CS2102
Database Systems

Slides adapted from Prof. Chan Chee Yong

LECTURE 02

INTRODUCTION

PREVIOUS LECTURE

Relational data model

History

- Introduced by **Edgar Codd** of IBM Research Lab in 1970
- Data is modeled using relations
  - Relations are simply tables with rows & columns

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

`(3118, 'Alice', 1998-12-25, 3.8)` is NOT a student but contains all information needed to model the student

  - **Definitions**
    - **Degree/Arity** : *number of columns*
    - **Cardinality** : *number of rows*

PREVIOUS LECTURE

Relational data model

Relation schema

- Each relation has a definition called a relation schema
  - Schema specifies attributes and data constraints
  - Data constrains include domain constraints
    - `Students (studentID: integer, name: string, birthDate: date, cap : numeric)`
  - Each row in a relation is called a tuple/record
    - It has one component for each attribute of relation
      - Example: `(1423, "Bob", 2000-05-07, 4.0)`

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

Tuple is like set, but position matters. Set of {A,B,C} is equivalent to set of {B,C,A}, but tuple (A,B,C) is not equivalent to tuple (B,C,A)

- Relational data model
- Integrity constraints
  - Key constraints
  - Foreign key constraints
- Relational algebra
  - Unary operators
  - Binary operators
  - Closure properties

Overview

- Relational data model
- Integrity constraints
  - Key constraints
  - Foreign key constraints
- Relational algebra
  - Unary operators
  - Binary operators
  - Closure properties

Relational data model

Relational data model

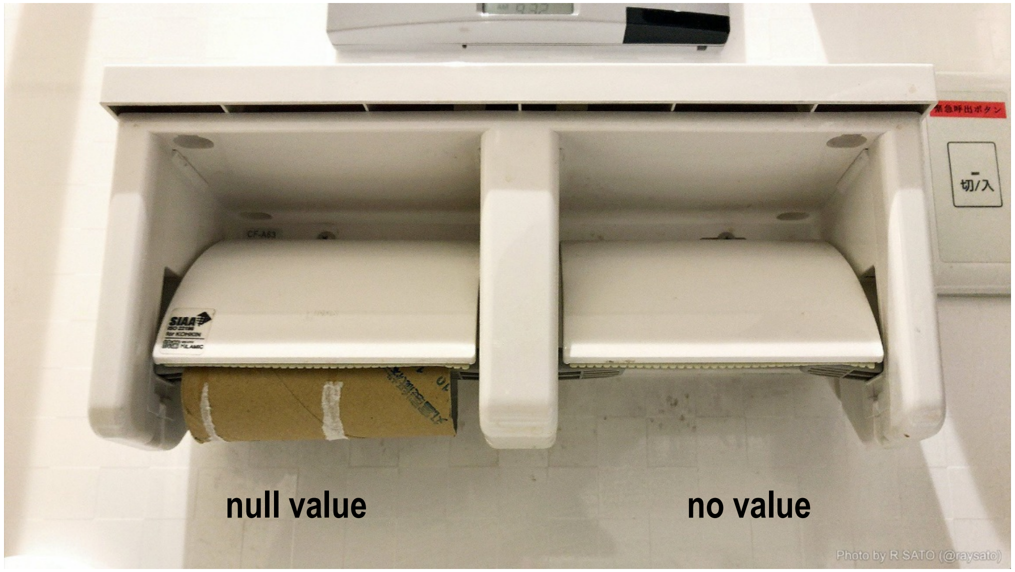
Domain

- Domain is defined as a set of *<u>atomic values</u>*
  - Examples: `integer, numeric, string`
  - The special value `null` is a member of each domain
  - A `null` value means value is either not applicable or unknown

Relational data model

Domain

- Domain is defined as a set of *atomic values*
  - Examples: `integer, numeric, string`
  - The special value `null` is a member of each domain
  - A `null` value means value is either not applicable or unknown



(image: reddit)

Relational data model

Domain

- Domain is defined as a set of *atomic values*
  - Examples: `integer, numeric, string`
  - The special value `null` is a member of each domain
  - A `null` value means value is either not applicable or unknown

HOW A 'NULL' LICENSE PLATE LANDED ONE HACKER IN TICKET HELL

(source: https://www.wired.com/story/null-license-plate-landed-one-hacker-ticket-hell/)

Relational data model

Relations

- A relation is defined as a set of *tuples*
  - Consider a relation schema $R(A_1, A_2, \dots, A_n)$ with n attributes $A_1, A_2, \dots, A_n$
  - Example
    - ```
      Students (studentID: integer, name: string,
                birthDate: date,    cap : numeric)
      ```
      - R
      - A_1
      - A_2
      - A_3
      - A_4
      ```

Relational data model

Relations

- A relation is defined as a set of *tuples*
  - Consider a relation schema $R(A_1, A_2, \dots, A_n)$ with n attributes $A_1, A_2, \dots, A_n$
  - Example
    - ```
      Students (studentID: integer, name: string,
                birthDate: date,    cap : numeric)
      ```
      - R Students
      - A_1 studentID
      - A_2 name
      - A_3 birthDate
      - A_4 cap
      ```

Relational data model

Relations

- A relation is defined as a set of *tuples*
  - Consider a relation schema $R(A_1, A_2, \ldots, A_n)$ with n attributes $A_1, A_2, \ldots, A_n$
  - Let D_i be the domain of attribute A_i (set of possible values of A_i)
  - Example
    - ```
      Students (studentID: integer, name: string,
                birthDate: date,    cap : numeric)
      ```
      - D_1
      - D_2
      - D_3
      - D_4
      ```

Relational data model

Relations

- A relation is defined as a set of *tuples*
  - Consider a relation schema $R(A_1, A_2, \dots, A_n)$ with n attributes $A_1, A_2, \dots, A_n$
  - Let D_i be the domain of attribute A_i (set of possible values of A_i)
  - Example
    - `Students (`***`studentID`***`: integer, `***`name`***`: string,`
      `birthDate``: date, `***`cap`***` : numeric)`
      - D_1 `integer`
      - D_2 `string`
      - D_3 `date`
      - D_4 `numeric`

Relational data model

Relations

- A relation is defined as a set of *tuples*
  - Consider a relation schema $R(A_1, A_2, \ldots, A_n)$ with n attributes $A_1, A_2, \ldots, A_n$
  - Let D_i be the domain of attribute A_i (set of possible values of A_i)
  - Each instance of schema R is a relation which is a subset of $\{(a_1, a_2, \ldots, a_n) | a_i \in D_i\}$

Relational data model

Relations

- A relation is defined as a set of *tuples*
  - Consider a relation schema $R(A_1, A_2, \dots, A_n)$ with n attributes $A_1, A_2, \dots, A_n$
  - Let D_i be the domain of attribute A_i (set of possible values of A_i)
  - Each instance of schema R is a relation which is a subset of $\underbrace{\{(a_1, a_2, \dots, a_n) | a_i \in D_i\}}_{\text{a set}}$

