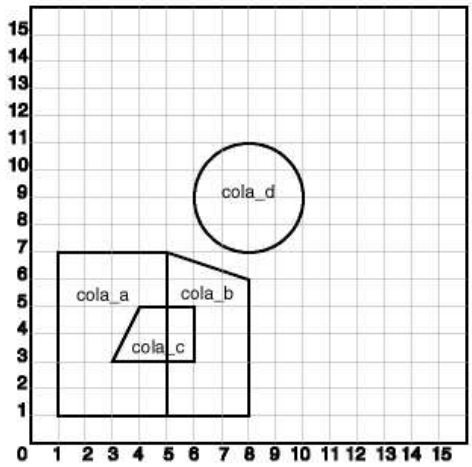
테이블 생성 및 삽입

간단한 예제를 통해 오라클에 공간 데이터를 삽입하고 검색하는 방법에 대해 알아보겠습니다. 삽입하려는 공간데이터는 [그림 1]과 같은 4개의 서로 다른 모양과 위치를 갖는 콜라 판매점에 대한 데이터입니다.



[그림 1] cola_market 예제 데이터

우선 데이터를 저장하기 위해 [그림 2]의 SQL문을 사용해서 cola_markets 라는 이름을 가진 테이블을 만들어 보겠습니다. cola_markets 테이블은 ① 숫자로 된 마켓id, ② 최대 32글자까지의 문자로 기술되는 판매점 이름, 그리고 ③ 판매점의 위치나 모양 등의 공간 정보를 저장하는 shape 이라는 3개의 칼럼으로 구성됩니다.

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

[그림 2] 콜라 판매점 테이블 생성

이제 공간 정보가 저장되는 칼럼인 shape의 오브젝트 타입인 **SDO_GEOMETRY** 에 대해서 조금 더 자세히 알아보겠습니다. SDO_GEOMETRY는 oracle에서 공간 정보를 저장하기 위해 사용하는 오브젝트 타입입니다. 공간 객체를 테이블의 행에 저장할 때, 객체의 공간적 속성 값은 SDO_GEOMETRY라는 타입을 갖는 칼럼에 저장됩니다. SDO_GEOMETRY 타입으로 저장되는 값들은 spatial table, 혹은 spatial geometry table로 불리는 별개의 테이블에 저장되며, 공간 객체의 primary key에 의해 식별됩니다.
Oracle spatial and Graph는 SDO_GEOMETRY를 [그림 3]와 같이 정의합니다.

```
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);
```

[그림 3] SDO_GEOMETRY 구조

SDO_GTYPE

SDO_GTYPE 속성은 4개의 digit으로 이루어진 DLTT format을 사용하며, geometry의 타입을 정의합니다. 각각의 digit이 의미하는 바는 다음과 같다.

Digit	Meaning
D	Number of dimensions(2, 3, or 4)
L	Linear referencing measure
TT	Types of object

두번째 digit인 Linear referencing measure는 벡터를 구성하는 line segment나 그래프의 vertex에 속성이나 이벤트를 추가하고 관련된 연산을 수행하는 방법을 정의합니다. 우리는 기본적으로 이 기능을 사용하지 않을 것이기 때문에 이 부분은 지금 굳이 이해할 필요는 없습니다. 사용하지 않을 경우 기본값은 0으로 설정합니다. Types of object는 객체의 geometry type을 정의하는 부분으로 [그림 4]와 같은 10가지 옵션이 있습니다. 표에서 좌측의 value 부분의 D는 앞서 말한 차원 수에 해당하는 digit입니다. 예를 들어 SDO_GTYPE의 값이 2003이라면, 이것은 2차원 폴리곤을 의미합니다.

