

**B12 A must have for your BI solution!: DB2 Spatial Support**

*Scott Howard, Global Team Leader, BI and Information Integration, IBM*

Geography is perhaps the most common Business Intelligence dimension second only to the time dimension. Yes geography is present in most business intelligence requirements and solutions. How do you deal with geography today? You know addresses, but can you relate that to location? Does your database's knowledge of address allow you to dynamically calculate location and distance to satisfy user requirements like, "how far will customers travel before considering our competition?" (optimum store placement), or "show me the location of infected individuals" (medical and environmental research) and much more. Yes you already can handle modern geography with the DB2 Spatial Extender and the Informix Spatial Data Blade. Scott will present a detailed introduction to the DB2 Spatial Extender illustrating how it can help better solve many of today's business problems. DB2 SE's technical capabilities and implications will be discussed along with implementation specifics.

B12

# A must have for your BI solution!: DB2 Spatial Support

Scott Howard, IBM Learning Services

scottho@us.ibm.com

A decorative graphic consisting of several green circles of varying sizes, some overlapping, arranged in a horizontal line. A central green rounded rectangle with a purple border is superimposed on this graphic.

**IBM Data Management Technical Conference**

Anaheim, CA

Sept 9 - 13, 2002

# Trademarks

THE FOLLOWING TERMS ARE TRADEMARKS OR REGISTERED TRADEMARKS OF THE IBM CORPORATION IN THE UNITED STATES AND/OR OTHER COUNTRIES:

AIX, AS/400, DB2, OPERATING SYSTEM/2, OS/400, ES/9000, OS/390, OS/2, RISC, RISC SYSTEM/6000, SQL, SQL/DS, VM/ESA, IBM, APPROACH, NOTES

THE FOLLOWING TERMS ARE TRADEMARKS OR REGISTERED TRADEMARKS OF THE

MICROSOFT CORPORATION IN THE UNITED STATES AND/OR OTHER COUNTRIES: MICROSOFT, WINDOWS, WINDOWS NT, ODBC, WINDOWS 95

ESRI, ArcInfo, ArcSDE8, and ArcView are registered trademarks; Avenue, Spatial Data Engine, SDE, Analyst, the ESRI corporate logo, the ArcView GIS logo, and the SDE logo are trademarks; and [www.esri.com](http://www.esri.com) is a service mark of Environmental Systems Research Institute, Inc.

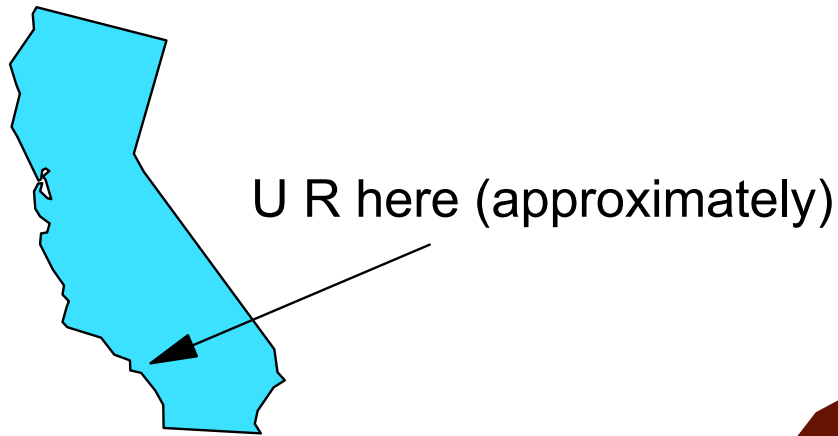
Java and HotJava are trademarks of Sun Microsystems, Inc.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited. (UNIX®)

Oracle is a registered trademark and Oracle8 is a trademark of Oracle Corporation.

# Why Spatial?

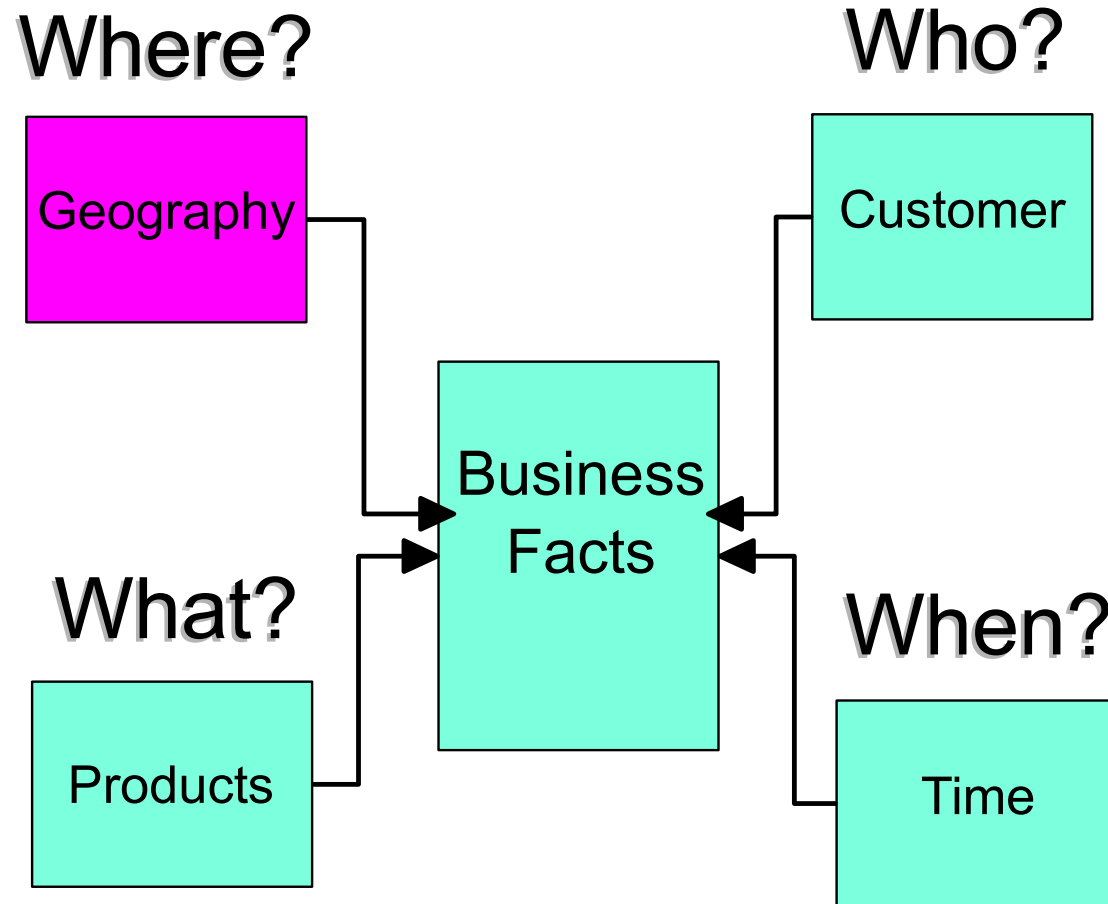
We don't do or need maps?



But do you do BI?



# Spatial is the Geography Dimension!



## The Traditional STAR Schema

# DB2 Spatial Extender

## Overview



IBM Data Management

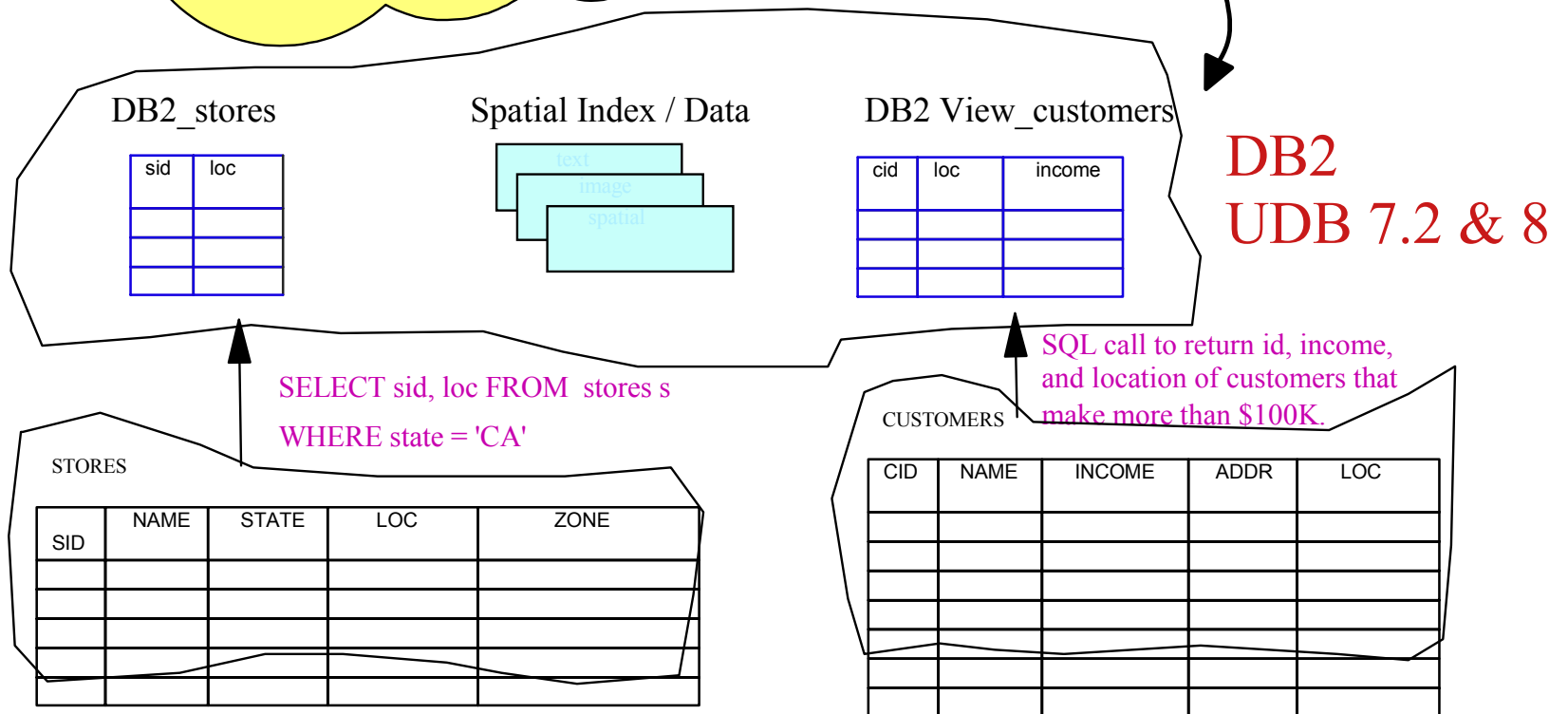
Anaheim, CA



# A Spatial Example

"Tell me the number and average income of customers who make more than \$100K and live within 100 miles of our California stores."

```
SELECT sid, count(*), avg(income)
FROM stores s, customers c
WHERE distance(s.loc, c.loc) < 100
AND s.state = 'CA'
AND income > 100000
GROUP BY sid;
```

