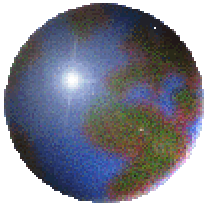


Chapter 3:Spatial Query Languages

3.1 *Standard Database Query Languages*

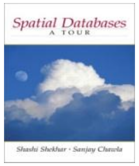
3.2 Relational Algebra

3.3 Basic SQL Primer

3.4 Extending SQL for Spatial Data

3.5 Example Queries that emphasize spatial aspects

3.6 Trends: Object-Relational SQL



Learning Objectives

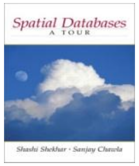
- Learning Objectives (LO)
  - LO1: Understand concept of a query language
    - What is a query language?
    - Why use query languages?
  - LO2 : Learn to use standard query language (SQL)
  - LO3: Learn to use spatial ADTs with SQL
  - LO4: Learn about the trends in query languages

- Mapping Sections to learning objectives

LO2	-	3.2, 3.3
LO3	-	3.4, 3.5
LO4	-	3.6

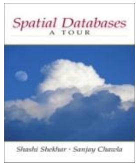
What is a query?

- 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

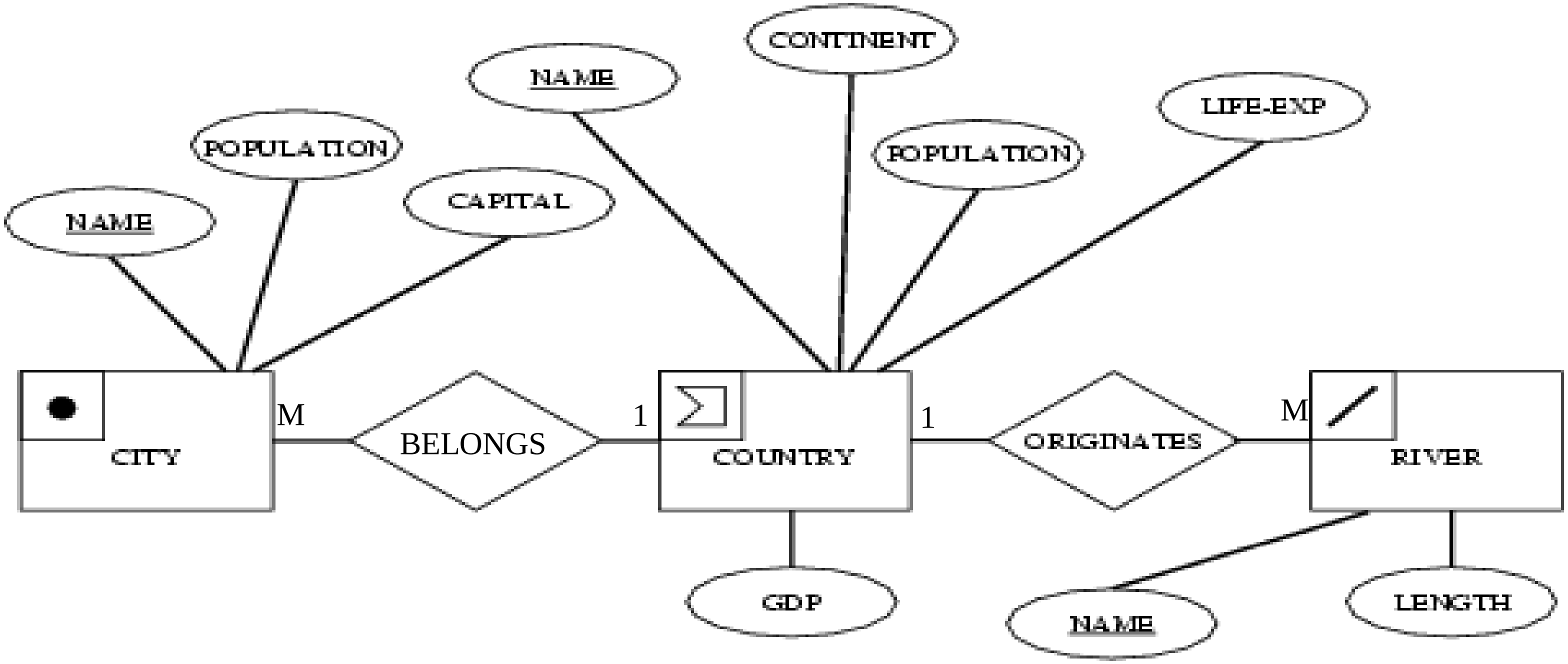


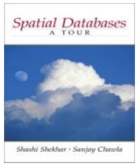
An Example World Database

- Design ER with pictograms for a *World* database:
  - The world has countries
  - Countries have cities
  - A country has a capital city
  - Rivers cross countries
- Since water management is a big issue:
  - Each river belongs to a single country (we say that a river *originates* from a country)

An Example World Database

- The pictogram





An Example Database - Logical Model

- 3 Relations

  Country(*Name*, *Cont*, *Pop*, *GDP*, *Life-Exp*, *Shape***)**

  City(*Name*, *Country*, *Pop*,*Capital*, *Shape***)**

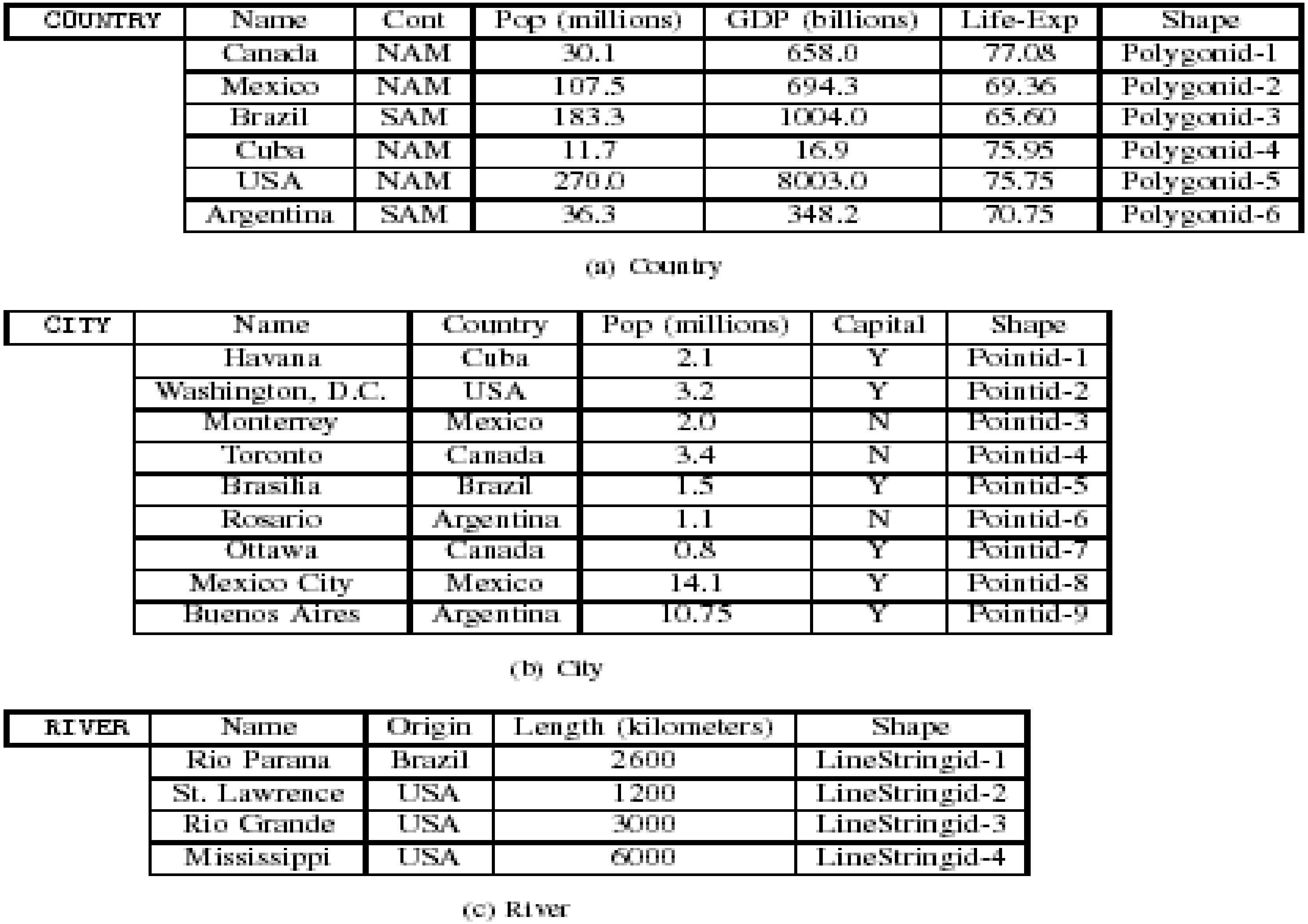
  River(*Name*, *Origin*, *Length*, *Shape***)**

