relational algebra & calculus
Relational DB: The Origins

Frege: FO logic

Tarski: Algebra for FO

Codd: Relational databases
relational calculus
Relational Calculus (aka FO)

• Models data manipulation core of SQL
  Idea: specify “what” not “how”

• General form:
  \{t | property (t)\}

• property (t) is described by a language based on predicate calculus (first-order logic)
Relational Calculus Example

Display the movie table

In SQL

```
SELECT *
FROM Movie
```

In words

*(making answer tuple explicit)*

The answer consists of tuples $m$ such that $m$ is a tuple in Movie

Need to say

“tuple $m$ is in relation $R$”: $m \in R$
Relational Calculus Example

Find the directors and actors of currently playing movies

In SQL

```
SELECT  m.Director, m.Actor
FROM    movie m, schedule s
WHERE   m.Title = s.Title
```

In words (making answer tuple explicit)

“The answer consists of tuples t s.t.
there exist tuples m in movie and s in schedule for which
t.Director = m.Director and t.Actor = m.Actor and m.Title = s.Title”

Need to say

“there exists a tuple x in relation R”:  \( \exists x \in R \)
Refer to the value of attribute A of tuple x:  \( x(A) \)

Boolean combinations
Relational Calculus Example

Find the directors and actors of currently playing movies

Need to say

“there exists a tuple x in relation R”:  \( \exists \ x \in R \)
Refer to the value of attribute A of tuple x:  \( x(A) \)
Boolean combinations

In logic notation (tuple relational calculus)

\[ \{ \; t: \text{Director, Actor} \mid \exists \ m \in \text{movie} \; \exists \ s \in \text{schedule} \; [\; t(\text{Director}) = m(\text{Director}) \land t(\text{Actor}) = m(\text{Actor}) \land m(\text{Title}) = s(\text{Title}) \; ] \; \} \]
Quantifiers

∃ m ∈ R: Existential quantification
“there exists some tuple m in relation R ....”

Sometimes need to say:
“for every tuple m ....”

e.g., “every director is also an actor”

Need to say:
“for every tuple m in movie there exists a tuple t in movie
Such that m.Director = t.Actor”

∀ m ∈ movie ∃ t ∈ movie [ m(Director) = t(Actor) ]

(The answer to this query is true or false)

∀ m ∈ R: Universal quantification
“for every tuple m in relation R ....”
Tuple Relational Calculus

• In the style of SQL: language talks about tuples

• What you can say:
  - Refer to tuples: tuple variables \( t, s, \ldots \)
  - A tuple \( t \) belongs to a relation \( R \): \( t \in R \)
  - Conditions on attributes of a tuple \( t \) and \( s \):
    • \( t(A) = (\neq)(\geq) \) constant
    • \( t(A) = s(B) \)
    • \( t(A) \neq s(B) \)
    • etc.

• Simple expressions above: atoms
Tuple Relational Calculus

• Combine properties using Boolean operators
  $\land$, $\lor$, $\neg$
  (abbreviation: $p \rightarrow q \equiv \neg p \lor q$)

• Quantifiers
  there exists: $\exists t \in R \varphi(t)$
  for every: $\forall t \in R \varphi(t)$
  where $\varphi(t)$ a formula in which t not quantified (it is “free”)

More on quantifiers

• **Scope** of quantifier:
  - scope of $\exists t \in \mathbb{R} \varphi(t)$ is $\varphi$
  - scope of $\forall t \in \mathbb{R} \varphi(t)$ is $\varphi$

• **Free** variable:
  - not in scope of any quantifier
  - free variables are the “parameters” of the formula

• **Rule:** in quantification $\exists t \in \mathbb{R} \varphi(t)$, $\forall t \in \mathbb{R} \varphi(t)$
  - $t$ must be free in $\varphi$
Quantifier Examples

