Chapter 4

Structured Query Language

Query

- A query is a “question” posed to a database
- Queries are expressed in a declarative manner

Examples:

- Mouse click on a Map Symbol (e.g River)
- Searching in search engine using keywords
- SELECT S.name FROM Employee E WHERE E.gender = ‘M’ means

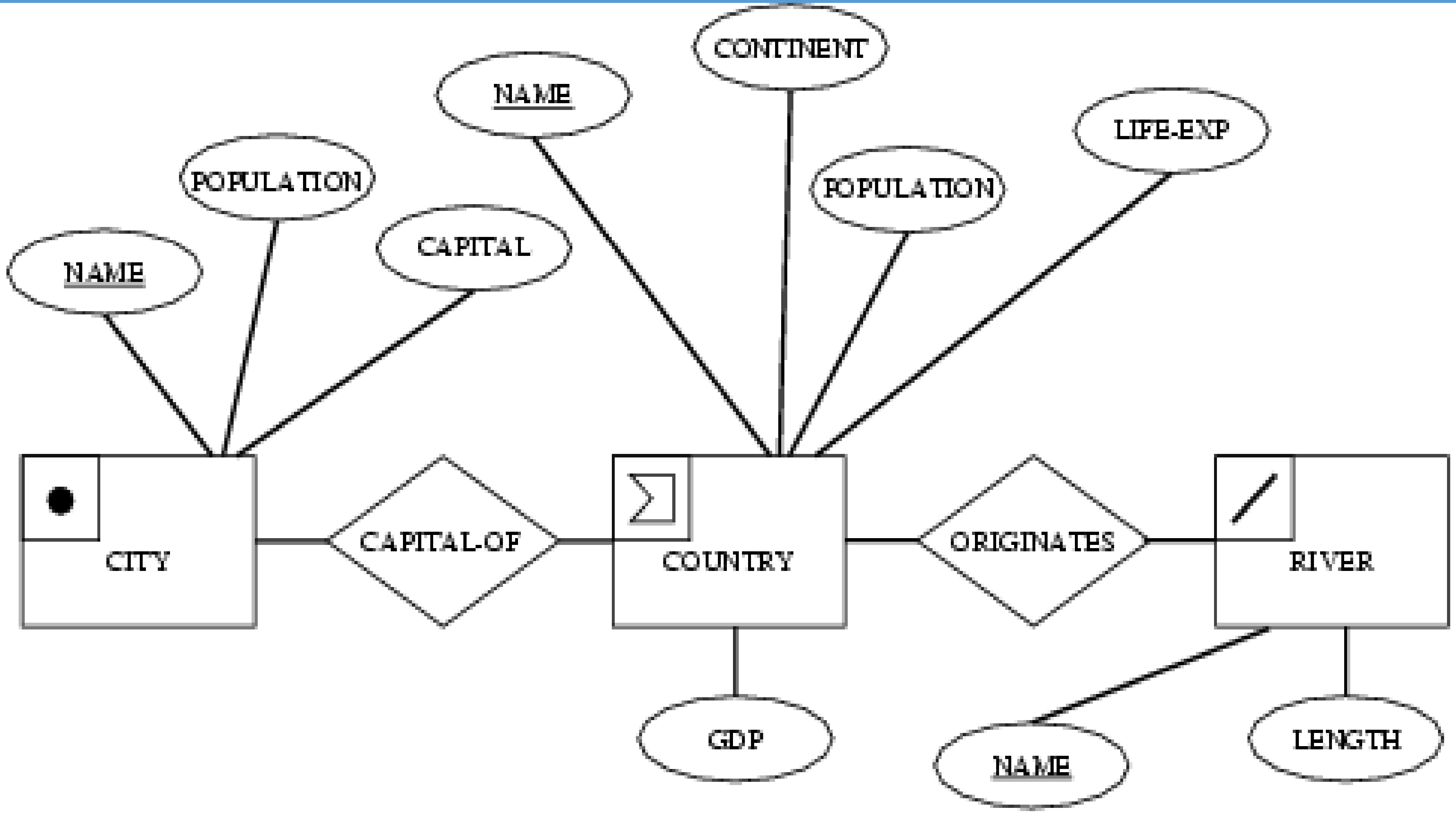
Query Language

- 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

Database Example

- Purpose: Use an example database to learn query language SQL
- Conceptual Model
  - 3 Entities: Country, City, River
  - 2 Relationships: capital-of, originates-in
  - Attributes
- 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

Database


CONTINENT
NAME
LIFE-EXP
POPULATION
POPULATION
NAME
CAPITAL
CITY
CAPITAL-OF
COUNTRY
ORIGINATES
RIVER
GDP
NAME
LENGTH

Structured Query Language

❖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!

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

Create

- **Table definition**
    - “CREATE TABLE” statement
    - Specifies table name, attribute names and data types
    - Create a table with no rows.