Table 2-1 Valid SDO_GTYPE Values

Value	Geometry Type	Description
*DL*00	UNKNOWN_GEOMETRY	Spatial and Graph ignores this geometry.
*DL*01	POINT	Geometry contains one point.
*DL*02	LINE or CURVE	Geometry contains one line string that can contain straight or circular arc segments, or both. (LINE and CURVE are synonymous in this context.)
*DL*03	POLYGON or SURFACE	Geometry contains one polygon with or without holes,[Foot 1] or one surface consisting of one or more polygons. In a three-dimensional polygon, all points must be on the same plane.
*DL*04	COLLECTION	Geometry is a heterogeneous collection of elements. COLLECTION is a superset that includes all other types.
*DL*05	MULTIPOINT	Geometry has one or more points. (MULTIPOINT is a superset of POINT.)
*DL*06	MULTILINE or MULTICURVE	Geometry has one or more line strings. (MULTILINE and MULTICURVE are synonymous in this context, and each is a superset of both LINE and CURVE.)
*DL*07	MULTIPOLYGON or MULTISURFACE	Geometry can have multiple, disjoint polygons (more than one exterior boundary), or surfaces (MULTIPOLYGON is a superset of POLYGON, and MULTISURFACE is a superset of SURFACE.)
*DL*08	SOLID	Geometry consists of multiple surfaces and is completely enclosed in a three-dimensional space. Can be a cuboid or a frustum.
*DL*09	MULTISOLID	Geometry can have multiple, disjoint solids (more than one exterior boundary). (MULTISOLID is a superset of SOLID.)

[그림 4] SDO_GTYPE value

SDO_SRID

SDO_SRID는 geometry의 좌표계를 의미합니다. SDO_SRID가 null이라면 일반적으로 생각하는 수직 좌표계인 cartesian coordinate system을 사용합니다. 오라클에서는 이 밖에도 geodetic coordinate, projected coordinate 등 다양한 좌표계를 지원하지만, 실습 단계에서는 이해하지 않으셔도 무방합니다. 오라클에서 지원하는 좌표계에 대한 정보는 다음 링크에서 알아볼 수 있습니다.

https://docs.oracle.com/database/121/SPATL/coordinate-systems-spatial-reference-systems.htm#SPATL050

SDO_POINT

SDO_POINT는 x, y, z의 세 가지 속성으로 정의되는 3차원 포인트 객체입니다. SDO_GEOMETRY 중 남은 두 가지 속성인 SDO_ELEM_INFO, SDO_ORDINATES의 값이 null이라면, SDO_POINT에 정의된 좌표값이 공간 객체의 좌표로 정의됩니다. 다른 경우에는, 이 속성은 무시됩니다.

SDO_ELEM_INFO

SDO_ELEM_INFO는 3가지 숫자로 이루어진 배열로 정의됩니다. 배열에 정의되는 값은 각각 SDO_STARTING_OFFSET, SDO_ETYPE, SDO_INTERPRETATION 으로, 이후에 기술할 SDO_ORDINATES에 저장된 좌표 값의 배열들을 해석하기 위한 메타 정보를 저장합니다. OFFSET은 첫 번째 ordinate가 저장된 주소를 말하며 기본적으로 0이 아닌 1부터 시작합니다. 간단한 도형을 만들 때에는 크게 신경쓰지 않아도 되는 부분이지만 여러개의 폴리곤이나 라인으로 이루어진 복잡한 도형이라면 각각의 element들이 시작되는 부분의 배열 주소를 적어줍니다. 이 부분은 잠시 후 예제를 통해 다시 알아보겠습니다.

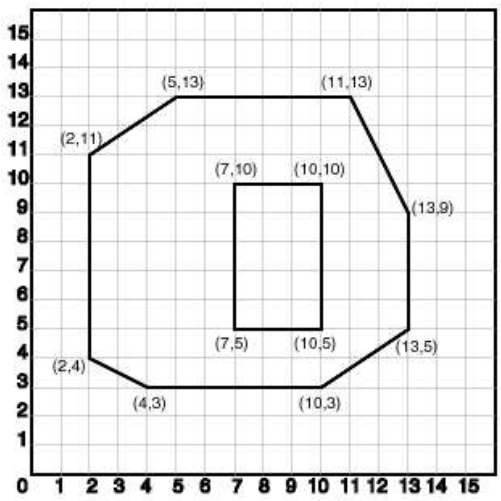
SDO_ETYPE과 SDO_INTERPRETATION은 각각 객체의 타입과 좌표를 읽어들이는 방법을 정의합니다. 저장하는 객체가 포인트인지, 사각형인지, 혹은 여러 개의 element로 구성된 compound element인지에 따라 저장된 좌표들을 읽고 해석하는 방법이 다르기 때문에, 이 두 속성 값의 조합을 통해 다양한 모양의 공간 객체를 정의할 수 있습니다. 예를 들어 SDO_ETYPE의 값이 1이고 INTERPRETATION의 값이 1이라면 하나의 포인트 데이터를 의미하지만, INTERPRETATION의 값이 3이라면 3개의 포인트로 이루어진 포인트 클러스터를 의미합니다. 모든 도형에 대한 표현식을 전부 다 이해하고 암기하실 필요는 없습니다. 표현할 수 있는 도형 및 속성 값의 조합은 다음 주소에서 확인할 수 있습니다.

