# Solutions to Assignment 2, Winter 2014

1. Indicate whether each of the following statements is `TRUE` or `FALSE`. Each correct answer worths 4 points while an incorrect answer leads to a deduction of 2 points. [8]

   (a) The GROUP BY CUBE cannot be expressed by any other standard SQL statements.
   **Solution:** FALSE.

   (b) The timestamps protocol does not lead to dead lock.
   **Solution:** TRUE.

2. What is the main difference between data classification and clustering? [8]

   **Solution:** The followings are main differences between the two.

   (a) The information about the set of classes to be identified including the number of classes and the names of all classes, are known ahead of time in data classification but not in clustering.

   (b) A similarity and/or distinctiveness measurement is given in clustering but not in data classification.

3. Consider a database with the following three tables with obvious meanings:
   course(course_id, course_name, no_credit),
   prerequisite(course_id, prerq_id),
   registration(student_id, course_id, grade).

   Note that the underlined attributes are keys of the tables. Write one or more triggers to impose the constraint that no student can register a course without satisfying its prerequisite. Note that a student `sid` satisfies the prerequisite of a course `cid` if for each row ⟨`cid`,`pid`⟩ in prerequisite, there exists a row ⟨`sid`,`pid`,`grade`⟩ in registration such that `grade` is NOT NULL and `grade` $\geq 1$.

   It is sufficient to (1) identify the list of all the events that must be monitored; and (2) present a correct trigger for any one event in the list. [12]

   **Solution** To impose the constraint, the following events must be monitored:

   - INSERT and UPDATE on Prerequisite
   - INSERT on Registration
   - UPDATE and DELETE on Registration

   Note that it is OK without the first event if we assume that the modified prerequisites will not affect any courses registered already. For simplicity, one may assume that the update of a grad on registration will not affect the eligibility of any registered courses.

   One trigger monitoring the 2nd event is given below.

```
CREATE TRIGGER check_prerequisite
BEFORE INSERT on Registration
FOR EACH ROW
DECLARE count INT
BEGIN
   SELECT  COUNT(*) INTO count
   FROM    Prerequisite p
```

```
    WHERE   p.course_id = new.course_id  and
            p.prerq_id NOT IN (SELECT course_id
                               FROM   registration r
                               WHERE  r.student_id = new.student_id AND
                                      r.grade is NOT NULL and r.grade >= 1
                               );

    IF (count > 0)
    THEN raise_application_error(-2000,'Prerequisite is not satisfied');
END
```

4. Consider a relation schema $R = ABCDE$, functional dependencies

   B $\rightarrow$ E, A $\rightarrow$ D, D $\rightarrow$ E

   and a decomposition $D = \{AB, BC, ADE\}$ of R. Is $D$ join lossless with the given FDs? Explain. [12]

   **Solution** It is not join lossless as demonstrated by the following counter example.

   Consider the following instance of $R$ which satisfies all the constraints mentioned above:

| A | B | C | D | E |
|---|---|---|---|---|
| $a_1$ | $b_1$ | $c_1$ | $d_1$ | $e_1$ |
| $a_2$ | $b_1$ | $c_2$ | $d_1$ | $e_1$ |

   $R1$:

| A | B |
|---|---|
| $a_1$ | $b_1$ |
| $a_2$ | $b_1$ |

   $R2$:

| B | C |
|---|---|
| $b_1$ | $c_1$ |
| $b_1$ | $c_2$ |

   $R3$:

| A | D | E |
|---|---|---|
| $a_1$ | $d_1$ | $e_1$ |
| $a_2$ | $d_1$ | $e_1$ |

   After joining these three tables ($\Pi_{AB} \bowtie \Pi_{BC} \bowtie \Pi_{ADE}$) we get spurious rows (2nd and 3rd) and hence this join is not lossless.

| A | B | C | D | E |
|---|---|---|---|---|
| $a_1$ | $b_1$ | $c_1$ | $d_1$ | $e_1$ |
| $a_1$ | $b_1$ | $c_2$ | $d_1$ | $e_1$ |
| $a_2$ | $b_1$ | $c_1$ | $d_1$ | $e_1$ |
| $a_2$ | $b_1$ | $c_2$ | $d_1$ | $e_1$ |