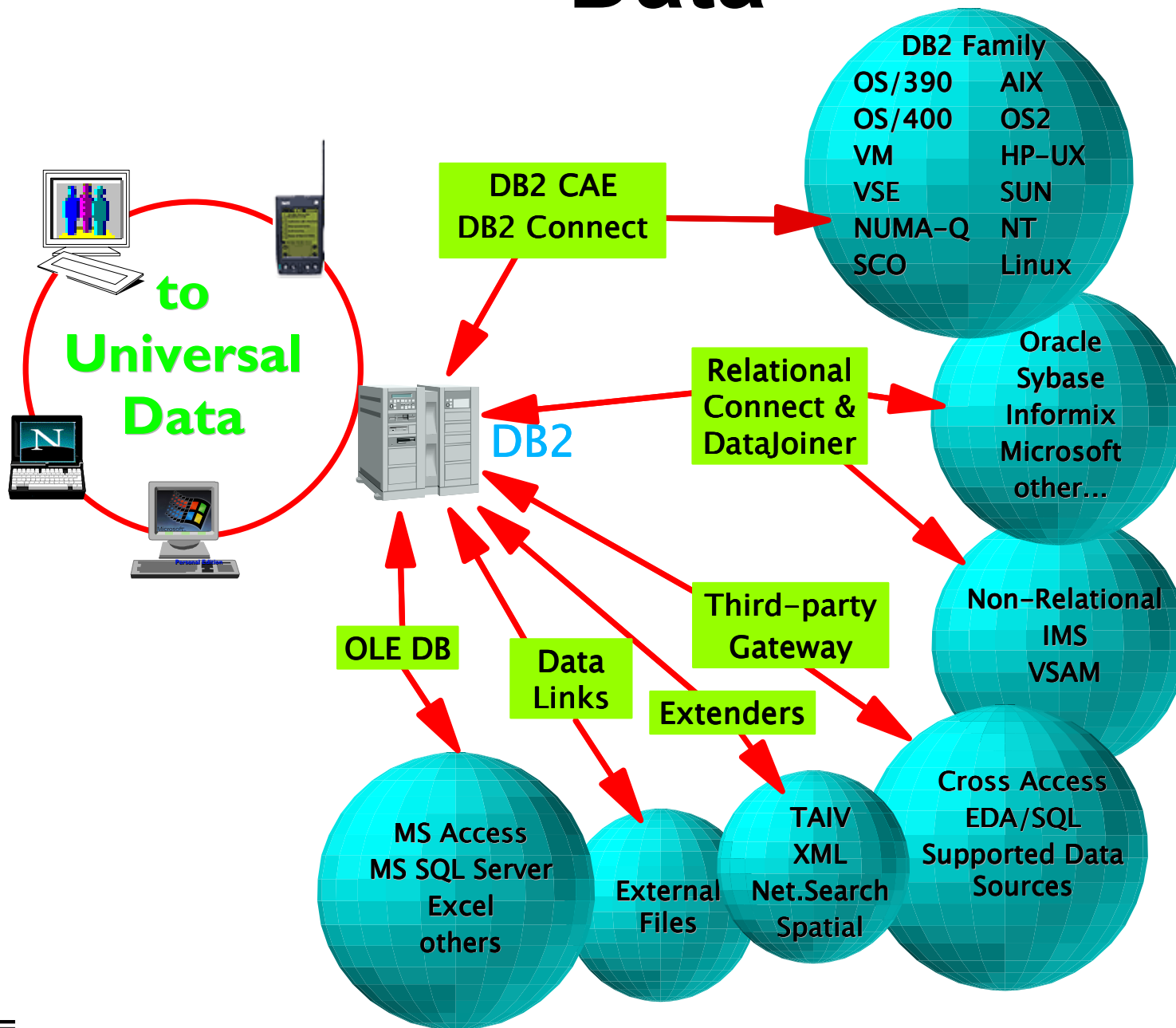


# IBM Spatial Extender Technology: - a joint IBM and ESRI effort

- IBM teams with the global GIS market leader ESRI:
  - ▶ Over 30 years of research and development of spatial technology
  - ▶ Broad portfolio of GIS products and tools
  - ▶ Collaboration combined best of both partners:
  - ▶ A strategic partnership
  - ▶ IBM provided object extensions for DB2 platforms
  - ▶ ESRI employed these extensions so they could construct spatial data types and functions.
- A history of partnership
  - ▶ Projects:
    - US Census DADS project
    - USDA Forest Service
  - ▶ Numerous local government engagements
  - ▶ Solutions:
    - IBM Petroleum Solutions Group
    - IBM Telecommunications Solutions Group
    - IBM Global Business Intelligence Group



# Universal Access to a Universe of Data



# More Types of Spatial Function

## ● *Comparison*

- Find the average customer distance from each department store

- ```
select s.id, AVG(db2gse.ST_Distance (c.location, s.location))
from customers c, stores s
where db2gse.ST_Within (c.location, s.zone) = 1
group by s.id
```

## ● *Data Exchange*

- Find the customer locations for those customers who live in the BayArea

- ```
select db2gse.ST_AsText (c.location, coordref (1))
from customers c
where db2gse.ST_Within(c.location, :BayArea) = 1
```

## ● *Transformation*

- Find the customers who live within the flood zone or within 2 miles from the boundary of the flood zone

- ```
select c.name, c.phoneNo, c.address
from customers c
where db2gse.ST_Within(c.location, ST_Buffer (:floodzone, 2)) = 1
```

## ● *Calculation*

- Find all streets longer than 10.5 miles

- ```
select s.name, s.id
from street s
where db2gse.ST_Length(s.path) >= 10.5
```

# Why keep your Spatial Data in DB2?

- Facilities for storing metadata
  - ▶ the Spatial Extender catalog
- Facilities for new data types
  - ▶ e.g. geometry, point, line, polygon, etc.
    - User-defined Structured Types
- Facilities for new operators on those types
  - ▶ e.g. intersects, contains, pointfromtext, etc.
    - User-defined Functions
- Facilities for storing the geometry
  - Large Objects, In-line Large Objects
- Facilities for properly indexing spatial data
  - ▶ e.g. grid indexes
    - Index Extensions
    - Spatial Index Advisor (tuning tool)
- Facilities for exploiting spatial index (optimization)

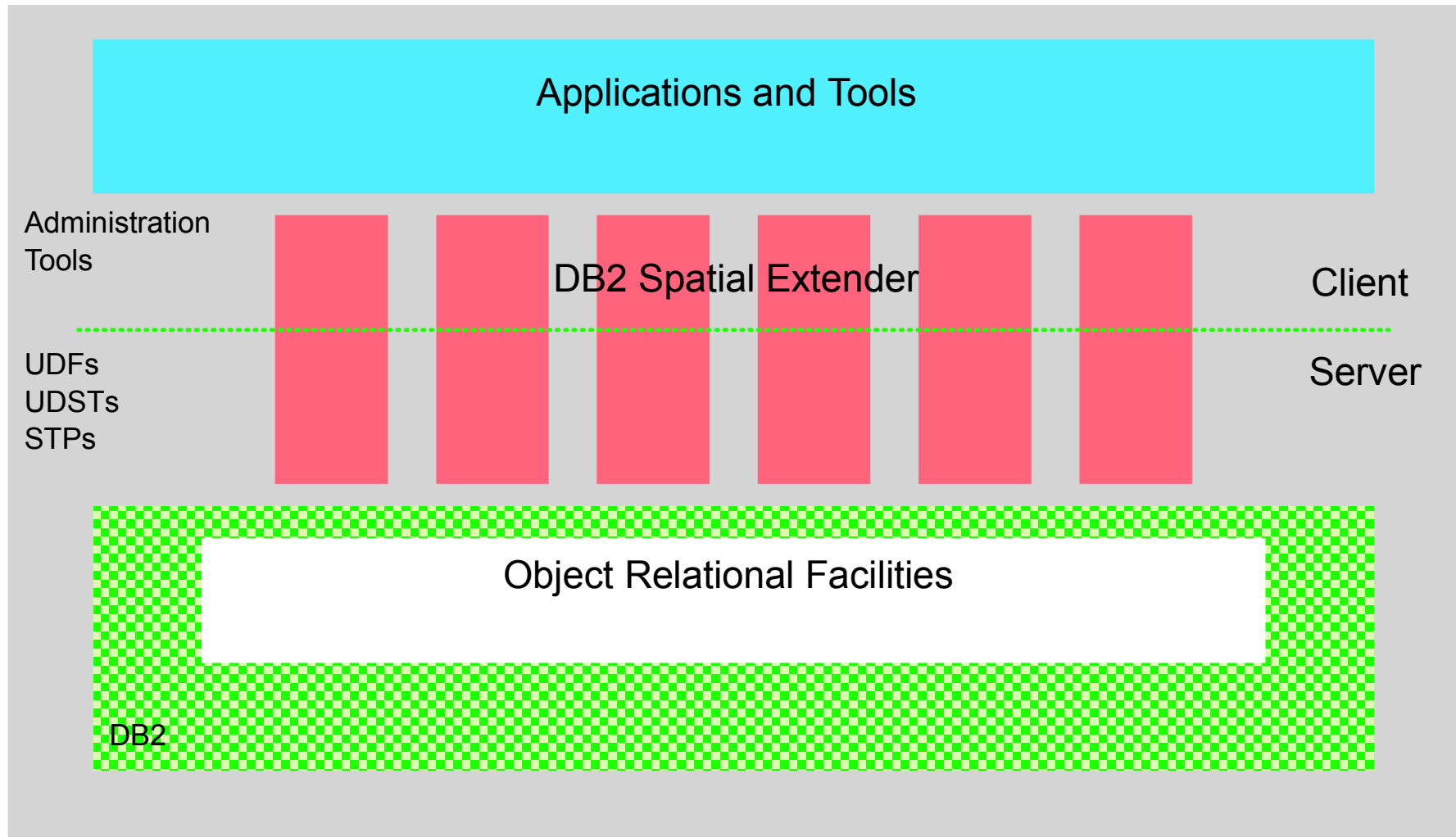
# DB2 Spatial Extender Features

- DB2 Control Center
  - ▶ Graphical User Interface
  - ▶ Includes spatial management functions
    - Enable/Disable spatial DB
    - Enable/Disable spatial table column
    - Create Spatial Reference System
    - Manage Spatial Layer Information
- Built with enhanced DB2 UDB capabilities
  - ▶ User Defined Structured Type (UDSTs)
  - ▶ Spatial Index extension
    - Fast search of two dimensional (x,y) data
    - Base on proven ESRI index
  - ▶ Spatial Index Advisor (gseidx)
  - ▶ Geocoder extension
    - ESRI's MATCH engine provided
    - Other geocoding tools may be added
    - Incremental geocoding supported
- New Improved loading/unloading tools
  - ▶ Shapefiles (ESRI)

