Relational Algebra

Set-theoretic operations:

Two relations are union-compatible if they have the same number of attributes and the domains of the corresponding attributes in the two relations are the same.

Consider two relations $r(R)$ and $s(S)$ that are union-compatible (normally $R = S$).

**Union:** $r \cup s = \{ t | t \in r \text{ or } t \in s \}$.

**Difference:** $r - s = \{ t | t \in r \text{ and } t \notin s \}$.

**Intersection:** $r \cap s = \{ t | t \in r \text{ and } t \in s \}$.
**Cartesian Product:** $r(R)$ and $s(S)$ on any schemes $R$ and $S$.

$$r \times s = \{t_1 \cdot t_2 | t_1 \in r \text{ and } t_2 \in s\},$$

where, $t_1 \cdot t_2$ is the concatenation of tuples $t_1$ and $t_2$ to form a larger tuple.
Example: set operations

\[ r \]

\[
\begin{array}{cc}
A & B \\
| & |
\end{array}
\]

\[
\begin{array}{cc}
a & b \\
a & c \\
b & d \\
a & e
\end{array}
\]

\[ s \]

\[
\begin{array}{cc}
A & B \\
| & |
\end{array}
\]

\[
\begin{array}{cc}
a & c \\
a & e
\end{array}
\]

\[ r \cup s \]

\[
\begin{array}{cc}
A & B \\
| & |
\end{array}
\]

\[
\begin{array}{cc}
a & b \\
a & c \\
b & d \\
a & e
\end{array}
\]

\[ r - s \]

\[
\begin{array}{cc}
A & B \\
| & |
\end{array}
\]

\[
\begin{array}{cc}
a & b \\
b & d \\
a & c
\end{array}
\]

\[ r \cap s \]

\[
\begin{array}{cc}
A & B \\
| & |
\end{array}
\]

\[
\begin{array}{cc}
a & c \\
\end{array}
\]

\[ r \times s \]

\[
\begin{array}{cccc}
\text{r.A} & \text{r.B} & \text{s.A} & \text{s.B}
\end{array}
\]

\[
\begin{array}{cccc}
a & b & a & c \\
a & b & a & e \\
a & c & a & c \\
a & c & a & e \\
b & d & a & c \\
b & d & a & e
\end{array}
\]
Relation-theoretic operations

Consider \( r(R) \) and \( s(S) \), two relations, where \( R = (A_1, \ldots, A_n) \) and \( S = (B_1, \ldots, B_m) \).

**Rename:** \( r(C_1, \ldots, C_n) = \{ t | t \in r \} \) with schema \((C_1, \ldots, C_n)\).

**Select:** \( \sigma_F(r) = \{ t | t \in r \text{ and } t \text{ satisfies } F \} \).

where \( F \) is a selection criteria involving constants and attributes of \( r \). (will discuss in examples how \( F \) is constructed)

**Project:** \( \pi_{D_1, \ldots, D_p}(r) = \{ t[D_1, \ldots, D_p] | t \in r \} \)

where \( D_i \) is one of \( A_1, \ldots, A_n \).

**theta-Join:** \( r \bowtie_F s = \{ t | (\exists u \in r)(\exists v \in s)(t = u.v \text{ and } F \text{ is satisfied by } u \text{ and } v) \} \)

where \( F \) is a conjunction of formulas relating attributes of \( r \) with attributes of \( s \). (will discuss in examples how \( F \) is constructed)

**Natural Join:** \( r \bowtie s = \{ t | (\exists u \in r)(\exists v \in s)(t[R] = u \text{ and } t[S] = v) \} \)

**Division:** Assume \( B_1, \ldots, B_m \subset A_1, \ldots, A_n \).

\( r \div s = \{ t | (\forall u \in s)(t.u \in r) \} \)
Examples: relation-theoretic operations

\[ r \]

\[ \begin{array}{ccc}
A & C & D \\
\hline
a & c & d \\
a & e & f \\
a & g & h \\
b & c & d \\
b & g & h \\
c & c & d \\
c & e & f \\
\end{array} \]

\[ \sigma_{A=B} \text{ or } C=A(r) \]

\[ \begin{array}{ccc}
A & C & D \\
\hline
a & c & d \\
b & c & d \\
b & g & h \\
c & c & d \\
\end{array} \]

\[ \pi_A(r) \]

\[ \begin{array}{c}
A \\
a \\
b \\
c \\
\end{array} \]
Database Systems

\[ r \]

<table>
<thead>
<tr>
<th>A</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>c</td>
<td>d</td>
</tr>
<tr>
<td>a</td>
<td>e</td>
<td>f</td>
</tr>
<tr>
<td>a</td>
<td>g</td>
<td>h</td>
</tr>
<tr>
<td>b</td>
<td>c</td>
<td>d</td>
</tr>
<tr>
<td>b</td>
<td>g</td>
<td>h</td>
</tr>
<tr>
<td>c</td>
<td>c</td>
<td>d</td>
</tr>
<tr>
<td>c</td>
<td>e</td>
<td>f</td>
</tr>
</tbody>
</table>

\[ t \]

\[ r \bowtie_{r.A=t.B} t \]