- Keys

  •Primary keys are Country.Name, City.Name, River.Name

  • Foreign keys are River.Origin, City.Country

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

Relational Algebra

- Select-Project
  - Retrieve selected rows - columns of a relations

$$\sigma_{<\text{selection operator}>}(\text{Relation})$$

$$\pi_{\text{Name}}(Country)$$

- Examples:
  - List all data of all countries in North America (NAM)
  - List all names of countries in the Country table
  - List all names of countries in North America

Relational Algebra

- Results

$$\sigma_{\text{cont}=\text{`North-America'}}(Country)$$

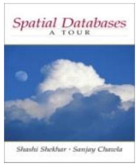
Name	Cont	Pop (millions)	GDP (billions)	Life-Exp	Shape
Canada	NAM	30.1	658.0	77.08	Polygonid-1
Mexico	NAM	107.5	694.3	69.36	Polygonid-2
Cuba	NAM	11.7	16.9	75.95	Polygonid-4
USA	NAM	270.0	8003.0	75.75	Polygonid-5

$$\pi_{\text{Name}}(Country)$$

Name
Canada
Mexico
Brazil
Cuba
USA
Argentina

$$\pi_{\text{Name}}(\sigma_{\text{Cont}=\text{`North-America'}})(Country)$$

Name
Canada
Mexico
Cuba
USA



Relational Algebra

- Set operations (for two Union-compatible relations R, S)
  - Union $R \cup S$
  - Difference $R - S$
  - Intersection $R \cap S = R - (R - S)$
- Examples:
  - List all countries either in N. America or with an originating river
  - List all countries in N. America without an originating river
  - List all countries both in S. America and with an originating river

Relational Algebra

- Results:

1. $R = \pi_{\text{Name}}(\sigma_{\text{Cont}=\text{`North-America'}}(\text{Country}))$
2. $S = \pi_{\text{Origin}}(\text{River})$
3. $R \cup S$.

NAME
Canada
Mexico
Brazil
Cuba
USA

1. $R = \pi_{\text{Name}}(\sigma_{\text{Cont}=\text{`North-America'}}(\text{Country}))$
2. $S = \pi_{\text{Origin}}(\text{River})$
3. $R - S$.

NAME
Canada
Mexico
Cuba

1. $R = \pi_{\text{Name}}(\sigma_{\text{Cont}=\text{`South America'}}(\text{Country}))$
2. $R = \pi_{\text{Origin}}(\text{River})$
3. $R \cap S$.