5. Consider $R = ABCDEGHI$ and the following set $F$ of functional dependencies:

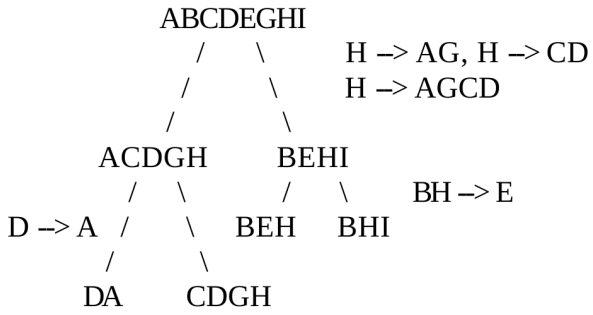
   H $\rightarrow$ AG  AH $\rightarrow$ CD
   D $\rightarrow$ A  AB $\rightarrow$ E

(a) Find a join loss-less and BCNF decomposition of $R$.

(b) Indicate whether your database schema is in dependency preserving with respect to $F$. Explain. [12]

**Solution:**

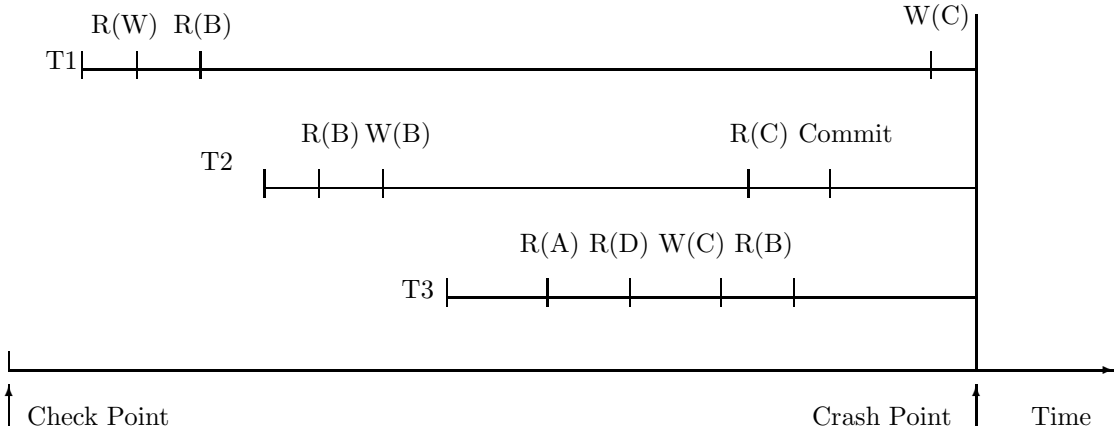
(a) Below is a join-less and BCNF decomposition of $R = ABCDEGHI$




That is, $\mathcal{D} = \{AD, CDGH, BEH, BHI\}$ is a join loss-less BCNF decomposition of $R$.

(b) The above decomposition is NOT dependency preserving because there exists no $F'$ of $FDs$ such that $F'$ is preserved by the database schema and $F'$ implies AB $\rightarrow$ E.

6. Consider the following log information that involves three transactions. Assume that the differed update protocol with check-pointing is used for crash recovery. And note that the log file does not necessarily record all operations of a transaction.




(a) Is the schedule, as shown in the log, recoverable? Why?

(b) If the schedule is recoverable, present the list of actions in the recovery procedure.

(c) If the schedule is NOT recoverable, what is your suggestion to make it recoverable? [12]

**Solution:**

(a) A schedule $S$ is recoverable if no transaction $T$ in $S$ commits until all transactions that have written an item that $T$ reads have committed. This schedule is NOT recoverable because $T3$ writes item $C$ without committing and $T2$ reads it afterwards and commits.

(c) $T2$ has to wait until $T3$ commits before committing.

