

CSCI4380 – Database Systems

Spring 2004, Examination II

(100 points)

1. (15 points) Consider the following relational schema:

Supplier(sname, itemname, price) – supplier **sname** sells **itemname** at **price**

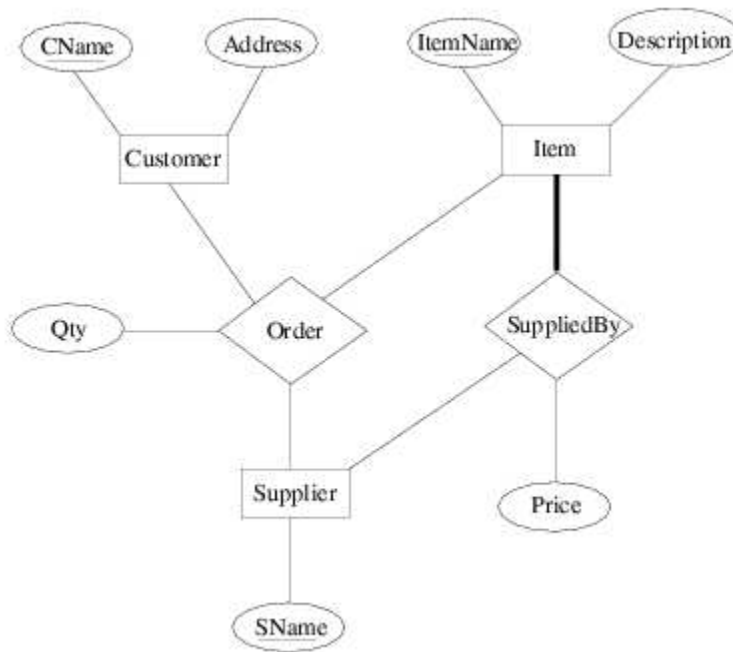
Order(cname, sname, itemname, qty) – customer **cname** has ordered **qty** of item **itemname** from supplier **sname**

Customer(cname, address) – customer **cname** lives at **address**

Item(itemname, description) – info about items

- (a) (10 points) Draw the E-R diagram from which the above schema might have been derived. Specify the keys.
- (b) (5 points) Add the following constraint to the diagram: *every item is supplied by some supplier*.

Note that from the tables, we infer three entities: supplier, customer, and item. Order is a relationship, which is 3-way, since it includes cname, sname and itemname; qty is an attribute of this relationship. In addition “sells” or “suppliedby” is another binary relationship between supplier and item. The figure below shows the ER diagram (part a), and the bold line shows the participation constraint (part b).



2. (10 points) Given FDs: $\{AB \rightarrow C, A \rightarrow D, D \rightarrow E\}$, show that $AB \rightarrow CDE$ can be derived using only the three Armstrong’s rules (reflexivity, augmentation and transitivity).

$AB \rightarrow C$ implies $AB \rightarrow AC$, augmenting by A - (1)

$A \rightarrow D$ implies $AC \rightarrow CD$, augmenting by C - (2)

Then by transitivity on (1) and (2), $AB \rightarrow CD$ - (3)

$D \rightarrow E$ implies $CD \rightarrow CDE$, augmenting by CD - (4)

Then by transitivity on (3) and (4), $AB \rightarrow CDE$ - (5)

3. (25 points) Given the following FDs over relation with attributes $\mathcal{R}(C, D, P, R, S, T)$:

- $DCS \rightarrow P$
- $PS \rightarrow C$
- $CST \rightarrow P$
- $PST \rightarrow CR$
- $PSCT \rightarrow R$
- $PST \rightarrow C$

(a) (10 points) Compute a minimal cover for the FDs.

After simplifying the right-hand sides, we note additional steps:

- $DCS \rightarrow P$: cannot reduce LHS
- $PS \rightarrow C$: cannot reduce LHS
- $CST \rightarrow P$: cannot reduce LHS
- $PST \rightarrow C$: we can remove T , since $PS \rightarrow C$ is implied by (b). But removing T , makes the dependency $PS \rightarrow C$, which is a duplicate. Throw away!
- $PST \rightarrow R$: cannot reduce LHS
- $PSCT \rightarrow R$: we can remove C , since $PST \rightarrow R$ is implied by (e). But removing T , makes the dependency $PST \rightarrow R$, which is a duplicate. Throw away!
- $PST \rightarrow C$: duplicate, throw away!

The minimal cover is therefore:

- $DCS \rightarrow P$
- $PS \rightarrow C$
- $CST \rightarrow P$
- $PST \rightarrow R$

(b) (5 points) Do a decomposition of \mathcal{R} using 3NF synthesis, show all resulting tables and FDs over those tables.

Using 3NF synthesis we get the following relations and FDs:

- $R_1 = DCSP, F_1 = \{DCS \rightarrow P, PS \rightarrow C\}$
- $R_2 = PSC, F_2 = \{PS \rightarrow C\}$
- $R_3 = CSTP, F_3 = \{CST \rightarrow P, PS \rightarrow C\}$
- $R_4 = PSTR, F_4 = \{PST \rightarrow R\}$
- $R_5 = PSTD, F_5 = \{\}$. We had to add this relation, since none of the above is a superkey for \mathcal{R} . $PSTD$ is one possible superkey.

(c) (5 points) Check if any of the tables is not in BCNF, and if so, decompose it further using BCNF decomposition.

