CSE 132A

Solutions to Practice Problems on Schema Design

1. We apply the lossless join test. The tableau corresponding to the decomposition ρ is:

A	B	C	D
a	b	$-$	$-$
$-$	b	c	$-$
$-$	$-$	c	d
a	b	c	d

After chasing this with respect to $F = \{B \to A, C \to B\}$ the last row becomes $< a, b, c, d >$.

2. The fds $AB \to C, C \to E, E \to C$ are obviously preserved because each applies to one relation in the decomposition. Consider $C \to D$, which does not apply to a single relation. We compute the closure of C relative to the local fds according to the algorithm described in class. Initially we have C in the relation CE and in ABC. The closure of C is CED, so we obtain E within CE. Now E is available in ADE. The closure of E is ECD so we obtain D within ADE. Since D is now on the list, it is in the closure of C wrt the local fds, so $C \to D$ is preserved. It remains to check $AB \to E$. We now start with AB, so we have AB within ABC, and A within ADE. The closure of AB is $ABCED$ so we obtain C within ABC. Now we have C within CE, and $C^+ = CED$ so we get E within CE. Thus, $AB \to E$ is also preserved.

3. We first rewrite the fds so that we only have single attributes on the righthand side:

$$A \to C, AB \to C, C \to I, C \to D, CD \to I, EC \to A, EC \to B, EI \to C$$

We next look at each of the fds and see if they are redundant. $A \to C$ is not, because A^+ (wrt the other fds) is A. $AB \to C$ is clearly redundant, since it is implied by $A \to C$. We eliminate it from the list. Similarly, $C \to D$ is not redundant. However, $C \to I$ is redundant, and we eliminate it. Next, $CD \to I$ is not redundant wrt the fds left on the list. Similarly, $EC \to A$, $EC \to B$, and $EI \to C$ are not redundant. The remaining list of fds is so far:

$$A \to C, C \to D, CD \to I, EC \to A, EC \to B, EI \to C.$$

Next, we check for redundant attributes on lefthand sides of fds. Consider $CD \to I$. We need to check whether C or D can be eliminated. C can be eliminated if $D \to I$ is implied by the fds on the list (the entire list!). Clearly, $D^+ = D$, so $D \to I$ is not implied. Next, D can be eliminated if $C \to I$ is implied. Now $C^+ = CDI$ so $C \to I$ is implied. So D is redundant and we replace $CD \to I$ by $C \to I$ in the list of fds. It easy to see that there are no redundant attributes in

$$EC \to A, EC \to B, EI \to C$$

so the final minimized set of fds is:

$$A \to C, C \to D, C \to I, EC \to A, EC \to B, EI \to C.$$

4.

(a) IS is a key, i.e. a minimal superkey. Indeed, $(IS)^+ = ISDBQO$. To see that it is minimal, note that I is not a key and S is not a key.

(b) IS is the only minimal key. To see this, it is enough to note that any key K must contain IS. This is obvious, because neither I nor S appear on the righthand side of any fd.

(c) A BCNF decomposition with lossless join obtained by the algorithm is:

$$\rho = \{SD, IB, IO, ISQ\}.$$

(See separate bcnf file.)

(d) It is easy to check that $S \to D, I \to B, IS \to Q, B \to O$ is minimal. Thus, $\{SD, IB, ISQ, BO\}$ is a 3NF decomposiion which is dependency preserving. Note that the key IS is a subset of one relation in the decomposition (ISQ) so there is no need to add it, and the decomposition also has lossless join.

5.

(a) By decomposing $ABCD$ using $D \to C$ (which violates BCNF within $ABCD$), we obtain $\{DC, ABD\}$. Clearly, DC is in BCNF (no violation can occur in a two-attribute relation). In ABD, the only violations could come from fds with a single attribute on the lefthand side. Thus, it is sufficient to check A^+, B^+, and D^+ within ABD: $A^+ \cap ABD = A, B^+ \cap ABD = B, D^+ \cap ABD = D$. So ABD is in BCNF and the final BCNF decomposition is $\{DC, ABD\}$.

(b) It is necessary to check preservation of $AB \to C$ and $B \to C$. Our algorithm shows that $AB \to C$ is preserved, but $B \to C$ is not.

(c) First, we rewrite the fds as

$$AB \to C, AB \to D, D \to C, B \to C.$$

Clearly, $AB \to C$ is redundant and the remainder set is minimal. Thus, the 3NF decomposition is $\{ABD, DC, BC\}$. Note that ABD contains the key AB, so there is no need to add a key to the schema. So the above 3NF decomposition is dependency preserving and has lossless join.