\{ t: \text{Director, Actor} \mid \exists m \in \text{movie} \exists s \in \text{schedule} \\
[ t(\text{Director}) = m(\text{Director}) \land t(\text{Actor}) = m(\text{Actor}) \land m(\text{Title}) = s(\text{Title}) ] \} \\

[ t(\text{Director}) = m(\text{Director}) \land t(\text{Actor}) = m(\text{Actor}) \land m(\text{Title}) = s(\text{Title}) ] \\
\text{free: } t, m, s

\exists s \in \text{schedule} \\
[ t(\text{Director}) = m(\text{Director}) \land t(\text{Actor}) = m(\text{Actor}) \land m(\text{Title}) = s(\text{Title}) ] \\
\text{free: } t, m

\exists m \in \text{movie} \exists s \in \text{schedule} \\
[ t(\text{Director}) = m(\text{Director}) \land t(\text{Actor}) = m(\text{Actor}) \land m(\text{Title}) = s(\text{Title}) ] \\
\text{free: } t
A statement about numbers:

$$\exists x \ \forall y \ \forall z \ [ x = y \ast z \rightarrow ((y = 1) \lor (z = 1))]$$

“there exists at least one prime number x”

A “query” on numbers:

$$\varphi(x) : \ \forall y \ \forall z \ [ x = y \ast z \rightarrow ((y = 1) \lor (z = 1))]$$

This defines the set \{x | \varphi(x)\} of prime numbers. It consists of all x that make \varphi(x) true.
Semantics of Tuple Calculus

• Active domain:
  A set of values in the database, or mentioned in the query result. Tuple variables range over the active domain

• Note:
  A query without free variables always evaluates to true or false

  e.g., “Sky is by Berto” is expressed without free variables:
  \[ \exists m \in \text{movie} \ [m(\text{title}) = \text{“Sky”} \land m(\text{director}) = \text{“Berto”}] \]
  This statement is true or false
Tuple Calculus Query

\{t: <att> | \varphi(t)\}

where \( \varphi \) is a calculus formula with only one free variable \( t \) produces as answer a table with attributes \( <att> \) consisting of all tuples \( v \) in active domain with make \( \varphi(v) \) true

Note:
\( \varphi(v) \) has no free variables so it evaluates to true or false
Movie Examples Revisited

Find titles of currently playing movies

```sql
select Title
from Schedule
```

Find the titles of all movies by “Berto”

```sql
select Title
from Movie
where Director=‘Berto’
```

Find the titles and the directors of all currently playing movies

```sql
select Movie.Title, Director
from Movie, Schedule
where Movie.Title = Schedule.Title
```
Movie Examples Revisited

Find titles of currently playing movies

\{t: title \mid \exists s \in schedule \ [s(title) = t(title)]\} 

Find the titles of all movies by “Berto”

\{t: title \mid \exists m \in movie \ [m(director) = “Berto” \land t(title) = m(title)]\} 

Find the titles and the directors of all currently playing movies

\{t: title, director \mid \exists s \in schedule \exists m \in movie \ [s(title) = m(title) \land t(title) = m(title) \land t(director) = m(director)]\}
Movie Examples Revisited

• Find actors playing in every movie by Berto

{a: actor | ∃y ∈ movie [a(actor) = y(actor) ∧
∀m ∈ movie [m(director) = “Berto” → ∃t ∈ movie (m(title) =
t(title) ∧ t(actor) = y(actor))]}]}
Movie Examples Revisited

- Find actors playing in every movie by Berto

\[ \{a: \text{actor} \mid \exists y \in \text{movie} [a(\text{actor}) = y(\text{actor}) \land \forall m \in \text{movie} [m(\text{director}) = \text{“Berto”} \rightarrow \exists t \in \text{movie} (m(\text{title}) = t(\text{title}) \land t(\text{actor}) = y(\text{actor}))]]} \]

Typical use of $\forall$:

$\forall m \in R \ [ \text{filter}(m) \rightarrow \text{property}(m)]$

Intuition: check $\text{property}(m)$ for those $m$ that satisfy $\text{filter}(m)$

we don’t care about the $m$’s that do not satisfy $\text{filter}(m)$
Movie Examples Revisited

- Find actors playing in every movie by Berto

\{a: actor \mid \exists y \in \text{movie} \ [a(\text{actor}) = y(\text{actor}) \land \\
\forall m \in \text{movie} \ [m(\text{director}) = \text{"Berto"} \rightarrow \exists t \in \text{movie} \ (m(\text{title}) = \\
\text{t(title)} \land t(\text{actor}) = y(\text{actor})))\} \}

Is this correct?

\{a: actor \mid \exists y \in \text{movie} \ [a(\text{actor}) = y(\text{actor}) \land \\
\forall m \in \text{movie} \ \exists t \in \text{movie} \ [m(\text{director}) = \text{"Berto"} \rightarrow (m(\text{title}) = \\
\text{t(title)} \land t(\text{actor}) = y(\text{actor})))\} \}

A: YES   B: NO
Movie Examples Revisited

Is this correct?

\{a: \text{actor} \mid \exists y \in \text{movie} \ [a(\text{actor}) = y(\text{actor}) \land \forall m \in \text{movie} \ \exists t \in \text{movie} \ [m(\text{director}) = \text{“Berto”} \rightarrow (m(\text{title}) = t(\text{title}) \land t(\text{actor}) = y(\text{actor}))])}\}

A: NO  B: NO

\exists t \ (\varphi \lor \psi) = \exists t \ \varphi \lor \exists t \ \psi

\exists t \ \varphi = \varphi \text{ if } t \text{ does not occur in } \varphi

Is the following correct:

\exists t \ (\varphi \land \psi) = \exists t \ \varphi \land \exists t \ \psi

A: YES  B: NO
Movie Examples Revisited

Correct:
{a: actor \mid \exists y \in \text{movie} \ [a(\text{actor}) = y(\text{actor}) \land
\forall m \in \text{movie} \ \exists t \in \text{movie} \ [m(\text{director}) = "Berto" \rightarrow (m(\text{title}) =
t(\text{title}) \land t(\text{actor}) = y(\text{actor}))]}\}

\exists t \ (\varphi \lor \psi) = \exists t \varphi \lor \exists t \psi

\exists t \varphi = \varphi \text{ if } t \text{ does not occur in } \varphi

\exists t \in \text{movie} \ [m(\text{director}) = "Berto" \rightarrow (m(\text{title}) =
t(\text{title}) \land t(\text{actor}) = y(\text{actor}))] =
\exists t \in \text{movie} \ [\neg m(\text{director}) = "Berto" \lor (m(\text{title}) =
t(\text{title}) \land t(\text{actor}) = y(\text{actor}))] =
[\exists t \in \text{movie} \ (\neg m(\text{director}) = "Berto") \lor \exists t \in \text{movie} \ (m(\text{title}) =
t(\text{title}) \land t(\text{actor}) = y(\text{actor}))] =
[\neg m(\text{director}) = "Berto" \lor \exists t \in \text{movie} \ (m(\text{title}) =
t(\text{title}) \land t(\text{actor}) = y(\text{actor}))] =
[m(\text{director}) = "Berto" \rightarrow \exists t \in \text{movie} \ (m(\text{title}) =
t(\text{title}) \land t(\text{actor}) = y(\text{actor}))]
Movie Examples Revisited

Correct:
\[
\{a: \text{actor} \mid \exists y \in \text{movie} \ [a(\text{actor}) = y(\text{actor}) \land \\
\forall m \in \text{movie} \ \exists t \in \text{movie} \ [m(\text{director}) = "Berto" \rightarrow (m(\text{title}) = \\
t(\text{title}) \land t(\text{actor}) = y(\text{actor})))]}\]

Is this also correct (can we switch $\forall$ and $\exists$)?