NAME
Brazil

Relational Algebra

- Join
  - Conditional join $R \bowtie_c S = \sigma_c(R \times S)$
  - Natural join $R \bowtie S$

- Examples:
  - List the names of countries with population larger than Mexico’s
  - List the population of countries with an originating river

Relational Algebra

- Results (conditional join)

1. $R = \pi_{\text{Name, Pop}}(\text{Country})$
2. $S = R$. (S is duplicate copy of R)
3. Form the cross-product $R \times S$. The schema of the $R \times S$ relation is

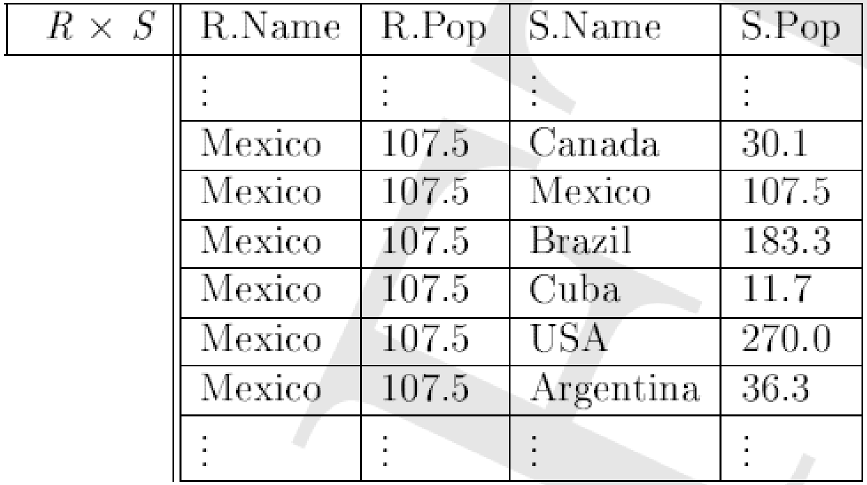
$R \times S$	R.Name	R.Pop	S.Name	S.Pop

4. Apply condition; that is, the population of a country in relation S is greater than the population of Mexico.

$$U = R \bowtie S = \sigma_{(\text{R.Name} = \text{`Mexico'}) \wedge (\text{R.Pop} > \text{S.Pop})}(R \times S)$$

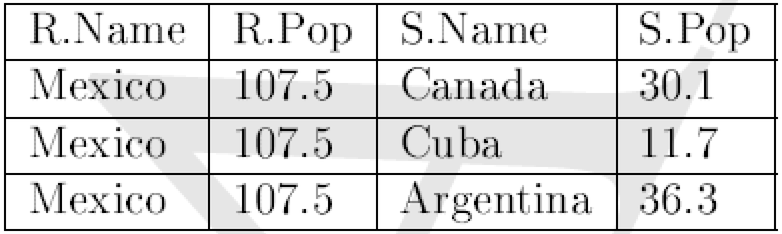
Relational Algebra

- Results (conditional join)



$R \times S$	R.Name	R.Pop	S.Name	S.Pop
	⋮	⋮	⋮	⋮
	Mexico	107.5	Canada	30.1
	Mexico	107.5	Mexico	107.5
	Mexico	107.5	Brazil	183.3
	Mexico	107.5	Cuba	11.7
	Mexico	107.5	USA	270.0
	Mexico	107.5	Argentina	36.3
	⋮	⋮	⋮	⋮

