Relational Algebra & Relational Calculus

Assoc.Prof.Dr. Dang Tran Khanh

Outline

- Relational Algebra
  - Unary Relational Operations
  - Relational Algebra Operations from Set Theory
  - Binary Relational Operations
  - Additional Relational Operations
- Brief Introduction to Relational Calculus
- Exercises
- Reading:
  - [1]: Chapter 8

Relational Algebra Overview

- Relational algebra is the basic set of operations for the relational model
  - These operations enable a user to specify **basic retrieval requests** (or **queries**)
- The result of an operation is a *new relation*, which may have been formed from one or more *input* relations
  - This property makes the algebra “closed” (all objects in relational algebra are relations)
- A sequence of relational algebra operations forms a **relational algebra expression**
  - The result of a relational algebra expression is also a relation that represents the result of a database query (or retrieval request)

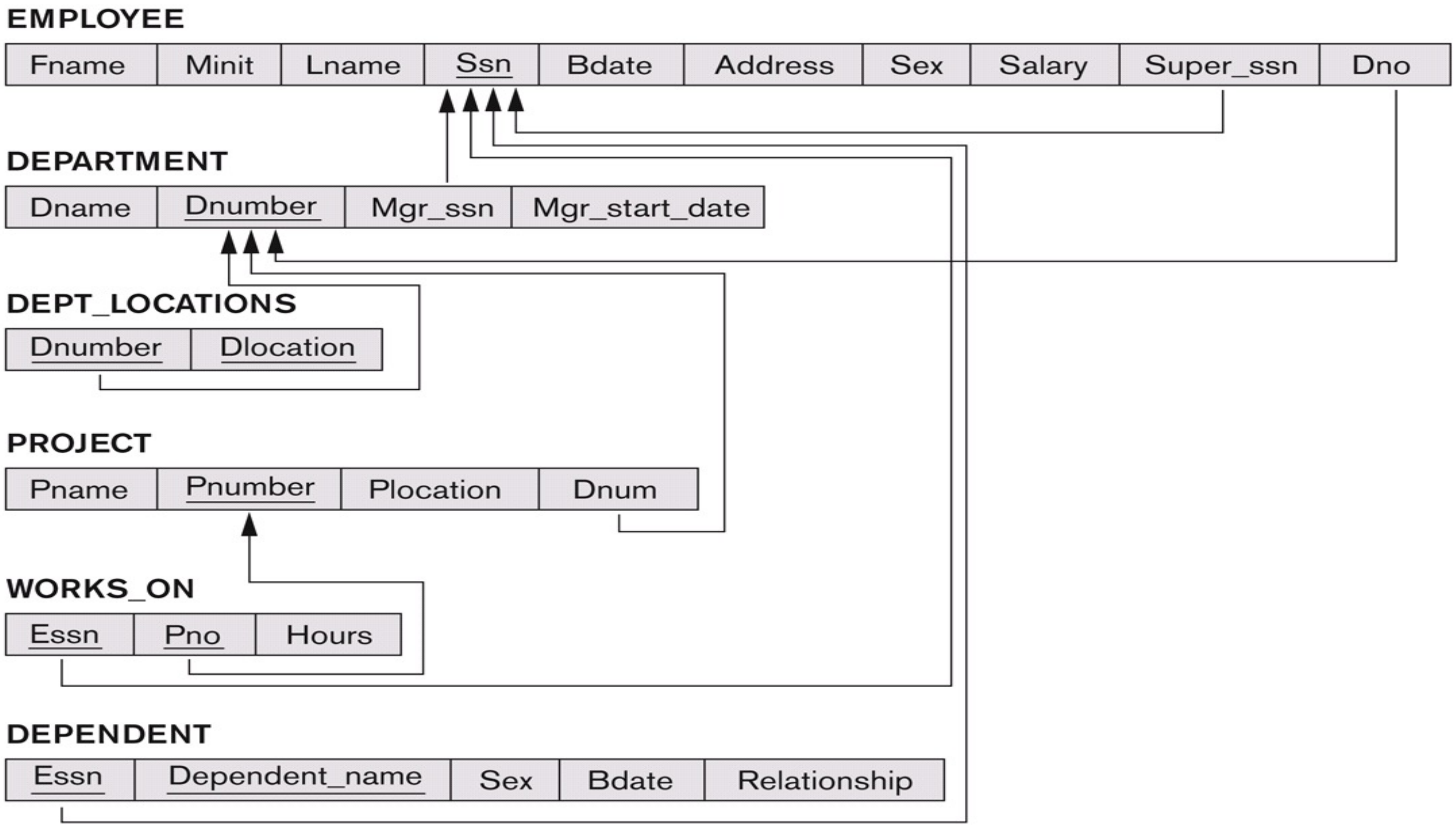
Relational Algebra Overview

- Relational Algebra consists of several groups of operations
  - Unary Relational Operations
    - →SELECT (symbol: σ (sigma))
    - →PROJECT (symbol: π (pi))
    - →RENAME (symbol: ρ (rho))
  - Relational Algebra Operations from Set Theory
    - →UNION ($\cup$), INTERSECTION ($\cap$), DIFFERENCE (or MINUS, **–**)
    - →CARTESIAN PRODUCT (**x**)
  - Binary Relational Operations
    - →JOIN (several variations of JOIN exist)
    - →DIVISION
  - Additional Relational Operations
    - →OUTER JOINS, OUTER UNION
    - →AGGREGATE FUNCTIONS (SUM, COUNT, AVG, MIN, MAX)