<table>
<thead>
<tr>
<th>A</th>
<th>C</th>
<th>D</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>c</td>
<td>d</td>
<td>b</td>
<td>c</td>
<td>d</td>
</tr>
<tr>
<td>b</td>
<td>c</td>
<td>d</td>
<td>b</td>
<td>e</td>
<td>f</td>
</tr>
<tr>
<td>b</td>
<td>g</td>
<td>h</td>
<td>b</td>
<td>c</td>
<td>d</td>
</tr>
<tr>
<td>b</td>
<td>g</td>
<td>h</td>
<td>b</td>
<td>e</td>
<td>f</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>c</td>
<td>d</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>e</td>
<td>f</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>g</td>
<td>h</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>c</td>
<td>d</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>g</td>
<td>h</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>c</td>
<td>d</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>e</td>
<td>f</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>c</td>
<td>d</td>
</tr>
<tr>
<td>b</td>
<td>e</td>
<td>f</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>C</th>
<th>D</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>c</td>
<td>d</td>
<td>b</td>
</tr>
<tr>
<td>a</td>
<td>e</td>
<td>f</td>
<td>b</td>
</tr>
<tr>
<td>b</td>
<td>c</td>
<td>d</td>
<td>b</td>
</tr>
<tr>
<td>c</td>
<td>c</td>
<td>d</td>
<td>b</td>
</tr>
<tr>
<td>c</td>
<td>e</td>
<td>f</td>
<td>b</td>
</tr>
</tbody>
</table>
Basic Relational Algebra Operations

- Basic set: union, difference, Cartesian product, rename, select, and project.

- none of them can be expressed in terms of the others.

- intersection, theta-join, natural join, and division can be expressed in terms of the basic operators as follows:

  **Intersection:** \( r \cap s = r - (r - s) \)

  **theta Join:** \( r \bowtie_F s = \sigma_F(r \times s) \)

  **Natural Join:** \( r \bowtie s = \pi_{R \cap S}(\sigma_F(r \times s)) \)

  where \( F \) is a selection condition which indicates that the tuple values under the common attributes of \( r \) and \( s \) are equal.

  **Division:** \( r \div s = \pi_{R-S}(r) - \pi_{R-S}(\pi_{R-S}(r) \times s) - r \)

Even though relation schemes are defined as sequences, they are treated as sets in these equalities for simplicity.
An explanation for the equality for division is in order!

- First, all candidate tuples for the result are calculated by the expression
  \[ \pi_{R \setminus S}(r) \]
- Next, these candidate tuples are combined with all tuples of \( s \) in the following expression
  \[ \pi_{R \setminus S}(r) \times s \]
  to give a relation containing all combinations of candidate tuples with all tuples of \( s \).
- Since we are looking for tuples under the scheme \( R - S \) which combine with all tuples of \( s \) and are also present in \( r \),
  if we subtract \( r \) from the previous expression, we will get all the combinations of tuples that are “missing” in \( r \).
  \[ (\pi_{R \setminus S}(r) \times s) - r \]
- By projecting these tuples on \( R - S \), we get all those tuples that should not go to the result in the following expression.
  \[ \pi_{R \setminus S}( (\pi_{R \setminus S}(r) \times s) - r ) \]
- Finally, we subtract this set from the set of all candidate tuples and obtain the output relation of the division operator.
  \[ r \div s = \pi_{R \setminus S}(r) \setminus \pi_{R \setminus S}( (\pi_{R \setminus S}(r) \times s) - r ) \]
Query Tree Representations

![Query Tree Diagram](image)

**Figure 6.9**
Query tree corresponding to the relational algebra expression for Q2.

Aggregate Functions

**Figure 6.10**
The aggregate function operation,
(a) $\rho_{Dno, \text{No}_{-}\text{of}_{-}\text{employees}, \text{Average}_{-}\text{salary}}(Dno \bowtie \exists \text{COUNT Ssn, AVERAGE Salary}(\text{EMPLOYEE}))$.
(b) $\exists \text{COUNT Ssn, AVERAGE Salary}(\text{EMPLOYEE})$.
(c) $\exists \text{COUNT Ssn, AVERAGE Salary}(\text{EMPLOYEE})$.

