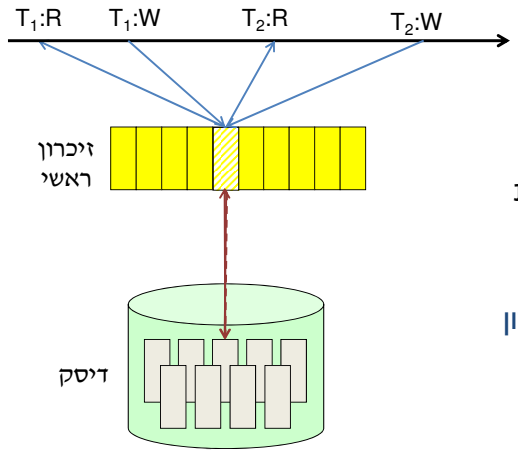
התאוששות מנפילות

תזכורת – שימוש במטמון לכתיבה ולקריאה

- שימוש בזיכרון הראשי על מנת לחסוך גישות לדיסק
- כתיבה מתבצעת בזיכרון ומעודכנת בדיסק מאוחר יותר
  - מתי?
- קריאת בלוק שנמצא בזיכרון חוזרת ללא גישה לדיסק (Cache hit)

- כל התנועות "רואות" את אותו זיכרון
- השימוש בזיכרון "שקוף" לתנועות



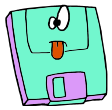
נפילות (failures)

סיבות לנפילה:

1. התנועה מפסיקה את עצמה (שגיאות אריתמטיות, יוזמת המשתמש, גישה לנתונים ללא הרשאה...).
2. הפסקה של תנועות (aborts) ביוזמת בקר המקביליות, למנוע חבק או להבטיח תזמון בר-סידור.
3. בעיות מפעיל: טעויות אנוש של טכנאי (למשל, מוחק את כל מסד הנתונים).
4. בעיות חומרה: תקלת דיסק, בקר, CPU...
5. תקלות חיצוניות: הפסקת חשמל, שריפה...

תוצאות הנפילה:

1. system failure — אובדן הזיכרון בר-החלוף (volatile) זיכרון ראשי, רגיסטרים.
1. media failure — אובדן הזיכרון הקבוע (stable, nonvolatile, persistent) דיסק, סרט מגנטי.



בעיות הנובעות מנפילות

- כתיבות של תנועה שלא סיימה מופיעות בדיסק
  - איך זה קורה? מה הבעיה?
- כתיבות של תנועה שסיימה לא מופיעות בדיסק
  - איך זה קורה? מה הבעיה?

איך מבטיחים תכונות ACID כאשר עלולות להיות נפילות?

- אלו תכונות ACID הובטחו על-ידי מנעולים או פרוטוקול 2PL?
- כיצד ניתן להבטיח Atomicity?
- כיצד ניתן להבטיח Durability?

נקודות התחייבות (commit)

לכל תנועה נגדיר נקודת התחייבות (commit)
- אם תנועה נפלה או הופסקה לפני ההתחייבות, כאילו שלא נעשתה כלל
- אחרי ההתחייבות התנועה לא תופסק, ותוצאותיה תישמרנה במסד הנתונים

בזמן נפילת המערכת:
- כל התנועות שלא התחייבו מופסקות
- הדיסק לא נמצא במצב עקבי (consistent):
  - תנועות שלא התחייבו ביצעו כתיבות
  - תנועות שהתחייבו טרם ביצעו את כל הכתיבות שלהן

עלינו להביא את הדיסק למצב עקבי:
- לבטל את התוצאות של תנועות שהופסקו
- לוודא שתוצאת הפעולות שהתחייבו נמצאת בדיסק

נעילה דו-שלבית חמורה Strict 2PL

תנועה מבצעת את כל הנעילות שלה לפני ההתחייבות ומשחררת את כל המנעולים שלה **אחרי ההתחייבות**.

⇦נעילה דו-שלבית.