https://docs.oracle.com/database/121/SPATL/sdo_geometry-object-type.htm#SPATL496

SDO_ORDINATES

SDO_ORDINATES 속성은 숫자로 된 배열로 공간 객체의 boundary를 정의하는 좌표들을 저장하기 위해 사용됩니다. 최대 1048576의 크기를 갖는 배열에 좌표들이 일괄적으로 저장되기 때문에 해석을 위해서는 SDO_ELEM_INFO 속성이 반드시 정의되어 있어야 합니다.

예제를 통해 각각의 속성 값들이 어떻게 정의되는지 알아보겠습니다. [그림 5]는 도형의 내부, 외부에 대한 두 개의 외곽선으로 이루어진 polygon ring입니다. 이것을 저장할때 SDO_GEOMETRY의 각 속성 값이 어떻게 설정되는지 알아보겠습니다.



[그림 5] Polygon with a Hole

그림 5에 대한 SDO_GEOMETRY 값은 다음과 같습니다

SDO_GTYPE = 2003
SDO_SRID = NULL
SDO_POINT = NULL
SDO_ELEM_INFO = SDO_ELEM_INFO_ARRAY(1,1003,1, 19,2003,1)
SDO_ORDINATES = 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)

먼저 GTYPE의 2003은 2차원의 Polygon이라는 의미입니다 (TT가 03이면 polygon)
좌표계인 SRID는 기본값이고, 포인트 데이터가 아니기 때문에 SDO_POINT 역시 null입니다.
SDO_ELEM_INFO의 경우 (1,1003,1)과 (19,2003,1) 이라는 두 개의 ELEM_INFO가 저장되어 있습니다.
이때 ETYPE인 1003은 도형의 바깥쪽 외곽선(exterior boundary)에 대한 Polygon ring을 의미하고, 2003은 안쪽 외곽선(interior boundary)에 대한 polygon ring을 의미합니다. ETYPE이 1003, 2003일때 NTERPRETATION 값을 1을 주게 되면 straight line segment로 이루어진 simple polygon이라는 의미입니다.
이 두 개의 polygon 외곽선은 SDO_ORDINATES에 저장되어 있습니다. SDO_ORDINATES 의 첫번째부터 18번째까지의 수가 외부 외곽선의 좌표들입니다. Polygon이기 때문에 시작점과 끝점은 항상 동일해야 합니다. 내부 외곽선의 좌표는 SDO_ELEM_INFO에 정의된대로, 19번째 수 부터 시작합니다. 마찬가지로 시작점과 끝점의 좌표가 동일합니다.

Geometic types

Geometry란 직선 또는 원의 호로 이어진 vertex 배열을 말합니다. Oracle spatial에서는 SDO_GTYPE, SDO_ELEM_INFO, SDO_ORDINATES 의 조합으로 다양한 형태의 도형을 정의할 수 있습니다.
Oracle에서 지원하는 2차원 geometry의 타입은 [그림 8]과 같습니다. 각각의 도형에 대한 표현 방법은 다음 주소에서 확인할 수 있습니다.
https://docs.oracle.com/cd/B28359_01/appdev.111/b28400/sdo_objrelschema.htm#SPATL516

