CSE 232

Solutions to Practice Problem Set #2

1. The tableau corresponding to ρ is:

A	B	C	D
a	b	c_1	d_1
a_2	b	c	d_2
a_3	b_3	c	d
a	b	c	d

Chasing this with respect to $F = \{B \to A, C \to B\}$ leads to identifying a_2 with a, b_3 with b, and finally a_3 with a (previously a_2). Thus, 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 minimal key. 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 Figure 1.)

(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 (this happens to be the same as the BCNF above). 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.

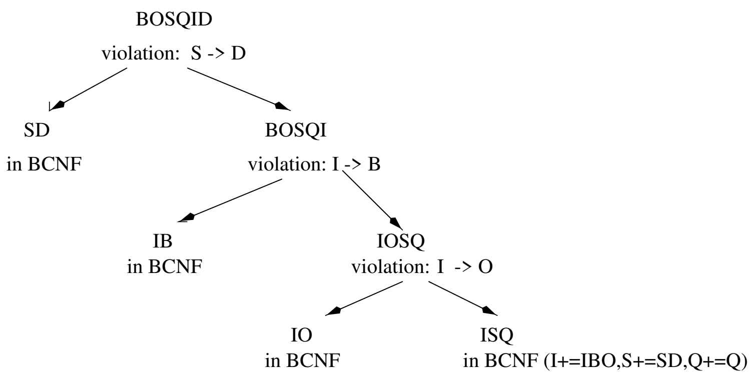


Figure 1: BCNF decomposition in problem 5(c).

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.

6. The tableau corresponding to the query is:

A	B	C	D	E
a	b_1	c	d_1	e
a	b_2	c_1	d	e
a_1	b	c	d	e_1
a	b	c	d	e

Chasing with respect to $A \to B$ and $E \to D$ yields:

A	B	C	D	E
a	b_1	c	d	e
a	b_1	c_1	d	e
a_1	b	c	d	e_1
a	b	c	d	e

Chasing with respect to $D \to\to E$ produces (after elimination of duplicates):

A	B	C	D	E
a	b_1	c	d	e
a	b_1	c_1	d	e
a_1	b	c	d	e_1
a	b_1	c	d	e_1
a_1	b	c	d	e
a	b_1	c_1	d	e_1
a	b	c	d	e

The minimized tableau is:

A	B	C	D	E
a	b_1	c	d	e
a_1	b	c	d	e
a	b	c	d	e

and the corresponding minimized query is:

$$\pi_{ACDE}(R) \bowtie \pi_{BCDE}(R)$$

7. (i) Suppose, towards a contradiction, that there is some relation R that satisfies $A \rightarrow\rightarrow B, C \rightarrow B$ and violates $A \rightarrow B$. Since R violates $A \rightarrow B$, there must exist in R tuples:

A	B	C
a	b_1	c_1
a	b_2	c_2

where $b_1 \neq b_2$. Since R satisfies $A \rightarrow\rightarrow B$, it must contain the tuples:

A	B	C
a	b_1	c_1
a	b_2	c_2
a	b_1	c_2
a	b_2	c_1

But then R violates $C \rightarrow B$, a contradiction. It follows that R could not violate $A \rightarrow B$.

(ii) We need to show that if R contains tuples

A	B	C	D
a	b	c_1	d_1
a	b_1	c	d

it must also contain the row $< a, b, c, d >$. Since R satisfies $A \rightarrow\rightarrow BC$, R must contain the rows:

A	B	C	D
a	b	c_1	d_1
a	b_1	c	d
a	b	c_1	d
a	b_1	c	d_1

Since R satisfies $A \to\to C$, it must also contain the row $< a, b, c, d >$ which is

$$\pi_{AC}(< a, b_1, c, d_1 >) \bowtie \pi_{ABD}(< a, b, c_1, d >)$$

It follows that R satisfies $A \to\to B$.

8. The serializability graph has the following edges:

$$< T3, T1 >, < T2, T3 >, < T1, T2 > .$$

The graph is cyclic so the schedule is *not* serializable.

9. An example is the following:

$$T1 = lockA; lockB; unlockA; unlockB$$

$$T2 : lockA; unlockA; lockB; unlockB$$

A nonserializable schedule for $T1$ and $T2$ is:

$$\begin{array}{ll} T2: & lockA \\ T2: & unlockA \\ T1: & lockA \\ T1: & unlockA \\ T1: & lockB \\ T1: & unlockB \\ T2: & lockB \\ T2: & unlockB \end{array}$$

The serializability graph for this schedule is $< T1, T2 >, < T2, T1 >$, which is cyclic.

