CS2102 Lecture 1
Introduction

Database Management System (DBMS)

- What is a DBMS?
  - Software for managing data
- Advantages of a DBMS
  - Data Independence
  - Efficient Data Access
  - Data Integrity & Security
  - Data Administration
  - Concurrent Access & Crash Recovery
  - Reduced Application Development Time

Study of DBMS

- Database design
  - How to model the data requirements of applications
  - How to organize data using a DBMS
  - Topics: relational model, ER model, schema refinement
- Database programming
  - How to create, query, and update a database
  - How to specify data constraints
  - How to use SQL in applications
  - Topics: SQL, relational algebra & calculus
- DBMS implementation
  - How to build a DBMS
  - Covered in CS3223

Describing Data in a DBMS

- A DBMS allows users to define and query data in terms of a *data model*
- A data model is a collection of concepts for describing data
- A schema is a description of the structure of a database using a data model
- A schema instance is the content of the database at a particular time

Data Models

- Network Model (e.g., General Electric’s IDS (1964))
- Hierarchical Model (e.g., IBM’s IMS (1966))
- Relational Model
    - **Commercial RDBMS**: IBM DB2, Microsoft SQL Server, Oracle, SAP ASE, etc.
    - **Open-source RDBMS**: MariaDB, MySQL, SQLite, etc.
- Object-oriented Model (e.g., ObjectStore 1988)
- Object-relational Model (e.g., Postgres 1986)
- etc.

Relational Data Model

- Introduced by **Edgar Codd** of IBM Research Laboratory in 1970
- Data is modeled using relations (tables with rows & columns)

Students

studentId	*name*	*birthDate*	*cap*
3118	Alice	1999-12-25	3.8
1423	Bob	2000-05-27	4.3
5609	Carol	1999-06-11	4.0

Degree/Arity = Num. of columns

Cardinality = Num. of rows

- Each relation has a definition called a relation schema
    - Schema specifies attributes and data constraints
    - Data constraints include domain constraints

  Students (*studentId*: **integer**, *name*: **string**, *birthDate*: **date**, *cap*: **real**)

- Each row in a relation is called a tuple/record; it has one component for each attribute of relation

  (1423, ‘Bob’, 2000-05-27, 4.3)

Relational Data Model (cont.)

- Domain - a set of atomic values
  - Examples: integer, real, string
  - The special value null is a member of each domain
  - A null value means value is either not applicable or unknown
- A relation is a set of tuples
  - Consider a relation schema $R(A_1, A_2, \cdots, A_n)$ with n attributes $A_1, \cdots, A_n$
  - Let D_i be the domain of attribute A_i (set of possible values for A_i)
  - Each instance of schema R is a relation which is a subset of $\{(a_1, a_2, \cdots, a_n) \mid a_i \in D_i\}$

Relational Data Model (cont.)

- A relational database schema consists of a set of relation schemas

  Students (*studentId*: **integer**, *name*: **string**, *age*: **integer**, *cap*: **real**)
  Courses (*courseId*: **integer**, *name*: **string**, *credits*: **integer**)
  Enrols (*sid*: **integer**, *cid*: **integer**, *grade*: **real**)

- A relational database is a collection of tables

Students

studentId	*name*	*age*	*cap*
3118	Alice	21	4.0
1423	Bob	18	4.3
5609	Carol	18	3.8

Courses

courseId	*name*	*credits*
101	Programming in C	5
112	Discrete Mathematics	4
204	Analysis of Algorithms	4
311	Database Systems	5

Enrols

sid	*cid*	*grade*
3118	101	5.0
3118	112	4.0
3118	204	3.0
1423	112	4.5
. . . . . .	. . . . . .	. . . . . .

- Relational database schema = relational schemas + data constraints

Integrity Constraints (ICs)

- Integrity constraint: a condition that restricts the data that can be stored in database instance
  - Specified when schema is defined
  - ICs are checked when relations are updated
- A legal relation instance is a relation that satisfies all specified ICs.
- A DBMS enforces ICs - allows only legal instances to be stored

Types of Integrity Constraints

- Domain constraints restrict attribute values of relations
- Key constraints
- Foreign key constraints
- Other general constraints

ICs: Key Constraints

- A superkey is a subset of attributes in a relation that uniquely identifies its tuples
    - No two distinct tuples of a relation have the same values in all attributes of superkey
- **Example**: What are the possible superkeys for the following relation?

sid	*cid*	*grade*
1	204	3.0
1	101	5.0
2	204	3.0
3	101	4.0
3	112	4.0

ICs: Key Constraints (cont.)

