RELATIONAL ALGEBRA

Corresponding Reading: Chapter 8.1, 8.2

8.3, 8.4

Relational Algebra Operations

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.	$\pi_{\text{<attribute list>}}(R)$
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.	$R_1 \times R_2$

Binary Relational Operations: JOIN

- The JOIN operation, denoted by ⋈ is used to combine related tuples from two relations into single tuples.
- Example:
  - Retrieve the name of the manager of each department.
    - To get name we need to combine each dept. tuple with the employee tuple whose SSN value matches the MGR_SSN value in the dept. tuple.
    - Use JOIN operation followed by a PROJECT

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

JOIN

- JOIN can be used in place of a CROSS PRODUCT followed by a SELECT
  - Last class we saw:

$$EMP\_DEPENDENTS \leftarrow EMPNAMES \times DEPENDENT$$
$$ACTUAL\_DEPENDENTS \leftarrow \sigma_{Ssn=Essn}(EMP\_DEPENDENTS)$$

  - This could be re-written with a JOIN:

$$ACTUAL\_DEPENDENTS \leftarrow EMPNAMES \bowtie_{Ssn=Essn} DEPENDENT$$

JOIN

- General form of a JOIN operation on two relations:

  $R(A_1, A_2, \ldots, A_n)$ $S(B_1, B_2, \ldots, B_m)$ is $R \bowtie_{\text{<join condition>}} S$

- The result of the JOIN is a relation Q with $n + m$ attributes

  $Q(A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m)$ in that order.

  - Q has one tuple for each combination of tuples (one from R and one from S) whenever the combination satisfies the JOIN CONDITION.

- FYI: This type of JOIN is referred to as an INNER JOIN. (We will see OUTER JOINs later.)

JOIN CONDITIONS

- A general JOIN CONDITION is of the form:

  <condition> **AND** <condition> **AND**...**AND** <condition>

  - Where each <condition> is of the form: $A_i \, \theta \, B_j$
    - A_i is an attribute of R
    - B_j is an attribute of S
    - A_i and B_j have the same domain
    - Theta is one of the comparison operators: $\{=, <, \leq, >, \geq, \neq\}$

  - A JOIN operation with this general JOIN CONDITION is called a **THETA JOIN**.

EQUIJOIN

- The most common use of JOIN involves join conditions with equality comparisons only.
  - **EQUIJOIN**: when the only comparison operator used is =
    - The previous examples of JOINs were EQUIJOINS
  - The result of an EQUIJOIN always has one or more pairs of attributes that have identical values in every tuple.

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	...

NATURAL JOIN

- To remove the superfluous duplicate attribute values in the result of the EQUIJOIN, a new operation called **NATURAL JOIN** was created.
  - Denoted by the following symbol: *
- The NATURAL JOIN requires that the two join attributes (or each pair of join attributes) have the same name in both relations.
  - If they are NOT the same, use a RENAME operation first.

NATURAL JOIN

- Example: Combine each PROJECT tuple with the DEPARTMENT tuple that controls the project.
  - Step 1: Rename the Dnumber attribute of DEPARTMENT to Dnum

$$DEPT \leftarrow \rho_{(Dname,\ Dnum,\ Mgr\_ssn,\ Mgr\_start\_date)}(DEPARTMENT)$$

  - Step 2: Apply the NATURAL JOIN operator

$$PROJ\_DEPT \leftarrow PROJECT * DEPT$$

  - Single In-line expression:

$$PROJ\_DEPT \leftarrow PROJECT * \rho_{(Dname,\ Dnum,\ Mgr\_ssn,\ Mgr\_start\_date)}(DEPARTMENT)$$

NATURAL JOIN

- Example: Combine the DEPARTMENT and DEPT_LOCATIONS relations.
  - Both relations have the same attribute names (Dnumber)
    - No need to rename the attributes

$$DEPT\_LOCS \leftarrow DEPARTMENT * DEPT\_LOCATIONS$$

DEPT_LOCS

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

N-way JOIN

- The NATURAL JOIN or EQUIJOIN operation can also be specified among multiple tables, which creates an n-way Join.
  - EXAMPLE: Three-way Join:
    - PROJECT, DEPARTMENT, EMPLOYEE

$$((\text{PROJECT} \bowtie_{\text{Dnum=Dnumber}} \text{DEPARTMENT}) \bowtie_{\text{Mgr\_ssn=Ssn}} \text{EMPLOYEE})$$

