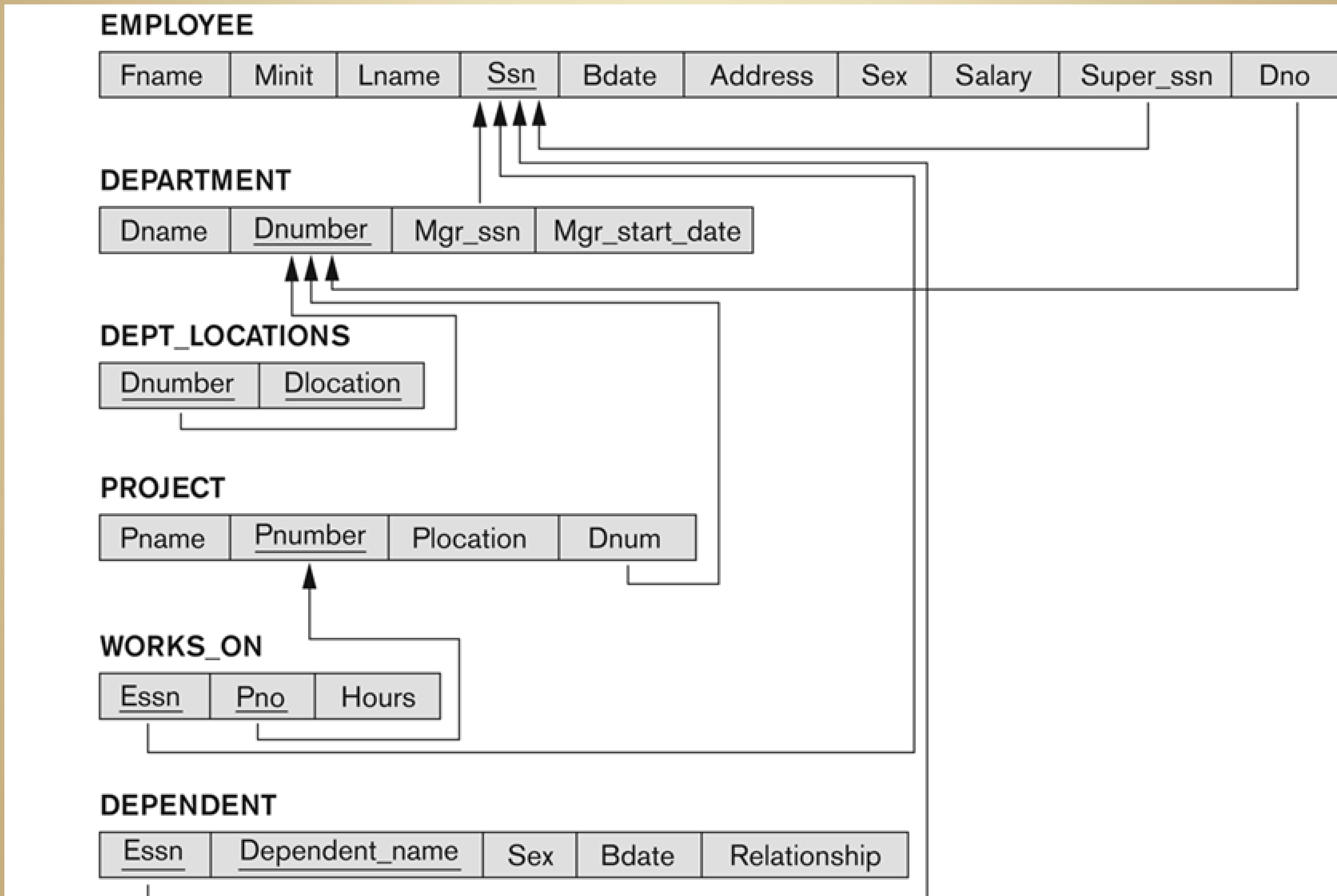
Advanced Database Management Systems

Lecture 7 – Chapter 6
More Relational Algebra

Relational Algebra Overview

- Unary Relational Operations
  - SELECT (symbol: **σ** (sigma))
  - PROJECT (symbol: **π** (pi))
  - RENAME (symbol: **ρ** (rho))
- Relational Algebra Operations From Set Theory
  - UNION (∪), INTERSECTION (∩), 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)

EXERCISE 2: Schema



EXERCISE 2: Queries

1. First and last names
   of all department managers.

2. Salaries of all employees
   who have worked on the Reorganization project.

3. SSN of all employees who have worked on a project
   that is controlled by a department different
   than the department that they are assigned to.

