Yohandi [SID: 1200400025]

CSC3170 - Assignment 1

Q1 (a) $\sigma_{p(\alpha)}(r \bowtie_{r.\beta = s.\beta} s) = \sigma_{p(\alpha)}(\sigma_{r.\beta = s.\beta}(r \times s))$

$= \sigma_{p(\alpha)}(\sigma_{r.\beta = s.\beta}(r) \times \sigma_{r.\beta = s.\beta}(s))$

$= \sigma_{p(\alpha)}(\sigma_{r.\beta = s.\beta}(r)) \times \sigma_{p(\alpha)}(\sigma_{r.\beta = s.\beta}(s))$

$= \sigma_{r.\beta = s.\beta}(\sigma_{p(\alpha)}(r)) \times \sigma_{r.\beta = s.\beta}(\sigma_{p(\alpha)}(s))$

$= \sigma_{r.\beta = s.\beta}(\sigma_{p(\alpha)}(r)) \times \sigma_{r.\beta = s.\beta}(s)$

$= \sigma_{r.\beta = s.\beta}(\sigma_{p(\alpha)}(r) \times s)$

$= \sigma_{p(\alpha)}(r) \bowtie_{r.\beta = s.\beta} s$ (both queries are equivalent)

(b) -> Query 1:

- Perform Cartesian product between r and s requires $|r||s| = mn$ operations
- Check $r.\beta = s.\beta$ and $p(\alpha)$ requires mn operations

(counted as one)

=> 2mn operations in total

-> Query 2:

- Check $p(\alpha)$ requires m operations
- If $p(\alpha)$ doesn't satisfy the condition, 0 operation required
- If $p(\alpha)$ satisfies the condition, join requires $m|s| = mn$ operations

=> $[m, m+mn]$ operations in total $= m + fmn$ operations in total, where $0 \leq f \leq 1$

Different number of operations are given by:

$2mn - (m + fmn)$, where $0 \leq f \leq 1 \geq 2mn - m - mn = mn - m = m(n-1) = |r|(|s|-1)$

Notice that $|r|(|s|-1)$ is larger than or equal to 0 when s is not empty; however, $|r|(|s|-1)$ is smaller than 0 when s is empty. As a result, we conclude that:

=> If s is empty, query 1 is more efficient (less operations required)

=> If s is not empty, query 2 is more efficient (less operations required)

However, with assumptions $m = |r| \geq 1$ and $n = |s| \geq 1$, we conclude that query 2 is more efficient than query 1.

