# Unit III: Relational Model
# Part-5

Dr. Anushree Tripathi

Department of Computer Science and Engineering

National Institute of Technology Patna (NITP)

20th September, 2024

# Overview

- Tuple relational calculus
- Domain relational calculus
- Extended relational-algebra operations

# Tuple relational calculus

- Nonprocedural query language

In tuple relational calculus, a query is expressed as:

$$\{t \mid P(t)\}$$

***t***

Tuple variable

Resulting tuples

Set of all tuples t such that predicate P is true for t

$t \in r$ denotes that tuple $t$ is in relation $r$

***P*(*t*)**

Formula similar to predicate calculus

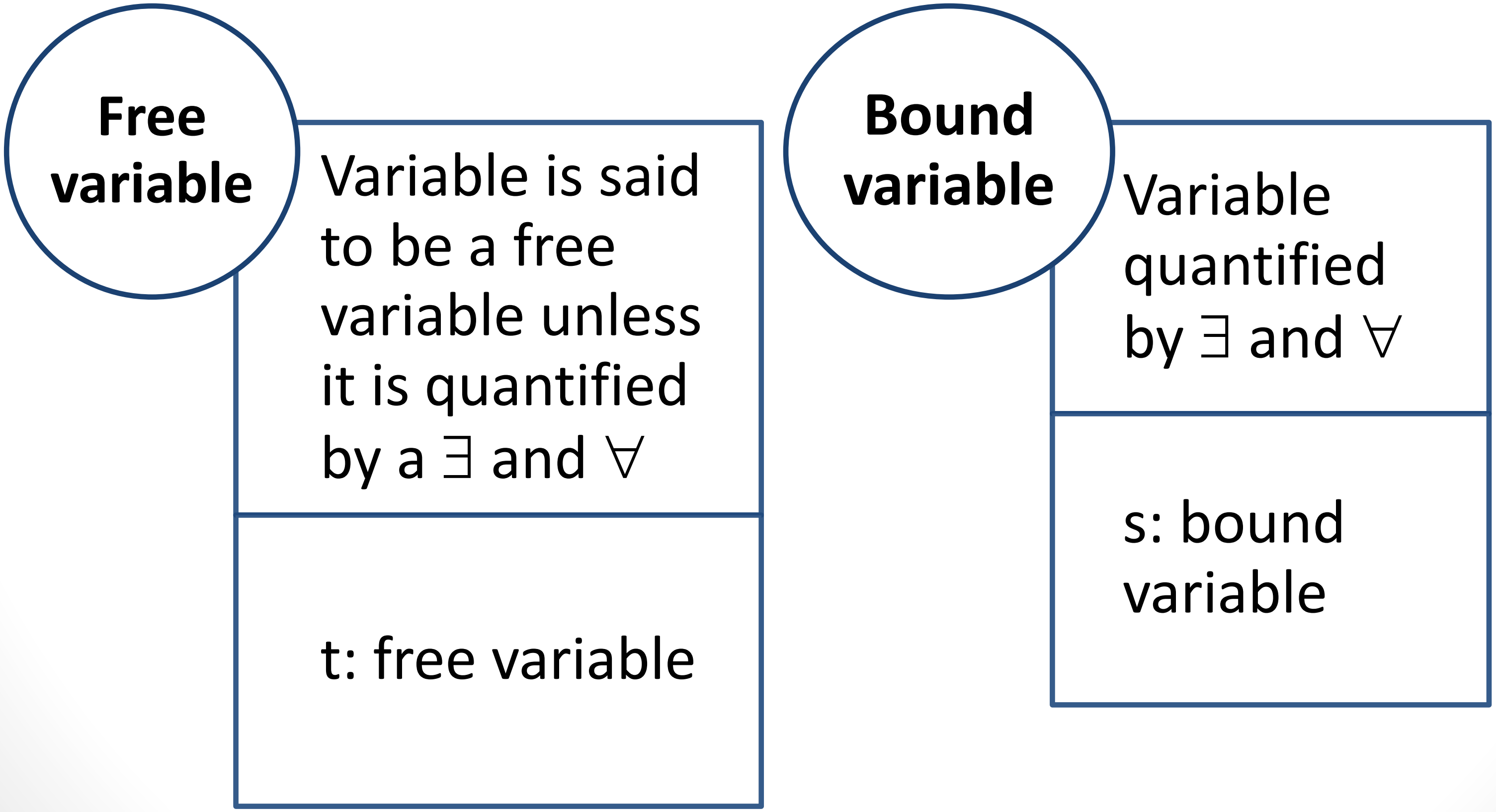
# Tuple relational calculus
## *Operations*

