ניהול תנועות
חלק 1

Transaction Management
Part 1

הקדמה: הבעיות

תוכנית הפועלת על מסד נתונים

1. תוכנית (אפליקציה):
   - קוראת נתונים ממסד הנתונים
   - מבצעת חישובים בזיכרון הפנימי המוקצה לה
   - כותבת נתונים חדשים למסד הנתונים
2. תוכנית נכתבת כך שתהיה נכונה בהנחה ש-
   - התוכנית רצה מתחילתה עד סופה ללא הפרעה, כלומר אינה עפה באמצע הריצה
   - התוכנית פועלת לבדה

תוכנית עפה באמצע פעולתה

- אם תוכנית עפה באמצע פעולתה, המסד עלול להישאר במצב לא עקבי
  - תוכנית מעבירה כסף מחשבון אחד לשני
    - התוכנית הורידה 100 ₪ מהחשבון הראשון
    - התוכנית עפה (או המחשב נפל) לפני שהיא הספיקה להוסיף 100 ₪ לחשבון השני
  - מסד הנתונים נשאר במצב שאינו עקבי!
- הפתרון: מנגנון שמאפשר גלגול לאחור (rollback) והתאוששות (recovery)

תוכניות רצות במקביל (concurrently)

- שתי תוכניות מזמינות מקום באותה טיסה
  - נשאר מקום אחד
  - כל תוכנית בודקת אם יש מקום פנוי ואח"כ מזמינה
  - כל אחת משתי התוכניות מבצעת את הבדיקה בהצלחה לפני שהתוכנית האחרת מספיקה להזמין המקום האחרון שנשאר
  - כל תוכנית הצליחה להזמין מקום, למרות שנשאר רק אחד!
- הפתרון: מנגנון לבקרת מקביליות (concurrency control)

המודל: הפשטה של תוכניות

למה צריך מודל?

- מסובך מדי להסתכל על הקוד של התוכניות שרצות על מסד הנתונים
  - למעשה, אין יכולת למצוא פתרון אם מסתכלים על הקוד (בעיה בלתי כריעה)
- לכן, מסתכלים על הפשטה (abstraction) של כל אחת מהתוכניות
  - למעשה, הפשטה של הפעולות שהתבצעו בפועל

דוגמה: תוכנית להזמנת מקום בטיסה

- נניח שיש במסד הנתונים פריט A, ששומר את מספר המקומות הפנויים
- התוכנית קוראת את הערך של A ממסד הנתונים
  - אם הערך גדול מאפס, התוכנית מקטינה את הערך של A באחד ומודיעה שההזמנה התבצעה
  - אם הערך של A שווה לאפס, התוכנית מודיעה שאי אפשר לבצע הזמנה

דוגמה לסידרת הפעולות של תוכנית המבצעת הזמנה בהצלחה

- Read A (from the DB)
- Is A>0 ?
- A := A –1
- Write A (into the DB)
- Print “Reservation confirmed”

- בצד שמאל מופיעות הפעולות שהתבצעו למעשה
- מכיוון שתוצאת הבדיקה הייתה ש- A גדול מאפס, התבצעו הפעולות של הקטנת ערכו של A והדפסת הודעה על ביצוע הזמנה

זוהי סידרת הפעולות המתבצעות כשאין מקום פנוי

- כשאין מקום פנוי (קרי, ערכו של A הוא אפס), אז אין צורך לשנות את הערך של A במסד

- Read A (from the DB)
- Is A>0 ?
- Print “No available seat”

סידרת הפעולות שהתוכנית מבצעת תלויה בערכים שהתוכנית קוראת ממסד הנתונים

פעולות על מסד הנתונים לעומת פעולות בזיכרון הפנימי של התוכנית

- Read A
- Is A>0 ?
- A := A –1
- Write A
- Print “Reservation confirmed”

- פעולות ה- Read וה- Write מתבצעות על מסד הנתונים
- שאר הפעולות מתבצעות בזיכרון הפרטי של התוכנית (כולל פעולות print וגם פעולות של קבלת נתונים מהמשתמש תוך כדי ריצה)

ההפשטה (אבסטרקציה) של התוכנית כוללת רק פעולות על מסד הנתונים

Read A
Write A

- זוהי ההפשטה כשמתבצעת הזמנה

Read A

- זוהי ההפשטה כשאין מקום פנוי

ההפשטה תלויה בערכים שהתוכנית קוראת ממסד הנתונים

ביצוע סידרתי לעומת ביצוע מקבילי

- למעשה יש תוכניות רבות שרצות בו זמנית מאתרים שונים ומנסות לבצע הזמנות
- ביצוע סדרתי פירושו שהתוכניות רצות אחת אחרי השנייה – כל פעם מתבצעת תוכנית אחת בלבד מתחילתה ועד סופה
- ביצוע מקבילי פירושו שהתוכניות מתבצעות בצורה משולבת, כלומר בכל נקודת זמן מתבצעת פעולה בודדת של אחת התוכניות