COMPANY Database Schema

- All examples discussed below refer to the COMPANY DB below:

Referential integrity constraints displayed on the COMPANY relational database schema.



The following query results refer to this database state

One possible database state for the COMPANY relational database schema

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

Unary Relational Operations: SELECT

- The SELECT operation (denoted by σ (sigma)) is used to select a *subset* of the tuples from a relation based on a **selection condition**.
- Examples:
  - Select the EMPLOYEE tuples whose department number is 4:

$$\sigma_{\text{DNO} = 4} \text{ (EMPLOYEE)}$$

  - Select the employee tuples whose salary is greater than $30,000:

$$\sigma_{\text{SALARY} > 30{,}000} \text{ (EMPLOYEE)}$$

Unary Relational Operations: SELECT

- In general, the *select* operation is denoted by $\sigma_{\text{<selection condition>}}(R)$ where
  - →the symbol **σ** (sigma) is used to denote the *select* operator
  - →the selection condition is a Boolean (conditional) expression specified on the attributes of relation R
  - →tuples that make the condition **true** appear in the result of the operation, and tuples that make the condition **false** are discarded from the result of the operation

Unary Relational Operations: SELECT

- SELECT Operation Properties
  - The relation S = $\sigma_{\text{<selection condition>}}(R)$ has the same schema (same attributes) as R
  - SELECT σ is commutative:

    $\rightarrow \sigma_{\text{<condition1>}}(\sigma_{\text{< condition2>}}(R)) = \sigma_{\text{<condition2>}}(\sigma_{\text{< condition1>}}(R))$

  - Because of commutativity property, a cascade (sequence) of SELECT operations may be applied in any order:

    $\rightarrow \sigma_{\text{<cond1>}}(\sigma_{\text{<cond2>}}(\sigma_{\text{<cond3>}}(R)) = \sigma_{\text{<cond2>}}(\sigma_{\text{<cond3>}}(\sigma_{\text{<cond1>}}(R)))$

    $= \sigma_{\text{<cond1> AND < cond2> AND < cond3>}}(R)))$

  - The number of tuples in the result of a SELECT is less than (or equal to) the number of tuples in the input relation R

Unary Relational Operations: PROJECT

- PROJECT Operation is denoted by π (pi)
- This operation keeps certain *columns* (attributes) from a relation and discards the other columns
  - PROJECT creates a vertical partitioning: the list of specified columns (attributes) is kept in each tuple, the other attributes in each tuple are discarded
- Example: To list each employee's first and last name and salary, the following is used:

$$\pi_{\text{LNAME, FNAME,SALARY}}(\text{EMPLOYEE})$$

Unary Relational Operations: PROJECT

- The general form of the *project* operation is:

$$\pi_{\text{<attribute list>}}(R)$$

  - <attribute list> is the desired list of attributes from relation R

- The project operation *removes any duplicate tuples* because the result of the *project* operation must be a *set of tuples and m*athematical sets *do not allow* duplicate elements

Unary Relational Operations: PROJECT

- PROJECT Operation Properties
  - The number of tuples in the result of projection $\pi_{<list>}(R)$ is always less or equal to the number of tuples in R

    →If the list of attributes includes a *key* of R, then the number of tuples in the result of PROJECT is *equal* to the number of tuples in R
  - PROJECT is *not* commutative
  - $\pi_{<list1>} (\pi_{<list2>} (R)) = \pi_{<list1>} (R)$ as long as <list2> contains the attributes in <list1>

    →If <list2> does not contain the attributes in <list1> ??

Examples of applying SELECT and PROJECT operations

Results of SELECT and PROJECT operations. (a) $\sigma_{(\text{Dno}=4 \textbf{ AND } \text{Salary}>25000) \text{ OR } (\text{Dno}=5 \textbf{ AND } \text{Salary}>30000)}$(EMPLOYEE). (b) $\pi_{\text{Lname, Fname, Salary}}$(EMPLOYEE). (c) $\pi_{\text{Sex, Salary}}$(EMPLOYEE).

