Outline of solutions to selected exercises in Mandatory assignment 1

Exercise 1 Some SQL operations

a)

```
SELECT S.Course_number, S.Semester, S.Year, COUNT(DISTINCT
Student_number)
FROM Section S, Grade_Report G
WHERE Instructor='King' AND S.Section_Identifier = G.Section_Identifier
GROUP BY S.Course_number, S.Semester, S.Year
```

b)

```
SELECT S.Name, S.Major
FROM Student S
WHERE NOT EXISTS (
    SELECT *
    FROM Grade_Report G
    WHERE G.StudentNumber = S.StudentNumber AND NOT
(G.Grade='A'))
```

c)

```
INSERT INTO Student VALUES ('Johnson', 25, 1, 'MATH')
```

d)

```
UPDATE Student SET CLASS = 2 WHERE Name = 'Smith'
```

e)

```
DELETE FROM Student WHERE Name = 'Smith' AND StudentNumber=17
```

Exercise 2 Introduction to DBMS Architecture

2.1 QTE schematics

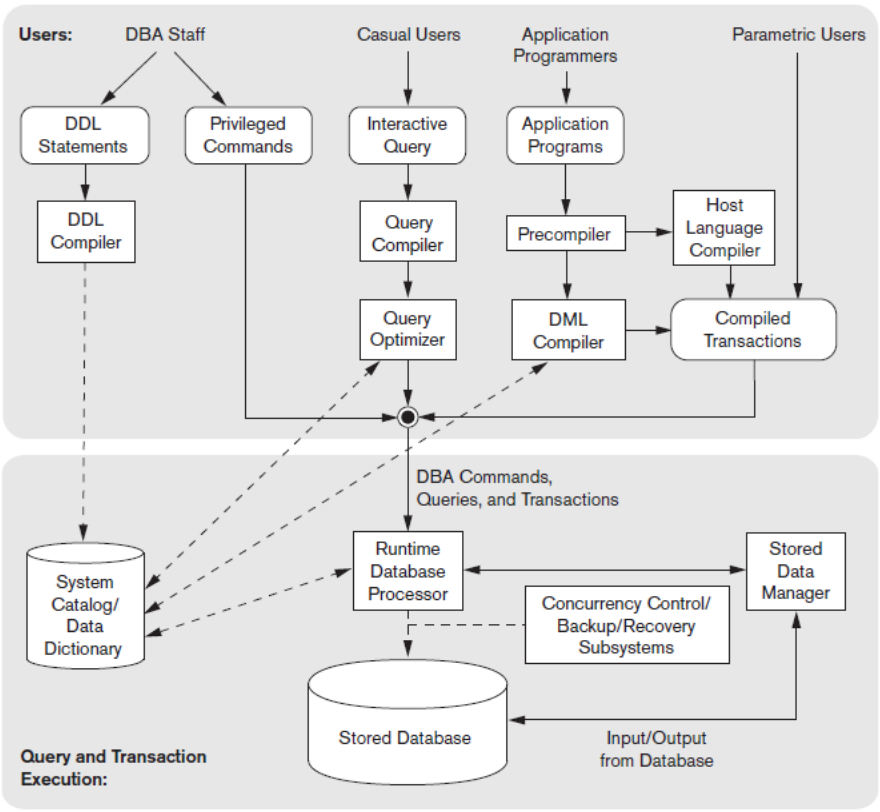


Figure 2.3
Component modules of a DBMS and their interactions.

Elmasri and Navathe. Fundamentals of Database Systems, Seventh edition.

[The task here was to describe the DBMS components in the lower box, following the books and/or the lectures while using your own words]

2.2 Central components and Higher-level components

[Please consult slides 7-11 in the link:
https://www.uio.no/studier/emner/matnat/ifi/IN3020/v21/lectures/021-2---dbms-architecture-dependencies.pdf]

Exercise 4 Relational Algebra

4.1

a) Rewrite the query:

```
SELECT P.C
FROM Purchase P
WHERE Exists
      (SELECT S.A
       FROM Supply S
       WHERE P.A=S.A AND S.D>0)
```

