

# Bases de Datos Geográficas



Taller de Sistemas de Información Geográfica



# Open Geospatial Consortium (OGC)

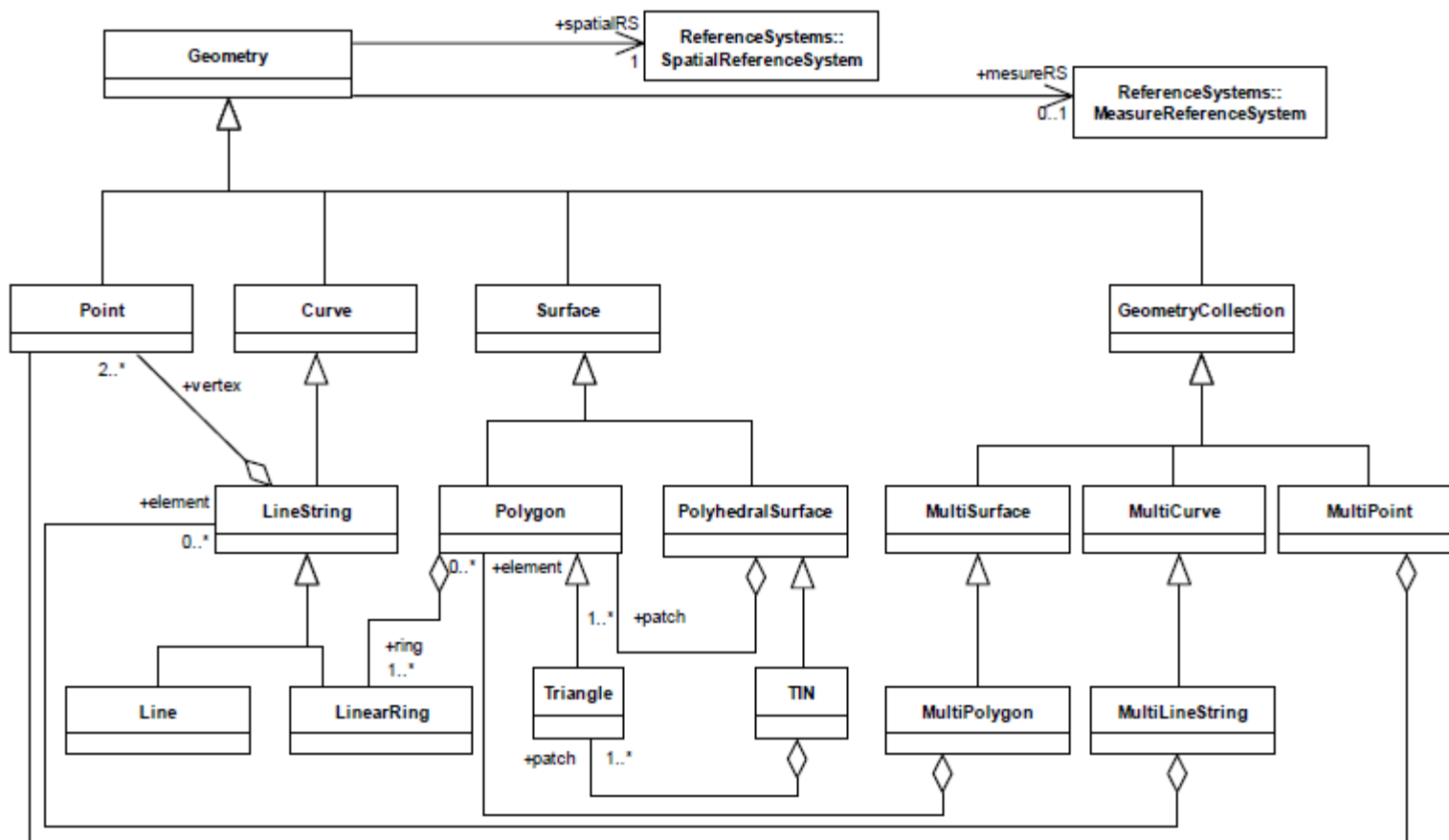
- ❑ El Open Geospatial Consortium (OGC) es un consorcio internacional que agrupa empresa, universidades y organismos estatales (alrededor de 400) con las siguientes metas fundamentales:
  - Proveer de estándares abiertos, gratuitos y públicos.
  - Liderar la creación de estándares que permitan que el contenido y los servicios geo-espaciales se integren a procesos cívicos y de negocios, la Web Espacial y los Sistemas Empresariales
  - Facilitar la adopción de arquitecturas de referencia abiertas en materia de información espacial.



# OGC Simple Features Standard (SFS)

- ❑ OGC define un modelo de datos para objetos espaciales conocido como Simple Features Standard (SFS)(Estandar ISO 19115).
- ❑ El estándar se divide en dos partes:
  1. Arquitectura común (Modelo de Objetos)
  2. Implementación en SQL (Permite construir bases de datos geográficas)





# Clases de Objetos Geográficos

## □ Geometry (abstracta)

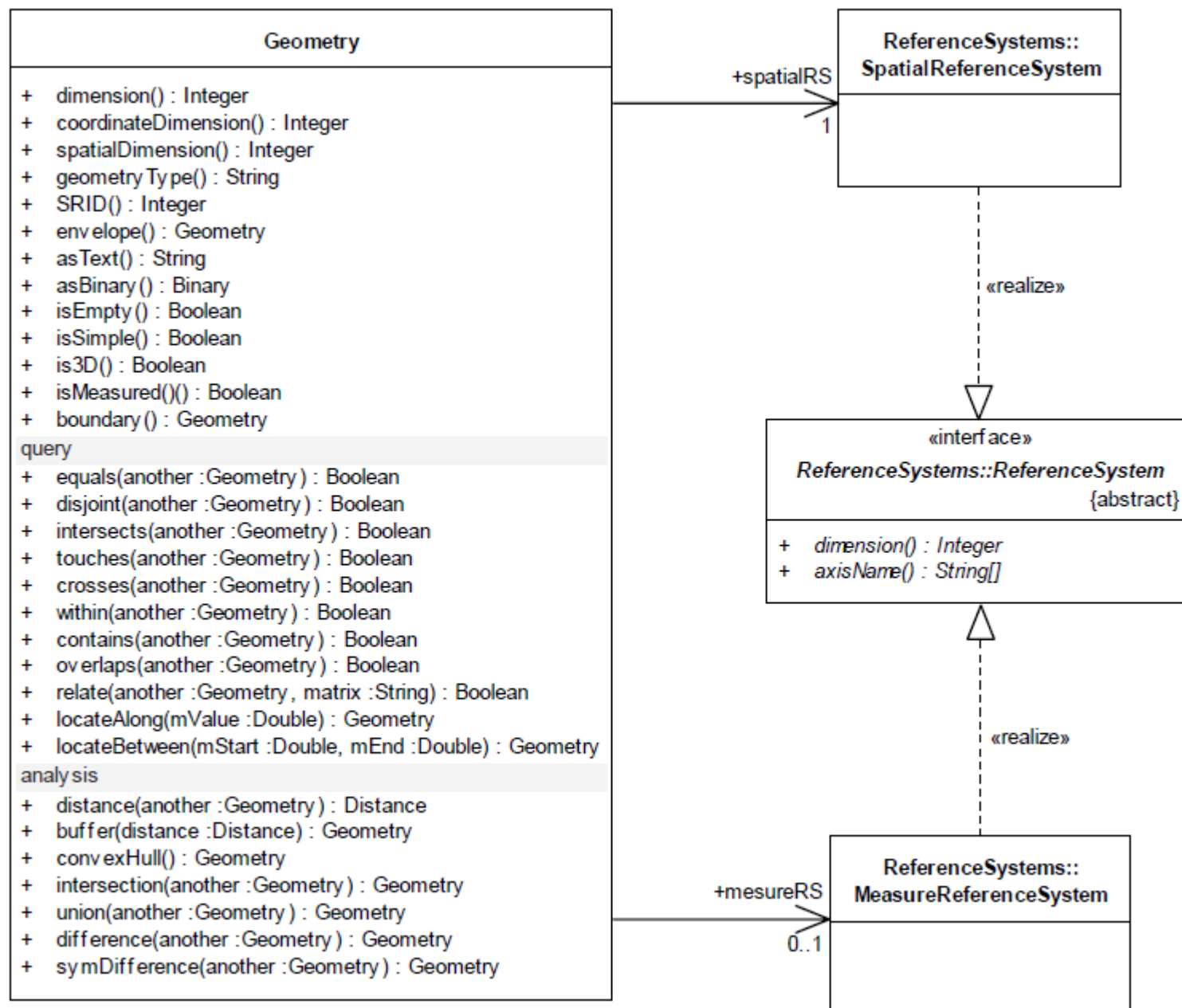
- Es la clase base de la jerarquía.
- Permite representar geometrías planas (hasta 2 dimensiones) en espacios de coordenadas R2, R3 y R4.
- Métodos básicos:
  - *dimension(): La dimensión inherente a la geometría. Devuelve*
    - 1 si geom. vacía
    - 0 si longitud=área=0
    - 1 si longitud  $\neq$  0, área=0
    - 2 si longitud  $\neq$  0, área  $\neq$  0
  - *geometryType(): Devuelve el subtipo concreto al que pertenece el objeto (ej. Polygon, MultiLineString, etc.)*