ביצוע סידרתי של שתי תוכניות להזמנת מקום

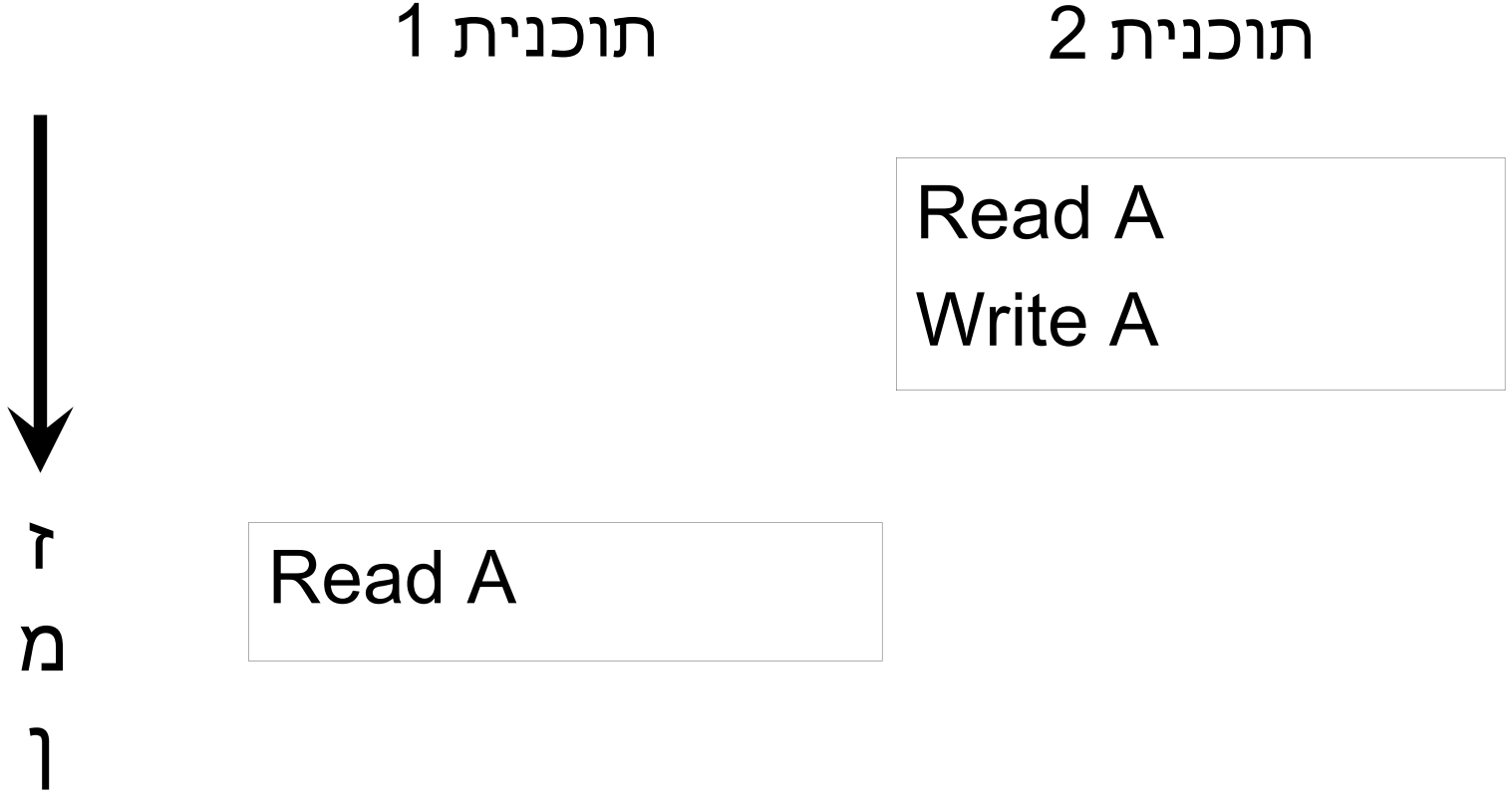
תוכנית 1 | תוכנית 2

זמן

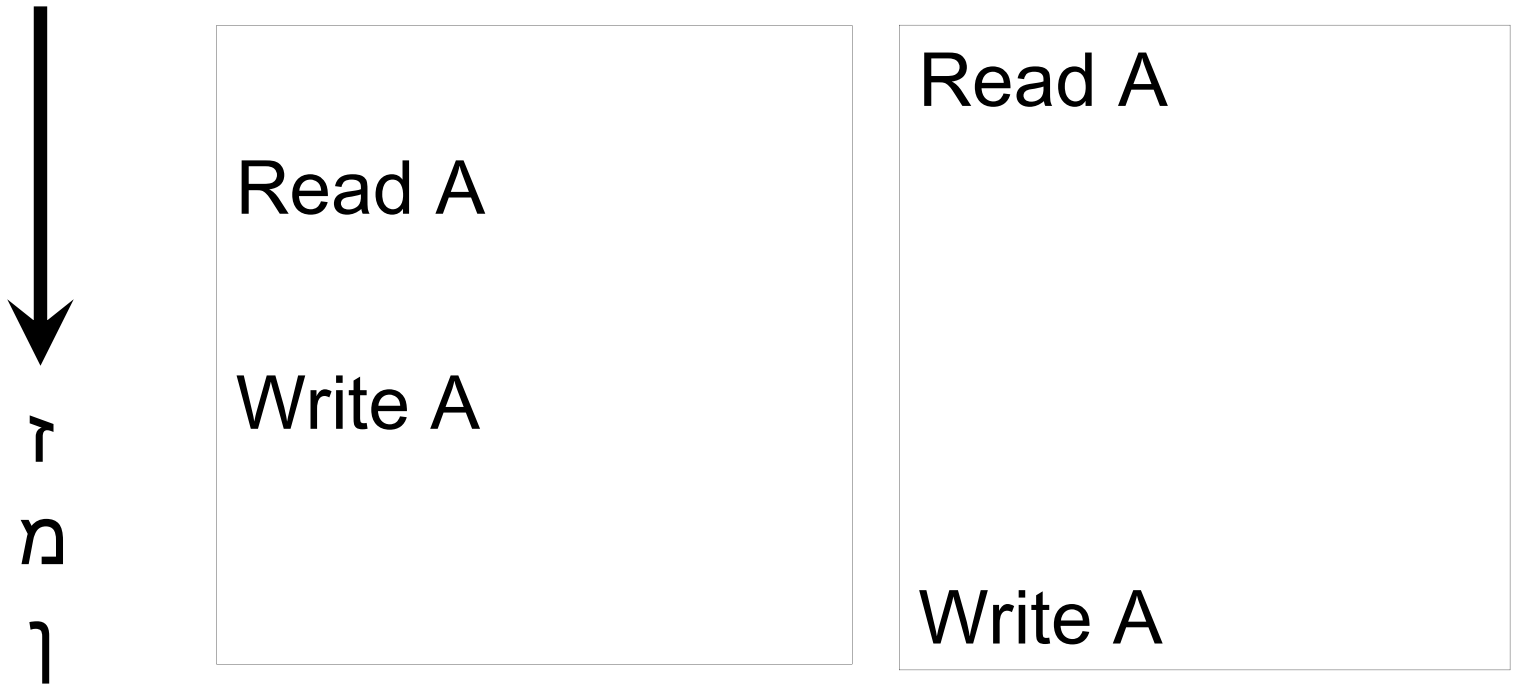
Read A
Write A

Read A
Write A

אפשרות נוספת של ביצוע סידרתי של שתי תוכניות להזמנת מקום



ביצוע מקבילי של שתי תוכניות להזמנת מקום



הגדרת הבעיה:
מתי ביצוע מקבילי הוא נכון

יתרונות של ביצוע מקבילי

- סיבות לביצוע מספר תוכניות במקביל:
  - אין סיבה להתעכב בגלל שתוכנית אחת מתעכבת (למשל, תוכנית מחכה לקלט מהמשתמש)
  - ניצול מקסימאלי של משאבי המחשוב, במטרה לסיים את ביצוע התוכניות מהר ככל האפשר
  - לדוגמה, משיכת כסף מבנקומט – מי רוצה לחכות?

הבעיה שמעורר ביצוע מקבילי (concurrent execution)

- תוכנית המתבצעת ע"י מערכת מסד נתונים אינה מודעת לכך שהיא מתבצעת במקביל לתוכניות אחרות
  - התוכנית נכתבה בהנחה שהיא מתבצעת לבדה
- המערכת (ולא המשתמש) חייבת לוודא שהביצוע המקבילי אינו גורם לשגיאות

דוגמה: העברת כסף משני חשבונות לחשבון שלישי

- חשבונות בנק A, B, C
- נעביר:
  - 100 ₪ מ- A ל- B
  - 100 ₪ מ- C ל- B
- מדובר למעשה בשתי תוכניות שמתבצעות במקביל (ראה דוגמה של ביצוע בשקף הבא)
  - הביצוע בדוגמה אינו נכון, כי בסיום מוסיפים 100 ₪ ל- B רק פעם אחת

למען הבהירות, הדוגמה בשקף הבא אינה הפשטה, אלא כוללת את כל הפעולות המתבצעות

ערכים התחלתיים במסד: A=900, B=500, C=100

Read(A)		A=900
A:=A-100		A=800
	Read(C)	C=100
Write(A)		A=800
	C:=C-100	C=0
Read(B)		B=500
	Write(C)	C=0
B:=B+100		B=600
	Read(B)	B=500
	B:=B+100	B=600
	Write(B)	B=600
Write(B)		B=600

ערכים מודגשים בקו נכתבים למסד

האם כל ביצוע מקבילי שגוי?

- כמובן שלא
- השקף הבא מראה ביצוע מקבילי של שתי התוכניות שרושם למסד את הנתונים הנכונים

ערכים התחלתיים במסד: A=900, B=500, C=100

Read(A)		A=900
A:=A-100		A=800
	Read(C)	C=100
Write(A)		A=800
	C:=C-100	C=0
Read(B)		B=500
	Write(C)	C=0
B:=B+100		B=600
Write(B)		B=600
	Read(B)	B=600
	B:=B+100	B=700
	Write(B)	B=700

ערכים מודגשים בקו נכתבים למסד

מתי ביצוע מקבילי הוא נכון?

- אם הוא שקול לביצוע סידרתי
- שימו לב שמספיק שהביצוע המקבילי יהיה שקול לביצוע סדרתי אחד
- בדוגמה שנתנו, לכל הביצועים הסדרתיים אותה תוצאה, לכן
  - אם ביצוע מקבילי שקול לביצוע סדרתי אחד, אז הוא שקול לכל הביצועים הסדרתיים
- אבל לא תמיד הדבר כך (דוגמה בשקף הבא)

נחזור לדוגמה של הזמנת מקום

- כשנשאר רק מקום אחד ורצות שתי תוכניות שמנסות לבצע הזמנה, אז התוכנית שרצה ראשונה תזכה במקום האחרון שנותר
- מבחינתנו אין זה משנה איזה תוכנית רצה ראשונה – שתי האפשרויות נכונות כל עוד שתי התוכניות רצות באופן סדרתי
- כאמור, ביצוע מקבילי הינו נכון אם הוא שקול לאחד הביצועים הסדרתיים
  - נובע מההנחות בסעיף 2 <u>בשקף 3</u>

הגדרת פורמאלית של המודל והמושגים הבסיסיים

המודל

- כיצד נבדוק האם ביצוע מקבילי הינו נכון?
  - התשובה עשויה להיות תלויה בחישוב המדויק שכל תוכנית מבצעת
- מסובך מדי (למעשה בלתי אפשרי) לנתח את החישוב המדויק שמבצעת כל תוכנית
- לכן, מתייחסים רק להפשטה של התוכניות, כלומר מתייחסים רק לדברים הבאים:
  - פעולות הקריאה מהמסד והכתיבה לתוך המסד
  - הסדר, על ציר הזמן, שבו מתבצעות פעולות אלה

צד ימין הוא ההפשטה (אבסטרקציה) של הביצוע

Read(A)	
A:=A-100	
	Read(C)
Write(A)	
	C:=C-100
Read(B)	
	Write(C)
B:=B+100	
	Read(B)
	B:=B+100
	Write(B)
Write(B)	

Read(A)	
	Read(C)
Write(A)	
Read(B)	
	Write(C)
	Read(B)
	Write(B)
Write(B)	

