Chapter 7

Spatial query language

-Niraj K.C.

Table of Content

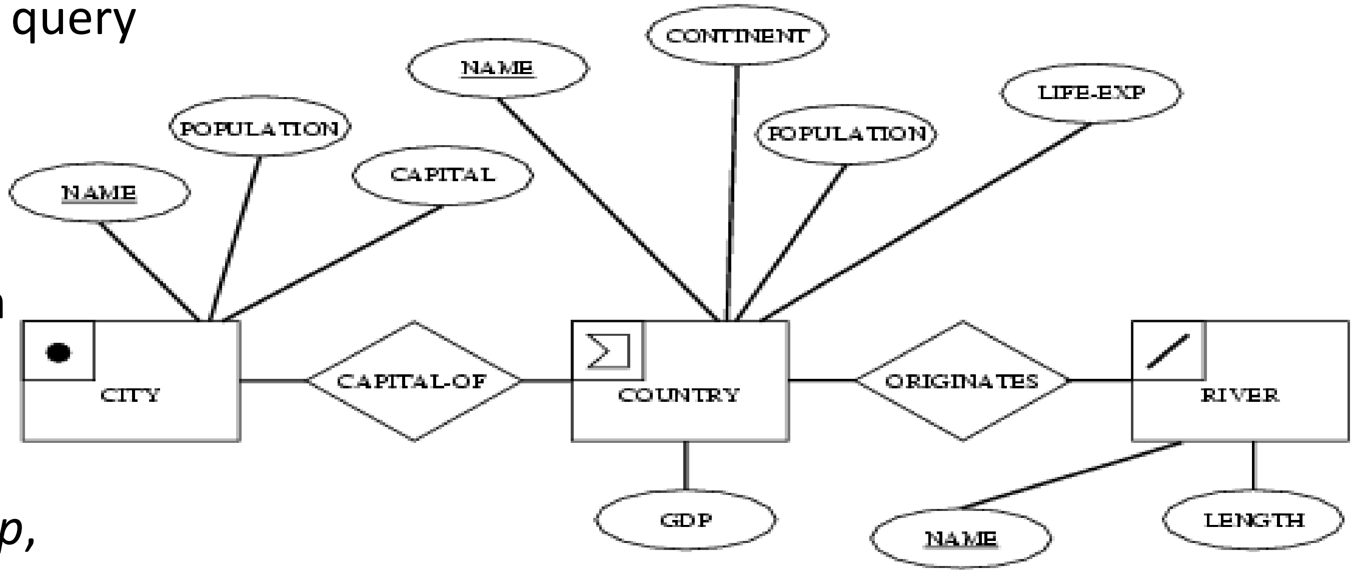
- Spatial Query Language
- SQL for Spatial databases,
- OGIS standard for extending SQL
- Object- relational SQL, object-relationship schema
- Understand concept of a query language
- Learn to use standard query language (SQL)
- Learn to use spatial ADTs with SQL
- Learn about OGIS standard spatial data types and operations
- Learn to use OGIS spatial ADTs with SQL
- Learn about the trends in query languages

What is a query language?

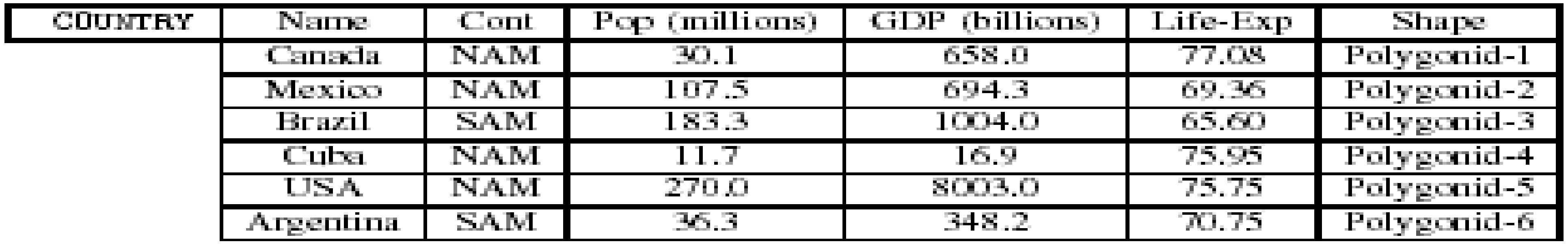
- What is a query language?
  - A language to express interesting questions about data
  - A query language restricts the set of possible queries
- Examples:
  - Natural language, e.g. English, can express almost all queries
  - Computer programming languages, e.g. Java,
    - can express computable queries
    - however algorithms to answer the query is needed
  - Structured Query Language(SQL)
    - Can express common data intensive queries
    - Not suitable for recursive queries
  - Graphical interfaces, e.g. web-search, mouse clicks on a map
    - can express few different kinds of queries

An Example World Database

- Purpose: Use an example database to learn query language SQL
- Conceptual Model
  - 3 Entities: Country, City, River
  - 2 Relationships: capital-of, originates-in
  - Attributes listed in Figure
- 3 Relations
  - **Country(***Name*, *Cont*, *Pop*, *GDP*, *Life-Exp*, *Shape***)**
  - **City(***Name*, *Country*, *Pop*,*Capital*, *Shape***)**
  - **River(***Name*, *Origin*, *Length*, *Shape***)**
- Keys
  - Primary keys are Country.Name, City.Name, River.Name
  - Foreign keys are River.Origin, City.Country
- Data for 3 tables

