מיון – מיזוג.

זמן מיון = זמן מעבר X מספר מעברים
הגדלת דרגת המיזוג מקטינה את מספר הסיבובים בגורם כפלי.

Replacement/Selection:יצירת מעלות התחלתיים בגודל $\leq M$.
זמן: זמן אתחול+ זמן כתיבה סדרתית של הקובץ לדיסק הפלט.
זמן אתחול = הגעה לתחילת הקובץ + מילוי הזכרון + מילוי חוצץ 1 רשומה קפואה: המפתח שלה קטן מהאחרון שהוצא. השאר– פעילות

המיון:

סדר המיזוג המקסימאלי: $k = \left\lfloor \frac{memory\_size}{2 \bullet buffer\_size} \right\rfloor - 1$

מספר מעברי המיזוג: $\lceil \log_k (\#run) \rceil$

גודל חוצץ מינימאלי : גודל של סקטור .

גודל חוצץ מקסימאלי: $\frac{M}{6}$

מעלות
N: מספר הרשומות בקובץ.
M: גודל הזכרון.

מספר מעלות צפוי: $\frac{N}{2M}$

תוחלת גודל של מעלה - $2M$.
מספר המעלות: מספר רשומות\גודל של מעלה.
אם עבור זיכרון בגודל M נקבל k מעלות, אז עבור זכרון גדול מ – M נקבל לכל היותר k מעלות.

עקרון החיזוי – כאשר חוצץ מתרוקן הקצה אותו למעלה שעתיד להגמר קודם- עבור כל מעלה, i נסמן ב: $last_i$ את המפתח הגדול ביותר שלו שנמצא בזיכרון. נקרא חוצץ עבור המעלה i המקיים $\min_i\{last_i\}$ – מבטיח פעולה רציפה.
כאשר זמן העיבוד > מזמן הקריאה – 3 חוצצים מבטיחים פעולה רציפה.

עצי +B

מבנה העץ:
שורש – דרגתו d - $2 \leq d \leq m$
צומת פנימי (שאינו שורש) – דרגתו d $\lceil m/2 \rceil \leq d \leq m$
עלים- כולם באותה רמה.
הרשומות - רק בעלים. מס' הרשומות בעלה: $\lceil b/2 \rceil \leq x \leq b$
גובה העץ – מספר הרמות לא כולל רמת העלים.
מספר העלים בעץ בגובה h - בין $2\lceil m/2 \rceil^{h-1} \leq L \leq m^h$.
אם בעץ יש L עלים – אז הגובה המינימאלי מקיים $h_{\min}$
$m^{h_{\min}-1} < L \leq m^{h_{\min}} \Leftarrow h_{\min} = \lceil \log_m L \rceil$.
מתקיים: $h_{\max} - h_{\min} \leq 2$
בביצוע הכנסות לעץ ריק **תפוסה ממוצעת של מס' מפתחות – 0.7 =** ln2 (בעלים ובצמתים הפנימיים)
תמיד קוראים וכותבים עלים שלמים, ולא חלקי עלים.

הכנסה לעץ:
מצא את העלה אליו צריך להוסיף,
אם יש בו פחות מ – b רשומות- הוסף. וסיים.
אחרת פצל ל- 2 עלים כאשר: $\lceil b/2 \rceil$ רשומות בישן
+1 $\lfloor b/2 \rfloor$ רשומות בחדש, הוסף מפתח לאבא.
אם יש גלישה בצומת פנימי (יותר מ – m בנים), פצל כמו קודם.
אם יש גלישה בשורש, פצל לשניים וצור שורש חדש.

תהלים ההוספה מלמטה למעלה.
זמן הכנסה (ללא פיצול) - $T_{insert} = T_{find} + rotation$
(ההנחה גודל עלה קטן ממסילה)

הוצאה מהעץ:
חפש את הרשומה:
אם בעלה יש יותר מ $\lceil b/2 \rceil$ רשומות, מחק את הרשומה וסיים.
אחרת,
אם לעלה של הרשומה יש אח שכן עם $\leq \lceil b/2 \rceil + 1$ העבר רשומות, עדכן את הצומת, האח והמפתח באבא.
אחרת אחד את שני האחים וסלק את המפתח שמפריד מהאבא, תוך שימוש באותו אלגוריתם כמו הוצאת רשומה מעלה.
אם לשורש נותר בן יחיד: סלק אותו והפוך את הבן לשורש החדש.
זמן הוצאת (ללא פיצול) – בדומה לזמן ההכנסה.

