The Relational Algebra

Part 2

Relational Algebra Operations from Set Theory

The UNION, INTERSECTION, and MINUS Operations

- **UNION**, **INTERSECTION**, and **SET DIFFERENCE** (also called **MINUS** or **EXCEPT**) are binary operations; that is, each is applied to two sets (of tuples).
- When these operations are adapted to relational databases, the two relations on which any of these three operations are applied must have the same **type of tuples**; this condition has been called *union compatibility* or *type compatibility*.
- Two relations $R(A_1, A_2, \ldots, A_n)$ and $S(B_1, B_2, \ldots, B_n)$ are said to be **union compatible** (or **type compatible**) if they have the same degree n and if dom(A_i) = dom(B_i) for $1 \leq i \leq n$.

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.

(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

- 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 because both are also *associative* operations; that is,

  $R \cup (S \cup T) = (R \cup S) \cup T$ and

  $(R \cap S) \cap T = R \cap (S \cap T)$

- The MINUS operation is not commutative; that is, in general,

  $R - S \neq S - R$

- Note that `INTERSECTION` can be expressed in terms of union and set difference as follows:

$$R \cap S = ((R \cup S) - (R - S)) - (S - R)$$

- $DEP5\_EMPS \leftarrow \sigma_{Dno=5}(EMPLOYEE)$
  $RESULT1 \leftarrow \pi_{Ssn}(DEP5\_EMPS)$
  $RESULT2(Ssn) \leftarrow \pi_{Super\_ssn}(DEP5\_EMPS)$
  $RESULT \leftarrow RESULT1 \cup RESULT2$

- In SQL, there are three operations—`UNION`, `INTERSECT`, and `EXCEPT`—that correspond to the set operations described here.
- In addition, there are multiset operations (`UNION ALL`, `INTERSECT ALL`, and `EXCEPT ALL`) that do not eliminate duplicates.

The CARTESIAN PRODUCT (CROSS PRODUCT) Operation

- The **CARTESIAN PRODUCT** operation—also known as **CROSS PRODUCT** or **CROSS JOIN**—is denoted by ×.
- In general, the result of $R(A_1, A_2, ..., A_n) \times S(B_1, B_2, ..., B_m)$ is a relation Q with degree $n + m$ attributes $Q(A_1, A_2, ..., A_n, B_1, B_2, ..., B_m)$, in that order.
- If R has n_R tuples (denoted as $|R| = n_R$), and S has n_S tuples, then $R \times S$ will have $n_R * n_S$ tuples.

- The *n*-ary CARTESIAN PRODUCT operation is an extension of the above concept, which produces new tuples by concatenating all possible combinations of tuples from *n* underlying relations.

- In general, the CARTESIAN PRODUCT operation applied by itself is generally meaningless.
- It is mostly useful when followed by a selection that matches values of attributes coming from the component relations.
- $\text{FEMALE\_EMPS} \leftarrow \sigma_{\text{Sex='F'}}(\text{EMPLOYEE})$
  $\text{EMPNAMES} \leftarrow \pi_{\text{Fname, Lname, Ssn}}(\text{FEMALE\_EMPS})$
  $\text{EMP\_DEPENDENTS} \leftarrow \text{EMPNAMES} \times \text{DEPENDENT}$
  $\text{ACTUAL\_DEPENDENTS} \leftarrow \sigma_{\text{Ssn=Essn}}(\text{EMP\_DEPENDENTS})$
  $\text{RESULT} \leftarrow \pi_{\text{Fname, Lname, Dependent\_name}}(\text{ACTUAL\_DEPENDENTS})$

- In SQL, `CARTESIAN PRODUCT` can be realized by using the `CROSS JOIN` option in joined tables.
- Alternatively, if there are two tables in the `WHERE` clause and there is no corresponding join condition in the query, the result will also be the `CARTESIAN PRODUCT` of the two tables.

