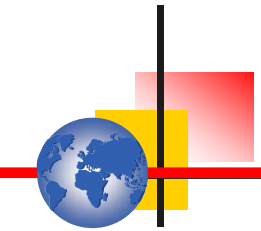
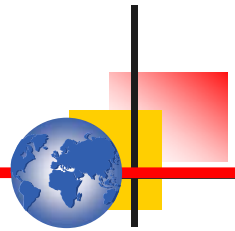
Oracle 10g Spatial
----空间数据模型

信息工程学院
万波
Tel: 13971097150
e-mail: wanbo@mapgis.net



主要讲解Oracle Spatial以下几个内容

- Oracle 10g Spatial 新特性
- Oracle Spatial概念
- 空间数据类型和元数据
- 加载空间数据
- 空间数据的检索和查询
- 其他相关概念



- Oracle 10g Spatial 新特性
- Oracle Spatial概念
- 空间数据类型和元数据
- 加载空间数据
- 空间数据的检索和查询
- 其他相关概念

1. Oracle Spatial 简介

- 2004年二月中旬，Oracle公司开始运营其数据库软件的最新版本，虽然Oracle公司一再强调其新数据库软件—Oracle 10g的强大管理能力和网格计算能力。但其新软件中也加入了关于空间数据的一些新特新。

- Oracle Spatial允许用户和应用程序开发人员将他们的空间数据紧密集成到企业应用程序中。Oracle Spatial根据相关数据的空间关系(例如，在给定的距离之内，存储位置与用户的接近程度，以及每个区域的销售收入)来进行分析。Oracle Spatial在行业标准的数据库中管理空间数据，从而导致了在数据服务器上进行的应用程序集成。这使得供应商工具和应用程序能够直接从Oracle数据库访问空间数据，从而提供互操作性并使最大程度地降低成本。

1.1 Oracle 10g Spatial 新特性

1、GeoRaster

GeoRaster是Oracle Spatial增加的一个功能。可以帮助我们完成 georaster数据，georaster数据是指栅格图形数据以及相关的空间矢量几何数据和元数据。GeoRaster提供Oracle Spatial 数据类型和一个对象关系方案来存储可以对应到地球表面位置和本地坐标系统上的多维栅格层和数字图像；

2、拓扑和网络管理

Oracle Spatial的拓扑和网络管理功能可以使我们用来作拓扑文件和网文件

3、空间数据分析和挖掘

我们可以使用Oracle数据挖掘（ODM）应用中的一些新的空间数据分析和挖掘子程序

1.1 Oracle 10g Spatial 新特性

4、Geocoding

可以对没有格式化的地址数据进行地理编码；

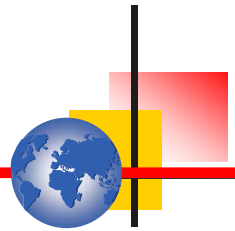
5、广泛采取R数索引，逐渐丢弃四叉数索引；

6、添加了一些新的空间操作函数、其他函数（GML支持函数）和操作符；

7、完善了一些函数，例如buffer函数可以生成一个多边形的内缓冲区

8、为空间数据提供了Java访问接口；

9、Oracle 10g spatial 不再支持关系几何模型，只支持对象关系模型。



- Oracle 10g Spatial 新特性
- Oracle Spatial概念
- 空间数据类型和元数据
- 加载空间数据
- 空间数据的检索和查询
- 其他相关概念

Oracle Spatial概念

- Oracle spatial通常被称为Spatial，即Oracle空间组件，它提供了一个基于SQL的方案和函数集来储存、检索、更新、查询Oracle数据库中的空间数据集。Spatial包含一下一些组件：

- 一套用来描述空间数据存储、语义和语法的方案；
- 一套空间索引机制；
- 一组操作符和函数用来进行面相关的查询、空间相交查询和其他空间分析操作；
- 管理模块。

2. Oracle Spatial概念

对Oracle Spatial概念 主要介绍一下几点：

- 对象关系模型
- 数据类型
- 数据模型
- 查询模型
- 数据索引
- 空间关系

2.1 对象关系模型

- Oracle 10g spatial 不再支持关系几何模型，只支持对象关系模型。在对象关系模型中用SDO_GEOMETRY类型列来描述SDO_GEOMETRY类型数据（几何图形数据）。每个实体的信息都写在表中的一行中。对象关系模型协同支持几何类型的SQL语句一起工作，来处理符合Open GIS ODBC/SQL规格的空间数据表。

- 对象关系模型优点

  1、支持多种几何数据类型，包括弧段、圆、复合多边形、复合线窜和优化矩形；

  2、能够方便的在空间数据查询时创建和维护索引；

  3、数据库自身维护索引；

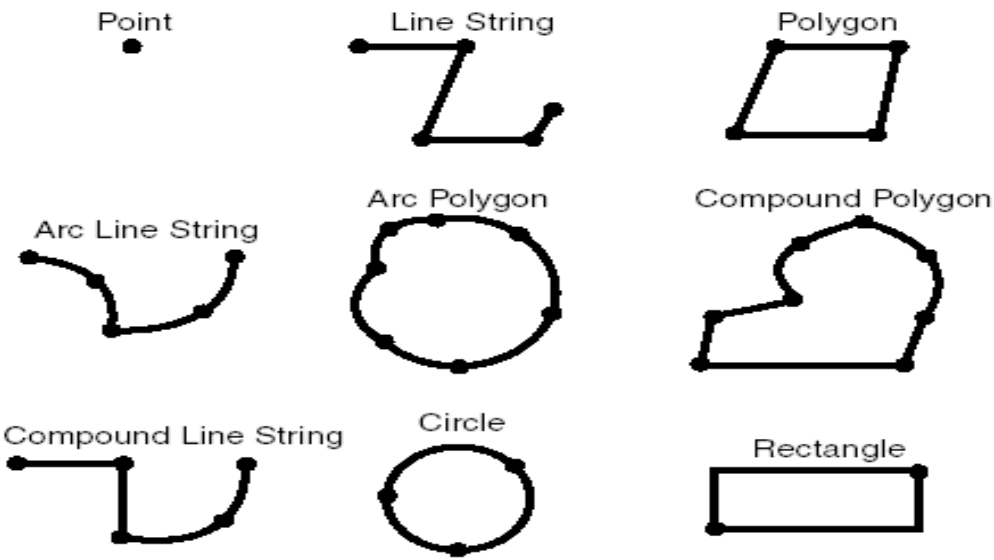
  4、几何模型存储在单行单列中；最佳的执行表现。

2.2 数据类型

- 点（串）　　Points and point clusters
- 线串　　Line strings
- 多边形　　*n*-point polygons
- 弧段　　Arc line strings (All arcs are generated as circular arcs.)
- 弧段多边形　　Arc polygons
- 复合多边形　　Compound polygons
- 复合线串　　Compound line strings
- 圆　　Circles
- 优化矩形　　Optimized rectangles

