CSC 261/461 – Database Systems
Lecture 13

Spring 2017
MW 3:25 pm – 4:40 pm
January 18 – May 3
Dewey 1101

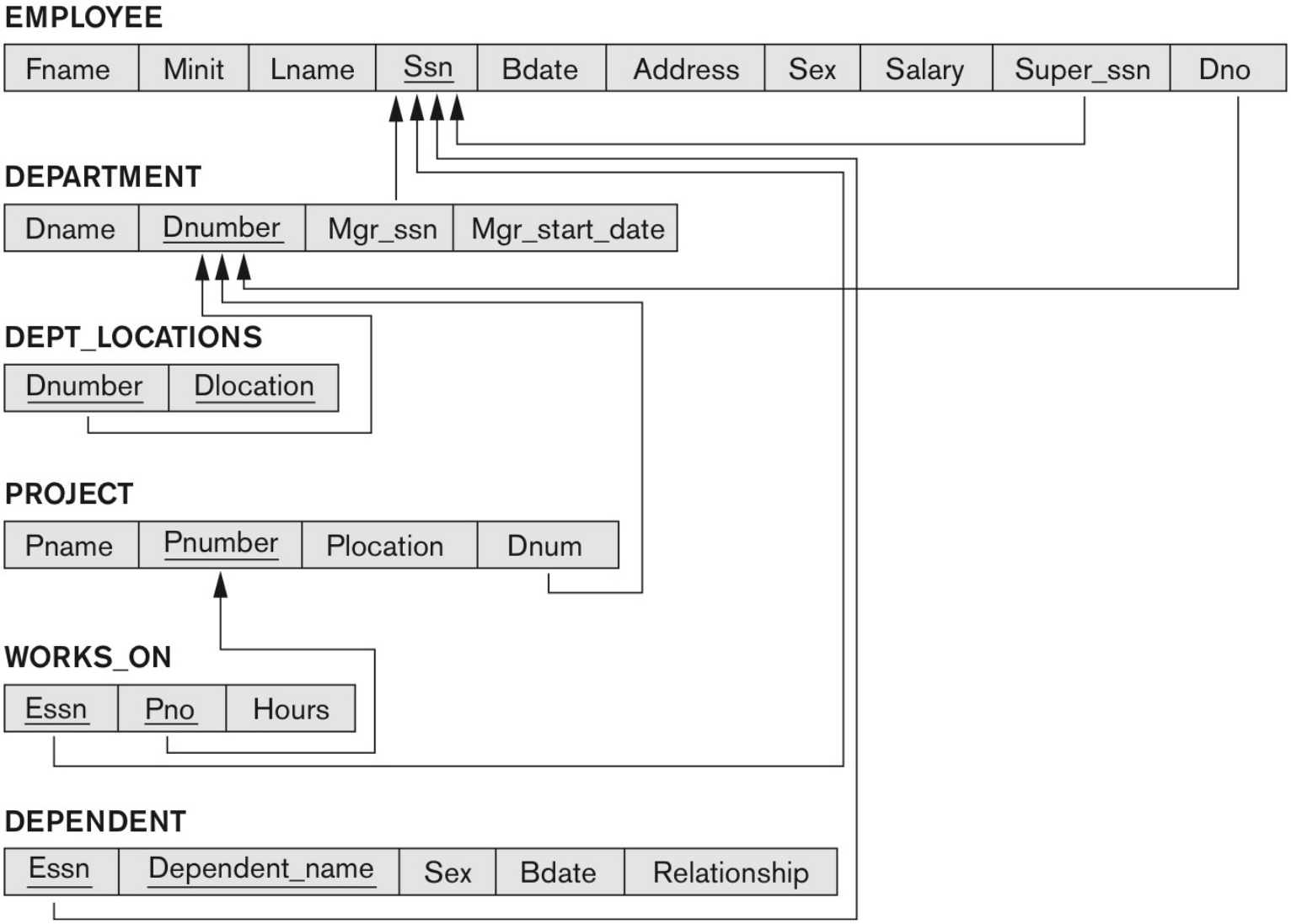
Announcement

- Start learning PHP
- Start studying for MT

Today's Lecture

1. Relational Algebra
2. Relational Calculus
3. PHP + SQL

Employee Database Schema



Examples of Queries in Relational Algebra : Procedural Form

- Q1: Retrieve the name and Address of all Employees who work for the 'Research' department.

