Weekly exercises IN3020 and IN4020
Week 6

March 17, 2022

1 Review

Questions adapted from [1].

1. What are the 3 main steps that a query compiler must perform (as outlined in the lectures)?

   Solution:

   (a) Parsing

   (b) Compiling

   (c) Optimization

2. What is the role of the parser in the query compiler?

   Solution: Transform an input string to a tree using some grammar

3. What is a logical query plan?

   Solution: Basically the Relational Algebra expression.

4. What is the (ideal) goal of query optimization?

   Solution: Idealistically: Find another query plan that:

   - Gives the same answer
   - Can be evaluated as quick as possible
   - Is quick to find

5. What is does our cost function include?

   Solution: The estimated sums of sizes of the intermediate relations.

2 Parsing

Questions adapted from [1].
Consider this grammar for a subset of SQL:

```
<Query> := SELECT <SelList> FROM <FromList> WHERE <Condition>

<SelList> := <Attribute>, <SelList>
<SelList> := <Attribute>

<FromList> := <Relation>, <FromList>
<FromList> := <Relation>


<Condition> := <Condition> AND <Condition>
<Condition> := <Attribute> IN ( <Query> )
<Condition> := <Attribute> = <Attribute>
<Condition> := <Attribute> LIKE <Pattern>
```

1. Extend (or change) the rules from `<Query>` to include the following features:
   (a) Produce a set using the `DISTINCT`-keyword
   (b) Sort the output using the `ORDER BY`
   (c) A query with no `WHERE`-clause

   Solution:

   (a) Many possibilities. One is to change the rules for `|<SelList>|`to

   ```
   <SelList> ::= <OptDistinct>, <SelList1>
   <OptDistinct> ::= DISTINCT
   <OptDistinct> ::= emptyString
   <SelList1> ::= <Attribute> , <SelList1>
   <SelList1> ::= <Attribute>
   ```

   (b) Can change the rules for `|<Query>|`to:

   ```
   <Query> := SELECT <SelList> FROM <FromList>
              WHERE <Condition> <OptOrdering>
   <OptOrdering> := emptyString
   <OptOrdering> := ORDER BY <OrderList>
   <OrderList> := <Attribute>, <OrderList>
   <OrderList> := <Attribute>
   ```

(c) Can change rules for `|<Query>|` to:

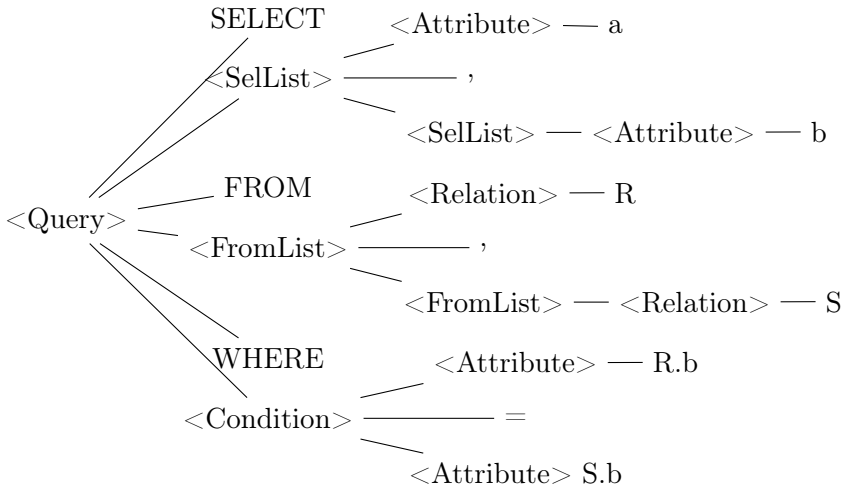
```
<Query> := SELECT <SelList> FROM <FromList> <OptCondition>
<OptCondition> := emptystring
<OptCondition> := WHERE <Condition>
```

