CPTS 415 – ASSIGNMENT 2

CHARLES LAMBERT NGUYEN – 011606177

1. Relation Algebra
   Q1: $\pi_{theater}\ \sigma_{title=Zootopia}Schedule$
   Q2: $\pi_{(Location.Theater,Location.Address)}\sigma_{Movies.Director=\text{Steven Spielberg}}(Movies \times Location)$
   Q3: $\pi_{(Location.Adress,Location.PhoneNumber)}\sigma_{Location.Theater=\text{Le Champo}}$
   Q4: $\pi_{(M1.Actor,M2.Actor)}\sigma_{(M1.Title=M2.Title\ \&\ M1.Actor\neq M2.Actor)}(\rho_{M1}Movies \times \rho_{M2}Movies)$

2. Relation $\{S: 20'000t, 10t\ per\ block \mid 2'000b\}$
   Relation $\{R: 100'000t, 10t\ per\ block \mid 10'000b\}$
   Assumption: neither is indexed.
   Constraint: 52 blocks of memory.
   a. Nested-Join: using two nested for loops (technically 4 but more on that later).
      i. For each block in S, $b_i < b_S$: (1)
         1. For each block in R, $b_j < b_R$: (2)
            a. For each tuple in S, $t_m < t_S$: (3)
               i. For each tuple in R, $t_n < t_R$: (4)
                  1. Test if (t_m, t_m) share condition $\theta_{S.t_S=R.t_R}$
               ii. End for
            b. End for
         2. End for
      ii. End for

      The cost of the two innermost loops is (10×10) tuples per $(1b_i + 1b_j) = 2b$ per each call on the second for-loop. Since memory can hold at most 52 blocks, we can abstract this cost away and consider only the two outermost loops on b_i and b_j.

      The cost, then, is:
      - Worst case: $(b_S + b_S \cdot b_R)$ for transfers, $(b_S + b_S \cdot 1)$ for seeks.
      - Best case: $(b_S + b_R)$ for transfers, $(1t + 1t)$ for seeks.
   b. Sort-Merge: using divide and conquer.
      Create sorted chunks that can fit in memory (52 blocks).
      - Recursively call the following till the end of the relation:
         a. Read M blocks of relation into memory
         b. Sort the in-memory blocks as sorted chunk
         c. Write sorted chunk to disk
      - Given that the input is much larger than the memory, we need several merge passes. Therefore, we read b_b blocks per chunk so that we can merge a group of $(\frac{M}{b_b} - 1)$ chunks per pass.
      - A pass reduces the number of chunks by a factor of $(\frac{M}{b_b} - 1)$, and creates chunks longer by the same factor.
      - Repeated passes are performed till all runs have been merged into one.

- Total number of merge passes required: $ceil(\log_{floor\left(\frac{M}{b_b}-1\right)}\left(\frac{b_S}{M}\right))$.
- $b_b < 52$ determines the trade-off between number of passes, and disk I/O operation time per pass.
- The cost is then:
    - Number of transfers: $2b_S \cdot (\log_{(\frac{M}{b_b}-1)}(\frac{b_S}{M}))$
    - Number of seeks: $\frac{2b_S}{b_b} \cdot (\log_{(\frac{M}{b_b}-1)}(\frac{b_S}{M}))$

c. Hash-Join: applicable for equijoins and natural joins.

- Perfect hashing breaks down S, R into 40 partitions $S\colon \{40p, \frac{50b}{p}\}$ and $R\colon \{40p, \frac{250b}{p}\}$
- Total cost, where $b_b = 10$:
    - Block transfers: $10 \cdot (2'000 + 10'000) = 120'000$
    - Seeks: $2 \cdot \left(\frac{2'000}{10} + \frac{10'000}{10}\right) = 2'400$

3. See separate XML files.
    - For DTD, keys are declared as attributes. Primary key is defined as $\#REQUIRED$. Foreign key is defined as $\#IMPLIED$.
    - For XML Schema, primary keys are identified by the $name$ attribute. The id attributes can serve as foreign keys.
4. See separate JSON files.