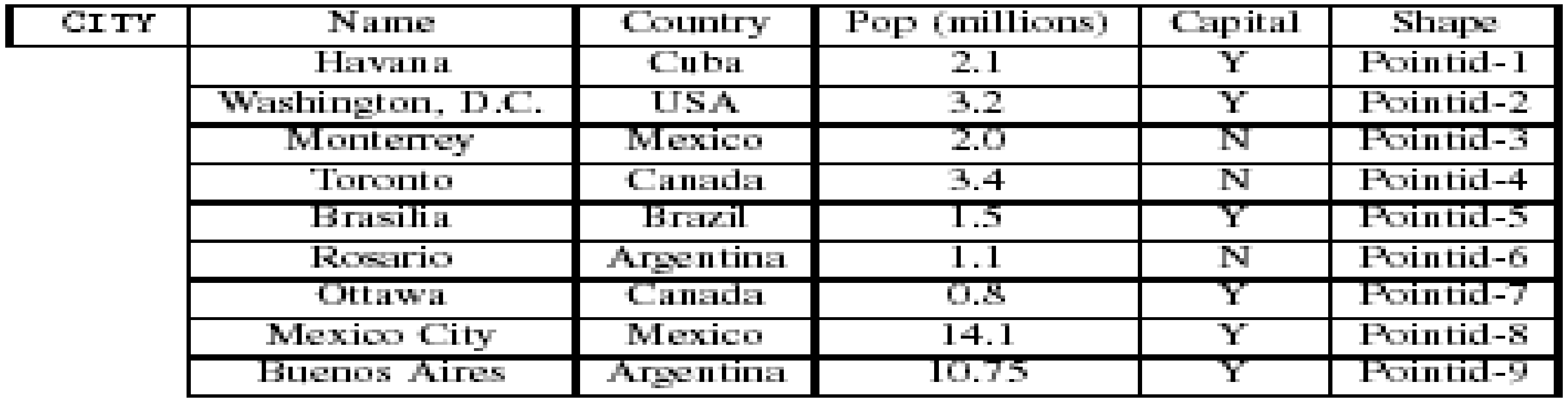


World database data tables



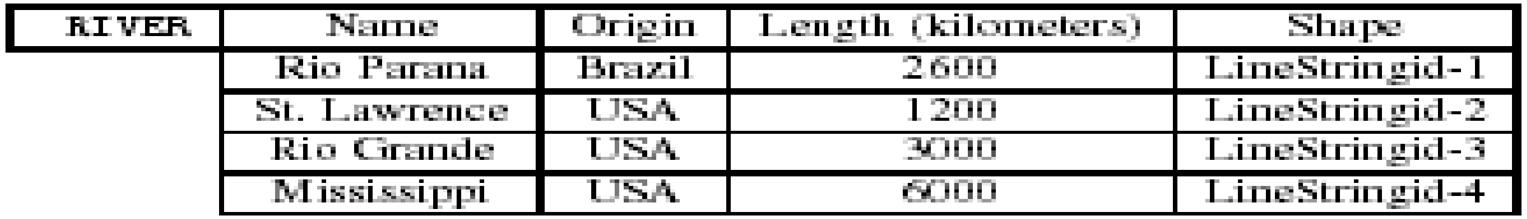
COUNTRY	Name	Cont	Pop (millions)	GDP (billions)	Life-Exp	Shape
	Canada	NAM	30.1	658.0	77.08	Polygonid-1
	Mexico	NAM	107.5	694.3	69.36	Polygonid-2
	Brazil	SAM	183.3	1004.0	65.60	Polygonid-3
	Cuba	NAM	11.7	16.9	75.95	Polygonid-4
	USA	NAM	270.0	8003.0	75.75	Polygonid-5
	Argentina	SAM	36.3	348.2	70.75	Polygonid-6

(a) Country



CITY	Name	Country	Pop (millions)	Capital	Shape
	Havana	Cuba	2.1	Y	Pointid-1
	Washington, D.C.	USA	3.2	Y	Pointid-2
	Monterrey	Mexico	2.0	N	Pointid-3
	Toronto	Canada	3.4	N	Pointid-4
	Brasilia	Brazil	1.5	Y	Pointid-5
	Rosario	Argentina	1.1	N	Pointid-6
	Ottawa	Canada	0.8	Y	Pointid-7
	Mexico City	Mexico	14.1	Y	Pointid-8
	Buenos Aires	Argentina	10.75	Y	Pointid-9

(b) City



RIVER	Name	Origin	Length (kilometers)	Shape
	Rio Parana	Brazil	2600	LineStringid-1
	St. Lawrence	USA	1200	LineStringid-2
	Rio Grande	USA	3000	LineStringid-3
	Mississippi	USA	6000	LineStringid-4

(c) River

What is SQL?

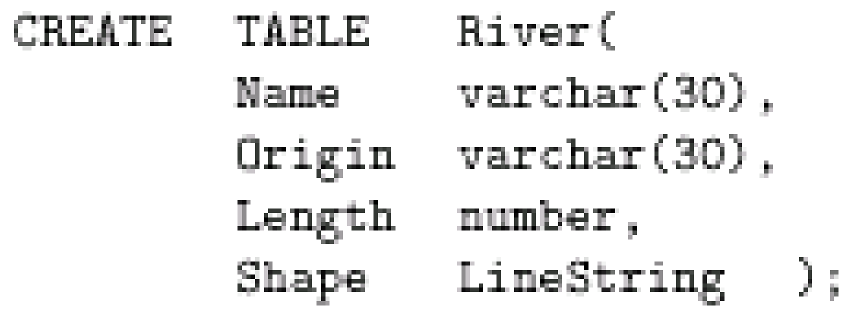
- SQL - General Information
  - is a standard query language for relational databases
  - It support logical data model concepts, such as relations, keys, ...
  - Supported by major brands, e.g. IBM DB2, Oracle, MS SQL Server, Sybase, ...
  - 3 versions: SQL1 (1986), SQL2 (1992), SQL 3 (1999)
  - Can express common data intensive queries
  - SQL 1 and SQL 2 are not suitable for recursive queries
- SQL and spatial data management
  - ESRI Arc/Info included a custom relational DBMS named Info
  - Other GIS software can interact with DBMS using SQL i.e POSTGRES
    - using open database connectivity (ODBC) or other protocols
  - In fact, many software use SQL to manage data in back-end DBMS
  - And a vast majority of SQL queries are generated by other software
  - Although we will be writing SQL queries manually!

