

מחלקה 3	מחלקה 2	מחלקה 1	התמחות 3	התמחות 2	התמחות ראשונה	כתובת סטודנט	שם סטודנט	מס' סטודנט
	כימיה	כלכלה		קשקוש	ברבור	באר יעקב	הרשלה	5743

טבלה 1

בדוגמה הנ"ל כל הנתונים של לימודי הסטודנטים מרוכזים בטבלה אחת. אוסף טבלאות נקרא מסד נתונים. ממבנה הטבלה הנ"ל ברור שיש בה שורה לכל קבוצת לימוד שהסטודנט השתתף בה. נניח שמאז שהוכן המחשוב באוניברסיטה למדו בה 100,000 סטודנטים ושסטודנט ממוצע משתתף ב-90 קבוצות לימוד במהלך לימודיו, אזי בשלב זה יש בטבלה 9,000,000 שורות. אם נניח שבכל שורה יש 200 בתים אזי כל הטבלה תופסת 1.8 GB.

חסרונות הטבלה:

א. כפילויות: נניח שישנו מרצה שמלמד כבר עשרות אלפי סטודנטים אז שמו, כתובתו ומשרדו כתובים בכל השורות של אותם סטודנטים.

החסרון בכפילויות:

- תופס מקום.
- מהירות עיבוד.
- נתונים סותרים.

- ישנם שדות שהנתונים האפשריים בהם מעטים וידועים בכל רגע (למשל: מחלקות, התמחויות וכו'). בנתונים אלו מומלץ להשתמש בקודים - חוסך מקום ומונע טעויות.

פתרון: ארגון הנתונים במספר רב של טבלאות:

צורת התמחות	
קוד צורת ההתמחות	צורת ההתמחות
1	מורחב
2	ראשי
3	משני
4	1 מ-3

מסגרת ההתמחות	
קוד מסגרת ההתמחות	מסגרת ההתמחות
1	מורחב
2	ראשי ומשני
3	3 משניים
4	מורחב ומשני

מחלקות	
מס' מחלקה	המחלקה
70	כלכלה
45	תנ"ך

מסגרת לימוד	
קוד למסגרת לימוד	המסגרת
1	הרצאה
2	תרגול

סטודנטים			
קוד מסגרת ההתמחות	כתובת סטודנט	שם סטודנט	מס' סטודנט
2	ת"א	משה	5743

התמחויות		
מס' המחלקה המציעה	ההתמחות	קוד התמחות
88	מדעי המחשב	17
74	לוגיסטיקה	27

קורסים		
מס' המחלקה המציעה	שם קורס	מס' קורס

התמחויות סטודנטים		
צורת התמחות	קוד התמחות	מס' סטודנט
2	25	5743
3	31	5743

מרצים			
משרד מרצה	כתובת מרצה	שם מרצה	מס' מרצה

הוראה					
מס' שעות	קוד מסגרת לימוד	סמסטר	שנה	מס' קבוצה	מס' קורס

עיתויים					
כיתה	שעת סיום	שעת התחלה	יום	מס' קבוצה	מס' קורס

רישום				
ציון	מס' קבוצה	שנה	מס' קורס	מס' סטודנט

בארגון מסד הנתונים הנ"ל אין כפילויות, בנוסף לכך שנעשה שימוש בקודים בכל מקום אפשרי. לכאורה, אפשר שלא להסכים לכך שבטבלה אין כפילות משום שישנם שדות שמופיעים בטבלאות רבות (למשל: מס' סטודנט). אולם ברור שבטבלה שבה מופיע מס' סטודנט - אם נשמיט שדה זה מהטבלה לא יהיה לה שום מובן. לכן ברור שבשום טבלה אין שדה מיותר והכפילות, כביכול, נועדה לקשר בין הטבלאות.

הגדרה: כפילות: נניח שבארגון מחדש מוחקים נתון מסוים בשורה מסוימת בטבלה מסוימת, אזי אם באמת אין שום כפילות אי-אפשר לשחזר את הנתון הזה משאר הנתונים במסד הנתונים.

הערות נוספות:

1. בטבלה ההוראה הוספנו את מס' השעות של כל קבוצת לימוד בכל סמסטר. אפשר לטעון שמס' השעות לא משתנה במשך השנים ולכן כדאי להצמידו לקורסים.
2. טבלת ההוראה וטבלת העיתויים הן טבלאות די דומות וייתכן שאפשר לשקול את איחודן לטבלה אחת. אבל זהו שיקול מוטעה: בטבלת ההוראה נכללים גם השנה והסמסטר, בכך יש רמז לכך שבטבלה זו נרצה לשמור את כל הנתונים ההיסטוריים מתחילת המחשוב. זאת מפני שהמשתמשים במע' חושבים שגם אחרי שנים צריך מידע על כל המרצים שלימדו בעבר. אבל, בטבלת העיתויים אין כל חשיבות למידע היסטורי, לכן בכל קיץ מוחקים את כל הנתונים מטבלת העיתויים וממלאים אותה מחדש בנתוני השנה הבאה. לעומת זאת, טבלת ההוראה גדלה ללא הרף ולעולם לא תימחקנה בה שורות.
3. בטבלת העיתויים לא רשום הסמסטר, זאת מפני שאצלנו מסופרות כל הקבוצות בסדר עולה מתחילת השנה, כלומר, נניח שיש קורס סמסטריאלי הניתן בסמסטר א' ב-4 קבוצות ובסמסטר ב' ב-2 קבוצות נוספות, אזי הקבוצות בסמסטר ב' מסופרות במספרים 5 ו-6. משום כך, מתוך מס' הקורס ומס' הקבוצה בטבלת העיתויים ניתן לדעת מהו הסמסטר מתוך טבלת ההוראה, שכן באותה שנה לא תהיינה 2 קבוצות שוות (בסמסטרים שונים) של אותו קורס.

שיטות לבניית מסד נתונים:

תלות פונקציונלית:

נניח שבטבלה ישנן השדות A,B,C,D,E. אזי השדה D תלוי פונקציונלית בשדות A,B,E (נסמן: $A,B,E \rightarrow D$) אם בכל השורות שבהן יש אותם ערכים ב-A, B ו-E חייב להיות אותו ערך גם ב-D (אפשר גם לומר שצירוף השדות A,B,E גורר את D).

בד"כ מצבי תלות פונקציונלית רבים אופייניים לטבלאות גרועות.

- ← למשל בטבלה 1: מס' קורס, מס' קבוצה, שנה
- ← סמסטר.
- ← מס' מרצה.
- ← סוג הוראה.

