Reference*, quadtree indexes must have the same SDO_LEVEL value.

1.3 Quadtree Indexing Metadata

This section describes Spatial metadata information that applies to quadtree indexing. It supplements the information about Spatial metadata in *Oracle Spatial User's Guide and Reference*.

1.3.1 Metadata View Information for Quadtree Indexes

The USER_SDO_INDEX_INFO and ALL_SDO_INDEX_INFO views include a column named SDO_INDEX_TYPE, of type VARCHAR2, which contains QTREE for a quadtree index or RTREE for an R-tree index.

The USER_SDO_INDEX_METADATA and ALL_SDO_INDEX_METADATA views include columns, shown [Table 1-2](#), that apply only to quadtree indexes.

Table 1-2 Columns in the xxx_SDO_INDEX_METADATA Views for Quadtree Indexes

Column Name	Data Type	Purpose
SDO_INDEX_TYPE	VARCHAR2	Contains QTREE (for a quadtree index) or RTREE (for an R-tree index).

Table 1–2 (Cont.) Columns in the xxx_SDO_INDEX_METADATA Views for Quadtree

Column Name	Data Type	Purpose
SDO_LEVEL	NUMBER	The fixed tiling level at which to tile all objects in the geometry column for a quadtree index.
SDO_NUMTILES	NUMBER	Suggested number of tiles per object that should be used to approximate the shape for a quadtree index.
SDO_MAXLEVEL	NUMBER	Maximum level for any tile for any object for a quadtree index. It will always be greater than the SDO_LEVEL value.

1.3.2 Spatial Index Table Definition for Quadtree Indexes

The information in each spatial index table (each SDO_INDEX_TABLE entry) depends on whether the index is an R-tree index or a quadtree index.

For a quadtree index, the spatial index table contains the columns shown in [Table 1–3](#).

Table 1–3 Columns in a Quadtree Spatial Index Data Table

Column Name	Data Type	Purpose
SDO_CODE	RAW	Index entry for the object in the row identified by SDO_ROWID.
SDO_ROWID	ROWID	Rowid of a row in a feature table containing the indexed object.
SDO_STATUS	VARCHAR2	Contains <i>I</i> if the tile is inside the geometry, or contains <i>B</i> if the tile is on the boundary of the geometry.
SDO_GROUPCODE	RAW	Index entry at level SDO_LEVEL (hybrid indexes only).

For a quadtree index, the SDO_CODE, SDO_ROWID, and SDO_STATUS columns are always present. The SDO_GROUPCODE column is present only when the selected index type is HYBRID.

1.4 SQL Reference Information for Quadtree Indexing

This section describes SQL reference information that applies specifically to creating and modifying spatial quadtree indexes. It supplements the SQL reference information in *Oracle Spatial User's Guide and Reference*.

[Table 1–4](#) lists features, explanations of the features, and the SQL statements to which the features apply.

Table 1–4 SQL Features for Quadtree Indexes

Feature	Explanation	SQL Statements
<code>sdo_commit_interval</code> keyword	Specifies the number of underlying table rows that are processed between commit intervals for the index data. The default behavior commits the index data only after all rows in the underlying table have been processed. Data type is NUMBER.	ALTER INDEX ALTER INDEX REBUILD CREATE INDEX
<code>sdo_level</code> keyword	Specifies the desired fixed-size tiling level. Data type is NUMBER.	ALTER INDEX REBUILD CREATE INDEX
<code>btree_initial</code> keyword	Is the same as INITIAL in the STORAGE clause of a CREATE INDEX statement for a standard B-tree index.	ALTER INDEX REBUILD CREATE INDEX
<code>btree_next</code> keyword	Is the same as NEXT in the STORAGE clause of a CREATE INDEX statement for a standard B-tree index.	ALTER INDEX REBUILD CREATE INDEX
<code>btree_pctincrease</code> keyword	Is the same as PCTINCREASE in the STORAGE clause of a CREATE INDEX statement for a standard B-tree index.	ALTER INDEX REBUILD CREATE INDEX

The following usage notes for the CREATE INDEX statement apply specifically to quadtree indexes.