(a)

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5

(b)

Lname	Fname	Salary
Smith	John	30000
Wong	Franklin	40000
Zelaya	Alicia	25000
Wallace	Jennifer	43000
Narayan	Ramesh	38000
English	Joyce	25000
Jabbar	Ahmad	25000
Borg	James	55000

(c)

Sex	Salary
M	30000
M	40000
F	25000
F	43000
M	38000
M	25000
M	55000

Relational Algebra Expressions

- We may want to apply several relational algebra operations one after the other
  - Either we can write the operations as a single **relational algebra expression** by nesting the operations, or
  - We can apply one operation at a time and create **intermediate result relations**.
- In the latter case, we must give names to the relations that hold the intermediate results.

Single expression versus sequence of relational operations

- To retrieve the first name, last name, and salary of all employees who work in department number 5, we must apply a select and a project operation
- We can write a *single relational algebra expression* as follows:
  - $\pi_{\text{FNAME, LNAME, SALARY}}(\sigma_{\text{DNO=5}}(\text{EMPLOYEE}))$
- OR We can explicitly show the *sequence of operations*, giving a name to each intermediate relation:
  - $\text{DEP5\_EMPS} \leftarrow \sigma_{\text{DNO=5}}(\text{EMPLOYEE})$
  - $\text{RESULT} \leftarrow \pi_{\text{FNAME, LNAME, SALARY}}(\text{DEP5\_EMPS})$

Unary Relational Operations: RENAME

- The RENAME operator is denoted by ρ (rho)
- In some cases, we may want to *rename* the attributes of a relation or the relation name or both
  - Useful when a query requires multiple operations
  - Necessary in some cases (see JOIN operation later)

Unary Relational Operations: RENAME

- The general RENAME operation ρ can be expressed by any of the following forms:
  - $\rho_{S\ (B1,\ B2,\ \ldots,\ Bn\ )}(R)$ changes both:
    →the relation name to S, *and*
    →the column (attribute) names to B1, B1, …..Bn
  - $\rho_S(R)$ changes:
    →the *relation name* only to S
  - $\rho_{(B1,\ B2,\ \ldots,\ Bn\ )}(R)$ changes:
    →the *column (attribute) names* only to B1, B1, …..Bn

Outline

- Relational Algebra
  - Unary Relational Operations
  - Relational Algebra Operations from Set Theory
  - Binary Relational Operations
  - Additional Relational Operations
- Brief Introduction to Relational Calculus
- Exercises
- Reading:
  - [1]: Chapter 8

Relational Algebra Operations from Set Theory: UNION

- UNION Operation
  - Binary operation, denoted by $\cup$
  - The result of $R \cup S$, is a relation that includes all tuples that are either in R or in S or in both R and S
  - Duplicate tuples are eliminated
  - The two operand relations R and S must be “type compatible” (or UNION compatible)
    - →R and S must have same number of attributes
    - →Each pair of corresponding attributes must be type compatible (have same or compatible domains)

Example of the result of a UNION operation

Result of the UNION operation RESULT ← RESULT1 ∪ RESULT2.

RESULT1

Ssn
123456789
333445555
666884444
453453453

RESULT2

Ssn
333445555
888665555

RESULT

Ssn
123456789
333445555
666884444
453453453
888665555

Relational Algebra Operations from Set Theory

- Type Compatibility of operands is required for the binary set operation UNION $\cup$, (also for INTERSECTION $\cap$, and SET DIFFERENCE –)
- The resulting relation for R1$\cup$R2 (also for R1$\cap$R2, or R1–R2) has the same attribute names as the *first* operand relation R1 (by convention)

Relational Algebra Operations from Set Theory: INTERSECTION

- INTERSECTION is denoted by $\cap$
- The result of the operation $R \cap S$, is a relation that includes all tuples that are in both R and S
  - The attribute names in the result will be the same as the attribute names in R
- The two operand relations R and S must be "type compatible"

Relational Algebra Operations from Set Theory: SET DIFFERENCE (cont.)

- SET DIFFERENCE (also called MINUS or EXCEPT) is denoted by –
- The result of R – S, is a relation that includes all tuples that are in R but not in S
  - The attribute names in the result will be the same as the attribute names in R
- The two operand relations R and S must be "type compatible"

Example to illustrate the result of UNION, INTERSECT, and DIFFERENCE

(a) **STUDENT**

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

INSTRUCTOR

Fname	Lname
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah

(b)

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert
John	Smith
Ricardo	Browne
Francis	Johnson

(c)

Fn	Ln
Susan	Yao
Ramesh	Shah

(d)

Fn	Ln
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

(e)

Fname	Lname
John	Smith
Ricardo	Browne
Francis	Johnson

The set operations UNION, INTERSECTION, and MINUS. (a) Two union-compatible relations. (b) STUDENT $\cup$ INSTRUCTOR. (c) STUDENT $\cap$ INSTRUCTOR. (d) STUDENT $-$ INSTRUCTOR. (e) INSTRUCTOR $-$ STUDENT.

