Relational Algebra

Relational algebra is the mathematical treatment of (finite) sets of tuples. A relation instance is represented by a finite set of tuples. These sets can be named by constants or variables.

Operations

union ($R \cup S$), intersection ($R \cap S$), (set) difference ($R - S$)

The relations must have schemas that have the same attributes, with the same domains, in the same order for these operations to be possible.

projection: $\pi_{A1, \ldots, An}(R)$

More Operations

selection: $\sigma_C(R)$

Condition C, a Boolean expression, refers to attributes of R.

Cartesian product: $R \cdot S$

If an attribute A appears in the schemas for both R and S, the corresponding attributes in the schema for the Cartesian product are often named R.A and S.A.

natural join: $R \bowtie S$

Only one copy of matched columns is retained.

More Operations (cont.)

theta join: $R \bowtie_C S$

This is a conditional join; C is the condition. It is a subset of the Cartesian product of R and S (not a subset of the natural join).

(re)naming: $\rho_{S(A1, \ldots, An)}(R)$

(Re)specifies schema for relation R. The S (new schema name) is often omitted.

Operation Combinations

Some operations can be defined in terms of other operations. Examples:

$R \cap S = R - (R - S)$

$R \bowtie_C S = \sigma_C(R \cdot S)$

$R \bowtie S = \pi_L(\sigma_C(R \cdot S))$ where, given that $A_1, \ldots, A_n$ are the attributes that are common to the schemas for both R and S, C is the condition $R.A_1 = S.A_1$ and . . . and $R.A_n = S.A_n$ and L is the list of attributes in the schema for R followed by the attributes in the schema for S that are not $A_1, \ldots, A_n$.

The Primitive Operatons

The primitive operations in relational algebra are **union, difference, selection, projection, product** and **renaming**. All other operations are definable in terms of these six. Some operations are natural enough (such as intersection, natural join) to name and use as if they were primitives.

Algebraic Computation

Assignment statements can be used to specify a sequence of computations that give the desired answer to a query.

$R := \sigma_{\text{μοδελ = ∍Τοψοτα∍}}(\text{Owner})$

$S := \sigma_{\text{ψεαρ = 2009}}(\text{Owner})$

$\text{Answer} = \pi_{\text{φιρστναμε, λαστναμε}}(R \cap S)$

“Who owns a 2009 Toyota?”

(The book gives a more elaborate notation, but even the authors don't use it when writing solutions to the exercises.)

Bank Database

Branch(name, city)

Customer(name, street, city)

Savings(account_no, branch_name, amount)

Depositor(name, account_no)

Loan(loan_no, branch_name, amount)

Borrower(name, loan_no)

A Query

Who are the customers with loans in the Newark branch?

R := $\sigma_{\text{Borrower.loan\_no = Loan.loan\_no}}$(Borrower · Loan)

S := $\sigma_{\text{branch\_name = 'Newark'}}$(R)

Answer := π_{name}(S)

Better:

R := $\sigma_{\text{branch\_name = 'Newark'}}$(Loan)

S := $\sigma_{\text{Borrower.loan\_no = Loan.loan\_no}}$(Borrower · R)

Answer := π_{name}(S)

Comparing Values

You can only compare values when they are in the same tuple.

What is the largest savings amount in the bank?

R := Savings

S := $\sigma_{\text{Savings.amount < R.amount}}$(Savings · R)

T := $\pi_{\Sigma\alpha\varpi\iota\nu\gamma\sigma.\alpha\mu o\upsilon\nu\tau}$(S) ← The values we don't want

V := $\pi_{\alpha\mu o\upsilon\nu\tau}$(Savings)

Answer := V − T

Algebra Based on Bags

For efficiency reasons, relational algebra is extended to use bags to represent relations. A **bag** is a collection of things that is unordered, but duplicates are allowed.

Avoiding duplicates every time we do a union, intersection, difference, projection, selection, product or join can be expensive, so it is often more efficient to let the duplicates happen only only occasionally get rid of duplicates if we don't want them.

Duplicates

If a tuple `t` occurs `n` times in R and `m` times in S, then `t` occurs

`n+m` times in R ∪ S

min(`n`, `m`) in R ∩ S

max(0, `n-m`) in R − S

\# of tuples in $\pi_{A1,\ldots,An}$(R) = # of tuples in R

duplicates are preserved in selection

If tuple t_1 occurs `n` times in R and tuple t_2 occurs `m` times in S, $t_1 t_2$ occurs `n*m` times in R · S. Joins are similar.

Extended Operations

Duplicate elimination: δ(R)

grouping operator: γ_L(R)

where L is a list of expressions of two types:

(1) an attribute of R [a **grouping attribute**]

(2)aggregate_function(an attribute of R) → new_attribute where aggregate_function = SUM, AVG, MIN, MAX, or COUNT. [in this case, the attribute of R is called an **aggregated attribute**]

More Extended Operations

projection: $\pi_L(R)$

where L is a list of expressions of two types:

(1) an attribute of R

(2) E → new_attribute where E is an expression computing a value from zero or more attributes of R.

sorting: $\tau_L(R)$

where L is a list of attributes of R on which R is sorted, lexicographically, from left to right.

Outerjoins

Natural outerjoin: $R \overset{\circ}{\bowtie} S$

Form natural join of R and S, add padded versions of tuples in R and S that did not participate in the formation of the natural join.

Left outerjoin: $R \overset{\circ}{\bowtie}_L S$

Just add the padded tuples from R.

Theta joins have similar outerjoins.

Right outerjoin: $R \overset{\circ}{\bowtie}_R S$

Just add the padded tuples from S.