By default, an R-tree index is created if the `index_params` string does not contain the `sdo_level` keyword or if the `sdo_level` value is zero (0). If the `index_params` string contains the `sdo_level` keyword with a nonzero value, a quadtree index is created.

For a quadtree index, the `index_params` string must contain `sdo_level`, and the value specified for this parameter must be valid.

For a quadtree index on mixed point and nonpoint data, specifying PARALLEL can degrade the performance of creating or rebuilding an index.

By default, the `sdo_commit_interval` keyword does not allow spatial data to be committed at intervals. Insertion of spatial index data is, by default, committed only at the end of the index creation process. That is, it is committed after all rows in the underlying table have been processed.

The `sdo_level` keyword value must be greater than zero for a quadtree index.

Interpretation of `sdo_level` values is shown in [Table 1–5](#).

Table 1–5 SDO_LEVEL Values

SDO_LEVEL	Action
Not specified or 0	R-tree index
>= 1	Quadtree index (indexing with fixed-size tiles)

The 'geodetic=FALSE' parameter allows you to bypass the restriction that a standard quadtree index cannot be used with geodetic data. However, using this parameter is not recommended, because much of the Oracle Spatial geodetic support will be disabled, and some Spatial operations that use the quadtree index with geodetic data will not work correctly or will return less accurate results. This parameter should only be used if you cannot yet reindex the data with an R-tree index and if the results using the non-geodetic quadtree index are acceptable. (See the usage notes for the CREATE INDEX statement in *Oracle Spatial User's Guide and Reference* for more information about the 'geodetic=FALSE' parameter.)

1.5 Deprecated Tuning Subprograms for Quadtree Indexing

The following SDO_TUNE subprograms specific to spatial quadtree indexing are deprecated:

- [SDO_TUNE.ESTIMATE_INDEX_PERFORMANCE](#)
- [SDO_TUNE.ESTIMATE_TILING_LEVEL](#)
- [SDO_TUNE.ESTIMATE_TILING_TIME](#)
- [SDO_TUNE.ESTIMATE_TOTAL_NUMTILES](#)
- [SDO_TUNE.HISTOGRAM_ANALYSIS](#)

These subprograms are documented in [Chapter 2](#).

Tuning Subprograms for Quadtree Indexes

This chapter contains descriptions of the tuning subprograms shown in [Table 2-1](#). These subprograms apply only to quadtree indexes, and they are deprecated.

Table 2-1 *Tuning Subprograms for Quadtree Indexes*

Subprogram	Description
SDO_TUNE.ESTIMATE_INDEX_PERFORMANCE	Estimates the spatial index performance, such as query selectivity and window query time.
SDO_TUNE.ESTIMATE_TILING_LEVEL	Estimates the appropriate tiling level for creating an index.
SDO_TUNE.ESTIMATE_TILING_TIME	Returns the estimated time (in seconds) to tessellate a column of type SDO_GEOMETRY for creating an index.
SDO_TUNE.ESTIMATE_TOTAL_NUMTILES	Estimates the total number of spatial tiles for creating an index on a column of type SDO_GEOMETRY.
SDO_TUNE.HISTOGRAM_ANALYSIS	Calculates statistical histograms for a spatial layer.

SDO_TUNE.ESTIMATE_INDEX_PERFORMANCE

Format

```
SDO_TUNE.ESTIMATE_INDEX_PERFORMANCE(  
    table_name    IN VARCHAR2,  
    column_name   IN VARCHAR2,  
    sample_ratio  IN INTEGER DEFAULT 20,  
    tiling_level  IN INTEGER,  
    num_tiles     IN INTEGER DEFAULT 0,  
    window_obj    IN SDO_GEOMETRY,  
    tiling_time   OUT NUMBER,  
    filter_time   OUT NUMBER,  
    query_time    OUT NUMBER  
) RETURN NUMBER;
```

Description

Estimates the spatial index performance, such as query selectivity and window query time, for a quadtree index to be created on a column of type SDO_GEOMETRY.