Relational data model

Relations

- A relation is defined as a set of ***tuples***
  - Consider a relation schema $R(A_1, A_2, \ldots, A_n)$ with n attributes $A_1, A_2, \ldots, A_n$
  - Let D_i be the domain of attribute A_i (set of possible values of A_i)
  - Each instance of schema R is a relation which is a subset of $\{(a_1, a_2, \ldots, a_n) | a_i \in D_i\}$

a set $\boxed{\text{consisting of tuples}}$

Relational data model

Relations

- A relation is defined as a set of *tuples*
  - Consider a relation schema $R(A_1, A_2, \dots, A_n)$ with n attributes $A_1, A_2, \dots, A_n$
  - Let D_i be the domain of attribute A_i (set of possible values of A_i)
  - Each instance of schema R is a relation which is a subset of $\{(a_1, a_2, \dots, a_n) | a_i \in D_i\}$

a set consisting of tuples $\boxed{\text{of } n \text{ elements}}$

Relational data model

Relations

- A relation is defined as a set of *tuples*
  - Consider a relation schema $R(A_1, A_2, \dots, A_n)$ with n attributes $A_1, A_2, \dots, A_n$
  - Let D_i be the domain of attribute A_i (set of possible values of A_i)
  - Each instance of schema R is a relation which is a subset of $\{(a_1, a_2, \dots, a_n) | a_i \in D_i\}$

a set consisting of tuples of n elements $\boxed{\text{where each element}}$

Relational data model

Relations

- A relation is defined as a set of *tuples*
  - Consider a relation schema $R(A_1, A_2, \dots, A_n)$ with n attributes $A_1, A_2, \dots, A_n$
  - Let D_i be the domain of attribute A_i (set of possible values of A_i)
  - Each instance of schema R is a relation which is a subset of $\{(a_1, a_2, \dots, a_n) | a_i \in D_i\}$

a set consisting of tuples of n elements where each element is within domain

Relational data model

Relations

- A relation is defined as a set of *tuples*
  - Consider a relation schema $R(A_1, A_2, \dots, A_n)$ with n attributes $A_1, A_2, \dots, A_n$
  - Let D_i be the domain of attribute A_i (set of possible values of A_i)
  - Each instance of schema R is a relation which is a subset of $\{(a_1, a_2, \dots, a_n) | a_i \in D_i\}$

a set   consisting of tuples   of n elements   where each element   is within domain

Relational data model

Relations

- A relation is defined as a set of *tuples*
  - Consider a relation schema $R(A_1, A_2, \ldots, A_n)$ with n attributes $A_1, A_2, \ldots, A_n$
  - Let D_i be the domain of attribute A_i (set of possible values of A_i)
  - Each instance of schema R is a relation which is a subset of $\{(a_1, a_2, \ldots, a_n) | a_i \in D_i\}$
  - Example
    - ```
      Students (studentID: integer, name: string,
                birthDate: date,    cap : numeric)
      ```
    - ```
      {(1423, "Bob"  , 2000-05-27, 4.0),
       (3118, "Alice", 1999-12-25, 3.8)}
      ```

Relational data model

Relational database schema

- A relational database schema consists of a set of *schemas*
  - Example:
    - ```
      Students (studentID: integer, name: string,
                birthDate: date,    cap : numeric)
      ```
    - ```
      Courses  (courseID: integer, name: string,
                credits : integer)
      ```
    - ```
      Enrolls  (sID: integer, grade: numeric,
                cID: integer)
      ```

❖ *Relational database schema*
= relational schemas + data constraints
      ```

Relational data model

Relational database

- A relational database is a collection of *tables*
  - Example:
    - Students

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

    - Courses

courseID	*name*	*credits*
101	Programming in C	3.8
112	Discrete Mathematics	4.0
311	Database Systems	4.3

    - Enrolls

sID	*cID*	*grade*
3118	101	3.0
3118	112	4.0
1423	311	4.5

- Relational data model
- Integrity constraints
  - Key constraints
  - Foreign key constraints
- Relational algebra
  - Unary operators
  - Binary operators
  - Closure properties

Integrity constraints

Integrity constraints (ICs)

Definitions

- Integrity constraint
  - A condition that *restricts* the data that can be stored in database instance
    - ICs are specified when schema is defined
    - ICs are checked when relations are updated
- Legal relation instance
  - A relation that satisfies all specified ICs

- A DBMS enforces ICs
  - Allow only legal instances to be stored

Integrity constraints (ICs)

Types

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

Key constraints

Superkey

- A superkey is a subset of attributes in a relation that *uniquely identifies* its tuples
  - No two distinct tuples of relation have the same values in all attributes of superkey

Key constraints

Superkey

- A superkey is a subset of attributes in a relation that *uniquely identifies* its tuples
  - No two distinct tuples of relation have the same values in all attributes of superkey
- Example
  - Which of the following could be a superkey of the table on the right?
    1. sID
    2. cID
    3. grade
    4. (sID, cID)
    5. (sID, grade)
    6. (cID, grade)
    7. (sID, cID, grade)

sID	***cID***	***grade***
3118	101	3.0
3118	112	4.0
5609	112	1.0
1423	311	4.5

Key constraints

Superkey

- A superkey is a subset of attributes in a relation that ***uniquely identifies*** its tuples
  - No two distinct tuples of relation have the same values in all attributes of superkey
- Example
  - Which of the following could be a superkey of the table on the right?

1. ~~sID~~
2. ~~cID~~
3. grade
4. (sID, cID)
5. (sID, grade)
6. (cID, grade)
7. (sID, cID, grade)

sID	*cID*	*grade*
3118	101	3.0
3118	112	4.0
5609	112	1.0
1423	311	4.5

Key constraints

Key

- A key is a superkey that satisfies the additional property
  - Not `null` & no *<u>proper subset</u>* of a key is a superkey
  - ➢ *<u>Minimal</u>* subset of attributes that *<u>uniquely identifies</u>* its tuples

Key constraints

Key

- A key is a superkey that satisfies the additional property
  - Not `null` & no *proper subset* of a key is a superkey
  - ➢ *Minimal* subset of attributes that *uniquely identifies* its tuples
- Example
  - Which of the following could be a key of the table on the right?
    1. sID
    2. cID
    3. grade
    4. (sID, cID)
    5. (sID, grade)
    6. (cID, grade)
    7. (sID, cID, grade)

sID	*cID*	*grade*
3118	101	3.0
3118	112	4.0
5609	112	1.0
1423	311	4.5

Key constraints

Key

- A key is a superkey that satisfies the additional property
  - Not `null` & no *proper subset* of a key is a superkey
  - ➢ *Minimal* subset of attributes that *uniquely identifies* its tuples
- Example
  - Which of the following could be a key of the table on the right?