- **Related statements**
    - ALTER TABLE statement modifies table schema if needed
    - DROP TABLE statement removes an empty table

Add Data

- 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

Query

- 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

BASIC Query

```
create table department (
Dept_name varchar (20),
building varchar (15),
budget numeric (12,2),
primary key (Dept_name));

 create table course (
Course_id varchar (7),
title varchar (50),
Dept_name varchar (20),
credits numeric (2,0),
primary key (course_id),
foreign key (Dept_name) references department);
```

Query on A Single Relation

```
select name
from instructor;

select dept_name
from instructor;

select distinct dept_name
 from instructor;

select all dept_name
from instructor;

select ins_id, ins_name, dept_name,
salary * 1.1
from instructor;
```

```
select ins_name
from instructor
where dept_name = 'Comp. Sci.' and
salary > 70000;
```

Query on Multiple Relation

```
 select ins_name, instructor.dept_name, building
from instructor, department
where instructor.dept_name= department.dept_name;

select ins_name, course_id
from instructor, teaches
where instructor.ID= teaches.ID and instructor.dept_name = 'Comp. Sci.';

select ins_name as instructor_name, course_id
from instructor, teaches
where instructor.ins_id= teaches.teach_id;

select T.ins_name, S.course_id
from instructor as T, teaches as S
where T.ins_id= S.teach_id;
```

Set Operation

Union

```
(select course_id, semester,sem_year
from section
where semester = 'Fall‘ and sem_year= 2002)
union
(select course_id, semester,sem_year
from section
where semester = 'Spring'and sem_year= 2002);
```

```
select course_id, semester,sem_year
from section
where semester = 'Fall‘ and sem_year= 2002
```

```
select course_id, semester,sem_year
from section
where semester = 'Spring'and sem_year= 2002
```

Set Operation

Intersection

```
(select course_id
from section
where semester = 'Fall' and sem_year= 2006)
intersect
(select course_id
from section
where semester = 'Spring' and sem_year= 2008);
```

```
select course_id, semester,sem_year
from section
where semester = 'Fall‘ and sem_year= 2006

select course_id, semester,sem_year
from section
where semester = 'Spring‘ and sem_year= 2008
```

Set Operation

The except expression

```
(select course_id
from section
where semester = 'Fall' and sem_year= 2006)
except
(select course_id
from section
where semester = 'Spring' and sem_year= 2008);
```

```
select course_id, semester,sem_year
from section
where semester = 'Fall‘ and sem_year= 2006
```

```
select course_id, semester,sem_year
from section
where semester = 'Spring‘ and sem_year= 2008
```

Null values

- Null values present special problems in relational operations, including arithmetic operations, comparison operations, and set operations

Comparision

- and: The result of true and unknown is unknown, false and unknown is false, while unknown and unknown is unknown.
- or: The result of true or unknown is true, false or unknown is unknown, while unknown or unknown is unknown.
- not: The result of not unknown is unknown.

```
select ins_name
from instructor
where salary is  null;
```

Aggregation Function

Average: avg
Minimum: min
Maximum: max
Total: sum
Count: count

```
select avg (salary) as avg_salary
from instructor
where dept_name = 'Comp. Sci.';
```

```
select count (distinct teach_id)
from teaches
where semester = 'Spring' and sem_year = 2010;
```

```
select count (*)
from course;
```

```
select dept_name, avg (salary) as avg_salary
from instructor
group by dept_name;
```

```
select ins_name, dept_name,salary, avg
(salary) as avg_salary
from instructor
group by ins_name, dept_name,salary
order by dept_name
```

```
select dept_name, count (distinct ins_id) as
instr_count
from instructor, teaches
where instructor.ins_id= teaches.teach_id and
semester = 'Spring' and sem_year = 2010
group by dept_name;
```

The Having Clause

```
select dept_name, avg (salary) as avg_salary
from instructor
group by dept_name
having avg (salary) > 42000
order by avg_salary
```

Nested Subqueries

Set Membership

```
select  distinct course_id
from section
where semester = 'Fall' and sem_year= 2006 and
course_id in (select course_id
from section
where semester = 'Spring' and sem_year= 2008);
```

Set Comparision

```
select ins_name
from instructor
where salary > some (select salary
from instructor
where dept_name = 'Biology')
```

```
select distinct T.ins_name
from instructor as T, instructor as S
where T.salary > S.salary and
S.dept_name = 'Biology';
```

Nested Subqueries

Test for empty relation

```
select course_id
from section as S
where semester = 'Fall' and sem_year= 2006 and
exists (select *
from section as T
where semester = 'Spring' and sem_year= 2008 and
S.course_id= T.course_id)
```

```
(select course_id
from section
where semester = 'Fall' and sem_year= 2006)
intersect
(select course_id
from section
where semester = 'Spring' and sem_year= 2008);
```

The With Clause

```
with max_budget(value) as
(select max(budget)
from department)
select budget
from department, max_budget
where department.budget = max_budget.value;

with dept_total (dept_name, value) as
(select dept_name, sum(salary)
from instructor
group by dept_name),
dept_total_avg(value) as
(select avg(value)
from dept_total)
select dept_name
from dept_total, dept_total_avg
where dept_total.value > dept_total_avg.value;
```

The Natural Join

```
select st_namename, course_id
from student, takes
where student.st_id = takes.t_id;
```

```
select st_namename, course_id
from student natural join takes
```

Outer Join

Left outer join

Right outer join

Full outer join