RA expression for the nested subquery:

$$\pi_{\mathrm{P}.A,\mathrm{P}.B,\mathrm{P}.C,\mathrm{S}.A,\mathrm{S}.D}\,[\sigma_{\mathrm{P}.A=\mathrm{S}.A\,\wedge\,\mathrm{S}.D>0}\{\rho_{\mathrm{P}}\ \text{Purchase}\ \times \rho_S\ \text{Supply}\ \}]$$

[*For Selection in the nested part, you keep all the attributes/columns, so you inadvertently do not remove any columns that might be needed for the final Selections as specified in the outer query*]

The first **FROM** part is written as:

$$\rho_{\mathrm{P}}\ \text{Purchase}$$

Joining these two, making some optimization on projection, i.e. removing redundant columns, and applying the final selection corresponding to the first line:

$$\pi_{\mathrm{P}.C}(\rho_{\mathrm{P}}\ \text{Purchase}\ \bowtie \pi_{\mathrm{P}.A,\mathrm{P}.B,\mathrm{P}.C}\,[\sigma_{\mathrm{P}.A=\mathrm{S}.A\,\wedge\,\mathrm{S}.D>0}\{\rho_{\mathrm{P}}\ \text{Purchase}\ \times \rho_S\ \text{Supply}\ \}])$$

b) Optimized query:

$$\pi_{\mathrm{P}.C}(\rho_{\mathrm{P}}\ \text{Purchase}\ \bowtie \pi_{\mathrm{P}.A,\mathrm{P}.B,\mathrm{P}.C}\,[\rho_{\mathrm{P}}\ \text{Purchase}\ \bowtie_{\mathrm{P}.A=\mathrm{S}.A}\{\sigma_{\mathrm{S}.D>0}\ \rho_{\mathrm{S}}(\text{Supply}\ \}])$$

By pushing selection and by converting Selection and Cartesian-join into a Theta-join

c) Selection has been applied on individual relation (S) rather than on a Join. Additionally, Cartesian-join is replaced by a Theta join.

4.2

a) Sequence of steps for the following RA transformation:

$$\pi_M(\sigma_{<c>\wedge<d>}(\mathrm{P}\bowtie\mathrm{Q}\bowtie\mathrm{R})) \rightarrow \pi_M[\{\pi_A(\sigma_{<c>}\ \mathrm{P}\ )\bowtie\pi_{A,H}(\sigma_{<d>}\ \mathrm{Q}\ )\}\bowtie\mathrm{R}]$$

- Associativity of join: $\pi_M[\sigma_{<c>\wedge<d>}(\ \mathrm{P}\quad\mathrm{Q}\quad\mathrm{R})]$
- Pushing selection over join: $\pi_M[\{\sigma_{<c>\wedge<d>}(\mathrm{P}\bowtie\mathrm{Q})\}\bowtie\mathrm{R}]$ $\quad$ <c>,<d> *not on* R
- Pushing selection to respective relation: $\pi_M[\{\sigma_{<c>}(\mathrm{P})\bowtie\sigma_{<d>}(\mathrm{Q})\}\bowtie\mathrm{R}]$

Since $<c>$ applies on P only and $<d>$ on Q only.

- Distribution of projection over join:

$$\pi_M[\pi_H\{\sigma_{<c>}(\mathrm{P}) \bowtie \sigma_{<d>}(\mathrm{Q})\} \bowtie \pi_{F,H,K,M}(\mathrm{R})]$$

- Since $\pi_{F,H,K,M}\ \mathrm{R} = \mathrm{R}$:

$$\pi_M[\pi_H\{\sigma_{<c>}(\mathrm{P}) \bowtie \sigma_{<d>}(\mathrm{Q})\} \bowtie \mathrm{R}]$$

