**Zhe Lin**

**001422116**

**Qestion1:**

1. What do these functional dependences mean? Describe them in words.

**Answer**: It means that there is a relationship or “function” exist, which can determine one attribute by another attributes.

1. Is the FD RDT → AP entailed by F? Explain and show your work with references to Armstrong’s axioms

**Answer:** we know F={RDT → P, RDT → C, C → A}

By decomposition: RDT → AP, then RDT → A, RDT → P

By transition: RDT → C, C → A, RDT → A

By Union: RDT → AP

1. Find all key(s) of relation Schedule. Show your work(i.e., how each key is derived)

**Answer:** RDT, PDT, CDT

RDT → P, RDT → C, C → A

By transitivity: RDT → A

By union: RDT → PCA

RDT+ = RDTPCA = F

PDT → R, PDT → C, C → A

By transitivity: PDT → A

By union: PDT → RCA

PDT+ = PDTRCA = F

CDT→P, CDT → R

By union: CDT → CDTPR

By decomposition: CDT → C

C → A

By transitivity: CDT → A

By union: CDT → CDTPRA

1. Is schedule in BCNF? If not, decompose it into smaller relations that are each in BCNF. Again, show your work.

**Answer:** a relation is said in BCNF, if and only if each function dependency X → Y has a determinant which is a Superkey, which means it determines all the other attributes of Schedule.

C+ = CA violate BCNF

RDT+ = RDTPCA does not violate BCNF

PDT+ = PDTRCA does not violate BCNF

CDT+ = CDTPRA does not violate BCNF

--- (CA) (DTPR): both of them are BCNF

1. We can create a new relation ProfsSchedule(D,T,P) by projecting some attributes of Schedule. Are there new function 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.

**Answer:**

There are not new function dependencies that hold over ProfsSchedule. There are

some possible situations for day, time and professors. For example, if different professor teaching different classes in same day and same time. Then we have dependency function on (D,T,P), its unable to find professor by day and time.

Therefore, we create whatever function dependencies, it will happen have the same tuples which conflict function dependencies. Consequently, there are not new function dependencies that hold over ProfsSchedule.

1. Find a minimal cover Fmin for F. Show all the steps in your derivation of Fmin.

**Answer:**

1st all RHS in single attribute, it is fine

H = {RDT → P, RDT → C, C → A, PDT → R, PDT → C, CDT → P, CDT → R}

2nd

J = H – {RDT → P} = { RDT → C, C → A, PDT → R, PDT → C, CDT → P, CDT → R}, RDT+ = {RDTCAPR}, it can be removed

J = H – {RDT → C} = {C → A, PDT → R, PDT → C, CDT → P, CDT → R}, RDT+ = {RDT}, it cannot be removed.

J = H – {C → A} C+ = {C}, it cannot be removed

J = H – {PDT → R} = {RDT → C, C → A, PDT → C, CDT → P, CDT → R}, PDT+ = {PDTCAR}, it can be removed

J = H – {PDT → C} = { RDT → P, RDT → C, C → A, PDT → R, CDT → P, CDT → R}, PDT+ = {PDTRCA}, it can be removed

J = H – {PDT → R} = {RDT → C, C → A, CDT → P, CDT → R}, PDT+ = {PDT}, it cannot be removed

J = H – { CDT → P } = { RDT → C, C → A, PDT → C, CDT → R}, CDT+ = {CDTRCA}, it cannot be removed

J = H – { CDT → R } = { RDT → C, C → A, PDT → C, CDT → P} , CDT+ = {CDTRCA}, it can be removed

After step2 H = { RDT → C, C → A, PDT → R, CDT → P}

3rd expand the LHS of the FDs:

C → A: which is single attribute in LHS, it is fine.

RDT → C: R+=R, D+=D, T+=T

PDT → R: P+=P, D+=D, T+=T

CDT → P: C+=CA, D+=D, T+=T

Nothing can be removed

4th

-H doesn’t change

H={RDT → C, C → A, PDT → R, CDT → P}

**Question2:**

1. Give a set of function dependencies(FDs) such that decomposition into R1 and R2 is lossless join and dependency preserving. Justify why your FDs satisfy the criteria.