(a) A portion of $R \times S$

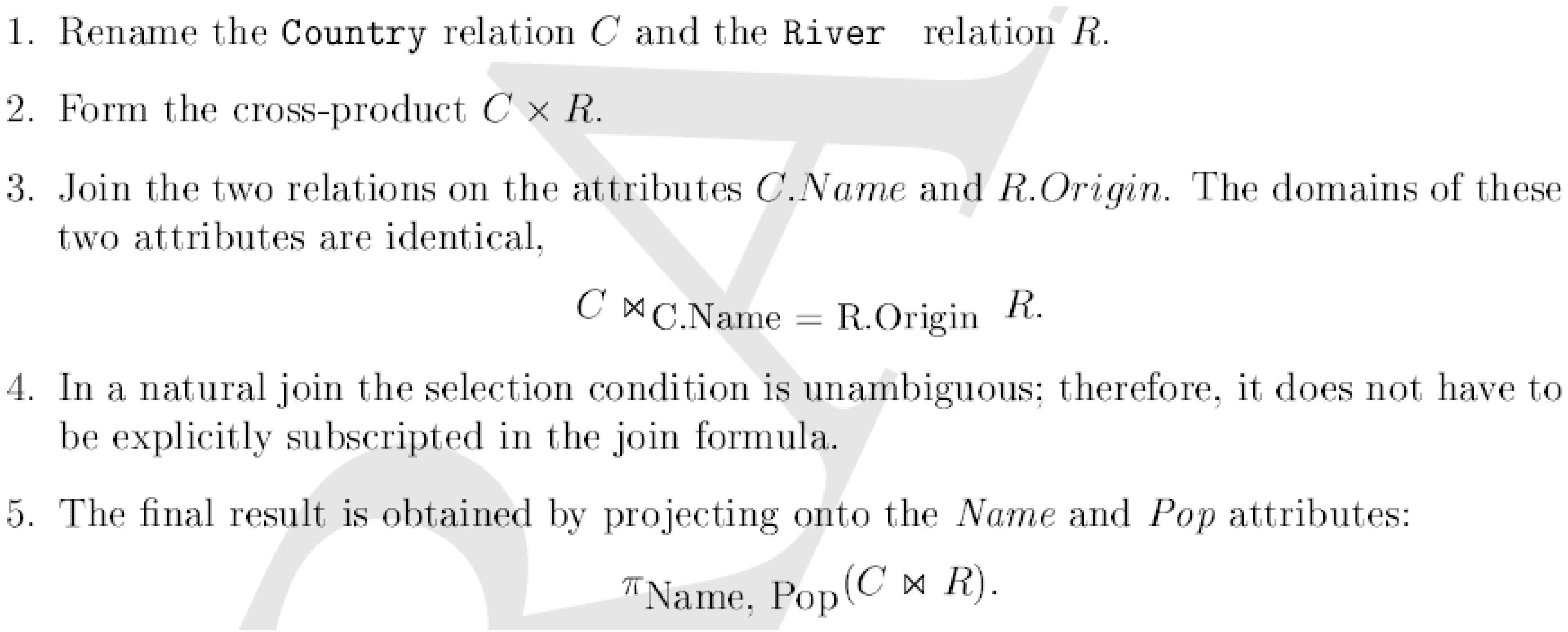


R.Name	R.Pop	S.Name	S.Pop
Mexico	107.5	Canada	30.1
Mexico	107.5	Cuba	11.7
Mexico	107.5	Argentina	36.3

(b) The select operation on $R \times S$

Relational Algebra

- Results (natural join)



1. Rename the `Country` relation C and the `River` relation R.

2. Form the cross-product $C \times R$.

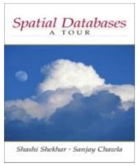
3. Join the two relations on the attributes $C.Name$ and $R.Origin$. The domains of these two attributes are identical,

$$C \bowtie_{\text{C.Name = R.Origin}} R.$$

4. In a natural join the selection condition is unambiguous; therefore, it does not have to be explicitly subscripted in the join formula.

5. The final result is obtained by projecting onto the $Name$ and Pop attributes:

$$\pi_{\text{Name, Pop}}(C \bowtie R).$$

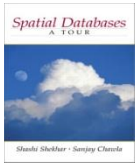


Learning Objectives

- Learning Objectives (LO)
  - LO1: Understand concept of a query language
  - LO2 : Learn to use standard query language (SQL)
    - How to create and populate tables?
    - How to query given tables?
  - LO3: Learn to use spatial ADTs with SQL
  - LO4: Learn about the trends in query languages

- Mapping Sections to learning objectives

LO2	-	3.2, 3.3
LO3	-	3.4, 3.5
LO4	-	3.6



What is SQL?

