CS 220
Relational Algebra

START RECORDING

- Notify about grading

Practice Quiz: Integrity Constraints

- What integrity constraints would be violated by the following operations, if any? (Operations don't affect each other.)
  1. DELETE FROM Employee WHERE SSN = 123456789
  2. DELETE FROM Employee WHERE SSN = 234567891
  3. DELETE FROM Department WHERE Name = "Research"
  4. DELETE FROM D_Locations WHERE Location = "Houston"
  5. UPDATE D_Locations SET Location = "Boston" WHERE Location = "Houston"
  6. INSERT INTO Employee (Name, SSN, Salary) VALUES ("John A. Smith", 123456789, 71000)
  7. INSERT INTO Employee (Name, Salary) VALUES ("John A. Smith", 71000)
  8. INSERT INTO Employee (Name, SSN, Salary) VALUES ("James Smith", "Unknown", 71000)

Employee

Name	SSN	Salary
John Smith	123456789	70000
Jane Smith	234567891	71000
Franklin Wong	345678912	72000

Department

Name	ID	Mgr_SSN
Research	1	345678912
Administration	2	234567891

Department_Locations

D_ID	Location
1	Houston
1	Boston
2	Boston

Today you will learn…

- How to retrieve information from a relational schema

Relational Query Languages

- Query = “retrieval program”
- Language examples:
  - Theoretical:
    1. Relational Algebra
    2. Relational Calculus
       a. tuple relational calculus (TRC)
       b. domain relational calculus (DRC)
  - Practical
    1. SQL (SEQUEL from System R)
    2. QUEL (Ingres)
    3. Datalog (Prolog-like)
- Theoretical QL’s:
  - give semantics to practical QL’s
  - key to understand query optimization in relational DBMSs

Chapter 8 Outline

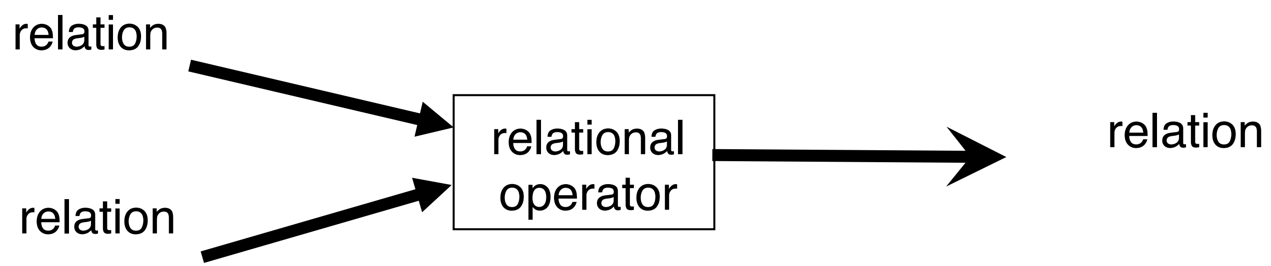
- Unary Relational Operations: SELECT and PROJECT
- Relational Algebra Operations from Set Theory
- Binary Relational Operations: JOIN and DIVISION
- Additional Relational Operations
- Examples of Queries in Relational Algebra
- The Tuple Relational Calculus
- The Domain Relational Calculus

The Relational Algebra and Relational Calculus

- **Relational algebra**
  - Basic set of operations for the relational model
- **Relational algebra expression**
  - Sequence of relational algebra operations
- **Relational calculus**
  - Higher-level declarative language for specifying relational queries

Relational Algebra

- Basic operators
  - select (σ)
  - project (π)
  - rename (ρ)
  - union ($\cup$)
  - set difference (-)
  - cartesian product (x)
- The operators take one or two relations as inputs and give a new relation as a result.



