Quotient (Division) Operation (II)

- At the data level
  - Simplified definition
    - Simplified assumptions: $r > s$, $S \neq \emptyset$, $A_r = B_s$, $A_{r\text{-}1} = B_{s\text{-}1}$, $A_{r\text{-}s+1} = B_1$, result relation has schema $(A_1, ..., A_{r\text{-}s})$
    - Result relation of the quotient:

      $R \div S = \{(a_1, ..., a_{r\text{-}s}) \mid \forall\ (b_1, ..., b_s) \in S : (a_1, ..., a_{r\text{-}s}, b_1, ..., b_s) \in R\}$
  - General definition
    - $R \div S = \pi_{\mathcal{R}-\mathcal{S}}(R) - \pi_{\mathcal{R}-\mathcal{S}}((\pi_{\mathcal{R}-\mathcal{S}}(R) \times S) - R)$
    - $\pi_{\mathcal{R}-\mathcal{S}}(R)$ projects on all attributes that are not in $\mathcal{S}$, that is, it denotes all "prefixes" from R
    - $\pi_{\mathcal{R}-\mathcal{S}}(R) \times S$ creates all tuples that can be obtained by connecting the prefixes of R with all tuples of S; the schema of these tuples is $\mathcal{R}$
    - $(\pi_{\mathcal{R}-\mathcal{S}}(R) \times S) - R$ finds out which tuples from the previous step are not in R; as a side effect, if a prefix is in R with all tuples in S, it will not appear in the result any more, otherwise, this prefix will "survive"

Quotient (Division) Operation (III)

- At the data level (*continued*)
  - General definition (*continued*)
    - $\pi_{\mathcal{R}-\mathcal{S}}((\pi_{\mathcal{R}-\mathcal{S}}(R) \times S) - R)$ determines the prefixes that do *not* appear in R with all tuples in S
    - $\pi_{\mathcal{R}-\mathcal{S}}(R) - \pi_{\mathcal{R}-\mathcal{S}}((\pi_{\mathcal{R}-\mathcal{S}}(R) \times S) - R)$ determines the prefixes that appear in R with *all* tuples in S

Natural Join Operation

Common attributes are a prerequisite

R		
A	*B*	*C*
a_1	b_1	c_1
a_2	b_2	c_2

$\bowtie$

S		
C	*D*	*E*
c_1	d_1	e_1
c_3	d_2	e_2

=

$R \bowtie S$				
A	*B*	*C*	*D*	*E*
a_1	b_1	c_1	d_1	e_1

Equal values of common attributes decide about the concatenation of two tuples

- At the schema level
  - Assumptions: R has $m + k$ attributes $A_1, \ldots, A_m$, $B_1, \ldots, B_k$, and S has $n + k$ attributes $B_1, \ldots, B_k$, $C_1, \ldots, C_n$
  - $R \bowtie S$ has the arity $m + n + k$
  - $k \in \mathbb{N}, \forall\ 1 \leq i \leq k : dom(R.B_i) = dom(S.B_i)$
  - $\forall\ 1 \leq i \leq m\ \forall\ 1 \leq j \leq n : A_i \neq C_j$
- At the data level
  - $R \bowtie S = \pi_{A_1,\ldots,A_m,R.B_1,\ldots,R.B_k,C_1,\ldots,C_n}(\sigma_{R.B_1=S.B_1 \wedge \ldots \wedge R.B_k=S.B_k}(R \times S))$

Theta Join Operation

- At the schema level
  - Given: relations $R(A_1, ..., A_r)$, $S(B_1, ..., B_s)$
  - Result schema has the $r + s$ attributes $A_1, ..., A_r, B_1, ..., B_s$
- At the data level
  - $R \bowtie_F S = \sigma_F(R \times S)$
  - F is a Boolean predicate that consists of
    - attribute names from R and S (equal attributes from different schemas are qualified by the name of the relation schema, e.g., $R.A$, $S.A$, $S.B$)
    - comparison operators $=, \neq, <, >, \leq, \geq$
    - logical operators $\wedge$ and $\vee$
    - constants (e.g., 5, “Smith”)
  - A theta join of the form $R \bowtie_{R.A_i = S.B_j} S$ is denoted as an equi-join
- A theta join is more general than a natural join (see definition of a natural join, subsequent projection needed)

Outer Join Operations (I)

- Joins introduced so far are also called inner joins: the result only contains those tuples that found a matching partner in the other relation

- Outer joins are an extension of the natural join and *additionally* consider *partnerless* tuples of a relation that do *not* find a matching tuple in the other relation; the result tuples are "filled" with *null* ($\perp$) values

- Example of a left outer join

R		
A	*B*	*C*
a_1	b_1	c_1
a_2	b_2	c_2

⟕

S		
C	*D*	*E*
c_1	d_1	e_1
c_3	d_2	e_2

=

R ⟕ *S*				
A	*B*	*C*	*D*	*E*
a_1	b_1	c_1	d_1	e_1
a_2	b_2	c_2	$\perp$	$\perp$

- Tuples from *R* are "rescued" by all means (green ellipses)