**Answer:**

F = {A → BC, D → EA}

R1 ∩ R2 = A and we have A → BC = R1

It is decomposition lossless.

F1 = {A → BC}, F2 = {D → EA}

F1 ∪ F2 = {A → BC, D → EA} = F

it is dependency preserving

1. Give a set of function dependencies(FDs) such that the decomposition into R1 and R2, but not dependency preserving.

**Answer:**

F = {A → BC, B → D, E → A}

R1 ∩ R2 = A and we have A → BC = R1

it is decomposition lossless

F1 = {A → BC}, F2 = {B → D, E → A}

F’ = F1 ∪ F2 = {A → BC, B → D, E → A}

It is not dependency preserving

1. Give a set of function 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 = {B → AC, E → AD}

R1 ∩ R2 = A and we do not have such A -> = R1 or R2

It is not decomposition lossless

F1 = {B → AC} F2 = {E → AD}

F’ = F1 ∪ F2 = {B → AC, E → AD} = F

It is dependency preserving

1. Is it True or False that every binary relation is in BCNF? If True, clearly justify and prove it. If false, given an example relation not in BCNF.

**Answer:** True

Suppose we have a binary relation R(A, B). Between A and B, it may has A → B, A is the key. B → A, B is the key, and both of them are keys. Either of them,

It satisfied BCNF and not violates it.

**Question3:**

1. X → Y and W → Z then XW → YZ:

Augmentation: XW → WY

Decomposition: XW → W and XW → Y

Transitivity: XW → W, W → Z, XW → Z

Union: XW → Y and XW → Z, then XW → YZ

1. C → D, BE → A, BEF → C

Transitivity: BEF → C, C → D, BEF → D

Union, BEF → C, BEF → D, BEF → CD

Augmentation: BE → A, BEF → AF

Union: BEF → AF, BEF → CD, BEF → AFCD,

Delete trivial FDs: BEF → AFCD, BEF → ACD(key)

**Question 4:**

1. Given an example of a conflict-serializable schedule that is not a serial schedule.

**Answer:** W1(A), R2(A), R1(A)

1. Serializability:

If T write on A, it obtains a lock on A. After it finishes writing on A, it will not allow others to read until it release block. Therefore, transaction can be serializability.

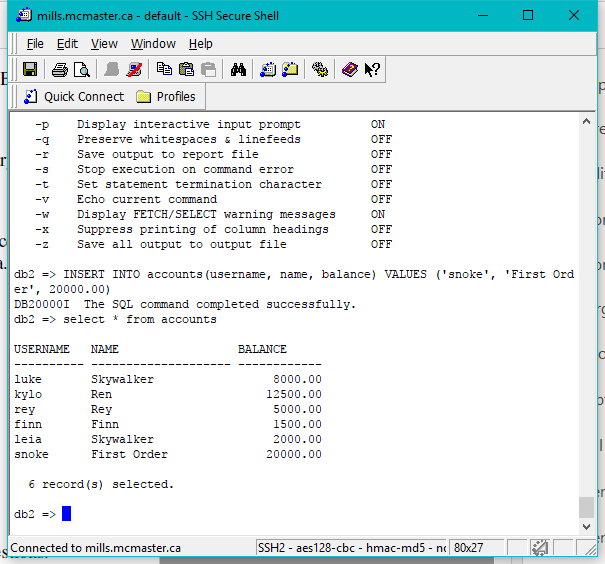
Recoverable:

If T starts its transaction, others only allowed to read or write until T commit. So there is nothing about others’ transaction before T commits.