Note: This function is deprecated, and will not be supported in future releases of Oracle Spatial. This function applies only to spatial quadtree indexing. You are strongly encouraged to use R-tree indexing instead of quadtree indexing.

Parameters

table_name

Spatial geometry table.

column_name

Geometry column for which the tiling time is to be estimated.

sample_ratio

Approximate ratio between the geometries in the original layer and those in the sample layer (to be generated in order to perform the estimate). The default is 20: that is, the sample layer will contain approximately 1/20 (5 percent) of the geometries in the original layer. The larger the `sample_ratio` value, the faster the function will run, but the less accurate will be the result (the estimate).

Spatial obtains the sample by using the `SAMPLE(sample_percent)` feature internally. For a description of this feature, see the `sample_clause` description in the `SELECT` statement section of *Oracle Database SQL Reference*.

tiling_level

Spatial index level at which the layer is to be tessellated.

num_tiles

Number of tiles for variable or hybrid tessellation. Should be 0 for fixed tessellation. The default is 0.

window_obj

Window geometry object.

tiling_time

Estimated tiling time in seconds.

filter_time

Estimated spatial index filter time in seconds.

query_time

Estimated window query time in seconds.

Usage Notes

The function returns a number between 0.0 and 1.0 representing estimated spatial index selectivity for a quadtree index. The larger the number, the better the selectivity.

The `sample_ratio` parameter lets you control the trade-off between speed and accuracy. Note that `sample_ratio` is not exact, but reflects an average. For example, a `sample_ratio` value of 20 sometimes causes fewer than 5 percent of geometry objects to be sampled and sometimes more than 5 percent, but over time an average of 5 percent will be sampled.

A return value of 0.0 indicates an error.

Examples

The following example estimates index performance information with a specified query window for a quadtree index to be created on the SHAPE column of the COLA_MARKETS table.

```
DECLARE
  table_name  VARCHAR2(32) := 'COLA_MARKETS';
  column_name VARCHAR2(32) := 'SHAPE';
  sample_ratio INTEGER := 15;
  tiling_level INTEGER := 4;
  num_tiles   INTEGER := 10;
  window_obj  SDO_GEOMETRY :=
  SDO_GEOMETRY(
    2003, -- two-dimensional polygon
    NULL,
    NULL,
    SDO_ELEM_INFO_ARRAY(1,1003,1), -- one polygon
    SDO_ORDINATE_ARRAY(3,3, 6,3, 6,5, 4,5, 3,3)
  );
  tiling_time NUMBER;
  filter_time  NUMBER;
  query_time  NUMBER;
  ret_number  NUMBER;
BEGIN
  ret_number := SDO_TUNE.ESTIMATE_INDEX_PERFORMANCE(
    table_name,
    column_name,
    sample_ratio,
    tiling_level,
    num_tiles,
    window_obj,
    tiling_time,
    filter_time,
    query_time
  );
END;
/
Tiling time = 3.28
Filter time = 1.2
Query time = .9
Result = 1
```

SDO_TUNE.ESTIMATE_TILING_LEVEL

Format

```
SDO_TUNE.ESTIMATE_TILING_LEVEL(  
    table_name      IN VARCHAR2,  
    column_name     IN VARCHAR2,  
    num_tiles       IN INTEGER  
    [, type_of_estimate IN VARCHAR2]  
    ) RETURN INTEGER;
```

Description

Estimates the appropriate tiling level for creating a quadtree index.

Note: This function is deprecated, and will not be supported in future releases of Oracle Spatial. This function applies only to spatial quadtree indexing. You are strongly encouraged to use R-tree indexing instead of quadtree indexing.