~~1. sID~~
~~2. cID~~
3. grade
4. (sID, cID)
~~5. (sID, grade)~~
~~6. (cID, grade)~~
~~7. (sID, cID, grade)~~

sID	*cID*	*grade*
3118	101	3.0
3118	112	4.0
5609	112	1.0
1423	311	4.5

Key constraints

Notation

- We indicate a key with an arrow
- Example
  - `Students (`***`studentID`***`: integer,` ***`name`***`: string,`
    `birthDate``: date,` ***`cap`*** `: numeric)`
  - If `studentID` is a key, then we write
    - `studentID → (studentID, name, birthDate, cap)`
      OR
      `studentID -> (studentID, name, birthDate, cap)`
      if there's no arrow symbol

Key constraints

Notation

- We indicate a key with an arrow
- Example
  - `Students (`***`studentID`***`: integer,` ***`name`***`: string,`
    `birthDate``: date,` ***`cap`*** `: numeric)`
  - If `studentID` is a key, then we write
    - `studentID → (studentID, name, birthDate, cap)`
      OR
      `studentID -> (studentID, name, birthDate, cap)`
      if there's no arrow symbol
  - More generally
    - `LHS → RHS`
    - If every unique value of `LHS` is associated with <u>*exactly one value*</u> of `RHS`
      - *Example:* the same `studentID` cannot belong to different `name`
      - *Therefore:* `studentID → name`

Key constraints

Properties

- A relation can have *multiple keys*
  - These are called candidate keys
  - One of the candidate keys is then selected as the primary keys
- We denote primary key with underline

Example

- `Students (studentID, name, birthDate, cap)`
  - `studentID` is the primary key, hence it is underlined
- `Enrolls (sID, cID, grade)`
  - `sID` and `cID` are the primary keys, hence they are underlined

Foreign key constraints

Foreign key

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

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

Students

sID	*cID*	*grade*
3118	101	3.0
3118	112	4.0
1423	311	4.5

Enrolls

courseID	*name*	*credits*
101	Programming in C	3.8
112	Discrete Mathematics	4.0
311	Database Systems	4.3

Courses

Foreign key constraints

Foreign key

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

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

Students

sID	*cID*	*grade*
3118	101	3.0
3118	112	4.0
1423	311	4.5

Enrolls

courseID	*name*	*credits*
101	Programming in C	3.8
112	Discrete Mathematics	4.0
311	Database Systems	4.3

Courses

Foreign key constraints

Foreign key

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

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

Students

referenced relation

sID	cID	grade
3118	101	3.0
3118	112	4.0
1423	311	4.5

Enrolls

referencing relation

courseID	name	credits
101	Programming in C	3.8
112	Discrete Mathematics	4.0
311	Database Systems	4.3

Courses

Foreign key constraints

Foreign key

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

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

Students

sID	***cID***	***grade***
3118	101	3.0
3118	112	4.0
1423	311	4.5

Enrolls

referencing relation

courseID	***name***	***credits***
101	Programming in C	3.8
112	Discrete Mathematics	4.0
311	Database Systems	4.3

Courses

referenced relation

Foreign key constraints

Foreign key

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

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

Students
referenced relation

sID	*cID*	*grade*
3118	101	3.0
3118	112	4.0
1423	311	4.5

Enrolls
referencing relation

courseID	*name*	*credits*
101	Programming in C	3.8
112	Discrete Mathematics	4.0
311	Database Systems	4.3

Courses
referenced relation

Foreign key constraints

Foreign key

- A subset of attributes in a relation is a foreign key if it *refers to the primary key of a second relation*
- Foreign key constraints
  - Each foreign key value in referencing relation must either
    - *Appear as primary key* value in referenced relation, *or*
    - Be a `null` value
  - Referencing & referenced relations could be the same relation
  - Also called referential integrity constraints

Foreign key constraints

Foreign key

- A subset of attributes in a relation is a foreign key if it *refers to the primary key of a second relation*
- Foreign key constraints
  - Each foreign key value in referencing relation must either
    - *Appear as primary key* value in referenced relation, *or*
    - Be a `null` value ← **(why?)**
  - Referencing & referenced relations could be the same relation
  - Also called referential integrity constraints

- Relational data model
- Integrity constraints
  - Key constraints
  - Foreign key constraints
- Relational algebra
  - Unary operators
  - Binary operators
  - Closure properties

Integrity constraints

Relational algebra

What is it?

- A formal language for asking queries on relations

Basic

- A query is composed of a collection of operators
  - Relational operators
- Each operator takes one/two relations as input and computes an output relation
- Basic relational algebra operators
  - **Unary operators** (*input: <u>one relation</u>*)
    - Selection σ; Projection π; Renaming ρ
  - **Binary operators** (*input: <u>two relations</u>*)
    - Cross-product $\times$; Union $\cup$; Intersection $\cap$; Difference $-$

Relational algebra

Example database

- Consider a database consisting of the following 6 relations
  - `Pizzas (pizza)`
  - `Contains (pizza, ingredient)`
  - `Restaurants (rname, area)`
  - `Sells (rname, pizza, price)`
  - `Customers (cname, area)`
  - `Likes (cname, pizza)`
- Foreign key constraints:
  - `Contains.pizza` is referencing `Pizzas.pizza`
  - `Sells.rname` is referencing `Restaurants.rname`
  - `Sells.pizza` is referencing `Pizzas.pizza`
  - `Likes.cname` is referencing `Customers.cname`
  - `Likes.pizza` is referencing `Pizzas.pizza`

Relational algebra

Example database

Pizzas

pizza
Diavola
Funghi
Hawaiian
Margherita
Marinara
Siciliana

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

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

Likes

cname	*pizza*
Homer	Hawaiian
Homer	Margherita
Lisa	Funghi
Maggie	Funghi
Moe	Funghi
Moe	Siciliana
Ralph	Diavola

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

Unary operators

Selection: σ_c

- $\sigma_c(R)$ selects tuples from relation R that satisfies selection condition c
  - Selection condition is a boolean combination of *terms*
  - A term is one of the following forms:

1. attribute **op** constant — ***op*** $\in \{=, \neq, <, \leq, >, \geq\}$
2. attribute_1 **op** attribute_2
3. $\text{term}_1 \wedge \text{term}_2$ — *Conjunction (and)*
4. $\text{term}_1 \vee \text{term}_2$ — *Disjunction (or)*
5. $\neg$ term — *Negation (not)*
6. (term)

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

Unary operators

Selection: σ_c

- $\sigma_c(R)$ selects tuples from relation R that satisfies selection condition c

➢ *Selection removes rows*

➢ *Better known as filter*

Table

attr1	***attr2***	***Attr3***
Val1_1	Val1_2	Val1_3
Val2_1	Val2_2	Val2_3
Val3_1	Val3_2	Val3_3
Val4_1	Val4_2	Val4_3
Val5_1	Val5_2	Val5_3
Val6_1	Val6_2	Val6_3
:	:	:

Unary operators

Selection: $\boldsymbol{\sigma_c}$

- $\sigma_c(R)$ selects tuples from relation R that satisfies selection condition c

➢ ***Selection removes rows***

➢ ***Better known as filter***

Table

attr1	***attr2***	***Attr3***
Val1_1	Val1_2	Val1_3
Val2_1	Val2_2	Val2_3
Val3_1	Val3_2	Val3_3
Val4_1	Val4_2	Val4_3
Val5_1	Val5_2	Val5_3
Val6_1	Val6_2	Val6_3
:	:	:

satisfy c?