# Clases de Objetos Geográficos

- *SRID(): Devuelve el ID del Sistema Espacial de Referencia utilizado para georreferenciar este objeto.*
- *asText(): Devuelve la representación WKT del objeto.*
- *asBinary(): Devuelve la representación WKB del objeto.*
- *isSimple(): Devuelve 1 si el objeto representa una geometría sin propiedades anómalas, como autointersecciones o autotangencia.*
- *isEmpty(): Devuelve 1 si el objeto representa la geom. vacía (ningún punto). El conjunto vacío es simple.*
- *is3D(): Devuelve 1 si el objeto tiene una coordenada z.*
- *isMeasured(): Devuelve 1 si el objeto tiene una coordenada m.*
- *boundary(): Devuelve otra geometría con la frontera de este objeto.*
- *envelope(): Devuelve el mínimo rectángulo delimitador (minimum bounding box, MBR) que contiene a la geometría.*





# Clases de Objetos Geográficos

## □ Point

- Objeto geométrico 0D (punto).
- Coordenadas x,y (z,m opcionales).
- Su frontera es el conjunto vacío.
- Posee métodos para obtener sus coordenadas





# Clases de Objetos Geográficos

## □ Curve (abstracta)

- Objeto geométrico 1D que representa la curva definida por una secuencia de puntos y la interpolación entre los mismos.
- Métodos:
  - *length(): el largo en su SRS asociado*
  - *startPoint(), endPoint(): punto inicial y final*
  - *IsClosed(): devuelve 1 si startPoint=endPoint*
  - *IsRing(): devuelve 1 si IsClosed()=1 y no pasa mas de una vez por el mismo punto (es simple).*



# Clases de Objetos Geográficos

## □ LineString

- Es una curva con interpolación lineal entre los puntos.
- Métodos:
  - *numPoints(): devuelve cantidad de puntos*
  - *pointN(): devuelve el punto N*

## □ Line

- Es un LineString de 2 puntos (un segmento de recta)

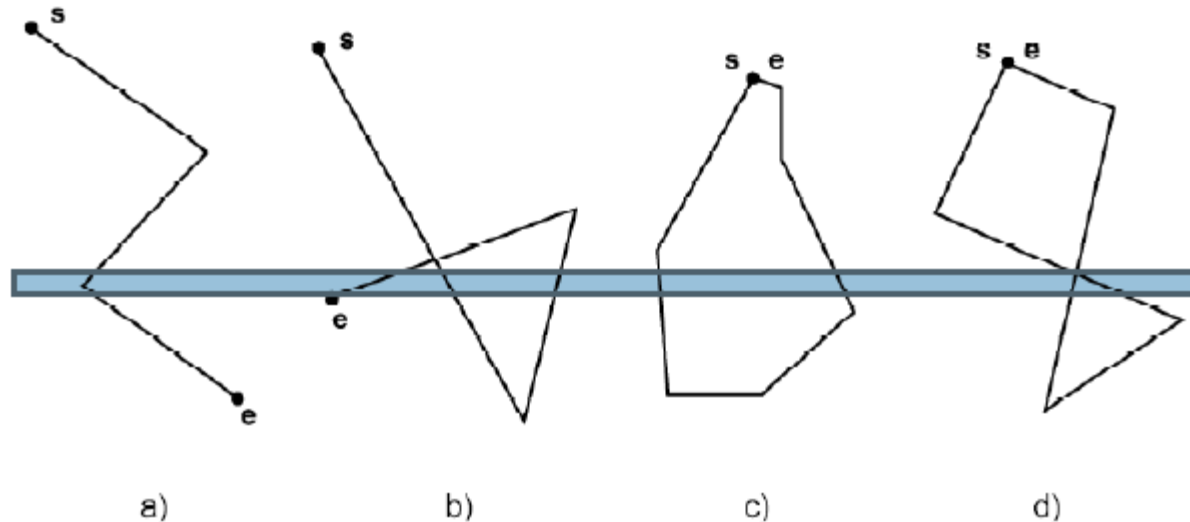
## □ LinearRing

- En un LineString simple y cerrado (un anillo)



# Clases de Objetos Geográficos

## □ Diferentes tipos de LineString



# Clases de Objetos Geográficos

## □ Surface (abstracta)

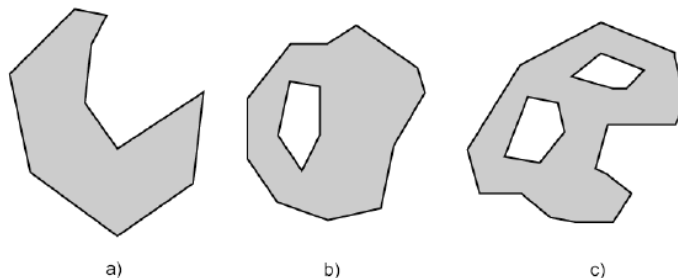
- Objeto geométrico 2D que representa una superficie.
- Métodos:
  - *area(): devuelve el área en el SRS correspondiente.*
  - *centroid(): devuelve el centroide (baricentro) de la superficie (puede ser un punto exterior)*
  - *pointOnSurface(): devuelve un punto cualquiera perteneciente a la superficie.*



# Clases de Objetos Geográficos

## □ Polygon

- Es una superficie plana definida por un LinearRing exterior y 0 o más LinearRings interiores, para permitir polígonos con huecos.
- Los polígonos son objetos simples: no existe intersección entre sus contornos.
- Métodos:
  - *exteriorRing(): devuelve el anillo exterior*
  - *numInteriorRings(): devuelve la cantidad de anillos interiores.*
  - *interiorRingN(): devuelve el anillo interior N*



# Clases de Objetos Geográficos

## ❑ GeometryCollection

- Colección de geometrías de cualquier tipo
- No existe relación implícita entre sus elementos.
- Todos sus elementos deben estar en el mismo SRS.
- Permite trabajar con un conjunto de geometrías como una unidad que tiene atributos no espaciales comunes.
  - *Ej. La red fluvial uruguaya, los territorios del Reino Unido y sus islas, etc.*

## ❑ MultiPoint

- Colección geométrica de puntos.
- Su frontera es el conjunto vacío.
- Un multipunto es simple si no tiene puntos repetidos.