Some properties of UNION, INTERSECT, and DIFFERENCE

- Notice that both union and intersection are *commutative* operations; that is
  - $R \cup S = S \cup R$, and $R \cap S = S \cap R$
- Both union and intersection can be treated as n-ary operations applicable to any number of relations as both are *associative* operations; that is
  - $R \cup (S \cup T) = (R \cup S) \cup T$
  - $(R \cap S) \cap T = R \cap (S \cap T)$
- The minus operation is not commutative; that is, in general
  - $R - S \neq S - R$

Relational Algebra Operations from Set Theory: CARTESIAN PRODUCT

- CARTESIAN (or CROSS) PRODUCT Operation
  - Denoted by R(A1, A2, . . ., An) x S(B1, B2, . . ., Bm)
  - Result is a relation Q with degree n + m attributes:

    →Q(A1, A2, . . ., An, B1, B2, . . ., Bm), in that order.
  - Hence, if R has n_R tuples (denoted as $|R| = n_R$), and S has n_S tuples, then R x S will have $n_R * n_S$ tuples
  - The two operands do NOT have to be "type compatible”

Binary Relational Operations: JOIN

- JOIN Operation (denoted by $\bowtie$)
  - The sequence of CARTESIAN PRODECT followed by SELECT is used quite commonly to identify and select related tuples from two relations
  - A special operation, called JOIN combines this sequence into a single operation
  - This operation is very important for any relational database with more than a single relation, because it allows us *combine related tuples* from various relations
  - The general form of a join operation on two relations R(A1, A2, . . ., An) and S(B1, B2, . . ., Bm) is:

$$R \bowtie_{\text{<join condition>}} S$$

→where R and S can be any relations that result from general *relational algebra expressions*.

