Relational Algebra

Relational Algebra

- Procedural language
- Six basic operators
  - select: σ
  - project: $\prod$
  - union: $\cup$
  - set difference: –
  - Cartesian product: x
  - rename: ρ
- The operators take one or two relations as inputs and produce a new relation as the result

Composition of Operations

- Building expressions using multiple operations
- Example: $\sigma_{A=C}(r \times s)$

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

A	B	C	D	E
α	1	α	10	a
β	2	β	10	a
β	2	β	20	b

Rename Operation

- Name, and therefore to refer to, the results of relational-algebra expressions
  - Refer to a relation by more than one name
- Example: $\rho_X(E)$ returns the expression E under the name X
- If a relational-algebra expression E has arity n, then $\rho_{X(A_1, A_2, \ldots, A_n)}(E)$ returns the result of expression E under the name X, and with the attributes renamed to A_1 , A_2 , …., A_n

Banking Example

- branch (branch_name, branch_city, assets)
- customer (customer_name, customer_street, customer_city)
- account (account_number, branch_name, balance)
- loan (loan_number, branch_name, amount)
- depositor (customer_name, account_number)
- borrower (customer_name, loan_number)

Example Queries

- Find all loans of over $1200

$$\sigma_{amount\ >\ 1200}\ (loan)$$

- Find the loan number for each loan of an amount greater than $1200

$$\Pi_{loan\_number}\ (\sigma_{amount\ >\ 1200}\ (loan))$$

loan (loan_number, branch_name, amount)

Example Queries

- Find the names of all customers who have a loan, an account, or both, from the bank

$$\Pi_{customer\_name}(\textit{borrower}) \cup \Pi_{customer\_name}(\textit{depositor})$$

- Find the names of all customers who have a loan and an account at the bank

$$\Pi_{customer\_name}(\textit{borrower}) \cap \Pi_{customer\_name}(\textit{depositor})$$

depositor (customer_name, account_number)
borrower (customer_name, loan_number)

Example Queries

- Find the names of all customers who have a loan at the Perryridge branch

$$\Pi_{customer\_name}(\sigma_{branch\_name=\text{"Perryridge"}}(\sigma_{borrower.loan\_number = loan.loan\_number}(borrower \times loan)))$$

- Find the names of all customers who have a loan at the Perryridge branch but do not have an account at any branch of the bank

$$\Pi_{customer\_name}(\sigma_{branch\_name = \text{"Perryridge"}}(\sigma_{borrower.loan\_number = loan.loan\_number}(\text{borrower x loan}))) - \Pi_{customer\_name}(\text{depositor})$$

Example Queries

- Find the names of all customers who have a loan at the Perryridge branch
  - Answer 1

    $\Pi_{customer\_name}(\sigma_{branch\_name = \text{“Perryridge”}}($
    $\sigma_{borrower.loan\_number = loan.loan\_number}(borrower \times loan)))$

  - Answer 2

    $\Pi_{customer\_name}(\sigma_{loan.loan\_number = borrower.loan\_number}($
    $(\sigma_{branch\_name = \text{“Perryridge”}}(loan)) \times borrower))$

Example Queries

- Find the largest account balance
  - Aggregate max is not directly supported in relational algebra
  - Find those balances that are not the largest
    - Rename account relation as d so that we can compare each account balance with all the others
  - Use set difference to find the max balance accounts

$$\Pi_{balance}(account) - \Pi_{account.balance}(\sigma_{account.balance < d.balance}(account \times \rho_d(account)))$$

account (account_number, branch_name, balance)

Formal Definition

- A basic expression in the relational algebra consists of either one of the following:
  - A relation in the database
  - A constant relation
- Let E_1 and E_2 be relational-algebra expressions; the following are all relational-algebra expressions:
  - $E_1 \cup E_2$
  - $E_1 - E_2$
  - $E_1 \times E_2$
  - $\sigma_p (E_1)$, P is a predicate on attributes in E_1
  - $\prod_s(E_1)$, S is a list consisting of some of the attributes in E_1
  - $\rho_x(E_1)$, x is the new name for the result of E_1

Additional Operations

- The additional operations do not add any power to the relational algebra, but can simplify writing common queries
  - Set intersection
  - Natural join
  - Division
  - Assignment

Set-Intersection Operation – Example

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

A	B
α	2

$r \cap s$

Set-Intersection Operation

- $r \cap s = \{ t \mid t \in r \text{ and } t \in s \}$
  - In basic operators, we only have set difference but no intersection
- Assume:
  - r, s have the same arity
  - attributes of r and s are compatible
- $r \cap s = r - (r - s)$