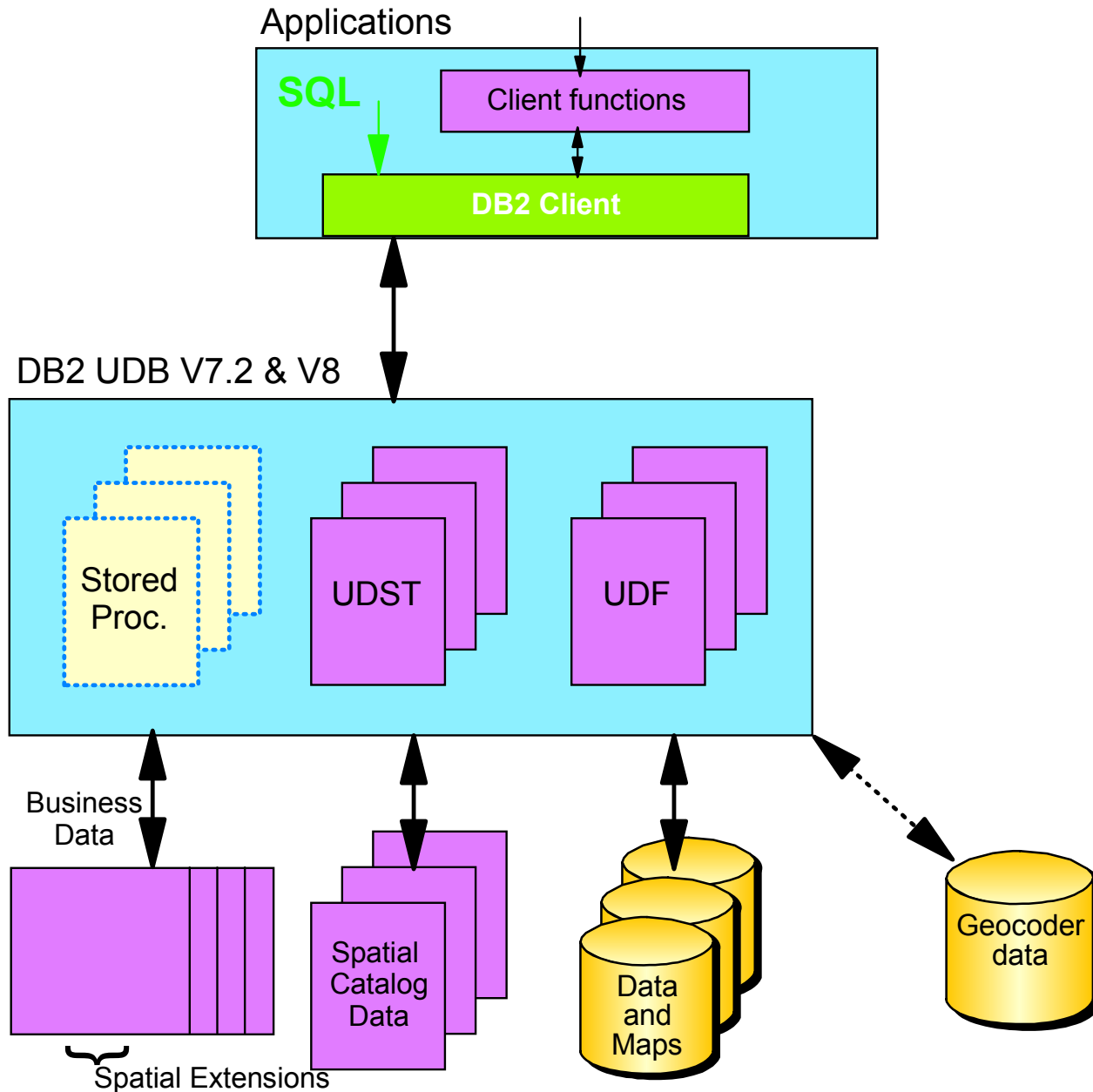
▶ SDE Import/Export (ESRI) ™ IBM Corporation 2002