    - Combine each project tuple with its controlling department tuple into a single tuple
    - Then combine that tuple with an employee tuple that is the department manager

Query Trees

- Query Trees are notations typically used in relational systems to represent queries internally.
  - A tree data structure that corresponds to a relational algebra expression
    - Represents the input relations of the query as leaf nodes
    - Represents the relational algebra operations as internal nodes
  - Execution of the query tree consists of:
    - Executing an internal node operation whenever its operands (child nodes) are available
    - Replace that internal node by the relation that results from executing the operation
    - Execution terminates when the root node is executed and produces the result relation for the query.

Query Tree - Example

- 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.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

DEPT_LOCATIONS

Dnumber	Dlocation

PROJECT

Pname	Pnumber	Plocation	Dnum

WORKS_ON

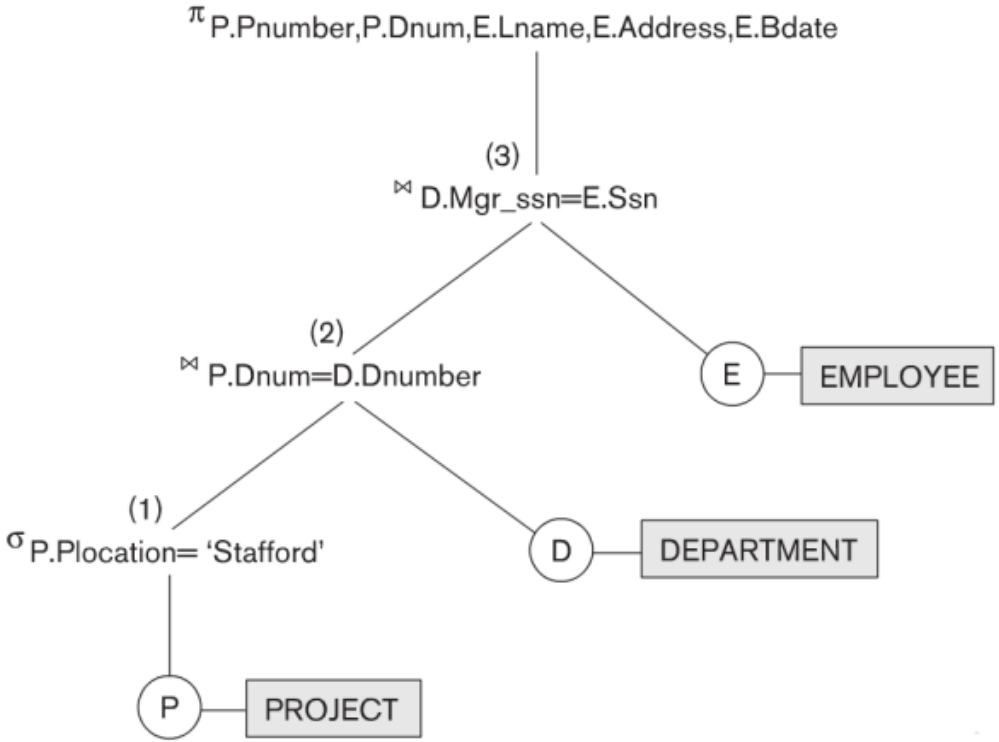
Essn	Pno	Hours

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

Query Tree - Example

$$\pi_{\text{Pnumber, Dnum, Lname, Address, Bdate}}(((\sigma_{\text{Plocation='Stafford'}}(\text{PROJECT})) \bowtie_{\text{Dnum=Dnumber}}(\text{DEPARTMENT})) \bowtie_{\text{Mgr\_ssn=Ssn}}(\text{EMPLOYEE}))$$