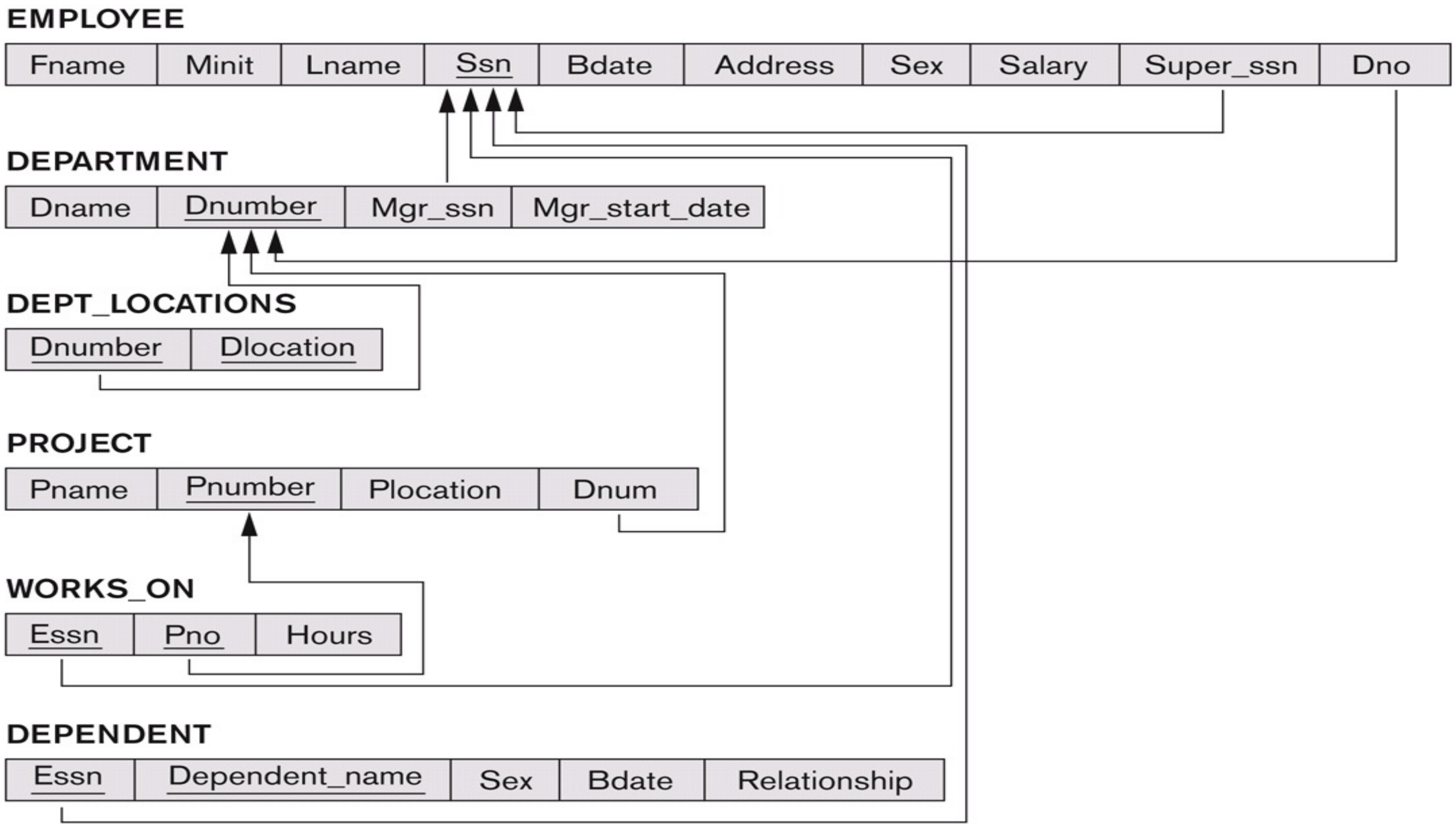
Binary Relational Operations: JOIN

- Example: Suppose that we want to retrieve the name of the manager of each department.
  - To get the manager's name, we need to combine each DEPARTMENT tuple with the EMPLOYEE tuple whose SSN value matches the MGRSSN value in the department tuple.
  - We do this by using the join $\bowtie$ operation.
  - DEPT_MGR $\leftarrow$ DEPARTMENT $\bowtie_{\text{MGRSSN=SSN}}$ EMPLOYEE
- MGRSSN=SSN is the join condition
  - Combines each department record with the employee who manages the department
  - The join condition can also be specified as DEPARTMENT.MGRSSN= EMPLOYEE.SSN

COMPANY Database Schema

- All examples discussed below refer to the COMPANY DB below:

Referential integrity constraints displayed on the COMPANY relational database schema.



Example of applying the JOIN operation

DEPT_MGR

Dname	Dnumber	Mgr_ssn	. . .	Fname	Minit	Lname	Ssn	. . .
Research	5	333445555	. . .	Franklin	T	Wong	333445555	. . .
Administration	4	987654321	. . .	Jennifer	S	Wallace	987654321	. . .
Headquarters	1	888665555	. . .	James	E	Borg	888665555	. . .

Result of the JOIN operation

$$\text{DEPT\_MGR} \leftarrow \text{DEPARTMENT} \bowtie_{\text{MGRSSN=SSN}} \text{EMPLOYEE}$$

Some properties of JOIN

- Consider the following JOIN operation:
  - R(A1, A2, . . ., An) $\bowtie_{R.Ai=S.Bj}$ S(B1, B2, . . ., Bm)
  - Result is a relation Q with degree n + m attributes:

    →Q(A1, A2, . . ., An, B1, B2, . . ., Bm), in that order
  - The resulting relation state has one tuple for each combination of tuples—r from R and s from S, but *only if they satisfy the join condition* r[Ai]=s[Bj]
  - Hence, if R has n_R tuples, and S has n_S tuples, then the join result will generally have *less than* n_R * n_S tuples.
  - Only related tuples (based on the join condition) will appear in the result

Some properties of JOIN

- The general case of JOIN operation is called a Theta-join: $R \underset{theta}{\bowtie} S$
- The join condition is called *theta*
- *Theta* can be any general boolean expression on the attributes of R and S; for example:
  - R.Ai<S.Bj AND (R.Ak=S.Bl OR R.Ap<S.Bq)

Binary Relational Operations: EQUIJOIN

- A join, where the only comparison operator used is =, is called an EQUIJOIN
  - In the result of an EQUIJOIN we always have one or more pairs of attributes (whose names need not be identical) that have identical values in every tuple

Binary Relational Operations: NATURAL JOIN Operation

- NATURAL JOIN Operation
  - Another variation of JOIN called NATURAL JOIN — denoted by * — was created to get rid of the second (superfluous) attribute in an EQUIJOIN condition
  - The standard definition of natural join requires that the two join attributes, or each pair of corresponding join attributes, *have the same name* in both relations
  - If this is not the case, a renaming operation is applied first
- Example: Q ← R(A,B,C,D) * S(C,D,E)
  - The implicit join condition includes *each pair* of attributes with the same name, "AND"ed together:

    →R.C=S.C AND R.D.S.D
  - Result keeps only one attribute of each such pair:

    →Q(A,B,C,D,E)

Complete Set of Relational Operations

- The set of operations $\{\boldsymbol{\sigma}, \boldsymbol{\pi}, \cup, -, \mathbf{X}\}$ is called a complete set because any other relational algebra expressions can be expressed by a combination of these five operations
- For example:
  - $R \cap S = (R \cup S) - ((R - S) \cup (S - R))$
  - $R \bowtie_{\text{<join condition>}} S = \sigma_{\text{<join condition>}} (R \; X \; S)$

Binary Relational Operations: DIVISION

- **DIVISION Operation**
  - The division operation is applied to two relations $R(Z) \div S(X)$, where $Z = X \cup Y$ (Y is the set of attributes of R that are not attributes of S
  - The result of DIVISION is a relation T(Y) that includes a tuple t if tuples t_R appear in R with t_R [Y] = t, and with

    t_R [X] = t_s *for every tuple* t_s in S, i.e., for a tuple t to appear in the result T of the DIVISION, the values in t must appear in R in combination with *every* tuple in S