2.1 对象关系模型

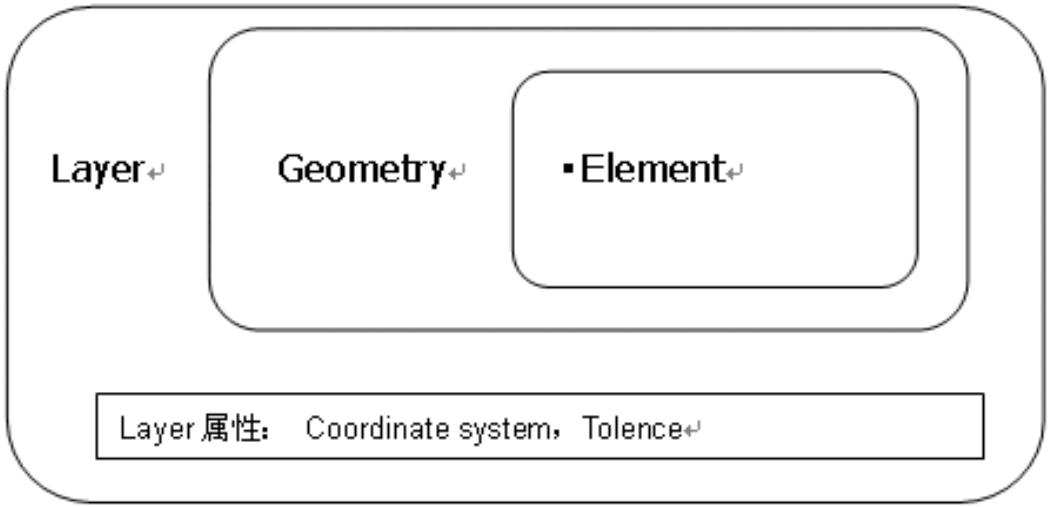
Figure 1–1 Geometric Types



- Oracle Spatial也支持三维和四维的空间数据对象数据的存储和索引，不过空间数据对象的维数要在空间数据对象被定义的时候给出。

2.3 数据模型

- Oracle 中的空间数据模型是一个层次结构，依次由层（Layer）、空间数据对象（Geometry）、元素（Element）构成。其中Element可以是Spatial支持的任何一种空间数据对象元素（点、线和多边形），一个空间数据对象有一个或者多个同类或异类的Element组成，Layer有具有相同属性的Geometry组成。

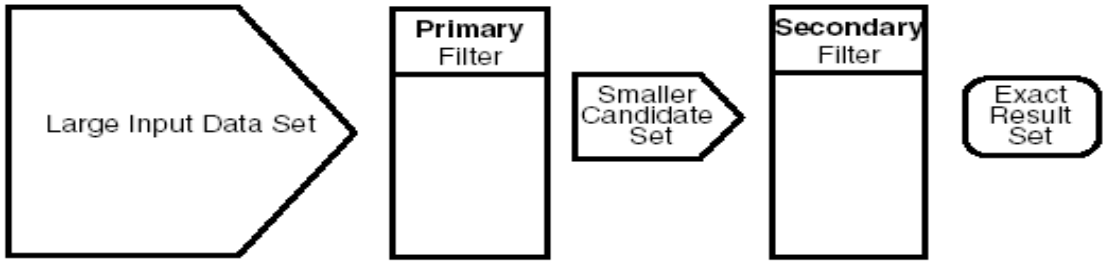


- Coordinate System，坐标系统，又称空间参考系统，负责指定位置的坐标和确定一组坐标之间的关系，可以让一组坐标对应到真实世界里的具体位置中。Tolence是容忍值，即判断两个空间数据对象是否分离的最小距离值。

2.4 查询模型

- Oracle spatial提供了两层查询模型，第一层是primary filter 初级过滤，第二层是secondary filter精确过滤。

Figure 1–2 Query Model



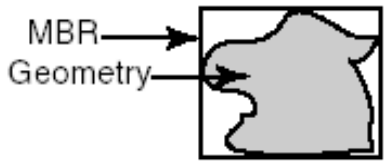
- Primary Filter初级过滤，速度比较快，而且可以满足我们大多数的的查询要求，比如进行窗口查询，显示指定范围内的空间数据。
- Secondary Filter精确过滤，主要用来确定多个空间数据对象之间的相关位置关系，例如两个实体之间的拓扑关系和距离。

2.5 数据索引

- 支持R树索引和四叉树索引。

  根据这两种索引的性能表形，在Oracle 10g Spatial 中，强烈推荐使用R树索引，尽量不要用四叉树索引。
- 最高可支持四维空间数据的索引。
- R树索引把每一个几何实体近似看作其最小外包矩形（MBR Minimum Bounding Rectangle）

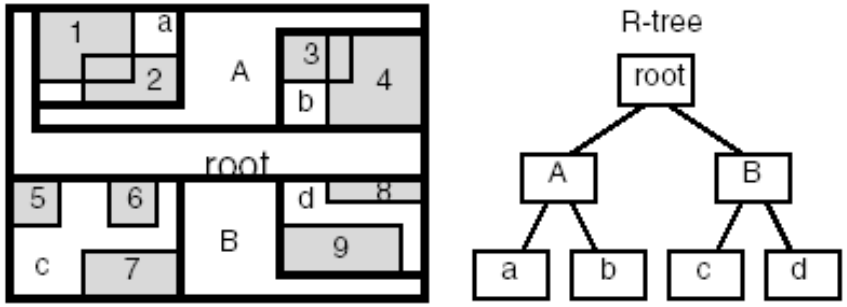
Figure 1–3 MBR Enclosing a Geometry



2.5 数据索引

- 在一个层中，R树索引为把每个空间数据对象的最小MBR建立一个层次结构。

Figure 1–4 R-Tree Hierarchical Index on MBRs



- 随着我们不断的插入和更新数据，R树索引的质量会随之下降，我们可以用SDO_TUNE.QUALITY_DEGRADATION函数来判断索引的状态，如果返回值大于2，那么我们就应该考虑用ALTER INDEX REBUILD函数来重新构建索引。