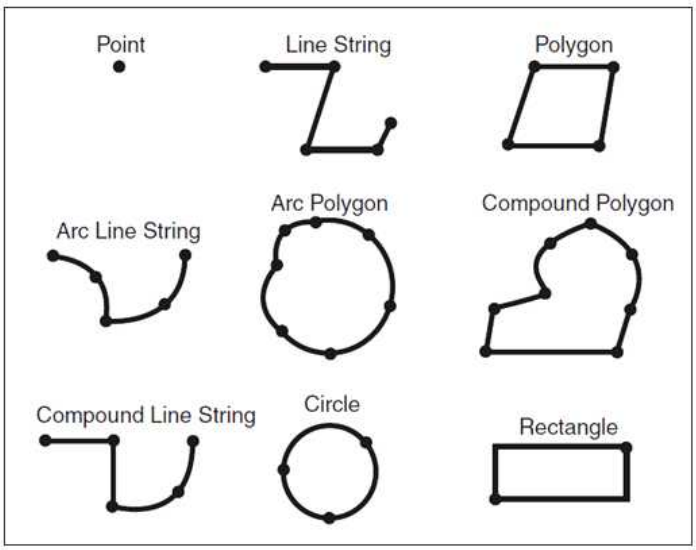


그림 6. Geometric Types

cola_markets table에 대한 삽입 예제

다시 [그림 1]의 예제로 돌아가서, cola_a부터 cola_d 까지 4개의 도형을 테이블에 삽입 해 보겠습니다.

cola_markets 예제에서는 [그림 6]의 geometry 중 polygon과 rectangle, circle이 사용되었습니다. Polygon은 polygon의 각 모서리의 좌표, rectangle은 대각선 방향의 두 꼭지점, circle은 원 위의 임의의 세 개의 점에 대한 좌표로 정의할 수 있습니다. 이 중 circle의 경우에는 임의의 점의 좌표를 계산하기보다는 중심 점의 좌표에서 원의 반지름을 각 축에 대해 더하거나 뺀 값을 사용하는 것이 편리합니다. [그림 7]은 사각형의 geometry를 갖는 cola_a를 삽입하는 SQL 코드입니다. market_id는 1, name은 cola_a입니다.

```
INSERT INTO cola_markets VALUES
(
  1,
  'cola_a',
  SDO_GEOMETRY(
    2003, -- 2차원의 폴리곤
    NULL,
    NULL,
    SDO_ELEM_INFO_ARRAY(1,1003,3), -- 1개의 사각형(1003 = exterior)
    SDO_ORDINATE_ARRAY(1,1, 5,7) -- 사각형을 정의하기 위한 두 개의 포인트
  )
);
```

[그림 7] cola_a 삽입

[그림 8]에서는 나머지 데이터들을 삽입하겠습니다. cola_b와 cola_c는 사각형이 아닌 polygon이고, cola_d는 원입니다.

```
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)
  )
);

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)
  )
```

```
);

-- Now insert an area of interest for Cola D. This is a
-- circle with a radius of 2. It is completely outside the
-- first three areas of interest.

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)  --circle은 원 위의 세 개의 점으로 정의
  )
);
```

[그림 8] cola b, c, d 삽입

공간 데이터 테이블에 대한 연산

공간 객체들간의 연산 방법은 ① operator를 이용하는 방법, ② function 과 procedure 이 있습니다. operator와 function은 동일한 연산을 하는 경우도 있습니다(ex. 9-intersection, within-distance) 두 방법의 차이점은 다음과 같습니다.

	Operator	Procedure/Function
최적화 여부	최적화 되어있음	되어있지 않음
인덱스	인덱스가 정의되어 있어야함	인덱스를 사용하지 않음
사용법	WHERE 절에서 사용	WHERE 절, subquery 에서 사용

우리는 아직 인덱스를 빌드하지 않았기 때문에, function을 사용하여 몇가지 예제에 대한 공간 연산을 수행 해 보겠습니다. 먼저 공간 객체들 간의 Topological relationship을 정의하는 연산자는 **SDO_GEOM.RELATE** 입니다. 포맷은 다음 두 가지가 있습니다.

```
SDO_GEOM.RELATE(
    geom1 IN SDO_GEOMETRY,
    dim1 IN SDO_DIM_ARRAY,
    mask IN VARCHAR2,
    geom2 IN SDO_GEOMETRY,
    dim2 IN SDO_DIM_ARRAY
    ) RETURN VARCHAR2;
```