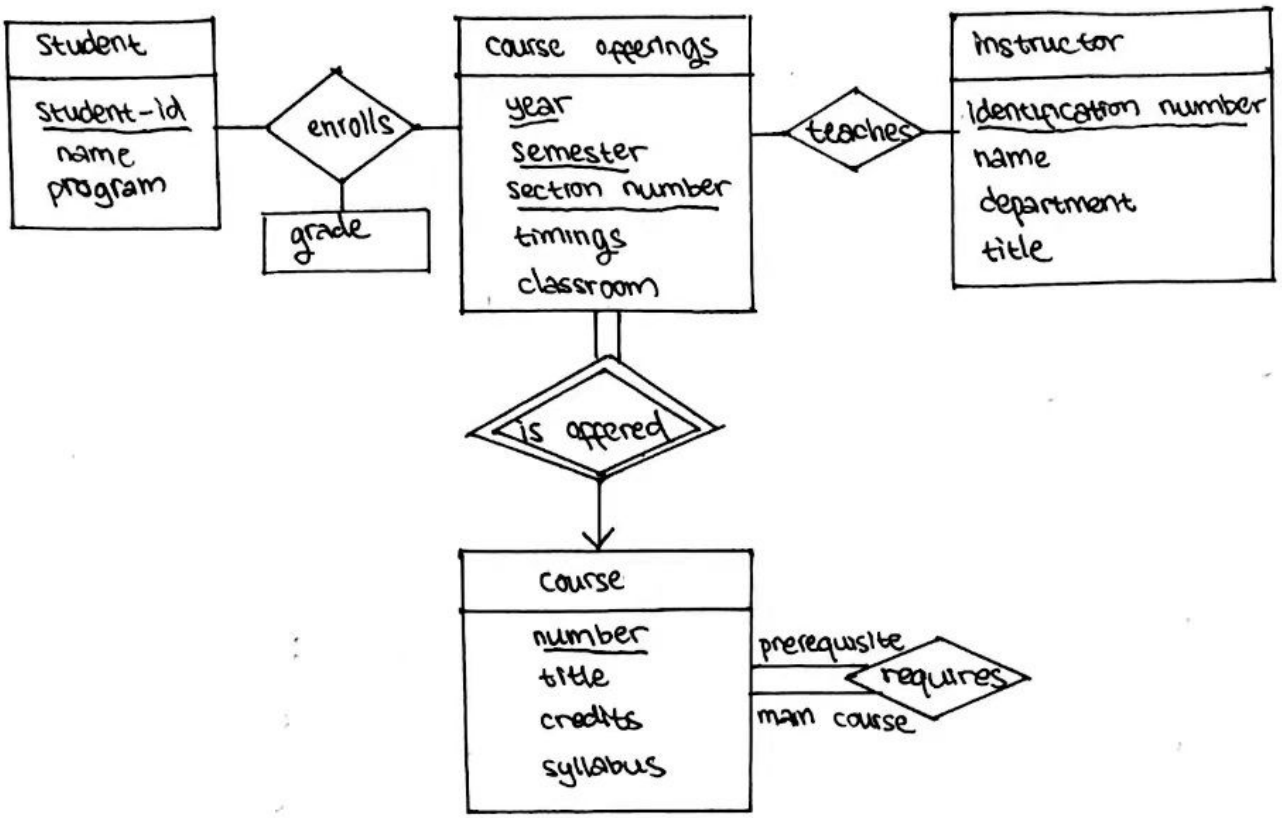
(c) In slide 2.30,

-> Query 1 uses a join operation ($\bowtie$) to combine the "instructor" and "teaches" table based on the condition that the "ID" column in the "instructor" table is equal to the "ID" column in the "teaches" table, and then applies a selection (σ) operation to only return the rows where the "dept_name" column in the resulting table is "Physics"

-> Query 2 also applies a selection operation to the "instructor" table to only return the rows where the "dept_name" column is "Physics", and then uses a join operation to combine the resulting table with the "teaches" table based on the same condition as in Query 1

Although both queries are not identical, they are in fact equivalent (proved in part (a)); which implies, they give the same result on any database. Moreover, it is also a fact that query optimizers in database systems typically look at result of expression computation and find an efficient way of computing that result, rather than following the exact same sequence of steps specified in the query. This implies that part (b) in this question is well-considered by the system that looks for efficiency for the mechanism

Q2 (a)



Assumptions made:

- Prerequisites attribute is treated separately

(b) student (student-id, name, program)

course (number, title, credits, syllabus)

course offerings (number, year, semester, section number, timings, classroom)

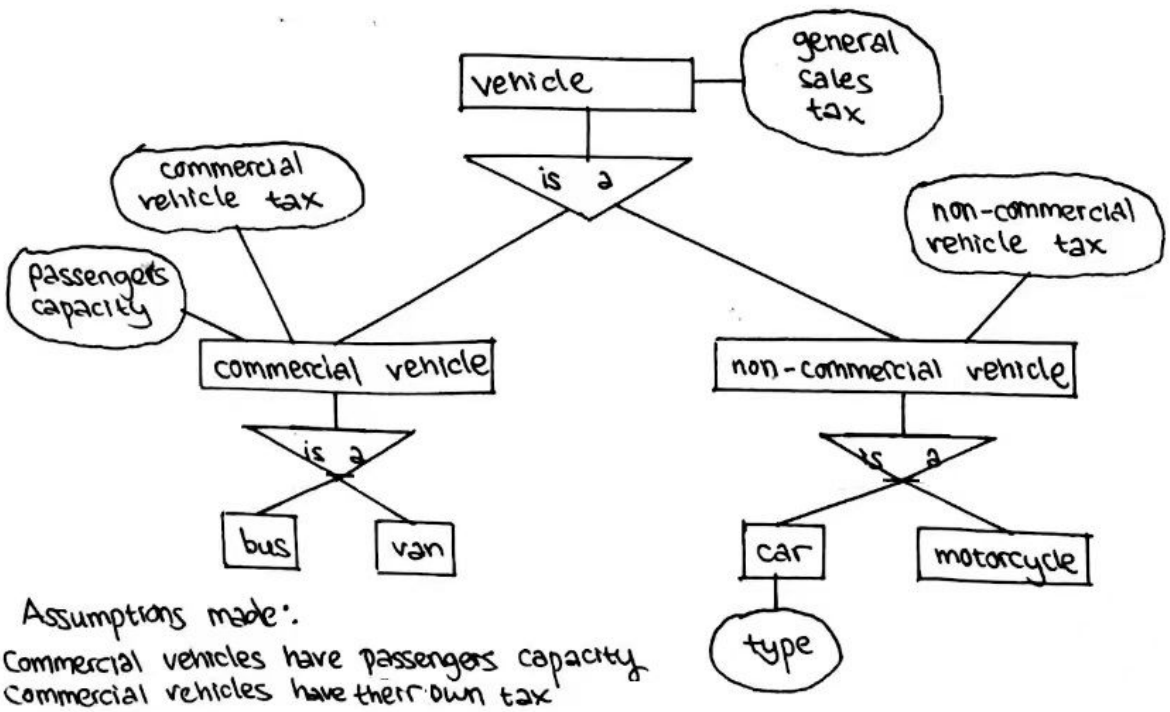
instructor (identification number, name, department, title)