תנועה (או עסקה) Transaction

- תנועה (טרנסקציה) היא סדרה של פעולות קריאה וכתיבה שמתבצעות על *פריטים* (items) מהמסד
- תנועה היא הפשטה של ביצוע של תוכנית
- הפריטים יכולים להיות רשומות, שדות של רשומות, בלוקים וכד'

Read(A)
Write(A)
Read(B)
Write(B)

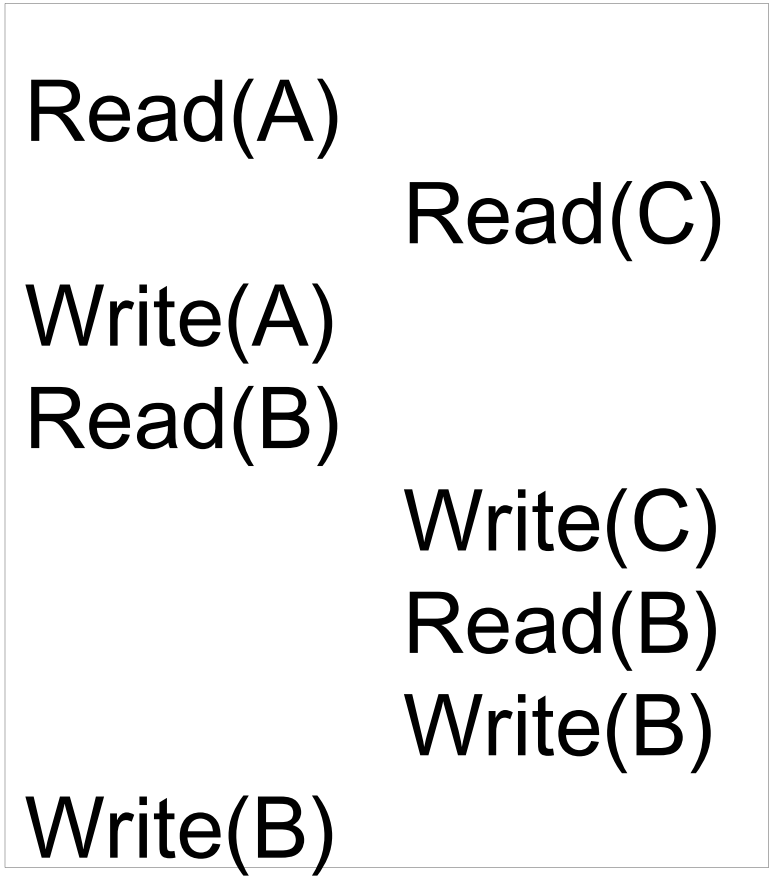
למען הפשטות נניח ש-

- תנועה לא קוראת אותו פריט יותר מפעם אחת
- תנועה לא כותבת אותו פריט יותר מפעם אחת
- אם תנועה גם קוראת וגם כותבת פריט A, אז התנועה מבצעת את הקריאה של A לפני הכתיבה של A
- <u>שימו לב</u>: תנועה יכולה רק לקרוא או רק לכתוב פריט A , וגם רשאית שלא לבצע אף פעולה על A

התזמון שומר, לגבי כל תנועה בנפרד,
על הסדר המקורי של הפעולות

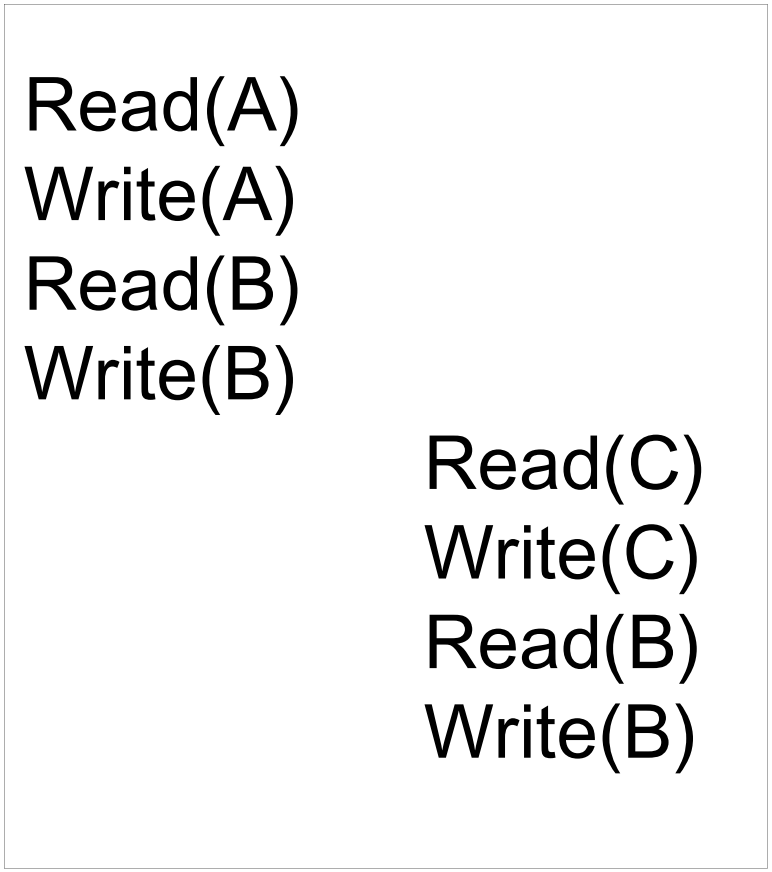
תזמון Schedule

- תזמון הוא ביצוע של מספר תנועות במקביל
- בכל נקודת זמן מתבצעת פעולה אחת בלבד
- פעולת קריאה/כתיבה בודדת מתבצעת בצורה אטומית
- התזמון מייצג את הסדר על ציר הזמן (הציר האנכי) בו מתבצעות הפעולות



תזמון סדרתי
Serial Schedule

- תזמון סדרתי הוא תזמון שבו התנועות מתבצעות זו אחר זו באופן סדרתי
- כל תזמון סדרתי הוא נכון



שקילות של תזמונים

הקדמה לא פורמלית

מתי תזמון שאינו סדרתי הוא נכון?

- אם הוא נותן אותה תוצאה כמו תזמון סדרתי כלשהו
  - מה זאת אומרת "אותה תוצאה"?
  - איך אפשר לבדוק שהוא נותן אותה תוצאה אם יש לנו רק הפשטה, שאינה כוללת את החישובים שכל תוכנית מבצעת?

איך אפשר להחליט האם שני התזמונים האלה נותנים אותה תוצאה?

S_1

T_1	T_2
Read(A)	
	Read(C)
Write(A)	
Read(B)	
	Write(C)
	Read(B)
	Write(B)
Write(B)	

S_2

T_1	T_2
Read(A)	
Write(A)	
Read(B)	
Write(B)	
	Read(C)
	Write(C)
	Read(B)
	Write(B)

- צריך להחליט אם הם נותנים אותה תוצאה מבלי להתייחס לחישוב פרטני (ספציפי) כלשהו
- כלומר, לכל סידרת פעולות חישוב שמבצעת כל אחת מהתנועות, שני התזמונים צריכים לתת אותה תוצאה

Read(A)	
	Read(C)
Write(A)	
Read(B)	
	Write(C)
	Read(B)
	Write(B)
Write(B)	

Read(A)	
Write(A)	
Read(B)	
Write(B)	
	Read(C)
	Write(C)
	Read(B)
	Write(B)

אפשר להראות שאין שקילות ע"י כך שמוצאים דוגמה נגדית

Read(A)	
	Read(C)
Write(A)	
Read(B)	
	Write(C)
	Read(B)
	Write(B)
Write(B)	

Read(A)	
Write(A)	
Read(B)	
Write(B)	
	Read(C)
	Write(C)
	Read(B)
	Write(B)

- בשקף הבא (שהוא זהה לשקף 21) בוחרים סידרת פעולות חישוב לכל תנועה, כך שהתוצאה של התזמון בצד שמאל אינה שקולה לתזמון בצד ימין
- למעשה התוצאה אינה שקולה לאף תזמון סדרתי

Read(A)	
	Read(C)
Write(A)	
Read(B)	
	Write(C)
	Read(B)
	Write(B)
Write(B)	

Read(A)	
Write(A)	
Read(B)	
Write(B)	
	Read(C)
	Write(C)
	Read(B)
	Write(B)

ערכים התחלתיים במסד: A=900, B=500, C=100

Read(A)		A=900
A:=A-100		A=800
	Read(C)	C=100
Write(A)		A=800
	C:=C-100	C=0
Read(B)		B=500
	Write(C)	C=0
B:=B+100		B=600
	Read(B)	B=500
	B:=B+100	B=600
	Write(B)	B=600
Write(B)		B=600

ערכים מודגשים בקו נכתבים למסד

דוגמה נוספת

- השקף הבא מתאר בצד שמאל תזמון (כולל פעולות חישוב) שנותן אותה תוצאה כמו תזמון סדרתי (ראו שקף 23)
- אבל האם גם ההפשטה בצד ימין נותנת, לכל חישוב אפשרי, אותה תוצאה כמו תזמון סידרתי?