Research_dept $\leftarrow \sigma_{\text{Dname='Research'}}(\text{Department})$

Research_emps $\leftarrow$ (RESEARCH_DEP $\bowtie_{\text{DNumber= Dno}}$ Employee)

Result $\leftarrow \pi_{\text{Fname, Lname, Address}}$ (Research_emps)

As a single expression, this query becomes:

$\pi_{\text{Fname, Lname, Address}}$ ($\sigma_{\text{Dname= 'Research'}}$ (Department) $\bowtie_{\text{Dnumber=Dno}}$ Employee)

Examples of Queries in Relational Algebra : Procedural Form

- Q6: Retrieve the names of Employees who have no dependents.

ALL_EMPS ← π_{SSN}(Employee)

EMPS_WITH_DEPS(SSN) ← π_{Essn}(DEPENDENT)

EMPS_WITHOUT_DEPS ← (ALL_EMPS - EMPS_WITH_DEPS)

RESULT ← $\pi_{Lname, Fname}$ (EMPS_WITHOUT_DEPS * Employee)

As a single expression, this query becomes:

$\pi_{Lname, Fname}((\pi_{Ssn}$ (Employee) − ρ_{Ssn} (π_{Essn} (Dependent))) * Employee)

Division

- T(Y) = R(Y,X) ÷ S(X)

- The complete division expression:
- $R \div S = \pi_Y R \ - \pi_Y\big((\pi_Y(R) \times S) - R\big)$

Ignoring the projections, there are just three steps:

- Compute all possible attribute pairings
- Remove the existing pairings
- Remove the non–answers from the possible answers

https://www2.cs.arizona.edu/~mccann/research/divpresentation.pdf

Division Example

R(Y,X)

Y	X
y1	x1
y1	x2
y2	x1
y3	x1
y3	x2
y3	x3

S(X)

X
x1
x2

$\pi_Y(R) \times S$

Y	X
y1	x1
y1	x2
y2	x1
y2	x2
y3	x1
y3	x2

$(\pi_Y(R) \times S) - R$

Y	X
y2	x2

$\pi_Y\big((\pi_Y(R) \times S) - R\big)$

Y
y2

$R \div S =$
$\pi_Y R \; - \pi_Y\big((\pi_Y(R) \times S) - R\big)$

Y
y1
y3

RELATIONAL CALCULUS

Relational Calculus

- In addition to relational algebra, Codd introduced relational calculus.
- Relational calculus is a declarative database query language based on first-order logic.
- Codd's main technical result is that
  - relational algebra and relational calculus have essentially the same expressive power.
- Relational calculus comes into two different flavors:
  - Tuple relational calculus
  - Domain relational calculus.

What is First Order Logic (Informally)

" First-Order Logic = Propositional Logic + (∃ and ∀)",
where
∃ and ∀ range over possible values occurring in relations.

Relational Algebra vs Relational Calculus

- In a calculus expression, there is no order of operations to specify how to retrieve the query result.
- A calculus expression specifies only what information the result should contain.
- A calculus expression specifies what is to be retrieved rather than how to retrieve it.

Relational Algebra vs Relational Calculus (cont.)

- Relational calculus is considered to be a nonprocedural or declarative language.
- This differs from relational algebra
- In relational algebra, we need to write a sequence of operations to specify a retrieval request
- hence
  - relational algebra can be considered as a procedural way of stating a query.
  - relational calculus can be considered as a declarative way of stating a query.

Tuple Relational Calculus

- The tuple relational calculus is based on specifying a number of tuple variables.
- Each tuple variable usually ranges over a particular database relation, meaning that the variable may take as its value any individual tuple from that relation.
- A simple tuple relational calculus query is of the form

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

  - where t is a tuple variable and COND (t) is a conditional expression or formula involving t.
  - The result of such a query is the set of all tuples t that satisfy COND (t).