**Question5:**

1. INSERT INTO accounts(username, name, balance) VALUES (‘snoke’, ‘First Order’, 20000.00)

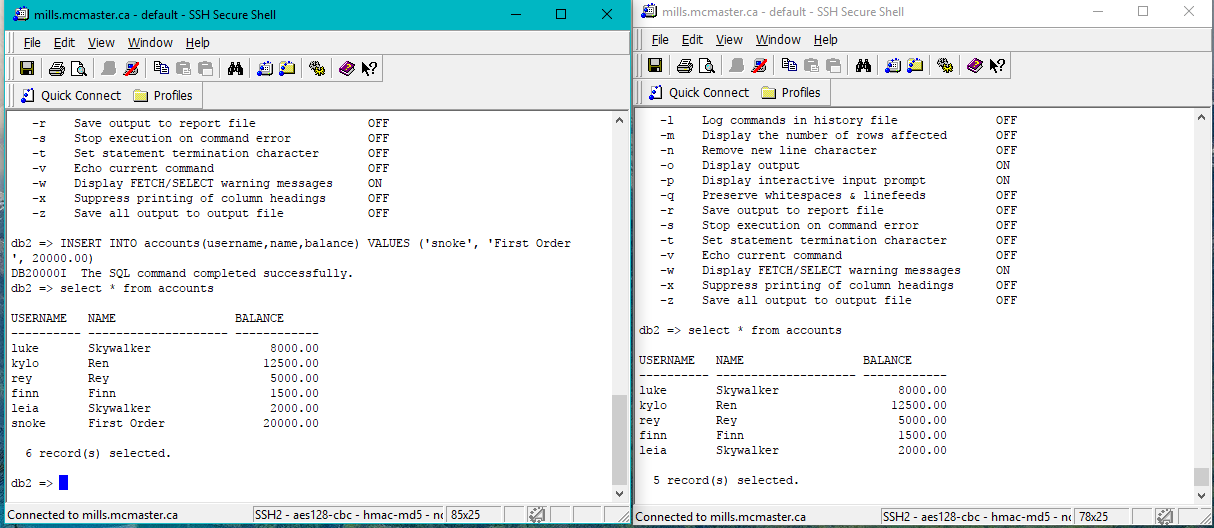
Select \* from account



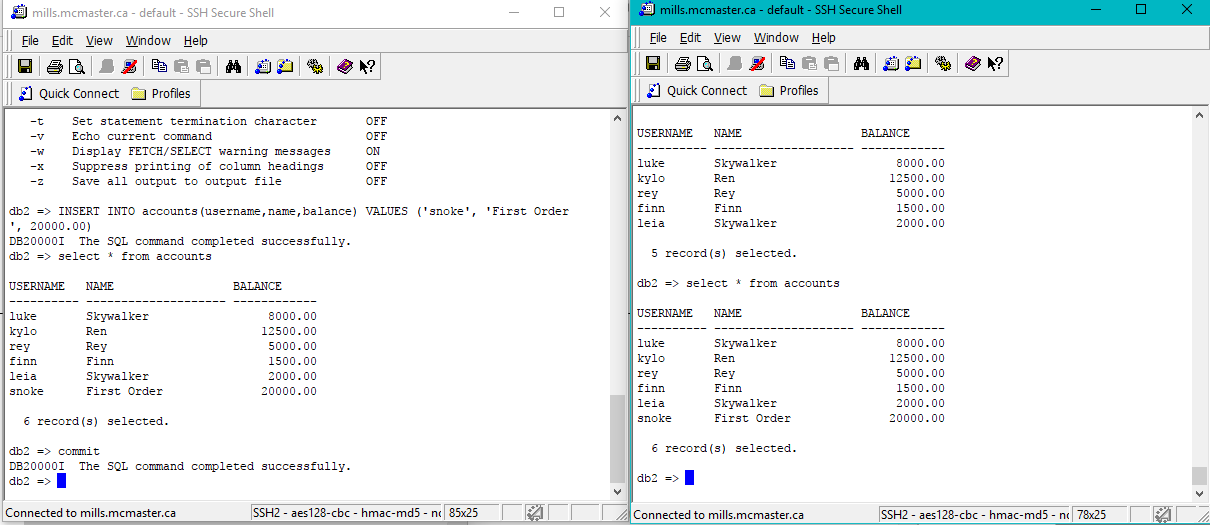
1. The output get different than in (1).