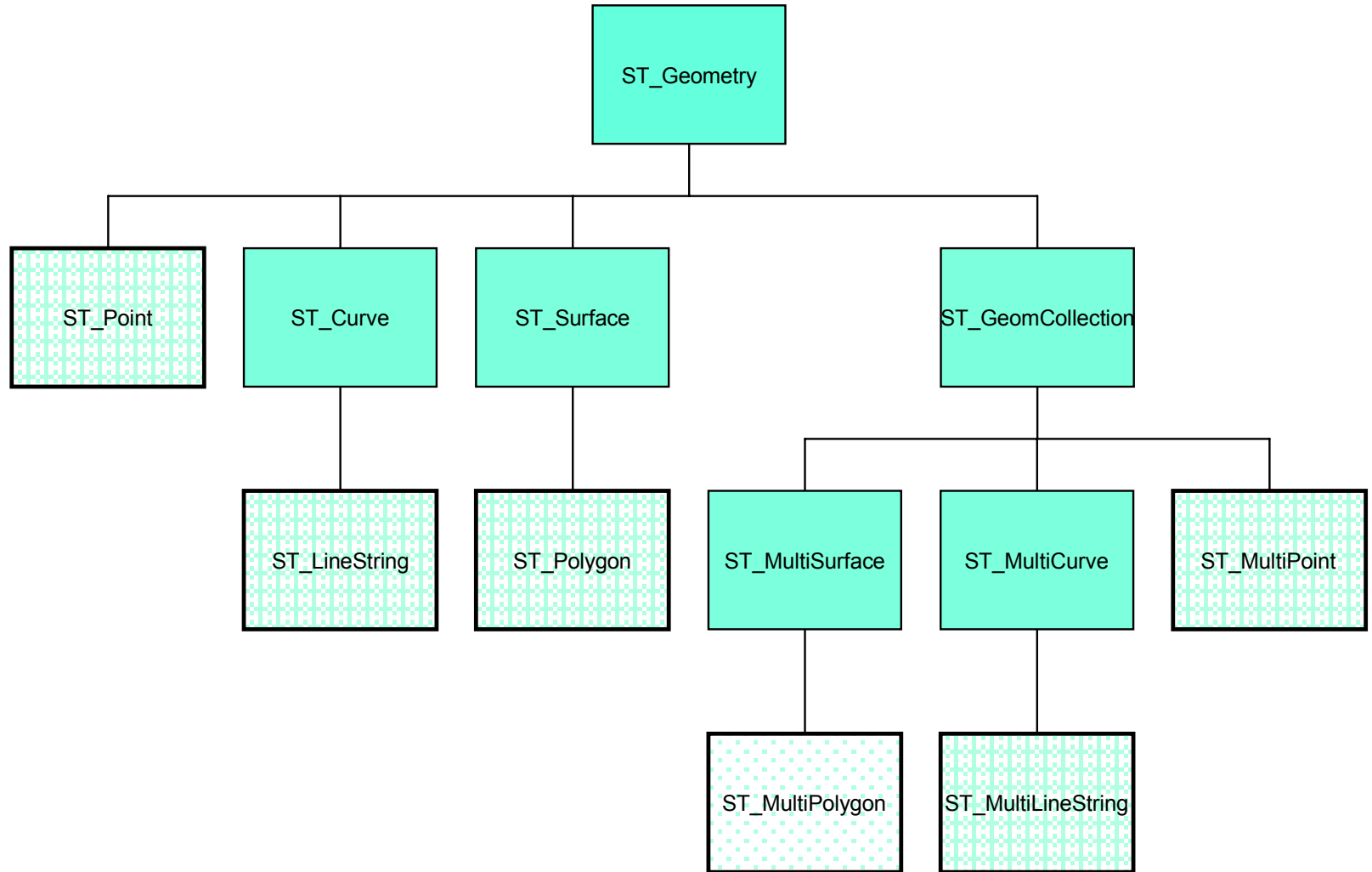
# Structure for DB2 Spatial Extender



# Architectural View of DB2 Spatial Extender



# Spatial Data Types



# DB2 Spatial Extender

Architectures



Data Management

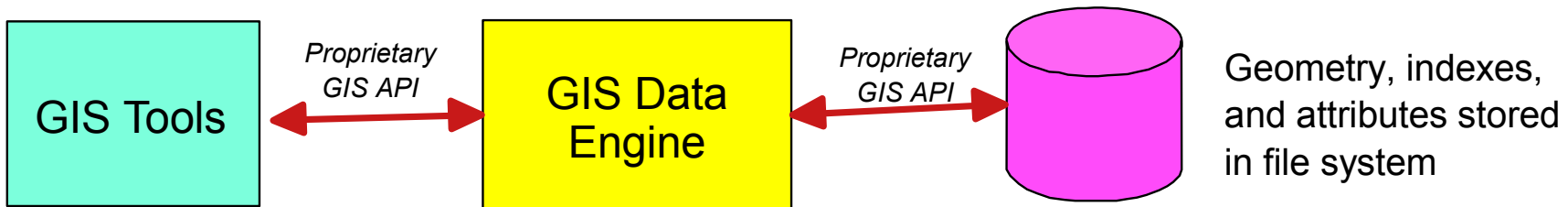
Anaheim, CA



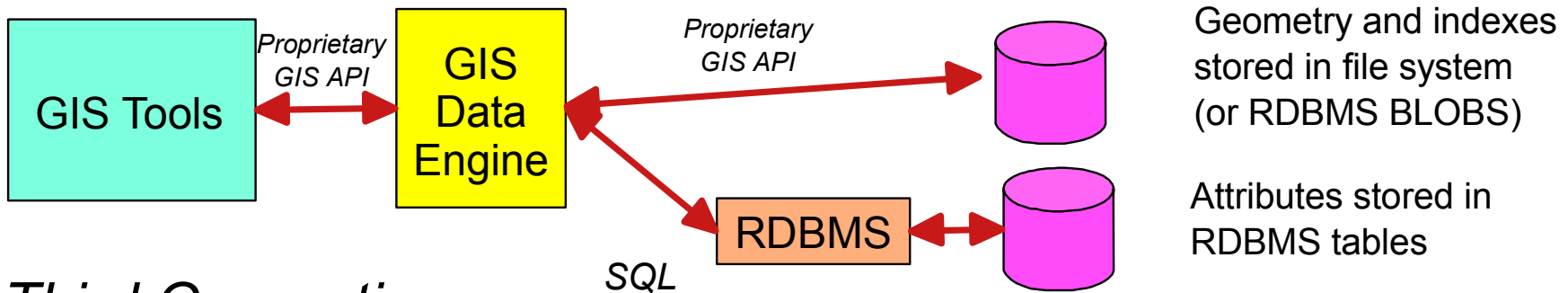


# Evolution of GIS Product Architectures

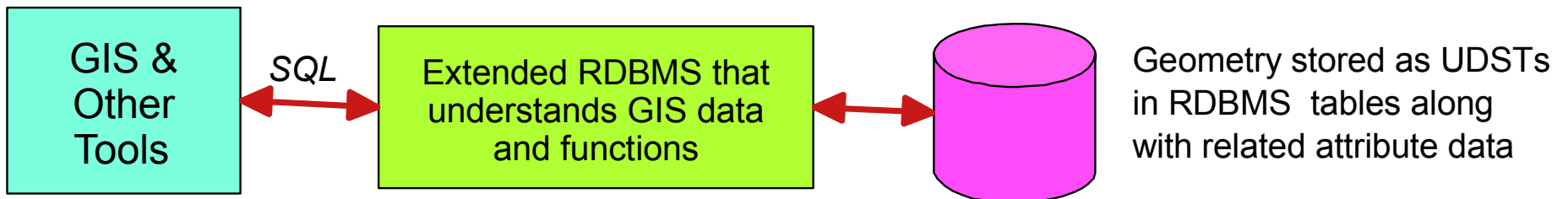
## First Generation:



## Second Generation:



## Third Generation:



# DB2 Spatial Extender

- DB2 Spatial Extender has the unique ability to "spatially enable" data across multiple, heterogeneous data sources.
  - ▶ Spatial Data stored locally, attribute data can be remote - e.g. geocoding address
- Users can now apply GIS capability to data stored in both DB2 and non-DB2 DBMSs
- Opens up significant opportunities to develop new applications and extend existing applications.

# Loosely Coupled Architecture

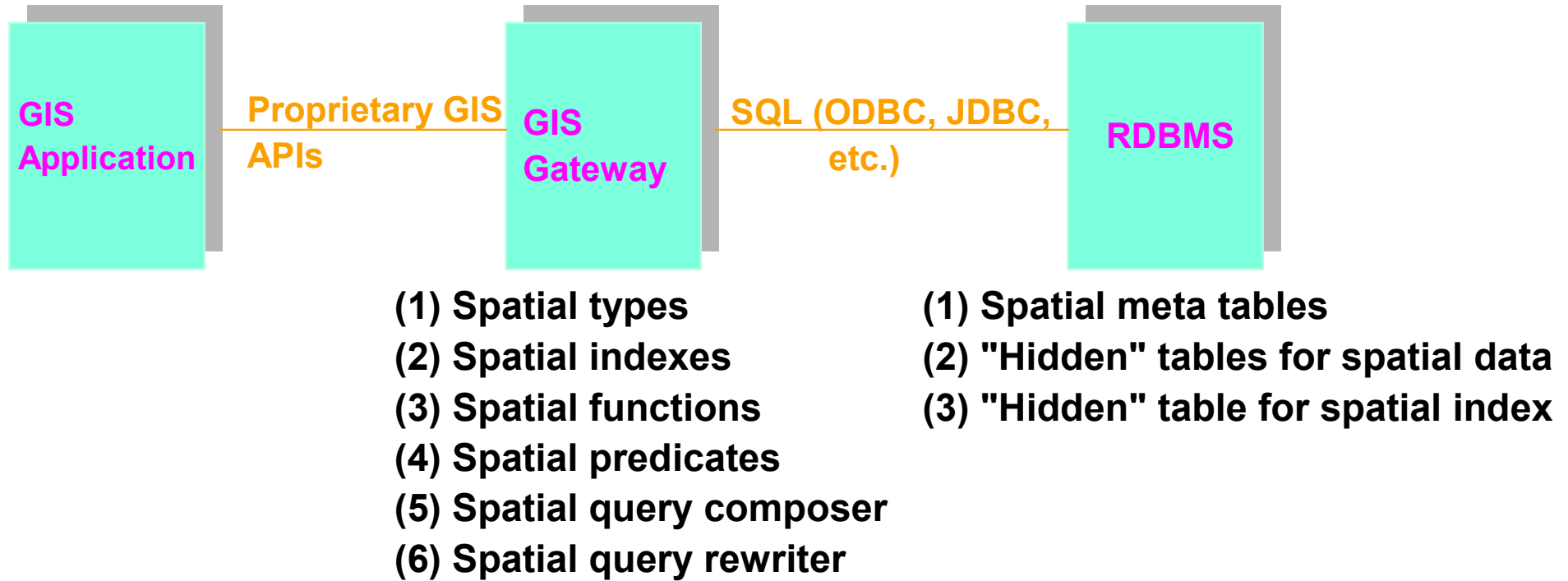
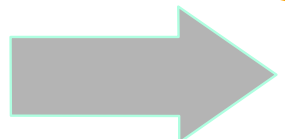


Table: **customers**  
 Constraint: **within(loc, :circle)**  
 Other constraint: **income > 50000**



```

    select * from customers c, index i, feature f
    where i.xmin > xmin(:circle) and
           i.ymin > ymin(:circle) and
           i.xmax < xmax(:circle) and
           i.ymax < ymax(:circle) and
           i.oid = f.oid and i.oid = c.oid and
           c.income > 50000
  
```

Some vendors call this a Third Generation Implementation

though it's really a Second Generation variant.



# Fully Integrated Architecture

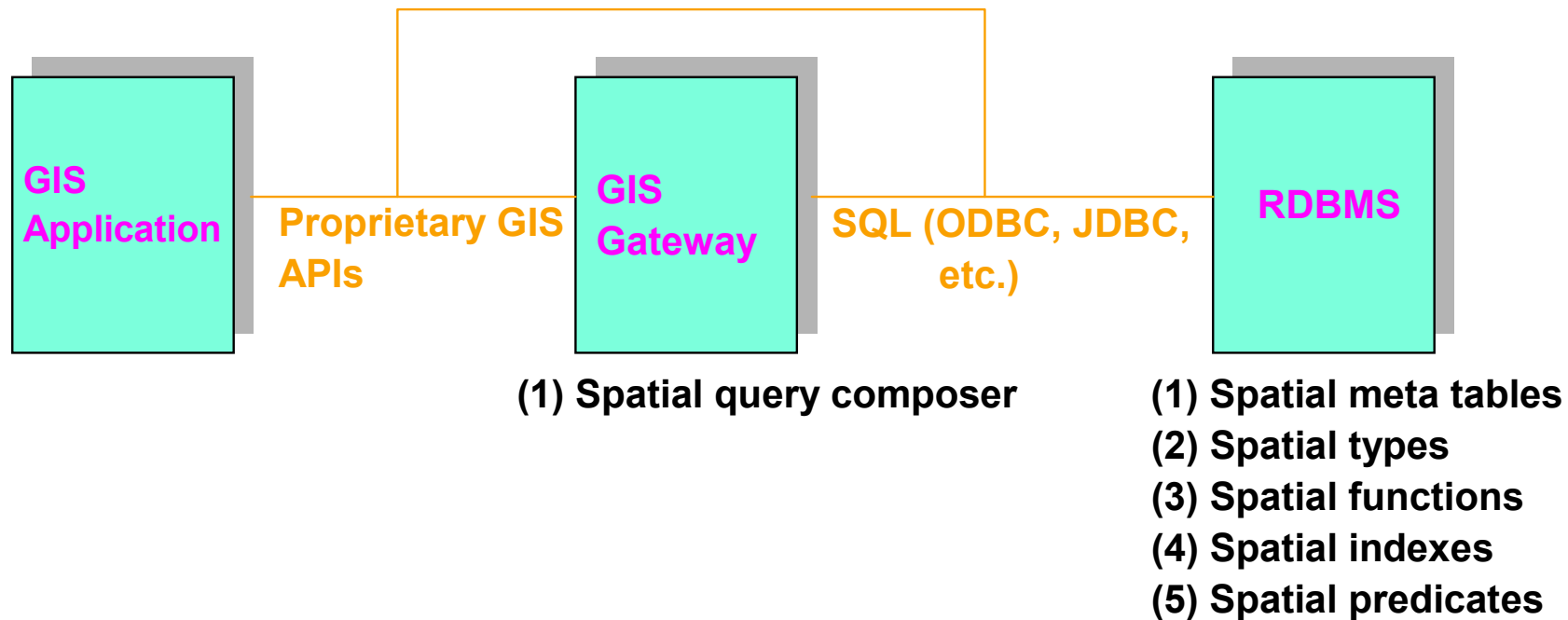
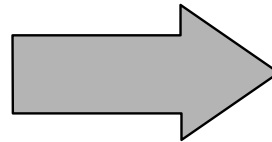


Table: **customers**

Constraint: **within(loc, :circle)**

Other constraint: **income > 50000**



**select \* from customers c**

**where within(c.location, :circle) and**

**c.income > 50000**

This a true Third Generation Implementation.