Three Components of SQL?

- Data Definition Language (DDL)
  - Creation and modification of relational schema
  - Schema objects include relations, indexes, etc.
- Data Manipulation Language (DML)
  - Insert, delete, update rows in tables
  - Query data in tables
- Data Control Language (DCL)
  - Concurrency control, transactions
  - Administrative tasks, e.g. set up database users, security permissions
- Focus for now
  - A little bit of table creation (DDL) and population (DML)
  - Primarily Querying (DML)

Creating Tables in SQL

- Table definition
  - "CREATE TABLE" statement
  - Specifies table name, attribute names and data types
  - Create a table with no rows.
  - See an example at the bottom
- Related statements
  - ALTER TABLE statement modifies table schema if needed
  - DROP TABLE statement removes an empty table



```
CREATE  TABLE   River(
        Name    varchar(30),
        Origin  varchar(30),
        Length  number,
        Shape   LineString  );
```

Populating Tables in SQL

- Adding a row to an existing table
  - "INSERT INTO" statement
  - Specifies table name, attribute names and values
  - Example:
  - INSERT INTO River(Name, Origin, Length) VALUES('Mississippi', 'USA', 6000)

- Related statements
  - SELECT statement with INTO clause can insert multiple rows in a table
  - Bulk load, import commands also add multiple rows
  - DELETE statement removes rows
  - UPDATE statement can change values within selected rows

Querying populated Tables in SQL

- SELECT statement
  - The commonly used statement to query data in one or more tables
  - Returns a relation (table) as result
  - Has many clauses
  - Can refer to many operators and functions
  - Allows nested queries which can be hard to understand
- Read and write simple SELECT statement
  - Understand frequently used clauses, e.g. SELECT, FROM, WHERE
  - Understand a few operators and function

SELECT Statement- General Information

- Clauses
  - SELECT specifies desired columns
  - FROM specifies relevant tables
  - WHERE specifies qualifying conditions for rows
  - ORDER BY specifies sorting columns for results
  - GROUP BY, HAVING specifies aggregation and statistics
- Operators and functions
  - arithmetic operators, e.g. +, -, …
  - comparison operators, e.g. =, <, >, BETWEEN, LIKE…
  - logical operators, e.g. AND, OR, NOT, EXISTS,
  - set operators, e.g. UNION, IN, ALL, ANY, …
  - statistical functions, e.g. SUM, COUNT, ...
  - many other operators on strings, date, currency, ...

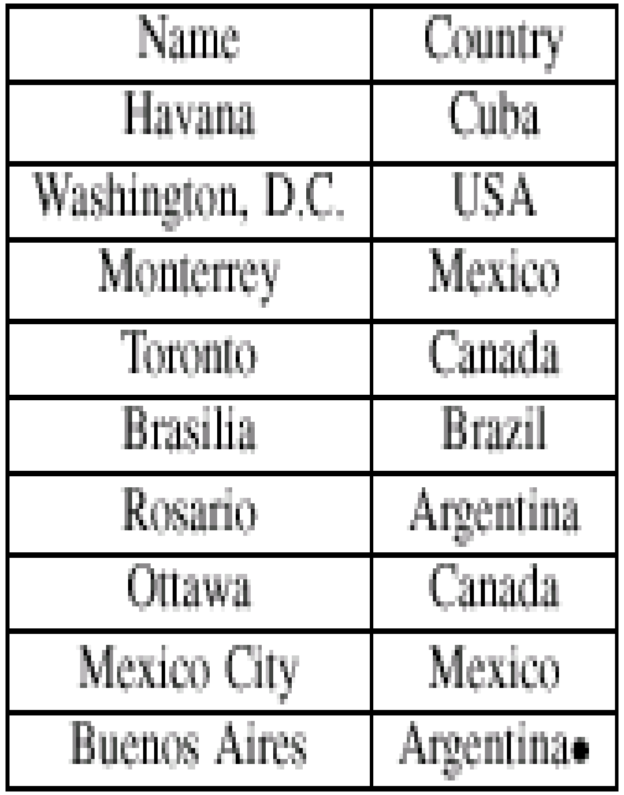
SELECT Example 1.

- Simplest Query has SELECT and FROM clauses
  - Query: List all the cities and the country they belong to.