קריאת תת-תחום:
קוראים אקראית את הרשומה הראשונה בתת בתחום, אח"כ קוראים לפי הסדר, עלה אחר עלה את כל הרשומות ששייכות לתת תחום, בדר"כ נניח שבכל עלה קיים מצביע לעלה הבא.
זמן הקריאה של כל עלה מחושב כזמן גישה אקראית כיוון שעלים עוקבים עפ"י סדר, לא יושבים זה אחרי זה פיסית.

כאשר יש הרבה קריאות אקראיות, כדאי להגדיר עלים קטנים
כאשר יש הרבה קריאות סדרתיות, כדאי להגדיר עלים גדולים

רשומות עם כמה מפתחות:
אינדקס ראשי: הרשומות יסודרו לפי מפתח ראשי למשל ב- B+
אינדקס משני: מבנה הוסף לפי המפתח המשני. (פוינטר לראשי). גם הם יכולים להיות מאורגנים ב B+.
אינדקס משני נעוץ:
במיני רשומה שבאינדקס המשני נשמור פוינטר לעלה המכיל את הרשומה המקורית
בהכנסה: אם העלה של האינדקס הראשי פוצל, יש לעדכן את כל המפתחות באינדקס המשני שהצביעו לרשומות שנדדו לעלה אחר.
השיטה עדיפה אם יש הרבה חיפושים.
לא טובה כאשר יש הרבה העברות של רשומות
אינדקס משני בלתי נעוץ:
במיני רשומה שבאינדקס המשני נשמור את המפתח הראשי של הרשומה המקורית.
בהכנסה: לא צריך לעדכן את העץ הראשי
השיטה עדיפה עבור מעט חיפושים, כי זמן החיפוש כפול.
אינדקס משולב נעוץ/בלתי נעוץ:
במיני רשומה שבאינדקס המשני נשמור הן את המפתח הראשי של הרשומה המקורית והן פוינטר למיקום (משוער) של הרשומה המקורית
את הפוינטר נעדכן כאשר נחפש רשומה: אם לא נמצא דרך הפוינטר נחפש דרך המפתח ואז נעדכן.

ערבול – Hashing

ערבול מאפשר למצוא רשומה בגישת דיסק בודדת
n מספר המפתחות
K תחום המפתחות (0…K-1).
M גודל הזיכרון – ברשומות.
פונק' צמצום: $h: 0..K-1 \to 0..M-1$ (מצמצמת לגודל הזיכרון)
פקטור העומס: $\alpha = n/M$.
שיטות לטיפול בהתנגשויות:התנגשות $x \neq y$ אבל $h(x) = h(y)$
סריקה לינארית: שים במקום: $h(x), h(x)+1, \ldots$
אם פקטור העומס <80% סריקה לינארית נותנת זמן סביר.
ערבול כפול: $h(x), h(x)+d(x), h(x)+2d(x) \ldots$
ערבול מחדש: $h_1(x), h_2(x), h_3(x) \ldots$
שרשראות: יצירת רשימה מקושרת מכל תא במערך.

ארגון המבנה על הדיסק:
כיוון שתמיד קוראים סקטור (לוגי) שלם שבו מקום ל – b רשומות, נשתמש בטבלה עם M דליים כל אחד עם b רשומות. נמפה רשומה לדלי ע"י ערבול, אם אין מקום בדלי נשים את הרשומה בגרורה.
גרורות: לרוב יהיו כמעט ריקות ולכן כדי לחסוך במקום לעיתים הגרורה האחרונה של דלי תהיה משותפת. (השיתוף לא מגדיל מס' גישות בחיפוש).

מיפוי דליים לדיסק:
מיפוי ישיר: נשלח את הדליים לסקטורים באופן רציף ונשאיר עודף אשר חשמש לגלישה.
אזורים רצופים: בכל איזור רצוף בזכרון, נמפה דליים, ובזכרון הראשי נחזיק מדריך.
אינדקס: טבלת הערבול תשמש רק כאינדקס. כדי לקרוא רשומה שהמפתח שלה הוא x נחשב את h(x) כתובת הדלי של הרשומה תמצא במקום h(x) באינדקס.