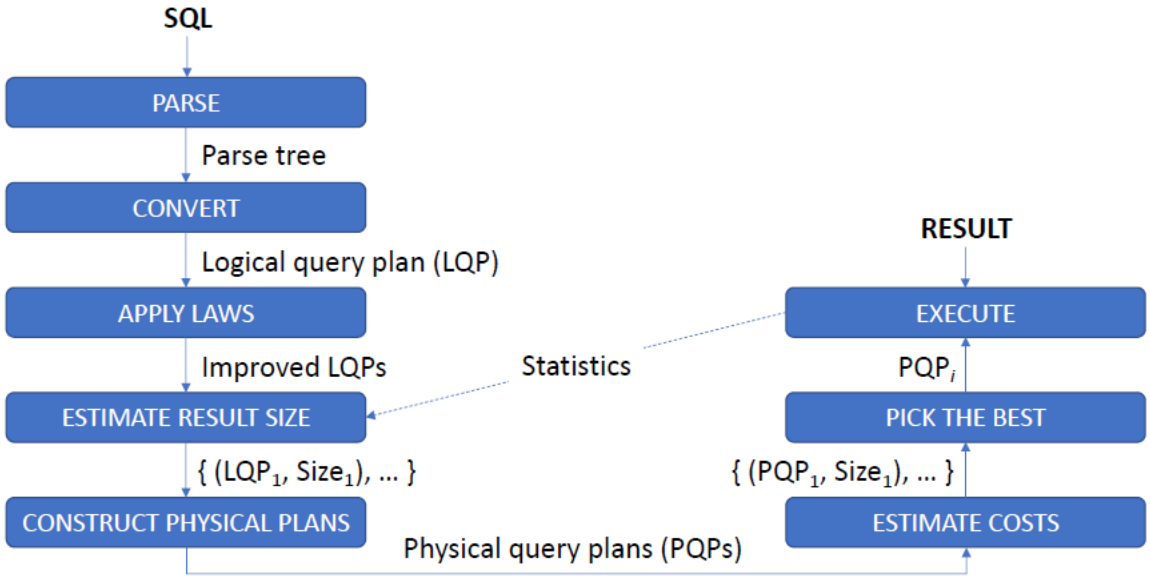
- Distribution of projection over join:

$$\pi_M[\{\pi_A(\sigma_{<c>}\ \mathrm{P}\ ) \bowtie \pi_{A,H}(\sigma_{<d>}\ \mathrm{Q}\ )\} \bowtie \mathrm{R}]$$

b) The final expression appears better because Selection has been split and pushed down over the join, and the large join $\mathrm{P} \bowtie \mathrm{Q} \bowtie \mathrm{R}$ at the very beginning has especially been avoided.

Exercise 5 – Query Compilation

5.1 Query processing



[The task was about describing this well-known process]

5.2 SQL, RA tree, Size computations

a) SQL query

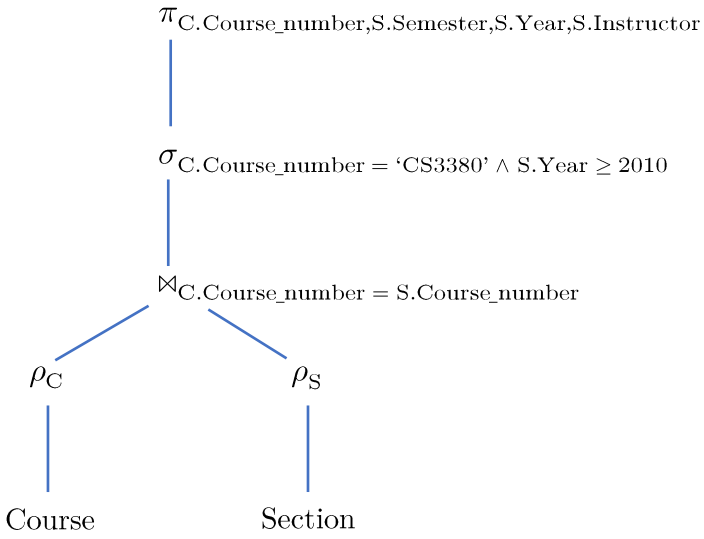
[Since this query involves joining two relations, it can be written in several possible ways, e.g., joining the relations first and filtering tuples/attributes later, or the other way around. About the join type, Cartesian product will be the simpler but costlier than other joins. Here, we start with the theta-join in the non-optimized version]

```
Select C.Course_name, S.Semester, S.Year, S.Instructor
From Course As C Join Section As S
On C.Course_number = S.Course_number
Where C.Course_number = 'CS3380' And S.Year>=2010
```

b) RA expression

$$\pi_{\text{C.Course\_name,S.Semester,S.Year,S.Instructor}}(\sigma_{\text{C.Course\_number= 'CS3380'} \wedge \text{S.Year} \geq 2010} (\rho_{\text{C}} \text{ Course } \bowtie_{\text{C.Course\_number} = \text{S.Course\_number}} \rho_{\text{S}} \text{ Section }))$$

