Chenhao Wu
Computer Science and Engineering
117010285



**Problem 1.**
Consider the database schema:

**STUDENT**(*Name, Student_number, Class, Major*),
**COURSE**(*Course_name, Course_number, Credit_hours, Department*),
**SECTION**(*Section_identifier, Course_number, Semester, Year, Instructor*),
**PREREQUISITE**(*Course_number, Prerequisite_number*),

where example values of *Department* are: CS, MATH, MUSIC, and example values for *Course_number* are: CS3130, MATH1003, MUSIC2343 (i.e. each *Course_number* has the *Department* as prefix).

**a.** If the name of the 'CS' (Computer Science) Department changes to 'CSSE' (Computer Science and Software Engineering) Department and the corresponding prefix for the course number also changes, identify the columns in the database that would need to be changed.
**b.** Can you restruct the columns in **COURSE**, **SECTION**, and **PREREQUISITE** tables so that only one column will need to be changed?

***Solution:***
**a.** If the name of CS changes to CSSE, the following columns need to be changed

- **STUDENT**: *Major*
- **COURSE**: *Course_number, Department*
- **SECTION**: *Course_number*
- **PREREQUISITE**: *Course_number, Prerequisite_number*

**b.** One possible approach is to add a new column *Course_department* for the prefix code, i.e. reconstruct the tables as following:

**COURSE**(*Course_name, Course_department, Course_number, Credit_hours*),
**SECTION**(*Section_identifier, Course_department, Course_number, Semester, Year, Instructor*),
**PREREQUISITE**(*Course_department, Course_number, Prerequisite_number*)

where the previous course code is split into two part, [*Course_department*][*Course_number*], such that in this new database system, only *Course_department* column need to be changed. □

**Problem 2.**
In relational algebra, the **DIVISION** operation, denoted by $\div$ is useful for certain kinds of queries that sometimes occur. Let $r(R)$ and $s(S)$ be two relations, and let $S \subseteq R$, i.e. every attribute of schema $S$ is also in schema $R$. The relation $r(R) \div s(S)$ is a relation on schema $R - S$ (i.e. on the schema containing all attributes of schema $R$ that are not in schema S). A tuple $t$ is in $r \div s$ if and only if the following two conditions hold:

1. $t$ is in $\Pi_{R-S}(r)$

2. For every tuple $t_s$ in $s$, there is a tuple $t_r$ in $r$, satisfying both of the following:
   (a) $t_r[S] = t_s[S]$
   (b) $t_r[R - S] = t$

   where the notation $t_r[S]$ means the $S$ attributes of tuple $t_r$

Express the **DIVISION** operation in terms of other relational algebra operators.

***Solution:***
From observation, the **DIVISION** operation finds out the relation of attributes s.t.

1. the result attribute are in $r$ but not in $s$
2. the result attribute are connected with all tuples in $s$

Based on this understanding, we conduct the **DIVISION** operation as following.
Assuming that $a_1, \ldots, a_n$ are attributes unique to relation $r$, and $b_1, \ldots, b_m$ are attributes in relation $s$.
Firstly, we would like to construct an extended table, which consists of all possible combinations between relation $s$ and attributes $a_1, \ldots, a_n$. The process is formulated as

$$e(a_1, \ldots, a_n, b_1, \ldots, b_m) \leftarrow \Pi_{a_1,\ldots,a_n}(r) \times s$$

Since this extended table is not "complete", which means it may not contain tuples that actually in relation $r$, we then find out these missing tuples using

$$m(a_1, \ldots, a_n, b_1, \ldots, b_m) \leftarrow r - e$$

To obtain tuples existing in relation $e$, we then project the relation $m$ with attributes $a_1, \ldots, a_n$ and subtract it with relation $r$

$$d(a_1, \ldots, a_n) \leftarrow \Pi_{a_1,\ldots,a_n}(r) - \Pi_{a_1,\ldots,a_n}(m)$$

Finally, one can verify that all the tuples in the resulting relation $d$ satisfy the constraints of **DIVISION** operation.
Combining all the procedures above we have

$$\textbf{DIVISION}(r, s) = \Pi_{a_1,\ldots,a_n}(r) - \Pi_{a_1,\ldots,a_n}(r - \Pi_{a_1,\ldots,a_n}(r) \times s)$$

□

**Problem 3.**
Consider the following database schema, where the primary keys are underlines.

**EMPLOYEE**(*Fname, Minit, Lname, Ssn, Bdate, Address, Sex, Salary, Super_ssn, Dno*),
**DEPARTMENT**(*Dname, Dnumber, Mgr_ssn, Mgr_start_date*),
**DEPT_LOCATIONS**(*Dnumber, Dlocation*),
**WORKS_ON**(*Essn, Pno, Hours*),
**PROJECT**(*Pname, Pnumber, Plocation, Dnum*),
**DEPENDENT**(*Essn, Dependent_name, Sex, Bdate, Relationship*)

where *Fname* signifies first name; *Minit*, middle initial; *Lname*, last name; *Ssn*, *Essn* are social security numbers; *Super_ssn* is the social security number of the supervisor; *Dname*, *Dnum*, *Dno*, *Dlocation* are department name, number and location (similarly for project); while other attributes have an obvious interpretation.

In answering the following questions, you may use any of the relational Algebra operations, including the natural join and division. (The natural join is equality join with the redundant column removed and may be simply denoted by *)

**a.** Retrieve the name and address of all employees who work for the "Research" department.

***Solution:***

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

□

**b.** For every project located in "Stafford", list the project number, the controlling department manager's last name, address, and birth date.

