# McMaster University
# CS 3DB3 Fall 2017
# Assignment 3 Solution
# Due: Dec. 6, 2017 at 10:00pm

December 12, 2017

## I. Database Design (60 marks)

### Question 1 (35 marks)

The Department has a database containing information about all lectures during a term. The `Schedule` relation has the following schema:

`Schedule` $(C, D, T, R, P, A)$

where $C$ represents a course, $D$ for day (weekday), $T$ for time, $R$ for room, $P$ for professor, and $A$ for head TA. The set of functional dependencies $F$ defined over `Schedule` are:
$F = \{RDT \rightarrow P, RDT \rightarrow C, C \rightarrow A, PDT \rightarrow R, PDT \rightarrow C, CDT \rightarrow P, CDT \rightarrow R\}$

Answer the following questions:

(a) (4 marks) What do these functional dependencies mean? Describe them in words.

(b) (4 marks) Is the FD $RDT \rightarrow AP$ entailed by $F$? Explain and show your work with references to Armstrong's axioms.

(c) (6 marks) Find all the key(s) of relation `Schedule`. Show your work (i.e., how each key is derived).

(d) (4 marks) Is `Schedule` in BCNF? If not, decompose it into smaller relations that are each in BCNF. Again, show your work.

(e) (6 marks) We can create a new relation `ProfsSchedule`$(D, T, P)$ by projecting some attributes of `Schedule`. Are there new functional dependencies that hold over `ProfsSchedule`? If so, state these FDs. If not, state why not. In both cases, show your work to justify your answer.

(f) (11 marks) Find a minimal cover $F_{min}$ for $F$. Show all the steps in your derivation of $F_{min}$.

Answer:

(a)
- RDT → P: No two professors share the same room/day/time.
- RDT→ C: No two courses share the same room/day/time combination.
- C → A: There is no more than one head TA per course.
- PDT → R: No two rooms share the same professor/day/time combination.
- PDT → C: No two courses share the same professor/day/time combination.
- CDT → P: No two professors share the same course/day/time combination.
- CDT → R: No two rooms share the same course/day/time combination.

(b) Yes. RDT → A is entailed by RDT → C and C → A, by transitivity. Then the union rule says that RDT → A and RDT → P entail RDT → PA.

(c) D and T must be part of any key, since they do not occur on the right-hand side of any FDs. However (DT)+ = DT, so they need at least one more attribute to form a key. The triples RDT, PDT, and CDT all have attribute closure CDTRPA, and hence are keys. ADT has closure ADT, and hence gives no new possibilities. So the three keys are RDT, PDT, and CDT.

(d) C → A violates BCNF, so we decompose into R1 = (CA, {C → A}) and R2 = (CDTRP; {RDT → P, RDT → C, PDT → R, PDT → C, CDT → P, CDT → R}). R1 must be in BCNF because it has just two attributes, and R2 is in BCNF because all the FDs that project onto it have a left-hand side that is a superkey for R, and hence R2.

(e) By checking the attribute closures D+, T+, P+, (DT)+, (DP)+, and (TP)+, it turns out that the new relation has no non-trivial FDs projected onto it.

(f) Multiple solutions possible. One answer is outlined here:

1) Decompose RHS $F = \{RDT \rightarrow P, RDT \rightarrow C, C \rightarrow A, PDT \rightarrow R, PDT \rightarrow C, CDT \rightarrow P, CDT \rightarrow R\}$