4. Last name of all employees who are not married.

Exercise 3: Schema

AIRPORT

Airport_code	Name	City	State

FLIGHT

Flight_number	Airline	Weekdays

FLIGHT_LEG

Flight_number	Leg_number	Departure_airport_code	Scheduled_departure_time	Arrival_airport_code	Scheduled_arrival_time

LEG_INSTANCE

Flight_number	Leg_number	Date	Number_of_available_seats	Airplane_id	Departure_airport_code	Departure_time	Arrival_airport_code	Arrival_time

FARE

Flight_number	Fare_code	Amount	Restrictions

AIRPLANE_TYPE

Airplane_type_name	Max_seats	Company

CAN_LAND

Airplane_type_name	Airport_code

AIRPLANE

Airplane_id	Total_number_of_seats	Airplane_type

SEAT_RESERVATION

Flight_number	Leg_number	Date	Seat_number	Customer_name	Customer_phone

EXERCISE 3: Queries

1. List all airplane types that can land at any airport in San Francisco.
2. List the ids and number of seats for all airplanes that can land at any airport in Chicago.
3. List the name and phone number of all customers with a seat reserved on a flight that leaves Chicago O'Hara airport (ORD) on October 31, 2008.
4. List all airlines that have seats available for flights leaving Los Angeles (LAX) on September 25, 2008.
5. List all airlines that operate at San Jose International Airport (SJC).

RENAME

- *Rename* the attributes of a relation
  or change the relation name
- The general RENAME operation ρ:

  - $\rho_{S\,(B1,\,B2,\,\ldots,\,Bn)}(R)$ 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

RENAME (shorthand)

- *shorthand* for renaming attributes :

$R1 \leftarrow \pi_{\text{FNAME, LNAME, SALARY}}$ (DEP5_EMPS)

R1 has same attribute names as DEP5_EMPS

$R2 \leftarrow \rho_{(F,L,S)} (\pi_{\text{FNAME, LNAME, SALARY}}$ (DEP5_EMPS))

R2 has attributes renamed to F, L and S

$R3\,(F,L,S) \leftarrow \pi_{\text{FNAME, LNAME, SALARY}}$ (DEP5_EMPS)

R3 also has attributes renamed to F, L and S

Aggregate Functions

- Aggregate functions apply mathematical operations to a collection of values.
- Examples:
  - compute average salary for all employees
  - compute maximum salary for all managers
  - count number of airports
- Common aggregate functions:
  - SUM
  - AVERAGE
  - MAX
  - MIN
  - COUNT

Aggregate Function

- Symbol for aggregate functions is $\mathcal{F}$
  - $\mathcal{F}_{\text{MAX Salary}}$ (EMPLOYEE)
    retrieves the maximum salary value from the EMPLOYEE relation
  - $\mathcal{F}_{\text{MIN Salary}}$ (EMPLOYEE)
    retrieves the minimum Salary value from the EMPLOYEE relation
  - $\mathcal{F}_{\text{SUM Salary}}$ (EMPLOYEE)
    retrieves the sum of the Salary from the EMPLOYEE relation
  - $\mathcal{F}_{\text{COUNT SSN, AVERAGE Salary}}$ (EMPLOYEE)
    computes the number of employees and their average salary

Grouping with Aggregation

- Grouping can be combined with Aggregate Functions
- Example:
  - For each department, retrieve the DNO, COUNT SSN, and AVERAGE SALARY
  - ${}_{\text{DNO}}\mathcal{F}_{\text{COUNT SSN, AVERAGE Salary}}$ (EMPLOYEE)
- Group employees by DNO,
  then compute aggregate functions on each group

Grouping with Aggregation

$$_{\text{DNO}}\mathcal{F}_{\text{COUNT SSN, AVERAGE Salary}}\text{(EMPLOYEE)}$$

Fname	Minit	Lname	Ssn	. . .	Salary	Super_ssn	Dno
John	B	Smith	123456789		30000	333445555	5
Franklin	T	Wong	333445555		40000	888665555	5
Ramesh	K	Narayan	666884444		38000	333445555	5
Joyce	A	English	453453453	. . .	25000	333445555	5
Alicia	J	Zelaya	999887777		25000	987654321	4
Jennifer	S	Wallace	987654321		43000	888665555	4
Ahmad	V	Jabbar	987987987		25000	987654321	4
James	E	Bong	888665555		55000	NULL	1

Dno	Count (*)	Avg (Salary)
5	4	33250
4	3	31000
1	1	55000

DIVISION

- Example:
  - Retrieve names of employees who work on ***all*** projects that John Smith works on.

- $R = S \div T$
  $x \in R$, iff for every $y \in T$, $\langle x,y \rangle \in S$

- The tuples in R come from the tuples in S that match every tuple in T

DIVISION Example

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

$T = R \div S$

T has all attributes of R that are not in S.

