

CS 448 Spring 2024 Homework 4

Sean Rettig

April 10, 2024

1

Consider the universal relation $R = \{A, B, C, D, E, F, G, H, I, J\}$ and the set of functional dependencies F .

$F =$

$\{\{A, B\} \rightarrow \{C, D\},$
 $\{D\} \rightarrow \{E, F, G\},$
 $\{F, G\} \rightarrow \{H\},$
 $\{A\} \rightarrow \{I\},$
 $\{A, B\} \rightarrow \{E, G\},$
 $\{A, I\} \rightarrow \{I, J\}\}$

- (a) **What is the closure of the set $\{A, D\}$?**

$\{A, D\}^+ = \{A, D, E, F, G, H, I, J\}$

- (b) **What is the key for R ?**

Key = $\{A, B\}$, since $\{A, B\}$'s closure includes all attributes.

- (c) **What is the minimum cover for the set of functional dependencies F ?**

Convert to canonical form:

$\{\{A, B\} \rightarrow \{C\},$
 $\{A, B\} \rightarrow \{D\},$
 $\{D\} \rightarrow \{E\},$
 $\{D\} \rightarrow \{F\},$
 $\{D\} \rightarrow \{G\},$
 $\{F, G\} \rightarrow \{H\},$
 $\{A\} \rightarrow \{I\},$
 $\{A, B\} \rightarrow \{E\},$
 $\{A, B\} \rightarrow \{G\},$
 $\{A, I\} \rightarrow \{I\}$
 $\{A, I\} \rightarrow \{J\}\}$

Eliminate extraneous stuff:

$\{\{A, B\} \rightarrow \{C\},$
 $\{A, B\} \rightarrow \{D\},$
 $\{D\} \rightarrow \{E\},$
 $\{D\} \rightarrow \{F\},$

$\{D\} \rightarrow \{G\},$
 $\{F, G\} \rightarrow \{H\},$
 $\{A\} \rightarrow \{I\},$
 $\{A, B\} \rightarrow \{E\},$
 $\{A, B\} \rightarrow \{G\},$
 $\{A\} \rightarrow \{J\}$

Eliminate redundancies:

$\{\{A, B\} \rightarrow \{C\},$
 $\{A, B\} \rightarrow \{D\},$
 $\{D\} \rightarrow \{E\},$
 $\{D\} \rightarrow \{F\},$
 $\{D\} \rightarrow \{G\},$
 $\{F, G\} \rightarrow \{H\},$
 $\{A\} \rightarrow \{I\},$
 $\{A\} \rightarrow \{J\}\}$

(d) **Decompose R into 2NF.**

$R = \{A, B, C, D, E, F, G, H, I, J\}$

Split up:

$R_1 = \{\underline{A}, \underline{B}, C, D, E, F, G, H\}; R_2 = \{\underline{A}, I, J\}$

This is because every nonprime element in R_1 is fully functionally dependent on $\{A, B\}$, and every nonprime element in R_2 is fully functionally dependent on $\{A\}$.

(e) Decompose R into 3NF

$R_1 = \{\underline{A}, \underline{B}, C, D\}; R_2 = \{\underline{A}, I, J\}; R_3 = \{\underline{D}, E, F, G\} R_4 = \{\underline{F}, \underline{G}, H\}$

2

- (a) **What are the possible sources of the information that defines the functional dependencies that hold among the attributes of a relation schema? Discuss two possible sources. Explain your answer.**

Functional dependencies must be defined explicitly by someone who knows the semantics of the attributes in the relation. So what are some possible sources for these semantics?

A possible source could be the entity that wants to use the database. Lets say its a company that want's to keep track of laptops. Perhaps they have a bunch of attributes that they want to be able to keep track of about the laptops. They can put together the attributes and then make some sort of ID for each laptop that has functional dependencies for the relation.

Another possible source could be some sort of relationship that is already defined in the nature of whatever you are already trying to model. For example, the relationship between a child and their mother. There is a functional dependency from the child to the mother because the child can only have one biological mother while the mother could have multiple children.

- (b) **What is meant by the closure of a set of functional dependencies? Illustrate with an example.**

The closure of a set of FD really just means that it includes all inferences that can be made from a sparser set of functional dependencies. It's the most explicit you can be about a group of relations.