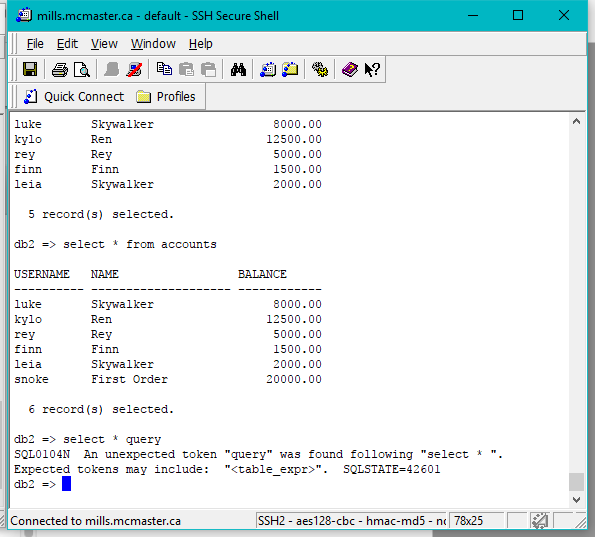
In Session, we turn off the auto-commit, so it has a lock and not commit. So B was denied for reading.

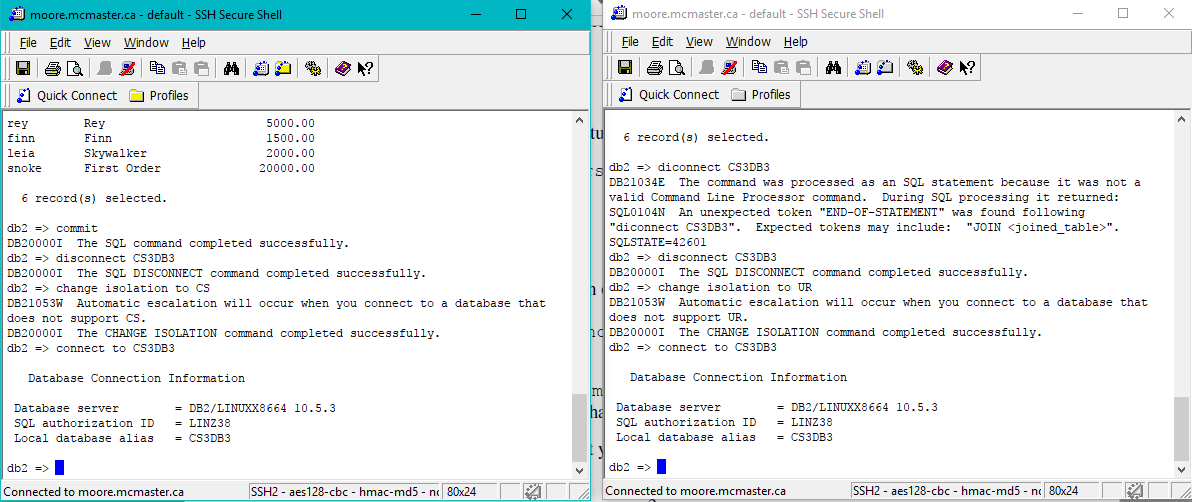
Possible solution: commit A

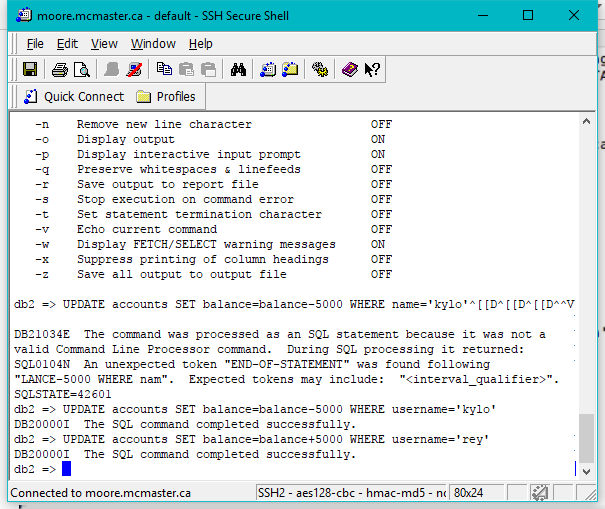


1. db2=>commit