Summary of Relational Algebra Operations

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 $R_1 * R_2$

OUTER JOIN

- Inner JOIN operation matches tuples based on a join condition (JOIN, NATURAL JOIN, EQUIJOIN)
- Outer JOIN operation allows us to keep all the tuples in the Joined relations regardless of whether or not they have matching tuples.
  - Three types of OUTER JOIN:
    - LEFT OUTER JOIN
      - Keeps every tuple in the first (left) relation R in $R \bowtie S$
    - RIGH OUTER JOIN
      - Keeps every tuple in the second (right) relation S in $R \bowtie S$
    - FULL OUTER JOIN
      - Keeps all tuples in both the left and right relations $\bowtie$

LEFT OUTER JOIN - Example

- Suppose that we want a list of all employee names
  - As well as the name of the departments they manage *if they happen to manage a department*; if they do not manage one, we can indicate it with a NULL value.

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

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

RESULT

Fname	Minit	Lname	Dname
John	B	Smith	NULL
Franklin	T	Wong	Research
Alicia	J	Zelaya	NULL
Jennifer	S	Wallace	Administration
Ramesh	K	Narayan	NULL
Joyce	A	English	NULL
Ahmad	V	Jabbar	NULL
James	E	Borg	Headquarters

Queries in Relational Algebra

All queries on the COMPANY database:

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

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.

Queries in Relational Algebra

All queries on the COMPANY database:

3. Make a list of project numbers for projects that involve an employee whose last name is 'Smith', either as a worker or as a manager of the department that controls the project.

4. Lists the names of managers who have at least one dependent.

Query 1

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

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

- Another way: Reverse the order of the JOIN and SELECT operations
- Another way: Replace the JOIN with a NATURAL JOIN after renaming the join attributes.

Query 2

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

- First select the projects located in Stafford
- Then JOIN them with their controlling departments
- Then JOIN the result with the department managers
- Finally, apply a project operation on the desired attributes.

Query 3

$$SMITHS(Essn) \leftarrow \pi_{Ssn}\ (\sigma_{Lname='Smith'}(EMPLOYEE))$$
$$SMITH\_WORKER\_PROJS \leftarrow \pi_{Pno}(WORKS\_ON * SMITHS)$$
$$MGRS \leftarrow \pi_{Lname,\ Dnumber}(EMPLOYEE \bowtie_{Ssn=Mgr\_ssn} DEPARTMENT)$$
$$SMITH\_MANAGED\_DEPTS(Dnum) \leftarrow \pi_{Dnumber}\ (\sigma_{Lname='Smith'}(MGRS))$$
$$SMITH\_MGR\_PROJS(Pno) \leftarrow \pi_{Pnumber}(SMITH\_MANAGED\_DEPTS * PROJECT)$$
$$RESULT \leftarrow (SMITH\_WORKER\_PROJS \cup SMITH\_MGR\_PROJS)$$

- Retrieve the project numbers for projects that involve an employee named 'Smith' as a worker in SMITH_WORKERS_PROJS.
- Retrieve the project numbers for projects that involve an employee named 'Smith' as a manager of the department that controls the project in SMITH_MGR_PROJS.
- Apply the UNION operation on both intermediate relations.

Query 4

$$\text{MGRS}(\text{Ssn}) \leftarrow \pi_{\text{Mgr\_ssn}}(\text{DEPARTMENT})$$
$$\text{EMPS\_WITH\_DEPS}(\text{Ssn}) \leftarrow \pi_{\text{Essn}}(\text{DEPENDENT})$$
$$\text{MGRS\_WITH\_DEPS} \leftarrow (\text{MGRS} \cap \text{EMPS\_WITH\_DEPS})$$
$$\text{RESULT} \leftarrow \pi_{\text{Lname, Fname}}(\text{MGRS\_WITH\_DEPS} * \text{EMPLOYEE})$$

- Retrieve the SSNs of managers (MGRS)
- Retrieve the SSNs of employees with at least one dependent (EMPS_WITH_DEPS)
- Use SET INTERSECTION to find the SSNs of managers who have at least one dependent.