Ex4_LuisZüttel_GionRubitschung_D1P

March 18, 2024

1 Aufgabe 1

Drücken Sie die gegebenen Abfragen in der relationalen Algebra aus.

$employee(\underline{person\_name}, street, city)$

$works(\underline{person\_name}, company\_name, salary)$

$company(\underline{person\_name}, city)$

$manages(\underline{person\_name}, manager\_name)$

1. Finde die Namen und Wohnorte aller Angestellten, welche für FBC arbeiten.

   $employee\_works \leftarrow employee \bowtie works$

   $fbc\_workers \leftarrow \sigma_{works.company\_name='FBC'}(employee\_works)$

   $\pi_{person\_name,city}(fbc\_workers)$

2. Finde die Namen und Wohnorte mit Strasse aller Angestellten, welche für FBC arbeiten und die mehr als CHF 100'000.- verdienen

   $\pi_{person\_name,street,city}(\sigma_{salary>100000}(fbc\_workers))$

3. Finde die Namen aller Angestellten, die in der Stadt arbeiten in der sie auch wohnen.

   $\pi_{person\_name}(employee\_works \bowtie company)$

4. Finde die Namen aller Angestellten, die in derselben Stadt an derselben Strasse wohnen, wie ihr Manager.

   $manager \leftarrow \rho_{manager\_name,street,city}(employee)$

   $employee \bowtie manages \bowtie manager$

5. Finde die Firma mit den meisten Angestellten

   $company\_employees \leftarrow_{company\_name} \mathcal{G}_{count(person\_name) as employees}(works)$

   $most\_employees \leftarrow \mathcal{G}_{max(company\_name) as max\_employees}(company\_employees)$

   $\pi_{company\_name}(\sigma_{employees=max\_employees})(company\_employees \times most\_employees)$

6. Finde die Firma, welche die kleinste Lohnsumme bezahlt.

   $sum\_salaries \leftarrow_{company\_name} \mathcal{G}_{sum(salary) as sum\_salaries}(works)$

   $min\_salary \leftarrow \mathcal{G}_{min(sum\_salaries) as min\_salary}(sum\_salaries)$

$\pi_{company\_name,min\_salary}(\sigma_{sum\_salaries=min\_salary}(sum\_salaries \times min\_salary))$

7. Finde diejenigen Firmen, deren Angestelle im Durchschnitt mehr verdienen, als der Durchschnittslohn der FBC.

   $fbc\_average\_salary \leftarrow \mathcal{G}_{avg(salary)asfbc\_average\_salary}(\sigma_{company\_name='FBC'}(works))$

   $company\_average\_salaries \leftarrow_{company\_name} \mathcal{G}_{avg(salary)asaverage\_salary}(works)$

   $\pi_{company\_name}(\sigma_{average\_salary>fbc\_average\_salary})(company\_average\_salaries \times fbc\_average\_salary)$

8. Finde alle Firmen, die in jeder Stadt sind, in der auch die FBC ist.

   $fbc \leftarrow \sigma_{company\_name='FBC'}(company)$

   $cities \leftarrow \rho_{city}(\pi_{company.city}(\sigma_{company.city=fbc.city}(company \times fbc)))$

   $cities - (\rho_{city}(fbc))$

2 Aufgabe 2

Gegeben sind folgende Relationen $r(A, B, C, D)$ und $s(A, E)$:

A	B	C	D
"A"	1000	3	" "
"A"	700	Null	"agh"
"A"	Null	0	"abcdef"
"A"	1000	4	Null
"B"	Null	Null	"bdf"
"B"	1500	Null	"c"
Null	1000	8	" "
Null	700	12	Null

A	E
"B"	1
"C"	2
"C"	3

1. Evaluieren Sie für jede Zeile von r den Wert des Prädikats p mit $p = (B \cdot C < 5000 or D is null)$

   Berechnen Sie:

2. $\sigma_p(r)$

A	B	C	D
"A"	1000	3	" "
"A"	1000	4	Null
Null	700	12	Null

3. ${}_{A}\mathcal{G}_{avg(B),sum(C)}(r)$

A	avg(B)	sum(C)
“A”	900	7
“B”	1500	Null
Null	850	20

4. ${}_{A}\mathcal{G}_{avg(B)}(\pi_{A,B}(r))$

A	B
“A”	850
“B”	1500
Null	850

5. $r \bowtie s$

A	B	C	D	E
“B”	Null	Null	“bdf”	1
“B”	1500	Null	“c”	1

6. $r \quad s$

A	B	C	D	E
“B”	Null	Null	“bdf”	1
“B”	1500	Null	“c”	1
“C”	Null	Null	Null	2
“C”	Null	Null	Null	3

7. $r \quad s$

A	B	C	D	E
“A”	1000	3	“ ”	Null
“A”	700	Null	“agh”	Null
“A”	Null	0	“abcdef”	Null
“A”	1000	4	Null	Null
“B”	Null	Null	“bdf”	1
“B”	1500	Null	“c”	1
“C”	Null	Null	Null	2
“C”	Null	Null	Null	3
Null	1000	8	“ ”	Null
Null	700	12	Null	Null

3 Aufgabe 3 (Implementation Kartesisches Produkt und Natural Join)

Gegeben seien Relationen $r(A, B)$ und $s(B, C)$. Die Relationen liegen in Form von Listen von Tupeln vor. Tupel sind Listen mit fester Länge, in unserem Falle mit Länge zwei. Zusätzlich zu den üblichen Funktionen `head` und `tail` gibt es auf den Listen (und damit auch auf Tupeln) die Funktion `list cons(element a, list l)` die die Liste liefert, die durch Voranstellen von Element `a` an die Liste `l` entsteht. Die leere Liste heisst `nil`. Andere Funktionen auf Listen gibt es nicht.

[ ]:
```python
def head(lst: list):
    return lst[0]

def tail(lst: list):
    return lst[1:]

def cons(element, lst: list) -> list:
    return [element] + lst

r = [('a1', 'b1'), ('a2', 'b2'), ('a3', 'b3')]
s = [('b1', 'c1'), ('b2', 'c2'), ('b4', 'c4')]
```

1. Geben Sie einen Algorithmus an, um das kartesische Produkt der beiden Relationen zu berechnen.

[ ]:
```python
def cartesian_product(r: list, s: list) -> list:
    if not r:
        return []
    else:
        a = head(r)
        product_tuples = []
        for b in s:
            product_tuples = cons((a, b), product_tuples)
        return product_tuples + cartesian_product(tail(r), s)

cartesian_product(r, s)
```

[ ]:
```
[(('a1', 'b1'), ('b4', 'c4')),
 (('a1', 'b1'), ('b2', 'c2')),
 (('a1', 'b1'), ('b1', 'c1')),
 (('a2', 'b2'), ('b4', 'c4')),
 (('a2', 'b2'), ('b2', 'c2')),
 (('a2', 'b2'), ('b1', 'c1')),
 (('a3', 'b3'), ('b4', 'c4')),
 (('a3', 'b3'), ('b2', 'c2')),
 (('a3', 'b3'), ('b1', 'c1'))]
```

2. Geben Sie einen Algorithmus an, um den natural join der beiden Relationen zu berechnen.

[ ]:
```python
def natural_join(r: list, s : list) -> list:
    if not r:
        return []
    else:
        a, b = head(r)
        matching_tuples = []
        for b_, c in s:
            if b == b_:
                matching_tuples = cons((a, b, c), matching_tuples)
        return matching_tuples + natural_join(tail(r), s)

natural_join(r, s)
```

[ ]:
```
[('a1', 'b1', 'c1'), ('a2', 'b2', 'c2')]
```