Distributed Databases 2018 HW 2

Student ID: **2018280070**   Name: **Peter Garamvoelgyi**

Problem 8.3

First, let us transform the query into its conjunctive normal form:

```
SELECT ENAME, PNAME
FROM EMP, ASG, PROJ
WHERE
      (DUR > 12 ∨ RESP = "Analyst")
    ∧ EMP.ENO = ASG.ENO
    ∧ (TITLE = "Elect. Eng." ∨ ASG.PNO < "P3")
    ∧ (DUR > 12 ∨ RESP NOT= "Analyst")
    ∧ ASG.PNO = PROJ.PNO
```

Then, let us normalize and simplify it. We will use the following three rules

1. AND clauses can be reordered (commutativity)
2. `(P1 ∨ P2) ∧ (P1 ∨ P3)` ⟺ `P1 ∨ (P2 ∧ P3)` (equivalence rule)
3. `P1 ∨ false` ⟹ `P1` (idempotency rule)

Using these, we can employ the following simplification:

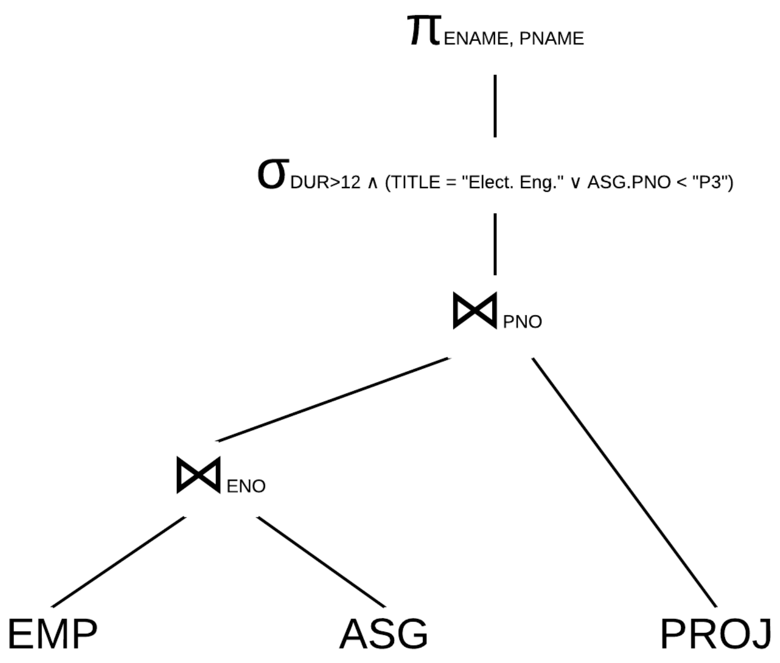
```
(DUR > 12 ∨ RESP = "Analyst") ∧ (DUR > 12 ∨ RESP NOT= "Analyst")
DUR > 12 ∨ (RESP = "Analyst" ∧ RESP NOT= "Analyst")
DUR > 12 ∨ false
DUR > 12
```

