CSC343 Worksheet 15

July 13, 2020

1. **Exercise 4.7.1:** Draw a UML diagram for the problem of Exercise 4.1.1.
2. **Exercise 4.7.2:** Modify your diagram from Exercise 4.7.1 in accordance with the requirements of Exercise 4.1.2.
3. **Exercise 4.7.3:** Repeat Exercise 4.1.3 using UML.
4. **Exercise 4.7.4:** Repeat Exercise 4.1.6 using UML.
5. **Exercise 4.7.5:** Repeat Exercise 4.1. 7 using UML. Are your subclasses disjoint or overlapping? Are they complete or partial?
6. **Exercise 4.7.6:** Repeat Exercise 4.1.9 using UML.
7. **Exercise 4.7.7:** Convert the E/R diagram of Fig. 4.30 to a UML diagram.
8. **Exercise 4.7.8:** How would you represent the 3-way relationship of Contracts among movies, stars, and studios (see Fig. 4.4) in UML?
9. **Exercise 4.7.9:** Repeat Exercise 4.2.5 using UML.
10. **Exercise 4.8.1:** Convert the UML diagram of Fig. 4.43 to relations.
11. **Exercise 4.8.2:** Convert the following UML diagrams to relations:

    a) Figure 4.37.

    b) Figure 4.40.

    c) Your solution to Exercise 4.7.1.

    d) Your solution to Exercise 4.7.3.

    e) Your solution to Exercise 4.7.4.

    f) Your solution to Exercise 4.7.6.

12. **Exercise 4.8.3:** How many relations do we create, using the object-oriented approach, if we have a three-level hierarchy with three subclasses of each class at the first and second levels, and that hierarchy is:

a) Disjoint and complete at each level.

b) Disjoint but not complete at each level.

c) Neither disjoint nor complete.

13. **Exercise 4.9.1:** In Exercise 4.1.1 was the informal description of a bank database. Render this design in ODL, including keys as appropriate.

14. **Exercise 4.9.2:** Modify your design of Exercise 4.9.1 in the ways enumerated in Exercise 4.1.2. Describe the changes; do not write a complete, new schema.

15. **Exercise 4.9.3:** Render the teams-players-fans database of Exercise 4.1.3 in ODL, including keys, as appropriate. Why does the complication about sets of team colors, which was mentioned in the original exercise, not present a problem in ODL?

16. **Exercise 4.9.4:** Suppose we wish to keep a genealogy. We shall have one class, Person. The information we wish to record about persons includes their name (an attribute) and the following relationships: mother, father, and children. Give an ODL design for the Person class. Be sure to indicate the inverses of the relationships that, like mother, father, and children, are also relationships from Person to itself. Is the inverse of the mother relationship the children relationship? Why or why not? Describe each of the relationships and their inverses as sets of pairs.

17. **Exercise 4.9.5:** Let us add to the design of Exercise 4.9.4 the attribute education. The value of this attribute is intended to be a collection of the degrees obtained by each person, including the name of the degree (e.g., B.S.), the school, and the date. This collection of structs could be a set, bag, list, or array. Describe the consequences of each of these four choices. What information could be gained or lost by making each choice? Is the information lost likely to be important in practice?

18. **Exercise 4.9.6:** In Exercise 4.4.4 we saw two examples of situations where weak entity sets were essential. Render these databases in ODL, including declarations for suitable keys.

19. **Exercise 4.9.7:** Give an ODL design for the registrar's database described in Exercise 4.1.9.

20. Exercise 4.10.1: Convert your ODL designs from the following exercises to relational database schemas.

    a) Exercise 4.9.1.

    b) Exercise 4.9.2 (include all four of the modifications specified by that exercise).

    c) Exercise 4.9.3.

    d) Exercise 4.9.4.

    e) Exercise 4.9.5.

21. **Exercise 4.10.2:** Consider an attribute of type Dictionary with key and range types both structs of primitive types. Show how to convert a class with an attribute of this type to a relation.

22. **Exercise 4.10.3:** We mentioned that when attributes are of a type more complex than a collection of structs, it becomes tricky to convert them to relations; in particular, it becomes necessary to create some intermediate concepts and relations for them. The following sequence of questions will examine increasingly more complex types and how to represent them as relations.

    a) A card can be represented as a struct with fields rank (2. 3, ... , 10, Jack, Queen, King, and Ace) and suit (Clubs, Diamonds, Hearts, and Spades). Give a suitable definition of a structured type Card. This definition should be independent of any class declarations but available to them all.

    b) A hand is a set of cards. The number of cards may vary. Give a declaration of a class Hand whose objects are hands. That is, this class declaration has an attribute theHand, whose type is a hand.

    c) Convert your class declaration Hand from (b) to a relation schema.

    d) A poker hand is a set of five cards. Repeat (b) and (c) for poker hands.

    e) A deal is a set of pairs, each pair consisting of the name of a player and a hand for that player. Declare a class Deal, whose objects are deals. That is, this class declaration has an attribute theDeal, whose type is a deal.

    f) Repeat (e), but restrict hands of a deal to be hands of exactly five cards.

    g) Repeat (e), using a dictionary for a deal. You may assume the names of players in a deal are unique.

    h) Suppose we defined deals to be sets of sets of cards, with no player associated with each hand (set of cards). It is proposed that we represent such deals by a relation schema Deals (dealiD, card), meaning that the card was a member of one of the hands in the deal with the given ID. What, if anything, is wrong with this representation? How would you fix the problem?