Example Instances

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Boats

Schema:
Boats(bid, bname, color)
Sailors(sid, sname, rating, age)
Reserves(sid, bid, day)

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S2

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

Unary Relational Operations

- Unary: applied to a single relation

- project (π)
- select (σ)
- rename (ρ)

The PROJECT Operation

- Selects columns from table and discards the other columns:

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

- **Degree**
  - Number of attributes in <attribute list>
- **Duplicate elimination**
  - Result of PROJECT operation is a set of distinct tuples

Projection

- **Examples:** $\pi_{age}(S2)$; $\pi_{sname,rating}(S2)$
- Retains only attributes that are in the "*projection list*".
- ***Schema*** of result:
  - exactly the columns in the projection list, with the same names that they had in the input relation.
- Projection operator has to *eliminate duplicates*
  - How do they arise? Why remove them?
  - Note: real systems typically don't do duplicate elimination unless the user explicitly asks for it. (Why not?)

Projection

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

S2

sname	rating
yuppy	9
lubber	8
guppy	5
rusty	10

$$\pi_{sname,rating}(S2)$$

age
35.0
55.5

$$\pi_{age}(S2)$$

Unary Relational Operations: SELECT

- The SELECT Operation
  - Subset of the tuples from a relation that satisfies a selection condition:

$$\sigma_{<\text{selection condition}>}(R)$$

    - Boolean expression contains clauses of the form
      <attribute name> <comparison op> **<constant value>**

      or

    - <attribute name> <comparison op> **<attribute name>**

Unary Relational Operations: SELECT

- Example:

$$\sigma_{(\text{Dno}=4\ \textbf{AND}\ \text{Salary}>25000)\ \textbf{OR}\ (\text{Dno}=5\ \textbf{AND}\ \text{Salary}>30000)}(\text{EMPLOYEE})$$

- <selection condition> applied independently to each individual tuple *t* in *R*
  - If condition evaluates to TRUE, tuple selected
- Boolean conditions **AND**, **OR**, and **NOT**

Unary Relational Operations: SELECT

- **Selectivity**
  - Fraction of tuples selected by a selection condition
- **Combine** SELECT operations into a single operation with **AND** condition

Selection (σ)

- **Selects rows that satisfy *selection condition*.**
- Result is a relation.

  Schema of result is same as that of the input relation

sid	sname	rating	age
28	yuppy	9	35.0
~~31~~	~~lubber~~	~~8~~	~~55.5~~
~~44~~	~~guppy~~	~~5~~	~~35.0~~
58	rusty	10	35.0

$$\sigma_{rating>8}(S2)$$

sname	rating
yuppy	9
rusty	10

$$\pi_{sname,rating}(\sigma_{rating>8}(S2))$$

Selection

- Notation: $\sigma_p(r)$
- *p* is called the selection predicate , **r** can be the name of a table, or another query
- Predicate:
  1. Simple
     - attr1 = attr2
     - Attr = constant value
       - (also, <, > , etc)
  2. Complex
     - predicate1 AND predicate2
     - predicate1 OR predicate2
     - NOT (predicate)

Rename (ρ)

- Allows us to refer to a relation by more than one name and to rename conflicting names

Example:

$$\rho\,(\mathrm{X}, E)$$

returns the expression *E* under the name *X*

- Rename relation and/or attributes

$$\rho_{S(B1,\, B2,\, \ldots,\, Bn)}(R) \quad \text{or} \quad \rho_S(R) \quad \text{or} \quad \rho_{(B1,\, B2,\, \ldots,\, Bn)}(R)$$

Sequences of Operations

- **In-line** expression:

$$\pi_{\text{Fname, Lname, Salary}}(\sigma_{\text{Dno}=5}(\text{EMPLOYEE}))$$

- Sequence of operations:

$$\text{DEP5\_EMPS} \leftarrow \sigma_{\text{Dno}=5}(\text{EMPLOYEE})$$
$$\text{RESULT} \leftarrow \pi_{\text{Fname, Lname, Salary}}(\text{DEP5\_EMPS})$$

- Renaming:
  - ρ*(first, last, salary)(RESULT)*

Binary Relational Operations

- Applied to two relations

- union ($\cup$)
- intersection ($\cap$)
- set difference (-)
- cartesian product (x)