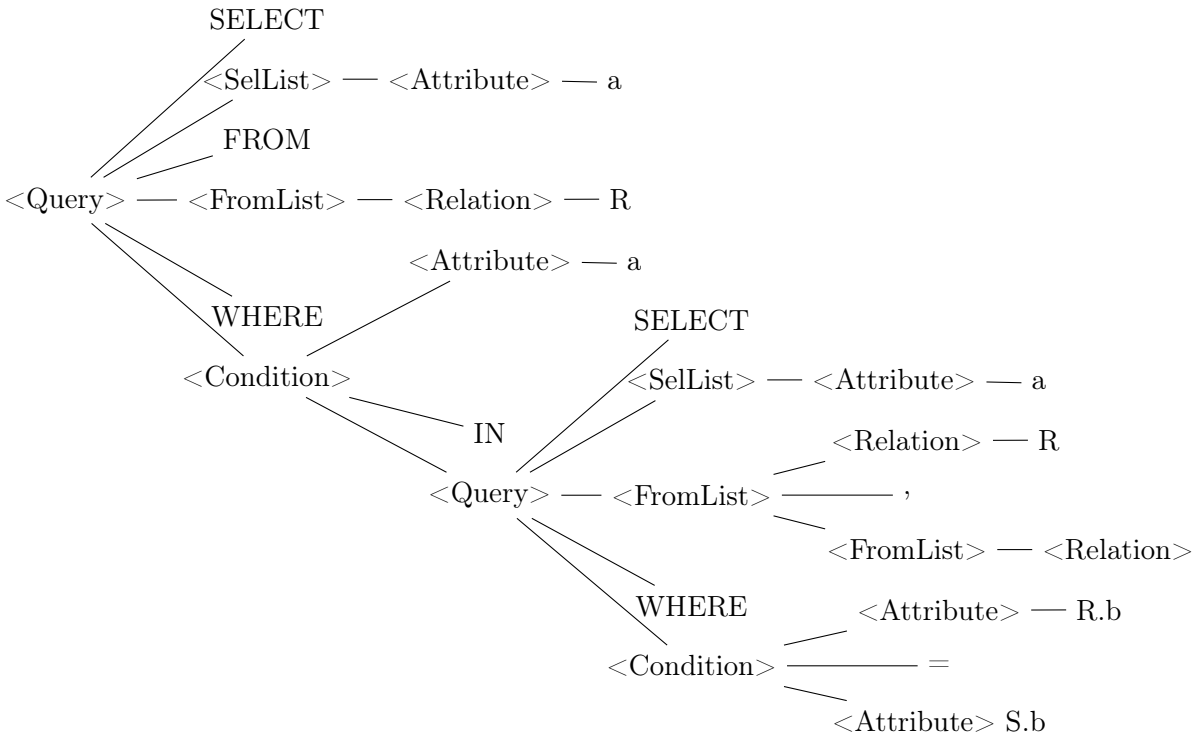
2. Use the subset of SQL grammar above and give parse trees for the following queries about relations `R(a,b)` and `S(b,c)`:

   (a) `SELECT a,c FROM R,S WHERE R.b = S.b;`

   (b)
   ```
   SELECT a FROM r WHERE a IN
     (SELECT a FROM R, S WHERE R.b = S.b);
   ```

Solution:





3 Compilation

Consider the following two schemas:

```
R[A,B,C]
S[A,D,F]
```

Rewrite the following SQL queries into Relational Algebra expressions:

3.1

```
SELECT A
FROM R
```

Solution: $\pi_A(R)$

3.2

```
SELECT *
FROM R
NATURAL JOIN S
```

Solution: $R \bowtie S$

3.3

```
SELECT *
FROM R
FULL OUTER JOIN S
ON R.B = S.D
```

Solution: $(R \rtimes (\rho_{D->B} S))$

3.4

```
SELECT A, SUM(B)
FROM R
GROUP BY A
```

Solution: $\pi_{A,sum}(\gamma_{A,SUM(B)\to sum}(R))$

3.5

```
SELECT R.C
FROM R
WHERE R.A IN
  (SELECT S.A FROM S WHERE S.D > 0)
```

Solution: $\pi_C(R \ltimes (\sigma_{D>0}(S)))$

3.6

```
SELECT R.C
FROM R
WHERE NOT EXISTS
  (SELECT S.A FROM S WHERE R.A = S.A)
```

Solution: $\pi_C(R \triangleright S)$

4 Optimization

Questions adapted from [1].

4.1

When it is possible to push a selection to both arguments of a binary operator, we need to decide whether or not to do so. How would the existence of indexes on one of the arguments affect our choice? Consider for instance an expression $\sigma_C(R \cap S)$, where there is an index on S.

Solution: If the index on S helps us find tuples that satisfy the condition C, then surely we should push the selection to S. Let the result of the selection on S be a relation S', which is a subset of S. Assuming there is no index on R that helps with condition C, then if we push the selection to R as well, we still have to intersect that result with S'. However, if we do not push the selection to R, then all that remains is intersecting R with S'. Performing either the selection or the intersection on R requires that we read all of R, so we are better off performing just the intersection rather than the selection plus a (smaller) intersection.

4.2

Give examples to show that:

a) Projection (π^b) cannot be pushed below set union

b) Projection (π^b) cannot be pushed below set or bag difference

c) Duplicate elimination (δ) cannot be pushed below projection (π^b).

Solution: In the solutions, let $R[A, B] = \{(1, 2)\}$ and $S[A, B] = \{(1, 3)\}$.

a) Then $\pi^b_A(R \cup S) = \pi^b_A(\{(1, 2), (1, 3)\} = \{(1), (1)\}$.
But $\pi^b_A(R) \cup \pi^b_B(S) = (1)$, since the set union removes duplicates from the union.

b) $\pi^b_A(R \setminus S) = \pi^b_A(\{(1, 2)\} \setminus \{(1, 3)\}) = \pi^b_A(\{(1, 2)\}) = \{(1)\}$
But $\pi^b_A(R) \setminus \pi^b_A(S) = \{1\} \setminus \{1\} = \emptyset$.
With bag difference, we get the same result.

c) $\delta(\pi^b_A(R \cup S)) = \delta(\{(1), (1)\}) = \{(1)\}$
But $\pi^b_A\delta(R \cup S) = \pi^b_A\delta(\{(1, 2), (1, 3)\} = \{(1), (1)\}$

4.3

Prove that we can always push a projection below both branches of a bag union.