c) RA tree



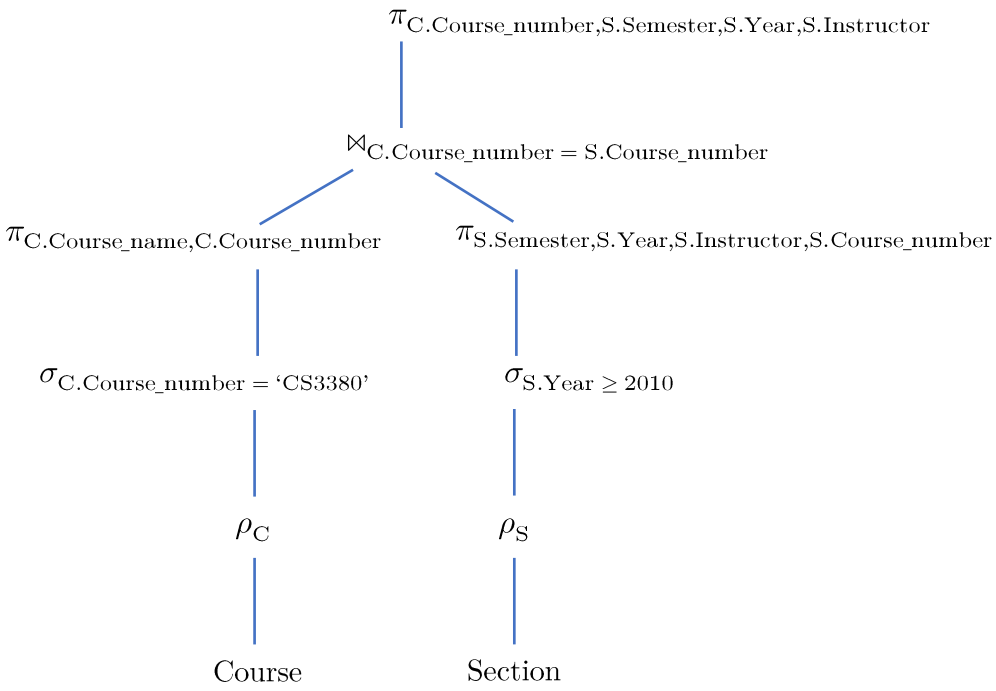
d) Improved RA expression and tree

[Apply selection and projection on the respective relations first, perform join afterward]

$$\pi_{\text{C.Course\_name,S.Semester,S.Year,S.Instructor}}[$$
$$\{\pi_{\text{C.C.Course\_name,C.Course\_number}}(\sigma_{\text{C.Course\_number= 'CS3380'}}\ \rho_{\text{C}}\ \text{Course}\ )\}$$
$$\bowtie_{\text{C.Course\_number = S.Course\_number}}$$
$$\{\pi_{\text{S.Semester,S.Year,S.Instructor,S.Course\_number}}(\sigma_{\text{S.Year} \geq 2010}\ \rho_{\text{S}}\ \text{Section}\ )\}\ ]$$

[Projections in the 2nd and 4th lines improve the RA expression further. However, even if we did not perform these projections, this expression is still likely to be better than the expression in b)]

RA tree of the improved expression:



[It might be convenient to replace the attribute names in the expression by suitable symbols with prior declaration, e.g., Course_name = Cn, Course_number = Cm, Year = Yr, etc.]