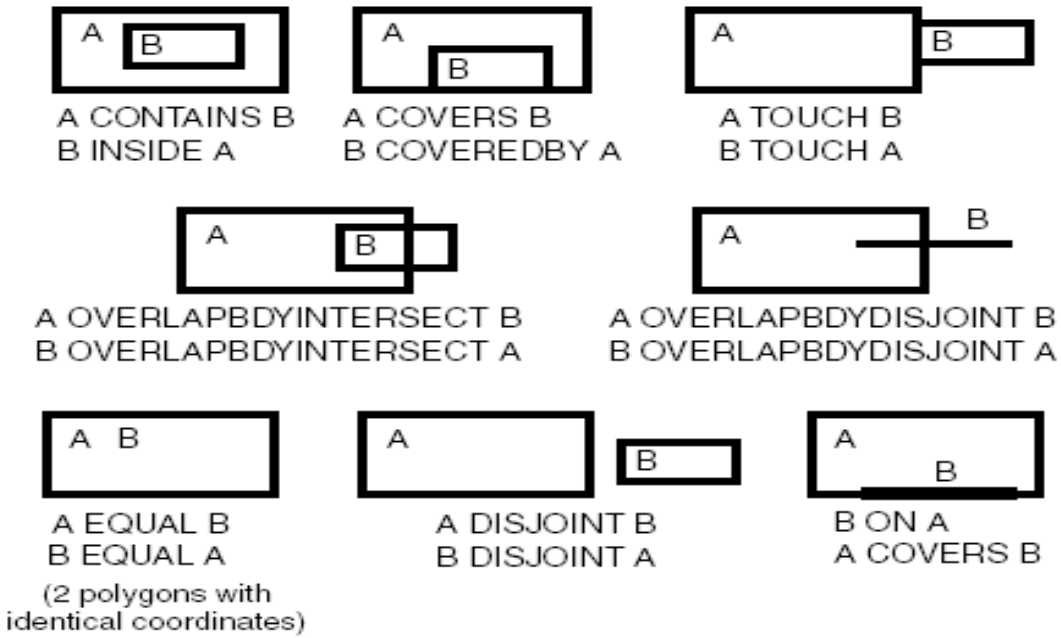
2.6 空间关系

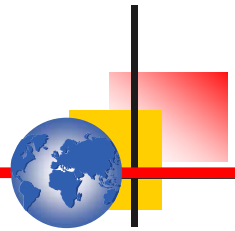
- SDO_RELATE 判断拓扑关系：
- 返回值：
- **DISJOINT** -- The boundaries and interiors do not intersect.
- **TOUCH** -- The boundaries intersect but the interiors do not intersect.
- **OVERLAPBDYDISJOIN**T -- The interior of one object intersects the boundary and interior of the other object, but the two boundaries do not intersect. This relationship occurs, for example, when a line originates outside a polygon and ends inside that polygon.
- **OVERLAPBDYINTERSECT** -- The boundaries and interiors of the two objects intersect.
- **EQUAL** -- The two objects have the same boundary and interior.
- **CONTAINS** -- The interior and boundary of one object is completely contained in the interior of the other object.
- **COVERS** -- The interior of one object is completely contained in the interior or the boundary of the other object and their boundaries intersect.
- **INSIDE** -- The opposite of CONTAINS. A INSIDE B implies B CONTAINS A.
- **COVEREDBY** -- The opposite of COVERS. A COVEREDBY B implies B COVERS A.
- **ON** -- The interior and boundary of one object is on the boundary of the other object (and the second object covers the first object). This relationship occurs, for example, when a line is on the boundary of a polygon. _ ANYINTERACT -- The objects are non-disjoint.
- **ANYINTERACT** -- The objects are non-disjoint.

2.6 空间关系

- SDO_WITHIN_DISTANCE 计算距离；
- SDO_NN 判断最近物体；
- 除了上面列举的函数外，Oracle Spatial还有很多函数用来进行空间分析。

Figure 1–6 Topological Relationships





- Oracle 10g Spatial 新特性
- Oracle Spatial概念
- 空间数据类型和元数据
- 加载空间数据
- 空间数据的检索和查询
- ． 其他相关概念

空间数据类型和元数据

Oracle 空间组件包含一系列的对象数据类型、类型方法、操作符、函数和用到这些类的存储过程。一个空间数据对象作为一个对象，存储在单一行里的SDO_GEOMETRY类型列里。本章主要讲述一下几个内容：

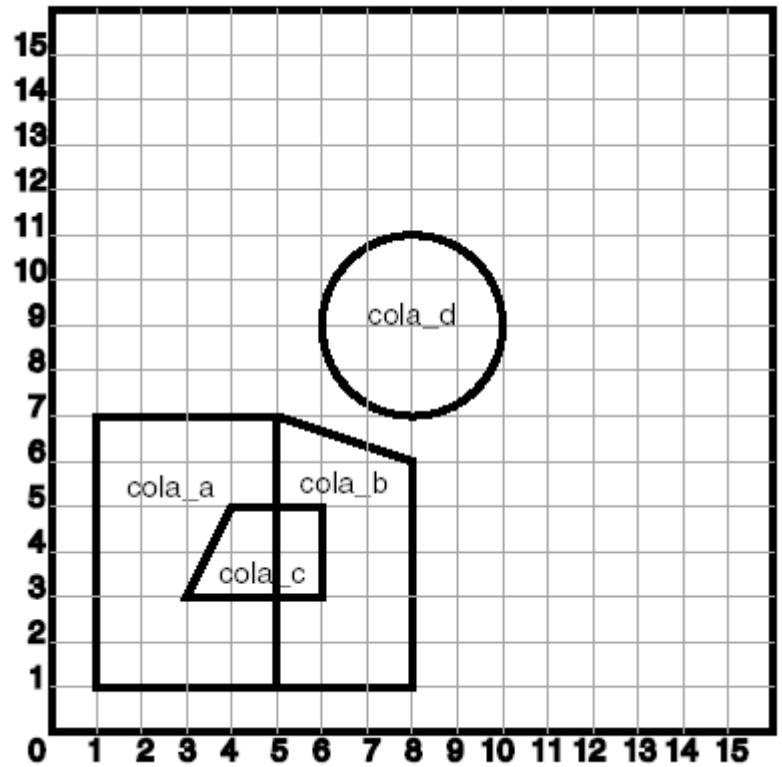
- 引例
- 空间数据对象
- 空间数据对象实例
- 元数据视图

3.1 例子

该例子为四个多边形的如下图，对其空间数据的存储分为一下几步：

1、创建表（COLA_MARKETS）用来存储空间数据；

```
CREATE TABLE cola_markets (
mkt_id NUMBER PRIMARY KEY,
name VARCHAR2(32),
shape SDO_GEOMETRY);
```

