COMP 430 Assignment 5

1. (35 points for COMP 430, 40 points for COMP 533) Implement the Assignment 4's swimming ERD as schemas. For the sake of consistency, use the sample solution ERD, rather than your own. Explain any choices that you have to make.

   The first choice is the representation of the super- and sub-class relationship between Org and Univ. Implementing only Org seems the simplest sufficient choice, adding a field to indicate whether the Org is a Univ.

   - Org(<u>id</u>, name, is_univ)
   - Meet(<u>name</u>, start_date, num_days, org_id) – org_id references Org.id
   - Participant(<u>id</u>, gender, org_id) – org_id references Org.id
   - Event(<u>id</u>, gender, stroke, distance) – stroke references Stroke.stroke, distance references Distance.distance
   - Stroke(<u>stroke</u>)
   - Distance(<u>distance</u>)
   - Heat(<u>id</u>, <u>event_id</u>, <u>meet_name</u>) – event_id references Event.id, meet_name references Meet.name
   - Swim(<u>heat_id</u>, <u>event_id</u>, <u>meet_name</u>, <u>participant_id</u>, time) – heat_id references Heat.id, event_id references Event.id, meet_name references Meet.name, participant_id references Participant.id

   The complexity of the primary key for Heat, and thus also for Swim, suggests that introducing a new field as a simpler single-field primary key would be helpful. This solution does not do that because it doesn't strictly follow the ERD implementation rules.

   (COMP 533 changes)

   - Leg(<u>leg</u>)
   - Event(<u>id</u>, gender, stroke, distance, leg) – stroke references Stroke.stroke, distance references Distance.distance, leg references Leg.leg
   - Swim(<u>heat_id</u>, <u>event_id</u>, <u>meet_name</u>, <u>participant_id</u>, leg, time) – heat_id references Heat.id, event_id references Event.id, meet_name references Meet.name, participant_id references Participant.id, leg references Leg.leg

2. (5 points – COMP 533 only) Identify which of the schemas in Problem 1 do not satisfy BCNF. Assume you have the following functional dependences. For each entity set, its key attributes functionally determines each of its non-key attributes. For each relation, the set of the key attributes from each of the related entity sets functional determines each of the relation's attributes.

3. (5 points) Using only Armstrong's Axioms and the set of functional dependences {AB $\rightarrow$ C, A $\rightarrow$ BE, C $\rightarrow$ D}, give a complete derivation of the functional dependency A $\rightarrow$ D.

Here is one possible derivation. Note that you do not need to present axioms as their own steps, as done here.

a. $A \rightarrow BE$, axiom
b. $A \rightarrow B$, decomposition from (a)
c. $A \rightarrow AB$, augmentation from (b) and observing AA=A
d. $AB \rightarrow C$, axiom
e. $A \rightarrow C$, transitivity from (c) and (d)
f. $C \rightarrow D$, axiom
g. $A \rightarrow D$, transitivity from (e) and (f)

4. (5 points – COMP 533 only) Assume we have a relation R(A,B,C). Create an SQL query that expresses what it means for functional dependency $B \rightarrow C$ to hold on R. It should be trivial to determine from the query's output whether or not the functional dependency holds. E.g., your query might return a table with just True/False, or it might return a non-empty/empty table.

5. (5 points – COMP 533 only) In class, we saw a quadratic algorithm to compute the closure of a set of attributes. The version here has been optimized from the original by removing each functional dependence from consideration once it has been applied. Briefly explain why this doesn't change the algorithm's correctness. Give and explain this version's asymptotic running time.

Let F be the set of functional dependences.
```
Closure(A) =
  Result = A
  Remaining = F
  Repeat until Result doesn't change:
    For each FD B → C in Remaining:
      if B ⊆ Result,
      then
        Add C to Result
        Remove B → C from Remaining
  Return Result
```

Once we use a FD, that FD's right-hand-side is added to Result. Nothing is ever removed from Result, so using that FD again would not change Result. So, it is safe to remove the FD after its use. Since that is the only change to the algorithm, its correctness is unaffected.

Each iteration uses and removes at least one FD. In the worst case, each outer iteration removes exactly one FD. The basic cost of each outer iteration is the number of remaining FDs, for a total cost of n + n-1 + n-2 + … + 1, which is still $O(n^2)$.

6. (40 points – 5 points for each relation/FD and part combination) For each of the following relations and sets of functional dependencies, do the following parts. While you can automate

substeps like calculating closures, you must show enough of your work that the reasoning and correctness clearly follows. Let the in-class activities 07-keys and 07-bcnf guide you in terms of how much you can automate.

a. R(A,B,C,D), AB $\rightarrow$ C, C $\rightarrow$ D, D $\rightarrow$ A
b. R(A,B,C,D), B $\rightarrow$ C, B $\rightarrow$ D

i. Show what the keys are.
ii. Show which FDs violate 3NF.
iii. Show which FDs violate BCNF.
iv. Decompose the relation into a collection of relations that are in BCNF.