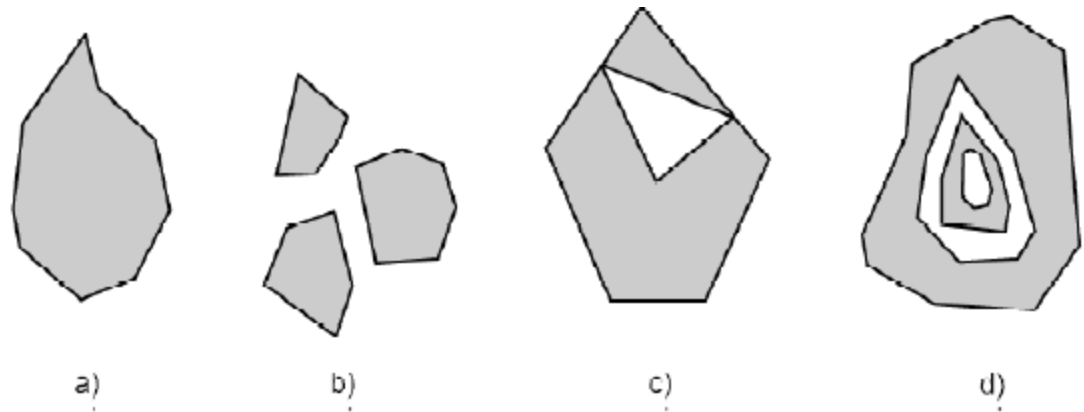
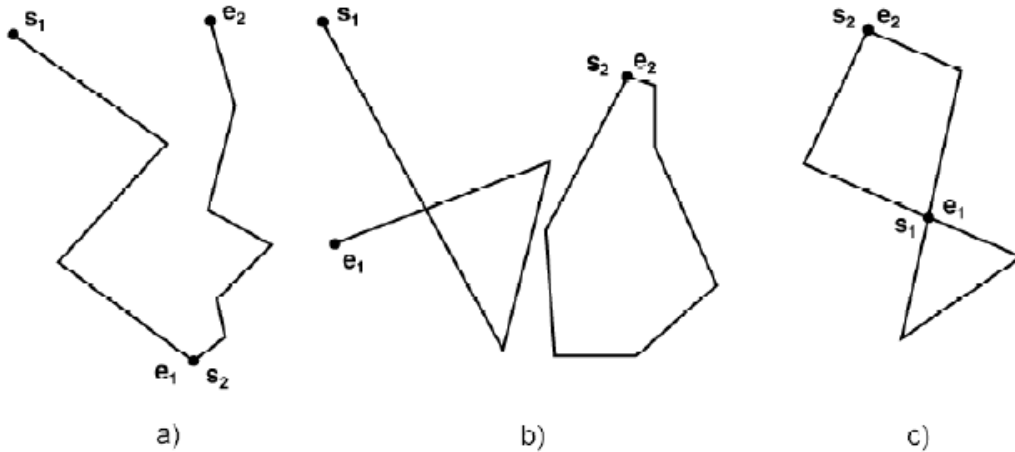


# Clases de Objetos Geográficos

- ❑ MultiLineString
  - Es una MultiCurve cuyos elementos son LineStrings.
- ❑ MultiPolygon
  - Es una MultiSurface cuyos elementos son Polygons.



# Clases de Objetos Geográficos





# Well-know Binary (WKB)

- Permite representar geometrías mediante un flujo de bytes.
- Ej. un polígono determinado por dos anillos puede representarse de la siguiente manera.

B=1	T=3	NR= 2	NP= 3	X1	Y1	X2	Y2	X3	Y3	NP= 3	X1	Y1	X2	Y2	X3	Y3
-----	-----	----------	----------	----	----	----	----	----	----	----------	----	----	----	----	----	----

B = orden de bytes

T = tipo de figura (polígono)

NR = numero de anillos (2)

NP = numero de puntos (3 para cada anillo)



# Well-know Text (WKT)

- ❑ Permite representar geometrías mediante un texto legible por personas.
- ❑ Ej. Punto, Línea, Multilínea, Polígono, Multipolígono en WKT.

```
POINT(2572292.2 5631150.7)

LINESTRING (2566006.4 5633207.9, 2566028.6 5633215.1, 2566062.3 5633227.1)

MULTILINESTRING((2566006.4 5633207.9, 2566028.6 5633215.1), (2566062.3 5633227.1, 2566083 5633234.8))

POLYGON (2568262.1 5635344.1, 2568298.5 5635387.6, 2568261.04 5635276.15, 2568262.1 5635344.1);

MULTIPOLYGON(((2568262.1 5635344.1, 2568298.5 5635387.6, 2568261.04 5635276.15, 2568262.1 5635344.1), (2568194.2 5635136.4, 2568199.6 5635264.2, 2568200.8 5635134.7, 2568194.2 5635136.4 )))
```



# Relaciones Topológicas

- ❑ La topología se define matemáticamente como la rama que estudia las propiedades de los cuerpos o figuras geométricas que se mantienen invariantes bajo una transformación continua.
- ❑ Informalmente, esas propiedades son las que no cambian cuando se realiza una deformación “elástica” sobre las figuras.
  - Ej. Dos países limítrofes en un mapa plano en papel deben seguir siéndolo en un globo terráqueo.
- ❑ En GIS, se focaliza en las reglas y propiedades que deben cumplir los elementos de una misma capa o de varias para considerarse bien definidos.
  - Ejs. Las líneas que definen un polígono deben ser cerradas; las líneas de una capa de calles deben coincidir con los límites de los polígonos de una capa de manzanas.



# Clasificación de Operaciones Espaciales

- ❑ Podemos clasificar las operaciones espaciales en 4 grupos:
  - Teoría de conjuntos: Unión, Intersección, Diferencia, etc.
    - *Ej. La intersección de dos polígonos produce un polígono, o una línea, o un punto, o vacío.*
  - Topológicas: Toca, Superpone, etc.
    - *Ej. La frontera de Uruguay toca la frontera de Argentina (Uruguay y Argentina son polígonos)*
  - Métricas: Área, Distancia, etc.
    - *Ej. La distancia entre Montevideo y Atlántida es de 45 km.*
  - Direccionales: Norte, Sur, Sureste, etc.
    - *Ej. La ciudad de Rocha se encuentra al Noreste de Punta de Este.*



# Operaciones Espaciales

- ❑ Equals: las dos figuras son iguales.
- ❑ Disjoint: las dos figuras son disjuntas.
- ❑ Touches/Meets: las dos figuras tienen al menos un punto común en sus fronteras.
- ❑ Crosses: las figuras se cruzan (significado varía según figura)
- ❑ Within/Inside: la primera figura está adentro de la segunda
- ❑ Contains: la primera figura contiene a la segunda
- ❑ Overlaps: al menos un punto común en interiores, exteriores y fronteras.



