Fall, 2005 CIS 550

Database and Information Systems
Homework 3 *Solutions*

Problem 1 [15 points]: Consider a relation R with four attributes $ABCD$. You are given the following dependencies: $A \rightarrow B$, $B \rightarrow D$, $CD \rightarrow AB$.

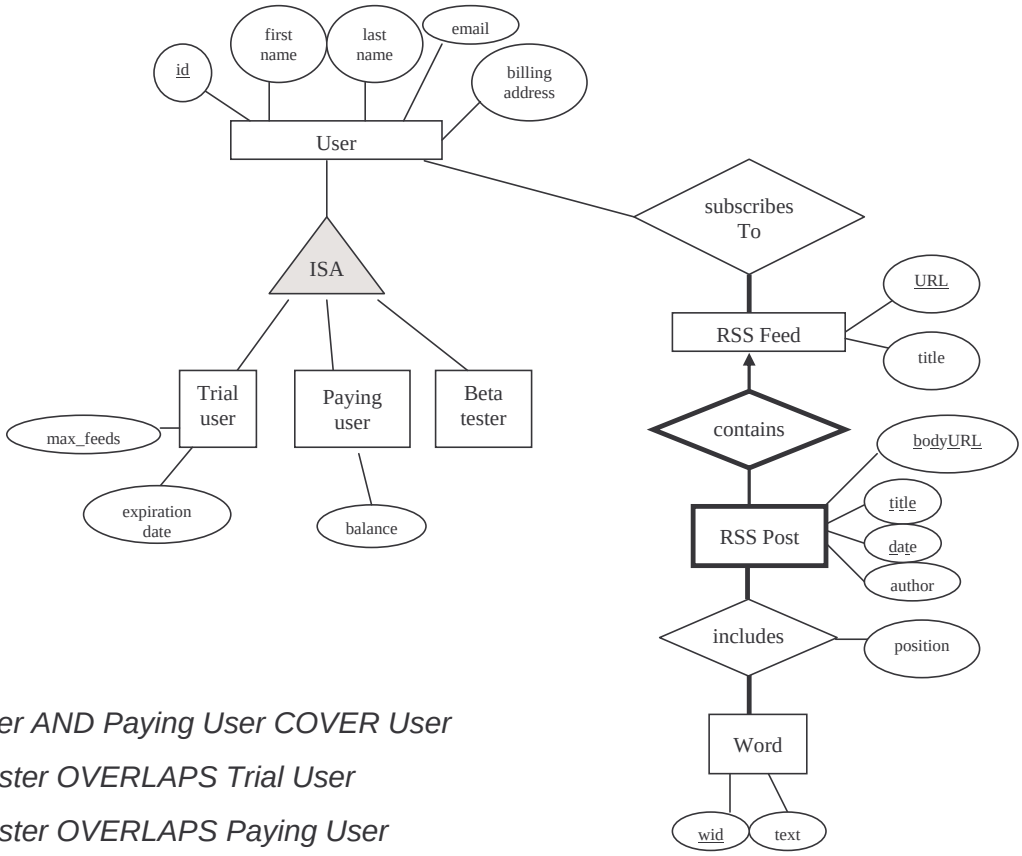
1. List all keys for R. (other than superkeys)
   AC, BC, CD
2. Is R in 3NF?
   Yes
3. Is R in BCNF?
   No, because A is not a superkey in $A \rightarrow B$, nor is B in $B \rightarrow D$

Problem 2 [30 points]: The task is to design a DBMS-based RSS aggregator system. This system will be used to store, search, and view users RSS feeds. A user will create an account, subscribe to feeds, and read or search the feeds.

- Each user will have a unique numeric ID. Additionally, the first and last names, email address, and billing address will need to be stored.
- Each user may subscribe to one or more RSS feeds. One RSS feed may be shared among multiple users.
- RSS feeds have URLs and titles.
- RSS feeds have multiple RSS posts.
- RSS posts have a title, date, author, and URL.
- Each RSS post consists of a set of word occurrences and their positions.
- Every user is either a paying customer or trial user.
- Trial users have expiration dates and a maximum number of feeds.
- Paying customers have no maximum but have a balance associated with them.
- Original users of the system (some still trial users and others paying customers) are further classified as beta testers.

Draw an ER diagram for the RSS aggregator system. The ER diagram should include various attributes, key, participation constraints, overlap and covering constraints.

Here is one of many possible ER diagrams:



Trial User AND Paying User COVER User

Beta Tester OVERLAPS Trial User

Beta Tester OVERLAPS Paying User

(We could also have added an RSS Post seq. no.)

Problem 3 [25 points]: Consider a relation R with six attributes $ABCDYZ$ and the FD set $F = \{AB \rightarrow Z, AC \rightarrow D, Y \rightarrow C, ZB \rightarrow D, BD \rightarrow Z\}$. Let F^+ denote the closure set of F.

1. For each of the following attribute sets, do the following: (i) write down a minimal cover of the subset of F^+ that holds over the set; (ii) name the strongest normal form that is not violated by the relation containing these attributes; (iii) decompose it into a collection of BCNF relations if it is not already in BCNF.

   (a) BDZ
   (i) $\{ZB \rightarrow D, BD \rightarrow Z\}$; (ii) BCNF; (iii) already in BCNF

   (b) $ABCD$
   (i) $\{AB \rightarrow D, AC \rightarrow D\}$ (ii) 1NF; (iii) $R1(ABD), R2(ACD), R3(ABC)$

2. For each of the following decompositions of $R = ABCDYZ$, with the same set of functional dependencies F, say whether the decomposition is (i) dependency preserving, and (ii) lossless join.

   (a) $\{ABYD, ABCYZ\}$
   lossless join, key is ABY

   (b) $\{ACD, YC, YZ, ABDZ\}$
   dependency preserving

Problem 4 [20 points]: Suppose you are given a relation $R(A, B, C, D, E)$. For each of the following (complete) sets of FDs, (i) identify the candidate key(s) for R, and (ii) state whether or not the proposed decomposition of R into smaller relations is a "good" decomposition and briefly explain why or why not.

1. $A \rightarrow B$, $B \rightarrow CE$, $C \rightarrow D$. Decompose into AB, BCE, and CD.
   (i) A (ii) good, dependency preserving and lossless join

2. $C \rightarrow A$, $B \rightarrow D$. Decompose into ACE and BD.
   (i) BCE (ii) not good, dependency preserving only, meaning it cannot be reassembled losslessly

Problem 5 [15 points]: For a case where dependency preserving decompositions aren't used, how could a hypothetical DBMS ensure that FDs are satisfied?

This hypothetical DBMS would need to be given a list of the original functional dependencies, along with the (decomposed set of) relations that form the schema. To validate functional dependencies, it would need to execute lossless joins to recompose *relations upon which the functional dependencies might be tested, and then test the FDs. Note that this requires a join for each insertion — which is why no real DBMS does this!*