2) For all $f$ in $F'$, test $J = (F' - f)^+$ equal to $F^+$

- $RDT \rightarrow P$ can be removed, since $\{R, D, T\}^+ = \{R, D, T, C, A, P\}$ $F = \{RDT \rightarrow C, C \rightarrow A, PDT \rightarrow R, PDT \rightarrow C, CDT \rightarrow P, CDT \rightarrow R\}$
- $RDT \rightarrow C$ can't be removed, since $\{R, D, T\}^+ = \{R, D, T\}$ $F = \{RDT \rightarrow C, C \rightarrow A, PDT \rightarrow R, PDT \rightarrow C, CDT \rightarrow P, CDT \rightarrow R\}$
- $C \rightarrow A$ can't be removed, since $\{C\}^+ = \{C\}$ $F = \{RDT \rightarrow C, C \rightarrow A, PDT \rightarrow R, PDT \rightarrow C, CDT \rightarrow P, CDT \rightarrow R\}$
- $PDT \rightarrow R$ can be removed, since $\{P, D, T\}^+ = \{P, D, T, C, A, R\}$ $F = \{RDT \rightarrow C, C \rightarrow A, PDT \rightarrow C, CDT \rightarrow P, CDT \rightarrow R\}$
- $PDT \rightarrow C$ can't be removed, since $\{P, D, T\}^+ = \{P, D, T\}$ $F = \{RDT \rightarrow C, C \rightarrow A, PDT \rightarrow C, CDT \rightarrow P, CDT \rightarrow R\}$
- $CDT \rightarrow P$ can't be removed, since $\{C, D, T\}^+ = \{C, D, T, A, R\}$ $F = \{RDT \rightarrow C, C \rightarrow A, PDT \rightarrow C, CDT \rightarrow P, CDT \rightarrow R\}$

- $CDT \rightarrow R$ can't be removed, since $\{C, D, T\}^+ = \{C, D, T, A, P\}$ $F = \{RDT \rightarrow C, C \rightarrow A, PDT \rightarrow C, CDT \rightarrow P, CDT \rightarrow R\}$

3) Check minimality of LHS.
Take $RDT \rightarrow C$ as an example. Let's try to reduce the LHS by removing each of $R, D, T$ in turn.

- $(RDT - R)^+ = (DT)^+ = DT$; can we achieve $C \in (DT)^+$ ? No, and hence, $R$ cannot be removed from the LHS.
- $(RDT - D)^+ = (RT)^+ = RT$; Again, $C$ is not in $(RT)^+$, and $D$ cannot be removed from the LHS.
- $(RDT - T)^+ = (RD)^+ = RD$; Again, $C$ is not in $(RD)^+$, and $T$ cannot be removed from the LHS.

Thus, we cannot reduce the LHS of $RDT \rightarrow C$.
Repeat these steps for the remaining FDs.

Hence, $F_{min} = \{RDT \rightarrow C, C \rightarrow A, PDT \rightarrow C, CDT \rightarrow P, CDT \rightarrow R\}$

## Question 2 (15 marks)

Consider the relation R(A,B,C,D,E), and the decomposition of R into R1(ABC) and R2(ADE).

(a) (4 marks) Give a set of functional dependencies (FDs) such that the decomposition into R1 and R2 is lossless join and dependency preserving. Justify why your FDs satisfy the criteria.
Answer: functional dependencies $A \rightarrow BC, CD \rightarrow E, B \rightarrow D, B \rightarrow A$. A is a key.

(b) (4 marks) Give a set of functional dependencies such that the decomposition into R1 and R2 is lossless join, but not dependency preserving. Justify why your FDs satisfy the criteria.
Answer: $F = \{A \rightarrow BC, B \rightarrow D\}$

$R1 \cap R2 = A$, $A \rightarrow BC$, we get $A \rightarrow ABC$ . Both R1 and R2 functionally determine all the attributes in R1. So, it's a lossless join. However, $B \rightarrow D$ is lost.

(c) (4 marks) Give a set of functional dependencies such that the decomposition into R1 and R2 is not lossless join, but dependency preserving. Justify why your FDs satisfy the criteria.
Answer: $F = \{A \rightarrow C, A \rightarrow E\}$

$R1 \cap R2 = A$, $A^+ = A$ . A is not a key for R1 and R2. Hence, the decomposition is not lossless.
$A \rightarrow C$ still holds over R1, and for R2 $A \rightarrow E$ still holds for dependency preservation.