# The Business Table

## Customers

Cid	Address	City	State	Zip	...	
1	4335 Queen Anne Drive	San Jose	CA	95129	-	-
2	555 Bailey Avenue	San Jose	CA	95120	-	-
...	...	...	...	...	...	...
3000	1256 Prince Drive	Cupertino	CA	95129	-	-

- A new or existing RDBMS table
- contains attribute columns
- Also referred to as the Attribute table

# Spatially Enabling The Business Table

## Customers

Cid	Address	City	State	Zip	...	Loc
1	4335 Queen Anne Drive	San Jose	CA	95129	-	-
2	555 Bailey Avenue	San Jose	CA	95120	-	-
...	...	...	...	...	...	...
3000	1256 Prince Drive	Cupertino	CA	95129	-	-

- ▶ A spatial column in the business table
- ▶ A table with a spatial column that has been "registered" is called a **layer**
- ▶ Each row in the table is a **feature**
- ▶ Each column is an **attribute**
- ▶ Each layer has a spatial index to speed up queries involving coordinates
- ▶ SQL interface allows multiple spatial columns in the table
- ▶ SDE interface allows a single spatial column in the table

# DB2 Spatial Extender

Underneath the OR Hood

DB2  
UNIVERSAL  
database



Anaheim, CA



# SQL99 Example: type and function definitions

1

```
CREATE TYPE envelope (  
  xmin INT,  
  ymin INT,  
  xmax INT,  
  ymax INT);
```

2

```
CREATE TYPE st_geometry (  
  gtype INT,  
  refsystem INT,  
  tolerance FLOAT,  
  area FLOAT,  
  length FLOAT,  
  mbr envelope,  
  numparts INT,  
  numpoints INT,  
  points BLOB(1m),  
  zvalue BLOB(500k),  
  measure BLOB(500k));
```

3

```
CREATE TYPE ST_POINT UNDER  
st_geometry;  
CREATE TYPE st_line UNDER st_geometry;  
CREATE TYPE ST_Polygon UNDER  
st_geometry;
```

4

```
CREATE FUNCTION distance  
(s1 st_geometry, s2 st_geometry)  
RETURNS INT  
EXTERNAL NAME  
'db2gsefn!shapedist'
```

```
CREATE FUNCTION st_within  
(s1 st_geometry, s2 st_geometry)  
RETURNS INT  
EXTERNAL NAME  
'db2gsefn!shapewithin'
```

...

...



# SQL99 Tables with Spatial Columns

5

```
CREATE TABLE customers (  
  cid      INT,  
  name    VARCHAR(20),  
  income  INT,  
  addr    CHAR(20)  
  loc     ST_POINT);
```

CUSTOMERS

CID	NAME	INCOME	ADDR	LOC

```
CREATE TABLE stores (  
  sid      INT,  
  name    VARCHAR(20),  
  addr    CHAR(20),  
  loc     ST_POINT,  
  zone    ST_Polygon);
```

STORES

SID	NAME	ADDR	LOC	ZONE

```
CREATE TABLE sales (  
  sid      INT,  
  cid      INT,  
  amount  INT);
```

SALES

SID	CID	AMOUNT

# User-defined Functions as User-defined Structured Type Constructor(s)

6

Simple UDST Constructor

```
CREATE FUNCTION shape4text  
(s VARCHAR(255))  
RETURNS st_geometry  
EXTERNAL NAME  
'db2gsefn!shape4texr'  
LANGUAGE c  
PARAMETER db2sql STYLE  
NOT VARIENT NOT FENCED  
NOT NULL CALL  
NO SQL NO ETERNAL ACTION;
```

7

Simple Trigger

```
CREATE TRIGGER cust_loc_insert...  
SET customers.loc =  
shape4text(customers.addr);
```

# User-defined Indexing on User-defined Structured Types

## 8 Create User-defined Index Extension

```
CREATE INDEX EXTENSION grids
  (levels VARCHAR(20))
  WITH INDEX KEYS FOR (s st_geometry)
  GENERATED BY gridentry
  (s..mbr..xmin,s..mbr..ymin,
   s..mbr..xmax,s..mbr..ymax)
  WITH SEARCH METHODS FOR INDEX
  KEYS
  (gx INT, gy INT, x1 INT, y1 INT,
   x2 INT, y2 INT)
  WHEN within_search (r st_geometry)
  RANGE THRU gridswithin
  (r..mbr..xmin,r..mbr..ymin,
   r..mbr..xmin,r..mbr..ymax,
   levels)
  CHECK WITH gridxwithin
  (gx,gy,x1,y1,x2,y2,
   r..mbr..xmin,r..mbr..ymin,
   r..mbr..xmin,r..mbr..ymax,
   levels)
  WHEN overlap_search (r geometry) ...
  WHEN contains_search (r geometry) ...
  WHEN withindist_search (r geometry,d INT)...
  ;
```

# Spatial Indexes for Spatial Columns

9

Create Spatial Indexes on Spatial Columns  
Using the Index Extension

CUSTOMERS

CID	NAME	INCOME	ADDR	LOC

STORES

SID	NAME	ADDR	LOC	ZONE

CREATE INDEX customersx  
ON customers (loc)  
EXTEND USING SPATIAL  
INDEX (10, 100, 1000);

CREATE INDEX storesx1  
ON stores (loc)  
EXTEND USING SPATIAL  
INDEX (10, 100, 1000);

CREATE INDEX storesx2  
ON stores (zone)  
EXTEND USING SPATIAL  
INDEX (50, 500, 5000);

# Spatial Relationships as User Defined Predicates

10

Create Spatial Functions and identify them as predicates that may exploit Spatial Index access

```
CREATE FUNCTION st_within
```

```
(s1 st_geometry, s2 st_geometry)
```

```
RETURNS INT
```

```
EXTERNAL NAME
```

```
'db2gsefn!shapewithin'
```

```
...
```

```
➔ AS PREDICATE WHEN st_within (x, y)=1
```

```
FILTER BY mbr_within (      x..mbr.xmin, x..mbr..ymin,  
                           x..mbr..xmax, x..mbr..ymax,  
                           y..mbr..xmin, ...)
```

```
SEARCH BY INDEX EXTENSION grids
```

```
WHEN KEY(x) USE within_search (y)
```

```
WHEN KEY(y) USE contains_search (x);
```

# Index Exploitation for complex predicates

11

Optimizer Can Now Use Spatial Index for Selected Predicates

## Simple Spatial Predicate

```
SELECT *  
FROM   stores s, customers c  
WHERE  st_distance (c.loc, s.loc)<100;
```

## Compound Spatial Predicate

```
SELECT *  
FROM   stores s, customers c  
WHERE  st_distance(c.loc, s.loc)<100  
OR  
       st_within(c.loc, s.zone)=1;
```

## Nested Spatial Function

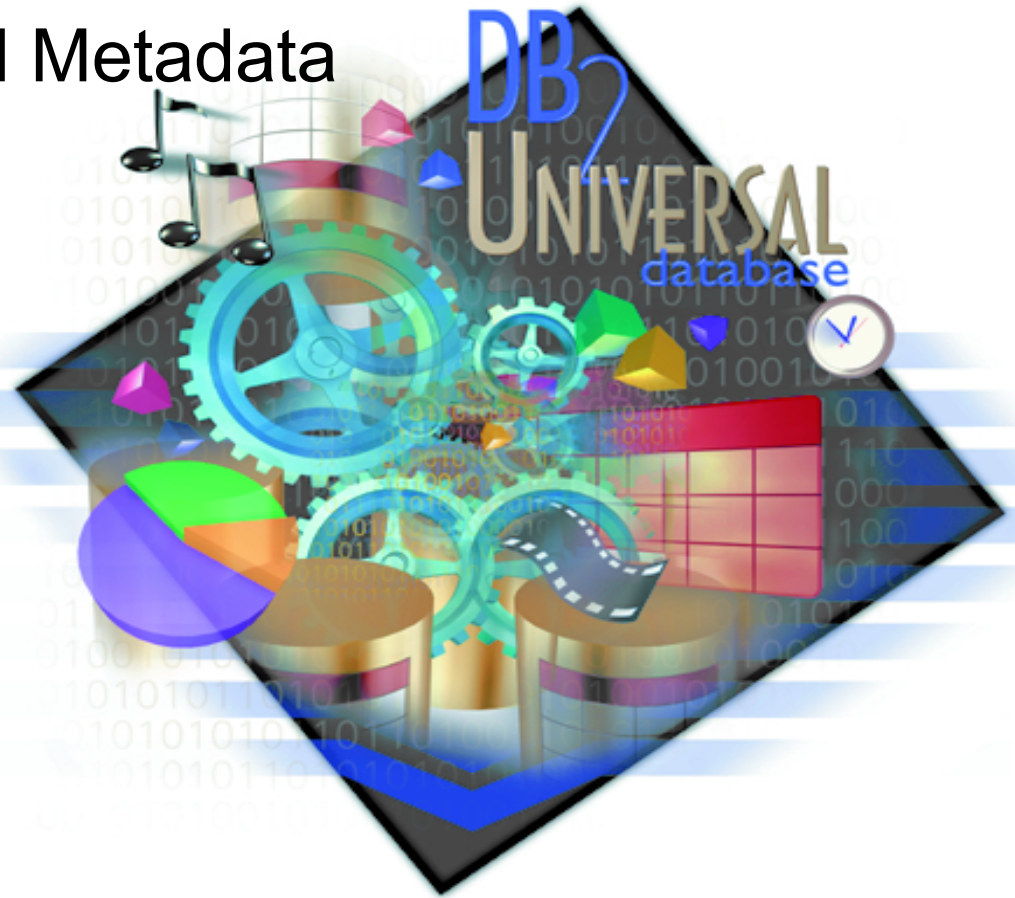
```
SELECT *  
FROM   stores s, customers c  
WHERE  st_within (c.loc,  
                 st_buffer(s.loc, 100))=1;
```

# DB2 Spatial Extender

Terminology and Metadata



Anaheim, CA



# The Spatial Extender Meta-Data Views

Catalog of spatial data in the database

Maintained by spatial extender administration facilities which populate and modify the rows in the tables/views

- DB2GSE.COORD\_REF\_SYS - the coordinate systems provided by the spatial extender
- DB2GSE.GEOMETRY\_COLUMNS - spatial layers, spatial column, spatial data type, and spatial reference system ID
- DB2GSE.SPATIAL\_REF\_SYS - the spatial reference systems you are using - or have created
- DB2GSE.SPATIAL\_GEOCODER - the available geocoders



# DB2 Spatial Extender

## Spatial User Defined Functions



IBM Data Management

Anaheim, CA



# Spatial Standards

- SQL99
  - ▶ new version of the SQL standard
  - ▶ contains major Object-Relational extensions
    - user-defined data types
    - user-defined functions (and procedures)
    - large object support
  - ▶ powerful SQL query extensions to model business rules
- SQL/MM Part 3
  - ▶ defines SQL extensions (user-defined types, functions, ...) for
    - Spatial (compatible with OGC)
    - Text (Part 2)
    - Still Image (Part 5)
  - ▶ exceptions
    - arrays
    - parametric curves
- OGIS/OGC
  - ▶ Open GIS Consortium Simple Features Specification defines interoperability standards for GIS data management
  - ▶ DB2 UDB 7.2 Spatial Extender is compliant with 1.1 release of the standard