7. Consider the following two join algorithms to compute $R_1 \bowtie R_2$:

   (a) nested loop approach (brute force),

   (b) using an access structure (index join).

   Present two scenarios $S_1$ and $S_2$ such that the number of page accesses using the brute force in $S_1$ is less than that using the index joint, and vise versa for $S_2$.

   For each scenario, one needs to specify all the relevant parameters such as the sizes of main memory, $R_1$, and $R_2$, and others. (Similar to the conditions given in Question 5 of Assignment 1.) [12]

   **Solution:**

   **Scenario** $S_1$: Main memory: 502 pages; $R_1$: 20000 pages with 200 rows per page; $R_2$: 50000 pages with 100 rows per page.

   Brute force:

   *case 1*: the number of page accesses for join $R_1 \bowtie R_2$

   $20000 + (20000/502) \times 50000 \simeq 2020000$

   *case 2*: the number of page accesses for join $R_2 \bowtie R_1$

   $50000 + (50000/502) \times 20000 \simeq 2050000$

   Index joint (each index search takes 1 disk accesses on average):

   *case 1*: index on the join attribute in $R_2$; Memory 500 pages for $R_1$, 1 page for $R_2$, 1 page for output. The number of page accesses:

   $20000 + 20000 \times 200 \times 1 \simeq 4020000$

   *case 2*: index on the join attribute in $R_1$; Memory 500 pages for $R_2$, 1 page for $R_1$, 1 page for output. The number of page accesses:

   $50000 + 50000 \times 100 \times 1 \simeq 5050000$

   **Scenario** $S_2$: Main memory: 502 pages; $R_1$: 20000 pages with 20 rows per page; $R_2$: 50000 pages with 5 rows per page.

   Brute force:

   *case 1*: the number of page accesses for join $R_1 \bowtie R_2$

   $20000 + (20000/502) \times 50000 \simeq 2020000$

   *case 2*: the number of page accesses for join $R_2 \bowtie R_1$

   $50000 + (50000/502) \times 20000 \simeq 2050000$

   Index joint (each index search takes 1 disk accesses on average):

   *case 1*: index on the join attribute in $R_2$; Memory 500 pages for $R_1$, 1 page for $R_2$, 1 page for output. The number of page accesses:

   $20000 + 20000 \times 20 \times 1 \simeq 420000$

   *case 2*: index on the join attribute in $R_1$; Memory 500 pages for $R_2$, 1 page for $R_1$, 1 page for output. The number of page accesses:

   $50000 + 50000 \times 5 \times 1 \simeq 300000$

8. Given the fact table below:

   **sales(market_id, item_id, date, sale)**

   and the following SQL statement:

```
SELECT market_id, item_id, date, SUM(sale)
FROM   sales
GROUP BY ROLLUP (market_id, item_id,date)
```

   Write a single SQL statements without using any GROUP BY CUBE/ROLLUP clauses such that it returns the same result table as the given one. [12]

   **Solution:**

```
CREATE GLOBAL TEMPORARY TABLE temp AS
SELECT market_id, item_id, date, SUM(sale)
FROM   sales
GROUP BY market_id, item_id, date;

SELECT market_id, item_id, NULL, SUM(sale)
From temp
GROUP BY market_id, item_id

UNION

SELECT market_id, NULL, NULL, SUM(sale)
From temp
GROUP BY market_id

UNION

SELECT NULL, NULL, NULL, SUM(sale)
From temp

UNION

temp;
```

9. Explain that, when evaluating possible association rules, the confidence is always larger than the support. [12]

   **Solution:** Assume the possible association rule is $X \Rightarrow Y$. Let N be the number of transactions, and $P(X)$, $P(X \cap Y)$ be the number of transactions that contains X, and both X and Y, respectively. Obvious, $P(X) \leq N$.

   By the definitions, the confidence is $\frac{P(X \cap Y)}{P(X)}$ while the support is $\frac{P(X \cap Y)}{N}$. Since $P(X) \leq N$, the confidence is always grater than or equal to the support.

   To have any sense of usefulness, it is reasonable to assume that the confidence is always greater than the support.