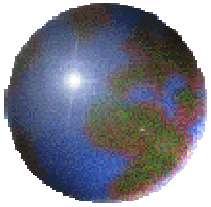
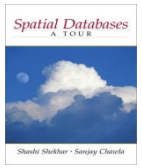
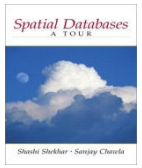


Chapter 3: Spatial Query Languages



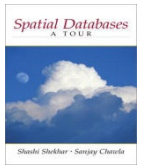
What is a Query?

- What is a Query ?
  - A query is a “question” posed to a database
  - Queries are expressed in a high-level declarative manner
    - algorithms needed to answer the query are not specified in the query
- Examples:
  - Mouse click on a map symbol (e.g. road) may mean
    - what is the name of road pointed to by mouse cursor ?
  - Typing a keyword in a search engine (e.g. google, yahoo) means
    - which documents on web contain given keywords?
  - SELECT S.name FROM Senator S WHERE S.gender = ‘F’ means
    - which senators are female?



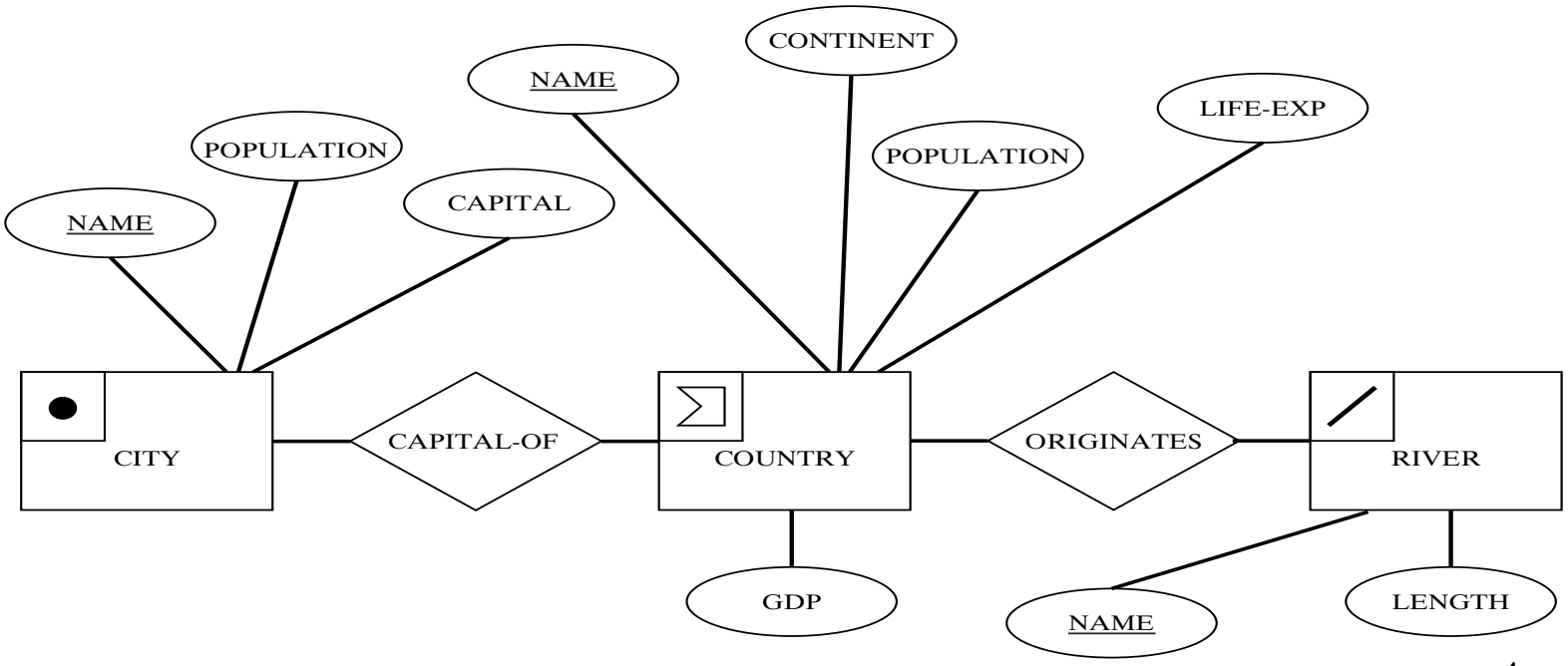
What is a Query Language?