Table

attr1	***attr2***	***Attr3***
Val1_1	Val1_2	Val1_3
~~Val2_1~~	~~Val2_2~~	~~Val2_3~~
~~Val3_1~~	~~Val3_2~~	~~Val3_3~~
Val4_1	Val4_2	Val4_3
Val5_1	Val5_2	Val5_3
~~Val6_1~~	~~Val6_2~~	~~Val6_3~~
:	:	:

Unary operators

Selection: σ_c

- $\sigma_c(R)$ selects tuples from relation R that satisfies selection condition c
   - *Selection removes rows*
   - *Better known as filter*

Table

attr1	*attr2*	*Attr3*
Val1_1	Val1_2	Val1_3
Val2_1	Val2_2	Val2_3
Val3_1	Val3_2	Val3_3
Val4_1	Val4_2	Val4_3
Val5_1	Val5_2	Val5_3
Val6_1	Val6_2	Val6_3
:	:	:

satisfy c?

Table

attr1	*attr2*	*Attr3*
Val1_1	Val1_2	Val1_3
~~Val2_1~~	~~Val2_2~~	~~Val2_3~~
~~Val3_1~~	~~Val3_2~~	~~Val3_3~~
Val4_1	Val4_2	Val4_3
Val5_1	Val5_2	Val5_3
~~Val6_1~~	~~Val6_2~~	~~Val6_3~~
:	:	:

remove

Table

attr1	*attr2*	*Attr3*
Val1_1	Val1_2	Val1_3
Val4_1	Val4_2	Val4_3
Val5_1	Val5_2	Val5_3
:	:	:
:	:	:
:	:	:
:	:	:

Unary operators

Selection: $\boldsymbol{\sigma_c}$

- $\sigma_c(R)$ selects tuples from relation R that satisfies selection condition c

Example: Find all restaurants, the pizzas that they sell, and their prices, where the price is under $20

Unary operators

Selection: σ_c

- $\sigma_c(R)$ selects tuples from relation R that satisfies selection condition c

Example: Find all restaurants, the pizzas that they sell, and their prices, where the price is under $20

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

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

Unary operators

Selection: σ_c

- $\sigma_c(R)$ selects tuples from relation R that satisfies selection condition c

Example: Find all restaurants, the pizzas that they sell, and their prices, where the price is under $20

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

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

price under $20?

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

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~~

Unary operators

Selection: σ_c

- $\sigma_c(R)$ selects tuples from relation R that satisfies selection condition c

Example: Find all restaurants, the pizzas that they sell, and their prices, where the price is under $20

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

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

Unary operators

Selection: σ_c

- $\sigma_c(R)$ selects tuples from relation R that satisfies selection condition c

Example: Find all restaurants, the pizzas that they sell, and their prices, where (1) either the price is under $20 or the pizza is “Marinara”, and (2) the pizza is not “Diavola”

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

rname	*pizza*	*price*
Corleone Corner	Margherita	19
Gambino Oven	Siciliana	16
Mamma’s Place	Marinara	22

Unary operators

Selection: σ_c

- $\sigma_c(R)$ selects tuples from relation R that satisfies selection condition c

Example: Find all restaurants, the pizzas that they sell, and their prices, where (1) either the price is under \$20 or the pizza is "Marinara", and (2) the pizza is not "Diavola"

- $\sigma_{(\text{price}<20\ \vee \text{pizza}=\text{"Marinara"})\ \vee (\text{pizza}\neq\text{"Diavola"})}(\text{Sells})$

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

$\sigma_{(\text{price}<20\ \vee \text{pizza}=\text{"Marinara"})\ \vee (\text{pizza}\neq\text{"Diavola"})}(\text{Sells})$

rname	*pizza*	*price*
Corleone Corner	Margherita	19
Gambino Oven	Siciliana	16
Mamma's Place	Marinara	22

Unary operators

Projection: π_ℓ

- $\pi_\ell(R)$ projects attributes given by a list ℓ of attributes from relation R

➢ *Projection removes columns*

➢ *Duplicate records are removed in the output relation*

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

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

Unary operators

Projection: π_ℓ

- $\pi_\ell(R)$ projects attributes given by a list ℓ of attributes from relation R

➢ *Projection removes columns*

➢ *Duplicate records are removed in the output relation*

Example: Find all restaurants and the pizzas that they sell

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

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

Unary operators

Projection: π_ℓ

- $\pi_\ell(R)$ projects attributes given by a list ℓ of attributes from relation R

➢ *Projection removes/rearranges columns*

➢ *Duplicate records are removed in the output relation*

Example: Find all restaurants area

Restaurants

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

area
North
Central
South
East

Unary operators

Projection: π_ℓ

- $\pi_\ell(R)$ projects attributes given by a list ℓ of attributes from relation R
  - *Projection removes/rearranges columns*
  - *Duplicate records are removed in the output relation*

Example: Find all restaurants area

- $\pi_{\text{area}}(\text{Restaurants})$

Restaurants

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

$\pi_{\text{area}}(\text{Restaurants})$

area
North
Central
South
East

Unary operators

Renaming: $\boldsymbol{\rho_{S(B_1,B_2,\ldots,B_n)}(R)}$

- $\rho_{S(B_1,B_2,\ldots,B_n)}(R)$ renames $R(A_1, A_2, \ldots, A_n)$ to $S(B_1, B_2, \ldots, B_n)$
  - When the attributes are not renamed $\boldsymbol{\rho_S(R)}$
  - When the table is not renamed $\boldsymbol{\rho_{(B_1,B_2,\ldots,B_n)}(R)}$

Example: Rename `Restaurants(rname,area)` to `Shops(sname,region)`

Restaurants

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

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

Unary operators

Renaming: $\boldsymbol{\rho_{S(B_1,B_2,\ldots,B_n)}(R)}$

- $\rho_{S(B_1,B_2,\ldots,B_n)}(R)$ renames $R(A_1, A_2, \ldots, A_n)$ to $S(B_1, B_2, \ldots, B_n)$
  - When the attributes are not renamed $\boldsymbol{\rho_S(R)}$
  - When the table is not renamed $\boldsymbol{\rho_{(B_1,B_2,\ldots,B_n)}(R)}$