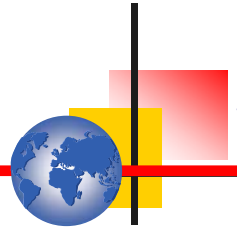


3.1 例子

- 2、在表中增加四条记录（如上图所示，cola_a,cola_b,cola_c,cola_d）

```
INSERT INTO cola_markets VALUES(
1,
'cola_a',
SDO_GEOMETRY(
2003, -- two-dimensional polygon
NULL,
NULL,
SDO_ELEM_INFO_ARRAY(1,1003,3),   -- one rectangle (1003 = exterior)
SDO_ORDINATE_ARRAY(1,1, 5,7)      -- only 2 points needed to
-- define rectangle (lower left and upper right) with
-- Cartesian-coordinate data
)
);

INSERT INTO cola_markets VALUES(
2,
'cola_b',
SDO_GEOMETRY(
2003, -- two-dimensional polygon
NULL,
NULL,
SDO_ELEM_INFO_ARRAY(1,1003,1), -- one polygon (exterior polygon ring)
SDO_ORDINATE_ARRAY(5,1, 8,1, 8,6, 5,7, 5,1)
)
);
```



3.1 例子

```
INSERT INTO cola_markets VALUES(
3,
'cola_c',
SDO_GEOMETRY(
2003, -- two-dimensional polygon
NULL,
NULL,
SDO_ELEM_INFO_ARRAY(1,1003,1), -- one polygon (exterior polygon ring)
SDO_ORDINATE_ARRAY(3,3, 6,3, 6,5, 4,5, 3,3)
)
);

INSERT INTO cola_markets VALUES(
4,
'cola_d',
SDO_GEOMETRY(
2003, -- two-dimensional polygon
NULL,
NULL,
SDO_ELEM_INFO_ARRAY(1,1003,4), -- one circle
SDO_ORDINATE_ARRAY(8,7, 10,9, 8,11)
)
);
```

3.1 例子

3、更新USER_SDO_GEOM_METADATA表，更新维数信息；

```
INSERT INTO USER_SDO_GEOM_METADATA
VALUES (
'cola_markets',
'shape',
SDO_DIM_ARRAY( -- 20X20 grid
SDO_DIM_ELEMENT('X', 0, 20, 0.005),
SDO_DIM_ELEMENT('Y', 0, 20, 0.005)
),
NULL -- SRID
);
```

4、建立索引

```
CREATE INDEX cola_spatial_idx
ON cola_markets(shape)
INDEXTYPE IS MDSYS.SPATIAL_INDEX;
```

3.1 例子

5、执行空间查询。

```
-- Return the topological intersection of two geometries.
SELECT SDO_GEOM.SDO_INTERSECTION(c_a.shape, c_c.shape, 0.005)
FROM cola_markets c_a, cola_markets c_c
WHERE c_a.name = 'cola_a' AND c_c.name = 'cola_c';
-- Do two geometries have any spatial relationship?
SELECT SDO_GEOM.RELATE(c_b.shape, 'anyinteract', c_d.shape, 0.005)
FROM cola_markets c_b, cola_markets c_d
WHERE c_b.name = 'cola_b' AND c_d.name = 'cola_d';
-- Return the areas of all cola markets.
SELECT name, SDO_GEOM.SDO_AREA(shape, 0.005) FROM cola_markets;
-- Return the area of just cola_a.
SELECT c.name, SDO_GEOM.SDO_AREA(c.shape, 0.005) FROM cola_markets c
WHERE c.name = 'cola_a';
-- Return the distance between two geometries.
SELECT SDO_GEOM.SDO_DISTANCE(c_b.shape, c_d.shape, 0.005)
FROM cola_markets c_b, cola_markets c_d
WHERE c_b.name = 'cola_b' AND c_d.name = 'cola_d';
-- Is a geometry valid?
SELECT c.name, SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT(c.shape, 0.005)
FROM cola_markets c WHERE c.name = 'cola_c';
SDO_GEOMETRY Object Type
-- Is a layer valid? (First, create the results table.)
CREATE TABLE val_results (sdo_rowid ROWID, result VARCHAR2(2000));
CALL SDO_GEOM.VALIDATE_LAYER_WITH_CONTEXT('COLA_MARKETS', 'SHAPE',
'VAL_RESULTS', 2);
SELECT * from val_results;
```

3.2 空间数据对象

- 空间数据对象类型即SDO_GEOMETRY类型，我们把空间数据存储在数据库中的一行的SDO_GEOMETRY类型列里。SDO_GEOMETRY是Oracle Spatial提供的数据类型。其类型定义如下：

```
CREATE TYPE sdo_geometry AS OBJECT (
SDO_GTYPE NUMBER,
SDO_SRID NUMBER,
SDO_POINT SDO_POINT_TYPE,
SDO_ELEM_INFO SDO_ELEM_INFO_ARRAY,
SDO_ORDINATES SDO_ORDINATE_ARRAY);
```

- SDO_GEOMETRY的定义中由涉及到了三个数据类型:
SDO_POINT_TYPE，SDO_ELEM_INFO_ARRAY，SDO_ORDINATE_ARRAY，他们的定义如下：

```
CREATE TYPE sdo_point_type AS OBJECT (
X NUMBER,
Y NUMBER,
Z NUMBER);
CREATE TYPE sdo_elem_info_array AS VARRAY (1048576) of NUMBER;
CREATE TYPE sdo_ordinate_array AS VARRAY (1048576) of NUMBER;
```

3.2 空间数据对象

- **SDO_GTYPE** 是NUMBER类型的值，由四个数字组成：abcd。这四个数字组合起来决定这个空间数据对象是何种类型。

  （1）a ： 表示空间数据对象的维数；

  （2）b ： 表示在线形参考系统中一个空间数据对象的线形参考量度维，在非线形参考系统，此位默认为0；

  （3）cd ： 表示几何类型，在下表中将详细说明。

SDO_GTYPE	几何类型	具体描述
ab00	未知类型	系统忽略此类型
ab01	点	包含一个点
ab02	线	可以是直线、弧段或者两者都有
ab03	多边形	一个多边形，可以含洞（注一）
ab04	集合体	不同种元素的集合体（注二）
ab05	点串	一个或者多个点
ab06	复合线	一条或者多条线
ab07	复合多边形	包含多个相离的多边形，
注一：含洞的多边形，先输入外多边形的数据； 注二：集合体中的多边形可以是相离的。		

3.2 空间数据对象

- **SDO_SRID** 是NUMBER类型的值，指定和空间数据相关的坐标系统。如果值为空（NULL），则表示没有于指定坐标系统。如果值不为空，那么一定要赋系统给定的值。
- **SDO_POINT** 是SDO_POINT_TYPE类型的值。是由三个坐标值（x,y,z）做成的一个结构体，在二维系统里只考虑前面两个值（x,y）。如果空间数据对象只是一个点时，就要用给这个值赋值，此时后面两个值SDO_ELEM_INFO，SDO_ORDINATES就为空。如果所定义的空间数据对象不是一个点，那么系统就忽略此值，给其赋值为NULL。
- **SDO_STARTING_OFFSET** 表示几何元素的第一个坐标值存储在SDO_ORDINATES时的偏移量。偏移量从1开始，而不是从0开始
- **SDO_ELEM_INFO** 是SDO_ELEM_INFO_ARRAY类型的数组，就存储的是一组不定长的三元数组。即它存储值的个数必须是三的倍数，例如（1，3，2）或（1，1，2，2，1，2）。它的值用来描述SDO_ORDINATES里的坐标值是怎样被存储的。其中每个三元组的每个值的类型依次为：