\[
\{a: \text{actor} \mid \exists y \in \text{movie} \ [a(\text{actor}) = y(\text{actor}) \land \\
\exists t \in \text{movie} \ \forall m \in \text{movie} \ [m(\text{director}) = "Berto" \rightarrow (m(\text{title}) = \\
t(\text{title}) \land t(\text{actor}) = y(\text{actor})))]}\]

A: YES  B: NO
Tuple Calculus and SQL

• Example:
  “Find theaters showing movies by Bertolucci”:

SQL:

```
SELECT s.theater
FROM schedule s, movie m
WHERE s.title = m.title AND m.director = "Bertolucci"
```

tuple calculus:

```
{ t: theater | ∃ s ∈ schedule ∃ m ∈ movie [ t(theater) = s(theater) ∧ s(title) = m(title) ∧ m(director) = Bertolucci ] }
```
Basic SQL Query

SQL

- **SELECT** $A_1, \ldots, A_n$
- **FROM** $R_1, \ldots, R_k$
- **WHERE** $\text{cond}(R_1, \ldots, R_k)$

Tuple Calculus

- $\{t : A_1, \ldots, A_n \mid \exists r_1 \in R_1 \ldots \exists r_k \in R_k \ [\land_j t(A_j) = r_{ij}(A_j) \land \text{cond}(r_1, \ldots, r_k)]\}$
- **Note:**
  - Basic SQL query uses only $\exists$
  - No explicit construct for $\forall$
Using Tuple Calculus to Formulate SQL Queries

Example: “Find actors playing in every movie by Berto”

• Tuple calculus
  \{a: actor | \exists y \in \text{movie} [a(\text{actor}) = y(\text{actor}) \land \\
  \forall m \in \text{movie} [m(\text{dir}) = “Berto” \rightarrow \exists t \in \text{movie} (m(\text{title}) = \\
  t(\text{title}) \land t(\text{actor}) = y(\text{actor}))]]\}

• Eliminate \( \forall \):
  \{a: actor | \exists y \in \text{movie} [a(\text{actor}) = y(\text{actor}) \land \\
  \neg \exists m \in \text{movie} [m(\text{dir}) = “Berto” \land \neg \exists t \in \text{movie} (m(\text{title}) = \\
  t(\text{title}) \land t(\text{actor}) = y(\text{actor}))]]\}

• Rule: \( \forall x \in R \varphi(x) \equiv \neg \exists x \in R \neg \varphi(x) \)
  
  “every x in R satisfies \( \varphi(x) \) iff there is no x in R that violates \( \varphi(x) \)”
Convert to SQL query

• Basic rule: one level of nesting for each “¬∃”

\{
  a: actor | ∃y ∈ movie [a(actor) = y(actor) ∧
  ¬∃m ∈ movie [m(dir) = “Berto” ∧ ¬∃t ∈ movie (m(title) = t(title)
  ∧ t(actor) = y(actor))]]}

SELECT y.actor  FROM movie y
WHERE NOT EXISTS
  (SELECT * FROM movie m
WHERE m.dir = ‘Berto’ AND
NOT EXISTS
  (SELECT *
FROM movie t
WHERE m.title = t.title  AND t.actor = y.actor )))
Another possibility (with similar nesting structure)

```sql
SELECT actor FROM movie
WHERE actor NOT IN
  (SELECT s.actor
   FROM movie s, movie m
   WHERE m.dir = 'Berto'
   AND s.actor NOT IN
     (SELECT t.actor
      FROM movie t
      WHERE m.title = t.title ))
```

- Note: Calculus is more flexible than SQL because of the ability to mix $\exists$ and $\forall$ quantifiers
relational algebra
Query Processing