Example: Rename `Restaurants(rname,area)` to `Shops(sname,region)`

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

Restaurants

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

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

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

Union compatibility

Definition: Two relations R_1 and R_2 are union compatible if

1. They have the same number of attributes, and
2. The corresponding attributes have the same domains

- The schema of the result of $R_1 \oplus R_2$ where $\oplus$ are binary operator requiring union compatibility is ***identical*** to the schema of R_1 and R_2 respectively

Union compatibility

Definition: Two relations R_1 and R_2 are union compatible if

1. They have the same number of attributes, and
2. The corresponding attributes have the same domains

- The schema of the result of $R_1 \oplus R_2$ where $\oplus$ are binary operator requiring union compatibility is *identical* to the schema of R_1 and R_2 respectively

Example: Consider the following database

- `Student (`***`sid`***`: integer, `***`dob`***`: date, `***`name`***`: string)`
- `GradStudent (`***`sid`***`: integer, `***`name`***`: string)`
- `Report (`***`reportID`***`: integer, `***`title`***`: string,`
  `pubdate``: date)`

Questions:

- Which of the following are *valid* binary operations?

1. Student ∪ GradStudent
2. $\pi_{\text{sid,name}}(\text{Student})$ ∪ GradStudent
3. Student ∩ Report
4. $\pi_{\text{sid,name,dob}}(\text{Student})$ − Report

Union compatibility

Definition: Two relations R_1 and R_2 are union compatible if

1. They have the same number of attributes, and
2. The corresponding attributes have the same domains

- The schema of the result of $R_1 \oplus R_2$ where $\oplus$ are binary operator requiring union compatibility is *identical* to the schema of R_1 and R_2 respectively

Example: Consider the following database

- `Student (`***sid***`: integer, `***dob***`: date, `***name***`: string)`
- `GradStudent (`***sid***`: integer, `***name***`: string)`
- `Report (`***reportID***`: integer, `***title***`: string,`
  pubdate`: date)`

Questions:

- Which of the following are *valid* binary operations?

1. ~~Student ∪ GradStudent~~
2. $\pi_{\text{sid,name}}(\text{Student}) \cup \text{GradStudent}$
3. ~~Student ∩ Report~~
4. $\pi_{\text{sid,name,dob}}(\text{Student}) - \text{Report}$

Binary operators

Union: $\boldsymbol{R} \cup \boldsymbol{S}$

- Returns a relation containing all tuples that occur in R, S, or both

Intersection: $\boldsymbol{R} \cap \boldsymbol{S}$

- Returns a relation containing all tuples that occur in both R and S

Set-difference: $\boldsymbol{R} - \boldsymbol{S}$

- Returns a relation containing all tuples that occur in R but not in S

- Union ($\cup$), intersection ($\cap$), and set-difference ($-$) operators require input relations to be union compatible

Binary operators

Union: $R \cup S$

- Returns a relation containing all tuples that occur in R, S, or both

Example:

- Find all customer/restaurant names
- **Solution:**

Restaurants

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

Customers

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

rname
Corleone Corner
Gambino Oven
Lorenzo Tavern
Mamma's Place
Pizza King
Homer
Lisa
Maggie
Moe
Ralph
Willie

Binary operators

Union: $\boldsymbol{R} \cup \boldsymbol{S}$

- Returns a relation containing all tuples that occur in R, S, or both

Example:

- Find all customer/restaurant names
- **Solution:** $\pi_{\text{rname}}(\text{Restaurants}) \cup \pi_{\text{cname}}(\text{Customers})$

Restaurants

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

Customers

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

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

rname
Corleone Corner
Gambino Oven
Lorenzo Tavern
Mamma's Place
Pizza King
Homer
Lisa
Maggie
Moe
Ralph
Willie

Binary operators

Intersection: $\boldsymbol{R \cap S}$

- Returns a relation containing all tuples that occur in both R and S

Example:

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

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

pizza
Diavola
Margherita
Siciliana

pizza
Diavola

pizza
Diavola

Binary operators

Intersection: $R \cap S$

- Returns a relation containing all tuples that occur in both R and S

Example:

- Find all pizzas that contain both cheese and chilli
- **Solution:** $\pi_{\text{pizza}}\left(\sigma_{\text{ingredient}='cheese'}(\text{Contains})\right) \cap \pi_{\text{pizza}}\left(\sigma_{\text{ingredient}='chilli'}(\text{Contains})\right)$

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

$\pi_{\text{pizza}}\left(\sigma_{\text{ingredient}='cheese'}(\text{Contains})\right)$

pizza
Diavola
Margherita
Siciliana

$\pi_{\text{pizza}}\left(\sigma_{\text{ingredient}='chilli'}(\textit{Contains})\right)$

pizza
Diavola

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

pizza
Diavola

Binary operators

Set-difference: $R - S$

- Returns a relation containing all tuples that occur in R but not in S

Example:

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

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

pizza
Diavola
Margherita
Siciliana

pizza
Margherita
Siciliana

pizza
Diavola

Binary operators

Set-difference: $R - S$

- Returns a relation containing all tuples that occur in R but not in S

Example:

- Find all pizzas that contain cheese but not chilli
- **Solution:** $\pi_{\text{pizza}}\left(\sigma_{\text{ingredient}='cheese'}(\text{Contains}) \right) - \pi_{\text{pizza}}\left(\sigma_{\text{ingredient}='chilli'}(\text{Contains}) \right)$

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

$\pi_{\text{pizza}}\left(\sigma_{\text{ingredient}='cheese'}(\text{Contains}) \right)$

pizza
Diavola
Margherita
Siciliana

$\pi_{\text{pizza}}\left(\sigma_{\text{ingredient}='chilli'}(\textit{Contains}) \right)$

pizza
Diavola

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

pizza
Margherita
Siciliana

Binary operator

Cross-product: $\times$

- Consider a relation $R_1(A, B, C)$ and $R_2(X, Y)$
- $R_1 \times R_2$ returns a relation with schema (A, B, C, X, Y) defined as follows:
  - $R_1 \times R_2 = \{(a, b, c, x, y) \mid (a, b, c) \in R_1, (x, y) \in R_2\}$
- Also known as cartesian product

Example

- Find all customer-restaurant pairs that are located in the central area
- Idea
  - Find all customers in central
  - Find all restaurants in central
  - Cross-product

Binary operator

Cross-product: $\times$

- Consider a relation $R_1(A, B, C)$ and $R_2(X, Y)$
- $R_1 \times R_2$ returns a relation with schema (A, B, C, X, Y) defined as follows:
  - $R_1 \times R_2 = \{(a, b, c, x, y) \mid (a, b, c) \in R_1, (x, y) \in R_2\}$
- Also known as cartesian product