M דליים, b רשומות בדלי, n רשומות בקובץ.
ההסתברות שרשומה אקראית תמופה לדלי i: $1/M = \alpha b/n$

t- זמן קריאת דלי.
ההסתברות שאין גלישה – P
ההסתברות שיש גלישה $1-P$
תוספת הזמן לקרוא גרורה $\Delta = rot\_delay + transfer$
זמן חיפוש ממוצע למפתח:
שאינו במבנה: $t + P \cdot \Delta$
שנמצא במבנה: $t + (1-P) \cdot 0.5 \cdot \Delta$

זמן הכנסה של רשומה x: $T_{unsuccessful-find} + rotation$
חפש "כאילו" x לא נמצא.
עדכן חוצץ.
כתוב.
זמן הוצאה של רשומה x: $T_{unsuccessful-find} + rotation$

אם מוכרחים לעבור סדרתית על מבנה הערבול עדיף למיינו קודם.

קבצי אי סדר חסום: צירוף של עץ +B וטבלת ערבול, האינדקס הוא עץ +B כך שכל האינדקס נשאר בזכרון הראשי, הרשומות בעלים כאשר על כלה יכיל טבלה ערבול.

Relations

שורה בטבלה – רשומה.
עמודה בטבלה – תכונה.
טבלה - יחס

פעולות שמחזירות יחס חדש:
איחוד: $R \cup S = \{r : r \in R \text{ or } r \in s\}$
חיתוך: $R \cap S = \{r : r \in R \text{ and } r \in s\}$
הפרש: $R \setminus S = \{r : r \in R \text{ and } r \notin s\}$
מכפלה קרטזית:
$R \times S = \{(r_1, r_2, ..., r_n, s_1, s_2, ..., s_n) : r \in R \text{ and } r \in s\}$
פעולה המנפחת את מס' הרשומות נמנע מלשמור על הדיסק.
Select: σ בוחר את הרשומות המקיימות תנאי נתון C.
ניתן להגדיר בפרט שאילתות טווח.
Projection: π מצמצמת את היחס לחלק מהעמודות המקוריות.
החלפת שם של שדה: $\pi_{A \to B}$
Join: $\rhd\lhd$ יחס חדש שמתקבל משני יחסים שיש להם תכונות בעלות אותו שם. היחס מתקבל משלוב הרשומות שמסכימות על התכונות בעלת השמות המשותפים.
הפעולה אסוציאטיבית וקומוטטבית
Distinct: δ - היחס הופך לקבוצה (השאר עותק אחד מכל רשומה).

פעולות שאינן מחזירות יחס חדש:
Average, min max: מחזירות מספר

Sort: מארגן מחדש את אותו היחס.

שיטות למימוש שאילתות רלציוניות:
B(R): מספר הבלוקים (סקטורים) של R.
T(R): מספר הרשומות של R.
V(R): מספר הרשומות השונות של R.
M: מספר הבלוקים שאפשר להזיק בזיכרון הראשי.
S,R – יחסים. מתקיים S<R.
הסיבוכיות ללא התחשבות בכתיבת הפלט.

מעבר יחיד:

הפעולה	דרישות הזיכרון	גישות דיסק
σ, π, ממוצע	O(1)	B(R)
$\cup$	O(1)	B(R) + B(S)
δ במעבר אחד	B(δ(R))	B(R)
$\setminus$ $\cap$ $\times$ $\bowtie$	B(δ(S))	B(R) + B(S)

כדי לאפשר קריאה רציפה של הרלציות, נזדקק לשני חוצצי קלט. וכדי לאפשר רציפה של הפלט גם כן נשתמש בשני חוצצים $(double\ buffering)$ אז כדי שהיחס ייכנס בזיכרון נדרוש $B(R) \le M - 4$.

דוגמא לחיתוך במעבר יחיד: קרא את כל הרשומות ב – S ושמור בזיכרון הראשי עותק אחד מכל שרומה, ומונה המציין כמה פעמים הרשומה s ב – S (נסמן s.count)
קרא את R ועבור כל רשומה r ב – R, אם קיימת בזיכרון רשומה s=r ו – s.count>0 אז הוצא את r לפלט והורד את s.count באחד.