SELECT Name, Country

FROM CITY

Result →



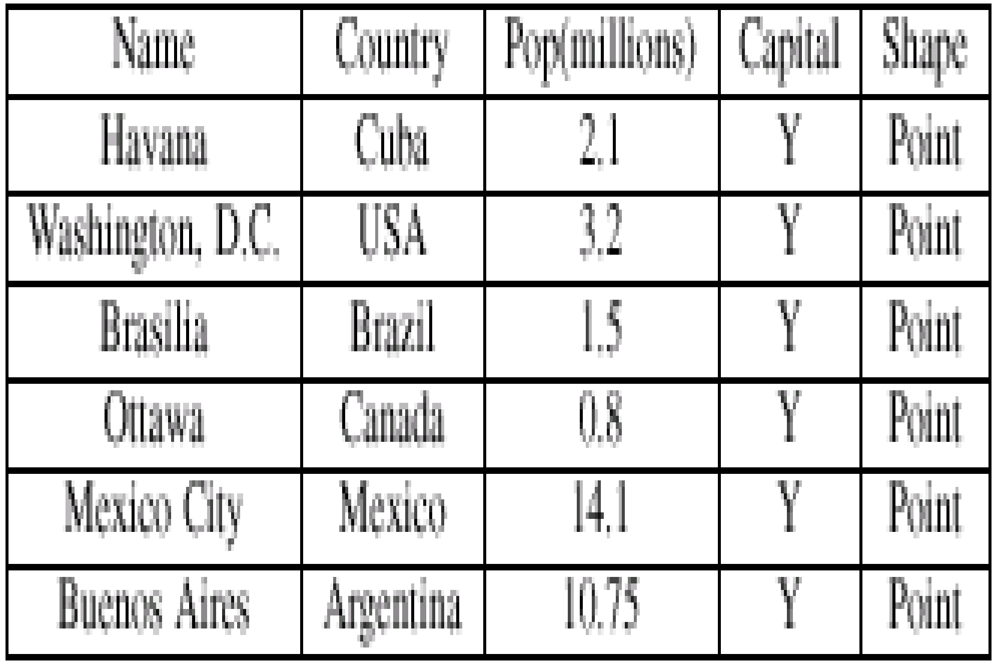
Name	Country
Havana	Cuba
Washington, D.C.	USA
Monterrey	Mexico
Toronto	Canada
Brasilia	Brazil
Rosario	Argentina
Ottawa	Canada
Mexico City	Mexico
Buenos Aires	Argentina•

SELECT Example 2.

- Commonly 3 clauses (SELECT, FROM, WHERE) are used
  - **Query:** List the names of the capital cities in the `CITY` table.

```
SELECT *
FROM CITY
WHERE CAPITAL=‘Y ’
```

Result →



Name	Country	Pop(millions)	Capital	Shape
Havana	Cuba	2.1	Y	Point
Washington, D.C.	USA	3.2	Y	Point
Brasilia	Brazil	1.5	Y	Point
Ottawa	Canada	0.8	Y	Point
Mexico City	Mexico	14.1	Y	Point
Buenos Aires	Argentina	10.75	Y	Point

Query Example…Where clause

Query: List the attributes of countries in the Country relation where the life-expectancy is less than seventy years.

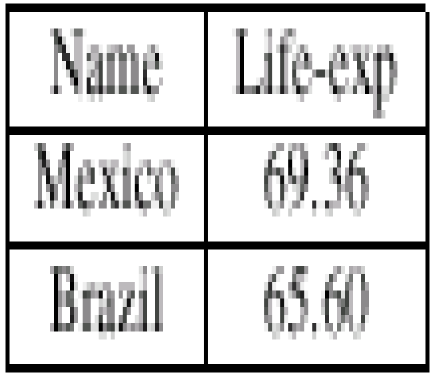
SELECT Co.Name,Co.Life-Exp

FROM Country Co

WHERE Co.Life-Exp <70

Note: use of alias ‘Co’ for Table ‘Country’

Result →



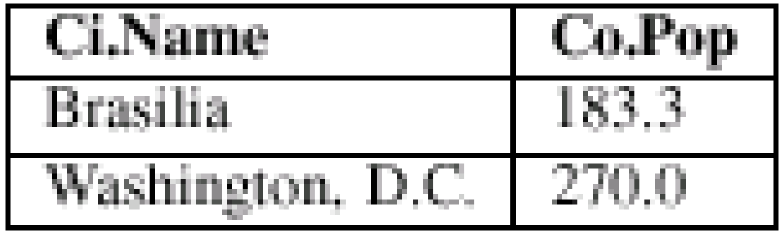
Name	Life-exp
Mexico	69.36
Brazil	65.60

Multi-table Query Examples

Query: List the capital cities and populations of countries whose GDP exceeds one trillion dollars.

Note:Tables City and Country are joined by matching “City.Country = Country.Name”. This simulates relational operator “join” discussed in 3.2

```
SELECT Ci.Name,Co.Pop
FROM City Ci,Country Co
WHERE Ci.Country =Co.Name
AND Co.GDP >1000.0
AND Ci.Capital=‘Y ’
```



Ci.Name	Co.Pop
Brasilia	183.3
Washington, D.C.	270.0

Multi-table Query Example

Query: What is the name and population of the capital city in the country where the St. Lawrence River originates?

SELECT Ci.Name, Ci.Pop
FROM City Ci, Country Co, River R
WHERE R.Origin =Co.Name
AND Co.Name =Ci.Country
AND R.Name =‘St.Lawrence ’
AND Ci.Capital=‘Y ’

Note: Three tables are joined together pair at a time. River.Origin is matched with Country.Name and City.Country is matched with Country.Name. The order of join is decided by query optimizer and does not affect the result.