משפט: אם כל התנועות בתזמון מקיימות את התנאים של נעילה דו-שלבית חמורה, אז אין צורך להתיר תנועות שביצעו התחייבות.

```
XLOCK A
WRITE A
SLOCK B
READ B
XLOCK C
WRITE C
COMMIT
UNLOCK A
UNLOCK C
UNLOCK B
```

מעתה נניח שאנו מבצעים נעילה דו-שלבית חמורה.

פעולות כתיבה שנעשו לפני ההתחייבות עלולות להתבטל, לכן
- יבוצעו להעתק, והכתיבה האמיתית תתבצע אחרי התחייבות
- או שנאפשר להתיר אותן (למשל, נזכור את הערך הישן)

קריאה מלוכלכת (dirty read)

קריאת נתונים שנכתבו ע"י תנועה T לפני ההתחייבות שלה.

אם T1 נפלה ובוטלה, ו-T2 קראה נתונים ש-T1 כתבה, יש להפסיק גם את T2!

התוצאה היא מפולת של הפסקות.

כיוון שזה קשה/יקר למימוש, רצוי לאסור קריאות מלוכלכות.

T1	T2
LOCK A	
READ A	
WRITE A	
LOCK B	
UNLOCK A	
	LOCK A
	READ A
READ B	
	WRITE A
	COMMIT
	UNLOCK A
ABORT	

תנועות שהופסקו מיוזמתן

כאשר תנועה מבצעת abort עליה:
- לשחזר את כל הנתונים שהיא כתבה.
- **רק אחר כך**, לשחרר את כל המנעולים שהיא לקחה.

לכן
- אם תנועה קראה נתון A של תנועה שהופסקה, היא הצליחה לנעול את A.
- מכאן שהתנועה שהופסקה שחררה את המנעול,
- מכאן שהתנועה שהופסקה שחזרה את הערך שלפני תחילת פעולתה.

לעומת זאת, תנועות שהופסקו בגלל נפילת המערכת אינן יכולות לשחזר את הנתונים שעליהם הן כתבו.

שמירת נתונים

כתיבה של תנועה **מתבצעת לחוצץ**, שנכתב מאוחר יותר לדיסק.

- גם אם התנועה ביצעה כתיבה, היא לא בהכרח מופיעה בדיסק.
- נוצרת בעיה אם חוצצים שלא נכתבו לדיסק אובדים עם נפילת המערכת.

שומרים מידע כדי לשחזר את הכתיבה (תוכן הסקטור) (after information).

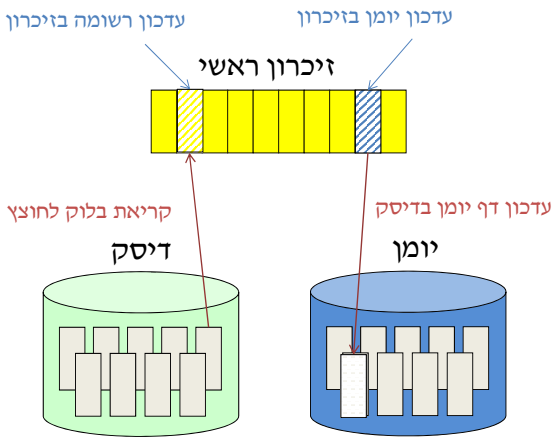
צריך לשמור נתונים שיאפשרו להתיר תנועות שהופסקו:
לפני כתיבה לדיסק של תנועה שטרם התחייבה, שומרים את ערך המשתנה לפני הכתיבה (before information).

