CSC343 Lab5

Yunkai Fan 1005088584

Tianshu Ni 1005682673

Apr. 5, 2022

Q1) Database Design

Schedule (course, professor, teaching assistant, location, day, time)

1. Functional Dependencies
a) {professor, day, time } $\longrightarrow$ course
b) {course } $\longrightarrow$ teaching assistant
c) {location, day, time } $\longrightarrow$ course
d) {location, day, time } $\longrightarrow$ professor
e) {course, day, time } $\longrightarrow$ professor

2. We will be designing a 3NF scheme for the Schedule Relation. For convinence, we will be using letters A to F to represent attributes in Schedule, where A=course, B=professor, etc. And the FDs will become:

BEF $\rightarrow$ A;
A $\rightarrow$ C;
DEF $\rightarrow$ A;
DEF $\rightarrow$ B;
AEF $\rightarrow$ B.

Note the FDs above are already in singleton. We can observe that AEF,BEF,DEF are the candidate keys, and notice that none of the above FDs can be reduced. Now we can look for redundant FDs to eliminate.
BEF $\rightarrow$ A will keep;
A $\rightarrow$ C will keep as there's no way to get C without this FD;
DEF $\rightarrow$ A will be discarded;
DEF $\rightarrow$ B will keep;
AEF $\rightarrow$ B will keep as AEF^+ = ACEF.

The following is the result FDs in a minimal basis:
BEF $\rightarrow$ A;
A $\rightarrow$ C;

DEF $\rightarrow$ B;
AEF $\rightarrow$ B.

Now we will use the 3NF synthesis algorithm to decompose the relation. We have:
R_1(A,B,E,F)
R_2(A,C)
R_3(B,D,E,F)
R_4(A,B,E,F)
We now remove the redundant relation schemas, and the following is the result of Scheudle in 3NF:

R_1(A,B,E,F) with FD (A $\rightarrow$ C)
R_2(A,C) with FD (DEF $\rightarrow$ B)
R_3(B,D,E,F) with FD (BEF $\rightarrow$ A; AEF $\rightarrow$ B)

Q2) Database Design

a) C $\rightarrow$ ACG $\Rightarrow$ C $\rightarrow$ A and C $\rightarrow$ C and C $\rightarrow$ G by decomposition;
C $\rightarrow$ AFH $\Rightarrow$ C $\rightarrow$ A and C $\rightarrow$ F and C $\rightarrow$ H by decomposition;
C $\rightarrow$ H and H $\rightarrow$ EA $\Rightarrow$ C $\rightarrow$ EA by transitivity;
C $\rightarrow$ EA $\Rightarrow$ C $\rightarrow$ E and C $\rightarrow$ A by decomposition;
H $\rightarrow$ EA $\Rightarrow$ H $\rightarrow$ EAH = AEH as it contains the trivial FD, i.e. reflexivity;
C $\rightarrow$ H and H $\rightarrow$ AEH and AEH $\rightarrow$ BD $\Rightarrow$ C $\rightarrow$ BD by transitivity;
C $\rightarrow$ BD $\Rightarrow$ C $\rightarrow$ B and C $\rightarrow$ D by decomposition. Hence C $\rightarrow$ R.
AEH $\rightarrow$ BD $\Rightarrow$ AEH $\rightarrow$ BDE as it contains the trivial FD, i.e. reflexivity;
AEH $\rightarrow$ BDE $\Rightarrow$ AEH $\rightarrow$ B and AEH $\rightarrow$ DE by decomposition;
H $\rightarrow$ AEH and AEH $\rightarrow$ DE and DE $\rightarrow$ R $\Rightarrow$ H $\rightarrow$ R.
DF $\rightarrow$ AC $\Rightarrow$ DF $\rightarrow$ A and DF $\rightarrow$ C by decomposition;
DF $\rightarrow$ C and C $\rightarrow$ R $\Rightarrow$ DF $\rightarrow$ R by transitivity.
E $\rightarrow$ F $\Rightarrow$ DE $\rightarrow$ DF by augmentation;
DE $\rightarrow$ DF and DF $\rightarrow$ R $\Rightarrow$ DE $\rightarrow$ R by transitivity.
A $\rightarrow$ B $\Rightarrow$ AD $\rightarrow$ BD by augmentation;
AD $\rightarrow$ BD $\Rightarrow$ AD $\rightarrow$ ABD as it contains the trivial FD, i.e. reflexivity;
ABD $\rightarrow$ FGH $\Rightarrow$ ABD $\rightarrow$ F and ABD $\rightarrow$ G and ABD $\rightarrow$ H by decomposition;
AD $\rightarrow$ ABD and ABD $\rightarrow$ H and H $\rightarrow$ R $\Rightarrow$ AD $\rightarrow$ R by transitivity.

Therefore, the candidate keys are C, H, DF, DE, AD.

b) i. True. The 3NF decomposition will result in the same candidate keys found in a). As the rough work indicated below, in the process of the decomposition we will convert the FDs with more than 1 attributes on the RHS first and then reduce the LHS using possible keys. This process is simply 'translating' the original FDs, i.e. we are operating on the same sets at all, so this will not affect the way we find candidate keys as we follow the exactly the same FDs as in part a) and even borrowing the results from a).

ii. False. There can be more than one 3NF decomposition. As we derive the minimal basis at first, the order we chose to eliminate the redundant FDs will affect the resulting decomposed relations. As a counter-example we can have 2 different decomposition of relations, as indicated below.