Query Examples…Aggregate Staistics

Query: What is the average population of the noncapital cities listed in the City table?

```
SELECT AVG(Ci.Pop)
FROM City Ci
WHERE Ci.Capital=‘N ’
```

Query: For each continent, find the average GDP.

```
SELECT Co.Cont,Avg(Co.GDP)AS Continent-GDP
FROM Country Co
GROUP BY Co.Cont
```

Query Example..Having clause, Nested queries

Query: For each country in which at least two rivers originate, find the length of the smallest river.

```
SELECT R.Origin, MIN (R.length) AS Min-length
FROM River
GROUP BY R.Origin
HAVING  COUNT  (*) > 1
```

Query: List the countries whose GDP is greater than that of Canada.

```
SELECT Co.Name
FROM Country Co
WHERE Co.GDP > ANY  (SELECT Co1.GDP
                     FROM Country Co1
                     WHERE Co1.Name =‘Canada ’)
```

Extending SQL for Spatial Data

- ➢SQL has simple atomic data-types, like integer, dates and string
- ➢Not convenient for spatial data and queries
    - Spatial data (e.g. polygons) is complex
    - Spatial operation: topological, euclidean, directional, metric

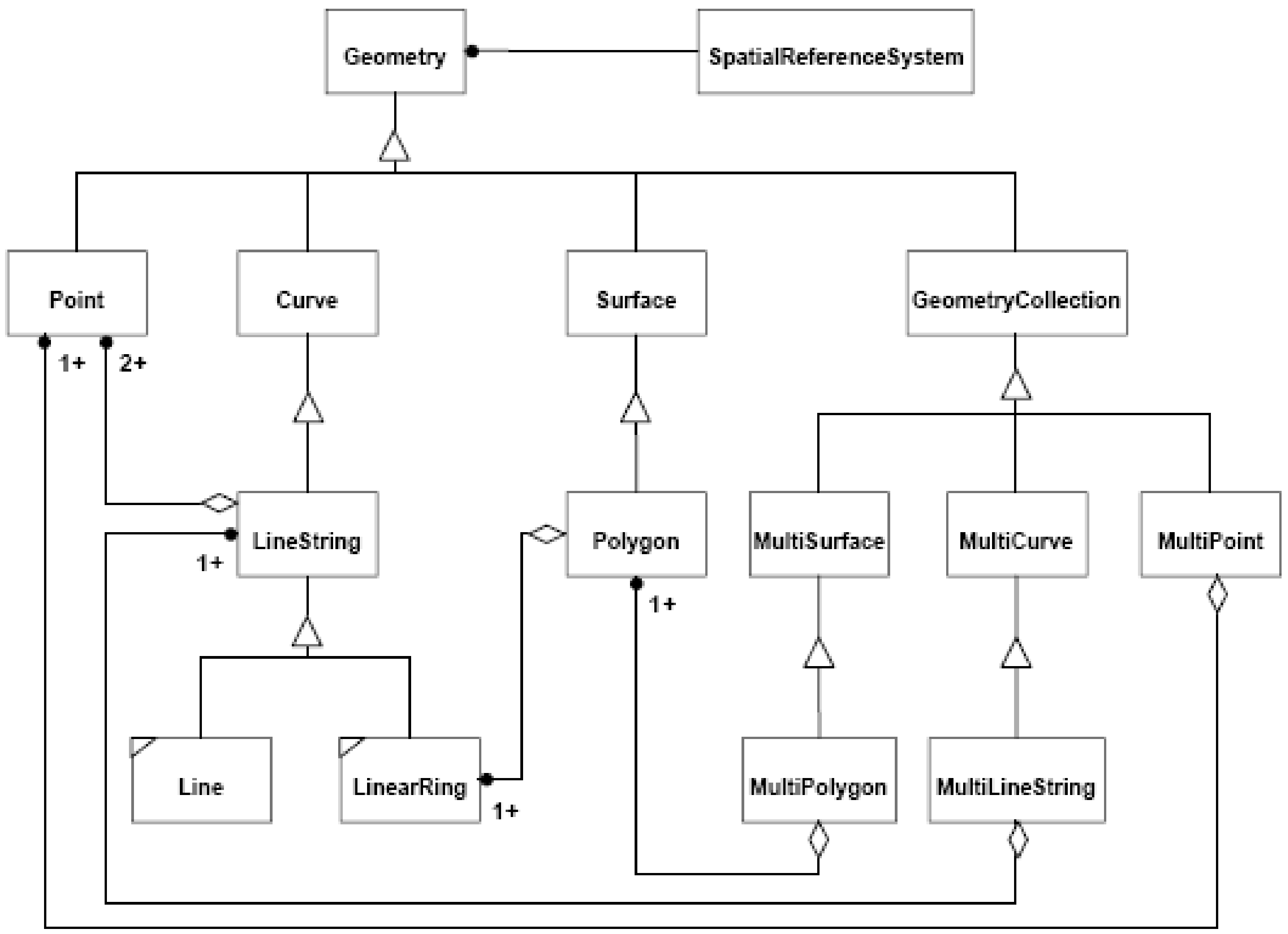
- ➢SQL 3 allows user defined data types and operations
    - Spatial data types and operations can be added to SQL3

- ➢OGC Standard
    - Half a dozen spatial data types
    - Several spatial operations
    - Supported by major vendors, e.g. ESRI, Intergraph, Oracle, IBM,...