# Operaciones Espaciales

## ❑ Modelo de las 9 Intersecciones (9IM)

- Se define una matriz binaria 3x3 para un par de figuras geométricas A y B de la siguiente manera, en base a los conjuntos frontera, interior y exterior

$$\mathfrak{S}_9(A, B) = \begin{pmatrix} A \cap B & A \cap \partial B & A \cap B^- \\ \partial A \cap B & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B & A^- \cap \partial B & A^- \cap B^- \end{pmatrix}$$

- Esto nos permite caracterizar las operaciones topológicas según la intersección sea vacía o no.
  - Ej. *disjoint* = ((0 0 1) (0 0 1) (1 1 1))

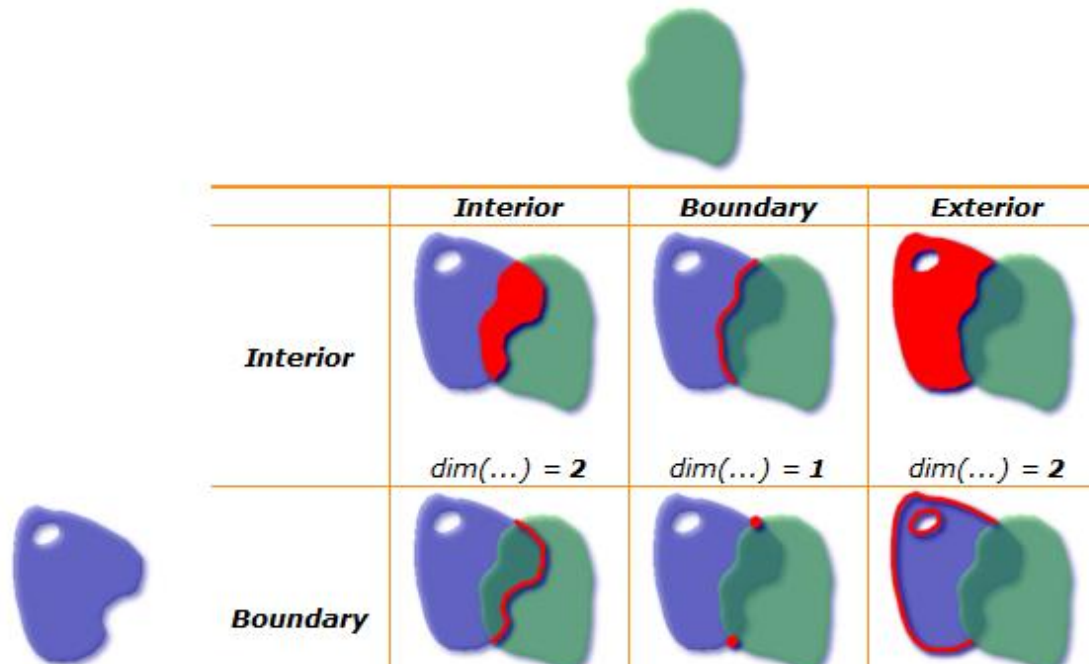











# Operaciones Espaciales

- ❑ Modelo dimensionalmente extendido de las nueve intersecciones (DE-9IM)
  - A veces no alcanza con saber que dos geometrías se intersecan. Por ejemplo, queremos saber si dos líneas se intersecan en un punto o en un segmento.
  - Para esto se extiende la matriz anterior con la dimensión de la intersección en lugar de un valor binario.



# Operaciones Espaciales



	<i>Interior</i>	<i>Boundary</i>	<i>Exterior</i>
<i>Interior</i>	 $\dim(\dots) = 2$	 $\dim(\dots) = 1$	 $\dim(\dots) = 2$
<i>Boundary</i>	 $\dim(\dots) = 1$	 $\dim(\dots) = 0$	 $\dim(\dots) = 1$
<i>Exterior</i>	 $\dim(\dots) = 2$	 $\dim(\dots) = 1$	 $\dim(\dots) = 2$



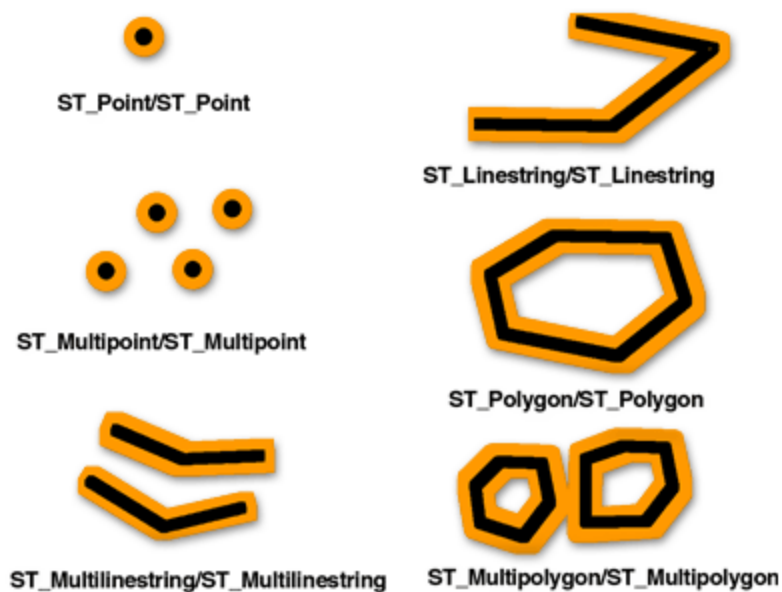


# Operaciones Espaciales

- Los valores de cada celda pueden ser 0, 1, 2, T, F, \*. T es cualquier valor en {0,1,2}, F es el conjunto vacío, \* es cualquier valor anterior.
- Ej. Dos líneas que se intersecan en un punto  
 $((0 * 1)(* * *) (1 * *))$
- Ej. Dos líneas que se intersecan en un segmento  
 $((1 * 1)(* * *) (1 * *))$