UNION, INTERSECTION, and MINUS

- **UNION**, **INTERSECTION**, and **MINUS** take two input relations, which must be *union-compatible*:
  - Same number of columns (attributes)
  - Corresponding columns have the same domain (type)

UNION

- UNION
  - $R \cup S$
  - Includes all tuples that are either in R or in S or in both R and S
  - Duplicate tuples eliminated

INTERSECTION

- INTERSECTION
  - $R \cap S$
  - Includes all tuples that are in both *R* and *S*

MINUS

- SET DIFFERENCE (or MINUS)
  - *R* – *S*
  - Includes all tuples that are in *R* but not in *S*

Union

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S1

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

S2

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0
44	guppy	5	35.0
28	yuppy	9	35.0

$$S1 \cup S2$$

Intersection

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S1

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

S2

sid	sname	rating	age
31	lubber	8	55.5
58	rusty	10	35.0

$S1 \cap S2$

Set Difference

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S1

sid	sname	rating	age
22	dustin	7	45.0

$S1 - S2$

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

S2

sid	sname	rating	age
28	yuppy	9	35.0
44	guppy	5	35.0

$S2 - S1$

The CARTESIAN PRODUCT (CROSS PRODUCT) Operation

- **CARTESIAN PRODUCT**
  - **CROSS PRODUCT** or **CROSS JOIN**
  - Denoted by $\times$
  - Relations do not have to be union compatible
  - Useful when followed by a selection that matches values of attributes

Cartesian-Product

- **S1 $\times$ R1: Each row of S1 paired with each row of R1.**

  Like the c.p for mathematical relations: every tuple of S1 "appended" to every tuple of R1

- Q: How many rows in the result?
- *Result schema* has one field per field of S1 and R1, with field names `inherited' if possible.
  - *May have a naming conflict*: Both S1 and R1 have a field with the same name.
  - In this case, can use the *renaming operator*…

Cartesian Product Example

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

R1

S1 X R1 =

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
58	rusty	10	35.0	22	101	10/10/96
58	rusty	10	35.0	58	103	11/12/96

A Complete Set of Relational Algebra Operations: Basic Operators

- Set of relational algebra operations {σ, π, $\cup$, ρ, $-$, $\times$} is a **complete set**
  - Any relational algebra operation can be expressed as a sequence of operations from this set

Compound Operators

- In addition to the 6 basic operators, there are several additional “Compound Operators”
  - These add no computational power to the language, but are useful shorthands.
  - Can be expressed solely with the basic ops.

Intersection, revisited

- Intersection takes two input relations, which must be *union-compatible*.
- Q: How to express it using basic operators?

$$\mathbf{R \cap S = R - (R - S)}$$

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S1

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

S2

Intersection

sid	sname	rating	age
31	lubber	8	55.5
58	rusty	10	35.0

$$S1 \cap S2$$

THETA JOIN

- The (Theta) **JOIN** Operation
  - Denoted by $\bowtie$
  - Combine related tuples from two relations into single “longer” tuples
  - General join condition of the form <condition> **AND** <condition> **AND**...**AND** <condition>
  - Each <condition> of the form $A_i \ \theta \ B_j$
  - A_i and B_j are attributes of R and S, respectively
  - A_i and B_j have the same domain
  - θ (theta) is one of the comparison operators:
    - {=, <, ≤, >, ≥, ≠}
  - Example:

$$\text{DEPT\_MGR} \leftarrow \text{DEPARTMENT} \bowtie_{\text{Mgr\_ssn=Ssn}} \text{EMPLOYEE}$$
$$\text{RESULT} \leftarrow \pi_{\text{Dname, Lname, Fname}}(\text{DEPT\_MGR})$$

THETA JOIN

- *Condition Join (or "theta-join")*:
- *Result schema* same as that of cross-product.
- May have fewer tuples than cross-product.

$$R \bowtie_c S = \sigma_c (R \times S)$$

$$S1 \bowtie_{S1.sid < R1.sid} R1$$

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	58	103	11/12/96

EQUIJOIN

- **EQUIJOIN**
  - Only = comparison operator used
  - Always have one or more pairs of attributes that have identical values in every tuple

