Database Outline

- **DBMS Overview**
- **Relational Algebra**
- **SQL**
- **ODBC/JDBC/Cocoon SQL Processor**

Why use a DBMS in your website?

Suppose we are building web-based music distribution site. Several questions arise:

- How do we store the data? (file organization, etc.)
- How do we query the data? (write programs…)
- Make sure that updates don't mess things up?
- Provide different views on the data? (registrar versus students)
- How do we deal with crashes?

Way too complicated!

Buy a database system!

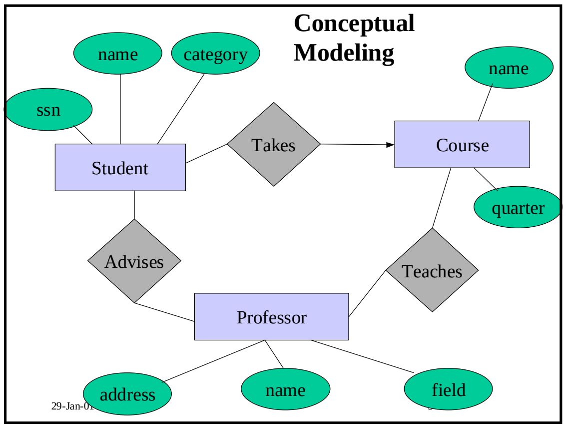
Functionality of a DBMS

- **Storage management**
- **Abstract data model**
- **High level query and data manipulation language**
- **Efficient query processing**
- **Transaction processing**
- **Resiliency: recovery from crashes**
- **Different views of the data, security**
- **Interface with programming languages**

Building an Application with a Database System

- **Requirements modeling (conceptual, pictures)**
  - Decide what entities should be part of the application and how they should be linked.
- **Schema design and implementation**
  - Decide on a set of tables, attributes.
  - Define the tables in the database system.
  - Populate database (insert tuples).
- **Write application programs using the DBMS**
  - Now much easier, with data management API

Conceptual Modeling



Schema Design & Implementation

- Table Students

Student	Course	Quarter
Charles	CS 444	Fall, 1997
Dan	CS 142	Winter, 1998
…	…	…

- Separates the logical view from the physical view of the data.

Querying a Database

- **Find all the students taking CSE490i in Q1, 2000**
- **S(tructured) Q(uery) L(anguage)**

  select E.name
  from Enroll E
  where E.course=CS490i and
  E.quarter="Winter, 2000"

- **Query processor figures out how to answer the query efficiently.**

Relational Algebra

- **Operators**
  - tuple sets as input, new set as output
- **Basic Binary Set Operators**
  - Result is table (set) with same attributes
    - Sets must be compatible!
      - $R1(A1,A2,A3) \cap R2(B1,B2,B3)$
      - $\therefore$ Domain(Ai) = Domain(Bi)
  - Union
    - All tuples in either R1 or in R2
  - Intersection
    - All tuples in both R1 and R2
  - Difference
    - All tuples in R1 but not in R2
  - ~~Complement~~ - *what's the universe?*
- **Selection, Projection, Cartesian Product, Join**

Selection σ

- **Grab a subset of the tuples in a relation that satisfy a given condition**
  - Use and, or, not, >, <… to build condition
- **Unary operation… returns set with same attributes, but 'selects' rows**

Selection Example

Employee

SSN	Name	DepartmentID	Salary
999999999	John	1	30,000
777777777	Tony	1	32,000
888888888	Alice	2	45,000

Select (Salary > 40000)

SSN	Name	DepartmentID	Salary
888888888	Alice	2	45,000

Projection π

- **Unary operation, selects columns**
- **Returned schema is *different*,**
  - So returned tuples are not subset of original set
  - Contrast with selection
- **Eliminates duplicate tuples**

Example: Projection Onto SSN, Name

Employee

SSN	Name	DepartmentID	Salary
999999999	John	1	30,000
777777777	Tony	1	32,000
888888888	Alice	2	45,000

SSN	Name
999999999	John
777777777	Tony
888888888	Alice

Cartesian Product X

- **Binary Operation**
- **Result is set of tuples combining all elements of R1 with all elements of R2, for R1 × R2**
- **Schema is union of Schema(R1) & Schema(R2)**
- **Notice we could do selection on result to get meaningful info!**

Cartesian Product Example

Employee

Name	SSN
John	999999999
Tony	777777777

Dependents

EmployeeSSN	Dname
999999999	Emily
777777777	Joe

Employee_Dependents

Name	SSN	EmployeeSSN	Dname
John	999999999	999999999	Emily
John	999999999	777777777	Joe
Tony	777777777	999999999	Emily
Tony	777777777	777777777	Joe

Join ⋈

- **Most common (and exciting!) operator…**
- **Combines 2 relations**
  - Selecting only related tuples
- **Equivalent to**
  - Cross product followed by selection
- **Result has all attributes of the two relations**
- **Equijoin**
  - Join condition is equality between two attributes
- **Natural join**
  - Equijoin on attributes of same name
  - result has only one copy of join condition attribute

Example: Natural Join

Employee

Name	SSN
John	999999999
Tony	777777777

Dependents

SSN	Dname
999999999	Emily
777777777	Joe

Employee ⋈ Dependents

Employee_Dependents

Name	SSN	Dname
John	999999999	Emily
Tony	777777777	Joe

Complex Queries

Product (pname, price, category, maker)
Purchase (buyer, seller, store, prodname)
Company (cname, stock price, country)
Person(per-name, phone number, city)

Find phone numbers of people who bought gizmos from Fred.

Find telephony products that somebody bought

Exercises

Product (pname, price, category, maker)
Purchase (buyer, seller, store, prodname)
Company (cname, stock price, country)
Person(per-name, phone number, city)

Ex #1: Find people who bought telephony products.
Ex #2: Find names of people who bought American products
Ex #3: Find names of people who bought American products and did not buy French products
Ex #4: Find names of people who bought American products and they live in Seattle.
Ex #5: Find people who bought stuff from Joe or bought products from a company whose stock prices is more than $50.