e) Size calculations

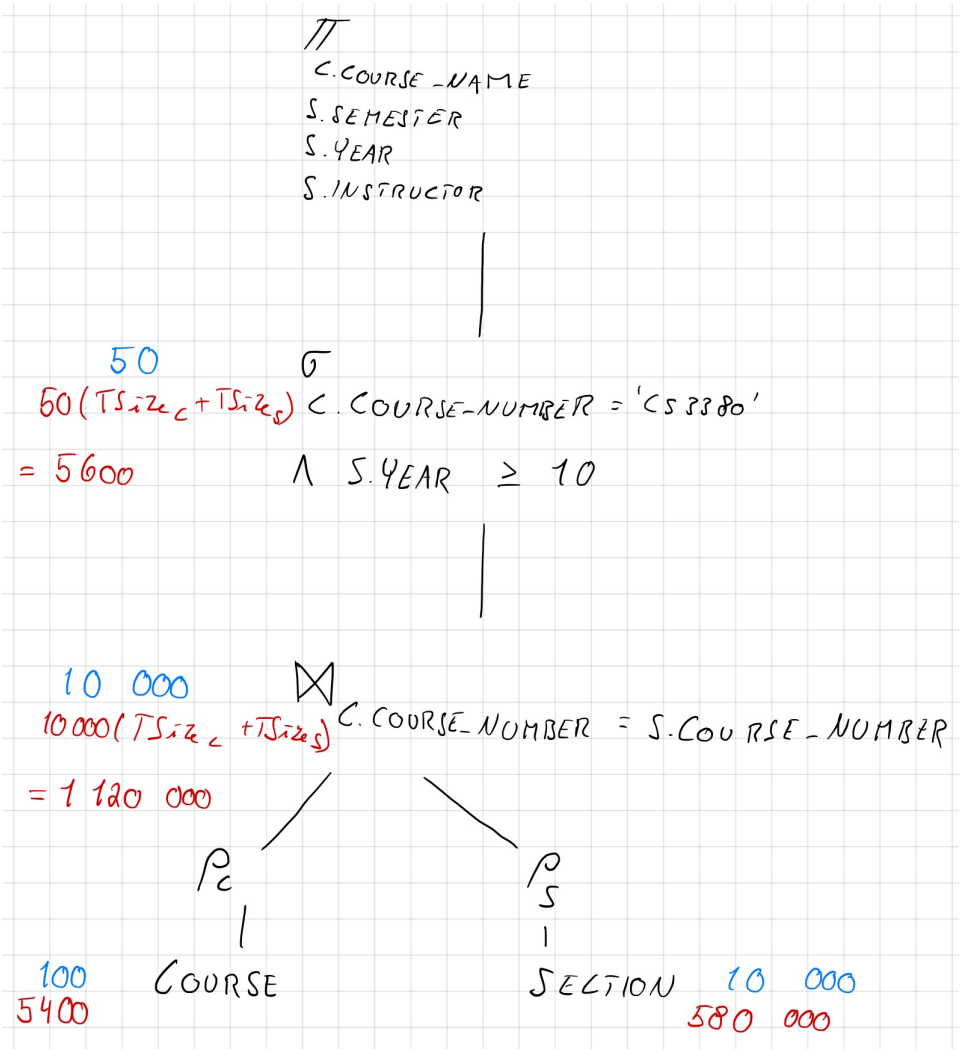
[This part allowed the students to exercise freedom in choosing the tuple numbers, tuple sizes, selectivity factors etc. and demonstrate their understanding of the logical flow in cost/size calculations. We did not make a detailed official solution in this part. Martin Strøm Olsen from IN 4020 has made a thorough calculations here, capturing the key considerations in such computation, which is presented as inspiration for others. Thanks, Martin.]

Course (C)	**Section (S)**
Tup(C) = 100	Tup(S) = 10000
Val_C(Course_Number) = 100	Val_S(Course_Number) = 100 Val_S(Year) = 100
TSize_C = (20, 20, 4, 10) bytes, with the order of attributes as given in the relation Course.	TSize_S = (4, 20, 10, 4, 20) bytes, with the order of attributes as given in the relation Section.

It is assumed that Course number is a PK of Course and FK in Section. The number of tuples (blue) and the estimated tuple size (red) for the original and the modified tree is shown below, together with the detailed computations.
Some notations defined:

$$c_1 := \text{C. Course\_number} = \text{'CS3380'}$$
$$c_2 := \text{S. Year} \geq 2010$$
$$c_3 := \text{C. Course\_number} = \text{S. Course\_number}$$

