CSCI 6333/6315 Database Design and Implementation

Spring 2020

ASSIGNMENT 2: Formal Relational Query Languages

1. (120) Consider the employee database of Fig. 1, where the primary keys are underlined. For each of the following questions, give (1) a relational algebra expression, (2) a tuple relational calculus expression, and (3) a domain relational calculus expression. Note that each problem has 15 points with 5 points for each of the three required expressions.

    a. Find the names of all employees who work for First Bank Corporation.

    a.1. Relational Algebra

$$\Pi_{\text{employee-name}}\left(\sigma_{\substack{employee.employee-id=works.employee-id \wedge \\ works.company-id=company.company-id \wedge \\ company-name="First\ Bank\ Corporation"}}(\text{employee} \times \text{works}) \times \text{company})\right)$$

    a.2. Tuple relational calculus

$$\{t \mid \exists s \in \text{employee}, \exists u \in works, \exists v \in \text{company}\ (t[\text{employee-name}] = s[\text{employee-name}] \wedge s[\text{employee-id}] = u[\text{employee-id}] \wedge u[\text{company-id}] = v[\text{company-id}] \wedge v[\text{company-name}] = \text{'First Bank Corporation'})\}$$

    a.3. Domain relational calculus

$$\{< en > \mid \exists eid, \exists st, \exists ec, \exists cid, \exists sal, \exists cn, \exists cc\ (< eid, en, st, ec > \in employee \wedge < eid, cid, sal > \in works, < cid, cn, cc > \in company \wedge\ cn = 'First\ Bank\ Corporation')\}$$

    b. Find the names and cities of residence of all employees who work for First Bank Corporation.

    b.1. Relational Algebra

$$\Pi_{\text{employee-name, city}}\left(\sigma_{\substack{employee.employee-id=works.employee-id \wedge \\ works.company-id=company.company-id \wedge \\ company-name="First\ Bank\ Corporation"}}(\text{employee} \times \text{works}) \times \text{company})\right)$$

    b.2. Tuple relational calculus

$$\{t \mid \exists s \in \text{employee}, \exists u \in works, \exists v \in \text{company}\ (t[\text{employee-name}] = s[\text{employee-name}] \wedge t[city] = s[city] \wedge s[\text{employee-id}] = u[\text{employee-id}] \wedge u[\text{company-id}] = v[\text{company-id}] \wedge v[\text{company-name}] = \text{'First Bank Corporation'})\}$$

b.3. Domain relational calculus

$$\{< en, ec > \mid \exists eid, \exists st, \exists cid, \exists sal, \exists cn, \exists cc\ (< eid, en, st, ec > \in employee \wedge < eid, cid, sal > \in works, < cid, cn, cc > \in company \wedge\ cn = \text{'First Bank Corporation'})\}$$

c. Find the names, street addresses, and cities of residence of all employees who work for First Bank Corporation and earn more than $10,000.

c.1. Relational Algebra

$$\Pi_{\text{employee-name, street, city}}(\ \sigma_{\substack{\text{company-name="First Bank Corporation"} \\ and\ salary > 10000}}\ (\text{employee} \bowtie \text{works}) \bowtie \text{company}))$$

c.2. Tuple relational calculus

$$\{t \mid \exists s \in \text{employee}, \exists u \in works, \exists v \in \text{company}\ (t[\text{employee-name}] = s[\text{employee-name}] \wedge t[city] = s[city] \wedge t[street] = s[street] \wedge\ s[\text{employee-id}] = u[\text{employee-id}] \wedge u[\text{company-id}] = v[\text{company-id}] \wedge u[salary] > 10000 \wedge v[\text{company-name}] = \text{'First Bank Corporation'})\}$$

c.3. Domain relational calculus

$$\{< en, st, ec > \mid \exists eid,\ \exists cid, \exists sal, \exists cn, \exists cc\ (< eid, en, st, ec > \in employee \wedge < eid, cid, sal > \in works \wedge < cid, cn, cc > \in company \wedge\ sal > 10000 \wedge\ cn = \text{'First Bank Corporation'})\}$$

d. Find all employees in the database who live in the same cities as the companies for which they work.

d.1. Relational Algebra

$\Pi_{\text{employee-name, city}}$ (employee ⋈ works) ⋈ company)

d.2. Tuple relational calculus

$$\{t \mid \exists s \in \text{employee}, \exists u \in works, \exists v \in \text{company}\ (t[\text{employee-name}] = s[\text{employee-name}] \wedge t[city] = s[city] \wedge s[\text{employee-id}] = u[\text{employee-id}] \wedge u[\text{company-id}] = v[\text{company-id}] \wedge t[city] = v[\text{city}]\}$$

d.3. Domain relational calculus

$$\{< en, ec > \mid \exists eid, \exists st, \exists cid, \exists sal, \exists cn\ (< eid, en, st, ec > \in employee \wedge < eid, cid, sal > \in works \wedge < cid, cn, ec > \in company)\}$$

e. Find all employees in the database who live in the same cities and on the same streets as do their managers.

e.1. Relational algebra
//find managers along their addresses

$$\text{manager}(\text{ID}, \text{street}, \text{city}) = \Pi_{\textit{employee.employee-id, street, employee.city}} (\sigma_{\textit{employee.employee-id=manages.manager-id}}(employee \times manages))$$

//find those employees who live in the same cities and
//on the same street as do their managers.