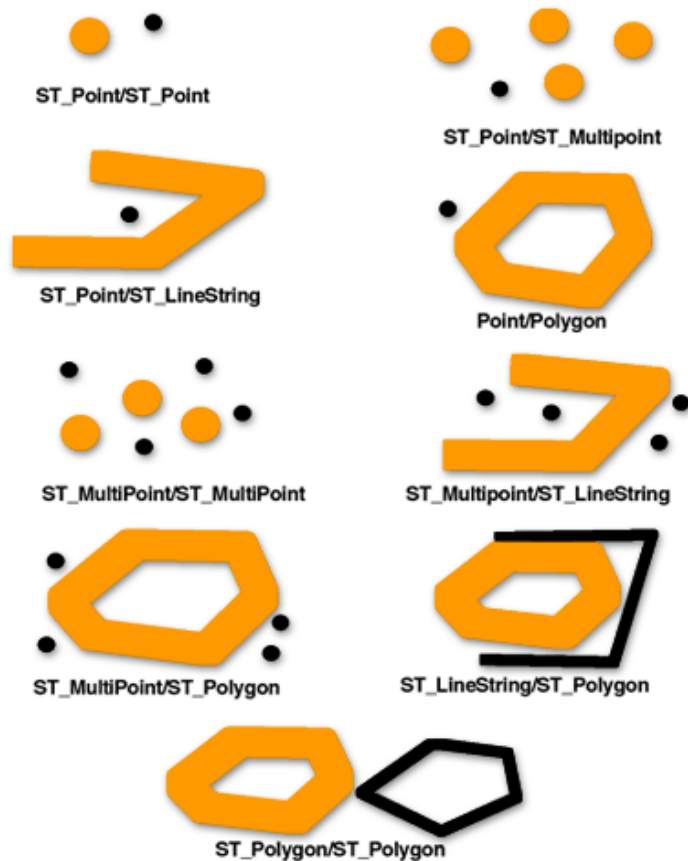


# Equals



Equ	I(b)	B(b)	E(b)
I(a)	T	*	F
B(a)	*	*	F
E(a)	F	F	*

# Disjoint



Dis	I(b)	B(b)	E(b)
I(a)	F	F	*
B(a)	F	F	*
E(a)	*	*	*

# Touches



ST\_Point/ST\_Linestring



ST\_Linestring/ST\_Linestring



ST\_MultiPoint/ST\_Linestring



ST\_Point/ST\_Polygon



ST\_MultiPoint/ST\_Polygon



ST\_Linestring/ST\_Polygon

Tou1	I(b)	B(b)	E(b)
I(a)	F	T	*
B(a)	*	*	*
E(a)	*	*	*

Tou2	I(b)	B(b)	E(b)
I(a)	F	*	*
B(a)	T	*	*
E(a)	*	*	*

Tou3	I(b)	B(b)	E(b)
I(a)	F	*	*
B(a)	*	T	*
E(a)	*	*	*



# Crosses



ST\_MultiPoint/ST\_LineString



ST\_MultiPoint/ST\_Polygon



ST\_LineString/ST\_LineString

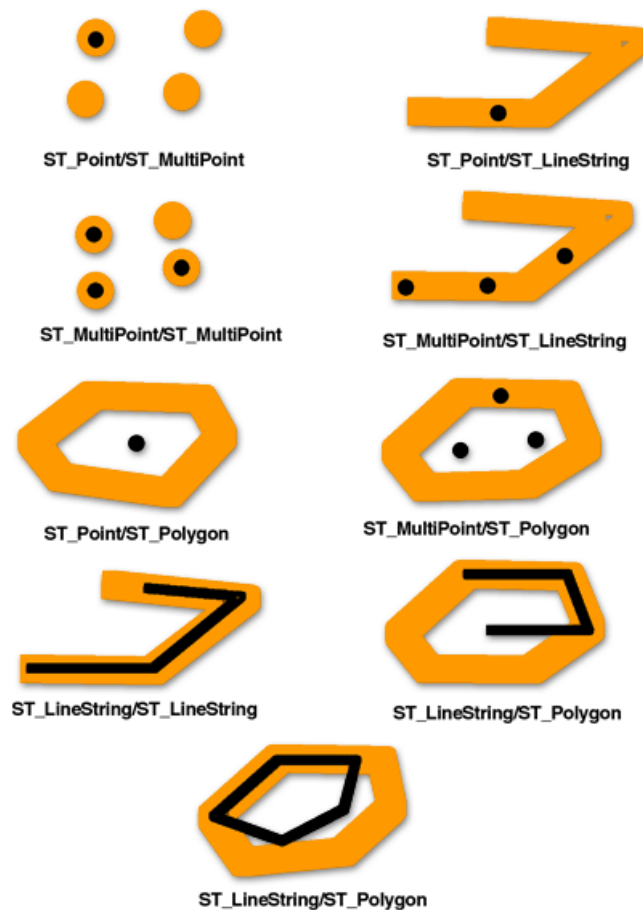


ST\_LineString/ST\_Polygon

Cro	I(b)	B(b)	E(b)
I(a)	T	*	T
B(a)	*	*	*
E(a)	*	*	*

Cro0	I(b)	B(b)	E(b)
I(a)	0	*	*
B(a)	*	*	*
E(a)	*	*	*

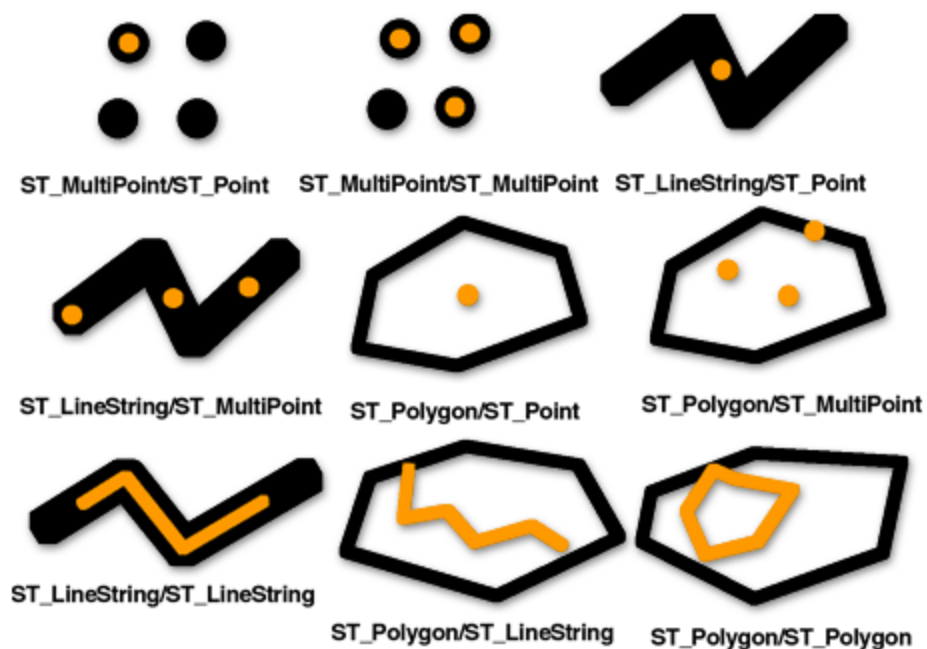
# Within



Wit	I(b)	B(b)	E(b)
I(a)	T	*	F
B(a)	*	*	F
E(a)	*	*	*



# Contains



Con	I(b)	B(b)	E(b)
I(a)	T	*	*
B(a)	*	*	*
E(a)	F	F	*



# Overlaps



ST\_LineString/ST\_LineString



ST\_Polygon/ST\_Polygon



ST\_MultiPoint/ST\_MultiPoint

Ove	I(b)	B(b)	E(b)
I(a)	T	*	T
B(a)	*	*	*
E(a)	T	*	*



# SFS - SQL

- ❑ Se definen 3 tipos de tablas:
  - FEATURE\_TABLE: Es toda tabla que almacena un conjunto de *features* (objetos geográficos). Corresponde al concepto de *layer* (capa geográfica). Cada fila es un objeto geográfico y cada columna es una propiedad de ese objeto. Una de esas columnas debe corresponder a la geometría de ese objeto (o una FK a la misma).
    - Ej. Una capa de polígonos que representan ciudades la representamos como una *feature table* Ciudad(nombre, país, población, geometría, *gid*)



# SFS - SQL

- GEOMETRY\_TABLE : Es toda tabla que almacena geometrías, en el caso que la *feature table* correspondiente no las almacene directamente. Cada fila posee un identificador geográfico (GID).
- GEOMETRY\_COLUMNS: Tabla de metadatos que posee una fila por cada columna geometría de la base, con los siguientes atributos:
  - ID de la *feature table* a la que corresponde la columna geometría
  - El nombre de la columna geometría
  - El SRID de la columna geometría
  - El tipo de geometría (Point, LineString, etc)
  - La dimensión del SRS utilizado (2D, 3D, etc.)
  - El ID de la *geometry\_table* que almacena la geometría (podría ser la misma *feature table*)