Our resulting query is

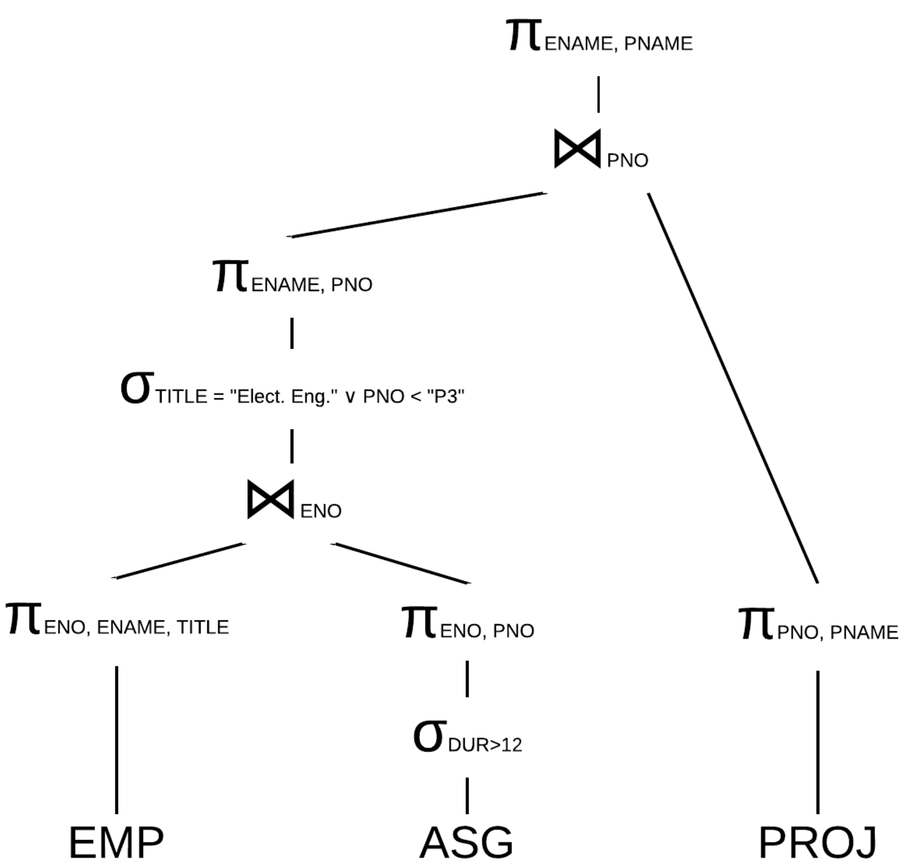
```
SELECT ENAME, PNAME
FROM EMP, ASG, PROJ
WHERE
      EMP.ENO = ASG.ENO
    ∧ ASG.PNO = PROJ.PNO
    ∧ (DUR > 12)
    ∧ (TITLE = "Elect. Eng." ∨ ASG.PNO < "P3")
```

As we have an OR clause, we cannot use a query graph here. What we can do, however, is to sketch a query tree and iteratively optimize it.

The original query tree is:

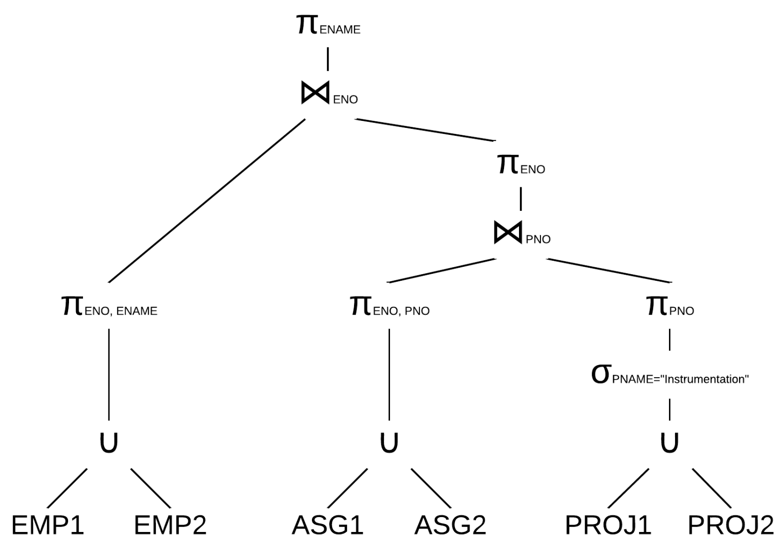


By moving unary operations towards the leaves and using projections to keep the relevant attributes only, we can get a query graph that can be evaluated much more efficiently:

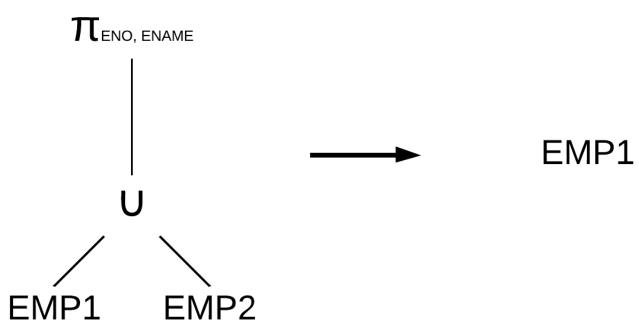


Problem 8.8

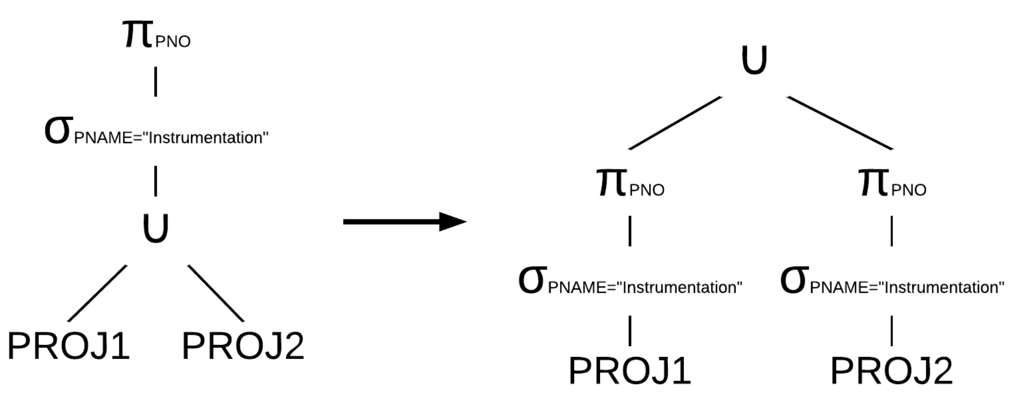
First, let us take a look at the query graph:



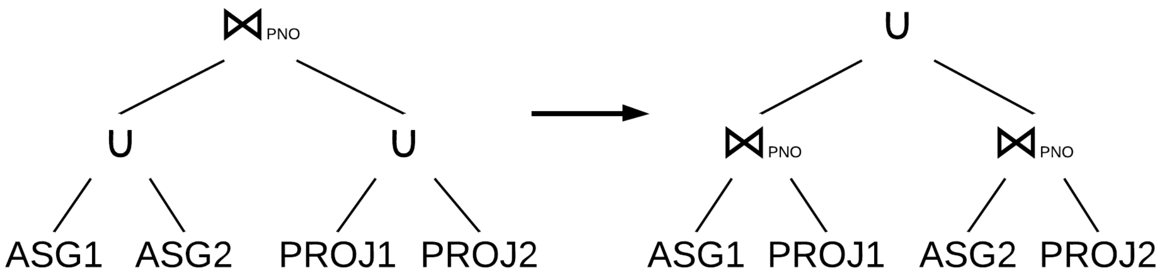
There are multiple places where we could optimize by moving operations into the fragments. First, let us notice that we do not need to use `EMP2`, as we only need `ENO` and `ENAME`, not `TITLE`.



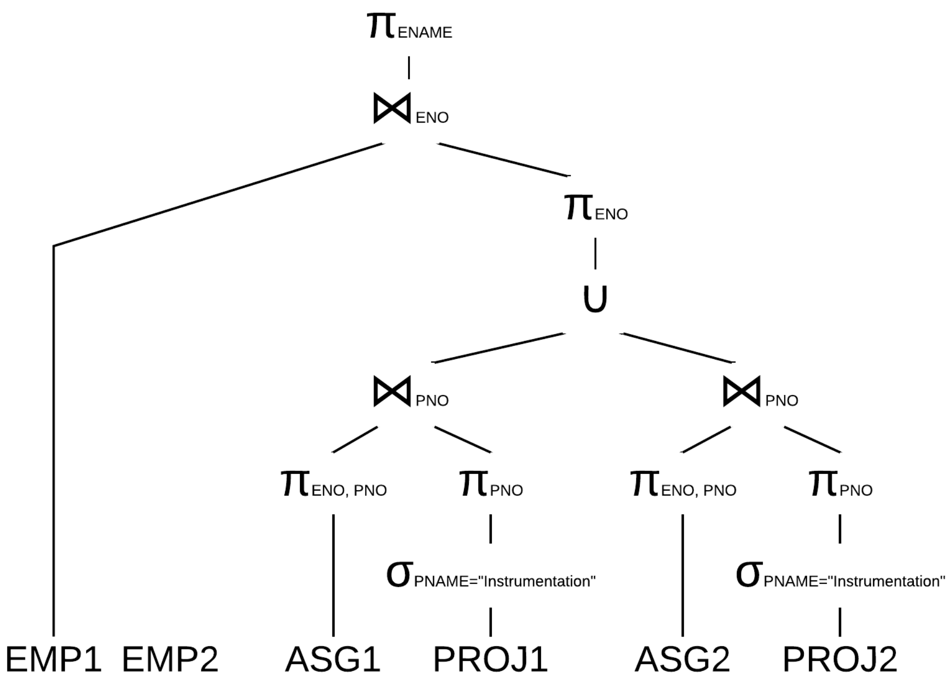
Second, selection and projection can be moved onto the fragments.



