Relational DB Design Exercises

1. Given a relational R and a set of functional dependencies F, what is the time complexity to test if R is in BCNF? If R is decomposed into $R_1, R_2, \ldots, R_n$, what is the time complexity to test if R_i is in BCNF, $1 \leq i \leq n$?

   To check whether R is in BCNF with respect to F is easy: Following the definition of BCNF, we need to check, for any nontrivial functional dependency $\alpha \rightarrow \beta$ in F, whether $\alpha \rightarrow R$, i.e., whether α is superkey for R. To do so, we need to compute α^+ and to check whether it contains R. Assume |R| = n and F has m functional dependencies. We need in the worst case to computer one attribute closure for every functional dependency in F. Since the time complexity of computing an attribute closure is $O(mn^2)$, so the total time needed is $O(m^2n^2)$.

   To check whether R_i is in BCNF is hard. One way to do this is as follows. First, compute the closure F^+, which takes $O(4^n)$ time. Second, for each nontrivial functional dependency $\alpha \rightarrow \beta$ in the restriction of $F^+$to R_i, check whether α is superkey for R_i, which takes $O(m^2n^2)$ for $\alpha \rightarrow \beta$. Hence, the total time is $O(m^2n^24^n)$.

2. Given a relation R and a set of functional dependencies F, what is the time complexity to test if R is in 3NF?

   Unlike the problem of testing whether R is BCNF, the problem of testing R is in 3NF is hard. For any nontrivial functional dependency $\alpha \rightarrow \beta$ where α is not a superkey for R, the third condition of the 3NF definition asks for each attribute *A* in $\beta - \alpha$ whether A is contained in a candidate key for *R.* This implies that to facilitate the testing for the third conditions, we need to find candidate keys for R, which is NP-hard. Hence, the time complexity of testing whether R is in 3NF is NP-hard.

3. What is time complexity of the BCNF decomposition algorithm?

   Although testing whether any given relation R with respect to a set of functional dependencies F is in BCNF is easy (i.e., this can be done in polynomial time), yet the problem of decomposing R into BCNF (if it is not in BCNF) is hard. To do so, we have to compute the closure of F, which take time in the magnitude of $O(4^n)$, where $|R| = n$.

4. What is the time complexity of the 3NF decomposition algorithm?

   Although testing whether any given relation R with respect to a set of functional dependencies F is in 3NF is hard (i.e., this is NP-hard), yet the problem of decomposing R into 3NF (if it is not in BCNF) is easy.

   Let $|R| = n, |F| = m$.

   First, we have to compute the canonical cover F_c of F, which requires checking for every functional dependency in F whether it has extraneous attributes, and the latter can be done by

calling attribute closure algorithm with $O(mn^2)$ time complexity. Thus, finding F_c takes $O(m^2n^3)$ time.

Second, we need to find one candidate key, which is done by starting with R and repeatedly checking whether we can remove attributes to the make the rest still a key. Again, the latter is done with the attribute closure algorithm of $O(mn^2)$ time complexity. So, the total time for this part is $O(mn^3)$.

Third, for each functional dependency $\alpha \rightarrow \beta$ in F_c, we add a relation schema $R_i = \alpha\beta$. Finally, we can check whether each smaller relation schema R_i contains a candidate for R or not. This can be like the way in the first step. If none of smaller relation schemas contain a candidate key for R, then add the candidate key that is obtained from the first step as an additional relation schema. The total time for this step is $O(m^2n^2)$.

Putting all together, we need time $O(m^2n^3) + O(m^2n^3) + O(m^2n^2) = O(m^2n^3)$ time in total.

5. Let R be a relation. Given any subset A of R, what is the time complexity to find the closure of A?

   Think about. Easy 😊

6. Let R be a relation and F be set of functional dependencies. Design an algorithm to find one (just one, whatever it is) candidate key for R. Explain the time complexity of your algorithm.
   In the actual implementation of 3NF decomposition, we want to include a candidate key for R in the decomposition. Why do we need to do this?

   Easy. 😊

7. If our design goal is to guarantee lossless-loin decomposition and functional dependency preservation, then which normal form shall we decompose a relation into?
   If our design goal is to guarantee lossless-loin decomposition and no information redundancy, then which normal form shall we decompose a relation into?

   Easy. 😊