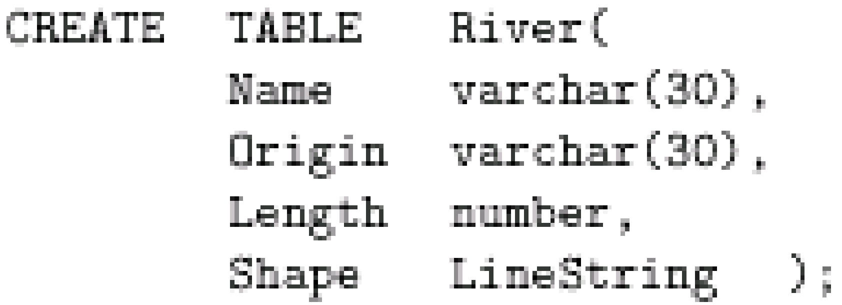
- SQL - General Information
  - is a standard query language for relational databases
  - It support logical data model concepts, such as relations, keys, ...
  - Supported by major brands, e.g. IBM DB2, Oracle, MS SQL Server, Sybase, ...
  - 3 versions: SQL1 (1986), SQL2 (1992), SQL 3 (1999)
  - Can 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, ...

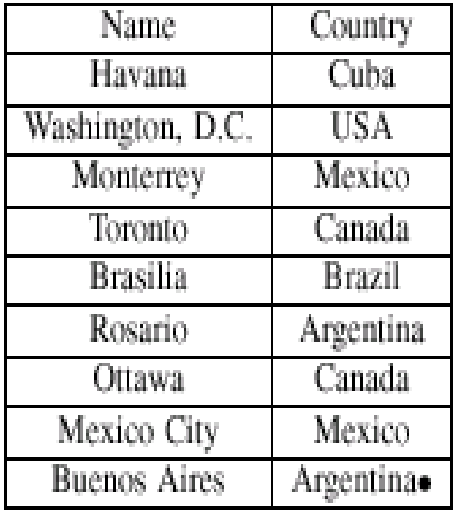
SELECT Example 1.

- Simplest Query has SELECT and FROM clauses
  - Query 1: 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.

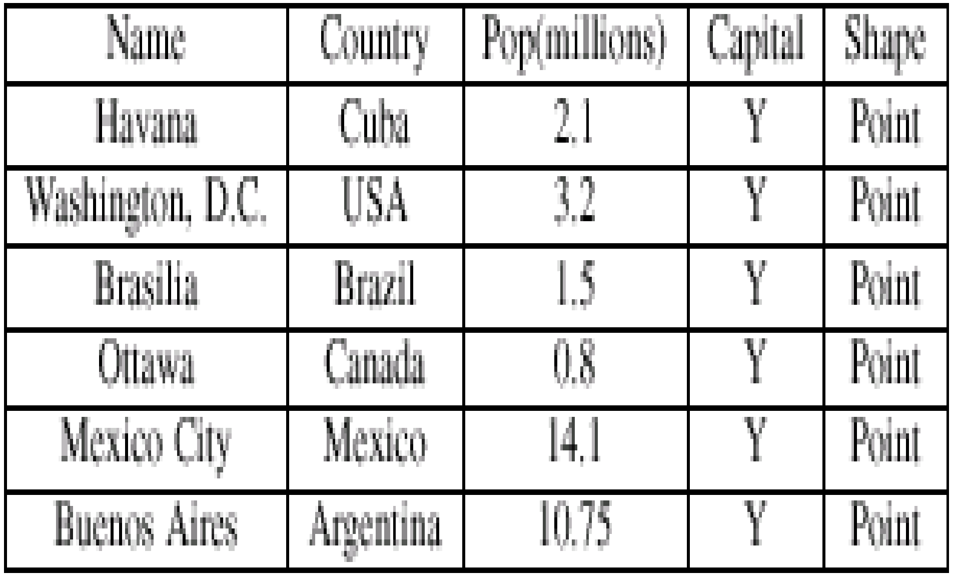
- Commonly 3 clauses (SELECT, FROM, WHERE) are used
  - **Query 2:** List the names of the capital cities in the CITY table.