לולאות מקוננות:

Join בלולאות מקוננות:
כל עוד נשארו בלוקים של S
קרא את $M-4$ הבלוקים הבאים של S
בצע join של הבלוקים של S שבזיכרון עם כל R.
מספר כולל של גישות לדיסק $\left\lceil \frac{B(S)}{M-4} \right\rceil B(R) + B(S)$

שיטות מבוססות מיון :

מיון היחסים ואז ביצוע במעבר יחיד($join$ $(\delta, \cap$.
Join מבוסס מיון: R,S ממויינים לפי A. מתקיים $|\sigma_{A=a}(S)| \le M$ מס' הרשומות בעלות אותו ערך A נכנס לזכרון.
עבור על S ועל R:
הכנס לזכרון את הרשומות הראשונות $s \in S, r \in R$.
אם $\pi_A(r) > \pi_A(s)$ החלף לרשומה הבאה של S.
אחרת אם $\pi_A(r) < \pi_A(s)$ החלף לרשומה הבאה של R.
אחרת, קרא לזכרון את כל הרשומות מS : $\pi_A(s') > \pi_A(s)$
סיבוכיות $B(S) + B(R)$ ללא זמן כתיבת הפלט.

אלגוריתמים מבוססי מיון: סיכום - הנחה: הרלציה ממויינת.

הפעולה	מספר גישות דיסק	זכרון ראשי
δ	B(R)	1
Min, Max	1	1
$\cap$, $\cup$ (שומר מיון)	B(S)+B(R)	2

שיטות מבוססות ערבול:

הרעיון: אם הדלי נכנס לזיכרון נוכל לבצע את הפעולה במעבר יחיד.

מספר הדליים המקסימאלי k מקיים: $2k+2=M$.

גודל דלי ממוצע $\frac{B(R)}{k}=\frac{2\cdot B(R)}{M-2}$

כדי שדלי ייכנס לזיכרון צריך להתקיים: $\frac{2\cdot B(R)}{M-2}\le M-4$

מספר הגישות לדיסק:

בשלב החלוקה לדליים של שני היחסים $2\cdot B(R)+2\cdot B(S)$.

לאחר מכן עוד מעבר יחיד $B(R)+B(S)$

סה"כ $3\cdot B(R)+3\cdot B(S)$.

אם מתקיים $B>\frac{M^2}{2}$ אז יצא סה"כ $5\cdot B(R)+5\cdot B(S)$

דוגמא: מבוסס ערבול δ: מחלקים את היחד לדליים, כל דלי מביאים לזיכרון ועושים עליו δ. עובד כאשר הדלי נכנס לזיכרון.

עבור $B(R)$ כללי: (לא ידוע אם דלי יכנס לזיכרון)

במעבר הראשון מחלקים ל $k=M/2-1$ דליים, ובכל מעבר מחלקים כל דלי ל – k דליים עד אשר נחלק לדליים שגודלם אינו עולה על M.

דוגמא: חיתוך בעזרת ערבול:

קריאת S וחלוקתו לדליים, קריאת R וחלוקתו לדליים.

מעבר על דליי S וכל פעם חיתוך עם דלי R מתאים

אותו הדבר לגבי join בעזרת ערבול – בעזרת אינדקס אפשר להקטין עוד יותר (ראה חוברת).

שימוש בעץ חישוב שמאלי : המאפשר ביצוע pipeline בכל צומת רצליה אחת מתקבלת מחישוב והשנייה מהדיס, נשים בעלה השמאלי ביותר את הרלציה הקטנה ביותר וכן הלאה.

בחירת עץ החישוב באמצעות גודל תוצאות הביניים

פעולה	נוסחה	הערות והסברים
$\sigma_{A=a}(R)$	$E(T(\sigma_{A=a}(R))=T(R)/V(R,A)$	לא תלוי בהתפלגות של A, מניחים רק ש a נבחר מבין הערכים האפשריים של A בהסתברות שווה
$\sigma_{A<a}(R)$	אם ידועה ההתפלגות: $E(T(\sigma_{A<a}(R))=P(A<a)*T(R)$ אחרת, נניח שההתפלגות היא יוניפורמית. במקרה הרציף אם ידוע התחום $[a_0,b_0]$, אז $E(T(\sigma_{A<a}(R))=(a-a_0)/(b_0-a_0)$ ובאופן דומה במקרה הדיסקרטי. אם התחום לא ידוע, נניח באופן שרירותי $E(T(\sigma_{A<a}(R))= T(R)/3$	
$\pi(R)$	$T(R)\cdot f$, כאשר גודל השדות שנבחרו f בתים.	
$R(X,Y)\triangleright\triangleleft S(Y,Z)$	$E(T(R\triangleright\triangleleft S))=\frac{T(R)\cdot T(S)}{\max\{V(S,Y),V(R,Y)\}}$	תחת ההנחות הבאות: • Y מכיל תכונה בודדת • אם Y_R ו Y_S הן קבוצות הערכים של התכונה Y ב R ו ב S אז $Y_R\supseteq Y_S$ או $Y_S\supseteq Y_R$. • שמירת ערכים: $V(R\triangleright\triangleleft S,A)=V(R,A)$
$\delta(R)$	$T(\delta(R))=V(R)=\min\left\{\prod_{i\le s\le n}V(R,a_i),\frac{1}{2}T(R)\right\}$	"כלל אצבע"