3 steps:
• Parsing & Translation
• Optimization
• Evaluation
Relational Algebra

- Simple set of algebraic operations on relations

Journey of a query

<table>
<thead>
<tr>
<th>SQL</th>
<th>select ... from...where</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relational algebra</td>
<td>$\pi_{13}(P \bowtie Q) \bowtie ...$</td>
</tr>
<tr>
<td>Query rewriting</td>
<td>$\pi_{14}(P \bowtie S) \bowtie Q \bowtie R$</td>
</tr>
</tbody>
</table>

- We use set semantics (no duplicates) and no nulls
- There are extensions with bag semantics and nulls
Relational Algebra

Projection

Eliminate some columns

\[ \pi_X(R) \]
Display only attributes X of relation R

where R: table name & X ⊆ attributes(R)

Example:
Find titles of current movies

\[ \pi_{TITLE}(\text{SCHEDULE}) \]
## Relational Algebra

### Projection

Eliminate some columns

<table>
<thead>
<tr>
<th>$\pi_X(R)$</th>
<th>Display only attributes $X$ of relation $R$</th>
</tr>
</thead>
</table>
| $\pi_X(R)$ | $\pi_X(R)$ =  
| $\pi_{AB}(R)$ | $\pi_{AB}(R)$ =  |

where $R$: table name & $X \subseteq \text{attributes}(R)$

Example:

<table>
<thead>
<tr>
<th>R</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

- $\pi_A(R)$ =
  - A
  - 0
  - 1
- $\pi_{AB}(R)$ =
  - A
  - B
  - 0 1
  - 0 2
  - 1 3

No repetitions of tuples!
Relational Algebra

Selection

Compute set union

\( \sigma_{\text{cond}}(R) \) Select tuples of R satisfying condition cond

where cond: condition involving only attributes of R (e.g., attr = value, attr ≠ value, attr1 = attr2, attr1 ≠ attr2, etc.)

Example:

\( \sigma_{A=0}(R) = \)

\[
\begin{array}{ccc}
0 & 1 & 2 \\
0 & 2 & 2 \\
1 & 3 & 1 \\
0 & 1 & 3 \\
\end{array}
\]

\( \sigma_{B=C}(R) = \)

\[
\begin{array}{ccc}
0 & 1 & 2 \\
0 & 2 & 2 \\
0 & 1 & 3 \\
\end{array}
\]
Relational Algebra

Selection

Compute set union

\[ \sigma_{\text{cond}}(R) \]

Select tuples of R satisfying condition cond

\[ \sigma_{\text{cond}}(R) \]

*where cond: condition involving only attributes of R*

(e.g., attr = value, attr ≠ value, attr1 = attr2, attr1 ≠ attr2, etc.)

Example:

\[ \sigma_{A \neq 0}(R) = \]

\[
\begin{array}{ccc}
0 & 1 & 2 \\
0 & 2 & 2 \\
1 & 3 & 1 \\
0 & 1 & 3 \\
\end{array}
\]

\[
\begin{array}{ccc}
A & B & C \\
1 & 3 & 1 \\
\end{array}
\]
Relational Algebra

Union

Compute set union

\[ R \cup S \]

Union of sets of tuples in R and S

where R, S: tables with same attributes

Example:

\[
\begin{array}{cc}
R & S \\
A & A \\
B & B \\
\alpha & \alpha \\
1 & 2 \\
\alpha & \beta \\
2 & 3 \\
\beta & \\
1 & \\
\end{array}
\]

\[ R \cup S = \]

\[
\begin{array}{cc}
A & B \\
\alpha & \alpha \\
\alpha & \beta \\
\beta & \\
1 & 1 \\
2 & 3 \\
\end{array}
\]
Relational Algebra

Difference

Compute set difference

<table>
<thead>
<tr>
<th>R - S</th>
<th>Difference of sets of tuples in R and S</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>where R, S: tables with same attributes</td>
</tr>
</tbody>
</table>

Example:

<table>
<thead>
<tr>
<th>R</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>α</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>β</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>β</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R - S</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>β</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Relational Algebra