enrolls (student-id, number, year, semester, section number, grade)

teaches (identification number, number, year, semester, section number)

requires (main course, prerequisite)

Q3.



Assumptions made:

- Commercial vehicles have passengers capacity
- Commercial vehicles have their own tax
- Non-commercial vehicles have their own tax
- All vehicles have general sales tax
- Cars have type

note that: for simplicity, I wrote $A \bowtie_{A.\alpha = B.\beta} B$ as $A \bowtie_{\alpha=\beta} B$

Q4. (1) Select "Research" department:

$\sigma_{Dname = "Research"}(DEPARTMENT)$

(2) Join EMPLOYEE with (1) based on Dno and Dnumber, respectively:

$EMPLOYEE \bowtie_{Dno = Dnumber} (1)$

(3) Project Fname, Minit, Lname, Address from (2):

$\pi_{Fname, Minit, Lname, Address}((2))$

$RESULT \leftarrow \pi_{Fname, Minit, Lname, Address}(EMPLOYEE \bowtie_{Dno = Dnumber}(\sigma_{Dname = "Research"}(DEPARTMENT)))$

Q5. (1) Select "Stafford" project:

$\sigma_{Plocation = "stafford"}(PROJECT)$

(2) Join DEPARTMENT with (1) based on Dnumber and Dnum, respectively:

$DEPARTMENT \bowtie_{Dnumber = Dnum} (1)$

(3) Join EMPLOYEE with (2) based on Ssn and Mgr_ssn, respectively:

$EMPLOYEE \bowtie_{Ssn = Mgr\_ssn} (2)$

(4) Project Pnumber, Lname, Address, Bdate from (3):

$\pi_{Pnumber, Lname, Address, Bdate}((3))$

$RESULT \leftarrow \pi_{Pnumber, Lname, Address, Bdate}(EMPLOYEE \bowtie_{Ssn = Mgr\_ssn}(DEPARTMENT \bowtie_{Dnumber = Dnum}(\sigma_{Plocation = "Stafford"}(PROJECT))))$

Q6. (1) Select "Smith" employee:

$\sigma_{Lname = "Smith"}(EMPLOYEE)$

(2) Project Ssn from (1):

$\pi_{Ssn}((1))$

(3) Join WORKS_ON with (2) based on Essn and Ssn, respectively:

$WORKS\_ON \bowtie_{Essn = Ssn} (2)$

(4) Project Pnumber from (3):

$\pi_{Pnumber}((3))$

(5) Join EMPLOYEE with DEPARTMENT based on Ssn and Mgr_ssn, respectively:

$EMPLOYEE \bowtie_{Ssn = Mgr\_ssn} DEPARTMENT$

(6) Join PROJECT with (5) based on Dnum and Dnumber, respectively:

$PROJECT \bowtie_{Dnum = Dnumber} (5)$

(7) Select "Smith" manager:

$\sigma_{Lname = "smith"}((6))$

(8) Project Pnumber from (7):

$\pi_{Pnumber}((7))$

(9) Union (4) and (8):