R_1 and R_3 are both not in BCNF, since PS is not a superkey for either, i.e., $PS \rightarrow C$ is a violation of BCNF. Using that we can thus decompose R_1 into $R_{1a} = PSC$ and $R_{1b} = DSP$. But $R_{1a} = R_2$ so it is discarded. Likewise, we can decompose R_3 into $R_{3a} = PSC$ and $R_{3b} = STP$, and once again, we can discard R_{3a} . The final decomposition is:

- $R_{1b} = DSP, F_1 = \{\}$
- $R_2 = PSC, F_2 = \{PS \rightarrow C\}$
- $R_{3b} = STP, F_3 = \{\}$
- $R_4 = PSTR, F_4 = \{PST \rightarrow R\}$
- $R_5 = PSTD, F_5 = \{\}$

(d) (5 points) Is the final result a dependency preserving decomposition?

No it is not, it is clear that the new tables do not satisfy all of the minimal cover FDs. In particular, $DCS \rightarrow P$ and $CST \rightarrow P$ have been lost!

4. (25 points) Answer the following:

(a) (5 points) Is the following schedule serializable: $r_1(x)w_2(y)r_1(z)r_3(z)w_2(x)r_1(y)$?

NO. The reason is that if T_1 happens before T_2 , then y value read by T_1 will be different. If T_2 happens before T_1 then T_1 will see a different x value.

(b) (10 points) Give an example of each of the following cases during concurrent transaction execution.

i. dirty read

$w_2(x)r_1(x)abort_2$

ii. non-repeatable read

$r_1(x)w_2(x)commit_2r_1(x)$ (I will also accept $r_1(x)w_2(x)r_1(x)$).

iii. lost update

$r_1(x)r_2(x)w_2(x)commit_2w_1(x)commit_1$ (I will also accept $r_1(x)r_2(x)w_2(x)w_1(x)$).

(c) (10 points) Given tables $T1(a1, a2, a3)$ and $T2(a1, a2, a3)$, and the following transaction:

```
SELECT T1.a1, T2.a1
FROM T1, T2
WHERE T1.a2=T2.a2 AND T1.a3=5 AND T2.a3=7
```

i. (5 points) Give a single statement that may cause a phantom.

One example is to insert a tuple with $a3=5$ in $T1$, another is to insert $a3=7$ in $T2$. There are other cases possible.

ii. (5 points) Describe how one can prevent this. Describe the locks one would need to hold (regular and intention locks) on the different objects (db, table, page, record, index) in the database.

Let's say that the select is transaction T_a , and the insert is T_b . To prevent phantoms, we need to acquire $T_a : IS(db)$ (meaning that T_a acquires an IS lock on database db), and $T_b : IX(db)$.

If there are no indexes used, then we must do: $T_a : S(T1, T2)$, and $T_b : X(T1, T2)$. This way the conflict is detected at table level.

If there are indexes, then we do: $T_a : IS(T1, T2)$, and $T_b : IX(T1, T2)$, and more importantly, $T_a : S(matchingindexpages)$, $T_b : S(matchingindexpages)$. This way conflicts are detected at index page level.

5. (10 points) Given the sequence of logs:

```
00 -- update: T1 writes P2
10 -- update: T1 writes P1
20 -- update: T2 writes P5
30 -- update: T3 writes P3
40 -- T3 commit
50 -- update: T2 writes P5
60 -- update: T2 writes P2
70 -- T2 abort
```

Show all steps taken to rollback T2.

Since only an abort happens and the system did not crash, we do not need to do any recovery or redo steps. All we have to do is UNDO the effect of T2 and write out CLRs. This is what we get:

```
00 -- update: T1 writes P2
10 -- update: T1 writes P1
20 -- update: T2 writes P5
30 -- update: T3 writes P3
40 -- T3 commit
50 -- update: T2 writes P5
60 -- update: T2 writes P2
70 -- T2 abort
80 -- CLR: Undo T2 LSN 60
90 -- CLR: Undo T2 LSN 50
100 -- CLR: Undo T2 LSN 20
110 -- T2 end
```

6. (15 points) Consider the following relational tables:

- student (id, name, address, status)
- professor (id, name, deptid)
- course (crscode, deptid, crsname, description)
- transcript (studentid, crscode, semester, grade)
- teaching (profid, crscode, semester)

Assume that these tables are published in one XML document as follows:

```
<tables>
  <student>
    <tuple>
      <id>. . .</id>
      <name>. . . </name>
      <address> . . .</address>
      <status>. . . </status>
    </tuple>
    . . .
  </student>
  <professor> . . .
  </professor>
  . . .
</tables>
```

As you can see, each table is under the root element **tables**. Each table is made up of **tuple** elements under the table name (e.g., **student**), which in turn are made up of the relational attribute. There will be one tuple per row in a table. The clip above shows an example of how student table might be stored. Other tables are recorded in a similar way.

Write XQueries for the following. Note that you can bind multiple variables in a single FOR or LET statement.

- (a) Find all profs who have taught CSCI4380.

```
FOR $p IN document('XXX.xml')//professor/tuple,  
  $t IN document('XXX.xml')//teaching/tuple[crscode='CSCI4380']  
WHERE $p/id = $t/id  
RETURN <result> $p </result>
```

Other variations are possible.

- (b) Find all course names in which “Joe Public” and got an A.

```
FOR $s IN document('XXX.xml')//student/tuple[name = 'Joe Public'],  
  $t IN document('XXX.xml')//transcript/tuple[grade = 'A'],  
  $c IN document('XXX.xml')//course/tuple  
WHERE $s/id = $t/studentid AND $t/crscode = $c/crscode  
RETURN <result> $c/crsname </result>
```

Other variations are possible.

- (c) Find all students who have taken more than 3 courses.

```
FOR $s IN document('XXX.xml')//student/tuple  
  LET $t = document('XXX.xml')//transcript/tuple[studentid = $s/id]  
  WHERE count ($t) > 3  
RETURN <result> $s </result>
```