Implementation of Databases Exercise 2

Ilya Kulikov 351063, Alina Shigabutdinova 351017, Oleg Chernikov 351016

November 16, 2015

Exercise 2.1

Notation: d = drinker, ba = bar, be = beer.

1. TRC:

$$\{l.d \mid l \in \text{likes} \wedge \exists s \in \text{serves}\ (s.be = l.be \wedge \exists f \in \text{frequents}\ (f.ba = s.ba \wedge f.d = l.d))\}$$

   DRC:

$$\begin{aligned}\{drinker \mid \exists bar, beer\ & \\ & (drinker, bar) \in \text{frequents} \wedge \\ & (bar, beer) \in \text{serves} \wedge \\ & (drinker, beer) \in \text{likes}\ \}\end{aligned}$$

2. TRC:

$$\begin{gathered}\{f.d \mid f \in \text{frequents} \wedge \exists f' \in \text{frequents}\ \wedge \exists l \in \text{likes} \wedge \\ l.be = \text{'Bitburger'} \wedge l.d = f'.d \wedge f'.d \neq f.d \wedge f'.ba = f.ba\}\end{gathered}$$

   DRC:

$$\begin{aligned}\{drinker \mid \exists d', bar\ & \\ & (drinker, bar) \in \text{frequents} \wedge \\ & (d', bar) \in \text{frequents} \wedge \\ & (d', \text{'Bitburger'}) \in \text{likes}\ \}\end{aligned}$$

3. TRC:

$$\{f.d \mid f \in \text{frequents} \wedge \forall s \in \text{serves}(s.ba = f.ba \wedge \forall l \in \text{likes} \wedge l.d = f.d \wedge s.be \neq l.be))\}$$

   DRC:

$$\begin{aligned}\{drinker \mid \exists bar\ \forall beer\ & \\ & (drinker, bar) \in \text{frequents} \wedge \\ & (bar, beer) \in \text{serves} \wedge \\ & (drinker, beer) \notin \text{likes}\ \}\end{aligned}$$

Exercise 2.2

1. Relational Algebra is used to internally represent queries and query evaluation plans because of several reasons: first of all, we can represent complicated queries by composing relational algebra operators with each other under some rules. Secondly, Relational Algebra is closed algebra under the finite relation domain, so we have definite result always. Finally, because of Codd's theorem, each RA expression could be represented using Relational calculus.

2. A query language is relational complete, iff one can use this language to describe any query from Relational Algebra or even more. SQL is relational complete, as *SELECT* corresponds to *Projection*, *WHERE* to *Selection* and *FROM* can perform *Join* or *Cartesian Product*. *Rename* operator can also be used in SQL with key word *AS*.

3. The intersection RA operator returns the same rows from two relations with equivalent schemas. From the set theory we know, that intersection could be defined using difference (but in this case we lose commutative property of original intersection, so we are not sure about omittability):

$$R \bowtie S = R \setminus (R \setminus S) \tag{1}$$

4. The TRC as the name suggests operates with tuples, while DRC uses attributes and values.

Exercise 2.3

1. For the pass 0 we produce:

$$n = \frac{N}{B} = \frac{25000}{8} = 3125 \tag{2}$$

   Pass 1: 447 runs
   Pass 2: 75 runs
   Pass 3: 15 runs
   Pass 4: 4 runs
   Pass 5: 2 runs
   Pass 6: 1 run

   So overall we produce 3669 runs

2. The formula is $2 * N * \lceil 1 + log_{B-1} \lceil N/B \rceil \rceil$, where $\lceil 1 + log_{B-1} \lceil N/B \rceil \rceil$ is corresponding to the number of passes. With $N = 25000$ and $B = 8$ we get 6 as an answer.

3. The number of passes is $\lceil 1 + log_{B-1} \lceil N/B \rceil \rceil = 2$, then $N/B = B - 1$. This boils down to a quadratic equation, where one of the roots is negative and the other one is $B = \frac{1}{2}(1 + \sqrt{1 + 4N})$. In case $N = 25000$ and after ceiling we get $B = 159$

4. Two-way merge sort requires $1 + \lceil log_2(N) \rceil$ passes, therefore the answer is 16.
   In the first pass we sort each page, so we require 25000 runs. For consequent passes we require 12500, 6250, 3125, ... runs. Overall, to sort the data entirely we have to use 50006 runs.

Exercise 2.4

1.

2.

3.

4.

Exercise 2.5

1. If each node is 70% full, there are 14 pointers at each node. Then, $H = \lceil \log_{14}(50000) \rceil = 5$.
2. a.

$$C = D * (1 + 3) = 4 \tag{3}$$

b.

$$C = 50000/20 = 2500 \tag{4}$$

c.

$$C = 4 + 2 = 6 \tag{5}$$

d.

$$C = 50000 * 0.3 + 5 = 15005 \tag{6}$$