Relational Algebra

Relational Algebra

- Procedural language
- Six basic operators
 - select: σ
 - project: ∏
 - union: ∪
 - 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: $\sigma_{A=C}(r \times s)$

| Α | В | | |
|---|---|--|--|
| α | 1 | | |
| β | 2 | | |
| r | | | |

| С | D | Ε |
|--------------------------|----------------------|-------------|
| β β γ | 10 10 20 10 | a a b |
| / | S | |

| Α | В | С | D | E |
|----------|---|----------|----|---|
| α | 1 | α | 10 | а |
| α | 1 | β | 10 | a |
| α | 1 | β | 20 | b |
| α | 1 | γ | 10 | b |
| β | 2 | α | 10 | а |
| β | 2 | β | 10 | a |
| β | 2 | β | 20 | b |
| β | 2 | γ | 10 | b |

| Α | В | С | D | E |
|---------|---|-------------------|----|---|
| α | 1 | α | 10 | а |
| β | 2 | $\mid \beta \mid$ | 10 | а |
| 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: ρ_x(E) returns the expression E under the name X
- If a relational-algebra expression E has arity n, then $\rho_{x(A_1,A_2,...,A_n)}(E)$ 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)

Find all loans of over \$1200

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

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

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

loan (loan_number, branch_name, amount)

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

```
\Pi_{customer\_name}(borrower) \cup \Pi_{customer\_name}(depositor)
```

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

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

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

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

```
\Pi_{customer\_name}(\sigma_{branch\_name="Perryridge"}(\sigma_{borrower.loan\_number=loan.loan\_number}(borrower x 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

```
\begin{split} &\Pi_{customer\_name}(\sigma_{branch\_name = "Perryridge"}(\\ &\sigma_{borrower.loan\_number = loan.loan\_number}(borrower \ x \ loan))) \ - \\ &\Pi_{customer\_name}(depositor) \end{split}
```

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

```
\Pi_{customer\_name}(\sigma_{branch\_name = "Perryridge"})
\sigma_{borrower.loan\_number = loan.loan\_number}(borrower x loan)))
```

Answer 2

```
\Pi_{customer\_name}(\sigma_{loan.loan\_number = borrower.loan\_number} (\sigma_{branch\_name = "Perryridge"} (loan)) x borrower))
```

- 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 x <math>\rho_d (account)))
```

account (account_number, branch_name, balance)

Formal Definition

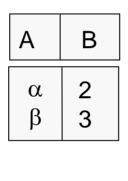
- A basic expression in the relational algebra consists of either one of the following:
 - A relation in the database
 - A constant relation
- Let E₁ and E₂ be relational-algebra expressions; the following are all relational-algebra expressions:
 - $E_1 \cup E_2$
 - $E_1 E_2$
 - $E_1 \times E_2$
 - $-\sigma_p$ (E₁), P is a predicate on attributes in E₁
 - $-\prod_s(E_1)$, 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

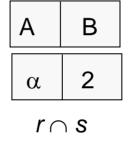
Additional Operations

- 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

| Α | В |
|---|---|
| α | 1 |
| α | 2 |
| β | 1 |
| , | • |





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

| Α | В | С | D |
|----------|---|----------|---|
| α | 1 | α | а |
| β | 2 | γ | а |
| γ | 4 | β | b |
| α | 1 | γ | а |
| δ | 2 | β | b |
| r | | | |

| В | D | E |
|-----------------------|-----------------------|---|
| 1 3 1 2 3 | a a a b b | $\begin{array}{c} \alpha \\ \beta \\ \gamma \\ \delta \\ \in \end{array}$ |
| | S | |

| Α | В | С | D | E |
|----------|---|----------|---|----------|
| α | 1 | α | а | α |
| α | 1 | α | а | γ |
| α | 1 | γ | а | α |
| α | 1 | γ | а | γ |
| δ | 2 | β | b | δ |
| rnac | | | | |

Natural-Join Operation

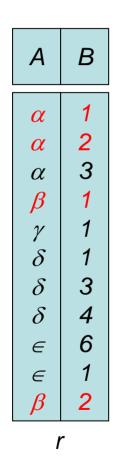
- Let r and s be relations on schemas R and S respectively. r ⋈ s is a relation on schema R
 S obtained as follows:
 - Consider each pair of tuples t_r from r and t_s from
 - If t_r and t_s have the same value on each of the attributes in R \cap S, add a tuple t to the result, where
 - t has the same value as t_r on r
 - t has the same value as t_s on s

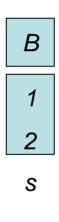
Example

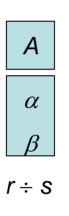
- R = (A, B, C, D)
- S = (E, B, D)
- Result schema = (A, B, C, D, E)
- r × s is defined as

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

Division Operation – Example



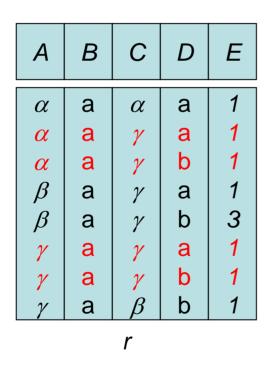


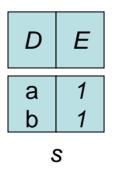


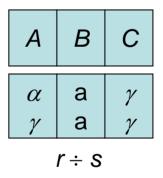
Division Operation

- Let r and s be relations on schemas R and S respectively where R = (A₁, ..., A_m, B₁, ..., B_n) and S = (B₁, ..., B_n)
 - The result of r ÷ s is a relation on schema R S= (A1, ..., Am)
 - $-r \div s = \{t \mid t \in \prod_{R-S}(r) \land \forall u \in s (tu \in r)\},$ 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







Properties of Division Operation

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

Assignment Operation

- The assignment operation (←) 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 ÷ s
 - temp₁ $\leftarrow \prod_{R-S}(r)$, temp₂ $\leftarrow \prod_{R-S}((temp_1 \times s) \prod_{R-S,S}(r))$ result = temp₁ - temp₂
- The result to the right of the ← is assigned to the relation variable on the left of the ←
 - 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

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

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 = "Downtown"} (depositor \bowtie account)) \cap 
\Pi_{customer\_name} (\sigma_{branch\_name = "Uptown"} (depositor \bowtie account))
```

Answer 2: using a constant relation

```
\Pi_{customer\_name, \ branch\_name}(depositor \bowtie account) \\ \div \rho_{temp(branch\_name)}(\{("Downtown"), ("Uptown")\})
```

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

```
\prod_{customer\_name, \ branch\_name} (depositor \bowtie account)
\vdots \prod_{branch\_name} (\sigma_{branch\_city = "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?