***Solution:***

$$\textbf{RESULT}(\textit{Pnumber, Dnum, Lname, Address, Bdate}) \leftarrow \Pi_{\textit{Pnumber, Dnum, Lname, Address, Bdate}}(\sigma_{\textit{Plocation}=\text{"Stanford"}}( (\textbf{PROJECT} \bowtie_{\textit{Dnum}=\textit{Dlocation}} \textbf{DEPTARTMENT}) \bowtie_{\textit{Mgr\_ssn}=\textit{Ssn}} \textbf{EMPLOYEE} ))$$

□

**c.** Find the names of employees who work on all the projects controlled by the department number 5.

***Solution:***

$$\textbf{DEPT\_5\_PROJS}(\textit{Pno}) \leftarrow \Pi_{\textit{Pnumber}}(\sigma_{\textit{Dnum}=5}(\textbf{PROJECT}))$$
$$\textbf{RESULT\_SSN}(\textit{Essn}) \leftarrow \Pi_{\textit{Essn, Pno}}(\textbf{WORK\_ON}) \div \textbf{DEPT\_5\_PROJS}$$
$$\textbf{RESULT}(\textit{Fname, Minit, Lname}) \leftarrow \Pi_{\textit{Fname, Minit, Lname}}(\textbf{RESULT\_SSN} \bowtie_{\textit{Essn}=\textit{Ssn}} \textbf{EMPLOYEE})$$

□

**d.** Make a list of project numbers for projects that involve an employee whose last name is "Smith".

***Solution:***

$$\textbf{RESULT}(\textit{Pno}) \leftarrow \Pi_{\textit{Pno}}(\textbf{WORK\_ON} \bowtie_{\textit{Essn}=\textit{Ssn}} \Pi_{\textit{Ssn}}(\sigma_{\textit{Lname}=\text{"Smith"}}(\textbf{EMPLOYEE})))$$

□

**e.** Retrieve the names of employees who have no dependents.

***Solution:***

$$\begin{aligned}
\textbf{ALL_EMPLOYEE}(Ssn) &\leftarrow \Pi_{Ssn}(\textbf{EMPLOYEE}) \\
\textbf{EMPLOYEE_WITH_DEP}(Ssn) &\leftarrow \Pi_{Essn}(\textbf{DEPENDENT}) \\
\textbf{EMPLOYEE_WITHOUT_DEP}(Ssn) &\leftarrow \textbf{ALL_EMPLOYEE} - \textbf{EMPLOYEE_WITH_DEP} \\
\textbf{RESULT}(Fname, Minit, Lname) &\leftarrow \\
&\quad \Pi_{Fname,\ Minit,\ Lname}(\textbf{EMPLOYEE} * \textbf{EMPLOYEE_WITHOUT_DEP})
\end{aligned}$$

□

**f.** List the names of managers who have at least one dependent.

***Solution:***

$$\begin{aligned}
\textbf{ALL_MANAGER}(Ssn) &\leftarrow \Pi_{Mgr_ssr}(\textbf{DEPARTMENT}) \\
\textbf{EMPLOYEE_WITH_DEP}(Ssn) &\leftarrow \Pi_{Essn}(\textbf{DEPENDENT}) \\
\textbf{MANGER_WITH_DEP}(Ssn) &\leftarrow \textbf{ALL_MANAGER} \cap \textbf{EMPLOYEE_WITH_DEP} \\
\textbf{RESULT}(Fname, Minit, Lname) &\leftarrow \Pi_{Fname,\ Minit,\ Lname}(\textbf{EMPLOYEE} * \textbf{MANAGER_WITH_DEP})
\end{aligned}$$

□

**g.** Find all employees directly supervised by "James Borg".

***Solution:***

$$\begin{aligned}
\textbf{JB_SSN}(Ssn) &\leftarrow \Pi_{Ssn}(\sigma_{Fname=\text{"James"}\ \textbf{and}\ Lname=\text{"Borg"}}(\textbf{EMPLOYEE})) \\
\textbf{RESULT}(Fname, Minit, Lname) &\leftarrow \Pi_{Fname,\ Minit,\ Lname}(\textbf{EMPLOYEE} \bowtie_{Super_ssn\ =\ Ssn} \textbf{JB_SSN})
\end{aligned}$$

□

**h.** Find all employees directly supervised by those directly supervised by "James Borg." Would it be possible to find all employees supervised by a given employee at all levels?

***Solution:***

$$\begin{aligned}
\textbf{JB_SSN}(Ssn) &\leftarrow \Pi_{Ssn}(\sigma_{Fname=\text{"James"}\ \textbf{and}\ Lname=\text{"Borg"}}(\textbf{EMPLOYEE})) \\
\textbf{1th_CONN}(Ssn) &\leftarrow \textbf{EMPLOYEE} \bowtie_{Super_ssn\ =\ Ssn} \textbf{JB_SSN} \\
\textbf{RESULT}(Fname, Minit, Lname) &\leftarrow \Pi_{Fname,\ Minit,\ Lname}(\textbf{EMPLOYEE} \bowtie_{Super_ssn\ =\ Ssn} \textbf{1th_CONN})
\end{aligned}$$

If this company maintains a tree hierarchy (which means there is no *cyclic* in supervision relation), it is possible to find all employees supervised by a given employee at all levels. By recursively performing the second query above and terminating *once query returns* ***null***, employees at all level heredity are obtained. □