(d) (3 marks) Is it true or false that every binary relation is in BCNF? If true, clearly justify and prove it. If false, give an example relation not in BCNF.
Answer: True.

A relation is in BCNF when all non-trivial FDs have superkey LHS.
For a binary relation R(A,B), we can only have 0, 1 or 2 non-trivial FDs.
For 0 FDs, an empty set, the relation is in BCNF by the trivial case.

For 1 FD, we get $A \rightarrow B$, or $B \rightarrow A$. In both cases, the LHS is a superkey.
For 2 FDs, $\{A \rightarrow B, B \rightarrow A\}$, A is a superkey and B is also a superkey.
Hence, every binary relation is in BCNF.

## Question 3 (10 marks)

Prove the following inference rules for functional dependencies using Armstrong's axioms (using only the axioms presented in class). Show the steps of your proof, and indicate which of Armstrong's axioms is applied in each step.

(a) (5 marks) If $X \rightarrow Y$ and $W \rightarrow Z$ then $XW \rightarrow YZ$.
Answer:
$XW \rightarrow YW$ augmentation with W
$XW \rightarrow W$ decomposition
$XW \rightarrow Y$ decomposition
$XW \rightarrow Z$ since W leads to Z given, transitivity
$XW \rightarrow YZ$ union

(b) (5 marks) Given the relational schema $R(A, B, C, D, E, F)$ and the FDs: $C \rightarrow D$, $BE \rightarrow A$, and $BEF \rightarrow C$. Show that BEF is a key.
Answer:
BE to A given
BEF to AF aug F
BEF to C given
BEF to ACF union
BEF to ACDF transitivity with C to D
BEF to ABCDEF reflexive with B, E

# II. Transactions and Concurrency (20 marks)

## Question 4 (8 marks)

a) (3 marks) Give an example of a conflict-serializable schedule that is not a serial schedule.
Answer:
One example: R2(A); R1(B); W2(A); R3(A); W1(B); W3(A); R2(B); W2(B)

b) (5 marks) Consider the following locking protocol: Before a transaction T writes a data object A, T has to obtain an exclusive lock on A. For a transaction T, we hold these exclusive locks until the end of the transaction. If a transaction T reads a data object A, no lock on A is obtained. State which of the following properties are ensured by this locking protocol: serializability, conflict-serializability, recoverability, avoids cascading aborts, avoids deadlock. Justify your answer for each property.

Answer:
- serializability: No
- CS: No, conflicts with reads and writes can occur with no shared lock requests needed.

- ACA: No. Since T2 can read A without a shared lock, it can do so while T1 holds an X lock and is still in progress (and has not yet committed).
- recoverable: No (similar reasoning to ACA)
- avoids deadlock: no (deadlocks can occur with X lock requests)

## Question 5 (12 marks)

In this question, you will execute a few concurrent transactions against a banking table. The `accounts` table is shown in Table 1. Create the `accounts` table in DB2. You may assume `username` is of type char(10), `name` is char(20), and `balance` is of type decimal(10,2). Write the appropriate INSERT INTO statements to load the five records shown into your `accounts` table.

Table 1: `accounts`

| username | name | balance |
|---|---|---|
| luke | Skywalker | 8000.00 |
| kylo | Ren | 12500.00 |
| rey | Rey | 5000.00 |
| finn | Finn | 1500.00 |
| leia | Skywalker | 2000.00 |

### Setup

You will simulate concurrent database queries, using two different database connections, against the `accounts` table. You will open two terminal sessions, and execute the following commands to prepare your environments. We will call the two sessions, A and B.