צד ימין הוא התזמון המופשט עבור הביצוע בצד שמאל

Read(A)	
A:=A-100	
	Read(C)
Write(A)	
	C:=C-100
Read(B)	
	Write(C)
B:=B+100	
Write(B)	
	Read(B)
	B:=B+100
	Write(B)

Read(A)	
	Read(C)
Write(A)	
Read(B)	
	Write(C)
Write(B)	
	Read(B)
	Write(B)

צד ימין הוא התזמון המופשט עבור הביצוע בצד שמאל

Read(A)	
A:=A-100	
	Read(C)
Write(A)	
Read(B)	
	Write(C)
Write(B)	
	Read(B)
B:=B+100	
Write(B)	
	Write(B)

- צד שמאל (קרי, החישוב הפרטני המבוצע בצד זה) שקול לתזמון סדרתי
- אבל אולי אפשר לבחור, עבור כל אחת משתי התנועות בצד ימין, סדרת פעולות חישוב כך שהתוצאה לא תתקבל ע"י אף תזמון סדרתי?

תרגיל

- נתונות שתי תנועות
  - אחת מעבירה 100 ₪ מחשבון A לחשבון B
  - השנייה מעבירה 200 ₪ מחשבון B לחשבון A
- כיתבו תזמון לא סדרתי של שתי התנועות כך ש-
  - אם מביאים בחשבון את פעולות החישוב המדויקות, אז התזמון משנה את הערכים במסד כמו תזמון סדרתי
  - אם מסתכלים רק על ההפשטה, אז לא לכל חישוב אפשרי יש אותה השפעה על המסד כמו תזמון סדרתי

המודל מספק תנאי מספיק אבל לא הכרחי

- שני תזמונים יכולים להיות שקולים אם מביאים בחשבון את החישובים שהם עושים, אבל לא להיות שקולים אם מסתכלים רק על ההפשטה
- ההיפך אינו נכון, כלומר
  - אם יש שקילות כאשר מסתכלים על ההפשטה, אז יש שקילות עבור החישובים המקוריים (כי אין דוגמה נגדית)
- **<u>מסקנה</u>**: שקילות במודל המופשט היא תנאי מספיק אבל לא הכרחי לשקילות עבור הבעיה המקורית

סיכום

- כדי להראות שתזמון אינו נכון, צריך להוכיח שלכל תזמון סדרתי של אותן עסקאות אפשר למצוא פעולות חישוב, כך ששני התזמונים אינם שקולים
- כלומר, לכל אחד מ- !n התזמונים הסדרתיים (כאשר n הוא מספר העסקאות), צריך למצוא פעולות חישוב שמראות שאין שקילות (בין התזמון הסדרתי למקורי)
  - אולי לכל אחת מ- !n האפשרויות, צריך למצוא פעולות חישוב אחרות כדי להראות שהתזמון אינו נכון

תזמונים שקולים
Equivalent Schedules

דיון פורמלי

ראשית

- אין משמעות לשקילות של שני תזמונים אלא אם כן בשניהם
  - אותן עסקאות
  - ולכל עסקה, אותה סדרת פעולות

כמובן שזה צריך להתקיים לכל חישוב אפשרי

הגדרה ראשונית

- שני תזמונים שקולים אם הם מייצרים אותה תוצאה
  - כלומר, בסיום כל אחד מהתזמונים, למסד יש אותם ערכים (בהנחה שגם בהתחלה היו למסד אותם ערכים)
- צריך גם לדרוש שכל עסקה קוראת אותם ערכים בכל אחד משני התזמונים
  - המשתמש מעוניין לדעת מה הערכים שהעסקה קוראת, לכן לא יכולה להיות שקילות אם העסקה קוראת ערכים שונים בכל תזמון

שקילות מבטים
View Equivalence

שקילות מבטים היא המושג הכללי ביותר
של שקילות בין תזמונים

הגדרה של שקילות מבטים

- תזמונים S_1 ו- S_2 הינם *שקולי מבטים* אם:
  - S_1 ו- S_2 מורכבים מאותן תנועות (ולכל תנועה אותה סידרת פעולות בשני התזמונים)
  - אם T_k קוראת את הערך ההתחלתי של A ב- S_1, אז T_k קוראת גם ב- S_2 את הערך ההתחלתי של A
  - אם T_k קוראת ערך של A שנכתב ע"י T_i ב- S_1, אז T_k קוראת גם ב- S_2 ערך של A שנכתב ע"י T_i
  - אם T_i כותבת ערך סופי של A ב- S_1, אז T_i כותבת גם ב- S_2 ערך סופי של A

לשני התזמונים אותו יחס של "קוראת מ" ואותו יחס של "כותבת ערך סופי למסד"

- תזמונים S_1 ו- S_2 הינם *שקולי מבטים* אם:
  - S_1 ו- S_2 מורכבים מאותן תנועות (ולכל תנועה אותה סידרת פעולות בשני התזמונים)
  - אם T_k קוראת את הערך ההתחלתי של A ב- S_1, אז T_k קוראת גם ב- S_2 את הערך ההתחלתי של A
  - אם T_k קוראת ערך של A שנכתב ע"י T_i ב- S_1, אז T_k קוראת גם ב- S_2 ערך של A שנכתב ע"י T_i
  - אם T_i כותבת ערך סופי של A ב- S_1, אז T_i כותבת גם ב- S_2 ערך סופי של A

במילים אחרות

- שקילות מבטים פירושה שלשני התזמונים יש אותו יחס של "קוראת מ" ואותו יחס של "כותבת ערך סופי למסד"
- ההבדל בין ההגדרה הכללית ביותר (שקף 48) להגדרה של שקילות מבטים הוא בכך
  - שהראשונה דורשת קריאת אותו ערך (עבור כל עסקה וכל פריט שהעסקה קוראת)
  - בעוד שהשנייה דורשת קריאת ערך מאותו מקור (ולכן באינדוקציה אפשר להראות שקוראת גם אותו ערך)
  - ובאופן דומה לגבי כתיבת ערך סופי למסד
- אפשר להראות שההגדרה של שקילות מבטים שקולה להגדרה הכללית של שקף 48
  - משתמשים בכך ששקילות חייבת להתקיים לכל חישוב אפשרי

דוגמה לשני תזמונים המקיימים את ההגדרה של שקילות מבטים

T_1	T_2	T_3
R(A)		
	W(A)	
W(A)		
		W(A)

T_1	T_2	T_3
R(A)		
W(A)		
	W(A)	
		W(A)

- יש רק פעולת קריאה אחת
  - בשני התזמונים, T_1 קוראת ערך התחלתי של A מהמסד
- A הוא הפריט היחיד ובשני התזמונים T_3 כותבת את ערכו הסופי למסד

מסקנות הנובעות מההגדרה של שקילות מבטים

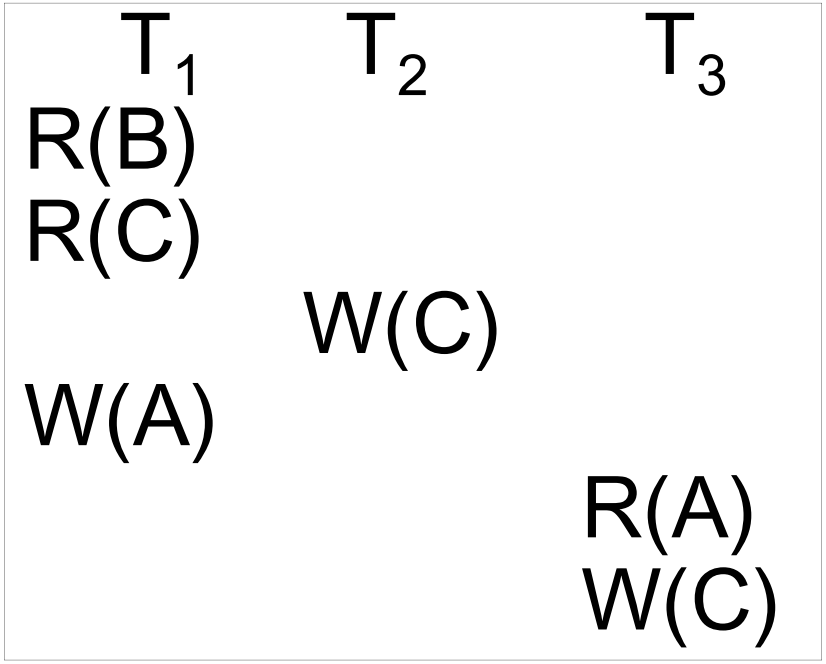
- ההגדרה של שקילות מבטים מבטיחה שבכל חישוב אפשרי, כל תנועה T_i קוראת אותם ערכים בשני התזמונים, ולפיכך
  - מייצרת אותה סדרת פעולות בשני התזמונים
    - ← כפי שאומנם צריך להתקיים בין שני תזמונים שקולים (ראו שקף 47)
  - וכותבת אותם ערכים למסד בשני התזמונים
- לערכים שתנועה T_i כותבת יש השפעה רק אם אלה ערכים סופיים שנכתבים למסד או שתנועה אחרת קוראת אותם