בהינתן ערכים לשדות אלו ייתכן שאין בכל הטבלה אפילו שורה אחת בעלת צירוף זה של ערכים (למשל, אם הקורס הזה בכלל לא התקיים באותה שנה). אבל ייתכן שיש שורות רבות מתאימות, אזי בכל השורות צריך להופיע אותו מס' מרצה. הצירוף הזה גורר גם שדות נוספים, כמו למשל, סוג הוראה (הרצאה, תרגיל וכו') או סמסטר...

נניח שאנו חושבים שהצירוף הזה גורר גם את היום, אזי הדבר נכון רק אם קיים כלל שכל קבוצת לימוד נאספת יום אחד בשבוע. אולם הדבר אינו נכון.

- נניח שקבוצה יכולה להתאסף מס' פעמים בשבוע אבל לא פעמיים באותו יום, אזי:
- ← מס' קורס, מס' קבוצה, שנה, יום
- ← שעת התחלה.
- ← כיתה.

יכול גם להיות, למשל: מס' מרצה ← שם מרצה. במצב כזה התלות אינה סימטרית מפני שיתכנו מרצים שונים בעלי אותו שם.

בטבלה 1 יש הרבה מאוד מצבי תלות פונקציונלית, ומכולם אפשר לבחור בצירוף של שדות המגדיר באופן יחיד שורה בטבלה. למשל: יום, שנה, מס' קבוצה, מס' קורס, מס' סטודנט. בהינתן ערכים לאותם שדות אזי ייתכן שאין בטבלה אפילו שורה אחת בעלת ערכים אלו. אבל אם ישנה שורה כזו אזי היא יחידה. לאוסף כזה של שדות קוראים מפתח.

לטבלה יכולים להיות צירופים שונים שכל אחד מהם הוא מפתח. הדבר אופייני לטבלאות גרועות. בטבלה 1 בוודאי ישנו מפתח אחר (צירוף שונה).

במקרה המינימלי המפתח מכיל רק שדה אחד. הדבר אופייני לטבלאות טובות (למשל טבלת קורסים, מרצים, סטודנטים וכו').

במקרה המירבי המפתח מכיל את כל השדות בטבלה:

בכל שורה נמצא צירוף של מחשב ותוכנה הנמכרים בחנות מסוימת. אין שום שדה בודד הגורר את 2 האחרים ואין שום צירוף של 2 שדות שגורר את ה-3. לכן, המפתח מורכב מכל השדות.

תוכנה	מחשב	מחשב
access	דיגיטל	באג
excel	דיגיטל	באג
excel	דיגיטל	מחשבנו

בכל טבלה, המפתח גורר את כל שאר השדות בטבלה. אבל קיימים מצבי תלות פונקציונלית שאינם רצויים לדוגמה:

כמות	מס' מחסן	מחיר יחידה	שם פריט	מס' פריט

המפתח בטבלה הוא הצירוף של מס' הפריט ומס' המחסן.

זאת מפני שכל פריט יכול להימכר במס' מחסנים וגם בגלל שבכל מחסן נמכרים פריטים רבים. בהינתן צירוף של ערכים לשדות אלו, יש לכל היותר שורה אחת בטבלה עם הצירוף הזה. לפי השורה אפשר לראות מהו שם הפריט, מהו מחיר יחידה ומהי הכמות. אבל, הכמות באמת תלויה במפתח בשלמותו: מס' מחסן, מס' פריט ← כמות. ואילו שם הפריט ומחיר יחידה תלויים רק במס' פריט: מס' פריט ← שם פריט / מחיר יחידה.

התלות של שם הפריט ומחירו במפתח נקראת תלות חלקית. זוהי תלות פסולה כי הדבר גורם לכפילויות. כלומר, שם הפריט ומחירו יופיעו בכל השורות של המחסנים השונים שבהם נמצא הפריט ← כפילות.

פתרון: פיצול הטבלה:

מלאי			פריטים		
כמות	מס' מחסן	מס' פריט	מחיר יחידה	שם פריט	מס' פריט

בטבלת מלאי המפתח הוא מס' פריט ומס' מחסן. בטבלת פריטים המפתח הוא מס' פריט. ב-2 הטבלאות אין תלות חלקית.

קיימת תלות פסולה נוספת - תלות עקיפה.

הוראה						
משרד מרצה	מס' מרצה	קוד סוג הוראה	סמסטר	שנה	מס' קבוצה	מס' קורס

המפתח בטבלה זו הוא הצירוף של מס' קורס, מס' קבוצה, שנה. אבל בין השדות שמחוץ למפתח ישנו השדה של משרד המרצה שהוא אמנם תלוי במפתח, אבל הוא תלוי, למעשה, דרך שדה מס' המרצה. כלומר, מס' המרצה באמת תלוי במפתח. ברור שמס' המרצה ← משרד המרצה, לכן, בגלל הטרנזיטיביות של יחס התלות הפונקציונלית, משרד המרצה גם תלוי במפתח. זוהי תלות פסולה משום שהיא מביאה לכפילויות. כלומר, משרד המרצה יהיה רשום בכל השורות המתארות את משימות ההוראה של המרצה בכל השנים.

פתרון:

הוראה						משרדי מרצים	
מס' מרצה	קוד סוג הוראה	סמסטר	שנה	מס' קבוצה	מס' קורס	משרד מרצה	מס' מרצה

טבלה טובה:

- בטבלה יש מפתח. כל שדות המפתח חייבים להיות בלתי תלויים בינם לבין עצמם.
- ברוב המקרים יש בטבלה שדות מחוץ למפתח. גם שדות אלו חייבים להיות בלתי-תלויים בינם לבין עצמם.
- ברור שכל שדה במפתח לא יהיה תלוי בשדות מחוץ למפתח.
- התלות היחידה הקיימת בטבלה היא תלות מהסוג: מפתח ← שדה מחוץ למפתח. כאשר אין שום תלות חלקית ואין שום תלות עקיפה.
- יש לוודא שבשום טבלה לא קיימת התופעה של תלות רב-ערכית.

עיצוב מסד נתונים:

תחילה מעצבים את הטבלאות באופן אינטואיטיבי. לאחר מכן, נאתר בכל טבלה את המפתח. בהמשך נבדוק האם מתקיימים בטבלה כל הכללים של טבלה טובה. אם לא, יש לפצל אותה לטבלאות טובות.

כבר בשלב ה-1 צריך להבטיח שכל פריטי המידע יוכלו להירשם בטבלאות, כולל כל הקשרים ביניהם.