OGIS Spatial Data Model

- Consists of base-class `Geometry` and four sub-classes:
  - `Point, Curve, Surface` and `GeometryCollection`
  - Figure lists the spatial data types in OGIS

OpenGIS Simple Features Specification for SQL, Revision1.1



OGIS Spatial Data Model

- Consists of base-class `Geometry` and four sub-classes:
  - `Point, Curve, Surface` and `GeometryCollection`
  - Figure lists the spatial data types in OGIS

- Operations fall into three categories:
  - Apply to all geometry types
    - SpatialReference, Envelope, Export, IsSimple, Boundary
  - Predicates for Topological relationships
    - Equal, Disjoint, Intersect, Touch, Cross, Within, Contains
  - Spatial Data Analysis
    - Distance,Buffer,Union, Intersection, ConvexHull, SymDiff

Spatial Queries with SQL/OGIS

- SQL/OGIS - General Information
  - Both standard are being adopted by many vendors
  - The choice of spatial data types and operations is similar
  - Syntax differs from vendor to vendor
  - Readers may need to alter SQL/OGIS queries given in text to make them run on specific commercial products
- Using OGIS with SQL
  - Spatial data types can be used in DML to type columns
  - Spatial operations can be used in DML

Spatial Operations with SQL/OGIS

- **Basic Functions**
  - SpatialReference(): Returns the underlying coordinate system.
  - Envolope(): Returns the Minimum Bounding Rectangle (MBR).
  - Export(): Returns the geometry in a different representation
  - IsEmpty(): Returns true if the geometry is the null set.
  - IsSimple(): true if no self intersections
  - Boundary(): returns the boundary of the geometry.

Spatial Operations with SQL/OGIS

➢Topological and Set comparison Operations

- Equal: True if the interior and boundary of two geometries are equal.
- Disjoint: True if the boundaries and interiors do not intersect
- Intersection: true if geometries are not disjoint
- Touch: true if the boundaries of two surfaces intersect but their interiors do not
- Cross: true if the interior of a surface intersects with a curve.
- Within: returns true if the interior of a given geometry does not intersect with the exterior of another geometry
- Contains: tests weather the given geometry contains another given geometry.
- Overlap: returns true if the interiors of two geometries have non-empty intersections.

Spatial Operations with SQL/OGIS

- **Spatial Analysis**
  - Distance: return the shortest distance between to geometries
  - Buffer: returns zone around some geometries
  - ConvexHull: returns the smallest convex geometric set enclosing a geometry.
  - Intersection: Returns the intersection of two geometries
  - Union: Returns the union of two geometries
  - Difference: returns the portion of a geometry that does not intersect with another given geometry.
  - SymmDif: returns the portions of two geometry that do not intersect with each other.

List of Spatial Query Examples

- Simple SQL SELECT_FROM_WHERE examples
  - Spatial analysis operations
    - Unary operator: Area
    - Binary operator: Distance
  - Boolean Topological spatial operations - WHERE clause
    - Touch
    - Cross
  - Using spatial analysis and topological operations
    - Buffer, overlap
- Complex SQL examples
  - Aggreagate SQL queries
  - Nested queries

Using spatial operation in SELECT clause

Query: List the name, population, and area of each country listed in the Country table.

SELECT C.Name,C.Pop, Area(C.Shape)AS "Area"

FROM Country C

Note: This query uses spatial operation, Area().Note the use of spatial operation in place of a column in SELECT clause.

Using spatial operator Distance

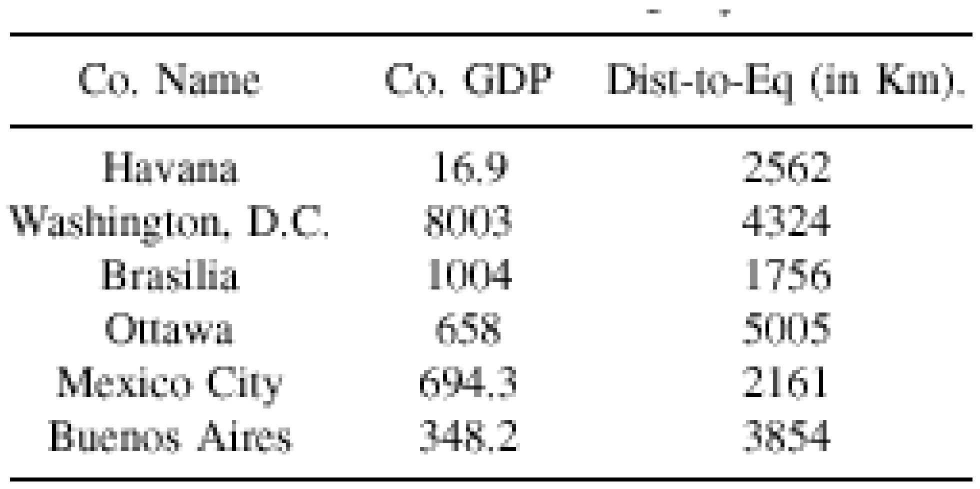
Query: List the GDP and the distance of a country's capital city to the equator for all countries.

SELECT Co.GDP, Distance(Point(0,Ci.Shape.y),Ci.Shape)
AS "Dist-to-Eq"

FROM Country Co,City Ci

WHERE Co.Name = Ci.Country

AND Ci.Capital ='Y '



Co. Name	Co. GDP	Dist-to-Eq (in Km).
Havana	16.9	2562
Washington, D.C.	8003	4324
Brasilia	1004	1756
Ottawa	658	5005
Mexico City	694.3	2161
Buenos Aires	348.2	3854

Using spatial operator Distance

Query: List the GDP and the distance of a country's capital city to the equator for all countries.

