Prof. Dr. Guido Moerkotte
Email: moerkotte@uni-mannheim.de

Query Optimization Exercise sheet 1

Exercise 1

You are given the following relations:

```
published(game, publisher)
designed(game, designer)
reviewed(game, reviewer)
```

Use relational algebra to construct a plan for each of the following queries. In addition, find a tree representation for each plan.

Exercise 1 a)

Find all designers of the game "Sokoban".

Solution

$\pi_{designer}(\sigma_{game='Sokoban'}(Designed))$

Exercise 1 b)

Find all publishers of all games designed by "Imabayashi".

Solution

$\pi_{publisher}(published \bowtie (\sigma_{designer='Imabayashi'}(Designed)))$

Exercise 1 c)

Find all reviewers who have reviewed at least one game that has not yet been published.

Solution

Using semi-join method:
$\pi_{reviewer}(reviewed - (reviewed \ltimes published))$
Using anti-join method:
$\pi_{reviewer}(reviewed \triangleright published)$

Exercise 2

How does `Select Distinct` simplify the produced results?

Solution

It enables us to work with `sets` instead of `bags`.

Some notes with regard to bags/multisets:
When we don't use `distinct` in SQL, we have to deal with duplicates. As a result, we cannot apply set theory to our use case - we need a notion of bags/multisets:

The definition of a multiset is simple: A set with duplicates, e.g., $\{1, 1, 1, 2, 2, 3\}_b$. The number of times an element occurs in the multiset is called its multiplicity. For a multiset A, the multiplicity of some element $x \in A$ is denoted by $m_A(x)$. The above multiset can also be specified using its multiplicities: $\{1^3, 2^2, 3^1\}_b$

For two multisets A and B, defined over a Domain D, we have the following operations:

- Inclusion: $A \subseteq B \iff \forall x \in D, m_A(x) \leq m_B(x)$
- Intersection: $A \cap B := \{x^{min(m_A(x), m_B(x))} \mid \forall x \in D\}_b$
- Union: $A \cup B := \{x^{max(m_A(x), m_B(x))} \mid \forall x \in D\}_b$
- Difference: $A \setminus B := \{x^{max(m_A(x) - m_B(x), 0)} \mid \forall x \in D\}_b$
- Sum: $\sum(A, B) := \{x^{m_A(x) + m_B(x)} \mid \forall x \in D\}_b$

The rules of commutativity, associativity and distributivity carry over from set theory.

For further details, see `https://en.wikipedia.org/wiki/Multiset`

However, have you tried `union all` in SQL (union without duplicate elimination)? The `union all` operator corresponds to the above sum operator. For this reason, in the context of query optimization, we will refer to the sum operation as union and denote it by $\cup_b$. In the context of query optimization, we leave the union operator, as defined above, undefined.

This redefinition of the union operator has a serious drawback: The properties of the operators change. Unlike for sets, for bags we have that $\cap$ is not distributive over $\cup_b$:

$$(A \cup_b B) \cap C \neq (A \cap C) \cup_b (B \cap C)$$

To see this, consider the example $A = \{1\}, B = \{1\}, C = \{1\}$.

For further details, see the script/book building query compilers, Chapter 7.1.2: Duplicate Data: Bags

---

Exercise 3

Consider the relational algebra expression

$$R \bowtie S \bowtie T.$$

Since $\bowtie$ is commutative and associative, there are 12 equivalent ways to compute the result of the above expression:

$$(R \bowtie (S \bowtie T))$$

$$((R \bowtie S) \bowtie T)$$
$$(R \bowtie (T \bowtie S))$$
$$((R \bowtie T) \bowtie S)$$
$$(S \bowtie (R \bowtie T))$$
$$((S \bowtie R) \bowtie T)$$
$$(S \bowtie (T \bowtie R))$$
$$((S \bowtie T) \bowtie R)$$
$$(T \bowtie (R \bowtie S))$$
$$((T \bowtie R) \bowtie S)$$
$$(T \bowtie (S \bowtie R))$$
$$((T \bowtie S) \bowtie R)$$

We will soon see that, despite all expressions have the same result, their cost of computing differs greatly.

Implement a program that prints all possible ways to join n relations.

Solution

Abstract algorithm:

- Let n be the number of relations.
- Compute all $n!$ permutations of the relations in the expression.
- For each permutation find all possible ways to associate the join operators (by parenthesizing them). There are $C_{n-1} = \frac{1}{n}\binom{2(n-1)}{n-1}$ possible associations, where C_{n-1} denotes $n-1$th Catalan number, see `https://en.wikipedia.org/wiki/Catalan_number`.
- Print each join order. In total there are $n!C_{n-1}$ join orders.

See code for details on how to compute all permutations and all associations.