The DIVISION operation

(a) Dividing SSN_PNOS by SMITH_PNOS

(b) $T \leftarrow R \div S$

(a)

SSN_PNOS	ESSN	PNO
	123456789	1
	123456789	2
	666884444	3
	453453453	1
	453453453	2
	333445555	2
	333445555	3
	333445555	10
	333445555	20
	999887777	30
	999887777	10
	987987987	10
	987987987	30
	987654321	30
	987654321	20
	888665555	20

SMITH_PNOS	PNO
	1
	2

SSNS	SSN
	123456789
	453453453

(b)

R	A	B
	a1	b1
	a2	b1
	a3	b1
	a4	b1
	a1	b2
	a3	b2
	a2	b3
	a3	b3
	a4	b3
	a1	b4
	a2	b4
	a3	b4

S	A
	a1
	a2
	a3

T	B
	b1
	b4

Recap of Relational Algebra Operations

Operations of Relational Algebra

Operation	Purpose	Notation
SELECT	Selects all tuples that satisfy the selection condition from a relation R.	$\sigma_{<\text{selection condition}>}(R)$
PROJECT	Produces a new relation with only some of the attributes of R, and removes duplicate tuples.	$\pi_{<\text{attribute list}>}(R)$
THETA JOIN	Produces all combinations of tuples from R_1 and R_2 that satisfy the join condition.	$R_1 \bowtie_{<\text{join condition}>} R_2$
EQUIJOIN	Produces all the combinations of tuples from R_1 and R_2 that satisfy a join condition with only equality comparisons.	$R_1 \bowtie_{<\text{join condition}>} R_2$, OR $R_1 \bowtie_{(<\text{join attributes 1}>), (<\text{join attributes 2}>)} R_2$
NATURAL JOIN	Same as EQUIJOIN except that the join attributes of R_2 are not included in the resulting relation; if the join attributes have the same names, they do not have to be specified at all.	$R_1 *_{<\text{join condition}>} R_2$, OR $R_1 *_{(<\text{join attributes 1}>), (<\text{join attributes 2}>)} R_2$ OR $R_1 * R_2$
UNION	Produces a relation that includes all the tuples in R_1 or R_2 or both R_1 and R_2; R_1 and R_2 must be union compatible.	$R_1 \cup R_2$
INTERSECTION	Produces a relation that includes all the tuples in both R_1 and R_2; R_1 and R_2 must be union compatible.	$R_1 \cap R_2$
DIFFERENCE	Produces a relation that includes all the tuples in R_1 that are not in R_2; R_1 and R_2 must be union compatible.	$R_1 - R_2$
CARTESIAN PRODUCT	Produces a relation that has the attributes of R_1 and R_2 and includes as tuples all possible combinations of tuples from R_1 and R_2.	$R_1 \times R_2$
DIVISION	Produces a relation $R(X)$ that includes all tuples $t[X]$ in $R_1(Z)$ that appear in R_1 in combination with every tuple from $R_2(Y)$, where $Z = X \cup Y$.	$R_1(Z) \div R_2(Y)$

Outline

- Relational Algebra
  - Unary Relational Operations
  - Relational Algebra Operations from Set Theory
  - Binary Relational Operations
  - Additional Relational Operations
- Brief Introduction to Relational Calculus
- Exercises
- Reading:
  - [1]: Chapter 8

Additional Relational Operations

- Aggregate Functions and Grouping
  - A type of request that cannot be expressed in the basic relational algebra is to specify mathematical **aggregate functions** on collections of values from the database
  - Examples of such functions include retrieving the average or total salary of all employees or the total number of employee tuples
  - Common functions applied to collections of numeric values include SUM, AVERAGE, MAXIMUM, and MINIMUM. The COUNT function is used for counting tuples or values

Examples of applying aggregate functions and grouping

The aggregate function operation.

(a) $\rho_{R(\text{Dno, No\_of\_employees, Average\_sal})}$ ($_{\text{Dno}}\mathfrak{I}_{\text{COUNT Ssn, AVERAGE Salary}}$ (EMPLOYEE)).

(b) $_{\text{Dno}}\mathfrak{I}_{\text{COUNT Ssn, AVERAGE Salary}}$ (EMPLOYEE).

(c) $\mathfrak{I}_{\text{COUNT Ssn, AVERAGE Salary}}$ (EMPLOYEE).