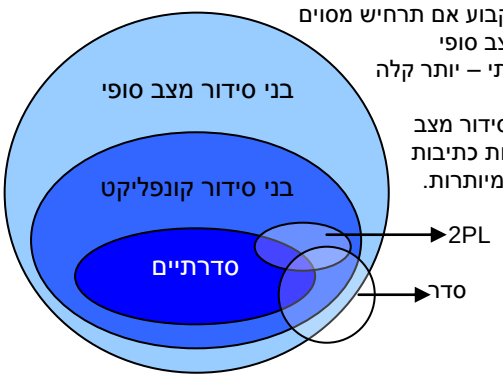
בקרת מקביליות

תנועה תקרא **עקבית** אם ביצועה שומר על נכונות מבנה הנתונים.

בתרחיש סדרתי כל תנועה מתחילה רק לאחר סיום התנועה הקודמת

תרחיש הוא **בר–סידור** (במצב סופי) אם הוא שקול (מצב סופי) לתרחיש סדרתי כלשהו

תרחישים H_1,H_2 הם **שקולי מצב סופי** אם מכל מצב תחילי של המערכת, ועבור כל סמנטיקה של התנועות הרצת H_1 מביאה את המערכת לאותו מצב סופי כמו הרצת H_2.



יהיה קשה לנו לקבוע אם תרחיש מסוים הוא בר סידור מצב סופי הכרעה אם סדרתי – יותר קלה

בתרחישים ברי סידור מצב סופי, יכולות להיות כתיבות עיוורות ופעולות מיותרות.

קריאה מיותרת: אם התנועה מצבעת קריאה שאין אחריה אף כתיבה.

תרחישים גרועים:

עדכון אבוד : עדכון שלא ניתן לקרוא אותו כי אחריו יש עדכון אותו משתנה על ידי תנועה אחרת.

קריאה מלוכלכת: תנועה אחת קוראת ערך ביניים של משתנה בזמן שינויו על ידי תנועה שנייה.

קריאה שלא ניתן לחזור עליה: כיוון שאחריה ישנו עדכון של המשתנה על ידי תנועה אחרת.

שתי תנועות תיקראנה **מתנגשות** אם הן מתייחסות לאותו פריט נתונים ולפחות אחת מן הפעולות היא פעולת כתיבה.

תרחיש ללא **פעולות מתנגשות** אינו מכיל **תרחישים גרועים.**

שקילות קונפליקט: שני תרחישים הינם שקולי קונפליקט, אם הם מוגדרים על אותה קבוצת תנועות. והם מסכימים על הסדר של פעולות מתנגשות.

תרחיש הוא בר סידור קונפליקט אם הוא שקול קונפליקט לתרחיש סדרתי.

תרחיש אחד שקול קונפליקט לתרחיש שני אמ"ם גרף התלויות שלהם זהה.

משפט: תרחיש H הוא בר סידור קונפליקט אמ"ם גרף התלויות שלו חסר מעגלים מכוונים.

משפט: אם אין פעולות מיותרות או כתיבות עיוורות, אז: H תרחיש בר סידור מצב סופי $\Leftrightarrow$ H תרחיש בר סידור קונפליקט.

יהיו H ו – H' תרחישים ו- T תנועה המשתתפת בהם. אם לכל פריט נתונים A ש- T קוראת, T קוראת אותו הערך מ – A גם ב – H וגם ב – H', אזי T מבצעת אותן הפעולות ב- H וב – H'. במקרה הזה נאמר ש – H ו – H' הן **שקולות מבט ביחס ל – T.**

1. אם i<j אז הכתיבה האחרונה של T_i לפריט נתונים A בתרחיש H מתבצעת לפני הגישה הראשונה של תנועה T_j ל- A בתרחיש H.
2. אם i<j אז הגישה האחרונה של T_i לפריט נתונים A בתרחיש H נעשתה לפני הכתיבה הראשונה של תנועה T_j ל- A בתרחיש H.

