Datenbanken Ubungsblatt 4¨

Aufgabe 1 (Abfragen mit Relationaler Algebra)

Dr¨ucken Sie die gegebenen Abfragen in der relationalen Algebra aus.

```
employee (personname, street, city)
works (personname, companyname, salary)
company (companyname, city)
manages (personname, managername)
```

1. Finde die Namen und Wohnorte aller Angestellten, welche f¨ur FBC arbeiten.
2. Finde die Namen und Wohnorte mit Strasse aller Angestellten, welche f¨ur FBC arbeiten und die mehr als CHF 100’000.- verdienen.
3. Finde die Namen aller Angestellten, die in der Stadt arbeiten in der sie auch wohnen.
4. Finde die Namen aller Angestellten, die in derselben Stadt an derselben Strasse wohnen, wie ihr Manager.
5. Finde die Firma mit den meisten Angestellten.
6. Finde die Firma, welche insgesamt die kleinste Lohnsumme bezahlt.
7. Finde diejenigen Firmen, deren Angestelle im Durchschnitt mehr verdienen, als der Durchschnittslohn der FBC.
8. Finde alle Firmen, die in jeder Stadt sind, in der auch die FBC ist.

solution

1.

```
select * from employee where works.companyname = "FBC" left join works on
employee.personname = works.personname
```

2.

```
select * from employee where works.companyname = "FBC" AND where
works.salary < 100000 left join works on employee.personname =
works.personname
```

3.

```
select * from employee where works.salary < 100000 left
join company on employee.city = company.city
```

4.

```
select * from employee as manager
join manages on employee.personname = managers.personname
```

5.

```
SELECT MAX(counts) FROM (SELECT COUNT(companyname) as counts
FROM works
GROUP BY companyname);
```

6.

```
SELECT MIN(sum) FROM (
    SELECT COUNT(salary) as lohn
    FROM works
    GROUP BY salary
);
```

7.

```
SELECT AVG(salary) FROM works  where companyname = "FBC"
```

8.

```
(SELECT * FROM company company1, company company2  where
company1.companyname = "FBC" and company1.city = company2.city and
company1.companyname IS NOT company2.companyname)
```

operatoren

|additional (helper) operator|equivalent basic relational algebra operation|description| |---|---|---| |table_a ∩ table_b|table_a – (table_a – table_b)|intersection / schnittmenge| |table_a ⋈ table_b|nothing|Natural join (⋈)/ takes only rows where column names and values are the same and merges them into one row on the result| |table_a ⟕ table_b |-|left outer join| |table_a ⟖ table_b|-|right outer join| |table_a ⟗ table_b|-|full outer join| |⊓ID, name, dept_name, salary/12 (table_human) |asdf|Generalized Projection/ like normal projection but functions with arithmetic operations and other column names can be used as the selected column names, see "salary/12" | |sum(column_name)(table_name)|-|Aggregation / a predefined function , which takes multiple records as input and returns single result (avg, min, max, sum, count)

Aufgabe 2 (NULL-Werte)

unknown operations

what is 'unknown'

if 'column_salary' value for this record is null, the result of the operation 'unknown' "column_salary > 50000” -> "null > 50000” => unknown

operation	result
unknown or true	true
unknown or false	unknown
unknown or unknown	unknown
true and unknown	unknown
false and unknown	false
unknown and unknown	unknown
not unknown	unknown
not NULL	NULL
NULL or false	NULL
NULL or true	true
NULL or NULL	NULL
NULL and false	false
NULL and true	NULL
NULL and NULL	NULL

Gegeben sind folgende Relationenr(A,B,C,D) unds(A,E):

truthtable / wahrheitstabelle

table/schema r(A,B,C,D)

A	B	C	D	B ·C < 5000	D is null	p = (B ·C < 5000 or D is null)
”A”	1000	3	””	true	false	true
”A”	700	Null	”agh”	unknown	false	unknown
”A”	Null	0	”abcdf”	unknown	false	unknown
”A”	1000	4	Null	true	true	true
”B”	Null	Null	”bdf”	unknown	false	unknown

A	B	C	D	B ·C < 5000	D is null	p = (B ·C < 5000 or D is null)
”B”	1500	Null	”c”	unknown	false	unknwon
Null	1000	8	””	false	false	false
Null	700	12	Null	false	true	true

table/schema s(A,E):

A	E
”B”	1
”C”	2
”C”	3

1.

Evaluieren Sie f ̈ur jede Zeile vonrden Wert des Pr ̈adikatspmit

solutions

1. see ###table/schema r(A,B,C,D)

2. σp(r) => select * from r where p => select * from r where p = true

result:

A	B	C	D	B ·C < 5000	D is null	p = (B ·C < 5000 or D is null)
”A”	1000	3	””	true	false	true
”A”	1000	4	Null	true	true	true
Null	700	12	Null	false	true	true

3. AGavg(B),sum(C)(r) => select avg(B), avg(c) from r

result:

|”A”|1000|3|””|true | false| true | |”A”|700|Null|”agh”|unknown| false| unknown | |”A”|Null|0|”abcdf”|unknown | false| unknown| |”A”|1000|4|Null|true | true | true | |”B”|Null|Null|”bdf”|unknown | false| unknown | |”B”|1500|Null|”c”|unknown | false| unknwon | |Null|1000|8|””|false | false | false | |Null|700|12|Null|false | true | true |

average of column 'B' = (1000 +700 +1000 +1500 +1000 +700) / 6 = 983,333333333

sum of column 'C' = (3+0+4+8+12) = 27

result:

B	C

B	C
983,333333333	27

4. AGavg(B)(πA,B(r))

r1 = πA,B(r) => select A, B from r

result:

A	B
”A”	1000
”A”	700
”A”	Null
”A”	1000
”B”	Null
”B”	1500
Null	1000
Null	700

AGavg(B)(r1)

average of column 'B' = (1000 +700 +1000 +1500 +1000 +700) / 6 = 983,333333333

result:

B
983,333333333

5. r ⋈ s

when no join "ON" condition/keyword is set, a corss join will be performed, which is a multiplication of every row of both tables with each other

https://stackoverflow.com/questions/16470942/how-to-use-mysql-join-without-on-condition/16471286

r crossjoin s

r.A	r.B	r.C	r.D	s.A	s.E
”A”	1000	3	””	”B”	1
”A”	700	Null	”agh”	”B”	1
”A”	Null	0	”abcdf”	”B”	1

r.A	r.B	r.C	r.D	s.A	s.E
”A”	1000	4	Null	”B”	1
”B”	Null	Null	”bdf”	”B”	1
”B”	1500	Null	”c”	”B”	1
Null	1000	8	””	”B”	1
Null	700	12	Null	”B”	1
”A”	1000	3	””	”C”	2
”A”	700	Null	”agh”	”C”	2
”A”	Null	0	”abcdf”	”C”	2
”A”	1000	4	Null	”C”	2
”B”	Null	Null	”bdf”	”C”	2
”B”	1500	Null	”c”	”C”	2
Null	1000	8	””	”C”	2
Null	700	12	Null	”C”	2
”A”	1000	3	””	”C”	3
”A”	700	Null	”agh”	”C”	3
”A”	Null	0	”abcdf”	”C”	3
”A”	1000	4	Null	”C”	3
”B”	Null	Null	”bdf”	”C”	3
”B”	1500	Null	”c”	”C”	3
Null	1000	8	””	”C”	3
Null	700	12	Null	”C”	3

result:

|A|B|C|D| B ·C < 5000 | D is null| p = (B ·C < 5000 or D is null) | E | |---|---|---|---|---|---|---|---| |”B”|Null|Null|”bdf”|unknown | false| unknown | 1 | |”B”|1500|Null|”c”|unknown | false| unknwon | 2 |

6. r ⋉ s

result:

A	B	C	D	B ·C < 5000	D is null	p = (B ·C < 5000 or D is null)	E
”A”	1000	3	””	true	false	true	null
”A”	700	Null	”agh”	unknown	false	unknown	null

A	B	C	D	B ·C < 5000	D is null	p = (B ·C < 5000 or D is null)	E
”A”	Null	0	”abcdf”	unknown	false	unknown	null
”A”	1000	4	Null	true	true	true	null
Null	1000	8	””	false	false	false	null
Null	700	12	Null	false	true	true	null

7. r ⋉ s

result:

A	B	C	D	B ·C < 5000	D is null	p = (B ·C < 5000 or D is null)	E
”A”	1000	3	””	true	false	true	null
”A”	700	Null	”agh”	unknown	false	unknown	null
”A”	Null	0	”abcdf”	unknown	false	unknown	null
”A”	1000	4	Null	true	true	true	null
”B”	Null	Null	”bdf”	unknown	false	unknown	1
”B”	1500	Null	”c”	unknown	false	unknwon	2
Null	1000	8	””	false	false	false	null
Null	700	12	Null	false	true	true	null

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 L¨ange, in unserem Falle mit L¨ange zwei. Zus¨atzlich zu den¨ublichen Funktionenheadundtailgibt es auf Listen (und damit auch auf Tupeln) die Funktionlist cons(element a, list l)die die Liste liefert, die durch Voranstellen von Elementaan die Listelentsteht. Die leere Liste heisstnil. Andere Funktionen auf Listen gibt es nicht.

1. Geben Sie einen Algorithmus an, um das kartesische Produkt der beiden Relatio- nen zu berechnen.
2. Geben Sie einen Algorithmus an, um den natural join der beiden Relationen zu berechnen.

2

```
f_a_cartesian_product(a_a, a_b):
    # a_a = [[2,3], [3,2], [22,3], ...]
    # a_b = [[33,12], [54,23], [2,1], ...]
    # a_product = [[2,3,33,12],[2,3,54,23],[2,3,2,1],[3,2,33,12]...]
    a_product = []
```

```
    for a_a_a in a_a:
        for a_a_b in a_b:
            a_product.append(a_a_a + a_a_b)

    return a_product


f_a_natural_join(a_a, a_b):
    # a_a = [[2,3], [3,2], ...]
    # a_b = [[2,5], [54,23], ...]
    # a_natrual_join = [[2,3,2,5]]
    a_natrual_join = []
    for a_a_a in a_a:
        for a_a_b in a_b:
            if(a_a_a[0] == a_a_b[0] or a_a_a[1] == a_a_b[1])
            a_natrual_join.append(a_a_a + a_a_b)

    return a_natrual_joing
```