כתיבה עיוורת (Blind Write)

- תנועה מבצעת כתיבה עיוורת אם היא כותבת פריט מבלי לקרוא אותו
- העסקה בצד ימין מבצעת כתיבה עיוורת של הפריט B
- אבל הערך שנכתב עבור B תלוי בערכים שנקראו עבור A ו- C

```
Read(A)
Write(A)
Read(C)
Write(B)
```

דוגמה של תנועה חסרת השפעה



- בדוגמה בצד שמאל כל הכתיבות הן עיוורות
- הערך של C שנכתב ע"י T_2 אינו נקרא ע"י אף תנועה אחרת ואינו הערך הסופי של C שנכתב למסד
- בנוסף, T_2 אינה קוראת אף ערך; לפיכך:
- ל- T_2 אין שום השפעה – אפשר פשוט למחוק אותה מהתזמון

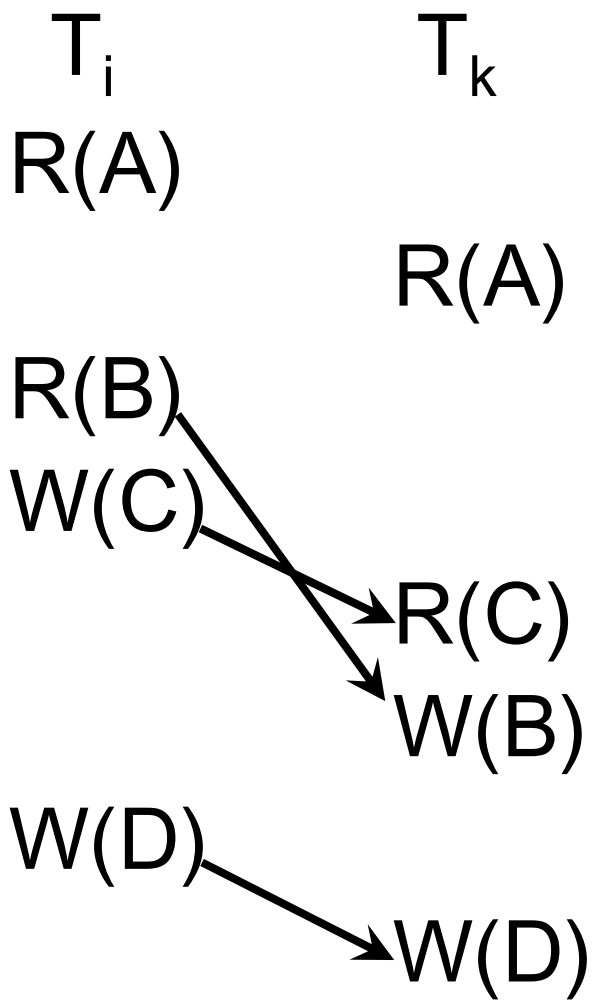
תרגיל

- כיתבו דוגמה של תזמון שבו תנועה חסרת השפעה, אבל לא כל פעולות הכתיבה המתבצעות בו הן עיוורות
- בהינתן שני תזמונים, כמה זמן לוקח לבדוק שקילות מבטים?
- כמה זמן לוקח לבדוק האם תנועה הינה חסרת השפעה?

קונפליקטים בין תנועות

קונפליקטים בין תנועות

- בין שתי פעולות, שלפחות אחת מהן היא פעולת כתיבה, קיים קונפליקט אם הן מתבצעות על אותו פריט ע"י שתי תנועות שונות
- בדוגמה מימין יש 3 קונפליקטים



הרעיון הבסיסי

- קונפליקט קובע סדר בין שתי פעולות המתבצעות על אותו פריט ע"י תנועות שונות
- אפשר לטעון שהסדר הזה חייב להישמר בכל תזמון סדרתי שקול
  - אבל האם הטענה נכונה?

שקילות קונפליקטים

הגדרה של שקילות קונפליקטים
Conflict Equivalence

- שני תזמונים הינם *שקולי קונפליקטים* אם
  - הם מורכבים מאותן תנועות (ולכל תנועה אותה סדרת פעולות בשני התזמונים)
  - כל צמד של פעולות *שיש* ביניהן קונפליקט מופיע בשני התזמונים באותו סדר
- *הערה:* הקונפליקטים הם
  - קריאה ולאחריה כתיבה על אותו פריט
  - כתיבה ולאחריה קריאה על אותו פריט
  - כתיבה ולאחריה כתיבה נוספת על אותו פריט

תרגיל

- בהינתן שני תזמונים, כמה זמן לוקח לבדוק שקילות קונפליקטים?

שקילות קונפליקטים גוררת שקילות מבטים

- <u>למה 1</u>: אם התזמונים S_1 ו- S_2 הינם שקולי קונפליקטים, אז הם גם שקולי מבטים
- <u>הערה</u>: הלמה נכונה גם אם יש כתיבות עיוורות
- <u>הוכחה</u>: תרגיל

תזמונים ברי-סדרתיות מבטית
View-Serializable Schedules

התזמונים ברי-סדרתיות

הינם נכונים

תזמון בר-סדרתיות מבטית

- <u>הגדרה</u>: תזמון הינו *בר-סדרתיות מבטית* אם הוא שקול מבטים לתזמון סדרתי כלשהו
- תזמון בר-סדרתיות הוא תזמון נכון
- זו לא רק בדיקה של שקילות, כי לא נתון התזמון הסדרתי השקול אם יש כזה
- צריך לגלות אם קיים תזמון סדרתי שקול

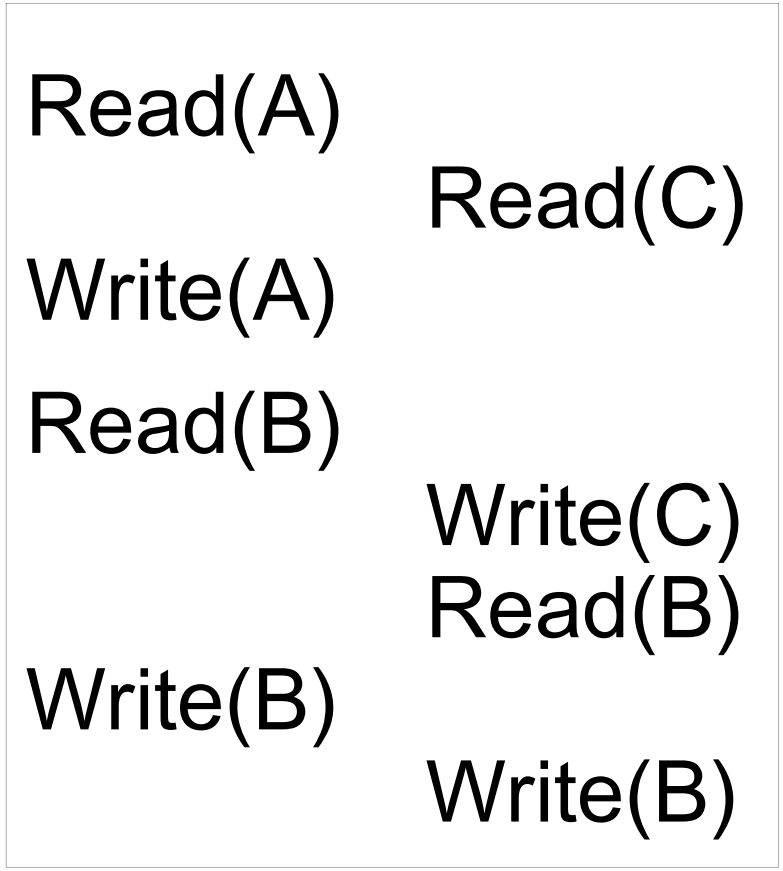
תזמון בר-סדרתיות והתזמון הסדרתי השקול לו

Read(A)	
	Read(C)
Write(A)	
Read(B)	
	Write(C)
Write(B)	
	Read(B)
	Write(B)

Read(A)	
Write(A)	
Read(B)	
Write(B)	
	Read(C)
	Write(C)
	Read(B)
	Write(B)

תזמון שאינו בר-סדרתיות מבטית

- אין תזמון סדרתי שקול
- שתי התנועות קוראות עבור B את הערך המקורי שנמצא במסד לפני שהן משנות אותו – דבר בלתי אפשרי בתזמון סדרתי