דוגמה: לפי עמ' 51, תרגיל 68:

פעילים		התמחויות פעילים		התמחויות פעילים	
שם	זיהוי	קוד ההתמחות	ההתמחות	קוד התמחות	זיהוי
אלימלך	5	7	מתלהב מן המניין	5	9
חושם	7	9	מתעלף לפי דרישה	5	15

9	יוסי	15	זועק בטמטום	9	30
		22	מנהיג מתלהמים	7	22
		30	מרצה מושלם	7	9

סיבות לאירועים	
קוד סיבה	הסיבה
7	הכתרת העם למנהיג
11	נגד "שלום"
13	שלום עכשיו
17	שוויון זכויות לנשים

אירועים			
זיהוי	אירוע	קוד מיקום	תאריך
			קוד סיבה

מיקומים			
קוד אירוע	זיהוי משתתף	קוד התמחות	מחיר

משתתפים באירועים			
קוד אירוע	זיהוי משתתף	קוד התמחות	מחיר
257	5	15	10.3
257	7	22	13.7
257	5	9	25.2

נמחיש את העובדה שהטבלאות מאפשרות את רישום המידע במלוא הפירוט בדוגמאות הבאות: נתייחס למחיר שגובה כל משתתף באירוע. ייתכן שבארגון מסוים המחיר למעשה צמוד להתמחות, כלומר התמחות 9, למשל, מזכה כל פעיל ב-18.9 ש"ח. אם זה המצב, אזי את המחיר נצמיד לטבלת ההתמחויות. לעומת זאת, נניח שלאנשים שונים יש תעריפים שונים לאותה התמחות, אבל התעריף אינו משתנה באירועים אזי נצמיד את המחיר לטבלת התמחויות הפעילים. לעומת זאת, העובדה שהצמדנו את המחיר לטבלת המשתתפים באירועים אומרת שאותו משתתף באותו תפקיד באירועים שונים יקבל תשלום שונה.

הטבלה שלפנינו מאפשרת את הפירוט הגדול ביותר. נשאלת השאלה: מדוע לא רשמנו את המשתתפים בטבלת האירועים אלא יצרנו טבלה חדשה של משתתפים באירועים? התשובה היא שבכל אירוע יש משתתפים רבים ולכן זהו אבסורד להצמיד את המשתתפים לטבלת האירועים. נניח שהמחיר נקבע רק לפי ההתמחות. אזי נצמיד את המחיר לטבלת ההתמחויות. נשאלת השאלה: מדוע לא נוסף טבלה חדשה עם השדות קוד התמחות ומחיר? התשובה היא שטבלה כזו אכן תענה לבעיה, אולם המפתח בטבלה זו (קוד ההתמחות) זהה למפתח בטבלת ההתמחויות ותמיד רצוי לאחד שתי טבלאות בעלות אותו מפתח, כי החזקתן בנפרד מהווה בזבוז מקום.

תלות רב-ערכית:

בטבלה זו המפתח מורכב מכל השדות בטבלה, זאת מפני שברור שאין שדה אחד הגורר את שני האחרים וגם אין צירוף של 2 שדות הגוררים את השלישי. בטבלה אין שדות מחוץ למפתח, לכן חלק מהכללים של טבלה טובה מתקיימים באופן ריק, לכן, לפי ההגדרות הקודמות, זוהי טבלה טובה. אבל, יש בה כפילויות. הדבר מתבטא בכך שידענו מהו התוכן של שורות חדשות על סמך ידיעת התוכן של שורות קודמות. לכן, יש כפילויות בטבלה אלא שהן כפילויות מסוג חדש. בטבלה יש תלות רב ערכית: התמחות \leftarrow קורס.

חובות		
קורס	התמחות	סטודנט
C	מחשבים	א'
אינפי	מחשבים	א'
מסדי נתונים	מחשבים	א'
חשמל	פיסיקה	ב'
C	מחשבים	ס'
אינפי	מחשבים	ס'
מסדי נתונים	מחשבים	ס'

תלות כזו אומרת שההתמחות לא גוררת קורס אחד אלא אוסף של קורסים. לכן, בטבלה רואים, למשל, שכל סטודנט המתמחה במחשבים חייב בדיוק באותה קבוצה של קורסים. זוהי כפילות.

פתרון:

התמחויות	
התמחות	סטודנט
מחשבים	א'
פיסיקה	א'
מחשבים	ס'
פיסיקה	ב'

קורסים	
קורס	התמחות
C	מחשבים
אינפי	מחשבים
מסדי נתונים	מחשבים
חשמל	פיסיקה
C	מחשבים
אינפי	מחשבים
מסדי נתונים	מחשבים

כעת אין כפילויות. מהטבלאות ניתן לקבל את כל המידע על חובות הסטודנט בכל התמחויותיו, לכן הטבלאות החדשות הן התיקון לטבלה הגרועה המקורית.

המפתח בטבלה הוא הצירוף של סטודנט והתמחות. הצירוף הזה גורר את היועץ, אבל בטבלה קיימת התלות יועץ ← התמחות. כלומר, שדה שבתוך המפתח (התמחות) תלוי בשדה שמחוץ למפתח (יועץ). זוהי תופעה פסולה שגורמת לכפילויות.

סטודנט	התמחות	יועץ
א'	מחשבים	מושקאט
ג'	פיסיקה	שלמה
א'	פיסיקה	שלמה
ס'	מחשבים	אריאל

פתרון:

יועצים	
יועץ	התמחות

"יועץ"	
סטודנט	יועץ

ארגון טוב של מסדי נתונים מונע כפילויות וסתירות. אבל זה גורר שצריך לטרוח בשביל לשלוח נתונים.

דוגמה: נניח שברצוננו לשלוח עבור סטודנט 6275 דו"ח נתונים במבנה: ציון, סמסטר, שנה, שם מרצה, שם קורס, מס' קורס. כלומר, ברצוננו ליצור טבלה חדשה בשם Dan שתכיל את כל המידע על קבוצות הלימוד שבהן השתתף הסטודנט. כל המידע המבוקש נמצא בטבלאות השונות. ישנן 6 פעולות המוגדרות במסד נתונים טבלאי באמצעותן אפשר לארגן כל טבלה חדשה שנרצה מטבלאות קיימות. כל אחת מ-6 הפעולות יוצרת טבלה חדשה, שזוהי תכונה חשובה מאוד (סגירות) כי ניתן לבצע פעולות נוספות על הטבלה.