Parameters

table_name

Spatial geometry table.

column_name

Geometry column for which the tiling level is to be estimated.

num_tiles

Maximum number of tiles that can be used to index the rectangle defined by `type_of_estimate`.

type_of_estimate

Keyword to specify the type of estimate:

- `LAYER_EXTENT` -- Uses the rectangle defined by your coordinate system.

- `ALL_GID_EXTENT` -- Uses the minimum bounding rectangle that encompasses all geometries in the column. This estimate is recommended for most applications.
- `AVG_GID_EXTENT` (default) -- Uses a rectangle representing the average size of the individual geometries within the column. This option is the default and performs the most analysis of the three types, but it takes the longest time to complete.

Usage Notes

The function returns an integer representing the level to use when creating a spatial index for the specified layer. The function returns NULL if the data is inconsistent.

If `type_of_estimate` is `ALL_GID_EXTENT`, a `num_tiles` value of 10000 is recommended for most applications.

Examples

The following example estimates the appropriate `SDO_LEVEL` value to use with the `SHAPE` column of the `COLA_MARKETS` table.

```
SELECT SDO_TUNE.ESTIMATE_TILING_LEVEL('COLA_MARKETS', 'SHAPE',  
                                     10000, 'ALL_GID_EXTENT')  
FROM DUAL;
```

```
SDO_TUNE.ESTIMATE_TILING_LEVEL('COLA_MARKETS', 'SHAPE', 10000, 'ALL_GID_EXTENT')
```

SDO_TUNE.ESTIMATE_TILING_TIME

Format

```
SDO_TUNE.ESTIMATE_TILING_TIME(  
    table_name    IN VARCHAR2,  
    column_name   IN VARCHAR2,  
    sample_ratio  IN INTEGER,  
    tiling_level  IN INTEGER,  
    num_tiles     IN INTEGER  
) RETURN NUMBER;
```

Description

Returns the estimated time (in seconds) to tessellate a column of type SDO_GEOMETRY for creating a quadtree index.

Note: This function is deprecated, and will not be supported in future releases of Oracle Spatial. This function applies only to spatial quadtree indexing. You are strongly encouraged to use R-tree indexing instead of quadtree indexing.

Parameters

table_name

Spatial geometry table.

column_name

Geometry column for which the tiling time is to be estimated.

sample_ratio

Approximate ratio between the geometries in the original layer and those in the sample layer (to be generated to perform the estimate). The default is 20: that is, the sample layer will contain approximately 1/20 (5 percent) of the geometries in the original layer. As you increase the `sample_ratio` value, the execution time for the function decreases, but the accuracy of the result (the estimate) decreases also.

Spatial obtains the sample by using the SAMPLE(sample_percent) feature internally. For a description of this feature, see the `sample_clause` description in the SELECT statement section of *Oracle Database SQL Reference*.

tiling_level

Spatial index level at which the layer is to be tessellated.

num_tiles

Number of tiles for variable or hybrid tessellation. This number should be 0 for fixed tessellation. The default is 0.

Usage Notes

A return value of 0 indicates an error.

The tiling time estimate is based on the tiling time of a small sample spatial geometry table that is automatically generated from the original table column. (This generated table is deleted before the function completes.)

The `sample_ratio` parameter lets you control the trade-off between speed and accuracy. Note that `sample_ratio` is not exact, but reflects an average. For example, a `sample_ratio` value of 20 sometimes causes fewer than 5 percent of geometry objects to be sampled and sometimes more than 5 percent, but over time an average of 5 percent will be sampled.

The CREATE TABLE privilege is required for using this function.