עוד תזמון בר-סדרתיות
והתזמון הסדרתי השקול לו

T_1	T_2
	Read(C)
	Write(C)
Read(A)	
Write(A)	
Read(B)	
Write(B)	
	Read(B)
	Write(B)

T_1	T_2
Read(A)	
Write(A)	
Read(B)	
Write(B)	
	Read(C)
	Write(C)
	Read(B)
	Write(B)

בדוגמה מהשקף הקודם

- התזמון המקורי שקול לתזמון הסדרתי למרות
  - שבתזמון המקורי T_2 קוראת וכותבת את C לפני ש- T_1 קוראת וכותבת את A
  - ובתזמון הסדרתי T_2 קוראת וכותבת את C אחרי ש- T_1 קוראת וכותבת את A
- אבל בשני התזמונים
  - T_1 קוראת וכותבת את B לפני ש- T_2 קוראת וכותבת את B

תזמונים ברי-סדרתיות קונפליקטית
Conflict-Serializable Schedules

קל לבדוק אם תזמון הינו
בר-סדרתיות קונפליקטית

הגדרה

- תזמון S_1 הינו בר-*סדרתיות קונפליקטית* אם קיים תזמון סדרתי S_2 כך ששני התזמונים הינם שקולי קונפליקטים

בר-סדרתיות קונפליקטית גוררת בר-סדרתיות מבטית

- *<u>למה 2</u>:* אם תזמון S הינו בר-סדרתיות קונפליקטית, אז S הוא גם בר-סדרתיות מבטית (כלומר, S שקול לתזמון סדרתי לפי ההגדרה של שקילות מבטים)
- המסקנה נובעת מלמה 1

אם אין כתיבות עיוורות, אז שני המושגים שקולים

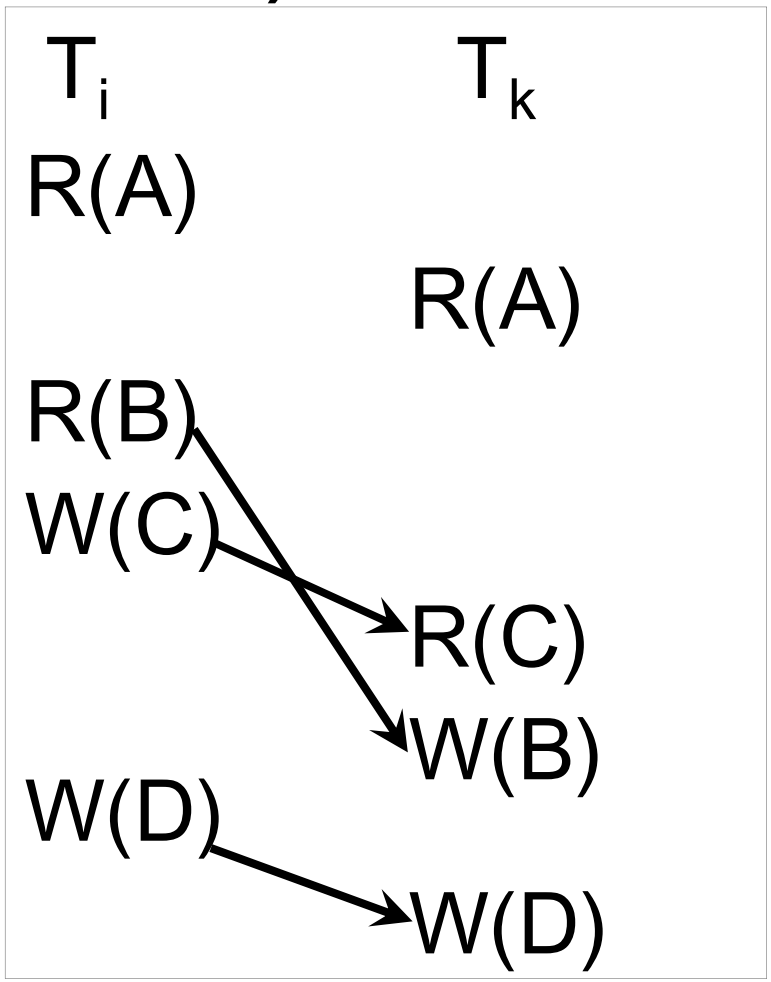
- *<u>משפט</u>:* אם S הינו תזמון ללא כתיבות עיוורות, אז S הינו בר-סדרתיות מבטית אם ורק אם S הינו בר-סדרתיות קונפליקטית
- *<u>הוכחה</u>:* תרגיל

הערה

- אם אין כתיבות עיוורות, אז שני הסוגים של בר סדרתיות (קרי, מבטית וקונפליקטית) זהים
- אבל זה אינו נכון לשני הסוגים של שקילות, כלומר שקילות מבטים ושקילות קונפליקטים אינן אותו דבר אפילו אם אין כתיבות עיוורות

קונפליקטים בין תנועות (דוגמה קודמת)

- בדוגמה מימין יש 3 קונפליקטים, ולפי כל אחד מהם T_i חייבת להופיע לפני T_k בכל תזמון סדרתי שקול
- לכן, התזמון הסדרתי שבו T_i מתבצעת ראשונה ו- T_k שנייה הנו שקול לתזמון הנתון (אם אין תנועות נוספות)



אלגוריתם לבדיקת
בר-סדרתיות קונפליקטית

גרף הקדימויות
Precedence Graph

- גרף הקדימויות עבור תזמון נתון מכיל
  - צומת לכל תנועה
  - צלע מכוונת מ- T_i ל- T_k אם יש קונפליקט בין T_i לבין T_k שקובע ש- T_i צריכה להופיע לפני T_k
- *<u>משפט</u>:* תזמון הנו בר-סדרתיות קונפליקטית אם ורק אם גרף הקדימויות שלו הנו חסר מעגלים

הוכחת המשפט

- אם אין מעגל, נבצע מיון טופולוגי
- תזמון סדרתי, שבו העסקאות מבוצעות בזו אחר זו לפי הסדר של המיון הטופולוגי, שקול (לפי ההגדרה של שקילות קונפליקטים) לתזמון המקורי
  - כי כל צמד פעולות שיש ביניהן קונפליקט מופיע באותו הסדר בשני התזמונים
- כדי להוכיח את הכיוון ההפוך נניח שיש מעגל

אם יש מסלול $T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_4$ אז נובע ש-

- T_1 חייבת להופיע לפני T_2 בכל תזמון סדרתי שהוא שקול קונפליקטים לתזמון המקורי
- T_2 חייבת להופיע לפני T_3 בכל תזמון סדרתי שהוא שקול קונפליקטים לתזמון המקורי
- T_3 חייבת להופיע לפני T_4 בכל תזמון סדרתי שהוא שקול קונפליקטים לתזמון המקורי
- ובאופן טרנזיטיבי:
  - T_1 חייבת להופיע לפני T_4 בכל תזמון סדרתי שהוא שקול קונפליקטים לתזמון המקורי

לפיכך, אם יש מעגל

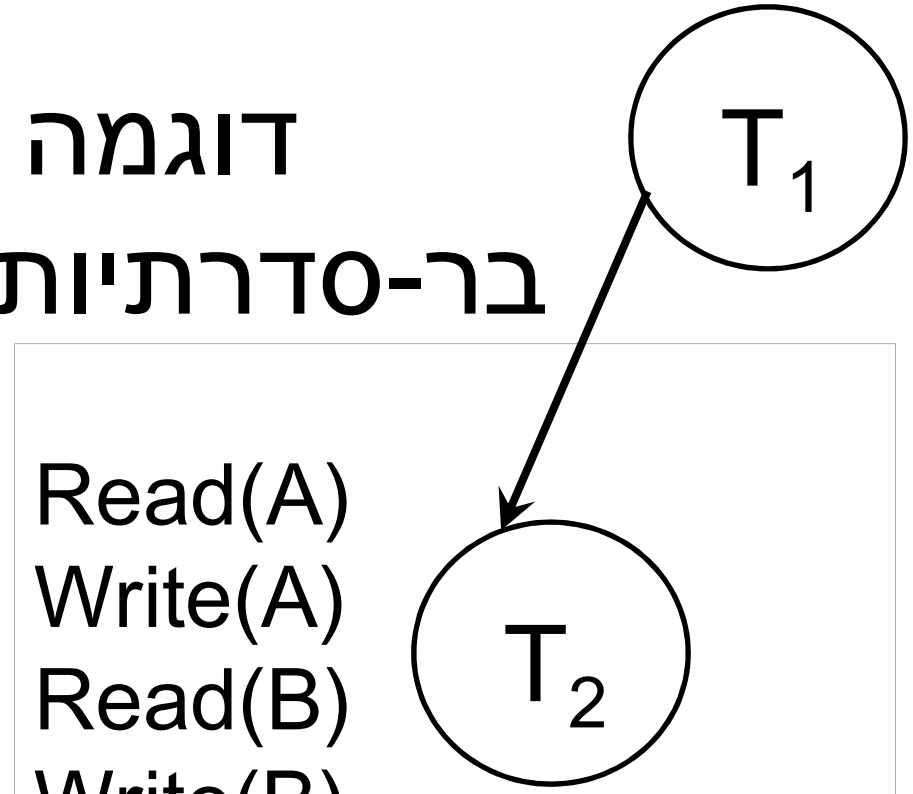
$$T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_4 \rightarrow T_1$$