Natural Join Operation – Example

A	*B*	*C*	*D*
α	*1*	α	a
β	*2*	γ	a
γ	*4*	β	b
α	*1*	γ	a
δ	*2*	β	b

r

B	*D*	*E*
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	$\in$

s

A	*B*	*C*	*D*	*E*
α	*1*	α	a	α
α	*1*	α	a	γ
α	*1*	γ	a	α
α	*1*	γ	a	γ
δ	*2*	β	b	δ

$r \bowtie s$

Natural-Join Operation

- Let r and s be relations on schemas R and S respectively. $r \bowtie s$ is a relation on schema $R \cup S$ 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 $R \cap S$, 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

Example

- R = (A, B, C, D)
- S = (E, B, D)
- Result schema = (A, B, C, D, E)
- r ⋈ s is defined as

  $\Pi_{r.A,\ r.B,\ r.C,\ r.D,\ s.E}\ (\sigma_{r.B\ =\ s.B\ \wedge\ r.D\ =\ s.D}\ (r\ \times\ s))$

Division Operation – Example

A	B
α	1
α	2
α	3
β	1
γ	1
δ	1
δ	3
δ	4
$\in$	6
$\in$	1
β	2

r

B
1
2

s

A
α
β

$r \div s$

Division Operation

- Let r and s be relations on schemas R and S respectively where $R = (A_1, \ldots, A_m, B_1, \ldots, B_n)$ and $S = (B_1, \ldots, B_n)$
  - The result of $r \div s$ is a relation on schema R – S = (A1, …, Am)
  - $r \div s = \{ t \mid t \in \Pi_{R-S}(r) \wedge \forall u \in s \, (tu \in r) \}$, where tu means the concatenation of tuples t and u to produce a single tuple
- Suited to queries that include the phrase “for all”

Another Division Example

A	*B*	*C*	*D*	*E*
α	a	α	a	*1*
α	a	γ	a	*1*
α	a	γ	b	*1*
β	a	γ	a	*1*
β	a	γ	b	*3*
γ	a	γ	a	*1*
γ	a	γ	b	*1*
γ	a	β	b	*1*

r

D	*E*
a	*1*
b	*1*

s

A	*B*	*C*
α	a	γ
γ	a	γ

$r \div s$

Properties of Division Operation

- Let $q = r \div s$, q is the largest relation satisfying $q \times s \subseteq r$
- Let r(R) and s(S) be relations, and let $S \subseteq R$, $r \div s = \Pi_{R-S}(r) - \Pi_{R-S}((\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r))$
  - $\Pi_{R-S,S}(r)$ simply reorders attributes of r
  - $\Pi_{R-S}(\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r))$ gives those tuples t in $\Pi_{R-S}(r)$ such that for some tuple $u \in s$, $tu \notin r$

Assignment Operation

- The assignment operation ($\leftarrow$) provides a convenient way to express complex queries
  - Write query as a sequential program consisting of a series of assignments followed by an expression whose value is displayed as a result of the query
  - Assignment must always be made to a temporary relation variable
- Example: compute $r \div s$
  - $temp_1 \leftarrow \Pi_{R-S}(r)$, $temp_2 \leftarrow \Pi_{R-S}((temp_1 \times s) - \Pi_{R-S,S}(r))$
    $result = temp_1 - temp_2$
- The result to the right of the $\leftarrow$ is assigned to the relation variable on the left of the $\leftarrow$
  - May use variable in subsequent expressions

Bank Example Queries

- Find the names of all customers who have a loan and an account at bank

$$\Pi_{customer\_name}\ (borrower) \cap \Pi_{customer\_name}\ (depositor)$$

- Find the name of all customers who have a loan at the bank and the loan amount

$$\Pi_{customer\text{-}name,\ loan\text{-}number,\ amount}\ (borrower \bowtie loan)$$

Bank Example Queries

- Find all customers who have an account from at least the "Downtown" and the "Uptown" branches
  - Answer 1

$$\Pi_{customer\_name}(\sigma_{branch\_name = \text{“Downtown”}}(depositor \bowtie account)) \cap$$

$$\Pi_{customer\_name}(\sigma_{branch\_name = \text{“Uptown”}}(depositor \bowtie account))$$

  - Answer 2: using a constant relation

$$\Pi_{customer\_name,\ branch\_name}(depositor \bowtie account)$$
$$\div\ \rho_{temp(branch\_name)}(\{(\text{“Downtown”}),\ (\text{“Uptown”})\})$$

Example Queries

- Find all customers who have an account at all branches located in Brooklyn city

$$\Pi_{customer\_name,\ branch\_name}\ (depositor \bowtie account) \div \Pi_{branch\_name}\ (\sigma_{branch\_city\ =\ \text{"Brooklyn"}}\ (branch))$$

Summary

- Examples of relational algebra expressions
- Additional operators
  - Do not add any power to the relational algebra, but can simplify writing common queries
  - Set intersection
  - Natural join
  - Division
  - Assignment

To-Do List

- Translate the relational algebra expression examples into SQL
- What can you observe?