NATURAL JOIN

- **NATURAL JOIN**
  - Denoted by *
  - Conceptually (though in practice done more efficiently):
    - Compute R $\times$ S
    - Select rows where attributes that appear in both relations have equal values
    - Project all unique attributes and one copy of each of the common ones.
  - Useful for putting "normalized" relations back together.
  - Removes second (superfluous) attribute in an EQUIJOIN condition

Natural Join Example

sid	bid	day
22	101	10/10/96
58	103	11/12/96

R1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S1

S1 * R1 =

sid	sname	rating	age	bid	day
22	dustin	7	45.0	101	10/10/96
58	rusty	10	35.0	103	11/12/96

Inner Joins

- Type of match and combine operation
- Defined formally as a combination of CARTESIAN PRODUCT and SELECTION
- Join selectivity
  - Expected size of join result divided by the maximum size $n_R * n_S$
- We've seen:
  - THETA JOIN
  - EQUIJOIN
  - NATURAL JOIN

DIVISION

- Denoted by $\div$
- Example: retrieve the names of employees who work on all the projects that 'John Smith' works on
- Apply to relations $R(Z) \div S(X)$
  - Attributes of R are a subset of the attributes of S

DIVISION

- Useful for expressing "for all" queries like:
  Find sids of sailors who have reserved all boats.
- For A/B attributes of B are subset of attrs of A.
  - May need to "project" to make this happen.
- E.g., let *A* have 2 fields, *x* and *y*; *B* have only field *y*:

$$A/B = \left\{ \langle x \rangle \middle| \forall \langle y \rangle \in B (\exists \langle x, y \rangle \in A) \right\}$$

A/B contains all tuples (x) such that for every y tuple in B, there is an xy tuple in A.

Examples of Division A/B

sno	pno
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

A

pno
p2

B1

pno
p2
p4

B2

pno
p1
p2
p4

B3

sno
s1
s2
s3
s4

A/B1

sno
s1
s4

A/B2

sno
s1

A/B3

Expressing A/B Using Basic Operators

- Division is not essential op; just a useful shorthand.
  - (Also true of joins, but joins are so common that systems implement joins specially.)
- *Idea*: For *A/B*, compute all *x* values that are not \`disqualified' by some *y* value in *B*.
  - *x* value is *disqualified* if by attaching *y* value from *B*, we obtain an *xy* tuple that is not in *A*.

Disqualified x values: $\pi_x((\pi_x(A) \times B) - A)$

A/B: $\pi_x(A) -$ Disqualified x values

Operations of Relational Algebra

Table 8.1 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 $R_1 \bowtie_{\text{(<join attributes 1>),(<join attributes 2>)}} R_2$
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 $R_1 *_{\text{(<join attributes 1>),(<join attributes 2>)}} R_2$ OR $R_1 * R_2$

Operations of Relational Algebra (cont'd.)

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 $Z = X \cup Y$.	$R_1(Z) \div R_2(Y)$

COMPANY Database

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

PROJECT

Pname	Pnumber	Plocation	Dnum

Examples of Queries in Relational Algebra

Query 1. Retrieve the name and address of all employees who work for the 'Research' department.

$$\text{RESEARCH\_DEPT} \leftarrow \sigma_{\text{Dname}=\text{'Research'}}(\text{DEPARTMENT})$$
$$\text{RESEARCH\_EMPS} \leftarrow (\text{RESEARCH\_DEPT} \bowtie_{\text{Dnumber}=\text{Dno}} \text{EMPLOYEE})$$
$$\text{RESULT} \leftarrow \pi_{\text{Fname, Lname, Address}}(\text{RESEARCH\_EMPS})$$

As a single in-line expression, this query becomes:

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

Examples of Queries in Relational Algebra (cont'd.)