- A key is a superkey that satisfies the additional property:
    - No *proper subset* of the key is a superkey
- Thus, a key is a minimal subset of attributes in a relation that uniquely identifies its tuples
- Key attribute values cannot be *null*

ICs: Key Constraints (cont.)

- A relation could have multiple keys called candidate keys
- One of the candidate keys is selected as the primary key
- **Example**:
  - `Students` (studentId, name, email, birthDate)
  - There are two candidate keys: {*studentId*} and {email}
  - Any one of them could be selected as the primary key

ICs: Foreign Key Constraints

- A subset of attributes in a relation is a foreign key if it refers to the primary key of a second relation

Enrolls

sid	*cid*	*grade*
1	**204**	3.0
1	**101**	5.0
2	**204**	3.0
3	**101**	4.0
3	**112**	4.0

Referencing relation

Courses

courseId	*name*	*credits*
101	Programming in C	5
112	Discrete Mathematics	4
204	Analysis of Algorithms	4
311	Database Systems	5

Referenced relation

- *cid* is a foreign key in `Enrolls` that refers to the primary key *courseId* in `Courses`
- Foreign key constraint: each foreign key value in referencing relation must either (1) appear as primary key value in referenced relation or (2) be a null value

ICs: Foreign Key Constraints (cont.)

- Foreign key constraints can arise within same relation
- **Example**: Each course has at most one pre-requisite course

Course_Prerequisites

cid	*prerequisite*
101	*null*
204	**101**
311	**101**
123	*null*
210	**123**

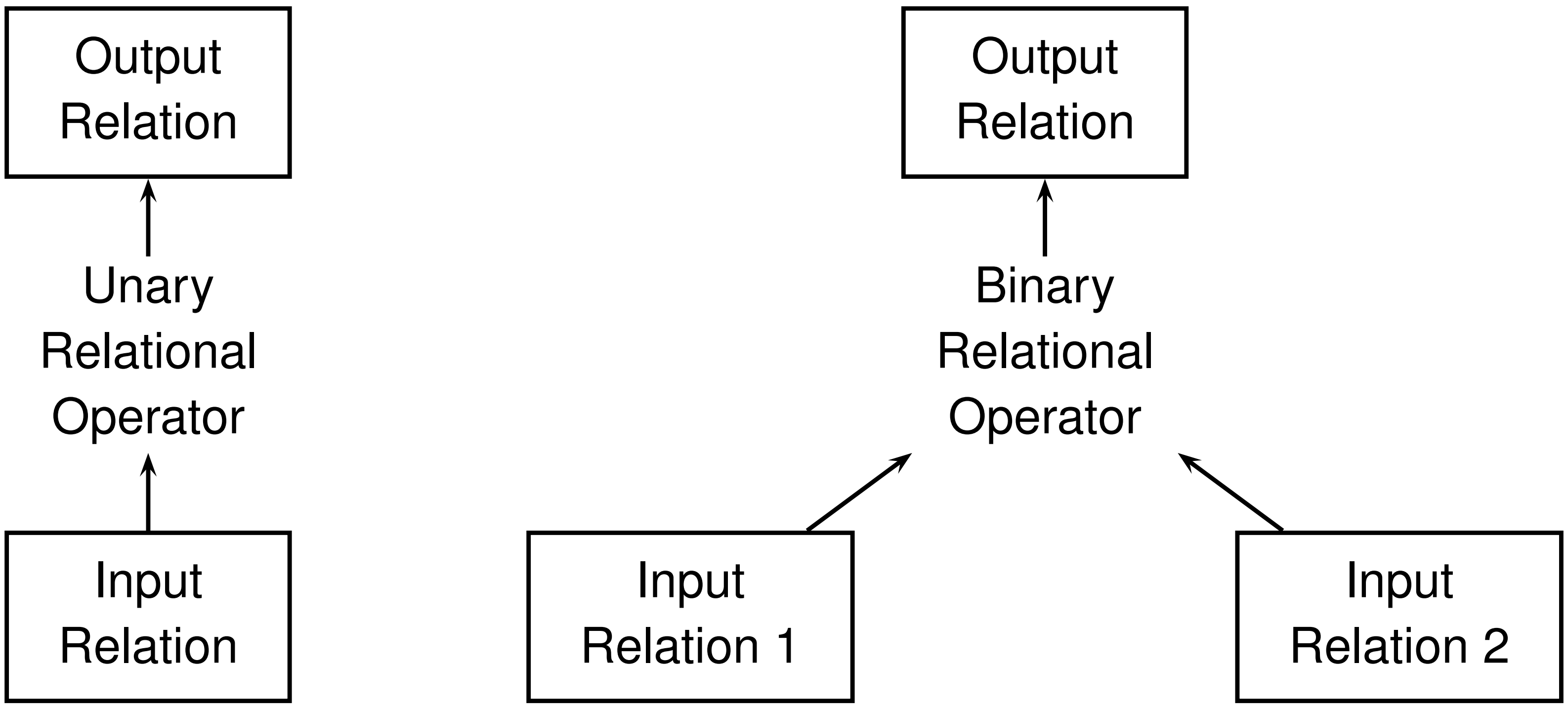
- Constraints in `Course_Prerequisite` table:
  - *cid* is the primary key
  - *prerequisite* is a foreign key that refers to *cid*
- Foreign key constraints = referential integrity constraints