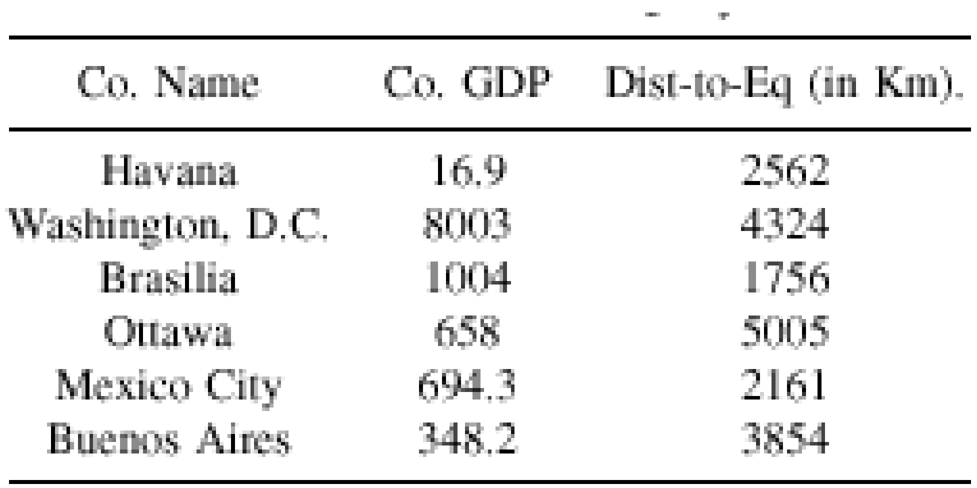
```
SELECT Co.GDP, Distance(Point(0,Ci.Shape.y),Ci.Shape) AS "Distance"
FROM Country Co,City Ci
WHERE Co.Name = Ci.Country
AND Ci.Capital ='Y '
```

Note:

Point(0,Ci.Shape.y)

means lat = 0,

long=C1.Shape.y



Co. Name	Co. GDP	Dist-to-Eq (in Km).
Havana	16.9	2562
Washington, D.C.	8003	4324
Brasilia	1004	1756
Ottawa	658	5005
Mexico City	694.3	2161
Buenos Aires	348.2	3854

Using Spatial Operation in WHERE clause

Query: Find the names of all countries which are neighbors of the United States (USA) in the Country table.

SELECT C1.Name AS "Neighbors of USA"
FROM Country C1,Country C2
WHERE Touch(C1.Shape,C2.Shape)=1
AND C2.Name =‘USA ’

Note: Spatial operator Touch() is used in WHERE clause to join Country table with itself. This query is an example of spatial self join operation.

Spatial Query with multiple tables

Query: For all the rivers listed in the River table, find the countries through which they pass.

SELECT R.Name, C.Name

FROM River R, Country C

WHERE Cross(R.Shape,C.Shape)=1

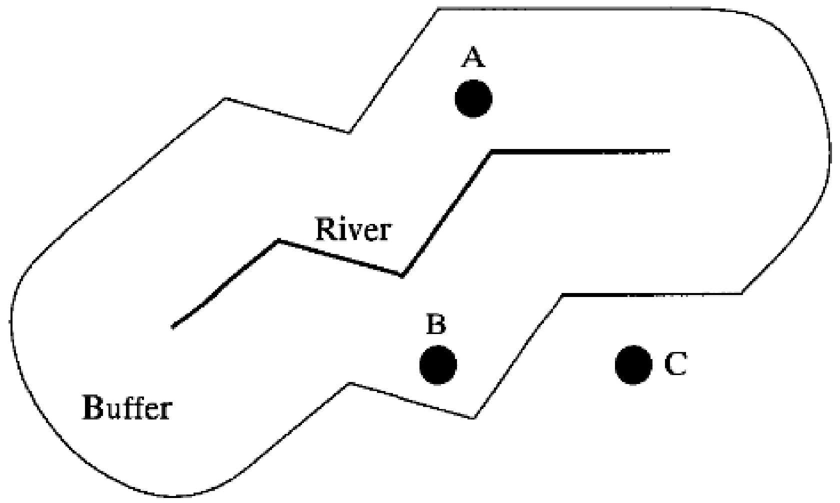
Note: Spatial operation “Cross” is used to join River and Country tables. This query represents a spatial join operation.

Example Spatial Query…Buffer and Overlap

Query: The St. Lawrence River can supply water to cities that are within 300 km. List the cities that can use water from the St. Lawrence.

```
SELECT Ci.Name
FROM City Ci, River R
WHERE  Overlap(Ci.Shape, Buffer(R.Shape,300))=1
AND R.Name ='St.Lawrence '
```

Note: This query uses spatial operation of Buffer, which is illustrated in Figure below



The buffer of a river and points within and outside.

Using spatial operation in an aggregate query

Query: List all countries, ordered by number of neighboring countries.

SELECT Co.Name, Count(Co1.Name)

FROM Country Co, Country Co1

WHERE Touch(Co.Shape,Co1.Shape)

GROUP BY Co.Name

ORDER BY Count(Co1.Name)

Notes: This query can be used to differentiate querying capabilities of simple GIS software (e.g. Arc/View) and a spatial database. It is quite tedious to carry out this query in GIS.

Earlier version of OGIS did not provide spatial aggregate operation to support GIS operations like reclassify.

Using Spatial Operation in Nested Queries

Query: For each river, identify the closest city.

```
SELECT C1.Name, R1.Name
FROM City C1, River R1
WHERE Distance (C1.Shape,R1.Shape) <= ALL (SELECT
Distance(C2.Shape)
                                           FROM City C2
                                           WHERE C1.Name <> C2.Name
                                           )
```

Note: Spatial operation Distance used in context of a nested query.

Exercise: It is interesting to note that SQL query expression to find smallest distance from each river to nearest city is much simpler and does not require nested query. Audience is encouraged to write a SQL expression for this query.

Nested Spatial Query

Query: List the countries with only one neighboring country. A country is a neighbor of another country if their land masses share a boundary. According to this definition, island countries, like Iceland, have no neighbors.

```
SELECT Co.Name
FROM Country Co
WHERE Co.Name IN (SELECT Co.Name
                  FROM Country Co,Country Co1
                  WHERE Touch(Co.Shape,Co1.Shape)
                  GROUP BY Co.Name
                  HAVING Count(*)=1)
```

Note: It shows a complex nested query with aggregate operations. Such queries can be written into two expression, namely a view definition, and a query on the view. The inner query becomes a view and outer query is run on the view. This is illustrated in the next slide.