- Tuples from *R* that cannot find a matching partner in *S* are filled up with null values (blue ellipse)

Outer Join Operations (II)

- Outer joins between two relations *R* and *S*
  - Left outer join (⟕): The tuples of *R* are preserved in any case
  - Right outer join (⟖): The tuples of *S* are preserved in any case
  - (Full) outer join (⟗): The tuples of *R* and *S* are preserved in any case
- Example of a full outer join

R		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

⟗

S		
C	D	E
c_1	d_1	e_1
c_3	d_2	e_2

=

R ⟗ *S*				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
a_2	b_2	c_2	$\perp$	$\perp$
$\perp$	$\perp$	c_3	d_2	e_2

Outer Join Operations (III)

- At the schema level
  - The assumptions are the same as for the natural join
  - The result schema is the same as for the natural join
- At the data level
  - Let $\mathcal{R}$ be the schema of relation R and $\mathcal{S}$ be the schema of relation S
  - Let $\mathcal{R} \cap \mathcal{S} \neq \varnothing$ (prerequisite of the natural join operation)
  - Let the relation U have the schema $\mathcal{S} - \mathcal{R}$. We define $U = \{(\perp, \ldots, \perp)\}$, that is, U is a relation with a *single* tuple and $|\mathcal{S} - \mathcal{R}|$ attributes, and all attribute values of this single tuple are *null*.
  - Correspondingly, let the relation V have the schema $\mathcal{R} - \mathcal{S}$. We define $V = \{(\perp, \ldots, \perp)\}$.
  - $R \mathbin{⟕} S = (R \bowtie S) \cup ((R - \pi_{\mathcal{R}}(R \bowtie S)) \times U)$
  - $R \mathbin{⟖} S = (R \bowtie S) \cup (V \times (S - \pi_{\mathcal{S}}(R \bowtie S)))$
  - $R \mathbin{⟗} S = (R \bowtie S) \cup ((R - \pi_{\mathcal{R}}(R \bowtie S)) \times U) \cup (V \times (S - \pi_{\mathcal{S}}(R \bowtie S)))$

Semijoin Operation

- A semijoin of R *with* S ($R \ltimes S$, $S \rtimes R$, $R \ltimes_F S$, $S \rtimes_F R$) returns all tuples of R that have a potential partner in S

- Example of a semi-natural join

R		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

$\ltimes$

S		
C	D	E
c_1	d_1	e_1
c_3	d_2	e_2

=

$R \ltimes S$		
A	B	C
a_1	b_1	c_1

- At the schema level
  - The assumptions are the same as for the natural join and theta join resp.
  - The result schema is $\mathcal{R}$ as the schema of relation R

- At the data level
  - $R \ltimes S = \pi_{\mathcal{R}}(R \bowtie S) = S \rtimes R$ — semi-natural join
  - $R \ltimes_F S = \pi_{\mathcal{R}}(R \bowtie_F S) = S \rtimes_F R$ — semi-theta join

Antijoin Operation

- An antijoin of R *with* S ($R \triangleright S$, $R \triangleright_F S$) is the complement of the semijoin and returns all tuples of R that do *not* have a potential partner in S
- At the schema level
  - The assumptions are the same as for the natural join and theta join resp.
  - The result schema is $\mathcal{R}$ as the schema of relation R
- At the data level
  - $R \triangleright S = R - R \ltimes S$ anti-natural join
  - $R \triangleright_F S = R - R \ltimes_F S$ anti-theta join

Example Queries (I)

- Query 1: Which students attend which lectures?
- RA expression: students $\bowtie$ attends $\bowtie$ lectures

students $\bowtie$ attends $\bowtie$ lectures						
reg-id	name	sem	id	title	credits	held_by
26120	Fichte	10	5001	foundations	4	2137
27550	Schopen.	12	5001	foundations	4	2137
27550	Schopen.	6	4052	logic	4	2125
...	...	...	...	...	...	...

- Query 2: Which lectures are held by which professors?
- RA expression: lectures $\bowtie$ $\rho_{\text{held\_by} \leftarrow \text{pers-id}}$(professors)
- Schema of the result relation (not shown) is (id, title, credits, held_by, name, room, rank)

Example Queries (II)

- Query 3: Find the personnel ids of all C4 professors who held at least one lecture, and assign the name *R* to the result relation.
- RA expression:

$$\rho_R(\underbrace{\pi_{\text{pers-id}}(\rho_{\text{pers-id}\leftarrow\text{held\_by}}(\text{lectures}))}_{\text{Personnel ids of professors who hold lectures}} \cap \underbrace{\pi_{\text{pers-id}}(\sigma_{\text{rank="C4"}}(\text{professors}))}_{\text{Personnel ids of C4 professors}})$$

R
pers-id
2125
2126
2137

- Query 4: Find the registration ids of those students who have attended *all* lectures with four credits.
- RA expression: attends $\div\ \pi_{\text{id}}(\sigma_{\text{credits=4}}(\text{lectures}))$
- The result schema has only the attribute "reg-id"; the result relation is empty