Moreover, we know that for `PROJ1`, the corresponding `PNO` values are in `ASG1`, and the same for `PROJ2` and `ASG2`. (This is because `ASG`'s horizontal fragmentation is derived based on `PROJ`.) This means that we can distribute joins over the union:



Putting these insights together, we get the new, localized query graph:



Now we can decompose the original query into multiple queries on the fragments:

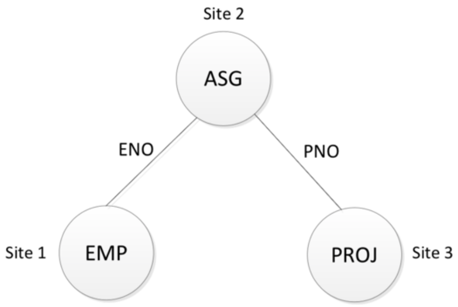
```
SELECT ENO, PNO                  SELECT ENO, PNO
INTO   A1                        INTO   A2
FROM   ASG1                      FROM   ASG2
```

```
SELECT PNO                       SELECT PNO
INTO   P1                        INTO   P2
FROM   PROJ1                     FROM   PROJ2
WHERE  PNAME="Instrumentation"   WHERE  PNAME="Instrumentation"
```

```
SELECT tmp.ENO INTO ASGPROJ
FROM (
    SELECT ENO, PNO FROM A1, P1 WHERE A1.PNO = P1.PNO
    UNION
    SELECT ENO, PNO FROM A2, P2 WHERE A2.PNO = P2.PNO
  ) as tmp
```

```
SELECT ENAME
FROM   EMP1, ASGPROJ
WHERE  EMP1.ENO = ASGPROJ.ENO
```

Problem 9.2




```
size(EMP)  = 100    size(EMP ∞ ASG)  = 200
size(ASG)  = 200    size(ASG ∞ PROJ) = 200
size(PROJ) = 300
```

We need to find the optimal join program w.r.t. transmission time. Let us assume that transmission time is proportional to the size of the data being transmitted (i.e. we will ignore the constant msg initialization time here). This way the problem becomes: How do we perform the two joins with the least amount of data transmission.

The order of joins is either `(EMP ⋈ ASG) ⋈ PROJ` or `EMP ⋈ (ASG ⋈ PROJ)`. Let us enumerate the possibilities:

	scenario	cost	
1	site1: X = (EMP ⋈ ASG) site2: X ⋈ PROJ	200 200 + 300	= 700
2	site2: X = (EMP ⋈ ASG) site2: X ⋈ PROJ	100 0 + 300	= 400
3	site2: X = (EMP ⋈ ASG) site3: X ⋈ PROJ	100 200 + 0	= 300
4	site3: Y = (ASG ⋈ PROJ) site2: EMP ⋈ Y	200 100 + 200	= 500
5	site2: Y = (ASG ⋈ PROJ) site2: EMP ⋈ Y	300 100 + 0	= 400
6	site2: Y = (ASG ⋈ PROJ) site1: EMP ⋈ Y	300 0 + 200	= 500

From the analysis above, we can see that scenario 3 has the lowest cost:

1. Copy `EMP` to `site2`
2. Compute `X=(EMP ⋈ ASG)` on `site2`
3. Copy `X` to `site3`
4. Compute `X ⋈ PROJ` on `site3`

The result will be on `site3`.

(Note: There are some possibilities not included above. However, as we need to join 3 tables across 3 sites, it's easy to see that we need at least 2 data transmissions. Given the costs above, the theoretical minimum cost of 2 data transmissions is 300.)