- Set of comparison operators: (e.g., $<$, $\leq$, $=$, $\neq$, $>$, $\geq$)
- Set of connectives: and ($\wedge$), or (v), not ($\neg$)
- Set of quantifiers
  - $\exists\ t \in r\ (Q(t)) \equiv$ ”there exists” a tuple in *t* in relation *r* such that predicate *Q(t)* is true
  - $\forall t \in r\ (Q(t)) \equiv Q$ is true “for all” tuples *t* in relation *r*

# Tuple relational calculus

## *Formal definition*

- Tuple relational calculus: $\{t \mid P(t)\}$
- $t \in loan \land \exists\, s \in customer\ (t[branch\text{-}name] = s[branch\text{-}name])$



**Free variable**

Variable is said to be a free variable unless it is quantified by a ∃ and ∀

t: free variable

**Bound variable**

Variable quantified by ∃ and ∀

s: bound variable

# Tuple relational calculus
## *Formal definition*

- Tuple relational calculus composed Atom has one of the following forms:

- $s \in r$, where $s$ is a tuple variable and $r$ is a relation
- $s[x]\ \Theta\ u[y]$, where $s$ and $u$ are tuple variables, $x$ is an attribute on which $s$ is defined, $y$ is an attribute on which $u$ is defined, and $\Theta$ is a comparison operator $(, \geq)$; we require that attributes $x$ and $y$ have domains whose members can be compared by $\Theta$
- $s[x]\ \Theta\ c$, where $s$ is a tuple variable, $x$ is an attribute on which $s$ is defined, $\Theta$ is a comparison operator, and $c$ is a constant in the domain of attribute $x$

# Tuple relational calculus
## *Example queries* (∃ )

- Find the ID, name, dept_name, salary for instructors whose salary is greater than $80,000

  {t | t ∈ instructor ∧ t[salary] > 80000}

- Find the instructor ID for each instructor with a salary greater than $80,000

  {t | ∃ s ∈ instructor (t[ID] = s[ID] ∧ s[salary] > 80000)}

# Tuple relational calculus

## *Safety of expressions*

### Issue need to be addressed

- Tuple-relational-calculus expression may generate an infinite relation

### Example

- {t | $\neg$ *t* $\in$ *r*} results in an infinite relation if the domain of any attribute of relation *r* is infinite

{t |¬ (t ∈ instructor )}

# Tuple relational calculus

## *Expressive power of languages*

Equivalent in expressive power to basic relational algebra

Every relational-algebra expression using basic operations, equivalent expression in tuple relational calculus and vice-versa

# Domain Relational Calculus

- Uses domain variables that take on values from an attributes domain, rather than values of entire tuple
- Theoretical basis of QBE (Query By Example) language

# Domain Relational Calculus

- Expression in Domain relational calculus

$$\{< x_1,\ x_2, \ldots, x_n > \ | \ P(x_1,\ x_2, \ldots, x_n)\}$$

- $x_1$, $x_2$,..., $x_n$ : Domain variables
- P: Represents formula composed of atoms
- Atom in domain relational calculus has one of the following forms:

➢ $< x_1, x_2, \ldots, x_n > \in r$

➢ x Θ y; Θ: comparison operator (e.g., $<, \leq, =, \neq, >, \geq$)