Relational Algebra

- A formal language for asking queries on relations
- A query is composed of a collection of operators called relational operators
- Each operator takes one/two relations as input and computes an output relation
- Basic relational algebra operators
    - **Unary operators**: selection σ, projection π, renaming ρ
    - **Binary operators**: cross-product $\times$, union $\cup$, intersect $\cap$, difference $-$

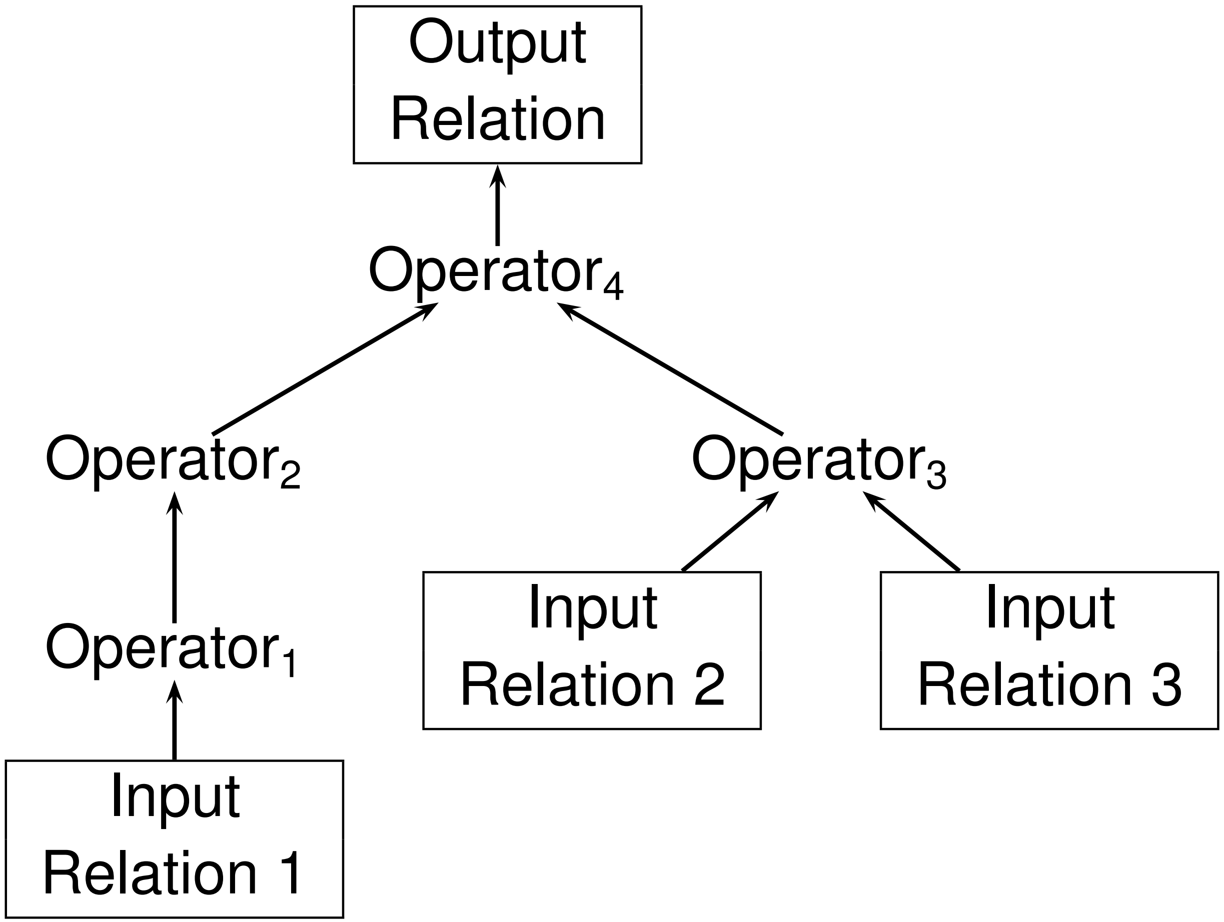
Closure Property

- Relations are *closed* under relational operators



Closure Property (cont.)