# SFS - SQL

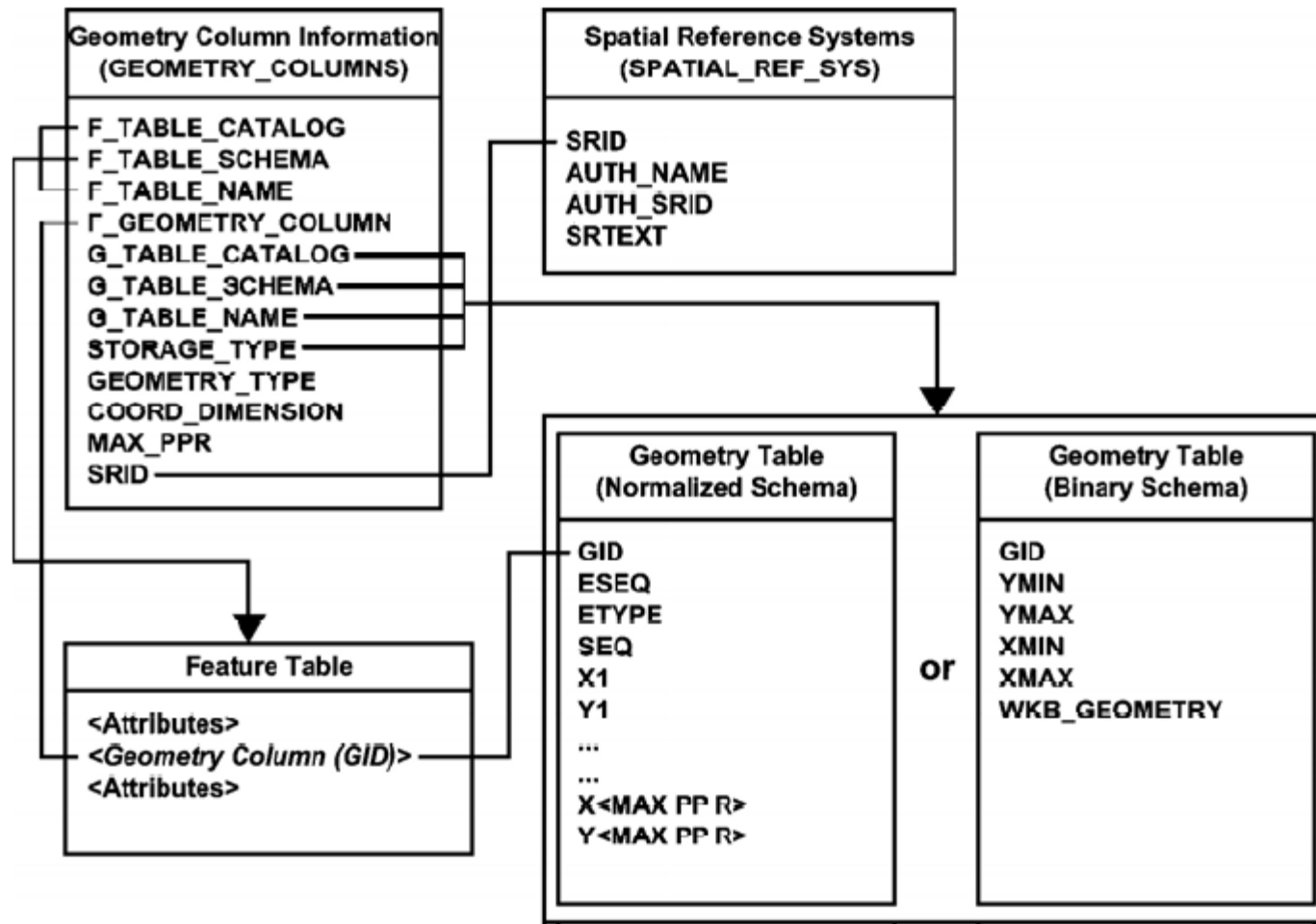
- SPATIAL\_REF\_SYS: Es el diccionario de códigos de sistemas de referencia espaciales (SRS). Solamente es necesaria para hacer operaciones de re-proyección, pero resulta útil para consulta durante el desarrollo. Dentro de esta tabla, interesan particularmente los campos SRID y STEXT.
  - Ej. Queremos obtener el SRID de la proyección que corresponde al datum Yacaré (  

```
SELECT srid FROM spatial_ref_sys WHERE srtext LIKE  
'%Yacare%';
```

  
srid = 4309



# SFS - SQL



# SFS - SQL

**Table 4: Geometry type codes**

Code	Geometry type	Coordinates
0	GEOMETRY	\\ IN X Y
1	POINT	\\ IN X Y
2	LINESTRING	\\ IN X Y
3	POLYGON	\\ IN X Y
4	MULTIPOINT	\\ IN X Y
5	MULTILINESTRING	\\ IN X Y
6	MULTIPOLYGON	\\ IN X Y
7	GEOMCOLLECTION	\\ IN X Y
13	CURVE	\\ IN X Y
14	SURFACE	\\ IN X Y
15	POLYHEDRALSURFACE	\\ IN X Y
1000	GEOMETRYZ	\\ IN X Y Z
1001	POINTZ	\\ IN X Y Z
1002	LINESTRINGZ	\\ IN X Y Z

Code	Geometry type	Coordinates
1003	POLYGONZ	\\ IN X Y Z
1004	MULTIPOINTZ	\\ IN X Y Z
1005	MULTILINESTRINGZ	\\ IN X Y Z
1006	MULTIPOLYGONZ	\\ IN X Y Z
1007	GEOMCOLLECTIONZ	\\ IN X Y Z
1013	CURVEZ	\\ IN X Y M
1014	SURFACEZ	\\ IN X Y M
1015	POLYHEDRALSURFACEZ	\\ IN X Y Z
2000	GEOMETRY	\\ IN X Y M
2001	POINTM	\\ IN X Y M
2002	LINESTRINGM	\\ IN X Y M
2003	POLYGONM	\\ IN X Y M
2004	MULTIPOINTM	\\ IN X Y M
2005	MULTILINESTRINGM	\\ IN X Y M
2006	MULTIPOLYGONM	\\ IN X Y M
2007	GEOMCOLLECTIONM	\\ IN X Y M
2013	CURVEM	\\ IN X Y M
2014	SURFACEM	\\ IN X Y M
2015	POLYHEDRALSURFACEM	\\ IN X Y M
3000	GEOMETRYZM	\\ IN X Y Z M
3001	POINTZM	\\ IN X Y Z M
3002	LINESTRINGZM	\\ IN X Y Z M



# Ejemplos de Operadores Espaciales en SQL

- ❑ Utilizaremos las siguientes *feature tables*:
  - Ciudades(gid, código, nombre, población, geom), capa de polígonos.
  - Calles (gid, código, nombre, geom), capa de líneas.
  - Hoteles (gid, nombre, dirección, estrellas, capacidad, geom) capa de puntos.
  - Rios(gid, nombre, geom), capa de líneas
- ❑ Operadores en el SELECT
  - Obtener nombre, código y área de la ciudad de Montevideo:  

```
SELECT c.nombre, c.codigo, ST_AREA(c.geom) AS area  
FROM Ciudades c WHERE c.nombre='Montevideo';
```



# Ejemplos de Operadores Espaciales en SQL

- Obtener nombre y longitud de la calle con codigo=223  
SELECT r.nombre, **ST\_LENGTH**(r.geom) AS longitud  
FROM Calles r WHERE r.codigo=223;
- Listar las nombre y densidad de poblacion de ciudades en orden decreciente de densidad:  
SELECT c.nombre, c.poblacion/**ST\_AREA**(c.geom) AS densidad  
FROM Ciudades c;  
ORDER BY densidad DESC;