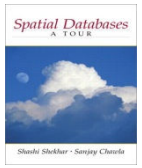
- What is a query language?
  - A language to express interesting questions about data
  - A query language restricts the set of possible queries
- Examples:
  - Natural language, e.g. English, can express almost all queries
  - Computer programming languages, e.g. Java,
    - can express computable 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 3.1

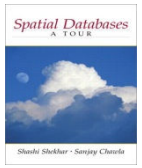




An Example Database - Logical Model

- 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
  - Shown on next slide



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

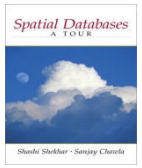
(a) Country

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

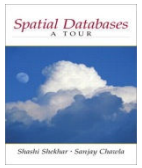
RIVER	Name	Origin	Length (kilometers)	Shape
	Rio Parana	Brazil	2600	LineStringid-1
	St. Lawrence	USA	1200	LineStringid-2
	Rio Grande	USA	3000	LineStringid-3
	Mississippi	USA	6000	LineStringid-4

(c) River



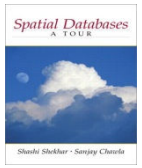
What is SQL?

- 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
    - 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?

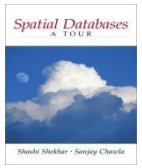
- Data Definition Language (DDL)
  - Creation and modification of relational schema
  - Schema objects include relations, indexes, etc.
- Data Manipulation Language (DML)
  - Insert, delete, update rows in tables
  - Query data in tables
- Data Control Language (DCL)
  - Concurrency control, transactions
  - 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   );
```

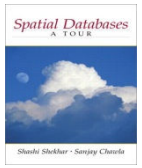


Populating Tables in SQL

- Adding a row to an existing table
  - INSERT INTO statement
  - Specifies table name, attribute names and values
  - Example:

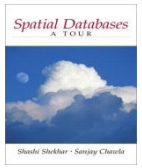
  INSERT INTO River(Name, Origin, Length) VALUES('Mississippi', 'USA', 6000)

- 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



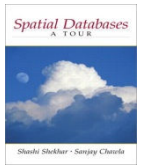
Querying Populated Tables in SQL

- 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
- Scope of our discussion
  - Learn enough SQL to appreciate spatial extensions
    - observe example queries
  - 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, ...



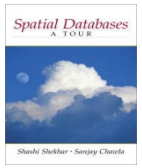
SELECT Example 1.

- 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



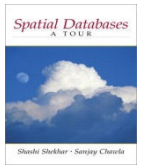
SELECT Example 2.

- 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

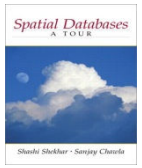
Query: List the attributes of countries in the Country relation where the life-expectancy is less than seventy years.

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



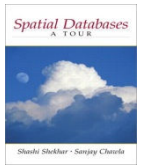
Multi-table Query Examples

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

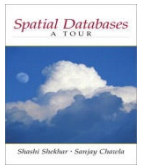


Multi-table Query Example

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.



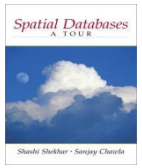
Query Examples…Aggregate Statistics

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



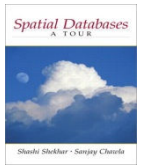
Query Example..Having clause, Nested Queries

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
```

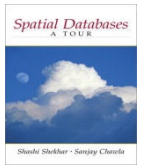
Query: List the countries whose GDP is greater than that of Canada.

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



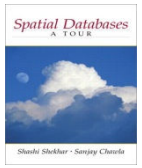
Extending SQL for Spatial Data

- Motivation
  - 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
- Open Geodata Interchange Standard (OGIS)
  - Half a dozen spatial data types
  - Several spatial operations
  - Supported by major vendors, e.g. ESRI, Intergraph, Oracle, IBM,...