# Over UDFs 80 and growing ...

ST\_Contains(g1,g2)  
ST\_Touches(g1,g2)  
ST\_Within(g1,g2)  
ST\_Overlaps(g1,g2)  
ST\_Intersects(g1, g2)  
ST\_Crosses(g1,g2)  
ST\_Disjoint(g1,g2)  
ST\_Relate(g1, g2, r1)  
ST\_Equals(g1,g2)  
ST\_OrderingEquals(g1, g2)  
EnvelopesIntersect(g1, g2)

ST\_Intersection(g1,g2)  
ST\_Difference(g1,g2)  
ST\_Union(g1,g2)  
ST\_SymmetricDiff(g1, g2)  
ST\_Buffer(g1)  
ST\_PointOnSurface(g1)  
ST\_Boundary(g1)  
ST\_Envelope(g1)  
ST\_Centroid(g1)  
ST\_Perimeter(g1)  
ST\_ConvexHull(g1)

ST\_Area(g1)  
ST\_Distance(g1, g2)  
ST\_Length(g1)  
LocateAlong(g1, d1)  
LocateBetween(g1, d1, d2)

GeometryFromShape(s1)  
PointFromShape(s1)  
LineFromShape(s1)  
PolyFromShape(s1)  
MPointFromShape(s1)  
MLineFromShape(s1)  
MPolyFromShape(s1)  
ShapeToSQL(s1)  
AsBinaryShape(g1)

ST\_GeomFromWKB(w1)  
ST\_PointFromWKB(w1)  
ST\_LineFromWKB(w1)  
ST\_PolyFromWKB(w1)  
ST\_MPointFromWKB(w1)  
ST\_MLineFromWKB(w1)  
ST\_MPolyFromWGB(w1)  
ST\_WKBToSQL(w1)  
ST\_AsBinary(g1)

ST\_GeometryFromText(t1)  
ST\_PointFromText(t1)  
ST\_LineFromText(t1)  
ST\_PolyFromText(t1)  
ST\_MPointFromText(t1)  
ST\_MLineFromText(t1)  
ST\_MPolyFromText(t1)  
ST\_WKTToSQL(t1)  
ST\_AsText(g1)

ST\_Point(d1, d2)  
ST\_Polygon(g1)  
ST\_Transform(g1, srs1)  
ST\_SRID(g1)  
ST\_X(p1)  
ST\_Y(p1)  
Z(p1)  
M(p1)

ST\_IsClosed(g1)  
ST\_IsEmpty(g1)  
ST\_IsRing(g1)  
ST\_IsSimple(g1)  
ST\_IsValid(g1)  
Is3D(g1)  
IsMeasured(g1)

ST\_StartPoint(c1)  
ST\_EndPoint(c1)  
ST\_CoordDim(g1)  
ST\_Dimension(g1)  
ST\_GeometryType(g1)  
ST\_NumPoints(g1)  
ST\_PointN(g1)  
ST\_ExteriorRing(p1)  
ST\_NumInteriorRing(g1)  
ST\_InteriorRingN(g1)  
ST\_NumGeometries(g1)  
ST\_GeometryN(g1)

# spatial functions

IBM Data Management Technical Conference



© IBM Corporation 2002

# Spatial Functions

## ■ Spatial Relationship Predicates



- ▶ ST\_Contains(g1,g2)
- ▶ ST\_Touches(g1,g2)
- ▶ **ST\_Within(g1,g2)**
- ▶ **ST\_Overlaps(g1,g2)**
- ▶ ST\_Intersects(g1, g2)
- ▶ ST\_Crosses(g1,g2)
- ▶ ST\_Disjoint(g1,g2)
- ▶ ST\_Relate(g1, g2, r1)
- ▶ ST\_Equals(g1,g2)
- ▶ ST\_OrderingEquals(g1, g2)
- ▶ EnvelopesIntersect(g1, g2)

# Spatial Functions

## ■ Spatial Operators

- ▶ ST\_Intersection(g1,g2)
- ▶ ST\_Difference(g1,g2)
- ▶ ST\_Union(g1,g2)
- ▶ ST\_SymmetricDiff(g1, g2)
- ▶ **ST\_Buffer(g1)**
- ▶ ST\_PointOnSurface(g1)
- ▶ ST\_Boundary(g1)
- ▶ ST\_Envelope(g1)
- ▶ ST\_Centroid(g1)
- ▶ ST\_Perimeter(g1)
- ▶ ST\_ConvexHull(g1)

# Spatial Functions

## ■ Geometry computation

- ▶ ST\_Area(g1)
- ▶ **ST\_Distance(g1, g2)**
- ▶ ST\_Length(g1)
- ▶ LocateAlong(g1, d1)
- ▶ LocateBetween(g1, d1, d2)

# Spatial Functions

## ■ Constructing Geometry from ESRI shape representation

- ▶ GeometryFromShape(s1)
- ▶ PointFromShape(s1)
- ▶ LineFromShape(s1)
- ▶ PolyFromShape(s1)
- ▶ MPointFromShape(s1)
- ▶ MLineFromShape(s1)
- ▶ MPolyFromShape(s1)
- ▶ ShapeToSQL(s1)
- ▶ AsBinaryShape(g1)

# Spatial Functions

## ■ Constructing Geometry from Well Known Binary

- ▶ ST\_GeomFromWKB(w1)
- ▶ ST\_PointFromWKB(w1)
- ▶ ST\_LineFromWKB(w1)
- ▶ ST\_PolyFromWKB(w1)
- ▶ ST\_MPointFromWKB(w1)
- ▶ ST\_MLineFromWKB(w1)
- ▶ ST\_MPolyFromWKB(w1)
- ▶ ST\_WKBToSQL(w1) ????
- ▶ ST\_AsBinary(g1)



# Spatial Functions

## ■ Constructing A Geometry from WellKnownText

- ▶ ST\_GeometryFromText(t1)
- ▶ **ST\_PointFromText(t1)**
- ▶ ST\_LineFromText(t1)
- ▶ ST\_PolyFromText(t1)
- ▶ ST\_MPointFromText(t1)
- ▶ ST\_MLineFromText(t1)
- ▶ ST\_MPolyFromText(t1)
- ▶ ST\_WKTTToSQL(t1)
- ▶ **ST\_AsText(g1)**

# Spatial Functions

## ■ Observer Functions

- ▶ ST\_X(p1)
- ▶ ST\_Y(p1)
- ▶ Z(p1)
  
- ▶ M(p1)
- ▶ ST\_Point(d1, d2)
- ▶ ST\_Polygon(g1)
  
- ▶ ST\_Transform(g1, srs1)
- ▶ ST\_SRID(g1)

# Spatial Functions

## ■ Geometry Boolean Tests

- ▶ ST\_IsClosed(g1)
- ▶ ST\_IsEmpty(g1)
- ▶ ST\_IsRing(g1)
- ▶ ST\_IsSimple(g1)
- ▶ ST\_IsValid(g1)
- ▶ Is3D(g1)
- ▶ IsMeasured(1)

# Spatial Functions

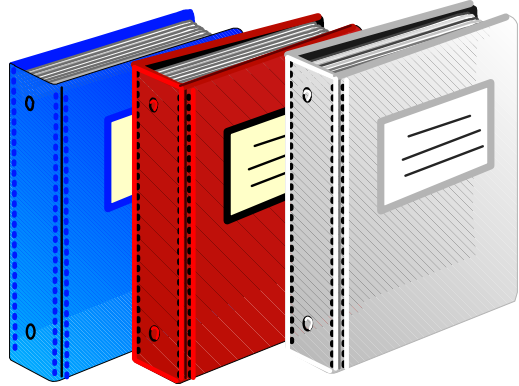
## ■ Geometry Properties

- ▶ ST\_StartPoint(c1)
- ▶ ST\_EndPoint(c1)
- ▶ ST\_CoordDim(g1)
- ▶ ST\_Dimension(g1)
- ▶ **ST\_GeometryType(g1)**
- ▶ **ST\_NumPoints(g1)**
- ▶ ST\_PointN(g1)
- ▶ ST\_ExteriorRing(p1)
- ▶ ST\_NumInteriorRing(g1)
- ▶ ST\_InteriorRingN(g1)
- ▶ **ST\_NumGeometries(g1)**
- ▶ ST\_GeometryN(g1)

# Spatial Predicates used to exploit the spatial index

- ST\_Equals
- ST\_Disjoint
- ST\_Touches, ST\_Overlaps, ST\_Crosses, ST\_Intersects
- ST\_Within, ST\_Contains
- ST\_Envelope
- ST\_Distance