Example

- Find all customer-restaurant pairs that are located in the central area
- Idea
  - Find all customers in central $R_1 = \pi_{\text{cname}}\left(\sigma_{\text{area}='Central'}(\text{Customers})\right)$
  - Find all restaurants in central $R_2 = \pi_{\text{rname}}\left(\sigma_{\text{area}='Central'}(\text{Restaurants})\right)$
  - Cross-product $R_1 \times R_2$

Binary operator

Cross-product: $\times$

Example

- Find all customer-restaurant pairs that are located in the central area
- Idea
  - Find all customers in central $R_1 = \pi_{\text{cname}}\left(\sigma_{\text{area}='Central'}(\text{Customers})\right)$
  - Find all restaurants in central $R_2 = \pi_{\text{rname}}\left(\sigma_{\text{area}='Central'}(\text{Restaurants})\right)$
  - Cross-product $R_1 \times R_2$

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

R_1

cname
Moe
Ralph

R_2

rname
Gambino Oven
Lorenzo Tavern

$R_1 \times R_2$

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

Closure properties

Definition

- A set S is closed under an operation $\oplus$ if for any two members of the set $x_1 \in S$ and $x_s \in S$, the result $x_1 \oplus x_2 \in S$ (i.e., the result is the member of the set S)
  - Quick examples
    - Positive integer is closed under addition (*but not subtraction*)
    - Integer is closed under addition, subtraction, and multiplication (*but not division*)

Closure of relation under unary operators

- Unary operator takes in a *relation as input* and gives a *relation as output*

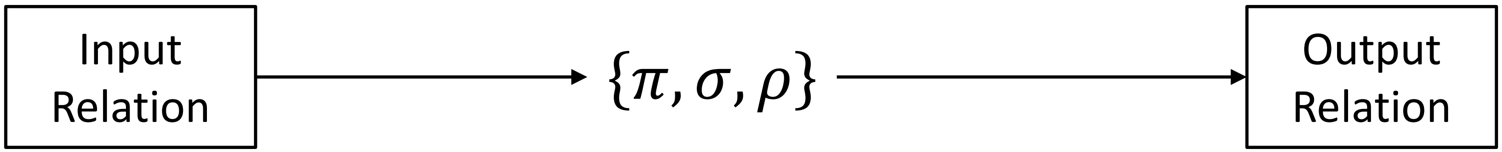
Closure of relation under binary operators

- Binary operator takes in two *relations as inputs* and gives a *relation as output*

Closure properties

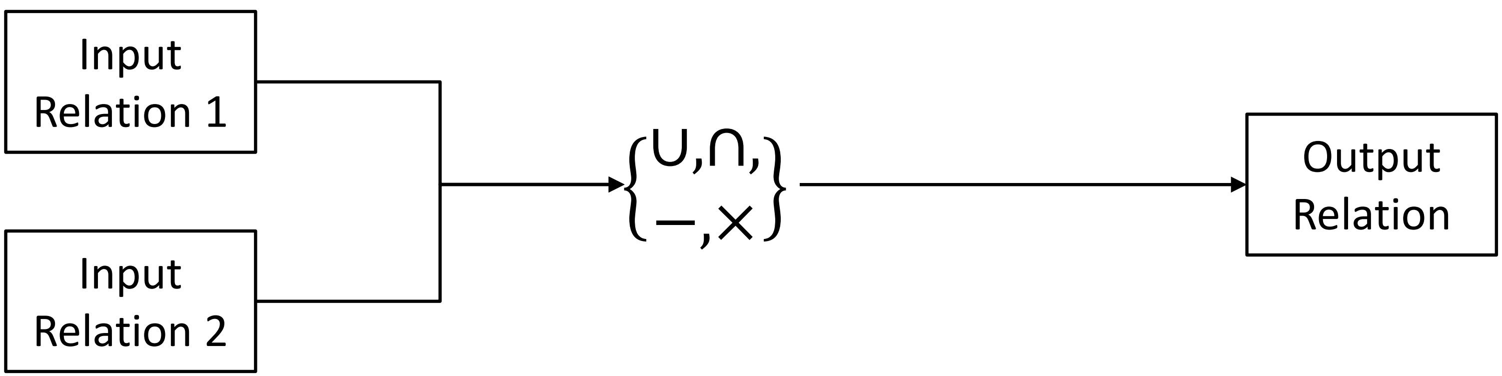
Closure of relation under unary operators *(as diagrams)*

- Unary operator takes in a *relation as input* and gives a *relation as output*



Closure of relation under binary operators *(as diagrams)*

- Binary operator takes in two *relations as inputs* and gives a *relation as output*



Closure properties

Composition

- Operators can be *composed* to form relational algebra expressions
- Examples:
  - $\pi_{\text{cname}}\left(\sigma_{\text{area}='Central'}(\text{Customers})\right)$
  - $\pi_{\text{pizza}}\left(\sigma_{\text{ingredient}='cheese'}(\text{Contains})\right)$
    $\cap$
    $\pi_{\text{pizza}}\left(\sigma_{\text{ingredient}='chilli'}(\text{Contains})\right)$
  - $\pi_{\text{cname}}\left(\sigma_{\text{area}='Central'}(\text{Customers})\right)$
    $\times$
    $\pi_{\text{rname}}\left(\sigma_{\text{area}='Central'}(\text{Restaurants})\right)$

Closure properties

Diagrams

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

Closure properties

Diagrams

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

Closure properties

Diagrams

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

Customers

Closure properties

Diagrams

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

Customers

Closure properties

Diagrams

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

$$\begin{array}{c} \sigma_{\text{area}='Central'} \\ \uparrow \\ \text{Customers} \end{array}$$

Closure properties

Diagrams

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

$$\begin{tikzcd}
\sigma_{\text{area}='Central'} \\
\text{Customers} \arrow[u]
\end{tikzcd}$$

Closure properties

Diagrams

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

  $\times$

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

$$
\begin{tikzcd}
\pi_{\text{cname}} \\
\sigma_{\text{area}='Central'} \arrow[u] \\
\text{Customers} \arrow[u]
\end{tikzcd}
$$

Closure properties

Diagrams

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

$$
\begin{tikzcd}
\pi_{\text{cname}} & & \pi_{\text{rname}} \\
\sigma_{\text{area}='Central'} \arrow[u] & & \sigma_{\text{area}='Central'} \arrow[u] \\
\text{Customers} \arrow[u] & & \text{Restaurants} \arrow[u]
\end{tikzcd}
$$