  SDO_STARTING_OFFSET

  SDO_ETYPE

  SDO_INTERPRETATION

3.2 空间数据对象

- **SDO_ETYPE** 当其值为1，2，1003，2003时，指当前三元组所指的空间数据对象是基本元素（点，线，多边形）。其中1003和2003都是指多边形其意义为：

  1003：外多边形，坐标逆时针存储；

  2003：内多边形，坐标顺时针存储。

  1 ：点

  2 ：线

  当其值为4，1005，2005时，指当前三元组要分解为基本三元组。

  4：该三元组是复合线组成的，要分解成基本元素；

  1005：该三元组是复合外多边形；

  2005：该三元组是复合内多边形。

- **SDO_INTERPRETATION** ：该值的意义需要和SDO_ETYPE协作表达：

  如果SDO_ETYPE 为1，2，1003，2003时：该值表示该三元组组成元素的类型，具体见表；

  如果SDO_ETYPE 为4，1005，2005时：该值表示该复合三元组组成元素的个数；

- **SDO_ORDINATES** SDO_ORDINATE_ARRAY类型的数组，保存该空间数据对象的所有坐标值。

3.2 空间数据对象

- SDO_ETYPE三元组组成元素的类型

SDO_ETYPE	SDO_INTERPRETATION	Meaning
0	任何数字	不支持类型
1	1	点
1	N > 1	点簇，由 N 个点组成
2	1	由直线段组成的线段
2	2	由弧段组成的线段
1003 或 2003	1	由直线段组成的多边形
1003 或 2003	2	由弧段组成的多边形
1003 或 2003	3	由左下角和右上角点决定的矩形
1003 或 2003	4	圆
4	N > 1	线段，一部分由直线段组成，一部分由弧线组成。N 为子元素的个数。
1005 或 2005	N > 1	多边形，一部分由直线段组成，一部分由弧线组成。N 为子元素的个数。

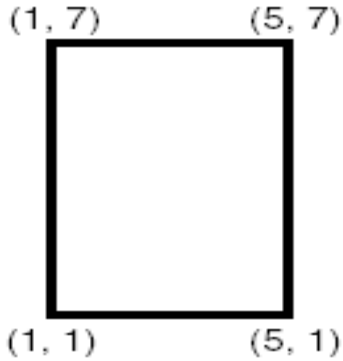
3.3 空间数据对象实例

- 以下是一些SDO_GEOMETRY类型的几何图形，以及其对应SDO_GEOMETRY类型的属性值。

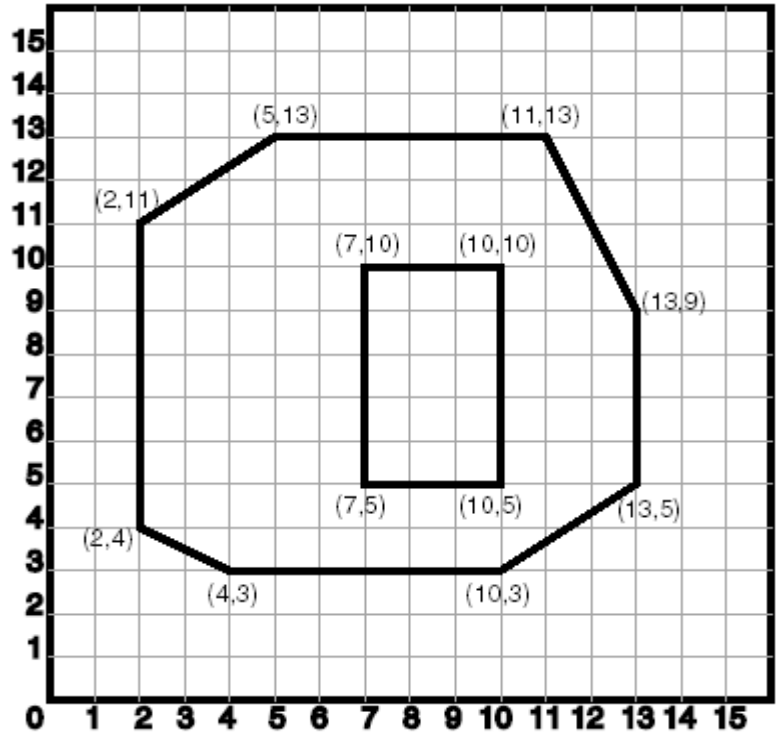
1、矩形
2、带洞的多边形
3、线
4、多边形

3.3.1 矩形

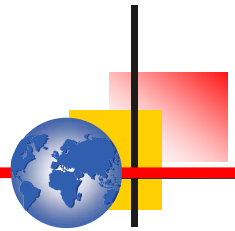
```
INSERT INTO cola_markets VALUES
(
1,
'cola_a',
SDO_GEOMETRY
(
2003,  //二维多边形
NULL,  //不是一个点
NULL,  //没有坐标系统
SDO_ELEM_INFO_ARRAY(1,1003,3), //外矩形，第一个点的坐标位置为1
SDO_ORDINATE_ARRAY(1,1, 5,7)
)
);
```



