The Relational Algebra

Part 3

Additional Relational Operations

Generalized Projection

- The generalized projection operation extends the projection operation by allowing functions of attributes to be included in the projection list.
- The generalized form can be expressed as:

  $$\Pi_{F1,\ F2,\ \ldots,\ Fn}\ (R)$$

  where F_1, F_2, ..., F_n are functions over the attributes in relation R and may involve arithmetic operations and constant values.

- 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:

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

Aggregate Functions and Grouping

- We define an AGGREGATE FUNCTION operation, using the symbol $\mathfrak{I}$ (pronounced *script F*), to specify types of requests as follows:

  $$\texttt{<grouping attributes>}\ \mathfrak{I}\ \texttt{<function list>}\ (R)$$

  where `<grouping attributes>` is a list of attributes of the relation specified in R, and `<function list>` is a list of `(<function> <attribute>)` pairs.
- In each such pair, `<function>` is one of the allowed functions—such as `SUM`, `AVERAGE`, `MAXIMUM`, `MINIMUM`, `COUNT`—and `<attribute>` is an attribute of the relation specified by R.

- The resulting relation has the grouping attributes plus one attribute for each element in the function list.

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

- If no grouping attributes are specified, the functions are applied to *all the tuples* in the relation, so the resulting relation has a *single tuple only*.
- $\Im_{\text{COUNT Ssn, AVERAGE Salary}}(\text{EMPLOYEE})$

Recursive Closure Operations

- An example of a recursive operation is to retrieve all supervisees of an employee e at all levels—that is, all employees e' directly supervised by e, all employees e'' directly supervised by each employee e', all employees e''' directly supervised by each employee e'', and so on.

- It is relatively straightforward in the relational algebra to specify all employees supervised by e *at a specific level* by joining the table with itself one or more times.
- However, it is difficult to specify all supervisees at *all* levels.

OUTER JOIN Operations

- A set of operations, called **outer joins**, were developed for the case where the user wants to keep all the tuples in R, or all those in S, or all those in both relations in the result of the `JOIN`, regardless of whether or not they have matching tuples in the other relation.

- The LEFT OUTER JOIN operation keeps every tuple in the *first*, or *left*, relation R in $R ⟕ S$; if no matching tuple is found in S, then the attributes of S in the join result are filled or padded with NULL values.

- For 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.

- $\text{TEMP} \leftarrow (\text{EMPLOYEE} ⟕_{\text{Ssn=Mgr\_ssn}} \text{DEPARTMENT})$
  $\text{RESULT} \leftarrow \pi_{\text{Fname, Minit, Lname, Dname}}(\text{TEMP})$

- A similar operation, RIGHT OUTER JOIN, denoted by $⋊$, keeps every tuple in the *second*, or *right*, relation S in the result of $R \rtimes S$.
- A third operation, FULL OUTER JOIN, denoted by $⟗$,keeps all tuples in both the left and the right relations when no matching tuples are found, padding them with NULL values as needed.

The OUTER UNION Operation

- The **OUTER UNION** operation was developed to take the union of tuples from two relations that have some common attributes, but are *not union (type) compatible*.

- For example, an OUTER UNION can be applied to two relations whose schemas are 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 resulting relation, STUDENT_OR_INSTRUCTOR, will have the following attributes: STUDENT_OR_INSTRUCTOR(Name, Ssn, Department, Advisor, Rank)
- Tuples appearing only in STUDENT will have a NULL for the Rank attribute, whereas tuples appearing only in INSTRUCTOR will have a NULL for the Advisor attribute.
- A tuple that exists in both relations, which represent a student who is also an instructor, will have values for all its attributes.

- Notice that the same person may still appear twice in the result.
- For example, we could have a graduate student in the Mathematics department who is an instructor in the Computer Science department.
- Although the two tuples representing that person in `STUDENT` and `INSTRUCTOR` will have the same (`Name`, `Ssn`) values, they will not agree on the `Department` value, and so will not be matched.
- This is because `Department` has two different meanings in `STUDENT` (the department where the person studies) and `INSTRUCTOR` (the department where the person is employed as an instructor).
- If we wanted to apply the OUTER UNION based on the same (`Name`, `Ssn`) combination only, we should rename the `Department` attribute in each table to reflect that they have different meanings and designate them as not being part of the union-compatible attributes.
- For example, we could rename the attributes as `MajorDept` in `STUDENT` and `WorkDept` in `INSTRUCTOR`.