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

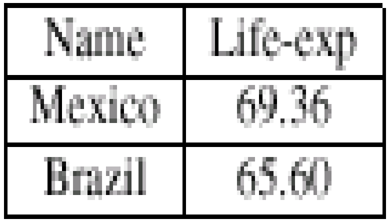
SELECT Co.Name,Co.Life-Exp

FROM Country Co

WHERE Co.Life-Exp <70

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

Result →



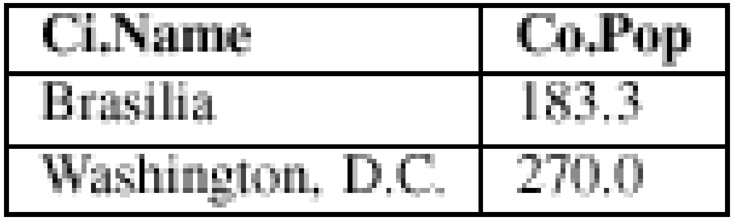
Name	Life-exp
Mexico	69.36
Brazil	65.60

Multi-table Query Examples

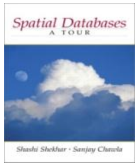
Query 4: List the capital cities and populations of countries whose GDP exceeds one trillion dollars.

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 ’
```



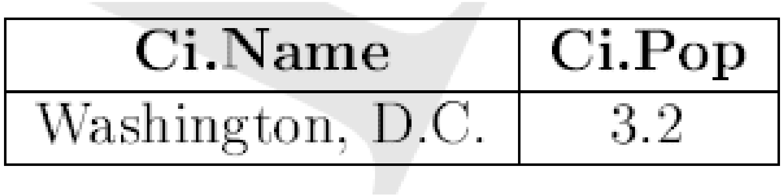
Ci.Name	Co.Pop
Brasilia	183.3
Washington, D.C.	270.0



Multi-table Query Example

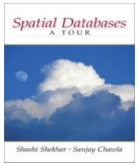
Query 5: What is the name and population of the capital city in the country where the St. Lawrence River originates?

```
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 ’
```



Ci.Name	Ci.Pop
Washington, D.C.	3.2

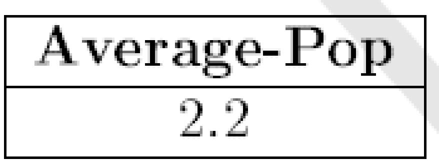
Note: Three tables are joined together pair at a time. River.Origin is matched with Country.Name and City.Country is matched with Country.Name. The order of join is decided by query optimizer and does not affect the result.



Query Examples…Aggregate Staistics

Query 6: 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 ’



Average-Pop
2.2

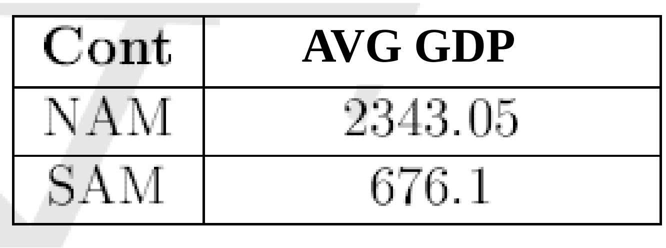
Query Examples…Aggregate Staistics

Query 7: For each continent, find the average GDP.

SELECT Co.Cont,Avg(Co.GDP)AS Continent-GDP

FROM Country Co

GROUP BY Co.Cont

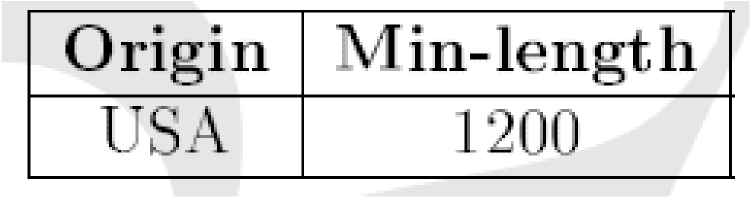


Cont	AVG GDP
NAM	2343.05
SAM	676.1

Query Example..Having clause, Nested queries

Query 8: 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
```

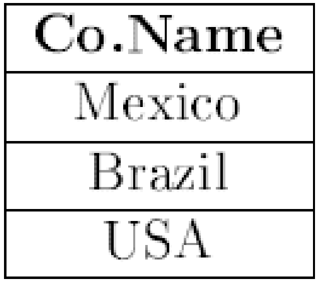


Origin	Min-length
USA	1200

Query Example..Having clause, Nested queries

Query 9: 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 ’)
```



Co.Name
Mexico
Brazil
USA