Join

Compute join

\[ R \bowtie S \]

Natural Join of \( R, S \)

where \( R, S: \) tables

Example:

\[
\begin{array}{c|c|c}
R & A & B \\
S & B & C \\
\end{array}
\]

\[ R \bowtie S = A \ B \ C \]

Note: More than one common attributes allowed!
Relational Algebra

Join

Compute join

\[ R \Join S \]

Natural Join of R, S

where \( R, S: \) tables

Example:

<table>
<thead>
<tr>
<th>R ( A )</th>
<th>B</th>
<th>S ( B )</th>
<th>C</th>
<th>( R \Join S ) =</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0 1 2</td>
<td>0 1 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>0 1 3</td>
<td>0 1 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>0 2 2</td>
<td>0 2 2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Definition of Join

Let $r$ and $s$ be relations on schemas $R$ and $S$ respectively. Then, $r \bowtie s$ is a relation with attributes $\text{att}(R) \cup \text{att}(S)$ obtained as follows:

Consider each pair of tuples $t_r$ from $r$ and $t_s$ from $s$.

If $t_r$ and $t_s$ have the same value on each of the attributes in $\text{att}(R) \cap \text{att}(S)$, add a tuple $t$ to the result, where

- $t$ has the same value as $t_r$ on $r$
- $t$ has the same value as $t_s$ on $s$

Note: if $R \cap S$ is empty, the join consists of all combinations of tuples from $R$ and $S$, i.e. their cross-product
Attribute Renaming

Rename attributes

\[ \delta_{A_1 \rightarrow A_2}(R) \]

Change name of attribute A1 in rel. R to A2

where \( R \): relation and A1: attribute in R

Example:

\[ \begin{array}{c|c|c} 
A & B \\
\hline 
\alpha & 1 \\
\alpha & 2 \\
\beta & 1 \\
\end{array} \]

\[ \delta_{A \rightarrow C}(R) = \begin{array}{c|c} 
C & B \\
\hline 
\alpha & 1 \\
\alpha & 2 \\
\beta & 1 \\
\end{array} \]

Contents remain unchanged!

Note: Can rename several attributes at once
Relational Algebra

• Basic set of operations: \( \pi, \sigma, \cup, -, \bowtie, \delta \)

• Back to movie example queries:

  1. Titles of currently playing movies:
     \( \pi_{\text{TITLE}}(\text{schedule}) \)

  2. Titles of movies by Berto:
     \( \pi_{\text{TITLE}}(\sigma_{\text{DIR}=\text{BERTO}}(\text{movie})) \)

  3. Titles and directors of currently playing movies:
     \( \pi_{\text{TITLE}, \text{DIR}}(\text{movie} \bowtie \text{schedule}) \)
4. Find the pairs of actors acting together in some movie

\[ \pi_{\text{actor1, actor2}} (\delta_{\text{actor} \rightarrow \text{actor1}} (\text{movie}) \bowtie \delta_{\text{actor} \rightarrow \text{actor2}} (\text{movie})) \]

5. Find the actors playing in every movie by Berto

\[ \pi_{\text{actor}} (\text{movie}) - \pi_{\text{actor}} [(\pi_{\text{actor}} (\text{movie}) \bowtie \pi_{\text{title}} (\sigma_{\text{dir} = \text{BERTO}} (\text{movie}))) - \pi_{\text{actor,title}} (\text{movie})] \]

actors for which there is a movie by Berto in which they do not act

In this case (not in general): Same as cartesian product
Relational Algebra

Cartesian Product

Compute cartesian product

<table>
<thead>
<tr>
<th>R × S</th>
<th>Cartesian Product of R, S</th>
</tr>
</thead>
<tbody>
<tr>
<td>where R, S: tables</td>
<td></td>
</tr>
</tbody>
</table>

Example:

<table>
<thead>
<tr>
<th>R</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R × S</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

