IN3020 Assignment 1

Espen Lønes

March 10, 2021

Exercise 1)
a)

```
SELECT s.Course_number, s.Semester, s.Year, COUNT(gr.Student_number)
FROM Section s
INNER JOIN Grade_Report gr USING(Section_Identifier)
WHERE s.Instructor = 'King'
GROUP BY s.Course_number, s.Semester, s.Year;
```

b)

```
SELECT st.Name, st.Major
FROM Student st
INNER JOIN Grade_Report gr USING(Student_number)
GROUP BY st.Student_number, st.Name, st.Major
HAVING COUNT(CASE WHEN gr.Grade <> 'A') = 0;
```

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 Student_number = '17';
DELETE FROM Grade_Report WHERE Student_number = '17';
```

Exercise 2)
2.1)

Execution manager:
Reads tuples from memory and is given administrative commands, executable query plans and transactions.
Transaction processing involves concurrency control, log manager, backup, manager and recovery manager.
Reading and writing tuples involves buffer manager and storage manager.

Buffer manager:
Manages blocks in memory. Keeps track of what is in memory and what is not in memory. The other parts of the system request blocks form the buffer manager. It also keeps track of whether a block is 'dirty', in use and how long it stays in memory.

Storage manager:
Controls placement of data on disk and moving data between disk and main memory.

Concurrency control:
Manages and controls concurrency between transactions.
Like avoiding blocking.

Log manager:
Logging updates of database for recovery purposes.
Also used for synchronization.

Backup manager:
Used to create database backups in case a recovery is needed.

Recovery manager:
Recovers the database to a stable state if something went wrong. Using either log or backup.

2.2)

The central components in a DBMS are the components that manage their racecourses directly down to the operating system. These are system buffer manager, lock component and log component. The high level components are those that need the central components as prerequisites. Like transaction management, access path management, sorting component etc.

Interaction between Log and Buffer:
WAL principle, log is written to disk before the actual data is written to disk. Also determines which logging protocol is applicable.

Interaction between Log and Lock:
Ensures ability to preform a 'safe' rollback (no other transactions must be effected).

Interaction between Lock and Buffer:
Main task of lock is to guarantee logical single user mode (isolation). This is done by the lock controlling which pages are fixed and which are unfixed in the buffer.

Exercise 3)
3.1)

B+ trees:
Efficient when doing interval search. For large n it is rarely necessary to split or merge nodes. Disk I/O can be reduced by keeping index blocks i memory. Not so good is that we have to start from the root node every time but the number of levels is usually low (usually 3).

Hash table:
Can be used to fast search on specific search keys. Fewer disk operations than with regular indices and B+ trees. But multiple entries can lead to

more blocks per bucket. Also it is bad for interval search.

k-d trees:
Useful for nearest neighbor searches and multidimensional search keys.

Quad-trees:
Used for partitioning a two dimensional (multidimensional) space like in image compression and spacial search.

R-trees:
Used for indexing multidimensional information such as geographical coordinates, rectangles or polygons.

Grid files:
Good for searches with multiple keys. But it uses a lot of space and needs organizing.

3.2)

An inverted index is used to access all relevant document ID's form a keyword. This allows fast full text search but increases the cost of adding the document to the database. These type of indices are commonly used in search engines and several general purpose database management systems like ADABAS, DATACOM/DB and Model 204.

3.3)

Overflow block(s) can be used to efficiently handle insertion in an ordered file.

Exercise 4)
4.1)

We have:

Purchase(A, B, C)
Supply(A, D, F)

