**Database Management Systems (COP 5725)**

**(Spring 2016)**

**Instructor: Dr. Markus Schneider**

**TA: Kyuseo Park, Lin Qi**

**Homework 2**

|  |  |
| --- | --- |
| **Name** | **Saurabh Prasad** |
| **UFID** | **88979398** |
| **Email Address** | **saurabh14@ufl.edu** |

Pledge (Must be signed according to UF Honor Code)

On my honor, I have neither given nor received unauthorized aid in doing this assignmen.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature

For scoring use only:

|  |  |  |
| --- | --- | --- |
|  | Maximum | Received |
| Exercise 1 | 25 |  |
| Exercise 2 | 25 |  |
| Exercise 3 | 50 |  |
| Total | 100 |  |

**Exercise 1**

1. Let W be the relation schema of a relation R, and let A, B ⊆ W. Please give the formal definition of a functional dependency (FD). [4 points]
2. B is functionally dependent on A i.e. B-->A, if for each value in A exactly one value belongs to B.

A --> B ⇔ ∀ t1, t2 ∈ R: t1[A] = t2[A] ⇒ t1[B] = t2[B] for all possible relations R over W.

1. Given a set F of functional dependencies, explain what the closure F+ of F is. Describe two main drawbacks in calculating the closure.
2. Closure of F, F+ is all the FDs which can be implied from the FDs of F.

Some of the main drawbacks are-

1. F+ has a large redundant set of FDs that has to be checked as consistency tests for database modifications.
2. In general, there are very many FDs in F+ so that the handling with F+ becomes difficult.

1. What are the rules for 𝐹𝑐 to be a canonical cover of a given set 𝐹 of FD’s?
2. Canonical cover Fc for F is a set of dependencies such that F logically implies all dependencies in Fc and Fc logically implies all dependencies in F, and further

– No functional dependency in Fc contains an extraneous attribute.

– Each left side of a functional dependency in Fc is unique.

1. What are the steps of computing the canonical cover?
2. To compute a canonical cover for F:

repeat

Use the union rule to replace any dependencies in F α1 → β1 and α1 → β2 with α1 → β1 β2 Find a functional dependency α → β with an extraneous attribute either in α or in β If an extraneous attribute is found, delete it from α → β

until

F does not change

1. What is 2NF? What is the kind of functional dependency it eliminates from 1NF? What is 3NF, what is the kind of functional dependency it eliminates from 2NF?
2. A relation schema is in 2NF then only if it is in 1NF. Also, for all FDs X -> {A} holds, if attribute A is not part of a key and X is a key, then there is no FD Y -> {A} with Y is a subset of X.

2NF eliminates partial functional dependencies.

A relation schema R with associated FDs F is in 3NF, if, and only if, it is in 2NF and for each FD A->B ∈ F at least one of the following conditions holds:

1. B⊆A

2. A is a super key of R.

3. B is a part of some candidate key of R.

1. What is BCNF? What’s the difference between 3NF and BCNF?
2. A relation schema R with FDs F is in Boyce-Codd normal form (BCNF), if and only if, it is in 3NF and for each FD A -> B ∈ F at least one of the following conditions holds:

B ⊆ A, i.e., the FD A → B is trivial.

A is a superkey of R.

BCNF eliminates dependencies among attributes that are part of a candidate key.

1. Suppose we have a relation R(A,B,C,D,E) and the FD's A → DE, D → B, and E → C. If we project R (and therefore its FD's) onto schema ABC, what is true about the key(s) for ABC? Explain why.

(a) Only ABC is a key

(b) Only A is a key

(c) Only DE is a key

(d) A, B, and C are each keys

(e) None of the above

1. (b) Only A is a key. First, note that A+ = ABCDE. Therefore, A is certainly a key. However, there cannot be any other keys, since neither B nor C has anything else in their closures, because they are not on the left sides of any FD's. Thus, A is the only key.
2. Please describe the three inference rules of the Armstrong axioms.
3. The three Armstrong axioms are-

* reflexivity rule: Let B ⊆A. Then always A -> B holds.
* augmentation rule: If A -> B holds, then also A ∪C -> B ∪C holds.
* transitivity rule: If A -> B and B -> C holds, then also A -> C holds.