$$\Pi_{\textit{employee.employee-name, street, employee.city}} (\sigma_{\textit{manages.manager-id=manager.ID and employee.street = manager.street and employee.city = manager.city}} (employee \bowtie manages \times manager))$$

e.2. Tuple relational calculus

$$\{t \mid \exists e \in \text{employee}, \exists m \in manages, \exists me \in employee, (t[\text{employee-name}] = e[\text{employee-name}] \wedge t[city] = e[city] \wedge t[street] = e[street] \wedge e[\text{employee-id}] = m[\text{employee-id}] \wedge m[\text{manager-id}] = me[\text{employee-id}] \wedge t[city] = me[city] \wedge t[street] = me[street])\}$$

e.3. Domain relational calculus

$$\{< tn, tst, tc > | \exists eid,\ \exists mid, \exists mn\ (< eid, tn, tst, tc > \in employee \wedge < eid, mid > \in manages \wedge < mid, mn, tst, t \in employee)\}$$

f. Find all employees in the database who do not work for the First Bank Corporation.

f.1. Relational algebra
//find all employees

$$allEmployee(ID, name) = \Pi_{\textit{employee-id, employee-name}}\ (employee)$$

//find all employees working for the First Bank Corporation

$$allFBCEmployee(ID, name) = \Pi_{\textit{employee-id, employee-name}} (\sigma_{\textit{EW.company-id = company.company-id} \wedge \textit{company.company-name="First Bank Corporation"}} \rho_{EW}(employee \bowtie works) \times company))$$

//finally, find those who do not work for First Bank Corporation
allEmploee – allFBCEmployee

f.2. Tuple calculus

$$\{t | \exists e \in \text{employee}, \forall w \in works, \forall c \in company\ (t[\text{employee-id}] = e[\text{employee-id}] \wedge t[\text{employee-name}] = e[\text{employee-name}] \wedge (e[\text{employee-id}] = w[\text{employee-id}] \wedge w[\text{company-id}] = c[\text{company-id}] \Rightarrow c[\text{company-name}] \neq \text{"First Bank Corporation"}))\}$$

f.3. Domain calculus

$$\{< tn, tst, tc > | \exists eid,\ \forall cid, \forall sal,\ \forall city, \forall cn\ (< eid, tn, tst, tc > \in employee \wedge\ (< eid, cid, sal > \in works \wedge < cid, cn, city > \in company \Rightarrow cn \neq \textit{'First Bank Corporation'}))\}$$

g. Find all employees in the database who earn more than each employee of Small Bank Corporation.

g.1. Relational algebra
//find Small Bank Corporation employee salaries

$$salarySBC(\text{employee-id, salary}) = \Pi_{\textit{employee-id, salary}}\ \sigma_{\textit{company-name="Small Bank Corporation"}}(works \bowtie company)$$

//find all employees who earn less than some
//Small Bank Corporation //employees

$$salaryEmp(employee\text{-}id, salary) = \Pi_{employee\text{-}id,\ salary}\ \sigma_{works.salary < salarySBC.salary}(works \times salarySBC)$$

//find those employees who earns more than each employee of small
//Bank Corporation

$$\Pi_{employee\text{-}id,\ employee\text{-}name}(employee \bowtie works) - \Pi_{employee\text{-}id,\ employee\text{-}name}(employee \bowtie salaryEmp)$$

g.2. Tuple calculus

$$\{t \mid \exists e \in \text{employee}, \exists w \in works, \forall u \in works, \forall c \in company\ (t[\text{employee-id}] = e[\text{employee-id}] \wedge t[\text{employee-name}] = e[\text{employee-name}] \wedge e[\text{employee-id}] = w[\text{employee-id}] \wedge (u[\text{company-id}] = c[\text{company-id}] \wedge c[company\text{-}name] = \text{"Small Bank Corporation"} \Longrightarrow w[\text{salary}] > u[\text{salary}])\}$$

g.3. Domain calculus

$$\{< eid, en > \mid \exists st,\ \exists c, \exists cid, \exists esal, \forall xeid, \forall xcid, \forall salx,\ \forall xc, \forall xn\ (< eid, en, st, c > \in employee \wedge < eid, cid, esal > \in work \wedge (< xeid, xcid, xsal > \in works \wedge < xcid, xn, xn > \wedge\ xn = \text{'First Bank Corporation'} \Longrightarrow esal > xsal))\}$$

h. Assume that the companies may be located in several cities. Find all companies located in every city in which Small Bank Corporation is located.

h.1. Relational algebra

//find cities Small Bank Corporation is located

$$citySBC(city) = \Pi_{city}\ \sigma_{company\text{-}name="Small\ Bank\ Corporation"}(company)$$

//find all companies located in every of those cites.

$$\Pi_{company\text{-}name}\ company - \Pi_{company\text{-}name}((\Pi_{company\text{-}name}\ company) \times \text{citySBC} - \Pi_{\text{company-name, city}}(company))$$

h.2. Tuple calculus

$$\{t | \exists s \in company, \forall c \in company, \exists w \in company \left(t[\text{company-name}] = s[\text{company-name}] \wedge (c[\text{company-name}] = \text{"}Small\ Bank\ Corporation\text{"} \Longrightarrow w[city] = c[city] \wedge w[\text{company-name}] = t[\text{company-name}])\right)\}$$

h.3. Domain calculus

$$\{< n > | \exists id, \exists c, \forall cid, \forall cn, \forall cc, \exists wid, \exists wc\ (< id, n, c > \in company \wedge < cid, cn, cc > \in company\ ] \wedge < wid, n, wc > \in company \wedge (cn = \text{First Bank Corporation} \Longrightarrow cc = wc)\}$$

employee(<u>employee-id</u>, employee-name, street, city)
works(<u>employee-id</u>, company-id, salary)
company(<u>company-id</u>, company-name, city)
manages(<u>employee-id</u>, manager-id)

Figure 1. Employee database

Note: For each tuple of the *manages* relation, the manager-id is the manager's employee id.