Query 2. For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birth date.

$$\text{STAFFORD\_PROJS} \leftarrow \sigma_{\text{Plocation}=\text{'Stafford'}}(\text{PROJECT})$$
$$\text{CONTR\_DEPTS} \leftarrow (\text{STAFFORD\_PROJS} \bowtie_{\text{Dnum}=\text{Dnumber}} \text{DEPARTMENT})$$
$$\text{PROJ\_DEPT\_MGRS} \leftarrow (\text{CONTR\_DEPTS} \bowtie_{\text{Mgr\_ssn}=\text{Ssn}} \text{EMPLOYEE})$$
$$\text{RESULT} \leftarrow \pi_{\text{Pnumber, Dnum, Lname, Address, Bdate}}(\text{PROJ\_DEPT\_MGRS})$$

Query Trees

- Represents the input relations of query as leaf nodes of the tree
- Represents the relational algebra operations as internal nodes

Query 2. For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birth date.

$$\text{STAFFORD\_PROJS} \leftarrow \sigma_{\text{Plocation}=\text{'Stafford'}}(\text{PROJECT})$$
$$\text{CONTR\_DEPTS} \leftarrow (\text{STAFFORD\_PROJS} \bowtie_{\text{Dnum}=\text{Dnumber}} \text{DEPARTMENT})$$
$$\text{PROJ\_DEPT\_MGRS} \leftarrow (\text{CONTR\_DEPTS} \bowtie_{\text{Mgr\_ssn}=\text{Ssn}} \text{EMPLOYEE})$$
$$\text{RESULT} \leftarrow \pi_{\text{Pnumber, Dnum, Lname, Address, Bdate}}(\text{PROJ\_DEPT\_MGRS})$$