- Operators can be *composed* to form relational algebra expressions



Example Database

Contains

pizza	ingredient
Diavola	cheese
Diavola	chilli
Diavola	salami
Funghi	ham
Funghi	mushroom
Hawaiian	ham
Hawaiian	pineapple
Margherita	cheese
Margherita	tomato
Marinara	seafood
Siciliana	anchovies
Siciliana	capers
Siciliana	cheese

Restaurants

rname	area
Corleone Corner	North
Gambino Oven	Central
Lorenzo Tavern	Central
Mamma's Place	South
Pizza King	East

Sells

rname	pizza	price
Corleone Corner	Diavola	24
Corleone Corner	Hawaiian	25
Corleone Corner	Margherita	19
Gambino Oven	Siciliana	16
Lorenzo Tavern	Funghi	23
Mamma's Place	Marinara	22
Pizza King	Diavola	17
Pizza King	Hawaiian	21

Customers

cname	area
Homer	West
Lisa	South
Maggie	East
Moe	Central
Ralph	Central
Willie	North

Likes

cname	pizza
Homer	Hawaiian
Homer	Margherita
Lisa	Funghi
Maggie	Funghi
Moe	Funghi
Moe	Sciliana
Ralph	Diavola

Selection: σ_c

- $\sigma_c(R)$ selects tuples from relation R that satisfy *selection condition c*
- **Example:** Find all restaurants, the pizzas that they sell and their prices, where the price is under \$20

Sells

rname	pizza	price
Corleone Corner	Diavola	24
Corleone Corner	Hawaiian	25
Corleone Corner	Margherita	19
Gambino Oven	Siciliana	16
Lorenzo Tavern	Funghi	23
Mamma's Place	Marinara	22
Pizza King	Diavola	17
Pizza King	Hawaiian	21

$\sigma_{\text{price}<20}(\text{Sells})$

rname	pizza	price
Corleone Corner	Margherita	19
Gambino Oven	Siciliana	16
Pizza King	Diavola	17

Selection Conditions

- Selection condition is a boolean combination of *terms*
- A term is one of the following forms:
  1. attribute **op** constant
  2. attribute_1 **op** attribute_2
  3. $\text{term}_1 \wedge \text{term}_2$
  4. $\text{term}_1 \vee \text{term}_2$
  5. $\neg\ \text{term}_1$
  6. (term_1)

$\boxed{\textbf{op} \in \{=, \neq, <, \leq, >, \geq\}}$

- Operator precedence: $()$, *op*, $\neg$, $\wedge$, $\vee$

Projection: π_ℓ

- $\pi_\ell(R)$ projects attributes given by a list ℓ of attributes from relation R
- **Example:** Find all restaurants and the pizzas that they sell

Sells

rname	pizza	price
Corleone Corner	Diavola	24
Corleone Corner	Hawaiian	25
Corleone Corner	Margherita	19
Gambino Oven	Siciliana	16
Lorenzo Tavern	Funghi	23
Mamma's Place	Marinara	22
Pizza King	Diavola	17
Pizza King	Hawaiian	21

$\pi_{\text{rname,pizza}}(\text{Sells})$

rname	pizza
Corleone Corner	Diavola
Corleone Corner	Hawaiian
Corleone Corner	Margherita
Gambino Oven	Siciliana
Lorenzo Tavern	Funghi
Mamma's Place	Marinara
Pizza King	Diavola
Pizza King	Hawaiian

Projection: π_ℓ (cont.)