Examples of Queries in Relational Algebra

- **Query 1**. Retrieve the name and address of all employees who work for the ‘Research’ department.
- $\text{RESEARCH\_DEPT} \leftarrow \sigma_{\text{Dname= 'Research'}}(\text{DEPARTMENT})$
  $\text{RESEARCH\_EMPS} \leftarrow (\text{RESEARCH\_DEPT} \bowtie_{\text{Dnumber=Dno}} \text{EMPLOYEE})$
  $\text{RESULT} \leftarrow \pi_{\text{Fname, Lname, Address}}(\text{RESEARCH\_EMPS})$

- **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.
- STAFFORD_PROJS $\leftarrow \sigma_{\text{Plocation= ' Stafford '}}$(PROJECT)
CONTR_DEPTS $\leftarrow$ (STAFFORD_PROJS $\bowtie_{\text{Dnum=Dnumber}}$ DEPARTMENT)
PROJ_DEPT_MGRS $\leftarrow$ (CONTR_DEPTS $\bowtie_{\text{Mgr\_ssn=Ssn}}$ EMPLOYEE)
RESULT $\leftarrow \pi_{\text{Pnumber, Dnum, Lname, Address, Bdate}}$(PROJ_DEPT_MGRS)

- **Query 3**. Find the names of employees who work on *all* the projects controlled by department number 5.
- DEPT5_PROJS $\leftarrow \rho_{(\text{Pno})}(\pi_{\text{Pnumber}}(\sigma_{\text{Dnum=5}}(\text{PROJECT})))$
  EMP_PROJ $\leftarrow \rho_{(\text{Ssn, Pno})}(\pi_{\text{Essn, Pno}}(\text{WORKS\_ON}))$
  RESULT_EMP_SSNS $\leftarrow$ EMP_PROJ $\div$ DEPT5_PROJS
  RESULT $\leftarrow \pi_{\text{Lname, Fname}}(\text{RESULT\_EMP\_SSNS} * \text{EMPLOYEE})$

- **Query 4**. 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.
- 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)

- **Query 5**. List the names of all employees with two or more dependents.
- $\mathrm{T1(Ssn,\ No\_of\_dependents)} \leftarrow {}_{\mathrm{Essn}}\,\Im\,{}_{\mathrm{COUNT\ Dependent\_name}}\mathrm{(DEPENDENT)}$
  $\mathrm{T2} \leftarrow \sigma_{\mathrm{No\_of\_dependents>2}}\mathrm{(T1)}$
  $\mathrm{RESULT} \leftarrow \pi_{\mathrm{Lname,\ Fname}}\mathrm{(T2 * EMPLOYEE)}$

- **Query 6**. Retrieve the names of employees who have no dependents.
- $\text{ALL\_EMPS} \leftarrow \pi_{\text{Ssn}}(\text{EMPLOYEE})$
  $\text{EMPS\_WITH\_DEPS}(\text{Ssn}) \leftarrow \pi_{\text{Essn}}(\text{DEPENDENT})$
  $\text{EMPS\_WITHOUT\_DEPS} \leftarrow (\text{ALL\_EMPS} - \text{EMPS\_WITH\_DEPS})$
  $\text{RESULT} \leftarrow \pi_{\text{Lname, Fname}}(\text{EMPS\_WITHOUT\_DEPS} * \text{EMPLOYEE})$

- **Query 7**. List the names of managers who have at least one dependent.
- $\text{MGRS(Ssn)} \leftarrow \pi_{\text{Mgr\_ssn}}(\text{DEPARTMENT})$
  $\text{EMPS\_WITH\_DEPS(Ssn)} \leftarrow \pi_{\text{Essn}}(\text{DEPENDENT})$
  $\text{MGRS\_WITH\_DEPS} \leftarrow (\text{MGRS} \cap \text{EMPS\_WITH\_DEPS})$
  $\text{RESULT} \leftarrow \pi_{\text{Lname, Fname}}(\text{MGRS\_WITH\_DEPS} * \text{EMPLOYEE})$