במקום אמין שומרים יומן (log) שמתעד כל פעולה, לפי אלגוריתם ההתאוששות, למשל:

- עבור כתיבה ל- A: before information, after information
- עבור תנועה T: סימון של התחלה, התחייבות (commit), הפסקה (abort)

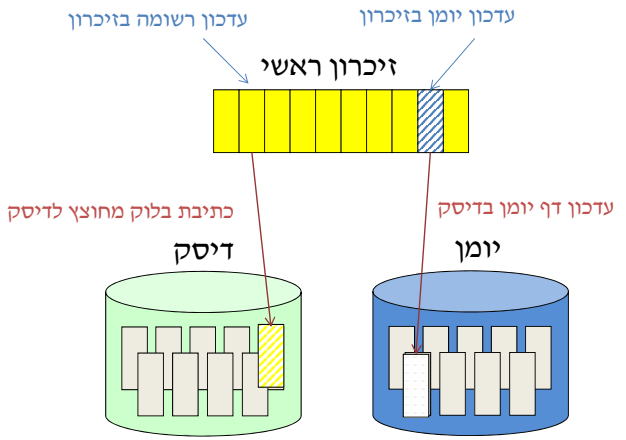
ניהול היומן והדיסק

מערכת ניהול תנועות



ניהול היומן והדיסק

מערכת ניהול תנועות



אלגוריתם התאוששות 1: Redo

- בהתחלה, הדיסק ריק: כל נתון בדיסק נכתב ע"י תנועה כלשהי
- ביומן יש את התנועות מזמן 0

היומן המקורי

```
T1 BEGIN
T1 WRITE 51 TO 1000
T2 BEGIN
T2 WRITE 61 TO 1001
T3 BEGIN
T3 WRITE 53 TO 1002
T1 WRITE 71 TO 1003
T4 BEGIN
T1 COMMIT
T2 ABORT
T4 WRITE 84 TO 1000
T4 COMMIT
T3 WRITE 93 TO 1001
```

failure

<u>אלגוריתם ההתאוששות</u>

1. מחק מהיומן את כל הפעולות של התנועות שלא ביצעו התחייבות (שהופסקו או שטרם הספיקו לכתוב את ההתחייבות ביומן)
2. עבור על היומן מההתחלה לסוף, ובצע מחדש כל פעולת כתיבה

אלגוריתם התאוששות 1: Redo

- בהתחלה, הדיסק ריק: כל נתון בדיסק נכתב ע"י תנועה כלשהי
- ביומן יש את התנועות מזמן 0

היומן לאחר המחיקות

```
T1 BEGIN
T1 WRITE 51 TO 1000   ←
T2 BEGIN
T2 WRITE 61 TO 1001
T3 BEGIN
T3 WRITE 53 TO 1002
T1 WRITE 71 TO 1003   ←
T4 BEGIN
T1 COMMIT
T2 ABORT
T4 WRITE 84 TO 1000   ←
T4 COMMIT
T3 WRITE 93 TO 1001
```

failure

אלגוריתם ההתאוששות

1. מחק מהיומן את כל הפעולות של התנועות שלא ביצעו התחייבות (שהופסקו או שטרם הספיקו לכתוב את ההתחייבות ביומן).
2. עבור על היומן מההתחלה לסוף, בצע מחדש כל פעולת כתיבה

תכונות אלגוריתם Redo

- ✓ מאפשר להתאושש מנפילת מדיה (בעזרת היומן).
- ✓ לא משתמש ב-before information
- ✓ אידמפוטנטי: להפעלה כפולה של האלגוריתם יש אפקט כמו הפעלה בודדת.

1. אי יעילות בהתאוששות אם יש כתובת שכתבו אליה כמה תנועות שהתחייבו. ניתן היה לשמור רק את הכתיבה האחרונה של התנועה שהתחייבה.
2. אפשר לנצל נקודות ביקורת (checkpoints), ולהתחיל לבצע מחדש רק כתיבות שנעשו אחריהן:

   השהה תנועות חדשות;
   חכה עד שכל התנועות הפעילות סיימו;
   וודא שכל הכתיבות בוצעו פיזית;
   קח גיבוי מלא של הדיסק.

Redo עם הכנות בזמן העבודה

1. כאשר תנועה T מתחילה: נרשום (T begin) ביומן.