Original tree



Estimates of number of tuples

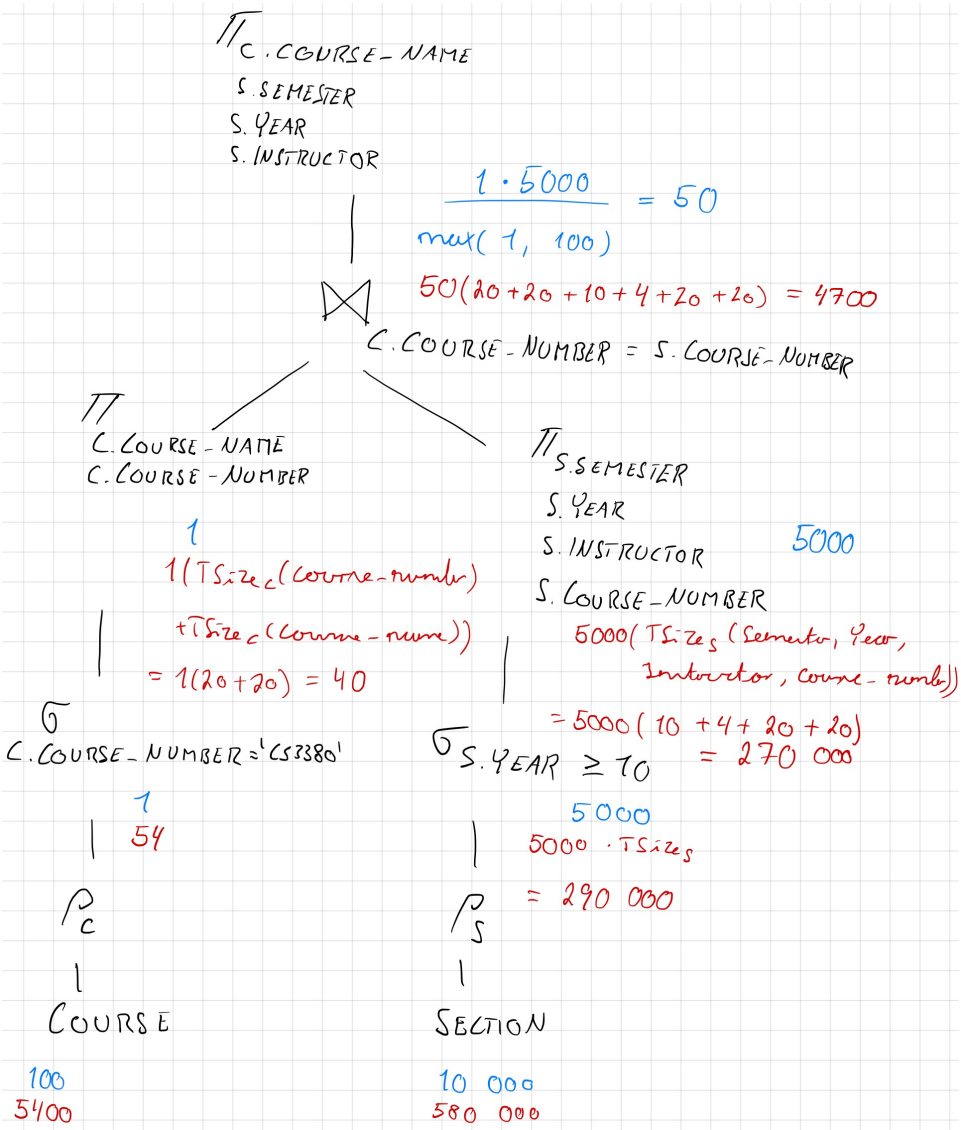
- $C \bowtie_{\text{C.Course\_number = S.Course\_number}} S$
  here Course_number is a PK in C and a FK in S which gives Tup = Tup(S) = 10000
- $\sigma_{c_1 \wedge c_2}$
  c_1 selectivity factor = $\frac{1}{Val_C(\text{Course\_number})} = \frac{1}{100}$
  c_2 selectivity factor: assume Year $\sim$ Uniform(5, 14), so 14 - 5 + 1 = 10 different values and Year $\geq$ 10 gives (10, 11, 12, 13, 14) 5 possible values. Selectivity factor = $\frac{5}{10}$
  Tup = Tup($\sigma_{c_1 \wedge c_2}(C \bowtie_{c_3} S)$) = Tup($C \bowtie_{c_3} S$)$\frac{1}{100}\frac{5}{10}$ = 10000$\cdot$ $\frac{1}{100} \cdot \frac{5}{10} = 50$

