Ben Juarez

CS 121 Final

Problem 2: Functional Dependencies

Given R(A, B, C, G, H, I) & F = {A➜B, BI➜C, AG➜H, G➜BH, A➜G)

a) Find a candidate key for R with proper demonstration:

- Let us test the attribute set AI. We will first show that it is a superkey by computation of its attribute-set closure. We will then show that it is a candidate key through the computation of the attribute-set closures of all the proper subsets of AI.
    - Start with (AI)+ = AI
        - A➜B
            - (AI)+ = ABI
        - A➜G
            - (AI)+ = ABGI
        - AG➜H
            - (AI)+ = ABGHI
        - BI➜C
            - (AI)+ = ABCGHI
- Therefore, we have shown that AI is a superkey since R ⊆ (AI)+
    - Start with (A)+ = A
        - A➜B
            - (A)+ = AB
        - A➜G
            - (A)+ = ABG
        - AG➜H
            - (A)+ = ABGH
    - Start with (I)+ = I
            - (I)+ = I
- Therefore, we have now clearly shown that AI is a candidate key since (A)+ and (I)+ are not R

---

b) Compute a canonical cover F_c from F:

- We shall compute F_c after each step.
    - Union: If A➜B and A➜G, then A➜BG
        - F_c = {A➜BG, BI➜C, AG➜H, G➜BH)
    - Decomposition: If G➜BH, then G➜B and G➜H
        - Therefore, by definition, G in AG➜H is extraneous
        - F_c = {A➜BG, BI➜C, A➜H, G➜BH}
    - Decomposition: If A➜BG, then A➜B and A➜G

    - Transitivity: If A➜G and G➜BH, then A➜BH
    - Decomposition: If A➜BH, then A➜B and A➜H
        - Therefore, by definition, B in A➜BG is extraneous
        - F_c = {A➜G, BI➜C, A➜H, G➜BH}
    - (By same logic) Transitivity: If A➜G and G➜BH, then A➜BH
    - Decomposition: If A➜BH, then A➜B and A➜H
        - Therefore, by definition, A➜H is extraneous
        - F_c = {A➜G, BI➜C, G➜BH}
- Thus, we arrive at our final computation since no functional dependency has extraneous attributes and all dependencies have a unique left-hand side:
    - F_c = {A➜G, BI➜C, G➜BH}

---

c) Create a 3NF decomposition of R while indicating all candidate keys on resulting schemas.

- Let us use the 3NF synthesis algorithm. We will use our previously established canonical cover F_c = {A➜G, BI➜C, G➜BH} along with our established candidate key AI.
    - We loop through each dependency in F_c and create schemas appropriately:
        - R1 = (A, G)
            - Primary key: A
        - R2 = (B, I, C)
            - Primary key: BI
        - R3 = (G, B, H)
            - Primary key: G
    - At this point, no schema contains the candidate key AI, so let us create another schema:
        - R4 = (A, I)
            - Primary key: AI
                - AI has already been established as a candidate key (2.a)
- Thus, we have generated a 3NF decomposition of R

Problem 3: Tutoring Schema

Given:

- *tutors(tutor_id, tutor_name, email, graduation_date)*
- *tutor_topics(tutor_id, course_number, course_name, topic)*
- + other appropriate definitions of foreign keys, etc.

a) *List all non-trivial functional and multivalued dependencies that hold on this database (w/ proper rationale).*

- Non-trivial functional dependencies:
  - tutor_id ➜ tutor_name, email, graduation_date
    - Since tutor_id is the primary key in tutors, we know that tutor_id functionally determines each of the attributes in tutors (plus their appropriate combinations)
  - *course_number ➜ course_name
    - Let us note that under normal assumptions, it seems legitimate that course_number could potentially functionally determine course_name since it seems very likely that a course name for an associated course number is not likely to change (clearly this would not hold if there are changes to the course names in relation to the same course number)
    - We will not officially consider this a non-trivial functional dependency, but it seemed appropriate to consider.
- Non-trivial multivalued dependencies:
  - tutor_id ➜➜ topic
    - We can imagine that a tutor can teach various subjects within the same course_number/course_name. So, given that the value of topic specifically comes from what the tutor is capable of addressing, we can clearly determine that topic is independent of the other attributes, but it is dependent on the tutor itself (tutor_id), thus generating a multivalued dependency. Clearly, this is not the case for any of the other attributes in tutor_topics
  - Let us note that we will note state the functional dependencies as multivalued dependencies.

---

b) *Determine if the above schema is a "good" design" considering the perspective of functional and multivalued dependency theory.*

Although the given schema would certainly be functional considering that the database will be relatively small/simple and that there are no redundancies through the few functional dependencies that already do exist (~BCNF), we can determine that this schema is not ideal due to the following rationale. Let us consider how tutor_topics is not as useful as it

could be since it does not have any functional dependencies, making it inefficient at retrieving certain pieces of data (which is clearly not an optimal database design). This aspect seems to defeat the purpose of tutor_topics itself. So, taking into account our establishments from part (a), we can see that it would not be as easy/convenient as it could be to find information relating tutors and topics due to the absence of certain dependencies. Therefore, considering functional/multivalued dependency theory, we can conclude that the given schema is not the best.

---

c) Determine if we can propose a "better" design (w/ proper rationale)

Let us propose a more optimal schema design considering 4NF decomposition with tutor_topics. With this, taking into account the multivalued dependency of tutor_id ➜➜ topic, we can imagine splitting up tutor_topics to make it more convenient and efficient. Let us propose the following design using the previously established dependencies:

- tutors(<u>tutor_id</u>, tutor_name, email, graduation_date)
- tutor_topics(<u>tutor_id, topic</u>)
- course_topics(<u>topic</u>, course_number, course_name)

Now, we utilize the multivalued dependency tutor_id ➜➜ topic in order to better relate tutors and topics themselves, as well associating the appropriate topics with their respective courses. Therefore, we can see that topic functionally determines course information, which seems quite appropriate in the context of this problem. To reemphasize, this new design generates more dependencies in regards to topic, thus allowing for more efficiency and practicality in the retrieval of its associated information. At this point, considering the size and complexity of this database, we can determine that this design is more ideal than the given design since it would be much easier to keep track of the availability of tutors, as well as what courses/topics they are familiar with.

---

d) Write a SQL query that retrieves the tutor_name and email of all tutors who can provide tutoring for the topic "Python" OR for the class "CS121".

```
SELECT tutor_name, email
FROM tutor_topics NATURAL LEFT JOIN tutors NATURAL LEFT JOIN course_topics
WHERE topic = 'Python' OR course_number = 'CS121';
```