Same as R × S, when R and S have no common attributes
Relational Algebra

Cartesian Product

Compute cartesian product

\[ R \times S \]

Cartesian Product of R, S

where R, S: tables

Example:

<table>
<thead>
<tr>
<th>R</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S</th>
<th>A</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

\[ R \times S = \]

<table>
<thead>
<tr>
<th>R.A</th>
<th>B</th>
<th>S.A</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

If 2 attributes in R, S have the same name A, they are renamed to R.A and S.A in the output
Other useful operations

- Intersection $R \cap S$
- Division (Quotient) $R \div S$

$R \div S$: \{a \mid <a, b> \in R \text{ for every } b \in S\}

Example:

<table>
<thead>
<tr>
<th>R</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>α</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>β</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>α</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>β</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>γ</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>α</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>α</td>
</tr>
<tr>
<td></td>
<td>β</td>
</tr>
</tbody>
</table>

$R \div S = \begin{array}{|c|}
\hline
0 \\
\hline
1 \\
\hline
\end{array}$
Another Division Example

Find the actors playing in every movie by Berto

\[ \pi_{\text{TITLE, ACTOR}}(\text{movie}) \div \pi_{\text{TITLE}}(\sigma_{\text{DIR=BERTO}}(\text{movie})) \]
Division by multiple attributes

Relations \( r, s \):

\[
\begin{array}{|c|c|c|c|c|}
\hline
A & B & C & D & E \\
\hline
\alpha & a & \alpha & a & 1 \\
\alpha & a & \gamma & a & 1 \\
\alpha & a & \gamma & b & 1 \\
\beta & a & \gamma & a & 1 \\
\beta & a & \gamma & b & 3 \\
\gamma & a & \gamma & a & 1 \\
\gamma & a & \gamma & b & 1 \\
\gamma & a & \beta & b & 1 \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
D & E \\
\hline
a & 1 \\
b & 1 \\
\hline
\end{array}
\]

\[ r \div s: \]

\[
\begin{array}{|c|c|c|}
\hline
A & B & C \\
\hline
\alpha & a & \gamma \\
\gamma & a & \gamma \\
\hline
\end{array}
\]
Relational Algebra

• Note:
  $\pi$ is like $\exists$ “there exists”…
  $\div$ is like $\forall$ “for all”…

• Expressing $\div$ using other operators:

\[
R \div S = \pi_A(R) - \pi_A((\pi_A(R) \bowtie S) - R)
\]

<table>
<thead>
<tr>
<th>R</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Similar to: \( \forall x \varphi(x) \equiv \neg \exists x \neg \varphi(x) \)
Calculus Vs. Algebra

- Theorem: Calculus and Algebra are equivalent
- Basic Correspondence:

<table>
<thead>
<tr>
<th>Algebra Operation</th>
<th>Calculus Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>π</td>
<td>∃</td>
</tr>
<tr>
<td>σ</td>
<td>t(A) comp c</td>
</tr>
<tr>
<td>∪</td>
<td>∨</td>
</tr>
<tr>
<td>△</td>
<td>∧</td>
</tr>
<tr>
<td>-</td>
<td>¬</td>
</tr>
<tr>
<td>+</td>
<td>∀</td>
</tr>
</tbody>
</table>
Example

• “Find theaters showing movies by Bertolucci”: SQL:
  
  • SELECT s.theater
    FROM schedule s, movie m
    WHERE s.title = m.title AND m.director = ‘Berto’

  tuple calculus:
  
  • \{ t: theater | \exists s \in schedule \exists m \in movie \left[ t(\text{theater}) = s(\text{theater}) \land s(\text{title}) = m(\text{title}) \land m(\text{director}) = \text{Berto} \right] \}

  relational algebra:
  
  \pi_{\text{theater}}(schedule \bowtie \sigma_{\text{dir} = \text{Berto}}(movie))

Note: number of items in FROM clause = (number of joins + 1)