Rewriting nested queries using Views

- Views are like tables
  - Represent derived data or result of a query
  - Can be used to simplify complex nested queries
  - Example follows:

CREATE VIEW Neighbor AS

SELECT Co.Name, Count(Co1.Name)AS num neighbors

FROM Country Co,Country Co1

WHERE Touch(Co.Shape,Co1.Shape)

GROUP BY Co.Name

```
SELECT Co.Name,num neighbors
FROM Neighbor
WHERE num neighbor = ( SELECT Max(num neighbors)
                       FROM Neighbor )
```

Defining Spatial Data Types in SQL3

- SQL3 User defined data type – Overview
  - CREATE TYPE statements
  - Defines a new data types
  - Attributes and methods are defined
  - Separate statements for interface and implementation

- Additional effort is needed at physical data model level

Defining Spatial Data Types in SQL3

- Libraries, Data cartridge/blades
  - Third party libraries implementing OGIS are available
  - Almost all user use these libraries
  - Few users need to define their own data types

Defining Spatial Data Types in SQL3

- SQL3/SQL99 User defined data type: ADT and Row Type
- ADT: Define ADT using a Create Type statement, can appear as column type

```
Create Type Point (
        X Number,
        Y Number,
        Function Distance(:u Point, :v Point)
                Returns Number );
```

- Row Type: A row type specifies the schema of a relation.

```
Create Row Type Point (
    x Number,
    y Number);
```

Table that instantiates the row type:

```
Create Table Point table of TYPE Point;
```

Object-Relational Schema

- OR-DBMS (Oracle 8): implements a part of the SQL3 standard, the ADT is called the object type.

Example: Point

•Create Type Point as Object (
x Number,
y Number,
Member FUNCTION Distance(P2 in Point) Return Number,
PRAGM RESTRICT_REFERENCES(Distance, WNDS));

PRAGMA indicates that the Distance function will not modify the state of the database: WNDS(Write No Database State)

Object-Relational Schema

- Example: LineString

- Create TYPE LineType as VARRAY(500) of Point;
- Create Type LineString as Object (

```
    Num_of_Points INT,
    Geometry LineType,
    Member FUNCTION Length(SELF IN) Return Number,
    PRAGM RESTRICT_REFERENCES(Length, WNDS));
```

Object-Relational Schema

- Example: Polygon

- Create TYPE PolyType as VARRAY(500) of Point;
- Create Type LineString as Object (

```
Num_of_Points INT,
Geometry PolyType,
Member FUNCTION Length(SELF IN) Return Number,
PRAGM RESTRICT_REFERENCES(Length, WNDS));
```

Example Queries

- List the names, populations, and areas of all countries adjacent to the USA.

```
Select C2.Name, C2.Pop, C2.Area() as “AREA”
From Country C1, Country C2
Where C1.Name = “USA” AND
      C1.Touch(C2. Shape) = 1
```

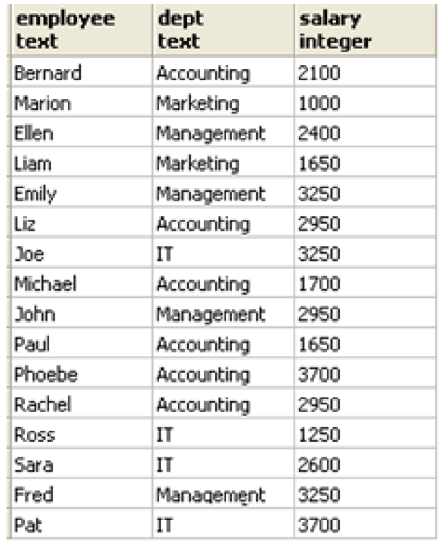
- List all the pairs of cities in the City table and the distance between them

```
Select C1.Name, C1.Distance(C2. Shape) As “Distance”
From City C1, City C2
Where C1.Name <> C2. Name
```

Summary

- Queries to databases are posed in high level declarative manner
- SQL is the “lingua-franca” in the commercial database world
- Standard SQL operates on relatively simple data types
- SQL3/OGIS supports several spatial data types and operations
- Additional spatial data types and operations can be defined
  - CREATE TYPE statement

```
-- 1
CREATE table staff  (
   employee text,
   dept text,
   salary int4,
   PRIMARY KEY (employee, dept),
  CONSTRAINT
  positive_salary CHECK (salary > 0));
-- 2
INSERT …
-- 3
SELECT * FROM staff;
```



employee text	dept text	salary integer
Bernard	Accounting	2100
Marion	Marketing	1000
Ellen	Management	2400
Liam	Marketing	1650
Emily	Management	3250
Liz	Accounting	2950
Joe	IT	3250
Michael	Accounting	1700
John	Management	2950
Paul	Accounting	1650
Phoebe	Accounting	3700
Rachel	Accounting	2950
Ross	IT	1250
Sara	IT	2600
Fred	Management	3250
Pat	IT	3700

Table 1

Assignment 6

1. What is Query language? Mention different forms of languages with suitable example.
2. Explain SQL for spatial databases with suitable example.
3. How OGIS standard extends SQL for spatial data?
4. List out and describe basic functions of spatial operations using OGIS.
5. List out steps for defining Spatial data types in SQL3.
6. What do you understand by object relational schema? Describe object relational schema in detail with suitable example of object type.