התחביר של הפעולה כולל את שם הפעולה, את הטבלה/ות שעליה היא מופעלת, שם של טבלה חדשה לרישום התוצאה (אם השם הזה כבר קיים זוהי בעיה של תוכנה) ותנאי האופייני לפעולה. 1. נשלוף את כל השורות של סטודנט 6275:

```
select
from רישום
into BC3
where מס'סטודנט=6275
כתוצאה מהפעולה תיווצר טבלה חדשה, BC3, בדיוק באותו מבנה של הטבלה רישום והיא תכיל בדיוק את השורות מטבלת רישום השייכות לסטודנט 6275. השדות בטבלה BC3: מס' סטודנט, מס' קורס, שנה, ציון.
```

2. join BC3 with קורסים into DX7 where (מס'קורס.קורסים = מס'קורס.BC3)

פעולה זו מצרפת מידע מ-2 טבלאות לטבלה אחת (פעולת ה-join היא קומוטטיבית). באופן מעשי, במקרה שלנו, הפעולה תוסיף לכל שורה בטבלה BC3 את הנתונים מטבלת הקורסים (שם הקורס, מס' מחלקה) המתאימים למס' הקורס באותה שורה. אפשר לראות את ביצוע הפעולה בצורה הבאה: ניקח את השורה הראשונה של הטבלה BC3, עבורה נסרוק את כל טבלת הקורסים בחיפוש אחרי שורה (או שורות) המקיימות את התנאי עם אותה שורה ראשונה. אצלנו ברור שבטבלת הקורסים צריכה להיות בדיוק שורה אחת מתאימה. בכל מקרה שנמצאה שורה מתאימה תיווצר שורה חדשה בטבלת התוצאה שתכיל את כל הנתונים מ-2 השורות. בהמשך נסרוק את טבלת הקורסים מחדש עבור השורה השנייה של הטבלה BC3 וכן הלאה עד השורה האחרונה.

```
join DX7 with הוראה into X3
where (מס'קבוצה.הוראה = מס'קבוצה.DX7) and (מס'קורס.הוראה = מס'קורס.DX7) and (שנה.הוראה = שנה.DX7)
```

במקרה הזה התנאי חייב להיות מורכב כי מחפשים בטבלת ההוראה את מס' המרצה, את הסמסטר, את קוד מסגרת הלימוד ואת מס' השעות המתאימים בדיוק לכל שורה בטבלה DX7. כדי לאתר בדיוק את השורה הזו חייב התנאי לכלול את כל המפתח של טבלת ההוראה.

```
join X3 with מרצים into X4
where מס'מרצה.מרצים = מס'מרצה.X3
```

```
join X4 with מסגרות-לימוד into X5
```

where (קוד-מסגרת-לימוד.מסגרות-לימוד = קוד-מסגרת-לימוד.X4) בטבלה X5 כבר ישנם כל השדות הדרושים אבל יש הרבה שדות שאינם דרושים:

- project מס' קורס, ... , ציון, ... , מס' קורס
from X5
into Dan

דוגמה: עמ' 31 תרגיל 21:

select from מאמנים
into X8
where מאמן = אלי

X8	
קבוצה	מאמן
4	אלי

project קבוצה
from X8
into X9

X9	
קבוצה	
4	

join X9 with שחקנים
where קבוצה.שחקנים = קבוצה.X9
into Y3

Y3	
קבוצה	שחקן
4	ק
4	י
4	ד

join X9 with שחקנים
where קבוצה.שחקנים != קבוצה.X9
into Y4

Y4		
קבוצה ב'	קבוצה א'	שחקן
4	1	א
4	3	ו
4	3	ח

בטבלה Y4 צריך להופיע גם הטור קבוצה מהטבלה X9 וגם הטור קבוצה מטבלת השחקנים כי הם אינם זהים. מצד שני, ל-2 הטורים יש אותם שמות. כל תוכנה מתגברת על-כך בצורה כלשהי. נבצע פעולות project על Y3 ועל Y4 במטרה לקבל את Y5 ואת Y6 שיכילו רק את השחקנים ולבסוף נעשה join על Y5 ו-Y6 ל-Y7 שבעצם תכיל את כל מצבי התיעוב הפוטנציאליים שיכלו להיווצר בין שחקנים של אלי לבין שחקנים אחרים. כדי למצוא מה הם המצבים שהוצאו מהכוח אל הפועל, כלומר השורות המופיעות גם ב-Y7 וגם בטבלת התיעוב:

- intersect Y7 with תיעוב
into Q3

פעולת החיתוך מופעלת אך ורק על טבלאות זהות בהגדרתן והיא יוצרת טבלה חדשה המכילה אך ורק את השורות המשותפות ל-2 הטבלאות. אולם בטבלת התיעוב לא משנה סדר רישום השחקנים בכל שורה, אבל מבחינת המחשב, ברור שאם 2 שחקנים מופיעים בסדר שונה בשורה של Y7 ובשורה של טבלת התיעוב - החיתוך לא יתפוס אותם. לכן, נניח שנהפוך בטבלת התיעוב את תוכן הטבלה אבל נשאיר את שמות השדות. נקרא לטבלה החדשה A ונעשה:

intersect תיעוב with A
into Q4

לבסוף, נצרף את Q3 ו-Q4.

- union Q3 with Q4
into Q5

פעולת האיחוד מופעלת על 2 טבלאות זהות בהגדרתן. טבלת התוצאה תכיל את כל השורות שמופיעות לפחות באחת מהן (שורה שמופיעה ב-2 הטבלאות תופיע בטבלת התוצאה רק פעם אחת).

נניח כעת שאנו מעוניינים לבדוק האם יש שחקנים שאינם מעורבים בשום מצב של תיעוב:

project שחקן
from שחקנים
into K7

כעת נבצע project על טבלת התיעוב שבו ניקח רק את שחקן א' לתוך טבלה חדשה L1 ונניח שאם יש שורות כפולות הן מבוטלות. בהמשך נבצע project נוסף על טבלת התיעוב בו ניקח רק את שחקן ב' לתוך L2. כעת נעשה איחוד L1 עם L2 לתוך L3. כדי לקבל את אלו שלא מתעבים אף אחד:

- subtract L3 from K7
into K10

ההפחתה תעתיק מ-K7 ל-K10 רק את השורות שאינן מופיעות ב-L3. ההפחתה אינה קומוטיבית (בניגוד ל-join, union, intersect).

פעולות עדכון של מסד הנתונים:

כל נתון במסד הנתונים ניתן לעדכון באופן שוטף. קטע תכנותי שעוסק בעדכון נתונים נקרא תנועה.



