Jacob Alspaw

Introduction to Databases
EECS 341 – Bebek
Assignment 3
15 October 2016

1a. $\Pi_{name,\ address}\ (\ \sigma_{location\ =\ \text{“Cleveland”}}\ (\ person \bowtie accident\ )\ )$

1b. $\sigma_{model\ =\ \text{“Fiat”}\ \wedge\ now().year\ -\ year\ >\ 3}\ (\ person \bowtie car\ )\ )$

1c. $\Pi_{date}\ (\ \sigma_{damage\text{-}amount\ >\ 20{,}000}\ (\ person \bowtie accident\ )\ )$

1d. $\Pi_{name}\ (\ person \bowtie accident\ )$

1e. $\Pi_{name}\ (\ \sigma_{numAccidents\ >\ 3}\ (\ _{name}\mathcal{G}_{count(report\text{-}number)\ \text{as}\ numAccidents}\ (\ person \bowtie accident\ )\ )\ )$

1f. $\Pi_{name}\ (\ \Pi_{name,\ max(damage\text{-}amount)}\ (\ _{name}\mathcal{G}_{sum(damage\text{-}amount)}\ (\ person \bowtie accident\ )\ )\ )$

1g. $\Pi_{report\text{-}number,\ location,\ date}\ (\ \sigma_{model\ =\ \text{“Fiat”}}\ (\ accident \bowtie car\ )\ )$

1h. $person\ -\ \Pi_{driver\text{-}id,\ address,\ name}\ (\ \sigma_{location\ =\ \text{“Cleveland”}}\ (\ person \bowtie accident\ )\ )$

For 1g, only accidents are linked to cities, so there is no guarantee that any query links a car to a city. This is because a car does not need to get in an accident to be considered as driving in a city. Therefore, I inferred the question was asking for all accidents where a Fiat was involved.

2a. Relational Algebra:

$\Pi_{\text{avg (A)}}\ (\ \sigma_{(count(A)\ /\ 2)\ -\ 1.5\ <\ rank\ <\ (count(A)\ /\ 2)\ +\ 0.5}\ (\text{partb})\ )$

Where “a < x < b” == “a < x ∧ x < b”. Partb is the result of part b.

SQL:

```
CASE COUNT(A) % 2
WHEN 1 THEN
       A - (SELECT TOP COUNT(A)/2 FROM R ORDER BY A DESC
            UNION
            SELECT TOP COUNT(A)/2 FROM R ORDER BY A ASC)
WHEN 0 THEN
       AVG(A - (SELECT TOP (COUNT(A)/2)-1 FROM R ORDER BY A DESC
                UNION
                SELECT TOP (COUNT(A)/2)-1 FROM R ORDER BY A ASC))
```

2b. Relational Algebra:

$\Pi_{r2.A,}\ \rho_{rank}\ (\ _{r2}\mathcal{G}_{count}\ (\ \sigma_{r1.A\ <\ r2.A}\ (\rho_{r1}\ (\ R\ )\ )\ )\ )\ (\ \rho_{r2}\ (\ R\ )\ )$

SQL:

```
SELECT r1.A, COUNT(r2.A) FROM R r1, R r2
GROUP BY r1.A HAVING r2.A < r1.A
```

3.

S	
B	C
0	1
1	2
2	3

R	
A	B
1	0
2	1
3	2

Query A result:

3 tuples → (1,1) ; (2,1) ; (3,1)

Query B result:

1 tuple → (1,1)