# Ejemplos de Operadores Espaciales en SQL

## ❑ Operadores en el WHERE (Join Espacial):

- Obtener la cantidad de hoteles 4 estrellas en la ciudad de Colonia:

```
SELECT COUNT(h.nombre) FROM Hoteles h, Ciudades c  
WHERE ST_CONTAINS(c.geom, h.geom)  
AND c.nombre='Colonia' AND h.estrellas=4
```

- Listar todas las ciudades que se encuentren a menos de 100 km del río Uruguay

```
SELECT c.nombre FROM Ciudades c, Rios r  
WHERE ST_OVERLAPS(c.geom, ST_BUFFER(r.geom, 100))  
AND r.nombre='Uruguay'
```





# Ejemplos de Operadores Espaciales en SQL

- Selecciona las calles que se cruzan con un río en un segmento:

```
SELECT c.nombre
```

```
FROM Calles c, Rios r
```

```
WHERE ST_Relate(c.geom, r.geom, '1*1***1**');
```



# Índices Espaciales

## ❑ Concepto de índice:

- Estructura de datos física de acceso que se define en base a uno o más campos de un archivo. En un DBMS, se definen índices sobre tablas (en lugar de archivos).
- Hacen más eficiente el acceso a registros en operaciones donde intervienen campos indizados.
- Clasificación según campos: índices primarios, índices de agrupamiento, índices secundarios.
- En los DBMS se utilizan comúnmente árboles B y B+.
- Los árboles B son apropiados para tipos de datos que pueden ser ordenados sobre un eje.
  - Ej.  $1 < 5$  (enteros), 'Brasilia' < 'Montevideo' (strings)
  - Ej. Cómo defino  $\text{Point}(1,0) < \text{Point}(0,1)$ ?



# Índices Espaciales

## □ Árboles R

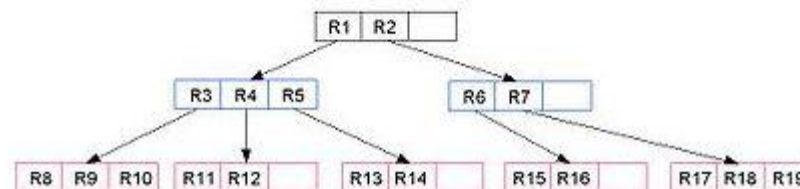
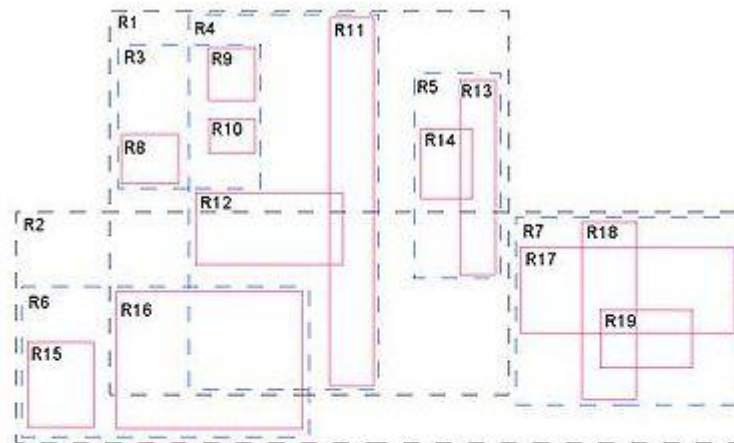
- Balanceados
- Cada nodo es un rectángulo
- Nodo hijo dentro de nodo padre (WITHIN)
- Superposición es posible entre rectángulos (OVERLAP)
- Idea: construyo un rectángulo por cada objeto geográfico. El rectángulo es el Minimum Orthogonal Bounding Rectangle (MBR), también llamado Bounding Box o Envelope (Rectángulo Delimitador). Luego construyo rectángulos que contienen completamente a los anteriores y así sucesivamente.

## □ Árboles R+

- No se permite superposición entre rectángulos del árbol.

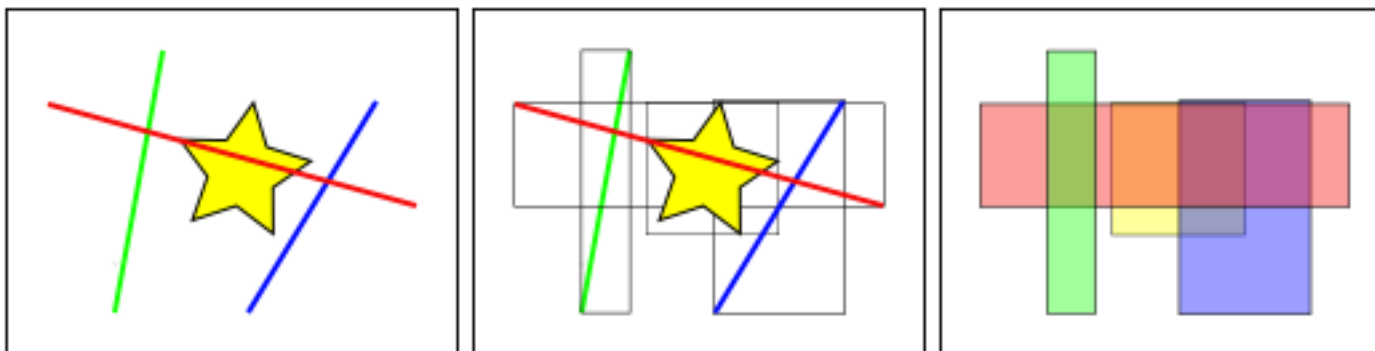


# Índices Espaciales



# Índices Espaciales

- ❑ Paradigma Filtrar-Refinar para procesar consultas
  - Filtrar: Encontrar un superconjunto del conjunto solución, más chico que el superconjunto total. Se utilizan operadores y tipos de datos aproximados. (OVERLAP y MBR)
  - Refinar: Encontrar conjunto solución. Se utilizan los operadores y tipos de datos exactos de la consulta.
  - Ej. Seleccionar la línea que cruza la estrella.



# Índices Espaciales

- *Filtrar: Encuentro líneas  $L$  tales que  $OVERLAP(MBR(E), MBR(L))$*
- *Refinar: Entre los  $L$  encontrados, aplico  $CROSSES(E, L)$  para llegar a la solución.*
- ❑ Ventaja de utilizar índices espaciales: En el primer paso, trabajo exclusivamente con geometrías del árbol (rectángulos) y una sola operación. Solamente en el segundo paso leo la geometría exacta del objeto geográfico y aplico la operación espacial exacta.
- ❑ Algunas implementaciones de SDBMS utilizan Árboles de Búsqueda Generalizados (Generalized Search Trees, GiST) para implementar Árboles R. Ej. PostGIS.



# OGC Geography Markup Language (GML)

- ❑ GML es una gramática escrita como XML Schema que permite almacenar y transportar información geo-espacial.

`xmlns:gml="http://www.opengis.net/gml"`

- ❑ Codificación estándar para modelos OGC (por ejemplo, para SFS).
- ❑ Posee un conjunto de esquemas (.xsd) para diferentes aplicaciones (llamados *application schemas*):  
Objetos geográficos, Sistemas de Referencia Coordinados (CRS), Topologías, Información Temporal y Objetos Dinámicos, Unidades de Medida, Direcciones, Observaciones, Coberturas, etc.