Closure properties

Diagrams

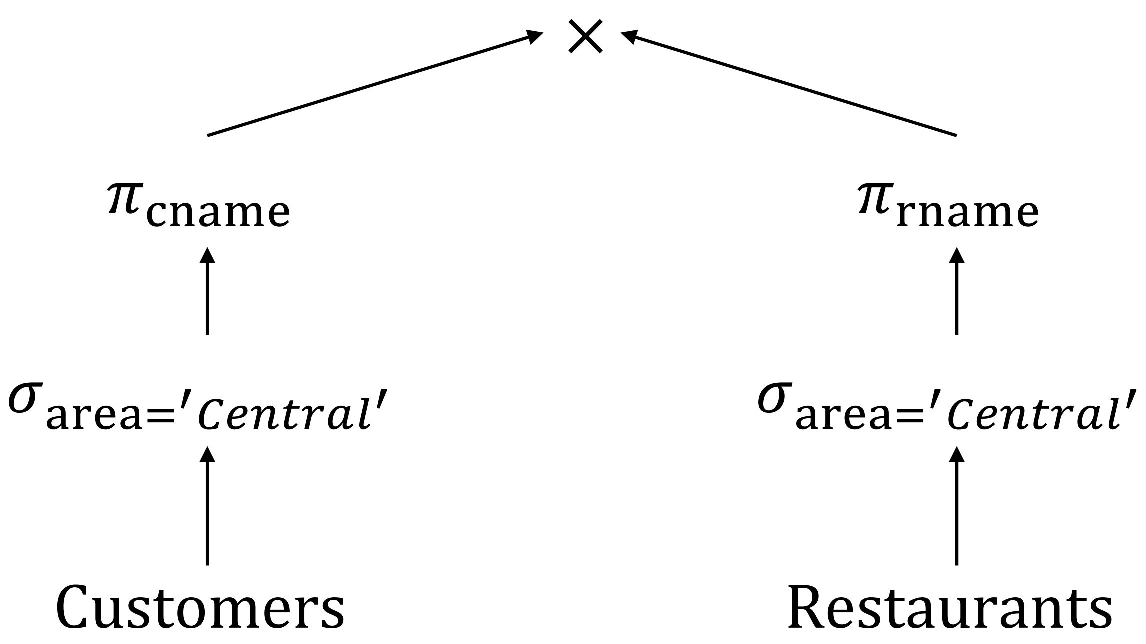
- $\pi_{\text{cname}}\left(\sigma_{\text{area}='Central'}(\text{Customers})\right)$
  $\times$
  $\pi_{\text{rname}}\left(\sigma_{\text{area}='Central'}(\text{Restaurants})\right)$

$$
\begin{tikzcd}
\pi_{\text{cname}} & & \pi_{\text{rname}} \\
\sigma_{\text{area}='Central'} \arrow[u] & & \sigma_{\text{area}='Central'} \arrow[u] \\
\text{Customers} \arrow[u] & & \text{Restaurants} \arrow[u]
\end{tikzcd}
$$

Closure properties

Diagrams

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



Relational algebra problems

Question

- Find customer pairs (C_1, C_2) such that they like some common pizza and $C_1 < C_2$ (*i.e., lexicographical order*)

Visualization

Likes

cname	*pizza*
Homer	Hawaiian
Homer	Margherita
Lisa	Funghi
Maggie	Funghi
Moe	Funghi
Moe	Siciliana
Ralph	Diavola

R

cname	*cname2*
Lisa	Maggie
Lisa	Moe
Maggie	Moe

- What is R?

Relational algebra problems

Question

- Find customer pairs (C_1, C_2) such that they like some common pizza and $C_1 < C_2$ (*i.e., lexicographical order*)

Visualization

Likes

cname	*pizza*
Homer	Hawaiian
Homer	Margherita
Lisa	Funghi
Maggie	Funghi
Moe	Funghi
Moe	Siciliana
Ralph	Diavola

R

cname	*cname2*
Lisa	Maggie
Lisa	Moe
Maggie	Moe

- What is R?
  - $R = \pi_{\text{cname,cname2}}\left(\sigma_{(\text{pizza=pizza2}) \wedge (\text{cname<cname2})}\left(\text{Likes} \times \rho_{\text{Likes2(cname2,pizza2)}}(\text{Likes})\right)\right)$
  - ***Too complicated!***

Relational algebra problems

Question

- Find customer pairs (C_1, C_2) such that they like some common pizza and $C_1 < C_2$ (*i.e., lexicographical order*)

Visualization

- What is R?
  - $R = \pi_{\text{cname,cname2}}\left(\sigma_{(\text{pizza=pizza2})\wedge(\text{cname<cname2})}\left(\text{Likes} \times \rho_{\text{Likes2(cname2,pizza2)}}(\text{Likes})\right)\right)$
  - Simplify:

Relational algebra problems

Question

- Find customer pairs (C_1, C_2) such that they like some common pizza and $C_1 < C_2$ (*i.e., lexicographical order*)

Visualization

- What is R?
  - $R = \pi_{\text{cname,cname2}}\left(\sigma_{(\text{pizza=pizza2})\wedge(\text{cname<cname2})}\left(\text{Likes} \times \rho_{\text{Likes2(cname2,pizza2)}}(\text{Likes})\right)\right)$
  - Simplify:
    - **Method 1:** draw diagram
      - *My drawing is not so good, any other method?*

Relational algebra problems

Question

- Find customer pairs (C_1, C_2) such that they like some common pizza and $C_1 < C_2$ (*i.e., lexicographical order*)

Visualization

- What is R?
  - $R = \pi_{\text{cname,cname2}}\left(\sigma_{(\text{pizza=pizza2})\wedge(\text{cname<cname2})}\left(\text{Likes} \times \rho_{\text{Likes2(cname2,pizza2)}}(\text{Likes})\right)\right)$
  - Simplify:
    - **Method 1:** draw diagram
      - *My drawing is not so good, any other method?*
    - **Method 2:** sequence of steps

Relational algebra problems

Question

- Find customer pairs (C_1, C_2) such that they like some common pizza and $C_1 < C_2$ (*i.e., lexicographical order*)

Visualization

- What is R?
  - $R = \pi_{\text{cname,cname2}}\left(\sigma_{(\text{pizza=pizza2})\wedge(\text{cname<cname2})}\left(\text{Likes} \times \rho_{\text{Likes2(cname2,pizza2)}}(\text{Likes})\right)\right)$
  - Simplify:
    - **Method 1:** draw diagram
      - *My drawing is not so good, any other method?*
    - **Method 2:** sequence of steps
      - $R_1 = \pi_{\text{cname}}\left(\sigma_{\text{area}='Central'}(\text{Customers})\right)$
      - $R_2 = \pi_{\text{rname}}\left(\sigma_{\text{area}='Central'}(\text{Restaurants})\right)$
      - $R_{answer} = R_1 \times R_2$

Relational algebra problems

Question

- Find customer pairs (C_1, C_2) such that they like some common pizza and $C_1 < C_2$ (*i.e., lexicographical order*)

Visualization

- What is R?
  - $R = \pi_{\text{cname,cname2}}\left(\sigma_{(\text{pizza=pizza2})\wedge(\text{cname<cname2})}\left(\text{Likes} \times \rho_{\text{Likes2(cname2,pizza2)}}(\text{Likes})\right)\right)$
  - Simplification using method 2