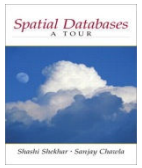
OGIS Spatial Data Model

- Consists of base-class Geometry and four sub-classes:
  - Point, Curve, Surface and GeometryCollection
  - Figure 2.2 (pp. 27) 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
  - Table 3.9 (pp.66) details spatial operations



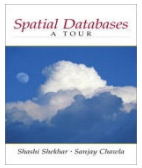
Spatial Queries with SQL/OGIS

- 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
- Scope of discussion
  - Illustrate use of spatial data types with SQL
  - Via a set of examples



List of Spatial Query Examples

- Simple SQL SELECT_FROM_WHERE examples
  - Spatial analysis operations
    - Unary operator: Area (Q5, pp.68)
    - Binary operator: Distance (Q3)
  - Boolean Topological spatial operations - WHERE clause
    - Touch (Q1, pp. 67)
    - Cross (Q2, pp. 68)
  - Using spatial analysis and topological operations
    - Buffer, overlap (Q4)
- Complex SQL examples
  - Aggregate SQL queries
  - Nested queries



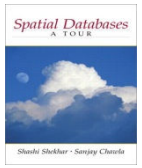
Using Spatial Operation in SELECT Clause

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

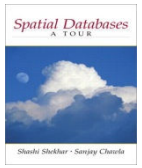
Query: List the GDP and the distance of a country's capital city to the equator for all countries.

SELECT Co.GDP, Distance(Point(0,Ci.Shape.y),Ci.Shape) AS "Distance"

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 Operation in WHERE Clause

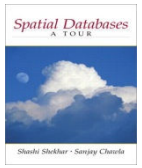
Query: Find the names of all countries which are neighbors of the United States (USA) in the Country table.

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.



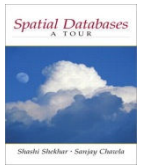
Spatial Query with Multiple Tables

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.

Exercise: Modify above query to report length of river in each country.
Hint: Q6, pp. 69

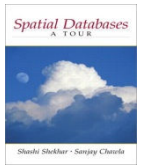


Example Spatial Query…Buffer and Overlap

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.

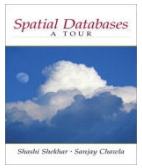
SELECT Ci.Name
FROM City Ci, River R
WHERE Overlap(Ci.Shape, Buffer(R.Shape,300))=1
AND R.Name ='St.Lawrence '

Note: This query uses spatial operation of Buffer, which is illustrated in Figure 3.2 (pp. 69).



Recall 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
  - Aggregate SQL queries (Q9, pp. 70)
  - Nested queries (Q3 pp. 68, Q10, pp. 70)



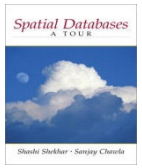
Using Spatial Operation in an Aggregate Query

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



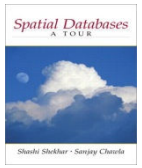
Using Spatial Operation in Nested Queries

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,R1.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.

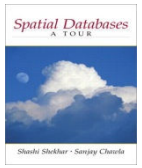


Nested Spatial Query

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)
```

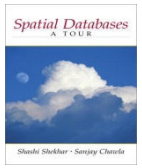
Note: It shows a complex nested query with aggregate operations. Such queries can be written into two expression, namely a view definition, and a query on the view. The inner query becomes a view and outer query is run on the view. This is illustrated in the next slide.



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:

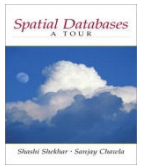
```
CREATE VIEW Neighbor AS
SELECT Co.Name, Count(Co1.Name) AS num.neighbors
FROM Country Co, Country Co1
WHERE Touch(Co.Shape,Co1.Shape)
GROUP BY Co.Name

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
    - examples of interface in Table 3.12 (pp. 74)

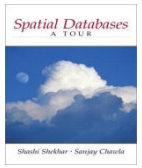
- Additional effort is needed at physical data model level



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

- We will not discuss the detailed syntax of CREATE TYPE
  - Interested readers are encouraged to look at section 3.6



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