$(4) \cup (8)$

$RESULT \leftarrow \pi_{Pnumber}(WORKS\_ON \bowtie_{Essn = Ssn}(\pi_{Ssn}(\sigma_{Lname = "smith"}(EMPLOYEE)))) \cup \pi_{Pnumber}(\sigma_{Lname = "Smith"}(PROJECT \bowtie_{Dnum = Dnumber}(EMPLOYEE \bowtie_{Ssn = Mgr\_ssn}(DEPARTMENT))))$

27. (1) Join DEPENDENT with EMPLOYEE based on Essn and Ssn, respectively:

DEPENDENT $\bowtie_{Essn=Ssn}$ EMPLOYEE

(2) Project Fname, Minit, Lname from (1):

$\Pi_{Fname, Minit, Lname}((1))$

(3) Project Fname, Minit, Lname from EMPLOYEE:

$\Pi_{Fname, Minit, Lname}(EMPLOYEE)$

(4) Set difference between (3) and (2):

(3) - (2)

RESULT $\leftarrow (\Pi_{Fname, Minit, Lname}(EMPLOYEE)) - (\Pi_{Fname, Minit, Lname}(DEPENDENT \bowtie_{Essn=Ssn} EMPLOYEE))$

28. (1) Project Essn from DEPENDENT:

$\Pi_{Essn}(DEPENDENT)$

(2) Rename attribute of (1) to Ssn:

$\rho_{(Ssn')}((1))$

(3) Project Mgr_ssn from DEPARTMENT:

$\Pi_{Mgr\_ssn}(DEPARTMENT)$

(4) Rename attribuse of (3) to Ssn:

$\rho_{(Ssn')}((3))$

(5) Set-intersection between (2) and (4):

(2) $\cap$ (4)

(6) Join EMPLOYEE with (5) based on Ssn and Ssn', respectively:

EMPLOYEE $\bowtie_{Ssn=Ssn'}$ (5)

(7) Project Fname, Minit, Lname from (6):

$\Pi_{Fname, Minit, Lname}((6))$

RESULT $\leftarrow \Pi_{Fname, Minit, Lname}(EMPLOYEE \bowtie_{Ssn=Ssn'} [\rho_{(Ssn')}(\Pi_{Essn}(DEPENDENT))] \cap \rho_{(Ssn')}(\Pi_{Mgr\_ssn}(DEPARTMENT)))$

Q9. (1) Select "James" employee:

$\sigma_{Fname="James"}(EMPLOYEE)$

(2) Select "Borg" employee from (1):

$\sigma_{Lname="Borg"}((1))$

(3) Project Ssn from (2):

$\Pi_{Ssn}((2))$

(4) Join EMPLOYEE with (3) based on super_ssn and Ssn, respectively:

$EMPLOYEE \bowtie_{super\_ssn=Ssn} (3)$

$RESULT \leftarrow EMPLOYEE \bowtie_{super\_ssn=ssn}(\Pi_{Ssn}(\sigma_{Lname="Borg"}(\sigma_{Fname="James"}(EMPLOYEE))))$

Q10. (1) Select "James" employee:

$\sigma_{Fname="James"}(EMPLOYEE)$

(2) Select "Borg" employee from (1):

$\sigma_{Lname="Borg"}((1))$

(3) Project Ssn from (2):

$\Pi_{Ssn}((2))$

(4) Join EMPLOYEE with (3) based on super_ssn and ssn, respectively:

$EMPLOYEE \bowtie_{super\_ssn=Ssn} (3)$

(5) Project Ssn from (4):

$\Pi_{Ssn}((4))$

(6) Join EMPLOYEE with (5) based on super_ssn and ssn, respectively:

$EMPLOYEE \bowtie_{super\_ssn=Ssn} (5)$

$RESULT \leftarrow EMPLOYEE \bowtie_{super\_ssn=Ssn}(\Pi_{Ssn}(EMPLOYEE \bowtie_{super\_ssn=ssn}(\Pi_{Ssn}(\sigma_{Lname="Borg"}($

$\sigma_{Fname="James"}(EMPLOYEE))))))$

If the levels are fixed (for some constant value), it is possible (simply perform recursion manually). However, if the levels are not fixed (for example, n), it is not possible as it requires looping mechanism. Also, a problem might occurs when the hierarchy levels do not satisfy the condition.