# OGC Geography Markup Language (GML)

## □ Ejs. LineString y Polygon

```
<gml:LineString>
```

```
  <gml:posList> 45.256 -110.45 46.46 -109.48 43.84 -109.86 </gml:posList>
```

```
</gml:LineString>
```

```
<gml:Polygon>
```

```
  <gml:exterior>
```

```
    <gml:LinearRing>
```

```
      <gml:posList> 45.256 -110.45 46.46 -109.48 43.84 -109.86 45.256 -  
110.45 </gml:posList>
```

```
    </gml:LinearRing>
```

```
  </gml:exterior>
```

```
</gml:Polygon>
```



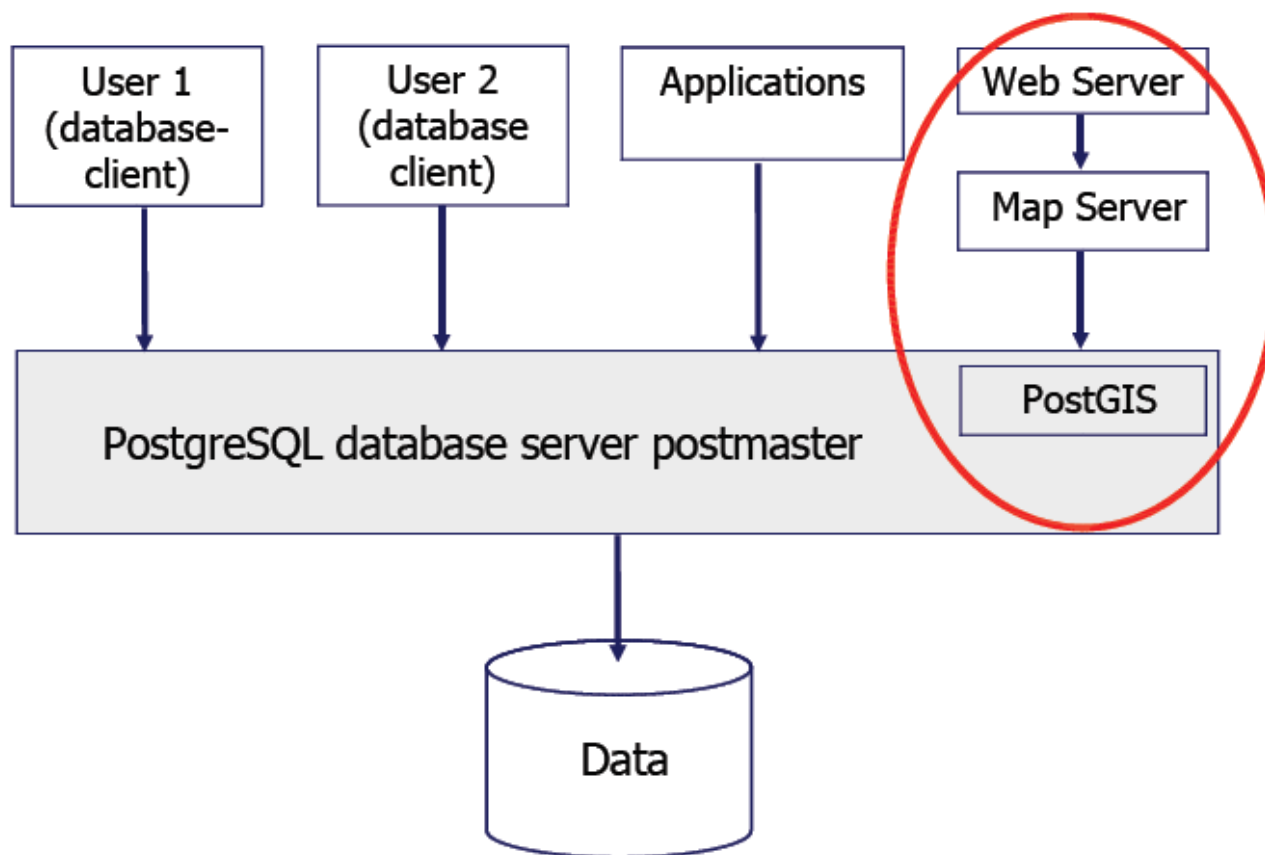


# Implementaciones de SFS-SQL

- ❑ Libres:
  - PostgreSQL/PostGIS (desde: 2001)
  - MySQL Spatial Extensions (desde: 2003)
- ❑ Algunas comerciales:
  - Oracle Spatial (desde: 1998)
  - ESRI ArcSDE (desde: 1996)
  - SQL Server (desde: 2008)



# Caso de estudio: PostGIS



# Caso de estudio: PostGIS

- ❑ Extiende el DBMS PostgreSQL.
- ❑ Incluye todos los tipos OGC SFS-SQL
- ❑ Agrega representación EWKT, EWKB que incluyen SRID en la geometría.
- ❑ Agrega algunas geometrías OGC SQL-MM (curvas con interpolación no lineal).
- ❑ Agrega tipo Geography: solo coordenadas geográficas. Ventaja: exactitud global (no hay proyecciones). Desventaja: Operaciones complejas (menos performance).
- ❑ Utiliza *Feature Tables* con GID y geometría (No hay *Geometry Tables*).
- ❑ Utiliza índices GiST.



# Caso de estudio: PostGIS

- ❑ Creación de una base de datos geográfica
  - Se utiliza el template *postgis* o se corren los script SQL *lwpostgis.sql* (crea tipos y funciones geo-espaciales) y *spatial\_ref\_sys.sql* (crea la tabla del mismo nombre).
- ❑ Creación de una capa geográfica a partir de un shapefile
  - Se utiliza el programa *shp2pgsql*  
Ej. *shp2pgsql -s 32721 -I departam\_shp departamento geodb > "departamento.sql"*  
  
En este caso se crea la tabla *departamento* de la base *geodb* a partir del shapefile *departam\_shp*. Se especifica el SRID 32721 y la creación de índice espacial (sobre el campo *geometry*).



# Caso de estudio: PostGIS

- ❑ Creación de una capa geográfica desde cero
  - Se crea una tabla sin la columna geográfica
  - Se utiliza la función AddGeometryTable que crea la columna y la registra en la tabla Geometry\_Columns.

```
CREATE TABLE calles (id integer, nombre varchar);
```

```
SELECT AddGeometryColumn( 'calles', 'the_geom', 32721 ,  
'LINESTRING', 2 );
```

```
CREATE TABLE hoteles (id integer, nombre varchar, lat real, lon  
real);
```

```
SELECT AddGeometryColumn( 'hoteles', 'the_geom', 32721 ,  
'POINT', 2 );
```



# Caso de estudio: PostGIS

- ❑ Insertar una geometría a partir de coordenadas
  - Creamos la geometría de cada registro a partir de los valores de los campos lat y lon de la tabla:  
UPDATE hoteles  
SET the\_geom = GeomFromText('POINT(' || lon || ' ' || lat ||  
' ',32721);
- ❑ Transformación a GML
  - Función AsGML(Geometry) permite obtener la representación en GML de un objeto geográfico.  
SELECT AsGML(the\_geom) from calles;



# Referencias

- ❑ Simple Features Standard

<http://www.opengeospatial.org/standards/sfs>

- ❑ PostGIS

<http://postgis.refrations.net/documentation/manual-1.4/>

- ❑ ArcSDE

<http://resources.arcgis.com/content/web-based-help>

- ❑ Curso “Análisis de Datos en SIG”. Docente: A. Vaisman