Solution: Idea: Show that if an element is in $\pi_A(R \cup^b S)$ it must also be in $\pi_A(R) \cup^b \pi_A(S)$.

Proof. Let $x' = (x_1, \ldots, x_N)$ be an arbitrary element in $\pi_A(R \cup^b S)$. And $(A_1, \ldots, A_N)$ be the attributes in A. Since x' is in the projection, there must be an element $x = (x_1, \ldots, x_n) \in (R \cup^b S)$. By the definition of $\cup^b$, we know that $x \in R \vee x \in S$. Since $x \in R \vee x \in S$ we know that $x' \in \pi_A(R) \vee x' \in \pi_A(S)$. Therefore $x' \in (\pi_A(R) \cup^b \pi_A(S))$. □

To complete the proof of equivalency, we need to also show that if an element is in $\pi_A(R) \cup^b \pi_A(S)$ then it must also be in $\pi_A(R \cup^b S)$. A similar argument can be used to show this.

4.4

Consider the statistics for the four relations W, X, Y and Z in Table 1. Estimate the sizes of relations (number of tuples) that are the results of the

Table 1: Statistics for W, X, Y and Z

$W(a,b)$	$X(b,c)$	$Y(c,d)$	$Z(d,e)$
$Tups(W) = 100$	$Tups(X) = 200$	$Tups(Y) = 300$	$Tups(Z) = 400$
$Vals(W.a) = 20$	$Vals(X.b) = 50$	$Vals(Y.c) = 50$	$Vals(Z.d) = 40$
$Vals(W.b) = 60$	$Vals(X.c) = 100$	$Vals(Y.d) = 50$	$Vals(Z.e) = 100$

following expressions:

a) $W \bowtie X \bowtie Y \bowtie Z$

b) $\sigma_{a=10}(W)$

c) $\sigma_{c=20}(Y)$

d) $\sigma_{c=20}(Y) \bowtie Z$

e) $W \times Y$

f) $\sigma_{c>10}(Z)$

g) $\sigma_{a=1 \ AND \ b=2}(W)$

Solution:

a) Start by taking the product of the sizes of the relations: $100 * 200 * 300 * 400 = 2,400,000,000$. Since attribute b appears in two relations, divide by the larger of $V(W,b)$ and $V(X,b)$, which is 60. Similarly, for c divided by 100, and for d divide by 50. The result is 8000 tuples.

b) The estimate is $Tup(W)/Val(W,a) = 100/20 = 5$.

c) Take $Tups(W)/Vals(W,a) = 300/50 = 6$ tuples.

d) $300/50 = 6$ tuples.

e) You could reuse the answer from c) you get:

- $Tups(\sigma_{c=20}(T)) = 6$
- $Vals(\sigma_{c=20}(T)) \leq 6$

- $Vals(\sigma_{c=20}(T), c) \leq 6$
- So: $(6 * 400)/max(40, 6) = 60$

- If you take the join first and the selection after, you get: $(300 * 400)/(50 * 50) = 48$

f) $Tups(W) * Tups(Y) = 100 * 300 = 30,000$ tuples.

g) We assume the query to retrieve a fraction less than half of the tuples. $Tups(Z)/3 \approx 133$ tuples.

h) $Tups(W)/Vals(W, a) * Vals(W, b) = 100/(20 * 60) = 1/12$ tuples.

4.5

How would you estimate the size of a semijoin?

Solution: A (naive) idea is to use the definition of semijoin. $R \ltimes S = \pi_{RAtts}(R \bowtie S)$. And calculate as if we were doing a natural join. However, we can use what we know about semijoin. That the size of the semijoin can not be larger than $Tups(R)$, and if S has fewer values than R, the size wil inevitably be smaller than $Tups(R)$. Using this information, we can estimate the size of $R \ltimes S$ as:

$$TupSize(R) * Tups(R) * Min\left(1, \frac{Tups(S)}{Tups(R)}\right)$$

4.6

Estimate the size of the join $R(a, b) \bowtie S(b, c)$ using histograms for $R.b$ and $S.b$. Assume $Vals(R.b) = Vals(S.b) = 20$, and the histograms for both attributes give the frequency of the four most common values, as in Table 2.

	0	1	2	3	4	others
$R.b$	5	6	4	6		32
$S.b$	10	8	5		7	32

Table 2: Frequency of most common values

Solution: 5 known values, 15 unknown values. That leaves $32/15 \approx 2$ tuples per each of the 15 unknown values.
For each of the known values, we can calculate the exact number of tuples: $5 * 10 + 6 * 8 + 4 * 5 + 6 * 0 + 0 * 7 = 118$ tuples.
Since the known values are the same for both relations, we use the estimated

2 tuples per unknown value and assume them to have the same unknown values: $2 * 2 * 15 = 90$ tuples.
In total we get an estimate of $118 + 90 = 208$ tuples.

References

[1] Hector Garcia-Molina. *Database systems: the complete book*. Pearson Education India, 2008.