# Complete list of spatial functions and predicates



Reference: Chapter 14 of the DB2 Spatial Extender User's Guide and Reference

<http://www.ibm.com/software/data/spatial>

Approximately 80 Spatial Functions  
for SQL Queries

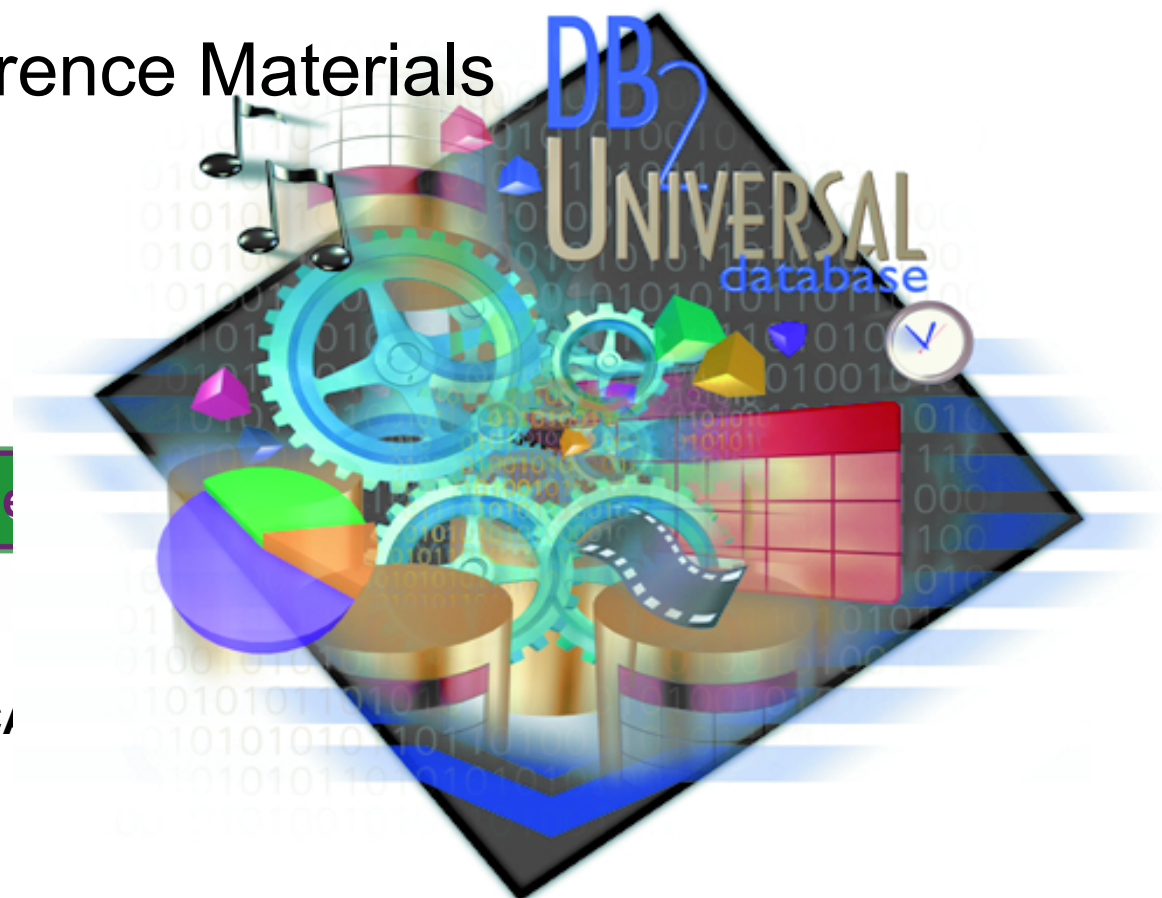
# DB2 Spatial Extender

Reference Materials

DB2  
UNIVERSAL  
database



Anaheim, CA



# Integer Coordinates and Scale

- All coordinates are stored as positive 32-bit integers
  - Integer math faster than floating point math
  - Integers require less storage space
  - Coordinates compressed using relative coordinates
- SDE converts coordinates to double-precision floating point format for the client
  - over 2 billion positive integer values in a 32 bit integer
  - x,y offsets position the coordinate space
  - xyscale controls the size of the cells, or minimum resolution
- Z and M coordinates have their own offsets and scales

Spatial Extender converts floating point to integer

$$\text{integer} = \text{truncate} (((\text{float} - \text{offset}) * \text{xyscale}) + 0.5)$$

Spatial Extender converts integer to floating point

$$\text{float} = ((\text{integer} / \text{xyscale}) + \text{offset})$$

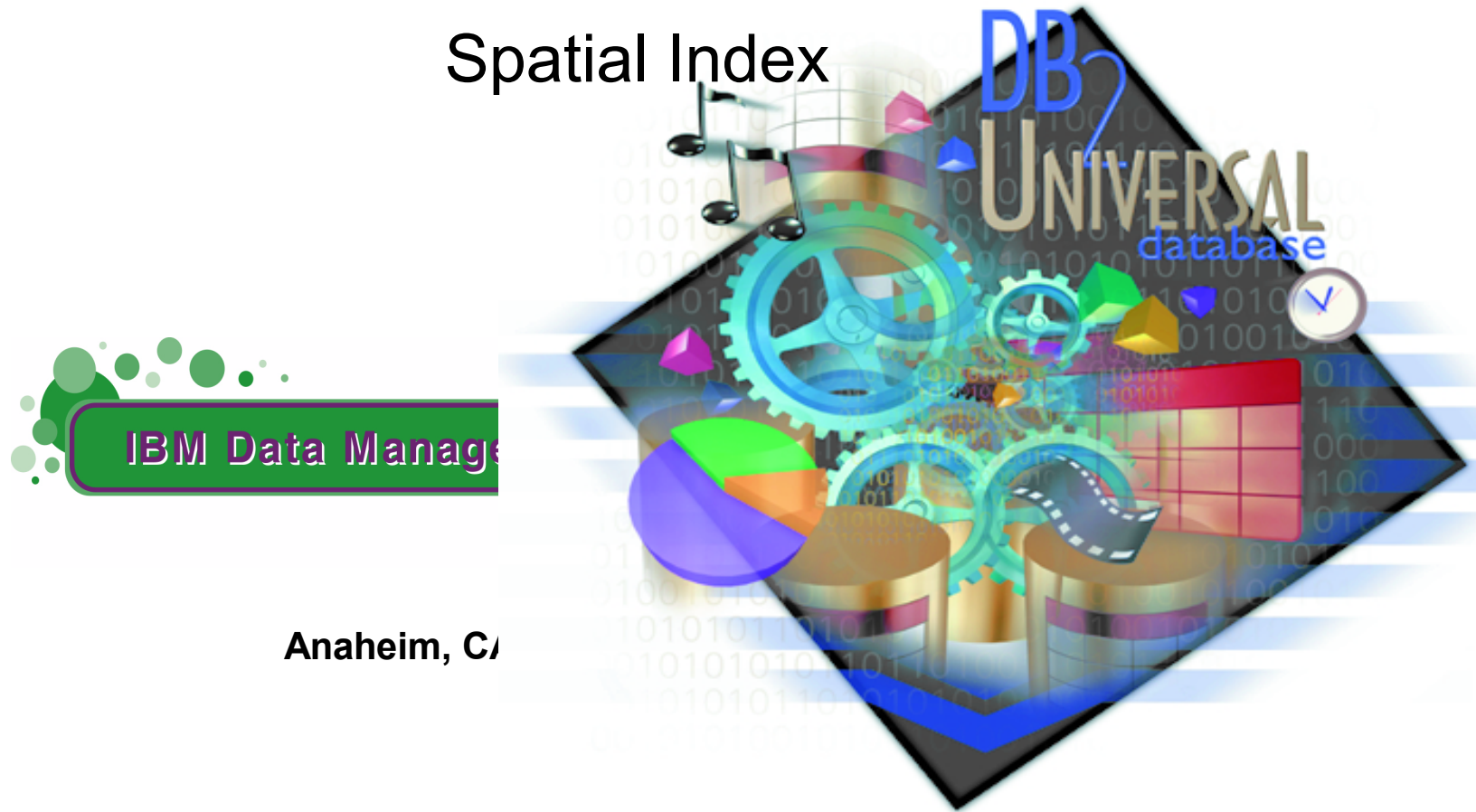


# Coordinate Compression

- Spatial Extender stores compressed relative coordinates
  - the lower left corner of the feature envelope is stored as an absolute coordinate
  - the features' coordinates are stored as relative offsets
- Compression ratios are data dependent
  - features with more vertices compress better
  - features that are small relative to xyscale compress better

# DB2 Spatial Extender

Spatial Index



IBM Data Management

Anaheim, CA

# The need for a spatial index

- Spatial queries are typically 2-dimensional
- Native B-tree index is insufficient
- Efficient index scheme is essential to the query performance !  
e.g. `select * from customers where within (location, :circle) = 1`
- 0-D objects (e.g. location of buildings) can be B-tree-indexed based on the X and Y coordinates  
But not with the best possible performance !
- Neither 1-D (e.g. road segment) nor 2-D (e.g. boundary of a lake) objects can be indexed this way
- The coordinate system is divided into grid blocks

# Rules for Index Exploitation

- The spatial predicate must be used in the WHERE clause
- The spatial predicate must be on the left hand side of the comparison
- Equality comparisons must use the integer constant 1
- There must be a spatial column used in the predicate as the search target and there must be a spatial index created on that column

## Which of the following are index eligible?

Select \* from customers c where db2sge.ST\_Within (c.location, :BayArea) = 1

Select \* from customers c where db2sge.ST\_Distance(c.location, :SanJose) < 10







Select \* from customers c where db2sge.ST\_Length (c.location) > 10

Select \* from customers c where 1 = db2sge.ST\_Within (c.location, :BayArea)

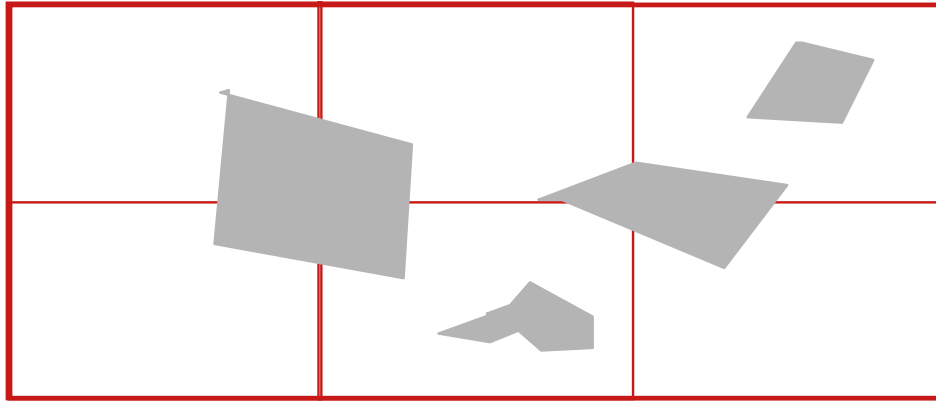
Select \* from customers c where db2sge.ST\_Within (c.location, :BayArea) = 2

Select \* from customers c where db2sge.ST\_Within (:SanJose, :BayArea) = 1

# Index Exploitation Examples - Answers

-  Select \* from customers c where db2sge.ST\_Within (c.location, :BayArea) = 1
-  Select \* from customers c where db2sge.ST\_Distance(c.location, :SanJose) < 10
-  Select \* from customers c where db2sge.ST\_Length (c.location) > 10  
ST\_Length is a function, not a spatial predicate
-  Select \* from customers c where 1 = db2sge.ST\_Within (c.location, :BayArea)  
The 1 must be on the right hand side of the predicate.
-  Select \* from customers c where db2sge.ST\_Within (c.location, :BayArea) = 2  
Equality comparisons must use the integer constant 1
-  Select \* from customers c where db2sge.ST\_Within (:SanJose, :BayArea) = 1  
A spatial column must be used in the predicate and there must be a spatial index on the column. SanJose and BayArea are host variables - not spatial columns.

