CS 4320/5320 Homework 3

Fall 2016

Due Sun Oct 30, 2016, 11:59pm

This assignment is due 30 October 2016. It is out of 100 points and counts for 10% of your overall grade.

1 Functional Dependencies (20 Points)

Prove each of the statements below. Remember that if your statement is an "if and only if" you need to prove both directions.

(a) (6 Points) Let X, Y be sets of attributes of a relation. Prove $(X)^+ = (Y)^+$ if and only if $X \to Y$ and $Y \to X$.

(b) (6 Points) Prove that relation R satisfies $X \to Y$ if and only if X is a superkey of $\pi_{XY}(R)$.

(c) (8 Points) Let X and Y be sets of attributes with $X \cap Y = Z$ and assume X is nonempty. Prove that for any relation R whose attribute set is exactly $X \cup Y$, if $Z \to Y$ holds on R, then $R = \pi_X(R) \bowtie \pi_Y(R)$.

2 Normal Forms (30 Points)

Consider a relation R with attributes $ABCDEFGH$ and the following functional dependencies:

- $B \to CDF$
- $A \to EH$
- $CDE \to BGF$

(a) (8 points) Find all keys of R **without** iterating over all possible subsets of attributes. Explain your process.

(b) (8 points) R is neither in BCNF nor in 3NF. For each of these normal forms, explain why R does not satisfy it.

(c) (14 points) Find a decomposition of R into 3NF that is both lossless-join and dependency-preserving. Write down your reasoning. (See Section 19.6.2 in your textbook for details of the algorithm(s) to use.)

3 Implementation Part (50 Points)

In this section, you will implement an algorithm to determine whether a decomposition of a relation into two relations is lossless and/or dependency preserving. Your textbook provides the criteria you need to check in Section 19.5. You will probably find it useful to implement an algorithm that computes attribute closures (Figure 19.4, page 614), and use that as a subroutine.

For checking whether a decomposition is lossless-join, the logic should be straightforward based on the criterion in the textbook p. 620.

For checking dependency-preservation, the following functional dependency statements are helpful:

1. Let X, Y, F, F_X, F_Y be as defined in your textbook on page 621. $(F_X \cup F_Y)^+ \subseteq F^+$ always holds.
2. Let F and G be sets of functional dependencies. $F^+ \subseteq G^+$ if and only if $\forall f \in F$, $G \models f$.

Because of Statement 1, you only need to check whether $F^+ \subseteq (F_X \cup F_Y)^+$ instead of checking $(F_X \cup F_Y)^+ = F^+$. To achieve this, Statement 2 indicates that you can check whether for every $f \in F$, $f \in (F_X \cup F_Y)^+$. This can be done naively by computing all of F_X and all of F_Y, but you don't want to do that as it would be a very inefficient exponential time computation. You should use the following polynomial time algorithm instead.

```
**Data**: Functional Dependency $f = \alpha \to \beta$, $F$, $X$, $Y$
**Result**: true if $f \in (F_X \cup F_Y)^+$, false otherwise
result = $\alpha$ ;
**while** *result changed in previous iteration* **do**
    **for** $Z \in \{X, Y\}$ **do**
        $temp = (result \cap Z)^+ \cap Z$;
        $result = result \cup temp$ ;
    **end**
**end**
**if** $\beta \subseteq result$ **then**
    return true;
**else**
    return false;
**end**
```

Algorithm 1: Algorithm for checking whether $f \in (F_X \cup F_Y)^+$. In computing the expression $(result \cap Z)^+$, we take attribute closure with respect to F.

We have provided you with the following skeleton code:

- **Attribute.java**: Represents a single attribute, immutable
- **AttributeSet.java**: A set of attributes, backed by a Java set
- **FunctionalDependency.java**: A simple functional dependency
- **FDChecker.java**: Main class to implement
- **Test.java**: Contains some junit tests

We only require you not change the method headers in FDChecker. You will be graded primarily on the ability to pass automated test cases, with a small number of points for code style. As before, the code style points will primarily be used to penalize egregiously bad code.

Submission Instructions

Prepare a .zip archive, *Submission.zip* containinig:

- A .pdf file containing all of your written answers for questions 1-3. As usual, these must be typed, scans of handwritten answers are not accepted.
- A .zip file containing all of your .java files, and a README file which includes your name, netid, and any comments about the code which might be relevant to the graders.

Ensure all files contain the netID of everyone in your group and clearly state which answer corresponds to which question. Submit this file to CMS by the deadline.