For example, if there is a functional dependency between my SSN and me ($A \rightarrow B$), and also a functional dependency between me and my mother ($B \rightarrow C$), the closure of the set of those two would also include the functional dependency that can be inferred between my ssn and my mother. ($A \rightarrow C$). Thus, the closure of the SSN would be $A^+ = \{A, B, C\}$.

- (c) **What is the lossless (or nonadditive) join property of a decomposition? Why is it important?**

This property ensures that spurious tuples, or tuples that were not in the original relation before decomposition, do not appear after rejoining the relations.

This is important so we can complete the normalization of the data and relations without losing information. We need the benefits that come with normalizing relations, but we cannot sacrifice data to achieve it.

3

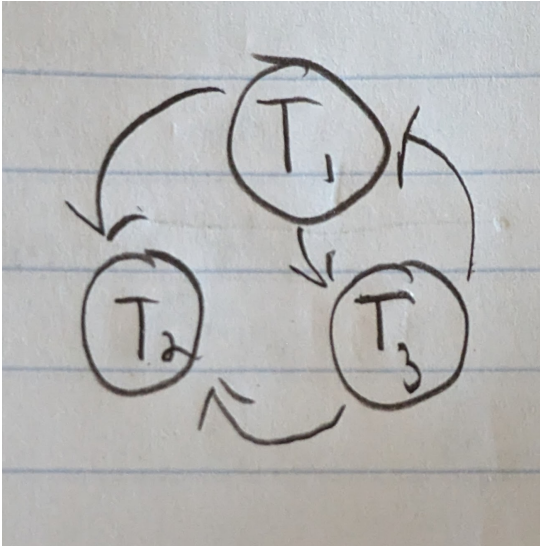
Consider the following four schedules for transactions T_1 , T_2 and T_3 :

- a $r_1(X); r_3(X); w_1(X); r_2(X); w_3(X); w_2(X)$
- b $r_1(Y); r_3(Y); w_3(X); w_1(X); r_2(X); w_2(Y)$
- c $w_2(X); r_1(Y); w_2(Y); r_2(X); w_2(Y); r_3(Y)$
- d $r_3(X); r_2(X); w_3(X); r_1(Y); w_1(X); r_2(Y)$
- e $r_3(X); r_2(X); r_1(X); w_3(X); w_1(X); w_2(X)$

- (a) For each schedule, specify whether or not the schedule is conflict serializable and why. To determine if a schedule is conflict serializable you can either use the precedence graph or swap conflicting operations.
- (b) Then, for each serializable schedule, give the equivalent serial schedule.

- (a) **$r_1(X); r_3(X); w_1(X); r_2(X); w_3(X); w_2(X)$**

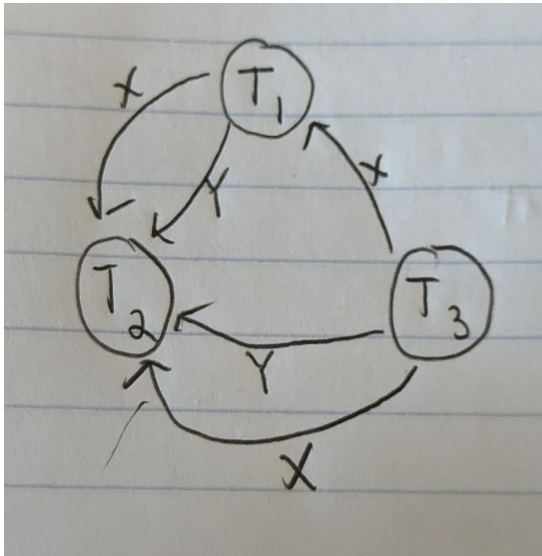
Not-Serializable. There is a cycle between T_1 and T_3 in the precedence graph.