```
SELECT P.C
FROM Purchase P
WHERE P.A IN
    (SELECT S.A FROM Supply S Where S.D > 0);
```

Translate this to EXIST:

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

Since it is an correlated subquery we must add all context relations to the FROM list (of the subquery) and add all parameters to the projection (of the subquery).
We then get that the subquery translates to:

$$\pi_{P.A,\ P.B,\ P.C,\ S.A}(\sigma_{S.D>0\ \wedge\ S.A=P.A}(\rho_P(Purchase)\ \times\ \rho_S(Suply)))$$

We then look at the from-where part of the whole query without subqueries.

$$\rho_P(Purchase)$$

We then use natural join on these two expressions:

$\rho_P(Purchase) \bowtie \pi_{P.A,\ P.B,\ P.C,\ S.A}(\sigma_{S.D>0\ \wedge\ S.A=P.A}(\rho_P(Purchase) \times \rho_S(Suply)))$

And finally translate the remaining projection to get the final expression.

$\pi_{P.C}(\rho_P(Purchase) \bowtie \pi_{P.A,\ P.B,\ P.C,\ S.A}(\sigma_{S.D>0\ \wedge\ S.A=P.A}(\rho_P(Purchase) \times \rho_S(Suply))))$

b)

We have:

$\pi_{P.C}(\rho_P(Purchase) \bowtie \pi_{P.A,\ P.B,\ P.C,\ S.A}(\sigma_{S.D>0\ \wedge\ S.A=P.A}(\rho_P(Purchase) \times \rho_S(Suply))))$

First we split the selection $\sigma_{aANDb}(R) = \sigma_a(\sigma_b(R))$

$\pi_{P.C}(\rho_P(Purchase) \bowtie \pi_{P.A,\ P.B,\ P.C,\ S.A}(\sigma_{S.D>0}(\sigma_{S.A=P.A}[\rho_P(Purchase) \times \rho_S(Suply)])))$

And then we use $\pi_L \sigma_C(P \times S) = P \bowtie S$ where c compares via AND each pair of tuples from P and S with the same name. And L is all attributes appropriately renamed.

$$\pi_{P.C}(\rho_P(Purchase) \bowtie \pi_{P.A,\ P.B,\ P.C,\ S.A}(\sigma_{S.D>0}(\rho_P(Purchase)\ \bowtie\ \rho_S(Suply))))$$

Then move in the selection $\sigma_c(P \bowtie S) = P \bowtie \sigma_c S$ (when it makes sense).

$$\pi_{P.C}(\rho_P(Purchase) \bowtie \pi_{P.A,\ P.B,\ P.C,\ S.A}(\rho_P(Purchase)\ \bowtie\ \sigma_{S.D>0}(\rho_S(Suply))))$$

We then use $\pi_L(R) \subseteq R$

$$\pi_{P.C}(\rho_P(Purchase) \bowtie \rho_P(Purchase)\ \bowtie\ \sigma_{S.D>0}(\rho_S(Suply)))$$

A table natural joined with it self is just itself.

$$\pi_{P.C}(\rho_P(Purchase)\ \bowtie\ \sigma_{S.D>0}(\rho_S(Suply)))$$

And finally $\pi_L(\sigma_C(R)) = \pi_L(\sigma_C(\pi_M(R)))$. If M contains attributes in L and C.

$$\pi_{P.C}(\pi_{P.A,\ P.C}[\rho_P(Purchase)]\ \bowtie\ \pi_{S.A}[\sigma_{S.D>0}(\rho_S(Suply))])$$

c)

One thing is that in the optimized version we selection and projection on the tables before the join/product. Furthermore in the optimized version we do an natural join while the original did an cross product which is makes a much bigger intermediate table.

4.2)
a)

$$\pi_M(\sigma_{<c>\wedge<d>}(P \bowtie Q \bowtie R)) \iff \pi_M((\sigma_{<c>}(P) \bowtie \sigma_{<d>}(Q) \bowtie R))$$

Because C is only in P and D is only in Q.

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

Because natural join is associative. A is only common attribute between P and Q. And H is only common attribute between Q and R. And P and Q don't share any attributes.

b)

I think the expression to the right is the most efficient because, by doing selection and projection before the joins. We (if selection removes a good amount of tuples) greatly decrease the number of comparisons done in during the joins. (aka. the intermediate tables are smaller)

Exercise 5)
5.1)

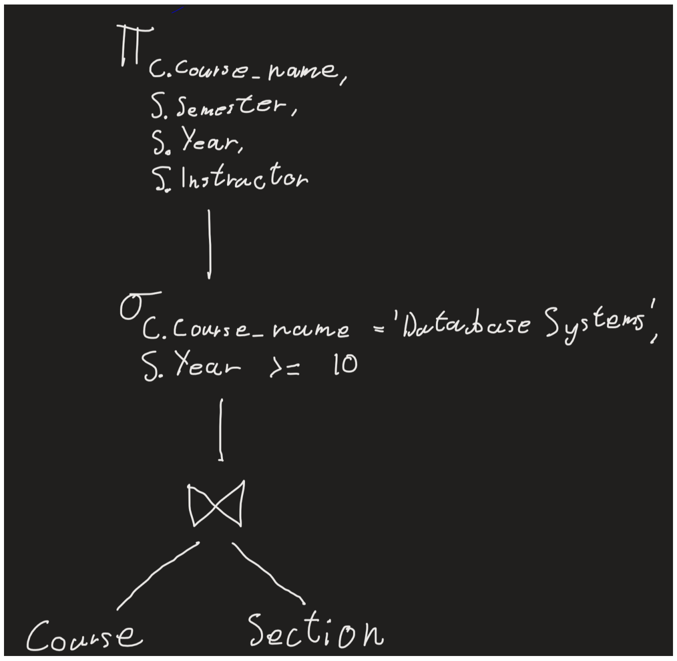
5.2)
a)