Estimating the size of the temporary relations

- $C \bowtie_{\text{C.Course\_number} = \text{S.Course\_number}} S$
  $\Sigma_1 = 10000(\text{TSize}_C + \text{TSize}_S) = 10000 \cdot (54 + 58) = 1120000$
- $\sigma_{c_1 \wedge c_2}$
  $\Sigma_2 = 50(\text{TSize}_C + \text{TSize}_S) = 50(54 + 58) = 5600$

then the estimated size (excluding root node and leaf nodes)
$\Sigma = \Sigma_1 + \Sigma_2 = 1120000 + 5600 = 1125600$

Modified tree



Estimates of number of tuples

- σ_{c_1}
  $\text{Tup}(\sigma_{c_1}) = \frac{\text{Tup}_C}{\text{Val}_C(\text{Course\_number})} = \frac{100}{100} = 1$

- σ_{c_2}
  $\text{Tup}(\sigma_{c_2}) = \text{Tup}_S \frac{5}{10} = 10000 \cdot \frac{5}{10} = 5000$
- $\pi_{\text{C.Course\_name, C.Course\_number}}$
  $\text{Tup} = \text{Tup}(\sigma_{c_1}) = 1$
- $\pi_{\text{S.Semester, S.Year, S.Instructor, S.Course\_number}}$
  $\text{Tup} = \text{Tup}(\sigma_{c_2}) = 5000$
- $C \bowtie_{\text{C.Course\_number = S.Course\_number}} S$
  $\text{Tup} = \frac{\text{Tup}_C \cdot \text{Tup}_S}{max(\text{Val}_C(\text{Course\_number}),\text{Val}_S(\text{Course\_number}))} = \frac{1 \cdot 5000}{max(1,100)} = \frac{5000}{100} = 50$

Estimating the size of the temporary relations

- σ_{c_1}
  $\Sigma_1 = 1 \cdot 54 = 54$
- σ_{c_2}
  $\Sigma_2 = 5000 \cdot 58 = 290000$
- $\pi_{\text{C.Course\_name, C.Course\_number}}$
  $\Sigma_3 = 1(20 + 20) = 40$
- $\pi_{\text{S.Semester, S.Year, S.Instructor, S.Course\_number}}$
  $\Sigma_4 = 5000(20 + 10 + 4 + 20) = 270000$
- $C \bowtie_{\text{C.Course\_number = S.Course\_number}} S$
  $\Sigma_5 = 50(20 + 20 + 20 + 10 + 4 + 20) = 4700$

then the estimated size (excluding root node and leaf nodes)
$\Sigma = \Sigma_1 + \Sigma_2 + \Sigma_3 + \Sigma_4 + \Sigma_5 = 54 + 290000 + 40 + 270000 + 4700 = 564794$

so the estimated size of the temporary relations (that are not equal in the two trees) is likely smaller in the modified tree, implying that the modified tree is likely to be more efficient than the original.

References

Akkøk, M. N. (2021a). Database Systems Spring 2021, Week 2.1 (part 1) Summary of DBMS Architecture [Lecture slides].

`https://www.uio.no/studier/emner/matnat/ifi/IN3020/v21/lectures/021-1---dbms-architecture.pdf`

Akkøk, M. N. (2021). Database Systems Spring 2021, Week 2.1 (part 2) [Lecture slides].

`https://www.uio.no/studier/emner/matnat/ifi/IN3020/v21/lectures/021-2---dbms-architecture-dependencies.pdf`

Elmasri, R. & Navathe, S. (2016). Fundamentals of database systems (7th ed.). Boston , Mass: Pearson.

Kostylev, E. V. (2021). Database Systems Spring 2021, Week 2.2 Indexing [Lecture slides].

https://www.uio.no/studier/emner/matnat/ifi/IN3020/v21/lectures/022---indexing.pdf

Kostylev, E. V. (2021a). Database Systems Spring 2021, Week 4.2-5.2 Query Compilation - Parts 1-3 [Lecture slides].

https://www.uio.no/studier/emner/matnat/ifi/IN3020/v21/lectures/042-052---qcomp.pdf

Kostylev, E. V. (2021b). Database Systems Spring 2021, Weeks 6.1-6.2 Efficient Query Execution - Parts 1-2 [Lecture slides].

https://www.uio.no/studier/emner/matnat/ifi/IN3020/v21/lectures/061-062---qeval.pdf

Sam, S. (2018, 22. february). How can we convert subqueries to INNER JOIN? Retrieved from

https://www.tutorialspoint.com/How-can-we-convert-subqueries-to-INNER-JOIN