CS 2541: In-Class Exercises 2. Relational Algebra and Calculus Queries.

- **lives**(person-name,street,city)
- **works**(person-name, company-name,salary)
- **located-in**(company-name,city)
- **manages**(person-name,manager-name)

For the above schema (the primary key for each relation is denoted by the underlined attribute), provide relational algebra expressions for the following queries:

Note: For notational convenience, I am using pname instead of person-name, cname instead of company-name, and mname instead of manager-name. In addition, instead of *lives*[*pname*] i am using the equivalent notation *lives.pname* in an SQL like syntax in some queries – you can use either.

1. Find all tuples in works of all persons who work for the City Bank company (which is a specific company in the database).

(a) $\sigma_{(cname='City\,Bank')}(works)$

(b) $\{p | p \in works \wedge (p[cname] =' City\,Bank'))\}$

2. Find the name of persons working at City Bank who earn more than $50,000.

(a) $\pi_{pname}(\sigma_{(cname='City\,Bank')\wedge(salary>50000)}(works))$

(b)$\{p \,|\, \exists w \in works((p[pname] = w[pname]) \wedge (w[cname] =' City\,Bank'))\}$

Since p only has attribute of pname in the query, its 'type' is a single attribute (person-name).

Notation options: The solutions for relational calculus use the notation tuplevariable[attribute] (for example $w[pname]$; you could also use the easier to read notation tuplevariable.attribute (for example $w.pname$.

3. Find the name and city of all persons who work for City Bank and earn more than 50,000. Similar to previous query, except we have to access the lives table to extract the city of the employee . Note the join condition in the query.

(a) $\pi_{lives.pname,lives.city}(\sigma_{((cname='City\,Bank')\wedge(salary>50000)\wedge(lives.pname=works.pname))})(lives \times works)$

We can use the equijoin operator *bowtie* instead of the cross product and σ operators and write the query as: $\pi_{lives.pname,lives.city}(\sigma_{((cname='City\,Bank')\wedge(salary>50000))})(lives \bowtie_{lives.pname=works.pname} works)$

(b) In the query below, the tuple p (the only free variable) refers only to attributes pname and city.

$$\begin{aligned}&\{p \,|\, \exists w \in works,\ \exists l \in lives\\ &\quad((w[cname] =' City\,Bank') \wedge (w[salary] > 50000) \wedge (w[pname] = l[pname])\\ &\quad\wedge(p[pname] = w[pname]) \wedge (p[city] = l[city]))\}\end{aligned}$$

This could also be written as (i.e., the quantifiers are 'nested')

$$\{p \,|\, \exists w \in works((p[pname] = w[pname]) \wedge (w[cname] =' City\,Bank') \wedge (w[salary] > 50000) \wedge \\ (\,\exists l \in lives(l[pname] = w[pname]) \wedge (p[city] = l[city])))$$

4. Find names of all persons who live in the same city as the company they work for. For this query we need to access the lives table to get city of the employee and the located-in table to get city of the company; plus the works table to associate employee with their company. The selection condition is then that the two cities are the same.

(a)

$$\pi_{lives.pname} \\ (\sigma_{((locatedin.cname=works.cname)\wedge(located-in.city=lives.city)\wedge(lives.pname=works.pname))} \\ (works \times lives \times locatedin))$$

Using the equijoin operator, we can write the above query as:

$$\pi_{lives.pname} \\ (\sigma_{(located-in.city=lives.city)} \\ ((works \bowtie_{(lives.pname=works.pname)} lives) \bowtie_{works.cname=locatedin.cname} locatedin))$$

(b)

$$\{p | \exists w \in works,\ \exists l \in lives\, \exists y \in locatedin \\ ((l.city = y.city) \wedge (w.cname = y.cname) \wedge (w.pname = l.pname) \wedge (p.pname = l.pname))\}$$

5. Find names of all persons who live in the same city and on the same street as their manager. This requires accessing lives table twice – once for finding city of employee and a second time for finding city of manager. Therefore we need two variables from lives with different names; one will refer to employee and the other will refer to the manager. So create a copy of lives, called mlives, which we use to find the address of the manager.

(a)

$$\pi_{lives.pname} \\ (\sigma_{((lives.city=mlives.city)\wedge(lives.street=mlives.street)\wedge(manages.pname=lives.pname)\wedge(mname=mlives.pname))} \\ (lives \times manages \times (\rho_{mlives}(lives))))$$

(b)

$$\{x \mid \exists y \in lives,\, \exists z \in lives\, \exists m \in manages$$
$$((z.city = y.city) \wedge (y.street = z.street) \wedge$$
$$(y.pname = m.pname) \wedge (z.pname = m.managername) \wedge (x.pname = y.pname))\}$$

6. Find names of all persons who do not work for City Bank. Can write this in multiple ways - one solution is to use set difference:

(a) $(\pi_{pname}(works)) - (\pi_{pname}(\sigma_{cname='City\, Bank'}(works)))$

(b) $\{x | \exists y \in works(x.pname = y.pname \wedge y.cname \neq' City\ Bank')\}$

7. Find the name of all persons who work for City Bank and live in DC. Similar to query 3, but select only with tuples where person city is DC.

(a) $\pi_{lives.pname}(\sigma_{((cname='City\, Bank') \wedge (lives.city='DC') \wedge (lives.pname=works.pname))})(lives \times works)$

(b)

$$\{p \mid \exists w \in works,\, \exists l \in lives$$
$$((w[cname] =' City\, Bank') \wedge (l[city] =' DC') \wedge (w[pname] = l[pname])$$
$$\wedge (p[pname] = w[pname]))\}$$