Formula and Atoms

- A formula is made up of atoms
- An atom can be one of the following:
  - $R(t)$
    - where R is a relation and t is a tuple variable
  - $t_i.A$ op $t_j.B$
    - op can be: $=, <, \leq, >, \geq, \neq$
  - $t_i.A$ op c
    - c is a constant.

Formula

- A formula (boolean condition) is made up of one or more atoms.
- Atoms are connected via AND, OR, and NOT

- Rules:
- Rule 1: Every atom is a formula
- Rule 2: If F_1 and F_2 are formulas, then so are:
  - (F_1 AND F_2),
  - (F_1 OR F_2),
  - NOT(F_1),
  - NOT(F_2)

Tuple Relational Calculus (continued)

- Example: Find the first and last names of all Employees whose salary is above $50,000

- Tuple calculus expression:

  {t.Fname, t.Lname | Employee(t) AND t.Salary > 50000}

$\pi_{Fname, Lname}$ $\sigma_{SALARY > 50000}$

The Existential and Universal Quantifiers

- Two special symbols called quantifiers can appear in formulas; these are the universal quantifier (∀) and the existential quantifier (∃).

- Informally, a tuple variable t is bound if it is quantified
  - meaning that it appears in an (∀ t) or (∃ t) clause
  - otherwise, it is free.

Rule 3 and 4

- If F is a formula, then so are (∃ t)(F) and (∀ t)(F)
- The formula (∃ t)(F) is true
  - if the formula F evaluates to true for some (at least one) tuple assigned to free occurrences of t in F
  - otherwise (∃ t)(F) is false.
- The formula (∀ t)(F) is true
  - if the formula F evaluates to true for every tuple assigned to free occurrences of t in F
  - otherwise (∀ t)(F) is false.

The Existential and Universal Quantifiers (continued)

- ∀ is called the universal or "for all" quantifier
- ∃ is called the existential or "there exists" quantifier

Example Query Using Existential Quantifier

- Example: Retrieve the name and Address of all Employees who work for the 'Research' department.

- Tuple calculus expression:

  {t.Fname, t.Lname, t.Address | Employee(t) and (∃ d)
  (Department(d) and d.Dname='Research' and d.DNumber=t.Dno) }

- If a tuple satisfies the conditions specified in the query, the attributes Fname, Lname, and Address are retrieved for each such tuple.

Example Query Using Existential Quantifier

- Example: For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, birth date, and address

- Tuple calculus expression:

  {p.Pnumber, p.Dnum, m.Lname, mBdate, m.Address | Project(p) AND Employee (m) AND p.Plocation = 'Stafford' AND ((∃d) (Department(d) AND p.Dnum = d.Dnumber AND d.Mgr_ssn = m.Ssn))}

Example Query Using Existential Quantifier

- Example: List the name of each employee who works on some project controlled by department number 5.
- Tuple calculus expression:

  {e.Lame, e.Fname | Employee(e) AND((∃x)(∃w) (Project (x) AND Works_on(w) AND x.Dnum = 5 AND w.Essn = e.Ssn AND x.pNumber = w.Pno))}

Example Query Using Universal Quantifier

- Example: List the name of each employee who works on all project controlled by department number 5.
- Tuple calculus expression:

{

e.Lame, e.Fname | Employee(e) AND

(

$(\forall x)$ (

NOT (Project (x)) OR NOT (x.Dnum = 5)

OR

((∃w) (Works_on(w) AND w.Essn = e.Ssn AND x.pNumber = w.Pno))

)

)

}

Example Query Using Universal Quantifier

- Example: List the name of each employee who works on all project controlled by department number 5.
- Tuple calculus expression:

{

e.Lame, e.Fname | Employee(e) AND F'} where F' =

(

$(\forall x)$ (

NOT (Project (x)) OR NOT (x.Dnum = 5)