Examples

The following example estimates the tiling time to tessellate the REGIONS column of the XYZ_MARKETS table.

```
DECLARE
  table_name  VARCHAR2(32) := 'XYZ_MARKETS';
  column_name VARCHAR2(32) := 'REGIONS';
  sample_ratio INTEGER := 15;
  tiling_level INTEGER := 6;
  num_tiles   INTEGER := 10;
  ret_number  NUMBER;
BEGIN
  ret_number := SDO_TUNE.ESTIMATE_TILING_TIME(
    table_name,
    column_name,
    sample_ratio,
    tiling_level,
    num_tiles
```

```
);  
END;
```

SDO_TUNE.ESTIMATE_TOTAL_NUMTILES

Format

```
SDO_TUNE.ESTIMATE_TOTAL_NUMTILES(  
    table_name    IN VARCHAR2,  
    column_name   IN VARCHAR2,  
    sample_ratio  IN INTEGER,  
    tiling_level  IN INTEGER,  
    num_tiles     IN INTEGER,  
    num_targetiles OUT INTEGER  
) RETURN INTEGER;
```

Description

Estimates the total number of spatial tiles for creating a quadtree index on a column of type SDO_GEOMETRY.

Note: This function is deprecated, and will not be supported in future releases of Oracle Spatial. This function applies only to spatial quadtree indexing. You are strongly encouraged to use R-tree indexing instead of quadtree indexing.

Parameters

table_name

Spatial geometry table.

column_name

Geometry column for which the total number of spatial tiles is to be estimated.

sample_ratio

Approximate ratio between the geometries in the original layer and those in the sample layer (to be generated to perform the estimate). The default is 20: that is, the sample layer will contain approximately 1/20 (5 percent) of the geometries in the

original layer. The larger the `sample_ratio` value, the faster the function will run, but the less accurate will be the result (the estimate).

Spatial obtains the sample by using the `SAMPLE(sample_percent)` feature internally. For a description of this feature, see the `sample_clause` description in the `SELECT` statement section of *Oracle Database SQL Reference*.

tiling_level

Spatial index level at which the layer is to be tessellated.

num_tiles

Number of tiles for variable or hybrid tessellation. Should be 0 for fixed tessellation. The default is 0.

num_largetiles

Output parameter to contain the number of spatial tiles that are of the same size as group tiles for hybrid indexing. (For fixed indexing, `num_largetiles` will be the same as the returned value: the total number of spatial tiles.)

Usage Notes

The estimate is based on the total number of tiles for a small sample layer that is automatically generated from the original layer. (This generated table is deleted before the function completes.)

The `sample_ratio` parameter lets you control the trade-off between speed and accuracy. Note that `sample_ratio` is not exact, but reflects an average. For example, a `sample_ratio` value of 20 sometimes causes fewer than 5 percent of geometry objects to be sampled and sometimes more than 5 percent, but over time an average of 5 percent will be sampled.

The `CREATE TABLE` privilege is required for using this function.

Examples

The following example estimates the total number of spatial tiles required to index the `REGIONS` column of the `XYZ_MARKETS` table.

```
DECLARE
  table_name  VARCHAR2(32) := 'XYZ_MARKETS';
  column_name VARCHAR2(32) := 'REGIONS';
  sample_ratio INTEGER := 15;
  tiling_level INTEGER := 4;
  num_tiles   INTEGER := 10;
  num_largetiles INTEGER;
```

```
    ret_integer INTEGER;
BEGIN
ret_integer := SDO_TUNE.ESTIMATE_TOTAL_NUMTILES (
    table_name,
    column_name,
    sample_ratio,
    tiling_level,
    num_tiles,
    num_targettiles
);
END;
/
```

SDO_TUNE.HISTOGRAM_ANALYSIS

Format

```
SDO_TUNE.HISTOGRAM_ANALYSIS(  
    table_name      IN VARCHAR2,  
    column_name     IN VARCHAR2,  
    result_table    IN VARCHAR2,  
    type_of_histogram IN VARCHAR2,  
    max_value       IN NUMBER,  
    intervals       IN INTEGER);
```

Description

Generates statistical histograms based on columns of type SDO_GEOMETRY.