➢ x Θ c; c: constant

x and y are domain variables

# Domain Relational Calculus

## *Example queries*

- Find the instructor ID, name, dept_name, and salary for instructors whose salary is greater than $80,000

  $\{< i, n, d, s > \mid < i, n, d, s > \in instructor \wedge s > 80000\}$

# Domain Relational Calculus

## *Safety of expressions*

- {< i, n, d,s > | < i, n, d,s > ∈ instructor ∧ s > 80000}

➢ Unsafe expression: Values not in domain

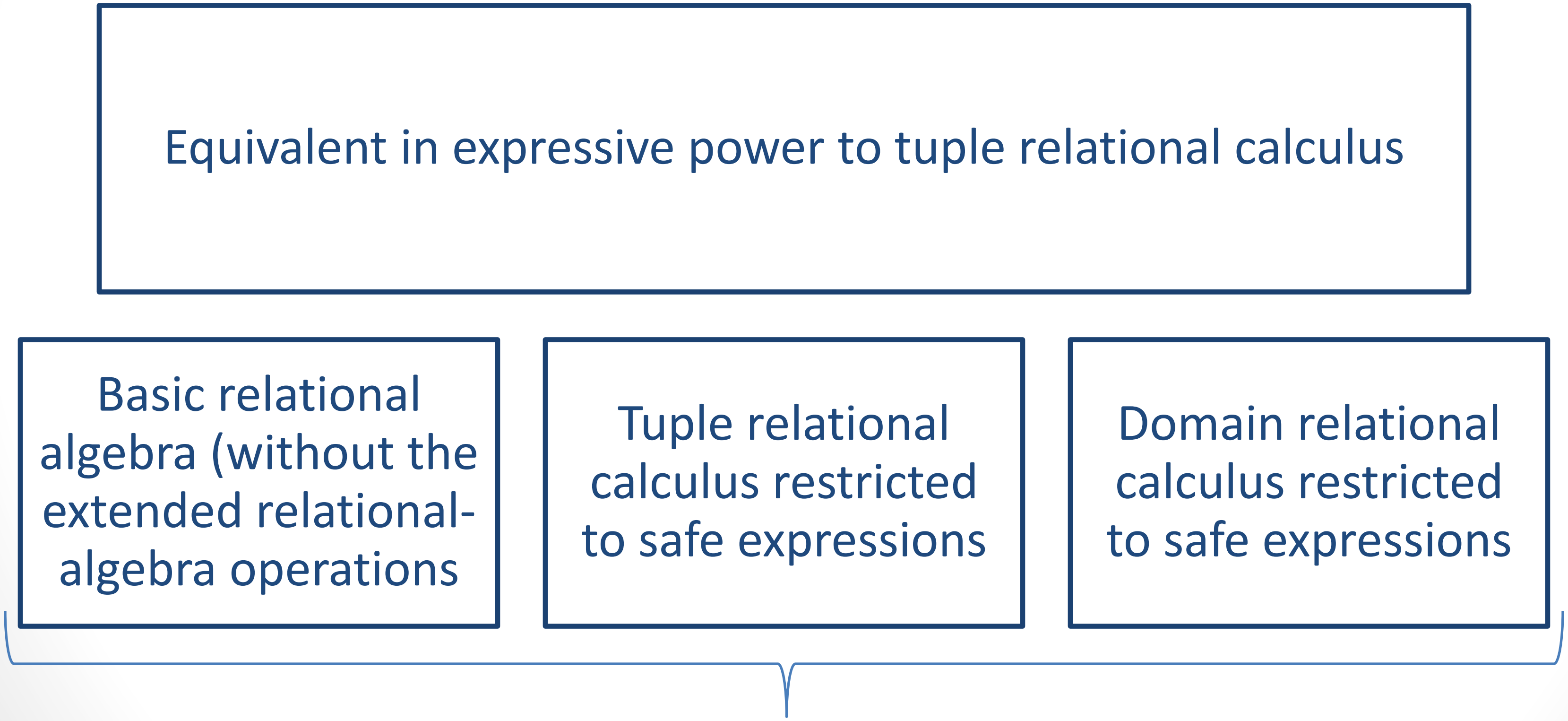
$\{ < x_1, x_2, \ldots, x_n > \mid P(x_1, x_2, \ldots, x_n)\}$

is safe if all of the following hold:

1. All values that appear in tuples of the expression are values from *dom*(*P*) (that is, the values appear either in *P* or in a tuple of a relation mentioned in *P*).
2. For every "there exists" subformula of the form $\exists\, x\, (P_1(x))$, the subformula is true if an only if $P_1(x)$ is true for all values *x* from $dom(P_1)$.
3. For every "for all" subformula of the form $\forall_x\ (P_1\ (x))$, the subformula is true if and only if $P_1(x)$ is true for all values *x* from $dom\ (P_1)$.

# Domain Relational Calculus

## *Expressive power of languages*



Equivalent in expressive power to tuple relational calculus

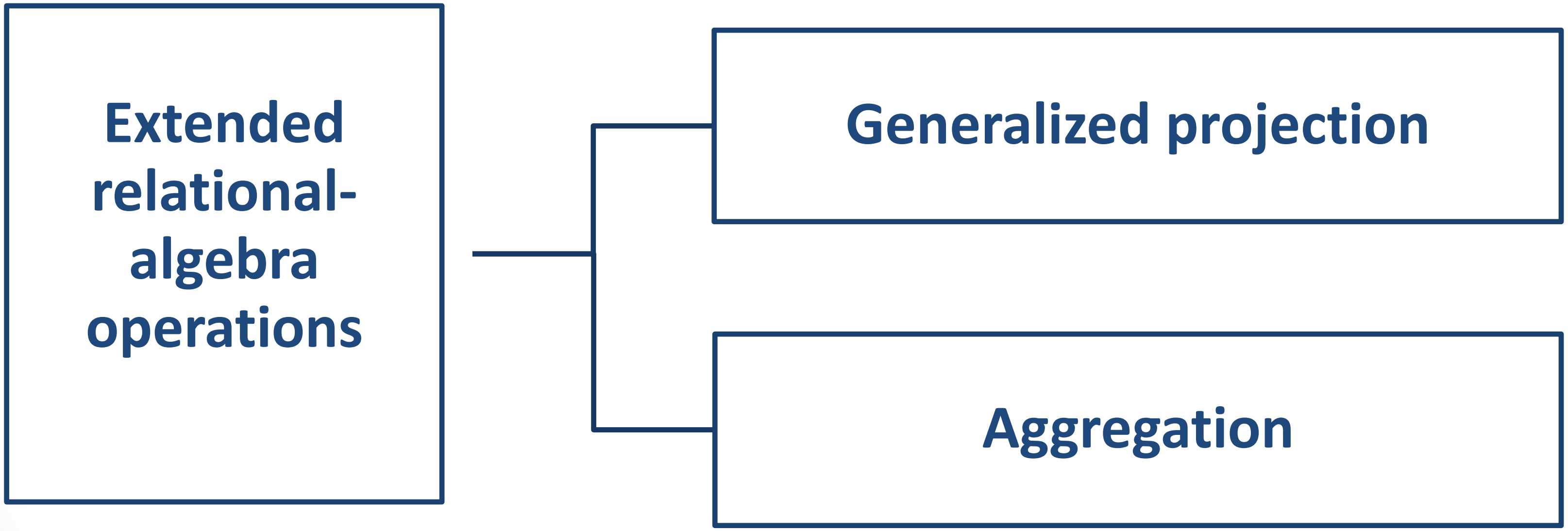
Basic relational algebra (without the extended relational-algebra operations

Tuple relational calculus restricted to safe expressions

Domain relational calculus restricted to safe expressions

Equivalent expressive power of languages

# Extended relational-algebra operations



Extended relational-algebra operations

Generalized projection

Aggregation

# Extended relational-algebra operations

## *Generalized projection*

- Extends the projection operation
- Allows arithmetic and string functions to be used in projection list
- General form:

$\prod_{F1,F2,\ldots,Fn}$ (E)

E: Any relational-algebra expression

$F_1, F_2, \ldots, F_n$: Arithmetic expression involving constants and attributes in the schema of E

- Permits operations on other data types such as concatenation of strings
- Example: $\prod_{ID,name,\ dept\ name,salary \div 12}$(instructor)

# Extended relational-algebra operations

## *Aggregation operation*

- Collection of values and return a single value as a result
- For example: sum takes a collection of values and returns the sum of values

  Sum = {1, 1, 3, 4, 4, 11} = 24

  Average = 4

  Count = 6

  Min = 1

  Max = 11
- Permits the use of aggregate functions such as min or average, on sets of values

# Extended relational-algebra operations
## *Aggregation operation*

- Query: To find out the sum of salaries of all instructors

Relational-algebra expression:

$\varsigma_{\text{sum(salary)}}$(instructor)

$\varsigma$: Signifies aggregation is to be applied

- Query: Find the total number of instructors who teach a course in the Spring 2010 semester

Count-distinct

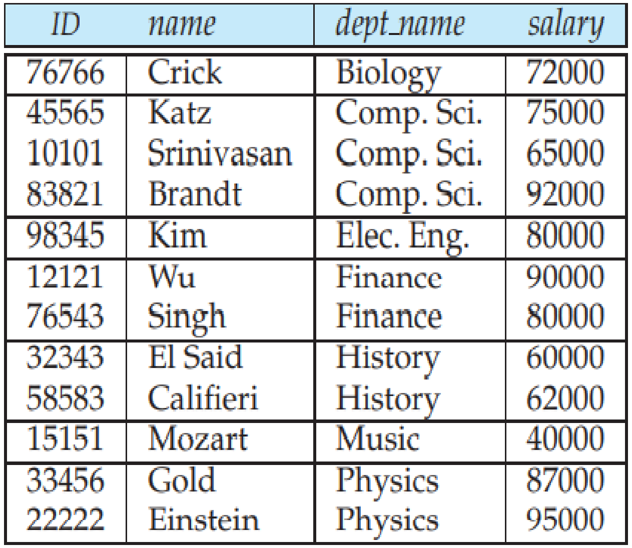
$\varsigma_{\text{count−distinct(ID)}}(\sigma_{\text{semester=“Spring”}\wedge \text{year =2010}}(\text{teaches}))$

# Extended relational-algebra operations

## *Aggregation operation*

- Find the average salary in each department

${}_{\text{dept name}}\,ς_{\text{average(salary)}}$(instructor)



| ID | name | dept_name | salary |
|---|---|---|---|
| 76766 | Crick | Biology | 72000 |
| 45565 | Katz | Comp. Sci. | 75000 |
| 10101 | Srinivasan | Comp. Sci. | 65000 |
| 83821 | Brandt | Comp. Sci. | 92000 |
| 98345 | Kim | Elec. Eng. | 80000 |
| 12121 | Wu | Finance | 90000 |
| 76543 | Singh | Finance | 80000 |
| 32343 | El Said | History | 60000 |
| 58583 | Califieri | History | 62000 |
| 15151 | Mozart | Music | 40000 |
| 33456 | Gold | Physics | 87000 |
| 22222 | Einstein | Physics | 95000 |

Tuples of the instructor relation, grouped by the dept_name attribute

- Query: Find the average salary of all instructors

$ς_{\text{average(salary)}}$(instructor)

# Extended relational-algebra operations

## *Aggregation operation*

General form of **Aggregation operation** ς

- E: ${}_{G_1,G_2,\ldots,G_n}\mathcal{G}_{F_1(A_1),\ F_2(A_2),\ldots,\ F_m(A_m)}(E)$
- $G_1$,$G_2$,…,$G_n$ : Constitute a list of attributes on which to group
- $F_i$ : Aggregate function
- $A_i$ : Attribute name