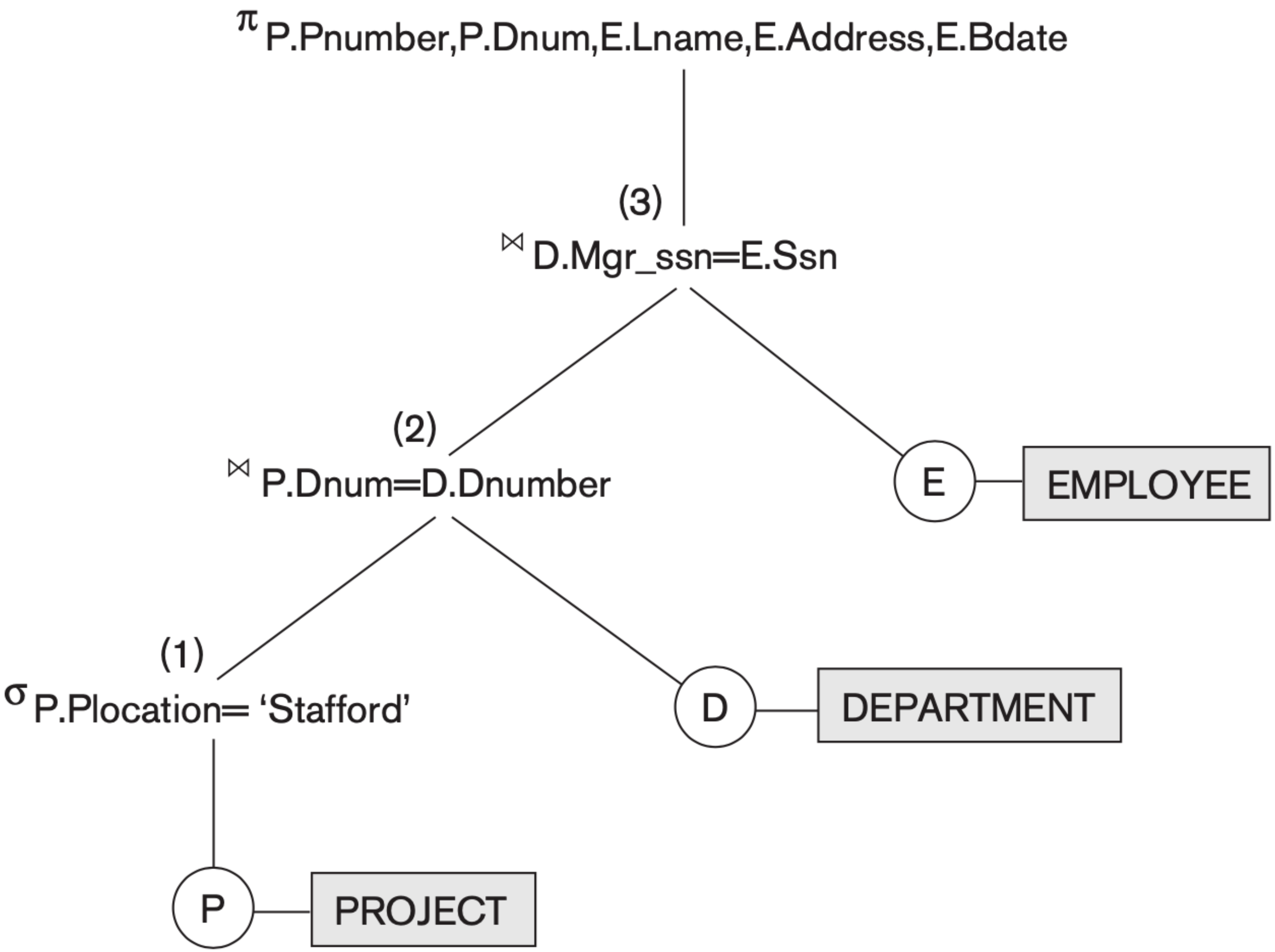


Figure 8.9
Query tree corresponding to the relational algebra expression for Q2.

Additional Ops: Generalized Projection

- Allows functions of attributes to be included in the projection list:

$$\pi_{F1,\, F2,\, ...,\, Fn}\,(R)$$

As an example, consider the relation

EMPLOYEE (Ssn, Salary, Deduction, Years_service)

A report may be required to show

Net Salary = Salary – Deduction,
Bonus = 2000 * Years_service, and
Tax = 0.25 * Salary

Then a generalized projection combined with renaming may be used as follows:

$$\text{REPORT} \leftarrow \rho_{(\text{Ssn, Net\_salary, Bonus, Tax})}(\pi_{\text{Ssn, Salary} - \text{Deduction, } 2000 * \text{Years\_service, } 0.25 * \text{Salary}}(\text{EMPLOYEE}))$$

Additional Ops: Aggregate Functions and Grouping

- **Aggregate functions**
  - Common functions applied to collections of numeric values
  - Include SUM, AVERAGE, MAXIMUM, and MINIMUM
- **Grouping**
  - Group tuples by the value of some of their attributes
  - Apply aggregate function independently to each group

$$_{<\text{grouping attributes}>}\, \mathfrak{I}_{\,<\text{function list}>} (R)$$

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

R

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

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

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

$\Im\ _{\text{COUNT Ssn, AVERAGE Salary}}(\text{EMPLOYEE}).$

Count_ssn	Average_salary
8	35125

OUTER JOIN Operations

- **Outer joins**
  - Keep all tuples in *R*, or all those in *S*, or all those in both relations regardless of whether or not they have matching tuples in the other relation
  - **Types**
    - **LEFT OUTER JOIN, RIGHT OUTER JOIN, FULL OUTER JOIN**
  - Example:

$$\text{TEMP} \leftarrow (\text{EMPLOYEE} ⟕_{\text{Ssn=Mgr\_ssn}} \text{DEPARTMENT})$$

$$\text{RESULT} \leftarrow \pi_{\text{Fname, Minit, Lname, Dname}}(\text{TEMP})$$

Left Outer Join Example

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S1 ⟕ R1 =

sid	sname	rating	age	bid	day
22	dustin	7	45.0	101	10/10/96
31	lubber	8	55.5	NULL	NULL
58	rusty	10	35.0	103	11/12/96

Summary

- Formal languages for relational model of data:
  - Relational algebra: operations, unary and binary operators
  - Some queries cannot be stated with basic relational algebra operations, but are important for practical use:
    - Aggregate functions and grouping
    - Recursive closure
- Next: relational calculus