Note: This procedure is deprecated, and will not be supported in future releases of Oracle Spatial. This procedure applies only to spatial quadtree indexing. You are strongly encouraged to use R-tree indexing instead of quadtree indexing.

Parameters

table_name

Spatial geometry table.

column_name

Geometry object column for which the histogram is to be computed.

result_table

Result table to hold the histogram.

type_of_histogram

Keyword to specify the type of histogram:

- **TILES_VS_LEVEL** (default) -- Provides the number of tiles at different spatial index levels. (Available only with hybrid quadtree indexes.) This histogram is

the default, and is used to evaluate the spatial index that is already built on the geometry column.

- **GEOMS_VS_TILES** -- Provides the number of geometries in different number-of-tiles ranges. This histogram is used to evaluate the spatial index that is already built on the geometry column.
- **GEOMS_VS_AREA** -- Provides the number of geometries in different size ranges. The shape of this histogram could be helpful in choosing a proper index level.
- **GEOMS_VS_VERTICES** -- Provides a histogram of the geometry count against the number of vertices. This histogram could help determine if spatial index selectivity is important for the layer. Because the number of vertices determines the performance of the secondary filter, selectivity of the primary filter could be crucial for layers that contain many complicated geometries.

max_value

The upper limit of the histogram. That is, the histogram runs in range (0, max_value).

intervals

Number of intervals between 0 and max_value .

Usage Notes

The procedure populates the result table with statistical histograms for a spatial geometry table. (HISTOGRAM_ANALYSIS is a procedure, not a function. Procedures do not return values.)

Before calling this procedure, create the result table (`result_table` parameter) with VALUE and COUNT columns. For example:

```
CREATE TABLE histogram (value NUMBER, count NUMBER);
```

Index

E

ESTIMATE_INDEX_PERFORMANCE
function, 2-2
ESTIMATE_TILING_LEVEL function, 2-5
ESTIMATE_TILING_TIME function, 2-7
ESTIMATE_TOTAL_NUMTILES function, 2-10
exchanging partitions including indexes, 1-11

F

fixed-size tiles, 1-7

H

HISTOGRAM_ANALYSIS procedure, 2-13
histograms
generating, 2-13

I

indexes
See quadtree indexes

L

LOCAL partitioning
spatial index, 1-10

M

metadata view information for quadtree
indexes, 1-11
Morton codes, 1-3

P

partitioned spatial index, 1-10
exchanging partitions, 1-11

Q

quadtree indexes, 1-2
determining index creation behavior, 1-7
estimating performance, 2-2
exchanging partitions including indexes, 1-11
metadata view information, 1-11
partitioned index on partitioned table, 1-10
SQL reference information, 1-12
when to use, 1-1

R

rollback space
reducing amount required for spatial index
creation, 1-7
R-tree indexes
when to use, 1-1

S

SDO_CODE column, 1-12
SDO_GROUPCODE column, 1-12
SDO_LEVEL value, 1-7
SDO_ROWID column, 1-12
SDO_STATUS column, 1-12
SDO_TUNE package
ESTIMATE_INDEX_PERFORMANCE, 2-2
ESTIMATE_TILING_LEVEL, 2-5

- ESTIMATE_TILING_TIME, 2-7
- ESTIMATE_TOTAL_NUMTILES, 2-10
- HISTOGRAM_ANALYSIS, 2-13
- Spatial Index Advisor
 - using to determine best tiling level, 1-9
- spatial index table definition for quadtree
 - indexes, 1-12
- SQL reference information for quadtree
 - indexing, 1-12
- subprograms
 - tuning, 2-1

T

- tessellation, 1-2
- tiling
 - estimating total number of tiles, 2-10
 - with quadtree indexing, 1-3
- tiling level
 - estimating, 2-5
 - setting, 1-7
- tiling time
 - estimating, 2-7
- tuning subprograms, 2-1

V

- views
 - metadata view information for quadtree
 - indexes, 1-11

Z

- z-values, 1-3