אז נובע ש-

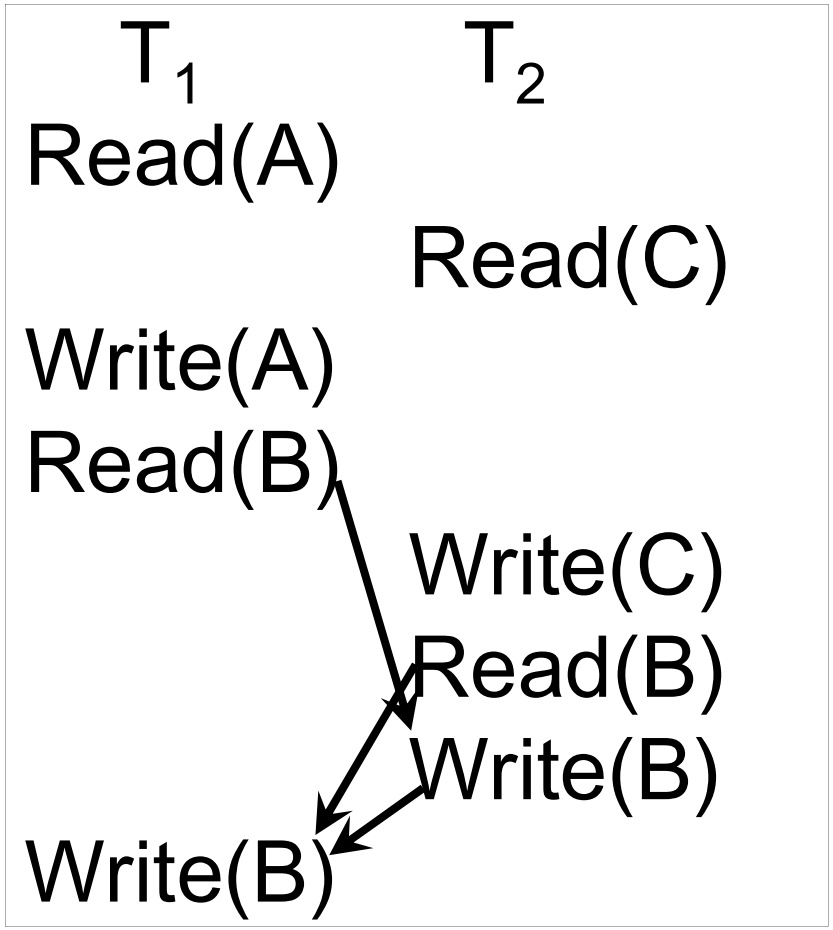
- T_1 חייבת להופיע לפני T_1 בכל תזמון סדרתי שהוא שקול קונפליקטים לתזמון המקורי
- מכיוון שהדבר בלתי אפשרי, נובע שאין תזמון סדרתי שהוא שקול קונפליקטים לתזמון המקורי
- מ.ש.ל.

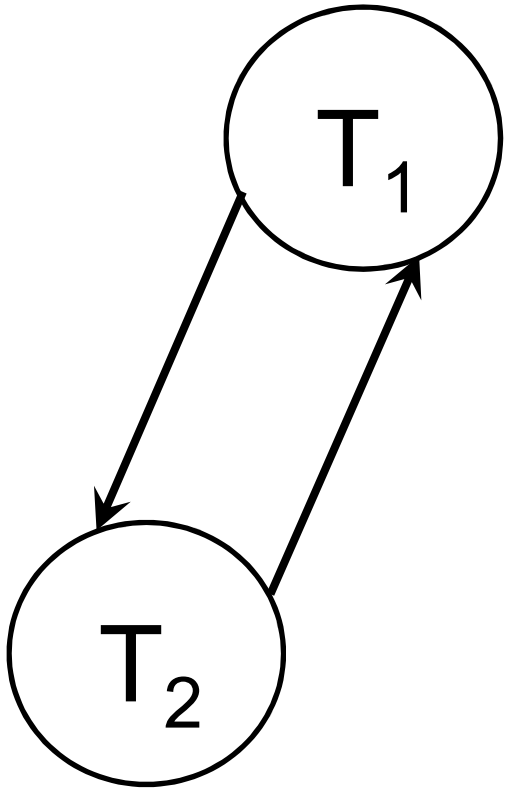
דוגמה לתזמון בר-סדרתיות קונפליקטית

T_1	T_2
Read(A)	
	Read(C)
Write(A)	
Read(B)	
	Write(C)
Write(B)	
	Read(B)
	Write(B)

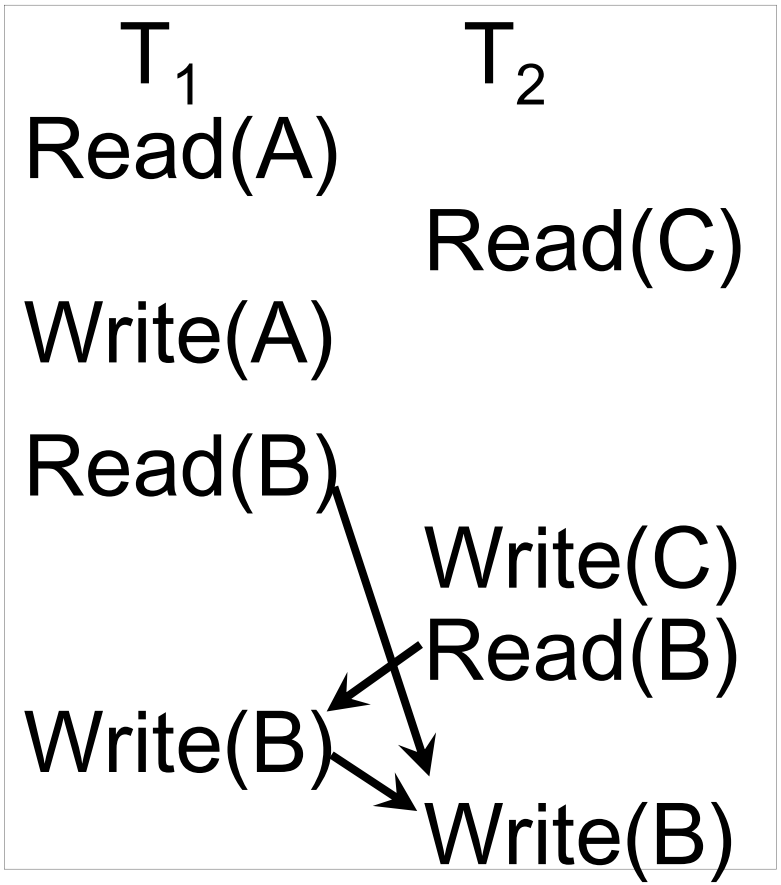


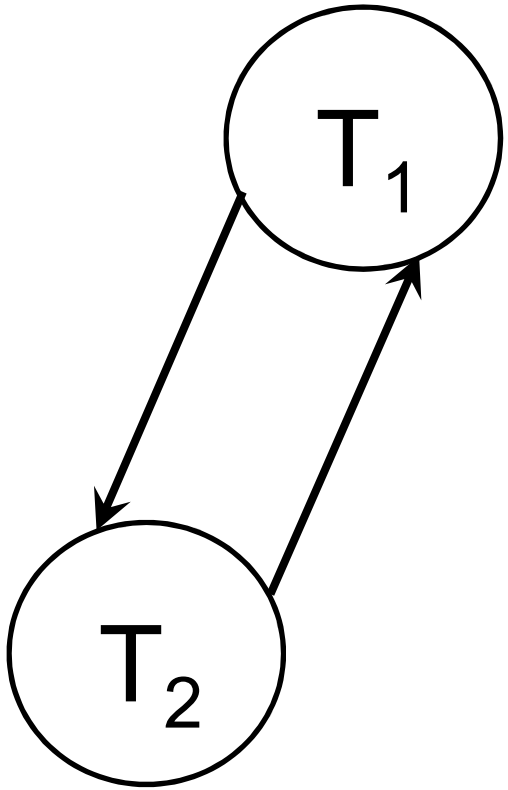
דוגמה לתזמון שאינו בר-סדרתיות קונפליקטית