למשל, עדכון בלוק בזיכרון ולא בדיסק

2. כאשר T כותבת ערך v ל A: נרשום (T,A,after=v) ביומן, נכתוב את v להעתק של A, אך לא ל-A
3. כאשר T מופסקת : נרשום (T abort) ביומן (אפשרי רק לפני ההתחייבות).
4. כאשר T מבצעת ההתחייבות: נרשום (T commit) ביומן, ונוודא שהיומן נכתב על הדיסק – זהו מימוש ההתחייבות.
5. אחרי ההתחייבות: נבצע את כל הכתיבות של התנועה על הדיסק, ולאחר מכן, נשחרר את המנעולים.

Redo: דוגמה בזמן העבודה

	התנועה	ביצוע	היומן
1.	BEGIN		(T begin)
2.	LOCK A	lock A	
3.	LOCK B	lock B	
4.	WRITE v to A	write A to copy	(T, A, after = v)
5.	WRITE w to B	write B to copy	(T, B, after = w)
6.	COMMIT	write log to disk	(T commit)
7.		write A to disk	
8.		write B to disk	
9.	UNLOCK A	unlock A	
10.	UNLOCK B	unlock B	

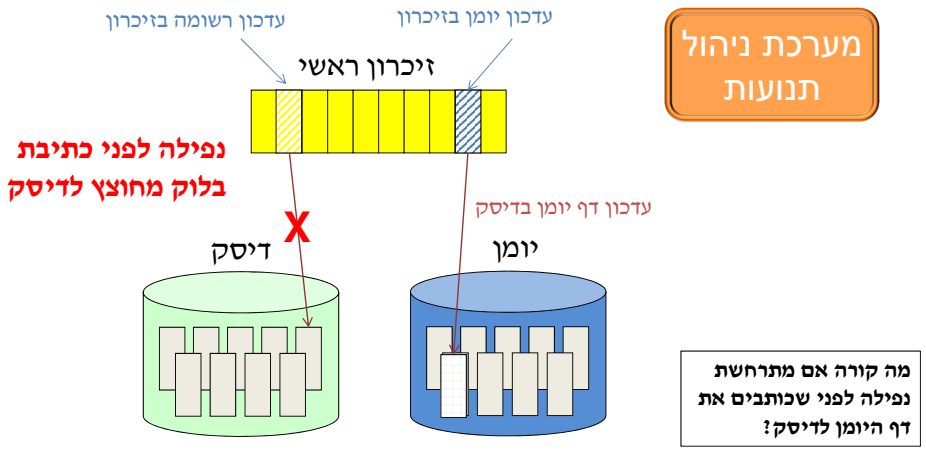
Redo בזמן ההתאוששות

1. נעתיק את הדיסק מהגיבוי.
2. נקרא את היומן.
3. נמחק מהיומן את התנועות שלא ביצעו התחייבות.
4. נבצע מחדש את העדכונים:
   נעבור על היומן מההתחלה לסוף
   ונכתוב לדיסק את כל פעולות
   הכתיבה שעדיין נמצאות ביומן

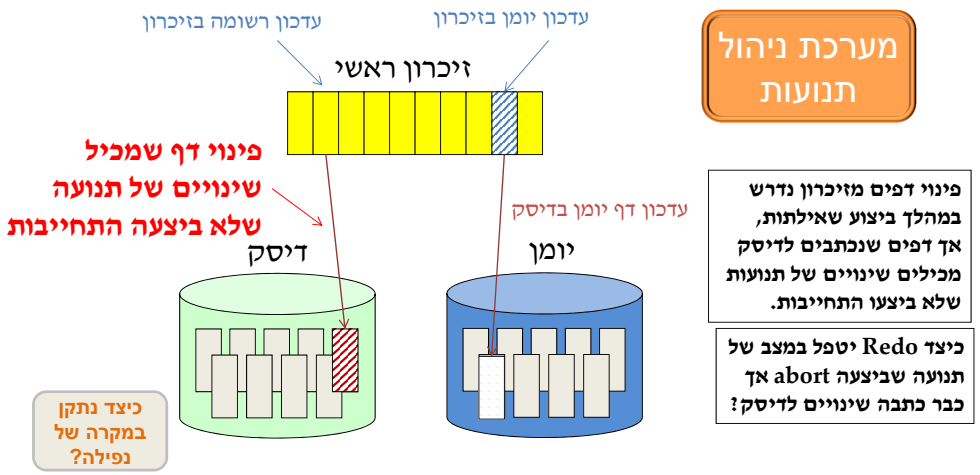
T1: lock A
T1: commit
T1: write A
T3: unlock A
T2: lock A
T2: abort A
T2: unlock A
T3: lock A
T3: commit
T3: write A
T3: unlock A
T4: lock A
T4: commit
T4: write A