Find tuples in R that match every tuple in S.
Project the attributes that are not in S.

DIVISION Example

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

SSNS = SSN_PNOS ÷ SMITH_PNOS

Retrieve names of employees who work on ***all*** projects that John Smith works on.

DIVISION

- $R = S \div T$ can be computed as

  $R1 = \pi_Y(S)$
  $R2 = \pi_Y((T \times R1) - S)$
  $R = R1 - R2$

JOIN VARIANTS

- Equijoin
- Natural join
- Outer join
- Outer union

THETA JOIN

- Theta Join is the general case
  STORESTOCK $\bowtie_{\text{<Item = ItemId>}}$ STOCKITEM
  - The join condition is called *theta*
- *Join condition* can be any Boolean expression on the attributes of R and S
  - R.Ai<S.Bj AND (R.Ak=S.Bl OR R.Ap<S.Bq)
- Most join conditions involve one or more equality conditions, connected with AND
  - R.Ai=S.Bj AND R.Ak=S.Bl AND R.Ap=S.Bq

EQUIJOIN

- The most common use of join involves join conditions with *equality comparisons* only

- When the only comparison operator used is =, we call it 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.

NATURAL JOIN

- NATURAL JOIN removes the superfluous attributes in an EQUIJOIN
  - 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.

- Operator for NATURAL JOIN is *

  R * S joins R and S with equality tests
  on all common attributes,
  then removes duplicate attributes

NATURAL JOIN

- $Q \leftarrow R(A,B,C,D) * S(C,D,E)$
  - The implicit join condition includes *each pair* of attributes with the same name, connected by AND
    - $R \bowtie_{R.C = S.C \text{ AND } R.D = S.D} S$
  - Result keeps only one attribute of each such pair:
    - $Q(A,B,C,D,E)$

OUTER JOIN

- THETA JOIN, NATURAL JOIN and EQUIJOIN, eliminate tuples without a *matching* (related) tuple
  - Tuples with null in the join attributes are also eliminated
  - This amounts to loss of information.
- OUTER JOINs keep the unmatched tuples and match them with NULLs
  - LEFT OUTER JOIN (=⋈): keep all tuples from left relation
  - RIGHT OUTER JOIN (⋈=): keep all tuples from right relation
  - FULL OUTER JOIN (=⋈=): keep all tuples from both relations

OUTER JOIN EXAMPLE

S

A	B
1	4
2	5
3	6

T

C	D
5	8
5	9
7	10

R1 = S =⋈ $_{B=C}$ T (left join)

A	B	C	D
1	4	NULL	NULL
2	5	5	8
2	5	5	9
3	6	NULL	NULL

R2 = S ⋈= $_{B=C}$ T (right join)

A	B	C	D
2	5	5	8
2	5	5	9
NULL	NULL	7	10

R3 = S =⋈= $_{B=C}$ T (full join)

A	B	C	D
1	4	NULL	NULL
2	5	5	8
2	5	5	9
3	6	NULL	NULL
NULL	NULL	7	10

notice that R3 = R1 U R2

OUTER UNION

- OUTER UNION computes
  the union of tuples from two relations
  when the relations are *not type compatible*
  - Tuples are included for all tuples from either relation that do not match a tuple in the other relation
  - Tuples are also included for pairs of tuples from each attribute that do match on common attributes
  - Remaining attribute values from both relations are kept
  - Missing values are replaced with NULL

OUTER UNION

- Example: OUTER UNION of STUDENT(Name, SSN, Department, Advisor) and INSTRUCTOR(Name, SSN, Department, Rank)
  - Tuples from the two relations are matched based on having the same combination of values of the shared attributes: Name, SSN, Department
  - The result relation will have the following attributes:

    STUDENT_OR_INSTRUCTOR (Name, SSN, Department, Advisor, Rank)
  - If a student is also an instructor, both Advisor and Rank will have a value; otherwise, one of these two attributes will be null

Recursive Closure

- Example:
  - Find all employees who work under (directly or indirectly) James Borg
- This operation is applied to a **recursive relationship**.
  - simplified schema: EMPLOYEE(name, ID, superID)
  - superId is the ID of an employees supervisor (may be NULL)