עוד דוגמה לתזמון שאינו בר-סדרתיות קונפליקטית



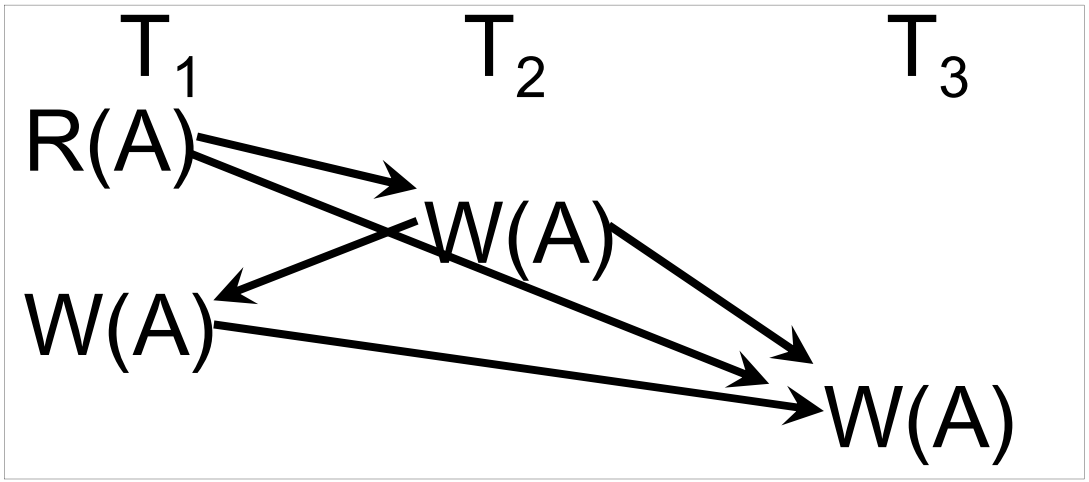


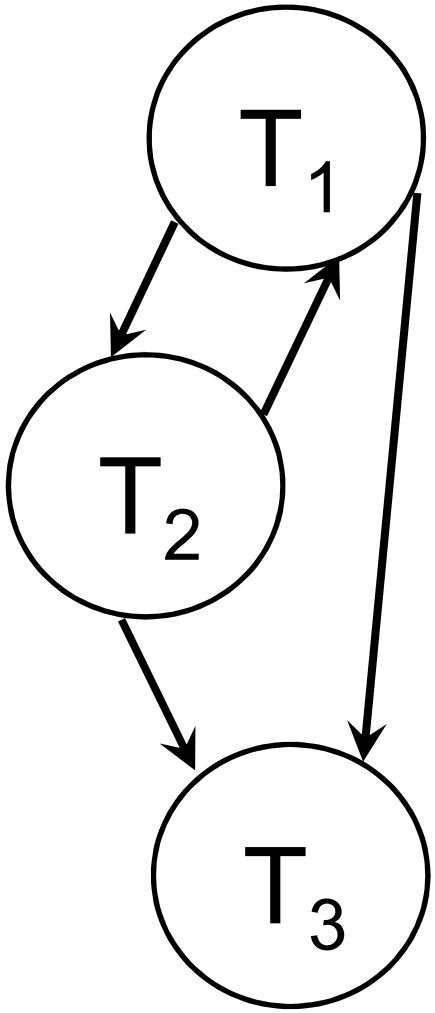
דוגמה לתזמון שהוא
בר-סדרתיות מבטית אבל
אינו בר-סדרתיות קונפליקטית

בר-סדרתיות מבטית
View Serializability

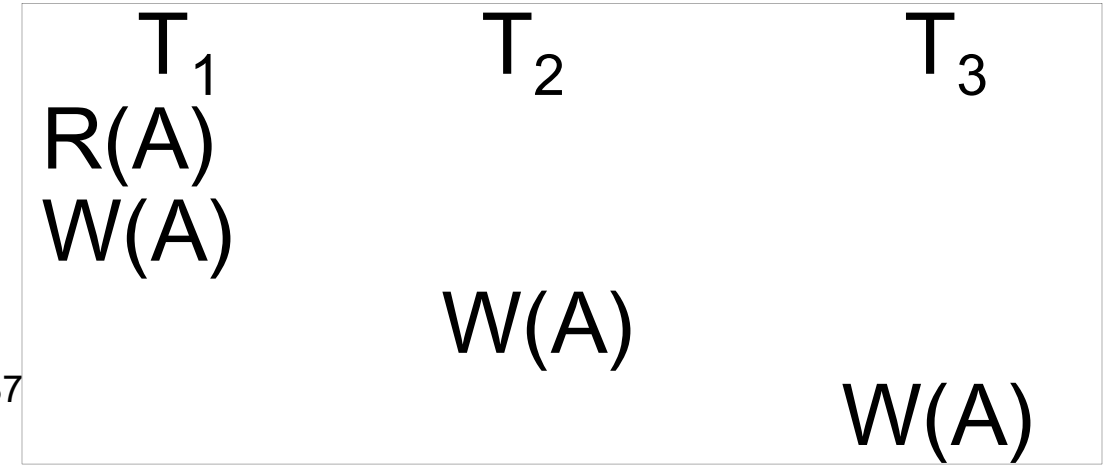
- כזכור, תזמון S_1 הנו *בר-סדרתיות מבטית* (view serializable) אם קיים תזמון סדרתי S_2 כך ששני התזמונים הינם שקולי מבטים
- לפי <u>למה 2</u>, תזמון בר-סדרתיות קונפליקטית הוא גם בר-סדרתיות מבטית (אבל ההיפך אינו בהכרח נכון כאשר יש כתיבות עיוורות)
- כאשר יש כתיבות עיוורות, קשה לבדוק האם תזמון S הנו בר-סדרתיות מבטית (זאת בעיה NP שלמה)

דוגמה לתזמון שהוא בר-סדרתיות מבטית אבל לא בר-סדרתיות קונפליקטית





תזמון סדרתי שקול מבטים:



שני התזמונים שקולי מבטים, כי

T_1	T_2	T_3
R(A)		
	W(A)	
W(A)		
		W(A)

T_1	T_2	T_3
R(A)		
W(A)		
	W(A)	
		W(A)

- בשניהם T_1 קוראת ערך התחלתי של A מהמסד
- אף עסקה, מלבד T_1, לא קוראת ערך של A
- בשני התזמונים T_3 כותבת ערך סופי של A למסד (זאת כתיבה עיוורת)
- אין פריטים נוספים חוץ מ- A

תרגיל

- בדוגמה הקודמת, T_2 היא חסרת השפעה ואם מוחקים אותה נשאר תזמון סדרתי
- מצאו דוגמה של תזמון שהוא בר-סדרתיות מבטית, אבל לא בר-סדרתיות קונפליקטית, כך שאין בו תנועות חסרות השפעה

התאוששות מנפילות

זוהי רק הקדמה קצרה ואינטואיטיבית
של המושגים הבסיסיים

דוגמה: העברת כסף מחשבון לחשבון

- חשבונות בנק A, B
- נעביר 100 ₪ מ- A ל- B
- התוכנית: A:=A-100

  B:=B+100
- התוכנית נכונה, אבל מה קורה אם המחשב נופל אחרי ביצוע הפקודה הראשונה?
  - הורדנו כסף מ- A ולא הוספנו ל- B
  - התוכנית הסתיימה בצורה שגויה

ביצוע אטומי של תוכנית

- תוכנית הפועלת על מסד נתונים צריכה להתבצע באופן *אטומי*, כלומר
  - התוכנית מתבצעת בשלמותה, או
  - שאינה מתבצעת כלל
- תוכנית *מתחייבת* (מבצעת *commit*) לאחר שביצעה את כל פעולותיה
  - אם התוכנית התחייבה, אז כל פעולותיה נשמרות
  - אחרת, שום פעולה שביצעה לא נשמרת במסד
- בתזמון אפשר לציין גם את נקודת ההתחייבות ע"י C

מנגנון התאוששות (Recovery)

- לכל מערכת מסד נתונים מנגנון התאוששות שאחראי על הדברים הבאים:
  - לבטל את כל הפעולות של התוכנית אם היא נפלה (abort) לפני שביצעה commit
  - לוודא שכל הפעולות של התוכנית נשמרות במסד אם התוכנית ביצעה commit
- מאוחר יותר נסביר איך פועל מנגנון ההתאוששות

נתון מלוכלך (Dirty Data)

- נתון שעסקה כותבת למסד לפני שהתחייבה הינו "מלוכלך"
- קריאה מלוכלכת (dirty read) הינה קריאת נתון שנכתב ע"י עסקה אחרת שטרם התחייבה