failure

Redo מטפל בנפילות לפני כתיבה לדיסק



פינוי דפים מזיכרון



אלגוריתם התאוששות 2: Undo

הנחות:

- לפני הנפילה, הדיסק מכיל את כל הכתיבות של תנועות שהתחייבו + כתיבות של תנועות שלא התחייבו
- היומן מכיל את כל הפעולות מ"בריאת העולם".

בכתובת A יש אינפורמציה לא תקינה אם הכתיבה האחרונה אליה היא של תנועה שלא ביצעה התחייבות (הכתובת A מקולקלת).

אלגוריתם ההתאוששות:

– לכל כתובת A נמצא ביומן את הכתיבה האחרונה ל-A ע"י תנועה שביצעה התחייבות.
– אם זו הכתיבה האחרונה ל-A, אזי הערך בכתובת A תקין.
– אחרת (A מקולקלת), מצא ביומן את הכתיבה הבאה ל-A ושחזר מה-before information של תנועה זו את הערך של A.

אלגוריתם Undo: דוגמה

	התנועה	ביצוע	היומן
1.	BEGIN		(T begin)
2.	LOCK A	lock A	
3.	LOCK B	lock B	
4.	WRITE A	u = READ A	(T, A, before = u)
		write log to disk	
		write new A	
5.	WRITE B	w = READ B	(T, B, before = w)
		write log to disk	
		write new B	
6.	COMMIT	write log to disk	(T commit)
7.	UNLOCK A	unlock A	
8.	UNLOCK B	unlock B	

תכונות האלגוריתם

1. תנועה מתחייבת רק אחרי שכתיבותיה בוצעו פיזית בדיסק.
2. ההתחייבות מתבצעת ע"י כתיבה פיזית של היומן.
3. כל תנועה כותבת את ה-before information ליומן לפני הכתיבה הפיזית לדיסק.
4. ניתן להתגבר על נפילת מערכת אבל לא על נפילת דיסק. למה?
5. לא משתמשים ב-after information.
6. האלגוריתם כפי שתואר מאד לא יעיל.
   נבצע אותו ע"י סריקה אחת של היומן מהסוף להתחלה.
7. אידמפוטנטי.

Undo: הכנות תוך כדי הריצה

1. כשתנועה T מתחילה — נרשום (T begin) ביומן.
2. כש-T רוצה לכתוב ערך v לכתובת A :
   נקרא את הערך הישן u;
   נרשום (T, A, before = u) ביומן, ונכתוב את היומן בדיסק;
   נכתוב את הערך החדש v לכתובת A.
3. כשתנועה T מבצעת abort —
   נרשום (T abort) ביומן, נבטל את הכתיבות של T, ונשחרר את המנעולים.
4. כשתנועה T מבצעת התחייבות —
   נוודא שכל הכתיבות של T בוצעו;
   נרשום (T commit) ביומן,
   ונוודא שהיומן נכתב על הדיסק — זהו מימוש ההתחייבות.
   בסוף נשחרר את המנעולים של T.