방법 1

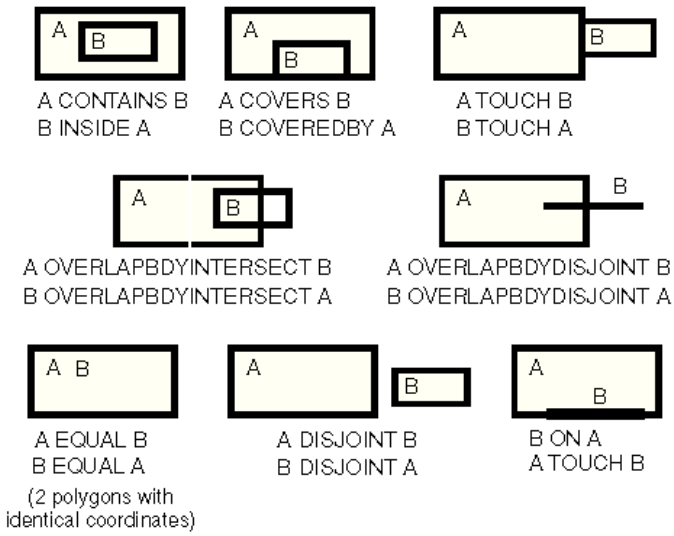
or

```
SDO_GEOM.RELATE(
    geom1 IN SDO_GEOMETRY,
    mask IN VARCHAR2,
    geom2 IN SDO_GEOMETRY,
    tol IN NUMBER
    ) RETURN VARCHAR2;
```

방법 2

공통적으로 두 개의 geometry를 인자로 받습니다. 이때 두 geometry의 차원 수는 동일해야 합니다. 차이점은 방법 2의 경우 tolerance factor를 사용한다는 것입니다. tolerance factor는 두 geometry가

정의된 좌표계에서 tolerance factor 만큼의 오차율을 용인하겠다는 의미입니다.
mask 부분에 9-intersection에 해당하는 키워드를 넣으면 해당 두 geometry가 그 관계를 가질 경우 true, 아닐 경우 false 값을 반환합니다. 공간 객체들 간의 topological relationship은 9-intersection model을 사용하여 정의할 수 있습니다. 9-intersection model에는 [그림 9]과 같은 종류가 있습니다.



[그림 9] Topological relationships

Mask에 9-intersection 외에, 넣을 수 있는 키워드가 몇가지 있습니다.

DETERMINE : 두 geometry가 가지는 모든 관계를 출력합니다.
ANYINTERACT : 9가지 intersection중 어떤 intersection을 가지면 true를 반환합니다.

SDO_GEOM.RELATE의 사용 방법과 결과에 대한 예시는 다음과 같습니다.

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

NAME       RELATIONSHIP
----------------------------
cola_a     TOUCH
cola_b     EQUAL
cola_c     OVERLAPBDYINTERSECT
cola_d     DISJOINT
```

SDO_GEOM.RELATE 이외에 더 많은 function에 대한 정보는 다음 주소에서 알아볼 수 있습니다.
https://docs.oracle.com/cd/B28359_01/appdev.111/b28400/sdo_objgeom.htm#SPATL120

[그림 10]은 cola_market에 대한 몇가지 연산에 대한 예제입니다. 예제에서 연산들은 tolerance factor를 사용하는 포맷을 사용합니다.

```
-- 2개의 도형과 교차하는 도형 얻기
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';

-- 2개의 도형(지오메트리, Geometry)에 대한 공간관계 연산의 결과 얻기
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';

-- 해당 공간 테이블의 전체 면적 얻기
SELECT name, SDO_GEOM.SDO_AREA(shape, 0.005) FROM cola_markets;

-- 특정 도형에 대한 면적 얻기
SELECT c.name, SDO_GEOM.SDO_AREA(c.shape, 0.005) FROM cola_markets c
WHERE c.name = 'cola_a';

-- 2개의 도형 간의 거리 얻기
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';

-- 특정 지오메트리가 올바르게 정의되어 있는가?
SELECT c.name, SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT(c.shape, 0.005)
FROM cola_markets c WHERE c.name = 'cola_c';
```

[그림 10] cola_market에 대한 연산 예제