3.3.2 带洞的多边形

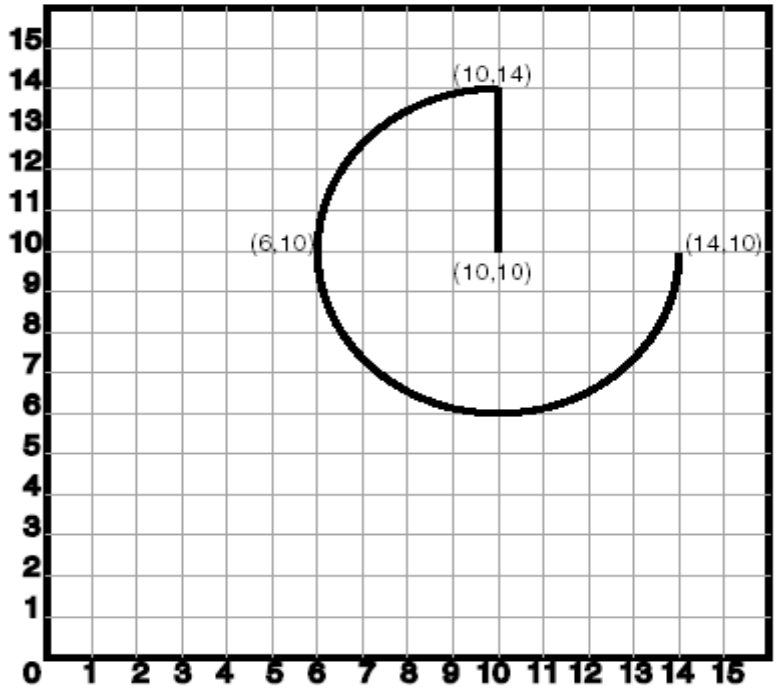



```
INSERT INTO cola_markets VALUES
(
10,
'polygon_with_hole',
SDO_GEOMETRY
(
2003,
NULL,
NULL,
SDO_ELEM_INFO_ARRAY(1,1003,1, 19,2003,1),
SDO_ORDINATE_ARRAY(2,4, 4,3, 10,3, 13,5, 13,9, 11,13, 5,13, 2,11, 2,4,
7,5, 7,10, 10,10, 10,5, 7,5)
)
);
```



- 本例中的图形是一个二维带有洞的多边形，SDO_GTYPE的值应为2003，SDO_SRID和SDO_POINT项的值都是空。再来看SDO_ELEM_INFO的值，因为这个几何实体是由两个子元素组成的，所以SDO_ELEM_INFO的值由两个三元组六个数值组成的。第一个三元组（1，1003，1）代表外面的由直线段组成的这个多边形的第一个坐标在SDO_ORDINATES的偏移量是1。第二个三元组（19，2003，1）中的19代表里面这个多边形的第一个坐标在SDO_ORDINATES的偏移量为19，后面两个数值（1003，1）查表可知其子元素是由直线段组成的多边形。

3.3.3 线

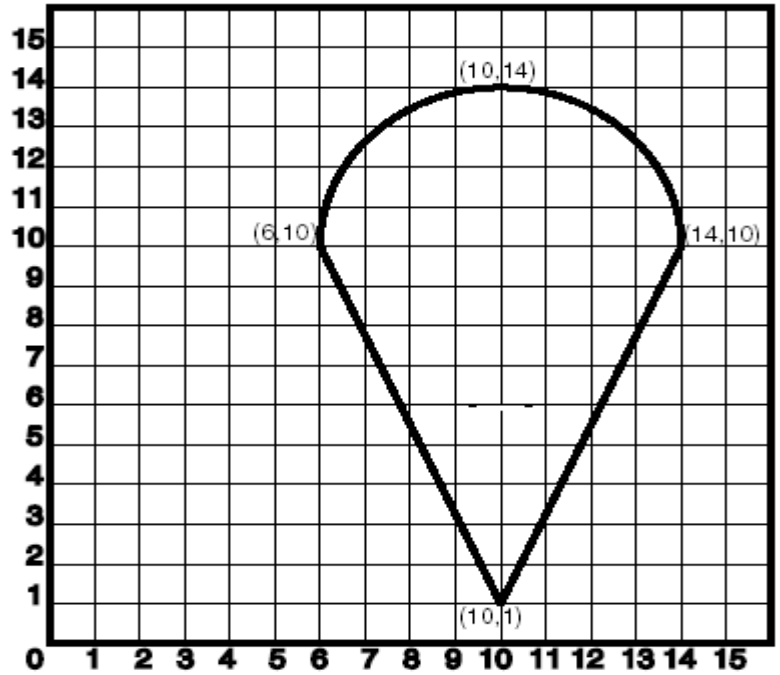



```
INSERT INTO cola_markets VALU
(
11,
'compound_line_string',
SDO_GEOMETRY
(
2002,
NULL,
NULL,
SDO_ELEM_INFO_ARRAY(1,4,2, 1,2,1, 3,2,2),
SDO_ORDINATE_ARRAY(10,10, 10,14, 6,10, 14,10)
)
);
```

3.3.3 线

- 本例中的图形是线段，一部分由直线段组成，一部分由弧线组成。那么查表可知SDO_GTYPE的值应为2002，SDO_SRID和SDO_POINT项的值都是空。再来看SDO_ELEM_INFO的值，因为这是一个复合几何实体是由两个不同种子元素组成的，所以SDO_ELEM_INFO的值由三个三元组九个数值组成的。第一个三元组（1，4，2）是头三元组，最后一个值2代表两个子元素。第二个三元组（1，2，1）中的1代表第一个子元素的第一个坐标在SDO_ORDINATES的偏移量，后面两个数值（2，1）结合在一起查表可知其第一个子元素是由直线段组成的线段。第三个三元组（3，2，2）中的3代表第二个子元素的第一个坐标在SDO_ORDINATES的偏移量，后面两个数值（2，2）结合在一起查table-2可知其第二个子元素是由由弧段组成的线段。

3.3.4 多边形




```
INSERT INTO cola_markets VALUES
(
12,
'compound_polygon',
SDO_GEOMETRY
(
2003,
NULL,
NULL,
SDO_ELEM_INFO_ARRAY(1,1005,2, 1,2,1, 5,2,2),
SDO_ORDINATE_ARRAY(6,10, 10,1, 14,10, 10,14, 6,10)
)
);
```

3.4 元数据视图

- 元数据是描述表或层的维数信息、最大最小边界值和每一委容忍度信息的数据，它被保存在MDSYS方案下的全局表中。每一个用户都用一下数据视图：

  USER_SDO_GEOM_METADATA 包含该用户的所有空间表的元数据信息。这是用户可以更新的唯一一个视图。用户必须把新建表的元数据信息添加到这个视图中。

  ALL_SDO_GEOM_MEATDATA 包含赋给该用户SELCET权限的所有空间表的元数据信息。

- 用户必须对这些视图进行维护，把每个表中用来存储空间数据的列名和其他元数据插入到USER_SDO_GEOM_METADATA中，oracle patial负责使ALL_SDO_GEOM_METADATA中的数据于USER_SDO_GEOM_METADATA一致。

每个元数据视图都包含如下定义：

```
(
TABLE_NAME      VARCHAR2(32),
COLUMN_NAME    VARCHAR2(32),
DIMINFO          SDO_DIM_ARRAY,
SRID              NUMBER
);
```

3.4 元数据视图

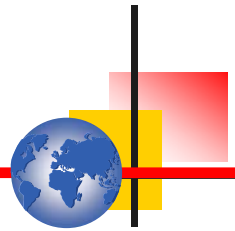
- TABLE_NAME 空间表的名字，此表中必须有SDO_GEOMETRY类型数据；
- COLUMN_NAME 表中存储SDO_GEOMETRY数据的列名；
- DIMINFO 是SDO_DIM_ARRAY类型数据，SDO_DIM_ARRAY定义如下：