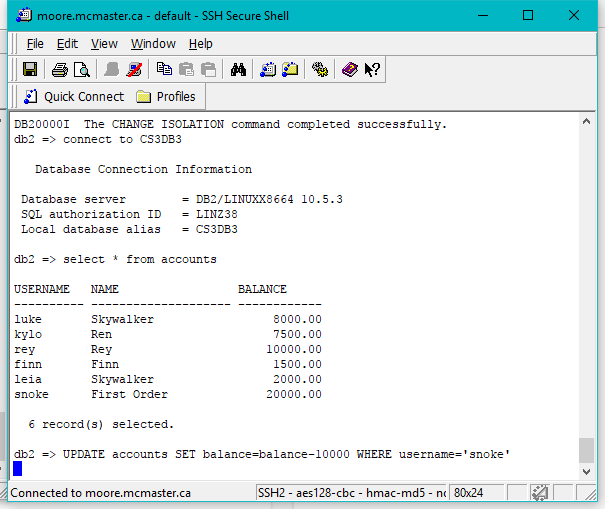


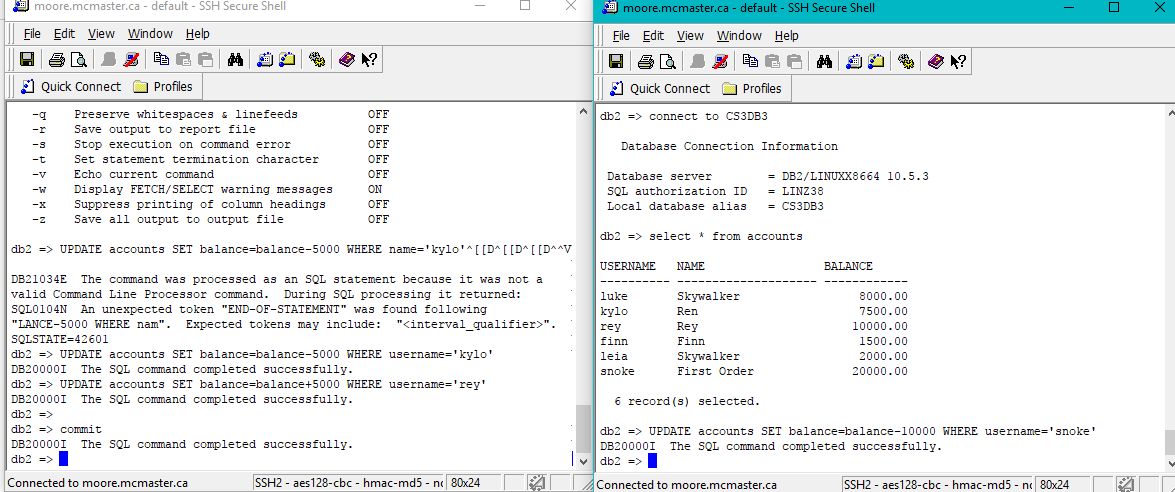


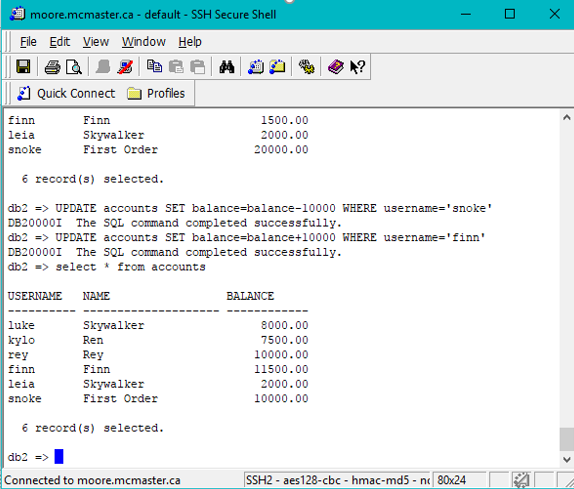


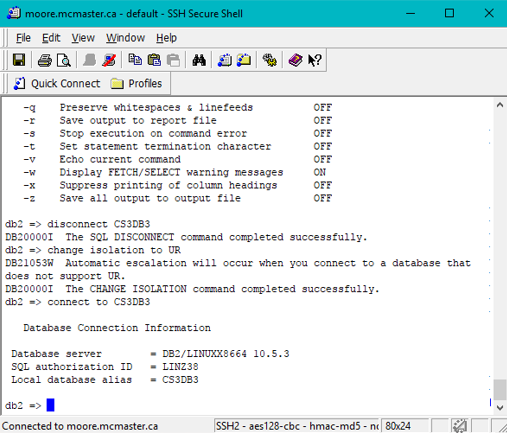
1. Not runnable

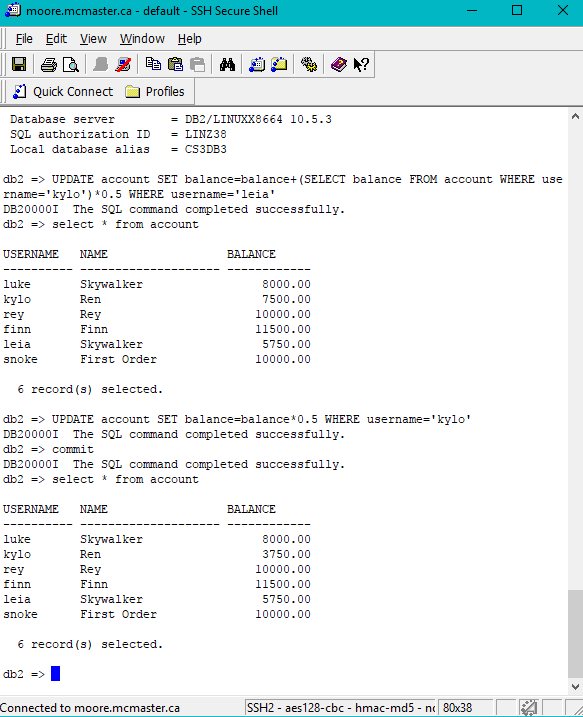
Which is block by Session A