OR

((∃w) (Works_on(w) AND w.Essn = e.Ssn AND x.pNumber = w.Pno))

)

)

Example Query Using Universal Quantifier

- Example: List the name of each employee who works on all project controlled by department number 5.
- Tuple calculus expression:

{

e.Lame, e.Fname | Employee(e) AND F'} where F' =

(

$(\forall x)$ (

NOT (Project (x)) OR F1)) Where F1 =

NOT (x.Dnum = 5)

OR

((∃w) (Works_on(w) AND w.Essn = e.Ssn AND x.pNumber = w.Pno))

Example Query Using Universal Quantifier

- Example: List the name of each employee who works on all project controlled by department number 5.
- Tuple calculus expression:

{

e.Lame, e.Fname | Employee(e) AND F'} where F' =

(

$(\forall x)$ (

NOT (Project (x)) OR F1)) Where F1 =

NOT (x.Dnum = 5)

OR

((∃w) (Works_on(w) AND w.Essn = e.Ssn AND x.pNumber = w.Pno))

A → B

- $A \rightarrow B = NOT\ A\ OR\ B$

A	B	A → B	NOT A OR B
0	0	1	1
0	1	1	1
1	0	0	0
1	1	1	1

- $XY \rightarrow Z = NOT\ X\ OR\ NOT\ Y\ OR\ Z$

Example Query Using Universal Quantifier

- Example: List the name of each employee who works on all project controlled by department number 5.
- Tuple calculus expression:

{

e.Lame, e.Fname | Employee(e) AND F'} where F' =

(

$(\forall x)$

((Project (x) AND x.Dnum = 5) $\rightarrow$

((∃w) (Works_on(w) AND w.Essn = e.Ssn AND x.pNumber = w.Pno))

The Domain Relational Calculus

- Another variation of relational calculus is called the
  - domain relational calculus

- Domain calculus differs from tuple calculus in the type of variables used in formulas:
  - Rather than having variables range over tuples, the variables range over single values from domains of attributes.

The Domain Relational Calculus (continued)

- An expression of the domain calculus is of the form

$$\{ x_1, x_2, \ldots, x_n \mid \text{COND}(x_1, x_2, \ldots, x_n, x_{n+1}, x_{n+2}, \ldots, x_{n+m})\}$$

  - where $x_1, x_2, \ldots, x_n, x_{n+1}, x_{n+2}, \ldots, x_{n+m}$ are domain variables that range over domains (of attributes)
  - and COND is a condition or formula of the domain relational calculus.

Example Query Using Domain Calculus

Retrieve the birthdate and Address of the Employee whose name is 'John B. Smith'.

- Query :

{**uv** | (∃ **q**) (∃ **r**) (∃ **s**) (∃ **t**) (∃ **w**) (∃ **x**) (∃ **y**) (∃ **z**)
(**Employee**(**qrstuvwxyz**) **and q='John' and r='B' and s='Smith'**)}

- **Abbreviated notation Employee**(**qrstuvwxyz**) uses the variables without the separating commas: **Employee**(**q,r,s,t,u,v,w,x,y,z**)

- Specify the *requested attributes*, Bdate and Address, by the free domain variables u for BDATE and v for Address.
- Specify the condition for selecting a tuple following the bar (|)

PHP + MYSQL

Web Database Programming Using PHP

- Techniques for programming dynamic features into Web
- PHP
  - Open source scripting language
  - Interpreters provided free of charge
  - Available on most computer platforms

PHP Guide

- References:
- https://www.w3schools.com/php/
- http://php.net/manual/en/langref.php

PHP+MySQL

- https://www.w3schools.com/php/php_mysql_intro.asp

- Use MySQL Improved Extension
- http://php.net/manual/en/book.mysqli.php

- Activity:
- http://betaweb.csug.rochester.edu/~tbiswas2/class_demo/li3/welcome.html

Acknowledgement

- Some of the slides in this presentation are taken from the slides provided by the authors.
- Many of these slides are taken from cs145 course offered by Stanford University.
- Thanks to YouTube, especially to Dr. Daniel Soper for his useful videos.