# The Spatial Index Grids



Records which grids each feature resides in

Objects are indexed by the grid level, the overlapped grid blocks, and the MBR of the object

- ▶ A uniformly-spaced square indexing grid
  - Each feature exists in one or more grids
  - Multiple features can exist in a single grid
- ▶ Features are not split by grid or stored by grids
  - are used to speed up envelope searches
- ▶ Up to 3 levels of grids for the spatial index
  - A spatial index is like a two-dimensional column index

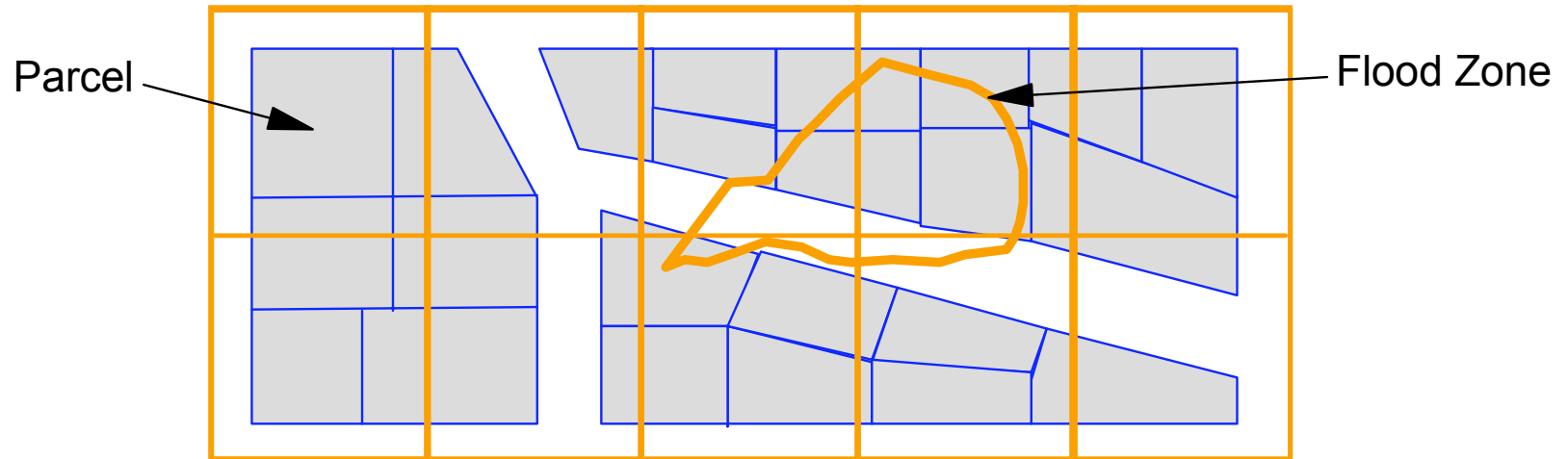
With "traditional" SDE the spatial index is implemented in the S table.  
With the spatial extender the spatial index is implemented as an index extension.

# How the Spatial Index Works

1. Eliminate features by grid of search space
  - compare the tiles of the filter to the tiles of the spatial index grid (gets you in the neighborhood)
2. Eliminate features by envelope search space
  - compare the envelope of the filter to the features' envelope
3. Eliminate features by coordinate comparison
  - compare the coordinates of the spatial filter against the coordinates of the remaining features (expensive)

Spatial index search avoids accessing too much data (full data scans)

# Let's walk through an example of Spatial Index Filtering



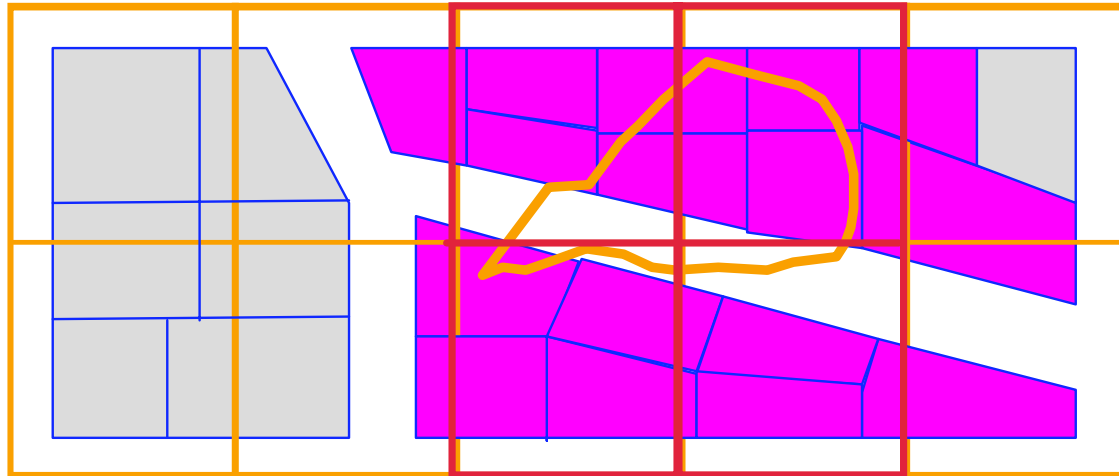
**Find all Parcels that intersect the Flood Zone?**

Remember:

The goal is to avoid comparing all shapes, therefore we will use a 3 level filtering process!



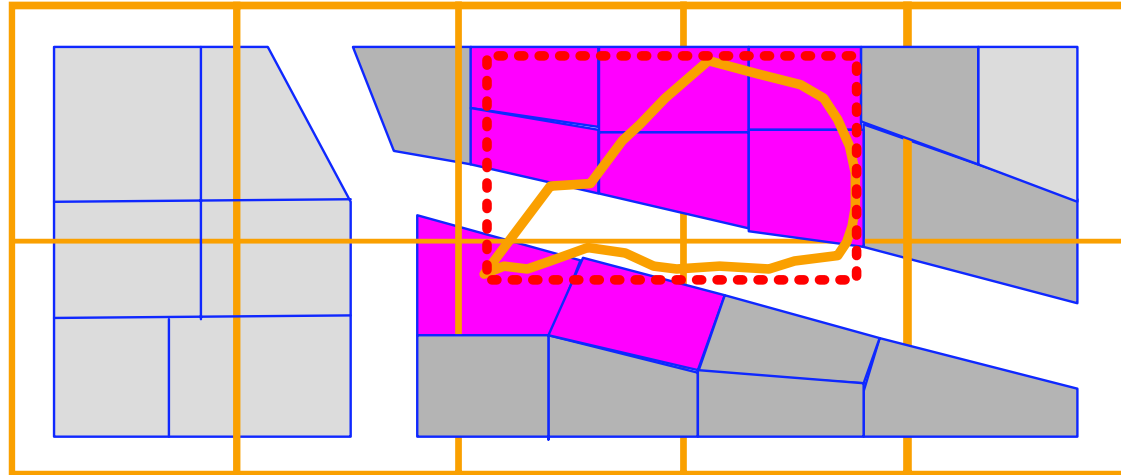
# Let's walk through an example of Spatial Index Filtering



## 1. Eliminate features by **grid of search space**

Eliminate all parcels that do not intersect with the 4 grids of the flood zone.

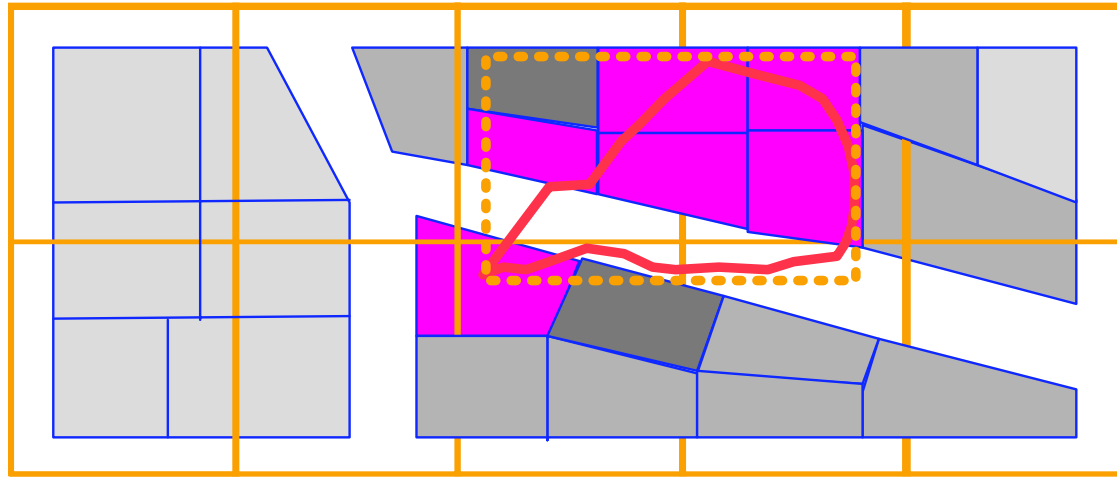
# Let's walk through an example of Spatial Index Filtering



1. Eliminate features by **grid of search space**
2. Eliminate features by **envelope of search space**

Eliminate all parcels that do not intersect the envelope of the flood zone.

# Let's walk through an example of Spatial Index Filtering



1. Eliminate features by **grid of search space**
2. Eliminate features by **envelope of search space**
3. **Eliminate features by coordinate comparison**

Eliminate all parcels that do not actually intersect with the flood zone.

# Spatial Index Grid Levels

## Up to 3 levels of grids

- Most layers have a single spatial index grid
  - Each grid requires a separate index search
  - Use multiple grid levels when features are vastly different in size - avoids needing lots of grid cells to cover a large feature
  - Maximum of 1000 grid cells/feature
- When multilevel grid is used
  - feature promoted to the next grid level when the feature covers more than four grid cells
  - Each level of grid must be at least 3\* larger than the prior level
- Use SQL statement to see number of grid cells per feature
- Avoid high number of grid cells/feature while tuning the grid cell size to approximate the average query window

# Grid Size

- The grid size is the size of the cells in the spatial index
- Up to 3 levels of grids can be provided
  - ▶ must be a minimum of 3\* the size of the preceding grid level
- During initial data loading you can use a large grid size and fine tune the spatial index later
  - does not apply to layers that are defined as a view
- If the grid size is too small, the spatial index will
  - be huge
  - take a long time to create
  - or fail to create
- Use the Spatial Index Advisor (gseidx in V8 only) to determine optimum grid level sizes

# Summary

- Your existing database can be easily spatially enabled
- A spatially aware database knows about location, provides types and functions related to location, and has integrated these capability seamlessly into the database.
- Exploiting location in your data is a natural process that allows you to get more value from your existing data, as well as take advantage of additional data.
- The world leaders in database and GIS technology - IBM and ESRI have partnered to provide you this capability.
- It is available now and greatly enhanced in V8:
  - ▶ Spatial Index Advisor
  - ▶ Enhanced Utilities
  - ▶ New administrative Stored Procedures (db2se replaces gseadm)