CS 4320/5320 Homework 3

Spring 2016

Due 3/25/2016, 11:59pm

This assignment is due March 25, 2016. It is out of 60 points and counts for 10% of your overall grade.

1 Functional Dependencies (15 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) (5 Points) Let A, B be sets of attributes of a relation. $A^+ = B^+$ if and only if $A \to B$ and $B \to A$.

(b) (5 Points) A relation R satisfies $X \to Y$ if and only if X is a key of $\pi_{XY}(R)$.

(c) (5 Points) Let A and B be sets of attributes with $A \cap B = X$ and assume X is nonempty. Prove that for any relation R whose attribute set is exactly $A \cup B$, if $X \to B$ holds on R, then $R = \pi_A(R) \bowtie \pi_B(R)$.

2 ER Diagram (10 Points)

You are thinking hard in your cubicle after returning from the weekly meeting. You work for the Investment Bank of Avatar as a system engineer in the back office. During the meeting, the risk management team complained to you. They need a database that keeps track of all derivative contracts that the front office trades or researches. Before setting up the database with your technology team, you need to present an ER diagram to your colleagues that models the following design.

Each **derivative contract** is uniquely identified by an *ID* and contains two other attributes, *name* and *contract size*. Each contract must be run by either a trader or a research analyst in the **front office**.

Traders are identified by a unique *ID* and also have a *name*. For each trader you also need to record their *PnL Explain* (the profit and loss explain that reports a trader's performance). **Research Analysts** are identified by their unique *ID*, and you need to record their *name* and their *team's name*.

Any contract can be evaluated for risk management purposes. Every time a contract is evaluated, the database must record the date (this is not unique, a contract can be evaluated several times on the same day) and the current settlement price at that moment. Every evaluation is recorded as an **Evaluation Event**. For a specific contract, every **Evaluation Event** is uniquely identified by a *number*. However, two different **Evaluation Event**s can have the same *number* if they relate to two different contracts.

Draw an ER diagram to represent the above scenario. If you need to make any assumptions about the data, state them clearly.

3 Normal Forms (10 Points)

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

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

(a) (2 points) Is it **always** beneficial to decompose a relation? Explain your answer.

(b) (3 points) Find all keys of R **without** iterating over all possible subsets of attributes. Explain your process.

(c) (2 points) R is neither in BCNF nor in 3NF. For each of these normal forms, explain why R does not satisfy it.

(d) (3 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.)

4 Implementation Part (25 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 on pages 620-621. You will probably find it useful to implement an algorithm that computes attribute closures (p. 614, Figure 19.4), 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

Submit two files:

- 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.