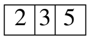
10.

(i) The protocol does not ensure serializability. For example, take the transactions T1, T2 in problem 8, where A is replaced by $A1$ and B by $A2$. Both transactions satisfy the protocol but the schedule shown in problem 8 is not serializable.

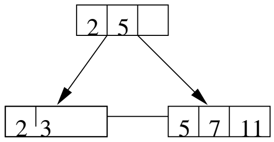
(ii) The protocol does prevent deadlock. To see this, suppose there is a cycle in the "waits-for" graph for the transactions. Suppose the nodes on the cycle are $T1, ..., Tn, T1$. It is easy to see that $T1$ must have requested an entity Ai after Aj with $i < j$. So $T1$ must violate the protocol.

11. The B+-trees obtained are the following:

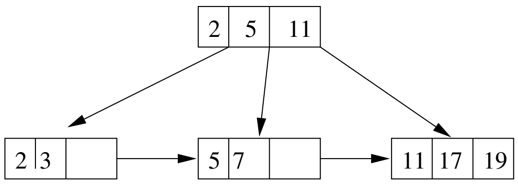
insert 2,3,5:



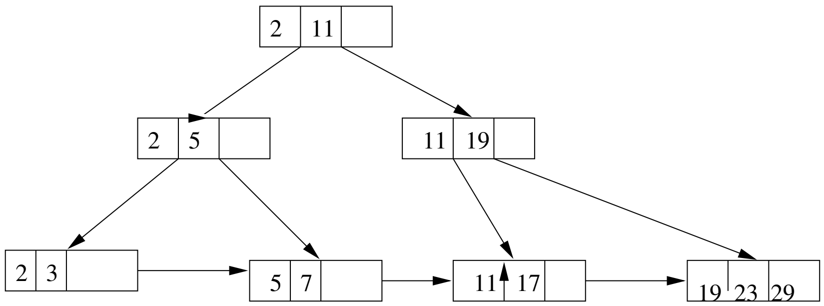
insert 7, 11



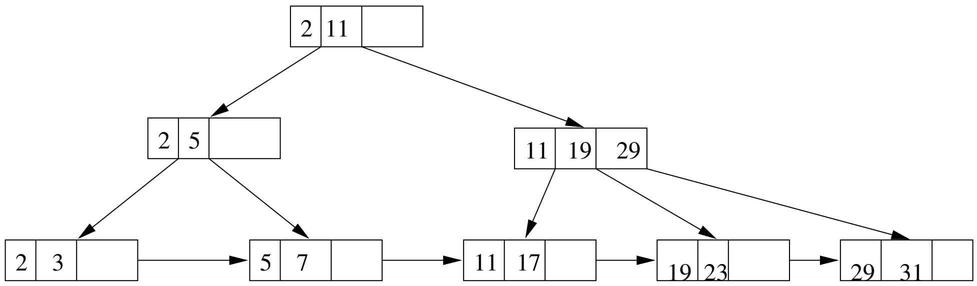
insert 17, 19



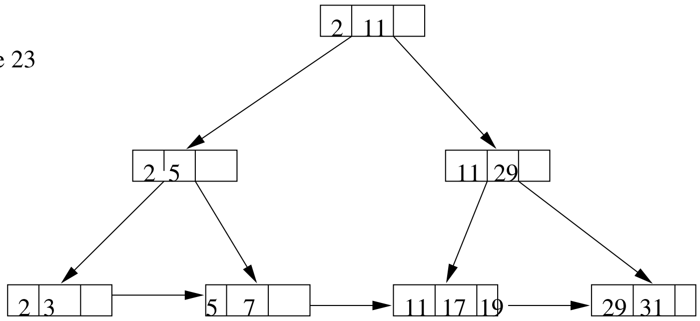
insert 23,29



insert 31



delete 23



12. The complex tuple calculus query is:

$$\begin{array}{l}\{t : [drinker : \mathbf{dom}, bars : \{\mathbf{dom}\}] \mid \\ \exists f : [drinker : \mathbf{dom}, bar : \mathbf{dom}](freq(f) \wedge t.drinker = f.drinker) \wedge \\ \forall b : \mathbf{dom}(b \in f.bars \longleftrightarrow \\ \exists g : [drinker : \mathbf{dom}, bar : \mathbf{dom}](freq(g) \wedge g.drinker = t.drinker \wedge g.bar = b))\}\end{array}$$