<table>
<thead>
<tr>
<th></th>
<th>Dno</th>
<th>No_of_employees</th>
<th>Average_salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>5</td>
<td>4</td>
<td>33250</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>3</td>
<td>31000</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>55000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Dno</th>
<th>Count_ssn</th>
<th>Average_salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>(b)</td>
<td>5</td>
<td>4</td>
<td>33250</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>3</td>
<td>31000</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>55000</td>
</tr>
</tbody>
</table>

Outer Join Operations: Left Outer Join, Right Outer Join, and Full Outer Join

Outer Union Operation
Company Database Queries

Q1 Retrieve the names and address of all employees who work for the Research department.

\[
\begin{align*}
\text{RESEARCH\_DEPT} &= \sigma_{\text{DNAME}="Research"}(\text{DEPARTMENT}) \\
\text{RESEARCH\_DEPT\_EMPS} &= (\text{RESEARCH\_DEPT} \bowtie_{\text{DNUMBER}=\text{DNO}} \text{EMPLOYEE}) \\
\text{RESULT} &= \Pi_{\text{FNAME}, \text{LNAME}, \text{ADDRESS}}(\text{RESEARCH\_DEPT\_EMPS})
\end{align*}
\]

Q2 For every project located in Stafford, list the project number, the controlling department number, and the department manager’s last name, address and birthdate.

\[
\begin{align*}
\text{STAFFORD\_PROJS} &= \sigma_{\text{LOCATION}="Stafford"}(\text{PROJECT}) \\
\text{CONTR\_DEPT} &= (\text{STAFFORD\_PROJS} \bowtie_{\text{DNUMBER}=\text{DNUMBER}} \text{DEPARTMENT}) \\
\text{PROJ\_DEPT\_MGR} &= (\text{CONTR\_DEPT} \bowtie_{\text{MGRSSN}=\text{SSN}} \text{EMPLOYEE}) \\
\text{RESULT} &= \Pi_{\text{PNUMBER}, \text{DNUM}, \text{LNAME}, \text{ADDRESS}, \text{BDATE}}(\text{PROJ\_DEPT\_MGR})
\end{align*}
\]

Q3 Find the names of employees who work on ALL the projects controlled by department number 5.

\[
\begin{align*}
\text{DEPT5\_PROJS} &= \Pi_{\text{PNUMBER}}(\sigma_{\text{DNUMBER}=5}(\text{PROJECT}))(\text{PNO}) \\
\text{EMP\_PROJ} &= \Pi_{\text{SSN}, \text{PNO}}(\text{WORKS\_ON})(\text{SSN}, \text{PNO}) \\
\text{RESULT\_EMP\_SSNS} &= \text{EMP\_PROJ} \div \text{DEPT5\_PROJS} \\
\text{RESULT} &= \Pi_{\text{LNAME}, \text{FNAME}}(\text{RESULT\_EMP\_SSNS} \bowtie \text{EMPLOYEE})
\end{align*}
\]
Q4 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.

\[
\begin{align*}
SMITHS &= \Pi_{SSN}(\sigma_{LNAME="Smith"}(EMPLOYEE))(ESSN) \\
SMITH\_WORKER\_PROJS &= \Pi_{PENO}(WORKS\_ON \bowtie SMITHS) \\
MGRS &= \Pi_{LNAME, DNUMBER}(EMPLOYEE \bowtie_{SSN=MGRSSN} DEPARTMENT) \\
SMITH\_MGRS &= \sigma_{LNAME="Smith"}(MGRS) \\
SMITH\_MANAGED\_DEPTS &= \Pi_{DNUMBER}(SMITH\_MGRS)(DNUM) \\
SMITH\_MANAGED\_PROJS &= \Pi_{PNUMBER}(SMITH\_MANAGED\_DEPTS \bowtie PROJECT)(PNO) \\
RESULT &= SMITH\_WORKER\_PROJS \cup SMITH\_MGR\_PROJS
\end{align*}
\]

Q5 List the names of employees with two or more dependents.

\[
\begin{align*}
TEMP1 &= \Pi_{ESSN, DEPENDENT\_NAME}(DEPENDENT)(SSN1, DNAME1) \\
TEMP2 &= \Pi_{ESSN, DEPENDENT\_NAME}(DEPENDENT)(SSN2, DNAME2) \\
RESULT\_SSNS &= \Pi_{SSN1}(TEMP1 \bowtie_{SSN1=SSN2 and DNAME1>DNAME2} TEMP2) \\
RESULT &= \Pi_{LNAME,FNAME}(RESULT\_SSNS \bowtie EMPLOYEE)
\end{align*}
\]

Q6 Retrieve the names of employees who have no dependents.

\[
\begin{align*}
ALL\_EMPS &= \Pi_{SSN}(EMPLOYEE) \\
EMP\_WITH\_DEPS &= \Pi_{ESSN}(DEPENDENT)(SSN) \\
EMP\_WITHOUT\_DEPS &= ALL\_EMPS - EMP\_WITH\_DEPS \\
RESULT &= \Pi_{LNAME,FNAME}(EMP\_WITHOUT\_DEPS \bowtie EMPLOYEE)
\end{align*}
\]
Q7 List the names of managers who have at least one dependent.

\[
MGRS = \Pi_{MGRSSN}(DEPARTMENT)(SSN) \\
EMPS\_WITH\_DEPS = \Pi_{ESSN}(DEPENDENT)(SSN) \\
MGRS\_WITH\_DEPS = MGRS \cap EMPS\_WITH\_DEPS \\
RESULT = \Pi_{LNAME,FNAME}(MGRS\_WITH\_DEPS \bowtie EMPLOYEE)
\]