Q2b Rough work

Finding minimal basis

Original:
A -> B
ABD -> FGH
AEH -> BD
BC -> EH
C -> ACG
C -> AFH
DE -> HB
DF -> AC
E -> F
H -> EA

Singletons of RHS and reducing the LHS:
A -> B
ABD -> F Can be reduced to AD -> F
ABD -> G Can be reduced to AD -> G
ABD -> H Can be reduced to AD -> H
AEH -> B Can be reduced to H -> B
AEH -> D Can be reduced to H -> D
BC -> E Can be reduced to C -> E
BC -> H Can be reduced to C -> H
C -> A
C -> C
C -> G
C -> A
C -> F
C -> H
DE -> H
DE -> B
DF -> A
DF -> C
E -> F
H -> E
H -> A

Result:
1. A -> B
2. AD -> F
3. AD -> G
4. AD -> H
5. C -> A
6. C -> E
7. C -> F
8. C -> H
9. C -> G
10. DE -> B
11. DE -> H
12. DF -> A
13. DF -> C
14. E -> F
15. H -> A
16. H -> B
17. H -> D
18. H -> E

Remove Redundancy:
(1) By the order of 9, 6, 5, 8, 7, 1, 14, 10, 11, 12, 13, 2, 3, 4, 16, 17, 18, 15 we discarded and kept some FDs, with the final simplification of A -> B, AD -> G, AD -> H, H -> D, H -> E, H -> A, C -> H, DE -> H, DF -> C, E -> F. By merging the RHS and apply the 3NF decomposition algorithm, we can get a relation for each FD, which are the results, i.e. R1(A, B), R2(A, D, G, H), R3(H, A, D, E), R4(C, H), R5(D, E, H), R6(D, F, C), R7(E, F).

(2) By the order of 1 - 18 we discarded and kept some FDs, with the final simplification of A -> B, AD -> H, C -> H, C -> G, DE -> H, DF -> C, E -> F, H -> A, H -> D, H -> E. By merging the RHS and apply the 3NF decomposition algorithm, we can get a relation for each FD, which are the results, i.e. R1(A, B), R2(A, D, H), R3(C, H, G), R4(D, E, H), R5(D, F, C), R6(E, F), R7(H, A, D, E).

Therefore (1)(2) are distinct decomposition of relations.

Q3) Transactions and Concurrency

QuestionProperty	Serializable	Conflict-Serializabile	View-Serializabile	Recoverable	A.C.A.	Strict
1.	?	X	?	?	?	X
2.	✓	✓	✓	X	X	X
3.	✓	✓	✓	X	X	X
4.	✓	✓	✓	✓	✓	✓
5.	X	X	X	✓	✓	✓

For Q1, it is undetermined for serializable since it does not have certain committed or aborted transactions. If they both T1 and T2 aborted then it is not serializable; if they both comitted then it can be serializable. Notice that the transactions are incomplete as there are no commit or abort on either transaction. There's a cycle between T1 and T2, thus it is not conflict serializable. We cannot determine the view-serializable as well because if they are aborted then it is not; if they commited then it is. We cannot determine recoverable and A.C.A as well since if we have T1: commit, T2: commit then we have ACA as well as recoverable; if we have T1: abort, T2: commit then we have neither recoverable nor A.C.A. The Scheule is also not strict as T2 reads A before T1 commits/abort.

For Q2 it is serializable because of the aborted transaction T2 does not affect T1 as the read in T1 is before the write in T2. Furthermore, as the conflict-serializable ignores the aborted transacitons, this Schedule is also conflict serializable, which also implies view-serializable. It is not recoverable since T2 aborts before T1 commits, which implies it is not A.C.A. The Schedule is also not strict as T1 overwrites A before T1 abort.

For Q3 it is serializable because of the aborted transaction T2 does not affect T1 as the write in T1 is before the read in T2. Furthermore, as the conflict-serializable ignores the aborted transacitons, this Schedule is also conflict serializable, which also

implies view-serializable. It is not recoverable since T2 aborts before T1 commits, which implies it is not A.C.A. The Scheule is also not strict as T2 reads A before T1 commits.

For Q4, there is no cycle within the Schedule, thus it is conflict serializable, which implies view-serializable and serializable. As A.C.A requires only read data from commited transactions, the schedule follows it. The Schedule is strict as T2 reads B after T1 commits, and strict implies recoverable and A.C.A.

For Q5, it is not serializable since the ordering of operations is not consistent with the serial schedule, i.e. T1: W(A), T1: W(A), T1: Commit, T2: W(A), T2: Commit, T3: R(A), T3: Commit. There's a cycle between T1 and T2, thus it is not conflict serializable. It is not view-serializable since it W2(A) then W1(A) but in the serial is W1(A) then W2(A). The Schedule is strict as T1 writes A after T2 write A commits, and strict implies recoverable and A.C.A.