Recursive Closure

- Example:
  - Find all employees who work under Miro

EMPLOYEE

Name	ID	superID
Ang	1	5
Wassim	2	3
Iria	3	NULL
Jung	4	6
Miro	5	3
Sally	6	5

RECURSIVE CLOSURE

- Solution: Retrieve employees at each level and build solution with union

$T1 = \pi_{ID}(\sigma_{Name=\text{“Miro”}}\ EMPLOYEE)$

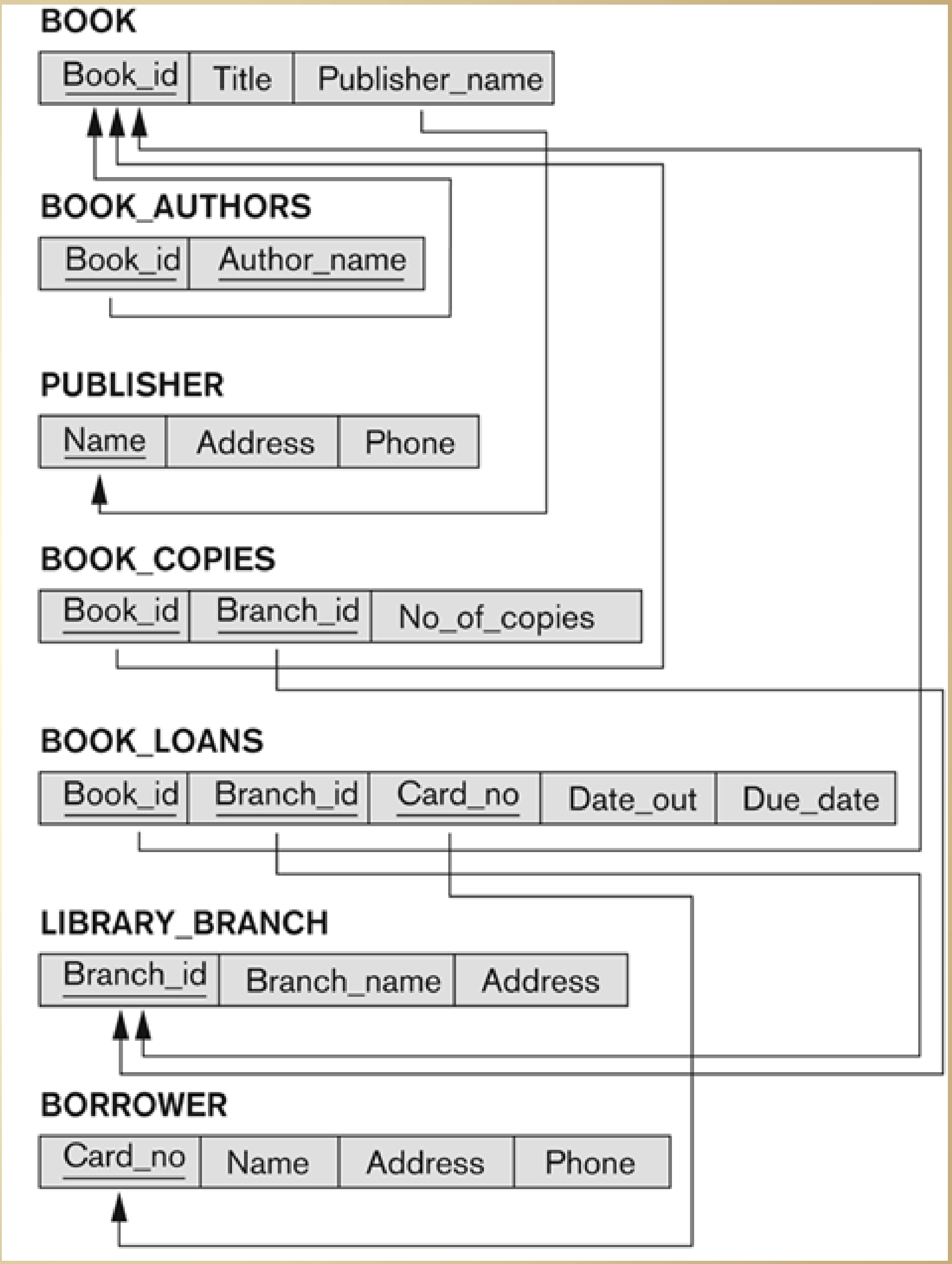
Repeat until T1 does not change:

$T2 = \pi_{ID}(EMPLOYEE \bowtie_{EMPLOYEE.superID=T1.ID} T1)$

$T1 = T1 \cup T2$

- cannot, in general, specify recursive closure without iteration (loops)
  - The SQL3 standard includes syntax for recursive closure.

EXERCISE 4: Schema



EXERCISE 4: Queries

1. Count the number of overdue books.
2. How many books by author Harry Crews are in the database?
3. Determine the number of library cards assigned to each borrower phone number.
4. Find names of all borrowers who do not have any book loans.
5. Do any library branches have every book?