- Duplicate records are removed in the output relation
- **Example**: Find all pizza ingredients

Contains

pizza	ingredient
Diavola	cheese
Diavola	chilli
Diavola	salami
Funghi	ham
Funghi	mushroom
Hawaiian	ham
Hawaiian	pineapple
Margherita	cheese
Margherita	tomato
Marinara	seafood
Siciliana	anchovies
Siciliana	capers
Siciliana	cheese

$\pi_{\text{ingredient}}(\text{Contains})$

ingredient
cheese
chilli
salami
ham
mushroom
pineapple
tomato
seafood
anchovies
capers

Renaming: ρ_S

- Given relation $R(A, B, C)$, $\rho_{S(X,Y,Z)}(R)$ renames $R(A, B, C)$ to $S(X, Y, Z)$

Restaurants

rname	**area**
Corleone Corner	North
Gambino Oven	Central
Lorenzo Tavern	Central
Mamma's Place	South
Pizza King	East

Shops = $\rho_{\text{Shops(sname,region)}}$(Restaurants)

sname	**region**
Corleone Corner	North
Gambino Oven	Central
Lorenzo Tavern	Central
Mamma's Place	South
Pizza King	East

Cross-Product: $\times$

- Consider relations $R(A, B, C)$ and $S(X, Y)$
- Cross-product: $R \times S$ returns a relation with schema (A, B, C, X, Y) defined as follows:

$$R \times S = \{(a, b, c, x, y) \mid (a, b, c) \in R,\ (x, y) \in S\}$$

- Cross-product operation is also known as cartesian product

Cross-Product: Example

- Find all customer-restaurant pairs that are located in the central area

Customers

cname	area
Homer	West
Lisa	South
Maggie	East
Moe	Central
Ralph	Central
Willie	North

Restaurants

rname	area
Corleone Corner	North
Gambino Oven	Central
Lorenzo Tavern	Central
Mamma's Place	South
Pizza King	East

$\pi_{cname}(\sigma_{area='Central'}(\text{Customers}))$

cname
Moe
Ralph

$\pi_{rname}(\sigma_{area='Central'}(\text{Restaurants}))$

rname
Gambino Oven
Lorenzo Tavern

$\pi_{cname}(\sigma_{area='Central'}(\text{Customers})) \times \pi_{rname}(\sigma_{area='Central'}(\text{Restaurants}))$

cname	rname
Moe	Gambino Oven
Moe	Lorenzo Tavern
Ralph	Gambino Oven
Ralph	Lorenzo Tavern

Relational Algebra: Example

- Find customer pairs $(C1, C2)$ such that they like some common pizza and $C1 < C2$

Likes

cname	pizza
Homer	Hawaiian
Homer	Margherita
Lisa	Funghi
Maggie	Funghi
Moe	Funghi
Moe	Sciliana
Ralph	Diavola

R

cname	cname2
Lisa	Maggie
Lisa	Moe
Maggie	Moe

$$R = \sigma_{(pizza=pizza2) \wedge (cname<cname2)}(Likes \times \rho_{Likes2(cname2,pizza2)}(Likes))$$

Set Operators

- Union: $R \cup S$ returns a relation containing all tuples that occur in R or S (or both)
- Intersection: $R \cap S$ returns a relation containing all tuples that occur in both R and S
- Set-difference: $R - S$ returns a relation containing all tuples in R but not in S
- Union ($\cup$), intersection ($\cap$), and set-difference ($-$) operators require input relations to be union compatible
- The schema of the result of "R op S", where $op \in \{\cup, \cap, -\}$, is identical to the schema of R

Set Operations: Union Compatibility

- Two relations are union compatible if
  1. they have the same number of attributes, and
  2. the corresponding attributes have the same domains
- Union compatible relations do not necessarily use the same attribute names
- **Example**:
  - Student (sid:**integer**, dob:**date**, name:**string**)
  - GradStudent (sid:**integer**, name:**string**)
  - Report (reportid:**integer**, title:**string**, pubdate:**date**)

Set Operators: Examples

- Consider the following database schema:
  - `Restaurants`(rname:**string**, area:**string**)
  - `Customers`(cname:**string**, area:**string**)
- **Example 1**: Find all customer/restaurant names

$$\pi_{rname}(\text{Restaurants}) \cup \pi_{cname}(\text{Customers})$$

- **Example 2**: Find all pizzas that contain both cheese and chilli

$$\pi_{pizza}(\sigma_{ingredient='cheese'}(\text{Contains})) \cap \pi_{pizza}(\sigma_{ingredient='chilli'}(\text{Contains}))$$

- **Example 3**: Find all pizzas that contain cheese but not chilli

$$\pi_{pizza}(\sigma_{ingredient='cheese'}(\text{Contains})) - \pi_{pizza}(\sigma_{ingredient='chilli'}(\text{Contains}))$$

Summary

- DBMS used to store, update, and query data
- Relational data model
    - Tabular representation of data
    - Integrity constraints specify restrictions on data based on application semantics
    - Relational algebra provides formal language for querying relations