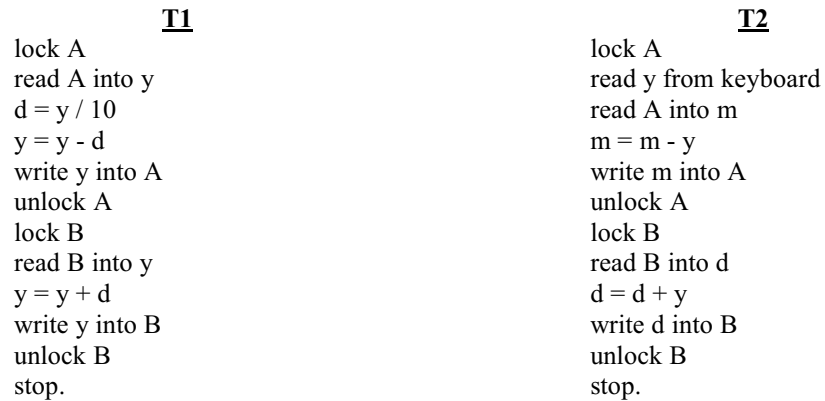
האותיות הגדולות מסמלות נתונים במסד הנתונים. למשל: A מסמנת את היתרה בחשבון עו"ש מסוים ו-B בחשבון אחר. התנועה T1 דואגת לכך שמעשר מן הכסף של חשבון A יועבר ל-B. התנועה T2 מקבלת סכום מלוח המקשים והסכום הזה יועבר מחשבון A ל-B. האותיות הקטנות מסמלות משתנים פנימיים שאינם משותפים ל-2 התנועות. נניח ש- $B = -700, A = 8,000$.

לאחר ביצוע תנועה T1: $B = 100, A = 7,200$. אחרי זה בוצעה T2 והסכום שנקרא מלוח המקשים הוא $y = 3,000$, אז נקבל: $B = 3,100, A = 4,200$. אם נבצע קודם את T2 ואח"כ את T1 נקבל בסוף: $B = 2,800, A = 4,500$. עצם העובדה שאין שום כפייה חיצונית על סדר ביצוע התנועות פירושו שכל תוצאה שתקבל תהיה נכונה.

אולם הפעולות אינן מתבצעות בסדר סדרתי אלא לפי שיתוף זמנים - כלומר, מתבצעות בו-זמנית הרבה תנועות. המחשב מקצה לכל תנועה זמן מסוים שבו היא מתבצעת ככל שהיא מספיקה, לאחר מכן התנועה נפסקת והשליטה נעברת לתנועה אחרת. כתוצאה מכך, 2 התנועות הנ"ל יכולות להתבצע בהרבה ביצועים לא סדרתיים. אבל בשום אופן אסור מצב שהביצוע הלא-סדרתי ייתן תוצאה השונה מ-2 התוצאות של 2 הביצועים הסדרתיים האפשריים. להלן ביצוע לא-סדרתי שנותן תוצאה שגויה:

מבוצע קטע 1, אח"כ קטע 2 ובסוף קטע 3. ביצוע כזה חייב לתת תוצאה שגויה, זאת מפני שכאשר T1 חוזר להתבצע הוא רושם ב-A את הערך של המשתנה הפנימי y ודורס את הערך של T2- כתב ב-A.

אחרי חלק 1: שום דבר לא נכתב. אחרי חלק 2: $B=2,300, A=5,000$. אחרי חלק 3: $B=3,100, A=7,200$. הפתרון: פקודת נעילה (l - lock) ושחרור (ul - unlock)



עכשיו מס' התזמונים הלא-סדרתיים שיכולים להיווצר הוא מאוד קטן. לדוגמה, התזמון הקודם לא יכול להתבצע כי אם תנועה רוצה לנעול נתון שכבר נעול ע"י תנועה אחרת היא לא תצליח ותיכנס למצב של המתנה עד שהנתון ישוחרר (יש נעילת כתיבה ונעילת קריאה).

לכאורה אם כל תנועה תקפיד לנעול כל נתון לפני שתיגש אליו ותשחרר אותו אחרי שתסיים לטפל בו, אזי כל תזמון בלתי סדרתי שיכול להיווצר חייב להיות נכון. אולם פתרון כזה זה אינו מספיק:

1	{	lock A	read A into y	$d = y / 10$	$y = y - d$	write y into A	unlock A	lock B	read B into y	$y = y + d$	write y into B	unlock B	stop.
---	---	--------	---------------	--------------	-------------	----------------	----------	--------	---------------	-------------	----------------	----------	-------

T1

T2	read y from keyboard
	lock A
	read A into m
	$m = m - y$
	write m into A
	unlock A
	lock B
	read B into d
	$d = d + y$
	write d into B
	unlock B
	lock d
	$g = m + d$
	write g into d
	unlock d
	stop.

נניח שתנאי ההתחלה הם: $B = 7,000, A = 8,000$ ו- $T1$ מתבצעת לפני $T2$.

$T1T2: A = 7,200, B = 1,500$

ונניח ש- $T2$ קוראת מלוח המקשים את הסכום 2,000 ולכן אחרי $T2: d=8,700, B = 3,500, A = 5,200$

נניח ש- $T2$ מתבצעת לפני $T1$:

$T2T1: A = 6,000, B = 2,700, d = 8,700$

$A = 5,400, B = 3,300$

רואים כי ב-2 התזמונים הסדרתיים מתקבלות תוצאות שונות. כלומר כל תזמון לא-סדרתי צריך להסתיים באחת מ-2 צורות אלו.

נניח שב- $T1$ יתבצע קטע 1, לאחר מכן תתבצע $T2$ בשלמותה ולבסוף תושלם $T1$. התוצאות תהיינה:

$(T1) A = 7,200 \quad (T2) A = 5,200, B = 2,700, d = 7,900 \quad (T1 \text{ contd.}) B = 3,500$

כלומר, מבחינת A ו- B התוצאות תהיינה כמו בתזמון הסדרתי $T1T2$ אבל d יהיה שונה. כלומר, קיבלנו תזמון סדרתי לא נכון וזאת למרות שהקפדנו לנעול ולשחרר כדרוש.

הפתרון: נעילה לפי פרוטוקול דו-שלבי: לפיו כל הנעילות חייבות להתבצע לפני כל השחרורים. כלומר, הפרוטוקול אינו מאלץ אותנו לנעול בהתחלה את כל הנתונים. מטעמי יעילות רצוי לנעול מאוחר יותר ככל האפשר. כמו כן, לא צריך לבצע את השחרורים בסוף. מטעמי יעילות מוטב לשחרר מוקדם ככל האפשר. יש לשים לב שהנעילה האחרונה בתנועה חייבת להתבצע לפני השחרור הראשון בתנועה. קל להוכיח שאם כל התנועות מקפידות על משטר זה אזי לא יכול להתבצע אף פעם תזמון לא סדרתי שגוי.