R

(a)

Dno	No_of_employees	Average_sal
5	4	33250
4	3	31000
1	1	55000

(b)

Dno	Count_ssn	Average_salary
5	4	33250
4	3	31000
1	1	55000

(c)

Count_ssn	Average_salary
8	35125

Additional Relational Operations

- **Use of the Functional operator $\mathcal{F}$**
  - $\mathcal{F}_{\text{MAX } \textit{Salary}}$ **(Employee**) retrieves the maximum salary value from the Employee relation
  - $\mathcal{F}_{\text{MIN } \textit{Salary}}$ **(Employee**) retrieves the minimum Salary value from the Employee relation
  - $\mathcal{F}_{\text{SUM } \textit{Salary}}$ **(Employee**) retrieves the sum of the Salary from the Employee relation
  - $_{\text{DNO}}\ \mathcal{F}_{\textit{COUNT SSN, AVERAGE Salary}}$ **(Employee**) groups employees by DNO (department number) and computes the count of employees and average salary per department
  - Note: count just counts the number of rows, without removing duplicates

Additional Relational Operations

- **Recursive Closure Operations**
  - Another type of operation that, in general, cannot be specified in the basic original relational algebra is **recursive closure.** This operation is applied to a **recursive relationship**
  - An example of a recursive operation is to retrieve all SUPERVISEES of an EMPLOYEE e at all levels
  - Although it is possible to retrieve employees at each level and then take their union, we cannot, in general, specify a query such as "retrieve the supervisees of 'James Borg' at all levels" without utilizing a looping mechanism
  - The SQL3 standard includes syntax for recursive closure
  - Details: homework !!
- Outer Join, Outer Union operations: homework !!

Outline

- Relational Algebra
  - Unary Relational Operations
  - Relational Algebra Operations from Set Theory
  - Binary Relational Operations
  - Additional Relational Operations
- Brief Introduction to Relational Calculus
- Exercises
- Reading:
  - [1]: Chapter 8

Brief Introduction to Relational Calculus

- A **relational calculus** expression creates a new relation, which is specified in terms of variables that range over rows of the stored database relations (in **tuple calculus**) or over columns of the stored relations (in **domain 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. This is the main distinguishing feature between relational algebra and relational calculus
- Relational calculus is considered to be a **nonprocedural** language. This differs from relational algebra, where we must write a *sequence of operations* to specify a retrieval request; hence relational algebra can be considered as a **procedural** way of stating a query

Brief Introduction to 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 | COND(t)}** where t is a tuple variable and COND (t) is a conditional expression involving t

Example: To find the first and last names of all employees whose salary is above $50,000, we can write the following tuple calculus expression:

{t.FNAME, t.LNAME | EMPLOYEE(t) AND t.SALARY>50000}

The condition EMPLOYEE(t) specifies that the **range relation** of tuple variable t is EMPLOYEE. The first and last name ($\pi_{\text{FNAME, LNAME}}$) of each EMPLOYEE tuple t that satisfies the condition t.SALARY>50000 ($\sigma_{\text{SALARY}>50000}$) will be retrieved

Brief Introduction to Relational Calculus

- 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**

Brief Introduction to Relational Calculus

- **Example 1:** retrieve the name and address of all employees who work for the 'Research' dept.

  {t.FNAME, t.LNAME, t.ADDRESS | EMPLOYEE(t) and (∃ d) (DEPARTMENT(d) and d.DNAME='Research' and d.DNUMBER=t.DNO) }

Brief Introduction to Relational Calculus

- **Example 2:** find the names of employees who work on *all* the projects controlled by department number 5

  {e.LNAME, e.FNAME | EMPLOYEE(e) and ((∀ x)(not(PROJECT(x)) or not(x.DNUM=5)

  OR ((∃ w)(WORKS_ON(w) and w.ESSN=e.SSN and x.PNUMBER=w.PNO))))}

- Details: [1] Chapter 8

Brief Introduction to Relational Calculus

- Another variation of relational calculus called the domain relational calculus, or simply, **domain calculus** is equivalent to tuple calculus and to relational algebra
- QBE (Query-By-Example): see Appendix D
- 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. To form a relation of degree n for a query result, we must have n of these **domain variables**—one for each attribute
- An expression of the domain calculus is of the form

  $\{x1, x2, . . ., xn \mid COND(x1, x2, . . ., xn, xn+1, xn+2, . . ., xn+m)\}$

  where x1, x2, . . ., xn, xn+1, xn+2, . . ., xn+m are domain variables that range over domains (of attributes) and COND is a **condition** or **formula** of the domain relational calculus

Brief Introduction to Relational Calculus

- Example: Retrieve the birthdate and address of the employee whose name is 'John B. Smith'.