Binary Relational Operations: JOIN and DIVISION

The JOIN Operation

- The `JOIN` operation can be specified as a `CARTESIAN PRODUCT` operation followed by a `SELECT` operation.
- The general form of a `JOIN` operation on two relations $R(A_1, A_2, \ldots, A_n)$ and $S(B_1, B_2, \ldots, B_m)$ 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*.

- 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, and θ (theta) is one of the comparison operators $\{=, <, \leq, >, \geq, \neq\}$.
- A JOIN operation with such a general join condition is called a **THETA JOIN**.

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

Variations of JOIN: The EQUIJOIN and NATURAL JOIN

- The most common use of JOIN involves join conditions with equality comparisons only.
- Such a JOIN, where the only comparison operator used is =, is called an **EQUIJOIN**.
- Notice that in the result of an EQUIJOIN we always have one or more pairs of attributes that have identical values in every tuple.
- Because one of each pair of attributes with identical values is superfluous, a new operation 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 join attributes) have the same name in both relations.
- If this is not the case, a renaming operation is applied first.

- PROJ_DEPT ← PROJECT * $\rho_{(\text{Dname, Dnum, Mgr\_ssn, Mgr\_start\_date})}$(DEPARTMENT)
- DEPT_LOCS ← DEPARTMENT * DEPT_LOCATIONS

- In general, the join condition for NATURAL JOIN is constructed by equating *each pair of join attributes* that have the same name in the two relations and combining these conditions with **AND**.

- Notice that if no combination of tuples satisfies the join condition, the result of a JOIN is an empty relation with zero tuples.
- In general, if R has n_R tuples and S has n_S tuples, the result of a JOIN operation $R \bowtie_{\texttt{<join condition>}} S$ will have between zero and $n_R * n_S$ tuples.
- The expected size of the join result divided by the maximum size $n_R * n_S$ leads to a ratio called **join selectivity**, which is a property of each join condition.
- If there is no join condition, all combinations of tuples qualify and the JOIN degenerates into a CARTESIAN PRODUCT, also called CROSS PRODUCT or CROSS JOIN.

- In SQL, JOIN can be realized in several different ways.
- The first method is to specify the `<join conditions>` in the `WHERE` clause, along with any other selection conditions.
- The second way is to use a nested relation.
- Another way is to use the concept of joined tables.

A Complete Set of Relational Algebra Operations

- It has been shown that the set of relational algebra operations $\{\sigma, \pi, \cup, \rho, -, \times\}$ is a complete set; that is, any of the other original relational algebra operations can be expressed as a *sequence of operations from this set*.

The DIVISION Operation

- An example is *Retrieve the names of employees who work on all the projects that 'John Smith' works on*.
- $\text{SMITH} \leftarrow \sigma_{\text{Fname='John' AND Lname='Smith'}}(\text{EMPLOYEE})$
  $\text{SMITH\_PNOS} \leftarrow \pi_{\text{Pno}}(\text{WORKS\_ON} \bowtie_{\text{Essn=Ssn}} \text{SMITH})$
  $\text{SSN\_PNOS} \leftarrow \pi_{\text{Essn, Pno}}(\text{WORKS\_ON})$
  $\text{SSNS(Ssn)} \leftarrow \text{SSN\_PNOS} \div \text{SMITH\_PNOS}$
  $\text{RESULT} \leftarrow \pi_{\text{Fname, Lname}}(\text{SSNS} * \text{EMPLOYEE})$

- In general, the DIVISION operation is applied to two relations $R(Z) \div S(X)$, where the attributes of S are a subset of the attributes of R; that is, $X \subseteq Z$.
- Let Y be the set of attributes of R that are not attributes of S; that is, $Y = Z - X$ (and hence $Z = X \cup Y$).
- 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.

- The DIVISION operation can be expressed as a sequence of π, $\times$, and $-$ operations as follows:

$$T1 \leftarrow \pi_{Y}(R)$$
$$T2 \leftarrow \pi_{Y}((S \times T1) - R)$$
$$T \leftarrow T1 - T2$$

- Most RDBMS implementations with SQL as the primary query language do not directly implement division.