	{	lock A	read A into y	$d = y / 10$	$y = y - d$	write y into A	lock B	unlock A	read B into y	$y = y + d$	write y into B	unlock B	stop.
--	---	--------	---------------	--------------	-------------	----------------	--------	----------	---------------	-------------	----------------	----------	-------

T1

T2	read y from keyboard
	lock A
	read A into m
	$m = m - y$
	write m into A
	lock B
	lock d
	unlock A
	lock B
	read B into d
	$d = d + y$
	write d into B
	unlock B
	$g = m + d$
	write g into d
	unlock d
	stop.

דוגמה:

בתנועה T2 נוצרה אי-יעילות בכך שנעלנו את d הרבה לפני המקום הדרוש. נאלצנו לעשות את זה כדי לשחרר את A. לחילופין יכולנו לכתוב את lock d, unlock A אחרי הכתיבה ל-, אולם היינו משלמים באי-יעילות אחרת: דחיית שחרור A.

כעת: T1 התחילה קודם, אולם T2 יכולה להידחק לכל המוקדם לאחר שחרור A ע"י T1. אם זה המצב אזי T2 תצליח להתבצע עד כתיבת A. עכשיו T2 תצטרך להמתין עד סיום T1 והתזמון שיתקבל יהיה נכון לחלוטין גם מבחינת d. באותה מידה לא יכול להיווצר שום תזמון בלתי סדרתי שגוי אחר. הפרוטוקול הזה מגביל במידה רבה את הביצוע המקבילי, אבל הגבלות אלו הן המינימום הדרוש כדי להבטיח נכונות.

עם זאת, ברור שבדוגמה הנ"ל המקביליות צומצמה במידה רבה כי בכוונה הכללנו בדוגמה 2 נתונים זהים (A ו-B) ב-2 התנועות. ברור שמבחינת ה"התנהגות הטבעית" ברוב המקרים מתבצעות במקביל תנועות זרות כמעט לחלוטין, לכן המקביליות אינה ניזוקה במידה מכרעת. בכל זאת ברור שהפרוטוקול דרוש לאותם מצבים נדירים שבהם מתבצעות בו-זמנית תנועות הפונות לאותם הנתונים.

חסרונות לפרוטוקול הדו-שלבי:

1. מגביל את המקביליות (חסרון קל).
2. חסרון חמור יותר: יכול להיווצר מצב קיפאון (deadlock). לדוגמה (דוגמה 4):

T1	T2
lock B	read y from keyboard
read B into y	lock A
$d = y / 10$	read A into m
$y = y - d$	$m = m - y$
write y into A	write m into A
lock A	lock B
unlock B	unlock A
read A into y	read B into d
$y = y + d$	$d = d + y$
write y into A	write d into B
unlock A	unlock B
stop.	stop.

נניח ש-T1 התחילה להתבצע ולפני שהיא נועלת את A הגיעה T2. T2 מצליחה להתחיל בביצוע כי היא נועלת את A שאליה עוד לא הגיעה. בשלב מסוים T1 תרצה לנעול את A, היא לא תצליח בכך כי T2 מחזיקה את A. אולם T2 לא תגיע לשחרור את B. אין כל דרך למנוע את המצב הזה. כל מה שאפשר לעשות הוא ליצור במע' ההפעלה מודול שיאתר מצב כזה ולאחר האיתור יפרק את המצב, כלומר ביטול תנועה אחת (כלומר, ביטול כל כתיבותיה). כאשר מבטלים תנועה ייתכן שהיא כתבה ערכים חדשים בנתונים שבהם השתמשו תנועות אחרות שאולי אפילו כבר הסתיימו.

נבצע כעת דוגמה זו (דוגמה 4) במחשב. נניח ש-T1 הגיעה ראשונה. התנועה מתחילה להתבצע ומקבלת 4 פקודות לביצוע בטרם תילקח השליטה ממנה. לאחר 4 פקודות מגיעה T2 ומעכשיו הן מתבצעות לסירוגין במנות של 4 פקודות. כמובן שהחוקיות הזו יכולה להשתבש אם תנועה רוצה לנעול נתון שהוא נעול ברגע זה ע"י חברתה. נוסף לכך, נוסף יומן. נניח שכל פעולה נמשכת יחידת זמן אחת:

זמן	T1	T2	A	B	d	יומן
1	lock A		8,000	-500	57	T1 starts
2	read A into y (y = 8,000)					
3	$d = y/10$ (d = 800)					
4	$y = y - d$ (y = 7,200)					
5		read y from KB (נניח קלטנו y = 1,700)				T2 starts
6		ניסיון כושל של T2 לנעול את A. T2 נכנס להמתנה.				
7	write y into A		7,200			T1, A, 7,200
8	lock B					

9	unlock A				
10		lock A			
11		read A into m (m = 7,200)			
12		m = m - y (m = 5,500)			
13		write m into A	5,500		T2, A, 5,500
14	read B into y (y = -500)				
15	y = y + d (y = 300)				
16	write y into B		300		T1, B, 300
17	unlock B				T1 commits
18		lock B			
19		lock d			
20		unlock A			
21		read B into d (d = 300)			
22	stop				
23		d = d + y (d = 2,000)			
24		write d into B	2,000		T2, B, 2,000
25		unlock B			
26		g = m + d (g = 5,700)			
27		write g into d		7,500	T2, d, 7,500
28		unlock d			T2 commits
29		stop			

אנו רואים ביצוע לא סדרתי מסוים של 2 התנועות. כמובן שיכולים להתקבל ביצועים לא סדרתיים אחרים, עם זאת, מספרם מוגבל מאוד בגלל הפרוטוקול הדו-שלבי (ראינו, למשל, ש-T2 נתקעה בשלב מסוים לזמן רב בניסיונה לנעול את A).
 אנו רואים שלמערכת התוסף יומן. כאשר תנועה כותבת ערך לנתון במסד הנתונים, הוא לא נכתב במסד הנתונים ממש אלא ביומן. ראינו שביומן נכתבת שורה המספקת את זיהוי התנועה, זיהוי הנתון והערך החדש. כמו כן, התנועה רושמת ביומן את העובדה שהיא התחילה והודעת commit ברגע שהיא מסיימת את הכתיבה האחרונה. לאחר ה-commit התנועה עדיין יכולה לבצע פעילויות, כגון: הפקת ד"חות וכדו', אבל לא כתיבת נתונים במסד הנתונים.
 קיים במע' מנגנון אשר סורק בקביעות את היומן ומעתיק מהיומן למסד הנתונים את כל הכתיבות של התנועות שהגיעו ל-commit. המנגנון הזה מאפשר להתגבר בצורה נכונה על כל תקלה, כפי שנראה בדוגמאות הבאות:

הנתונים לאחר הנתונים לאחר ההתאוששות	הפעולות לאחר התיקון	מצב היומן	תקלה לאחר הרישום ביומן של תוצאות הפעולה
כמו בהתחלה	1. כלום. 2. מתחילים את T1 מחדש.	T1 starts	T1 ע"י $d = y / 10$

התקלה קרתה עוד לפני ש-T1 (וכמובן T2) הגיעה ל-commit. לכן, ברור שמסד הנתונים עדיין לא השתנה, לכן, לאחר התיקון פשוט נתחיל מחדש את התנועות שכבר התחילו (אצלנו זה רק T1).

כמו בהתחלה	1. כלום. 2. מתחילים את T1 מחדש וגם את T2.	T1 starts T2 starts T1, A, 7,200 T2, A, 5,500	T1 ע"י $y = y + d$
A = 7,200 B = 300 d = 57	1. redo(T1). 2. מתחילים את T2 מחדש.	T1 starts T2 starts T1, A, 7,200 T2, A, 5,500	T2 ע"י read B into d

		T1, B, 300 T1 commits	
--	--	--------------------------	--

בתקלה ה-3 אחרי התיקון מסתכלים ביומן ורואים ש-T1 כבר הגיעה ל-commit, לכן מעתיקים את עדכונה מהיומן למסד הנתונים. זו אינה הפעלת התנועה מחדש, אלא פרוצדורה שנקראת redo. אפשר לשאול: אולי כבר בוצע redo? אולם אין שום נזק בביצוע redo אחרי redo.

A = 5,500 B = 2,000 d = 7,500	redo(T1) redo(T2)	T1 starts T2 starts T1, A, 7,200 T2, A, 5,500 T1, B, 300 T1 commits T2, B, 2,000 T2, d, 7,500 T2 commits	לאחר T2 commits
-------------------------------------	----------------------	--	-----------------

בדוגמה 4 גם T2 הגיע ל-commit, לכן נבצע גם redo(T1) וגם redo(T2)

A = 5,500 B = 2,000 d = 7,500	redo(T1) redo(T2)	T1 starts T2 starts T1, A, 7,200 T2, A, 5,500 T1, B, 300 T1 commits T2, B, 2,000 T2, d, 7,500 T2 commits	תוך כדי התאוששות מתקלה 4
-------------------------------------	----------------------	--	--------------------------

בדוגמה 5 קרתה תקלה תוך כדי ביצוע ה-redo של התקלה הקודמת. לכן, לאחר התיקון, נחזור על מה שאולי הספקנו לעשות לאחר התקלה הקודמת.

בשיטת היומן רושמים ביומן רק את הערכים החדשים של הנתונים המתעדכנים. מסיבה זו אפשר להעתיק אותם רק לאחר ה-commit. זאת מפני שאם נעתיקם לפני ה-commit ותרחש תקלה לפני ה-commit ונצטרך לשחזר את הערכים המקוריים - לא יהיה מהיכן לעשות זאת. לכן, יש שיטת יומן חלופית שבה רושמים ביומן גם את הערכים המקוריים. הרישום הזה מאפשר להעתיק מהיומן למסד הנתונים את הערכים המקוריים מתוך היומן. נמחיש את פעולת היומן הזה בדיוק עבור אותן דוגמאות של היומן הקודם:

הנתונים במסד הנתונים לאחר ההתאוששות	הפעולות לאחר התיקון	מצב היומן	תקלה לאחר הרישום ביומן של תוצאות הפעולה
כמו בהתחלה	1. undo(T1) 2. מתחילים את T1 מחדש.	T1 starts	T1 ע"י $d = y / 10$

לאחר תיקון התקלה מסתבר שקיימת תנועה אחת, T1, שכבר התחילה להתבצע. לכן, יש חשש שהיא כבר כתבה עדכונים ביומן (במקרה שלנו זה עוד לא קרה). בשיטת יומן זו יש חשש שהעדכונים כבר הועתקו למסד הנתונים. מצד שני, הואיל והתנועה לא הגיעה ל-commit, חייבים לבטל את כל פעולותיה ולהתחילה מחדש.

לכן מבצעים פרוצדורת undo - פעולה המבטלת עדכונים שנעשו ע"י T1 וכבר נכתבו במסד הנתונים. שחזור הערכים המקוריים אפשרי הפעם. לאחר ה-undo נתחיל את T1 מהתחלה.

כמו בהתחלה	1. undo(T2) undo(T1) 2. מתחילים מחדש את T1 ו-T2	T1 starts T2 starts T1, A, 8,000, 7,200 T2, A, 7,200, 5,500	T1 ע"י $y = y + d$
------------	---	--	--------------------

בדוגמה 2 התקלה קרתה לאחר שגם T1 וגם T2 כתבו עדכונים ביומן ועדיין לא הגיעו ל-commit. לכן, צריך לבטל את הפעולות של שתיהן ע"י undo. לאחר מכן מתחילים את שתיהן מחדש.

<p>A = 7,200 B = 300 היתר כמו בהתחלה</p>	<p>1. redo(T1) 2. מתחילים מחדש את T2 undo(T2)</p>	<p>T1 starts T2 starts T1, A, 8,000, 7,200 T2, A, 7,200, 5,500 T1, B, -500, 300 T1 commits</p>	<p>T2 read B into d ע"י</p>
--	---	--	-----------------------------

בתקלה ה-3 תנועה T1 כבר הגיעה ל-commit לכן נבצע redo שלה כדי להיות בטוחים שהערכים החדשים ייכתבו במסד הנתונים. לעומת זאת, התנועה T2 לא הגיעה ל-commit לכן נבצע undo כדי להבטיח את שחזור הערכים המקוריים. כמובן, שלבסוף, נתחיל את T2 מחדש. בפעולת ה-undo של T2, למשל, בוחנים כל שורה ביומן שנכתבה ע"י T2. משחזרים את הערך המקורי של המשתנה הכתוב בשורה זו רק אם ברגע זה כתוב במסד הנתונים הערך הנקוב באותה שורה (כי אחרת העדכון הזה כבר נמחק ע"י תנועה אחרת ואין מקום לשחזר את הערך המקורי). במחשבה שנייה, הסדר של ביצוע ה-redo וה-undo צריך להיות לא בהתאם לערכים. הקריטריון הנכון הוא: יש לאתר את הביצוע הסדרתי השקול לביצוע הלא-סדרתי שנוצר. במקרה שלנו זהו הביצוע T1 ואחרי זה T2. תמיד חייב להיות ביצוע סדרתי שקול כתוצאה מההקפדה על הפרוטוקול הדו-שלבי ועכשיו נבצע את ה-redo וה-undo בדיוק לפי סדר הביצוע הסדרתי הזה.