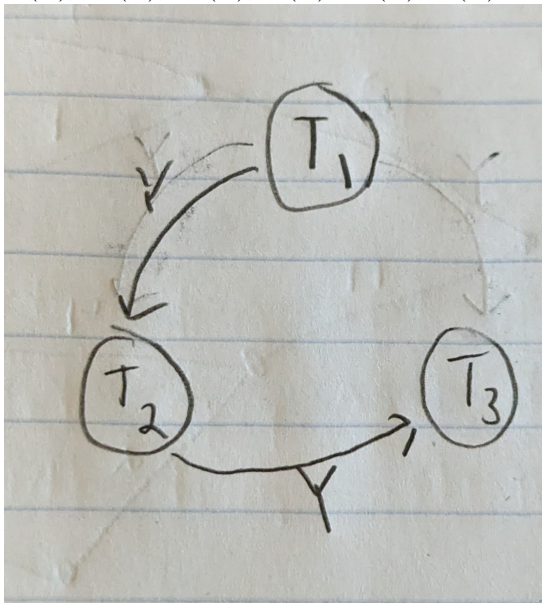
- (b) **$r_1(Y); r_3(Y); w_3(X); w_1(X); r_2(X); w_2(Y)$**

Serializable.

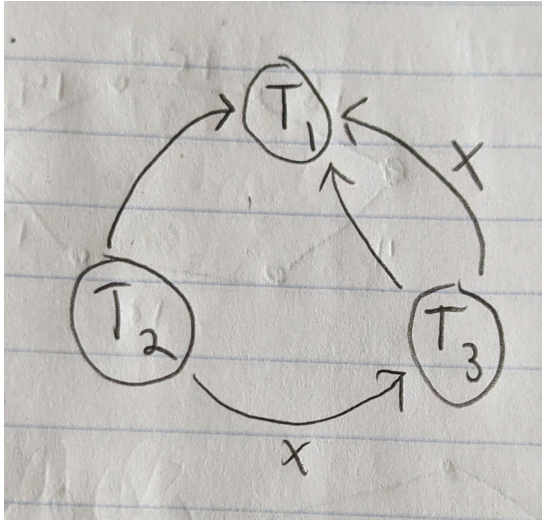
$r_3(Y); w_3(X); r_1(Y); w_1(X); r_2(X); w_2(Y)$



- (c) **w2(X); r1(Y); w2(Y); r2(X); w2(Y); r3(Y)**
 Serializable.
 r1(Y); w2(X); w2(Y); r2(X); w2(Y); r3(Y)

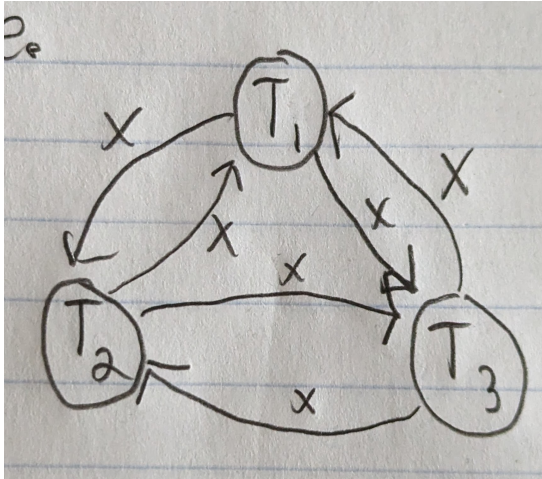


- (d) **r3(X); r2(X); w3(X); r1(Y); w1(X); r2(Y)**
 Serializable.
 r2(X); r2(Y); r3(X); w3(X); r1(Y); w1(X)



(e) $r3(X); r2(X); r1(X); w3(X); w1(X); w2(X)$

Not-Serializable. There is a cycle between all edges in the precedence graph.



4 (0.75 points)

Consider the following actions taken by transaction T1 on a database with objects X and Y:

R1(X), W1(X), R1(Y), W1(Y)

Without altering the order of operations of T1, give example schedules with an additional Transactions T2 (with 4 operations accessing both database objects X and Y) that fulfill the following conditions:

- (a) **Give an example schedule using two transactions (T1 and T2) which is conflict serializable.**

w2(X); r1(X); r2(Y); w2(Y); r2(X); w1(X); r1(Y); w1(Y)

The precedence graph shows no cycles.

- (b) **Give an example schedule using two transactions (T1 and T2) which is not conflict serializable. Explain why the schedule is not conflict serializable.**

w2(X); r1(X); r2(Y); w2(Y); w1(X); r2(X); r1(Y); w1(Y)

The precedence graph has a cycle between T_1 and T_2 . T_1 writes before T_2 reads, but T_2 has already written before T_1 .