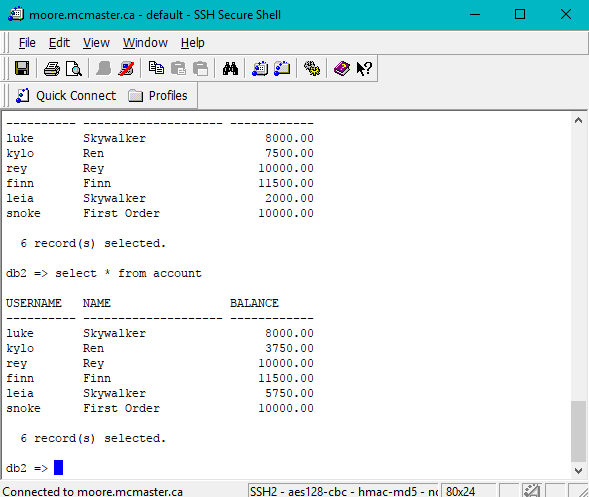


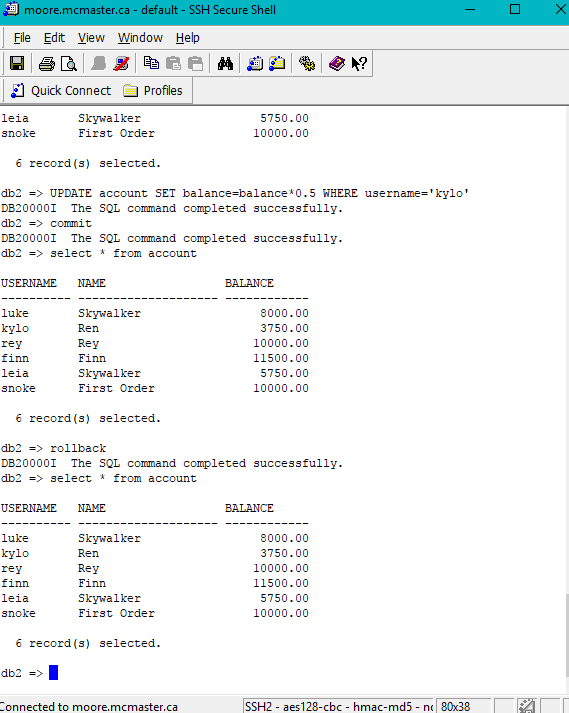
1. $8000, yes it reflects the latest transfer from snoke.

In CS and UR isolation level, which allow read uncommit.

B cannot update before A’s commit.

RS cannot read uncommit





1. **They are same!**