**In Session A**

1. Invoke db2 using `db2 +c`. This command turns off the AUTO COMMIT feature in DB2. You will now be at the DB2 command line.
2. Verify AUTO COMMIT is turned OFF by running the command `list command options`. (You should see "Auto-Commit OFF" in the second entry of the table.)
3. Change the transaction isolation level to 'Read Stability' by running the command `change isolation to rs`. Transaction isolation levels define the degree of access and interaction among a set of concurrent transactions which operate against the same data. Please see the DB2 reference for more information on isolation levels in DB2.
4. Connect to your database.

**In Session B**

1. Invoke DB2 by typing `db2` (do not turn AUTO COMMIT off).
2. Run `change isolation to rs`
3. Connect to your database.

**Transactions**

Run the following commands (in the given order) and provide your answers to the stated questions.

1. In Session A, insert the record `('snoke', 'First Order', 20000.00)`, followed by a `select * from accounts`.
2. In Session B, run `select * from accounts`. Is the output you get the same or different than in (1)? Why did this occur? What is a possible solution?
3. Return to Session A, and implement your solution from step #2.
4. In Session B, do the SELECT * query again (to list all records). Provide your output.
5. We will update accounts from two different transactions. In Session A, change the isolation level to cursor stability (CS). In Session B, change the isolation level to uncommitted read (UR). (You will have to disconnect and reconnect to the database.)
6. In Session A, transfer $5000 from Kylo's account into Rey's account.
7. Then, in Session B, list all accounts and their balances. Then issue a $10000 transfer from Snoke's account to Finn's account. What happens and why? Explain the interaction between the two sessions.
8. Commit the transaction in Session A. Report the latest balances.
9. Change the transaction isolation level in Session A to 'Uncommitted Read'.
10. Now, in Session A, transfer 50% of Kylo's account balance to Leia's account. Commit the transaction.
11. In Session A, transfer all of Snoke's funds to Luke's account. In session B, list all data records. What is Luke's balance? Does it reflect the latest transfer from Snoke? Based on your transaction in step (7) and this step, what can you say about the allowed actions in CS and UR isolation levels? How do these compare with the RS isolation level?
12. Abort the Snoke to Luke transfer transaction in Session A, by executing the command `rollback`.
13. In session A and B, list all data records. What are the final balances?

Answer:

- Session in B hangs because Snoke's uncommitted record cannot be viewed until after a COMMIT from session A.
- 

```
NAME                 BALANCE
-------------------- ------------
luke                      8000.00
kylo                     12500.00
rey                       5000.00
finn                      1500.00
leia                      2000.00
snoke                    20000.00
```

- In Session B, after transfer $5000 transfer from Kylo to Rey:

```
NAME                 BALANCE
-------------------- ------------
luke                      8000.00
kylo                      7500.00
rey                      10000.00
finn                      1500.00
leia                      2000.00
snoke                    20000.00
```

- The transfer from Snoke's account of $10,000 to Finn's account hangs in session B, because we cannot issue an update against the same table, only reads are allowed under the CS option in Session A, not writes.

- Balances are:

```
NAME                 BALANCE
-------------------- ------------
luke                      8000.00
kylo                      7500.00
rey                      10000.00
finn                     11500.00
leia                      2000.00
snoke                    10000.00
```

- Luke's balance is 18,000, and does reflect the latest Snoke transfer. In both cases, uncommitted reads are allowed in CS and UR isolation levels. However, updates are not allowed in session B if there is a pending transaction from session A.

```
NAME                 BALANCE
-------------------- ------------
luke                     18000.00
kylo                      3750.00
rey                      10000.00
finn                     11500.00
leia                      5750.00
snoke                        0.00
```

- Final balances are:

```
NAME                 BALANCE
-------------------- ------------
luke                      8000.00
kylo                      3750.00
rey                      10000.00
finn                     11500.00
leia                      5750.00
snoke                    10000.00
```

# Grading

This assignment is worth 14% towards your final grade.

# Submission

All files are to be submitted using the Avenue system. Please ensure your answers are typed and clearly readable. Include your name and student ID number in the file. Submit your solutions to all questions in a file called **`asg3.pdf`**.