```
SELECT c.Course_name, s.Semester, s.Year, s.Instructor
FROM Course c
INNER JOIN Section s USING(Course_number)
WHERE c.Course_name = 'Database Systems'
AND s.Year >= 10;
```

b)

$\pi_{C.Course\_name,\ S.Semester,\ S.Year,\ S.Instructor}($
$\sigma_{C.Course\_name='DatabaseSystems'\ \wedge\ S.Year\geq 10}(\rho_C(Course) \bowtie \rho_S(Section)))$

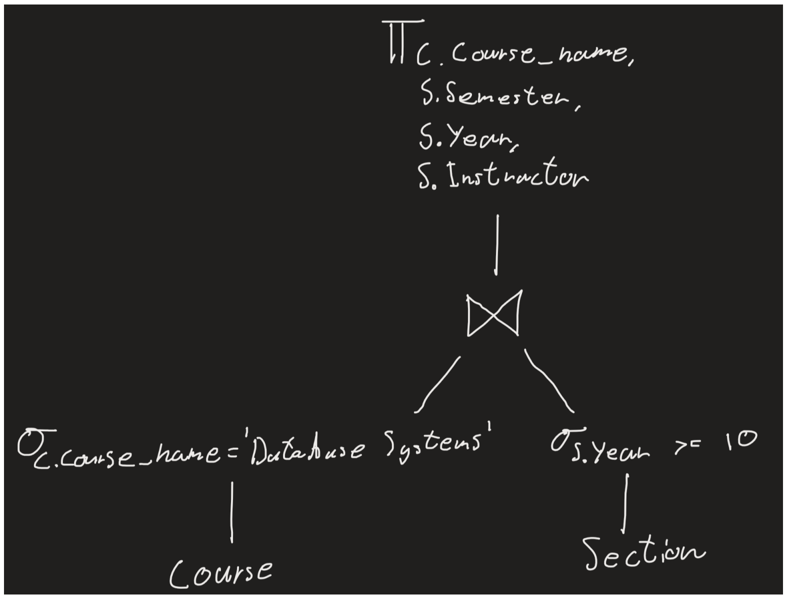
c)


π C.Course_name, S.Semester, S.Year, S.Instructor
σ C.Course_name = 'Database Systems', S.Year >= 10
$\bowtie$
Course
Section

d)

Split the selection and push down each part.

$\pi_{C.Course\_name,\ S.Semester,\ S.Year,\ S.Instructor}($
$\sigma_{C.Course\_name='DatabaseSystems'}(\rho_C(Course)) \bowtie \sigma_{S.Year \geq 10}(\rho_S(Section)))$


$\Pi_{C.Course\_name,\ S.Semester,\ S.Year,\ S.Instructor}$
$\bowtie$
$\sigma_{C.course\_name='Database\ Systems'}$
$\sigma_{S.Year >= 10}$
Course
Section

e)