Relational algebra problems

Question

- Find customer pairs (C_1, C_2) such that they like some common pizza and $C_1 < C_2$ (*i.e., lexicographical order*)

Visualization

- What is R?
  - $R = \pi_{\text{cname,cname2}}\left(\sigma_{(\text{pizza=pizza2})\wedge(\text{cname<cname2})}\left(\text{Likes} \times \rho_{\text{Likes2(cname2,pizza2)}}(\text{Likes})\right)\right)$
  - Simplification using method 2
    - $R_1 = \text{Likes} \times \rho_{\text{Likes2(cname2,pizza2)}}(\text{Likes})$

Relational algebra problems

Question

- Find customer pairs (C_1, C_2) such that they like some common pizza and $C_1 < C_2$ (*i.e., lexicographical order*)

Visualization

- What is R?
  - $R = \pi_{\text{cname,cname2}}\left(\sigma_{(\text{pizza=pizza2})\wedge(\text{cname<cname2})}\left(\qquad R_1 \qquad \right)\right)$
  - Simplification using method 2
    - $R_1 = \text{Likes} \times \rho_{\text{Likes2(cname2,pizza2)}}(\text{Likes})$

Relational algebra problems

Question

- Find customer pairs (C_1, C_2) such that they like some common pizza and $C_1 < C_2$ (*i.e., lexicographical order*)

Visualization

- What is R?
  - $R = \pi_{\text{cname,cname2}}\left(\sigma_{(\text{pizza=pizza2})\wedge(\text{cname<cname2})}\left(R_1 \right)\right)$
  - Simplification using method 2
    - $R_1 = \text{Likes} \times \rho_{\text{Likes2(cname2,pizza2)}}(\text{Likes})$
    - $R_2 = \sigma_{(\text{pizza=pizza2})\wedge(\text{cname<cname2})}(R_1)$

Relational algebra problems

Question

- Find customer pairs (C_1, C_2) such that they like some common pizza and $C_1 < C_2$ (*i.e., lexicographical order*)

Visualization

- What is R?
  - $R = \pi_{\text{cname,cname2}}\left(\qquad R_2 \qquad \right)$
  - Simplification using method 2
    - $R_1 = \text{Likes} \times \rho_{\text{Likes2(cname2,pizza2)}}(\text{Likes})$
    - $R_2 = \sigma_{(\text{pizza=pizza2}) \wedge (\text{cname<cname2})}(R_1)$

Relational algebra problems

Question

- Find customer pairs (C_1, C_2) such that they like some common pizza and $C_1 < C_2$ (*i.e., lexicographical order*)

Visualization

- What is R?
  - $R = \pi_{\text{cname,cname2}}\left(\qquad R_2 \qquad \right)$
  - Simplification using method 2
    - $R_1 = \text{Likes} \times \rho_{\text{Likes2(cname2,pizza2)}}(\text{Likes})$
    - $R_2 = \sigma_{(\text{pizza=pizza2}) \wedge (\text{cname<cname2})}(R_1)$
    - $R = \pi_{\text{cname,cname2}}(R_2)$

Relational algebra problems

Question

- Find customer pairs (C_1, C_2) such that they like some common pizza and $C_1 < C_2$ (*i.e., lexicographical order*)

Visualization

- What is R?
  - $R = \pi_{\text{cname,cname2}}\left(\sigma_{(\text{pizza=pizza2})\wedge(\text{cname<cname2})}\left(\text{Likes} \times \rho_{\text{Likes2(cname2,pizza2)}}(\text{Likes})\right)\right)$
  - Simplification using method 2
    - $R_1 = \text{Likes} \times \rho_{\text{Likes2(cname2,pizza2)}}(\text{Likes})$
    - $R_2 = \sigma_{(\text{pizza=pizza2})\wedge(\text{cname<cname2})}(R_1)$
    - $R = \pi_{\text{cname,cname2}}(R_2)$

Relational algebra problems

Computation

- $R_1(\text{cname}, \text{pizza}, \text{cname2}, \text{pizza2}) = \text{Likes} \times \rho_{\text{Likes2(cname2,pizza2)}}(\text{Likes})$
- $R_2(\text{cname}, \text{pizza}, \text{cname2}, \text{pizza2}) = \sigma_{(\text{pizza}=\text{pizza2}) \wedge (\text{cname}<\text{cname2})}(R_1)$
- $R(\text{cname}, \text{cname2}) = \pi_{\text{cname,cname2}}(R_2)$

Likes

cname	*pizza*
Homer	Hawaiian
Homer	Margherita
Lisa	Funghi
Maggie	Funghi
Moe	Funghi
Moe	Siciliana
Ralph	Diavola

R_1

cname	*pizza*	*cname2*	*pizza2*
Homer	Hawaiian	Homer	Hawaiian
Homer	Hawaiian	Homer	Margherita
...	...	...	...
Lisa	Funghi	Maggie	Funghi
Lisa	Funghi	Moe	Funghi
...	...	...	...
Maggie	Funghi	Moe	Funghi
...	...	...	...
Ralph	Diavola	Moe	Siciliana
Ralph	Diavola	Ralph	Diavola

R

cname	*cname2*
Lisa	Maggie
Lisa	Moe
Maggie	Moe

Relational algebra problems

Computation

- $R_1(\text{cname}, \text{pizza}, \text{cname2}, \text{pizza2}) = \text{Likes} \times \rho_{\text{Likes2(cname2,pizza2)}}(\text{Likes})$
- $R_2(\text{cname}, \text{pizza}, \text{cname2}, \text{pizza2}) = \sigma_{(\text{pizza=pizza2}) \wedge (\text{cname<cname2})}(R_1)$
- $R(\text{cname}, \text{cname2}) = \pi_{\text{cname,cname2}}(R_2)$

Likes

cname	*pizza*
Homer	Hawaiian
Homer	Margherita
Lisa	Funghi
Maggie	Funghi
Moe	Funghi
Moe	Siciliana
Ralph	Diavola

R_1

cname	*pizza*	*cname2*	*pizza2*
Homer	Hawaiian	Homer	Hawaiian
Homer	Hawaiian	Homer	Margherita
...	...	...	...
Lisa	Funghi	Maggie	Funghi
Lisa	Funghi	Moe	Funghi
...	...	...	...
Maggie	Funghi	Moe	Funghi
...	...	...	...
Ralph	Diavola	Moe	Siciliana
Ralph	Diavola	Ralph	Diavola

49 rows!

R

cname	*cname2*
Lisa	Maggie
Lisa	Moe
Maggie	Moe

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
    - Selection σ
    - Projection π
    - Renaming ρ
    - Union $\cup$
    - Intersection $\cap$
    - Set-difference $-$
    - Cross-product $\times$