Undo: דוגמה בזמן הריצה

ערכים התחלתיים:

1000:20 1001:21 1002:22 1003:23

	היומן
T1 BEGIN	
T1 WRITE 51 TO 1000	Before = 20
T2 BEGIN	
T2 WRITE 61 TO 1001	Before = 21
T3 BEGIN	
T3 WRITE 53 TO 1002	Before = 22
T1 WRITE 71 TO 1003	Before = 23
T4 BEGIN	
T1 COMMIT	
T2 ABORT	
T4 WRITE 84 TO 1000	Before = 51
T4 COMMIT	
T3 WRITE 93 TO 1001	Before = 21
FAILURE	

מדוע 21 ולא 61?

Undo: בזמן התאוששות

קבוצת התנועות שהתחייבו committed = ∅

קבוצת הכתובות שכבר שוחזרו restored = ∅

נעבור על היומן מהסוף להתחלה:

עבור (T, commit): נוסיף את T ל-committed.

עבור (T, A, before = u):

אם T ∈ committed : נוסיף את A ל-restored.

אחרת:

אם A ∈ restored: אל תעשה כלום.

(אחרי T יש תנועה 'T שכתבה ל-A והתחייבה)

אחרת: נכתוב את u לכתובת A בדיסק.

האם ניתן להוסיף את A ל-restored?

מה קורה אם אותה תנועה T כתבה ל-A מספר פעמים

הערך הנכון של A

checkpoint

T1: lock A
T1: write A
T1: commit
T1: unlock A

T2: lock A
T2: abort
T2: unlock A

T3: lock A
T3: write A
T3: write A ← ערך נכון
T3: commit
T3: unlock A

T4: lock A
T4: write A
T4: abort
T4: unlock A

T5: lock A
T5: write A
T5: write A ← ערך בזמן הנפילה

failure

Undo: בהתאוששות

ערכים התחלתיים:

1000:20 1001:21 1002:22 1003:23

פעולות השחזור המתאימות	היומן	
T1 BEGIN		
T1 WRITE 51 TO 1000	Before = 20	
T2 BEGIN		
T2 WRITE 61 TO 1001	Before = 21	Write 21 to 1001
T3 BEGIN		
T3 WRITE 53 TO 1002	Before = 22	Write 22 to 1002
T1 WRITE 71 TO 1003	Before = 23	restored = {1000, 1003}
T4 BEGIN		
T1 COMMIT		Committed = {T1, T4}
T2 ABORT		
T4 WRITE 84 TO 1000	Before = 51	restored = {1000}
T4 COMMIT		Committed = {T4}
T3 WRITE 93 TO 1001	Before = 21	Write 21 to 1001
FAILURE		Committed = ∅ restored = ∅

סיכום הדוגמה

לאחר השחזור, הזיכרון צריך להכיל את הכתיבות של T1 ו-T4.

כתובת	לפני	בנפילה	שחזור
1000	20	84	84
1001	21	93	21
1002	22	53	22
1003	23	71	71

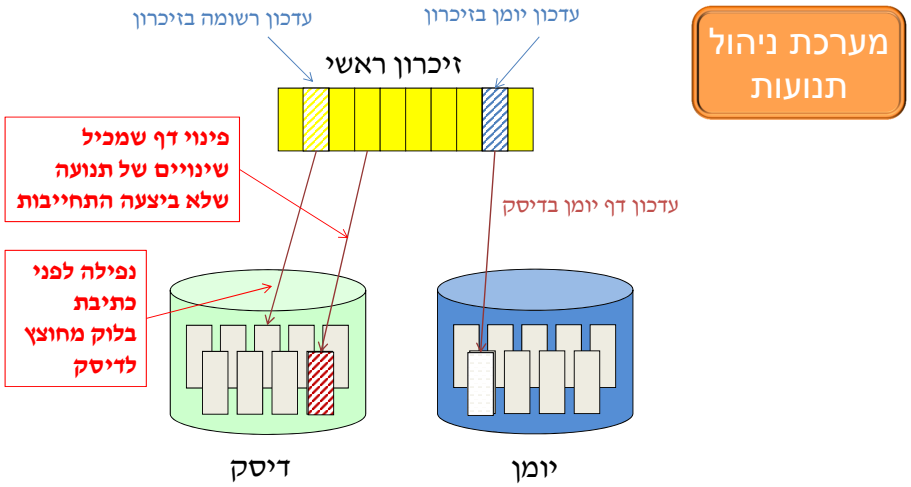
שאלות:

1. האם צריך לשמור את restored אם מובטח לנו שפעולות התנועות שהופסקו טרם הנפילה נמחקו גם מהיומן?
2. מה קורה לתנועות שטרם סיימו בזמן הנפילה?
3. מה קורה אם יש נפילה תוך כדי התאוששות?

Undo לעומת Redo

- באלגוריתם ה-Redo כל הכתיבות לדיסק של תנועה מבוצעות בבת אחת.
- Redo מחייב לשמור את כל הכתיבות (מאז ה checkpoint).
- אלגוריתם ה-Undo מחייב קריאה לפני כתיבה.
  מכיוון שבד"כ המערכת ממילא קוראת, המחיר אינו גבוה במיוחד.
- Undo אינו מאפשר שחזור דיסק מגיבוי.
- Redo אינו מתאים לטיפול בנפילות של תנועות שכתבו לדיסק.
- אפשר לשלב בין Undo ו-Redo

רוצים גם לטפל בנפילות לפני כתיבה לדיסק וגם בנפילות אחרי כתיבה ללא התחייבות



אלגוריתם התאוששות 3: Undo / Redo

באלגוריתם המשולב רושמים לכל פעולה ביומן אינפורמציית before וגם after

Update A $\Rightarrow$ <T_i, A, New A val, Old A val>

ניתן לכתוב את A לפני או אחרי ש T_i מתחייבת (commit).
אבל קודם נוודא שהיומן נכתב.
בזמן התחייבות, דואגים לכתוב את היומן

אלגוריתם Undo / Redo

בזמן התאוששות:

- מעבר undo אחורה עד ל checkpoint האחרון
  - אוספים קבוצה S של תנועות שהתחייבו
  - מבטלים כתיבות של פעולות שאינן ב S
- מעבר redo קדימה (מה checkpoint האחרון עד לסוף היומן)
  - ביצוע מחדש של כתיבות של התנועות ב S

היומן
checkpoint
…
T1:A,10,15
…
T1:B,20,23
T1:commit
…
T2:C,30,38
T2:,D,40,41
…
crash

undo
redo

סוגים שונים של נקודות ביקורת

1. גיבוי מלא:
   - משהים את כל התנועות החדשות.
   - מחכים עד שכל התנועות הפעילות ייגמרו.
   - כותבים את כל החוצצים לדיסק.
   - מבצעים גיבוי מלא של הדיסק (על דיסק אחר או על מדיום אחר).
   - גיבוי זה מאפשר להתגבר על נפילת הדיסק.
2. כתיבת כל החוצצים: מאפשר לשחזר את מצב הדיסק למצב עקבי:
   - בצע אלגוריתם undo מנקודת הנפילה אחורה, כדי לקבל תמונה נכונה של הדיסק בנקודת הביקורת.
   - בצע אלגוריתם redo מנקודת הביקורת עד הנפילה.

   מאפשר להתגבר על נפילת מערכת—לא על נפילת מדיה.

   כתיבת כל החוצצים בנקודת הביקורת פוגעת ביעילות.
3. כתיבה רק של חוצצים חדשים.

סיכום אלגוריתמי התאוששות

	התאוששות מנפילות	התאוששות מנפילת דיסק	טיפול יעיל ב- abort כשיש דפדוף	bursty
Redo	איטי	כן	לא	כן
Undo	מהיר	לא	כן	לא
Undo/ Redo	מהיר	כן	כן	לא