  Create Type SDO_DIM_ARRAY as VARRAY(4) of SDO_DIM_ELEMENT;

  其中SDO_DIM_ELEMENT定义如下：

  Create Type SDO_DIM_ELEMENT as OBJECT (
  SDO_DIMNAME VARCHAR2(64), //维名：例如x
  SDO_LB NUMBER, //最小范围
  SDO_UB NUMBER, //最大范围
  SDO_TOLERANCE NUMBER); //容忍度

  其中DIMINFO中包含有n个SDO_DIM_ELEMENT对象，n为表中数据的维数，如果表中的数据是2维的，那么DIMINFO中就有2个SDO_DIM_ELEMENT对象，为每一维添加信息。
- SRID 空间坐标系统值，可以为NULL，如果赋值，必须是系统指定的值。



- Oracle 10g Spatial 新特性
- Oracle Spatial概念
- 空间数据类型和元数据
- 加载空间数据
- 空间数据的检索和查询
- 其他相关概念

4. 加载空间数据

- 本章讲述怎样把空间数据装载到Oracle数据库中，包括把SDO_GEOMETRY类型空间数据对象存到指定的数据表中去。当把数据装载到数据库以后，就可以对其建立索引和空间分析和查询。
  有两种装载数据的方法：

  1、批量装载： 用SQL*Loader工具大批量的装载数据；

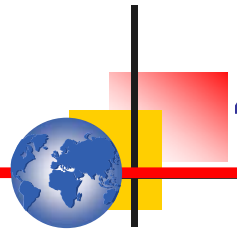
  2、语句插入： 用SQL语句中的INSERT命令向数据库中加入数据。

4.1 批量装载

- 批量装载使用SQL*Loader工具完成的，可以把大量的ASCII码数据装入数据库中，下面将举例进行说明。

  下面是向cola_markets 表中批量装载数据的控制代码：

```
LOAD DATA
INFILE *
TRUNCATE
CONTINUEIF NEXT(1:1) = '#'
INTO TABLE COLA_MARKETS
FIELDS TERMINATED BY '|'
TRAILING NULLCOLS (
mkt_id INTEGER EXTERNAL,
name CHAR,
```



4.1 批量装载

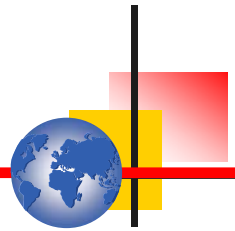
```
shape COLUMN OBJECT
(
SDO_GTYPE INTEGER EXTERNAL,
SDO_ELEM_INFO VARRAY TERMINATED BY '|/'
(elements FLOAT EXTERNAL),
SDO_ORDINATES VARRAY TERMINATED BY '|/'
(ordinates FLOAT EXTERNAL)
)
)
begindata
1|cola_a|
#2003|1|1003|3|/
#1|1|5|7|/
2|cola_b|
#2003|1|1003|1|/
#5|1|8|1|8|6|5|7|5|1|/
3|cola_c|
#2003|1|1003|1|/
#3|3|6|3|6|5|4|5|3|3|/
4|cola_d|
#2003|1|1003|4|/
#8|7|10|9|8|11|/
```

4.2 单行装载

- 出了上面的情况还可以用标准的SQL语句把数据插入到数据库的表中。

例如：

```
INSERT INTO cola_markets VALUES
(
1,
'cola_a',
SDO_GEOMETRY
(
2003,  //二维多边形
NULL,  //不是一个点
NULL,  //没有坐标系统
SDO_ELEM_INFO_ARRAY(1,1003,3), //外矩形，第一个点的坐标位置为1
SDO_ORDINATE_ARRAY(1,1, 5,7)
)
);
```



- Oracle 10g Spatial 新特性
- Oracle Spatial概念
- 空间数据类型和元数据
- 加载空间数据
- 空间数据的检索和查询
- 其他相关概念

5. 空间数据的检索和查询

- 当我们把数据装入Oracle 数据库中以后，我们应该为这个空间数据建立索引，方便我们进行查询。本章要讲的内容如下：

1、**创建空间索引**；

2、在很好的理解Oracle spatial 空间数据查询模型的基础上，有效的进行**空间数据查询**。

5.1 创建索引

- 我们在创建索引时，如果没有指定任何关于四叉树的参数，那么系统会为我们创建R树索引。

```
CREATE INDEX territory_idx ON territories (territory_geom)
INDEXTYPE IS MDSYS.SPATIAL_INDEX;
```

- 我们可以为不同坐标系统的数据创建索引，如果我们为使用测量坐标的数据创建索引的时候应该使用测量坐标数据索引，即在创建索引的时候加上'geodetic=false'；

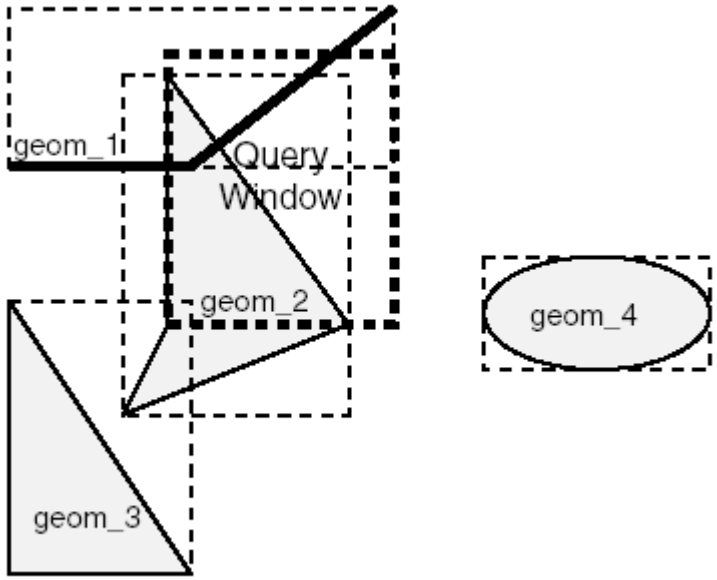
5.2 空间查询

- Oracle spatial提供了两层查询模型，第一层是primary filter 初级过滤，第二层是secondary filter精确过滤。

  典型的空间查询就是在一个给定的窗口中查找出所有的空间数据对象，这个给定的动态窗口即矩形没有在系统中定义，我们需要在使用它之前对他进行定义。

  在下面这个例子中，我们可以看到许多的带有外包矩形的空间数据对象，以及一个用虚线画出来的查询窗口。

- 右图中，查询窗口与geom_1和geom_2相交，并且与geom_3的最小外包矩形相交，但并不与geom_3图形相交。在Oracle Spatial中进行初级过滤（Primary Filter）时，只是对空间数据对象的外包矩形进行操作，返回所有满足要求的对象几何。