<p>A = 5,500 B = 2,000 d = 7,500</p>	<p>3. redo(T1) redo(T2)</p>	<p>T1 starts T2 starts T1, A, 8,000, 7,200 T2, A, 7,200, 5,500 T1, B, -500, 300 T1 commits T2, B, 300, 2000 T2, d, -57, 7500 T2 commits</p>	<p>לאחר T2 commits</p>
--	---------------------------------	---	------------------------

בדוגמה 4 התקלה קרתה אחרי ה-commit של 2 התנועות לכן מבצעים redo לשתיהן.

<p>A = 5,500 B = 2,000 d = 7,500</p>	<p>1. redo(T1) redo(T2)</p>	<p>T1 starts T2 starts T1, A, 8,000, 7,200 T2, A, 7,200, 5,500 T1, B, -500, 300 T1 commits T2, B, 300, 2000 T2, d, -57, 7500 T2 commits</p>	<p>תוך כדי התאוששות מתקלה 4</p>
--	---------------------------------	---	---------------------------------

בדוגמה 5 קרתה תקלה תוך כדי ביצוע ה-redo של התקלה הקודמת. לכן, לאחר התיקון, נחזור על מה שאולי הספקנו לעשות לאחר התקלה הקודמת.

משטר תוויות הזמן:

במשטר זה אין סכנה של deadlock אולם יש לו חסרונות אחרים. ההחלטה באיזה משטר להשתמש היא החלטת המע' והתוכנה שבה משתמשים אמורה לאפשר את 2 האופציות. במשטר זה אין נעילות ושחרורים. כלומר, בתכנות התנועות בכלל לא מוטרדים מהעובדה שהתנועה תבוצע במקביל לתנועות אחרות. כל תנועה שמתחילה להתבצע תקבל את תווית הזמן שהיא הזמן המדויק שבו היא נכנסה לפעולה. מניחים שהשעון של המחשב כ"כ מדויק עד שכל תנועה מקבלת תווית זמן שונה.

כל תנועה אמורה לקרוא נתונים ולכתוב נתונים. לכן, לכל נתון במסד הנתונים תוצמדה 2 תוויות זמן. $tr(A)$ היא תווית הקריאה של הנתון. זוהי תווית הזמן של התנועה האחרונה שקראה את הנתון. למשל, נניח שאנחנו עכשיו בזמן 10,000 והנתון A נקרא לאחרונה ע"י תנועה שהתחילה בזמן 2,000 אזי $tr(A) = 2,000$. כלומר, זהו אינו זמן הקריאה בפועל. בהחלט ייתכן שהתנועה התבצעה מזמן 2,000 עד 7,000 והנתון A נקרא רק בזמן 6,300. באותה מידה יש לנתון זמן כתיבה - $tw(A)$ - זמן ההתחלה של התנועה האחרונה שכתבה את הנתון. (ייתכן $tr(A) = tw(A)$). **חסרון:** התוויות מוצמדות בצמוד לנתון במסד הנתונים, כלומר נפח מסד הנתונים יגדל פי 3 בערך.

34		read A into m (m = 8,000)		33							
35		m = m - y (m = 7,300)									
36	רוצים לכתוב את y לתוך A אבל זה אסור										
37	מתחילים מחדש את T1. read A into y (y = 8,000)			37							
38		d = y / 10 (d = 800)									
39		ניסיון לכתוב את m ל-A. הניסיון נכשל.									

בזמן 30 התחילה התנועה T1. היא קיבלה את תוויית הזמן 30 שתלווה את כל קריאותיה ואת כל כתיבותיה. T1 מייד רוצה לקרוא את A. הואיל ותוויית הזמן של T1 גדולה מזמן הכתיבה של A (שהוא 2), הקריאה מתבצעת. במקביל לביצוע הכתיבה, מעדכנים את תוויית הקריאה של A ל-30 כי היא גדולה מתוויית הקריאה הנוכחית של A (5).

בזמן 33 התחילה T2 וקיבלה את תוויית הזמן 33. גם היא רוצה לקרוא את A. אין בעיה - קוראים את A ובמקביל מעדכנים את תוויית הקריאה של A ל-33.

בהמשך רוצה T1 לכתוב את A ואז מתברר שתוויית הזמן של T1 (30) קטנה מתוויית הקריאה של A (33). כזכור, בפעולת כתיבה חייבת תוויית הזמן של התנועה הכותבת להיות גדולה גם מתוויית הקריאה של הנתון וגם מתוויית הכתיבה של הנתון. אצלנו הדבר לא מתקיים עבור תוויית הקריאה. פירוש הדבר הוא שהתנועה T2 קראה, כביכול, ערך לא נכון של A, כי היה עליה לקרוא את הערך ש-T1 הייתה אמורה לכתוב. אם נרשה עכשיו ל-T1 לכתוב נקבל בסוף תוצאות שגויות (כלומר, תוצאות שאינן שקולות לתוצאות של הביצוע T1T2 או של T2T1). לכן צריך להתחיל את T1 מחדש (בזמן 37). למרבה המזל, T1 עוד לא הספיקה לכתוב כלום, אחרת, היינו צריכים לבטל את כל הכתיבות של T1 והדבר היה עלול לגרום לבעיות עם תנועות אחרות.

לאחר ש-T1 התחילה מחדש בזמן 37 היא שוב קוראת את A. כתוצאה מכך זמן הקריאה של A יהיה 37. שוב זוהי סיבה לצרה חדשה, מפני שבהמשך T2 תרצה לכתוב את A והכתיבה תיכשל כי תוויית הזמן של T2 קטנה מזמן הקריאה של A. לפנינו מצב דומה ל-deadlock, מפני שבכל פעם שאחת התנועות תגיע לכתובת A היא תיתקל באותה בעיה.

אפשר לפתור זאת, למשל, ע"י הגדלת מס' הפקודות שכל תנועה תבצע בכל סבב, ואכן אם נבצע את התנועות עם 4 פעולות בסבב - הכל מסתדר בפעם הראשונה.