Relational Algebra Query Examples (I)

- The following examples demonstrate how colloquial queries can be translated into Relational Algebra expressions (queries)

- The examples also show the possible large complexity of Relational Algebra expressions

- We assume the following database schema

  Flights(flightNumber, from, to, distance)

  Aircraft(planeID, planeName, range)

  Pilots(employeeID, planeID)

  Employees(employeeID, employeeName, salary)

  All attributes are of type *string* except for the attributes *distance*, *range*, and *salary* that are of type *integer*

Relational Algebra Query Examples (II)

- Flights(flightNumber, from, to, distance)

  Aircraft(planeID, planeName, range)

  Pilots(employeeID, planeID)

  Employees(employeeID, employeeName, salary)

- Query 1: Find the employee identifiers of pilots who can operate the aircrafts of types "Boeing 747" and "Boeing 777".

- RA expression:

$$\pi_{employeeID}(\sigma_{planeName\ =\ \text{'Boeing 747'}}(Aircraft \bowtie Pilots))$$

$$\cap$$

$$\pi_{employeeID}(\sigma_{planeName\ =\ \text{'Boeing 777'}}(Aircraft \bowtie Pilots))$$

Relational Algebra Query Examples (III)

- Flights(flightNumber, from, to, distance)
  Aircraft(planeID, planeName, range)
  Pilots(employeeID, planeID)
  Employees(employeeID, employeeName, salary)

- Query 2: Find the identifiers of the pilot(s) with the highest salary.

- RA expression:

  $\rho_{AllPilots}(\pi_{employeeID}(Pilots) \bowtie Employees)$

  $(\pi_{employeeID}(AllPilots) -$
  $\pi_{P2.employeeID}(\sigma_{P1.salary > P2.salary}(\rho_{P1}(AllPilots) \times \rho_{P2}(AllPilots))))$

Relational Algebra Query Examples (IV)

- Flights(flightNumber, from, to, distance)

  Aircraft(planeID, planeName, range)

  Pilots(employeeID, planeID)

  Employees(employeeID, employeeName, salary)

- Query 3: Find the airplane identifiers of aircrafts that cannot fly non-stop from ATL to JFK.

- RA expression:

  $\pi_{\text{planeID}}(\sigma_{\text{range} < \text{distance}}(\text{Aircraft} \times \sigma_{\text{from} = \text{'ATL'} \wedge \text{to} = \text{'JFK'}}(\text{Flights})))$

Relational Algebra Query Examples (V)

- Flights(flightNumber, from, to, distance)
  Aircraft(planeID, planeName, range)
  Pilots(employeeID, planeID)
  Employees(employeeID, employeeName, salary)

- Query 4: Find the names of pilots who can operate planes with a range greater than or equal to 1,500 miles but cannot operate "Boeing 747" and "Boeing 777" aircrafts.

- RA expression:

$$
\begin{aligned}
&\pi_{employeeName}(\text{Employees} \bowtie \\
&\quad (\pi_{employeeID}(\sigma_{range >= 1500}(\text{Aircraft} \bowtie \text{Pilots})) \\
&\quad - \\
&\quad (\pi_{employeeID}(\sigma_{planeName = \text{'Boeing 747'}}(\text{Aircraft} \bowtie \text{Pilots})) \\
&\quad \cap \\
&\quad \pi_{employeeID}(\sigma_{planeName = \text{'Boeing 777'}}(\text{Aircraft} \bowtie \text{Pilots})))))
\end{aligned}
$$

Relational Algebra Query Examples (VI)

- Flights(flightNumber, from, to, distance)
  Aircraft(planeID, planeName, range)
  Pilots(employeeID, planeID)
  Employees(employeeID, employeeName, salary)

- Query 5: Find the flight numbers of flights that can be piloted by every pilot whose salary is under $100,000.

- RA expression:

$$\pi_{\text{flightNumber, planeID}}(\sigma_{\text{distance <= range}}(\text{Flights} \times \text{Aircraft})) \div \pi_{\text{planeID}}(\sigma_{\text{salary < 100000}}(\text{Pilots} \bowtie \text{Employees})))$$

Relational Algebra Query Examples (VII)

- Flights(flightNumber, from, to, distance)
  Aircraft(planeID, planeName, range)
  Pilots(employeeID, planeID)
  Employees(employeeID, employeeName, salary)

- Query 6: Find the identifiers of the pilots that fly all aircrafts.

- RA expression:
  $\text{Pilots} \div \pi_{\text{planeID}}(\text{Aircraft})$

Relational Algebra Query Examples (VIII)

- Flights(flightNumber, from, to, distance)
  Aircraft(planeID, planeName, range)
  Pilots(employeeID, planeID)
  Employees(employeeID, employeeName, salary)

- Query 7: Find the names of all employees who are not pilots.

- RA expression:
  $\pi_{employeeName}(Employee \bowtie (\pi_{employeeID}(Employee) - \pi_{employeeID}(Pilot)))$