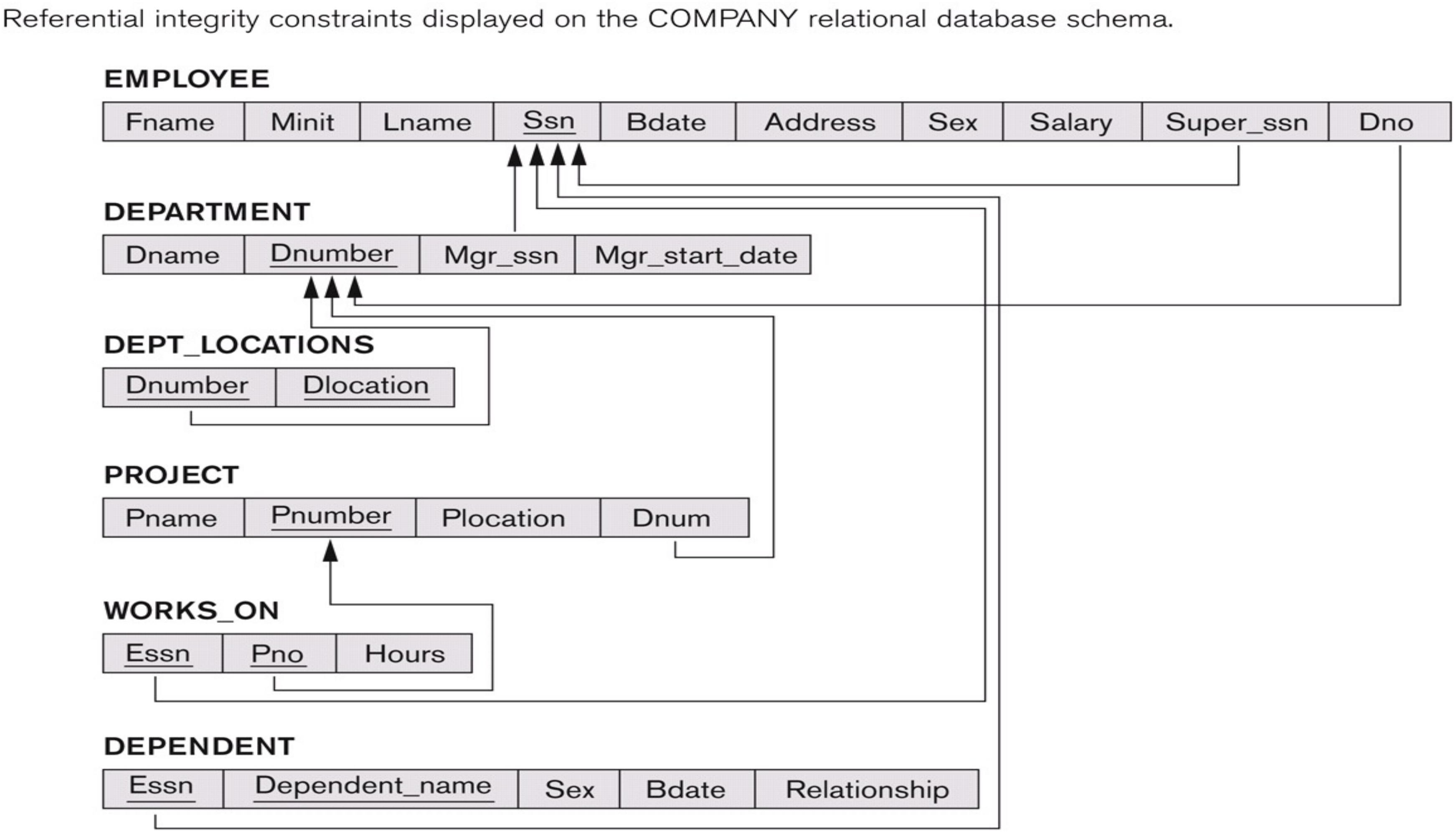
  {uv | (∃ q) (∃ r) (∃ s) (∃ t) (∃ w) (∃ x) (∃ y) (∃ z) (EMPLOYEE(qrstuvwxyz) and q='John' and r='B' and s='Smith')}

Outline

- Relational Algebra
  - Unary Relational Operations
  - Relational Algebra Operations from Set Theory
  - Binary Relational Operations
  - Additional Relational Operations
- Brief Introduction to Relational Calculus
- Exercises
- Reading:
  - [1]: Chapter 8

COMPANY Database Schema

- All examples discussed below refer to the COMPANY DB below:



Exercise

- **Using relational algebra: retrieve the name and address of all employees who work for the 'Research' department**

$RESEARCH\_DEPT \leftarrow \sigma_{DNAME='Research'}(DEPARTMENT)$

$RESEARCH\_EMPS \leftarrow (RESEARCH\_DEPT \bowtie_{DNUMBER=DNO} EMPLOYEE)$

$RESULT \leftarrow \pi_{FNAME, LNAME, ADDRESS}(RESEARCH\_EMPS)$

Summary

- Relational Algebra (an integral part of the relational data model)
  - Unary Relational Operations
  - Relational Algebra Operations from Set Theory
  - Binary Relational Operations
  - Additional Relational Operations
- Brief Introduction to Relational Calculus
  - Tuple Relational Calculus (the basis of SQL)
  - Domain Relational Calculus (e.g., QBE language in MS Access)
- Exercises
- Next Lecture
  - SQL - Structured Query Language
  - Reading:
    - →[1]: Chapters 6,7
    - →*www.oracle.com*
  - Exercises

Q&A

Question ?