5.2.1 初级过滤

- 下面是用初级过滤操作符SDO_FILTER进行窗口查询操作，其中SDO_FILTER的原型为：

  SDO_FILTER(geometry1 SDO_GEOMETRY, geometry2 SDO_GEOMETRY)

  geometry1为表中存放SDO_GEOMETRY类型数据的列名，而且此列必须建立的空间索引。
  geometry2是一个SDO_GEOMETRY类型的对象，可以来源于表中，也可以是临时定义的，如果来源于表，表中的此列数据不需要建立索引。

- 以下是具体代码，将返回geom_1，geom_2和geom_3。

```
SELECT A.Feature_ID FROM TARGET A
WHERE sdo_filter(A.shape, SDO_geometry(2003,NULL,NULL,
SDO_elem_info_array(1,1003,3),
SDO_ordinate_array(x1,y1, x2,y2))) = 'TRUE';
```

5.2.2 初级过滤和精确过滤

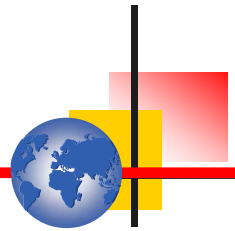
- SDO_RELATE是分别执行初级过滤和精确过滤的操作，此操作符只能用于二维数据坐标系统中。其原型定义如下：

  SDO_RELATE(geometry1 SDO_GEOMETRY,
  geometry2 SDO_GEOMETRY, param VARCHAR2)

  geometry1为表中存放SDO_GEOMETRY类型数据的列名，而且此列必须建立的空间索引。
  geometry2是一个SDO_GEOMETRY类型的对象，可以来源于表中，也可以是临时定义的，如果来源于表，表中的此列数据不需要建立索引。
  param，是定义精确过滤选项，可以是任何系统定义的值，如‘mask=anyinteract’。

- 具体代码如下，此时返回geom1和geom2。

```
SELECT A.Feature_ID FROM TARGET A
WHERE sdo_relate(A.shape,
SDO_geometry(2003,NULL,NULL,
SDO_elem_info_array(1,1003,3),
SDO_ordinate_array(x1,y1, x2,y2)),
'mask=anyinteract') = 'TRUE';
```



- Oracle 10g Spatial 新特性
- Oracle Spatial概念
- 空间数据类型和元数据
- 加载空间数据
- 空间数据的检索和查询
- 其他相关概念

6. 其他相关概念

- 地理编码地址数据
- 坐标系统
- 线形参考系统

6.1 地理编码地址数据

- 了解地理编码数据（Geocoding Address Data ）之前，我们先来看看地理编码（Geocoding），地理编码是把空间位置（如经纬度坐标）和邮政地址建立联系的过程。那么地理编码数据就是其空间位置和邮政地址建立了联系的空间数据。
- 其中的地址数据可以系统预定义结构化的，也可以非结构化的。用SDO_GEO_ADDR类型数据描述空间对象的结构化地址。SDO_GEO_ADDR Type 字段表见下表

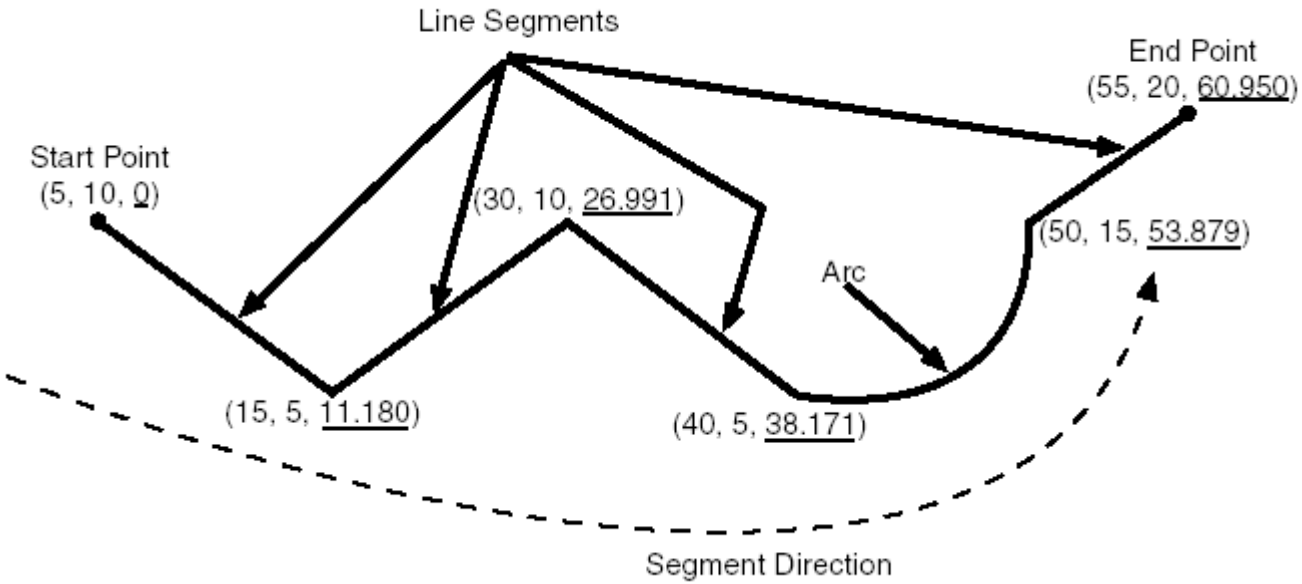
Attribute	Data Type	Description
Id	NUMBER	(Not used.)
AddressLines	SDO_KEYWORDARRAY	Address lines.
PlaceName	VARCHAR2(200)	(Not used.)
… …	… …	… …
	VARCHAR2(30)	Match code
Longitude	NUMBER	Match mode
Latitude	NUMBER	Longitude coordinate value.
		Latitude coordinate value.

6.2 坐标系统

Oracle Spatial 支持一下几种坐标系统：

- **Cartesian Coordinates**
- 笛卡儿坐标系，最常见的坐标系。
- **Geodetic Coordinates (Geographic Coordinates)**
- 测量坐标系，或者图形坐标系里表示的是角度坐标，比如经纬度。
- **Projected Coordinates**
- 投影坐标系，里面的坐标是从球面上投影到平面上的笛卡儿坐标。
- **Local Coordinates**
- 本地坐标系，其坐标值是与地球表面无关的笛卡儿坐标

6.3 线形参考系统


Line Segments
End Point
(55, 20, 60.950)
Start Point
(5, 10, 0)
(30, 10, 26.991)
(50, 15, 53.879)
Arc
(15, 5, 11.180)
(40, 5, 38.171)
Segment Direction