מנעולים

SLOCK: מנעול קריאה (shared lock) – יכול להיות על אובייקט יותר ממנעול קריאה אחד. לקריאה.
XLOCK: מנעול כתיבה (exclusive lock) – מנעול לקריאה ולכתיבה. אם לאובייקט יש מנעול כתיבה אין לשים עליו מנעול נוסף.

תנועה חייבת לשחרר את כל המנעולים שלה לפני סיומה.

בקר מקביליות הוא נכון אם מאפשר רק תהליכי ביצוע בר סידור.

2PL: 2 phase locking תנועה תעמוד בדירה אמ"ם מתקיימים:
1. תנאי הנעילה והגישה.
2. כל הנעילות שהתנועה מבצעת מופיעות לפני השחרור הראשון.

משפט: אם כל התנועות בתרחיש מקיימות את דרישת 2PL אז התרחיש הוא בר סידור קונפליקט.
<u>הערות:</u> 1.תהליך ביצוע מקיים 2PL רק אם כל תנועותיו מקיימות אותו
2. תהליך ביצוע המקיים 2PL עלול להיקלע לחבק.

פרוטוקול הסדר :

נמספר את המשתנים (אליהם נעשות הכתיבות/הקריאות) : $X_1, X_2, ..., X_n$. כל תנועה יכולה לנעול משתנה X_i בתנאי שלא נעלה משתנה X_j עבור $i \le j$. ניתן לשחרר מנעולים בכל עת.
(הפרוטוקול משתמש במנעולי Xlock בלבד.)
פרוטוקול הסדר **מונע חבק**
תרחיש המורכב כולו מתנועות המקיימות את הפרוטוקול **אינו בהכרח בר סידור קונפליקט.**

פרוטוקול העץ:

1. חוץ מהצומת הראשון שננעל, אסור לנעול צומת, אלא אם מחזיקים מנעול על אביו (לא חייבים להתחיל מהשורש).
2. אף תנועה אינה נועלת אותו הצומת פעמיים.

משפט: כל התרחישים של תנועות המקיימות את פרוטוקול העץ הם בני- סידור קונפליקט.
משפט: בתהליך ביצוע המקיים את פרוטוקול העץ **לא ייתכן חבק.**
<u>הערות</u>: 1. תהליך ביצוע מקיים את פרוטוקול העץ רק אם **כל תנועותיו** מקיימות אותו.
2. הפרוטוקול משתמש במנעולי **Xlock** בלבד.
3. תהליך ביצוע המקיים את פרוטוקול העץ **לא בהכרח מקיים 2PL**. התנועות מתקדמות על העץ בגלים.

פרוטוקול מבוסס – timestamps

1. כל תנועה T מקבלת תאריך ייחודי TS(T) timestamp בהתחלה
2. תנועה T2 <u>מאוחרת</u> מתנועה T1 אם $TS(T1) < TS(T2)$, (זהו יחס סדר מלא: לכל שתי תנועות מתקיים $TS(T1) < TS(T2)$ או $TS(T1) > TS(T2)$.

תרחיש כזה שקול קונפליקט לתרחיש הסדרתי המסודר לפי תאריכים.
<u>תנועה מופסקת אם:</u>
1. קוראת אובייקט שנכתב ע"י תנועה מאוחרת יותר.
2. מנסה לכתוב לאובייקט שנקרא או נכתב ע"י תנועה מאוחרת יותר.
3. קראה נתון שנכתב ע"י תנועה שהופסקה.

משפט: יהי H תרחיש המקיים את פרוטוקול ה- timestamp אם נסלק מ – H את התנוכות שהופסקו אז נקבל תרחיש בר סידור קונפליקט.
מימוש הפרוטוקול: לכל אובייקט A, נשמור שני משתנים:
LASTW(A)- התנועה המאוחרת ביותר שכתבה ל – A.
LASTWR(A) – התנועה המאוחרת ביותר שכתבה/קראה מ – A.
תמיד מתקיים TS(LASTWR(A)) ≤ TS(LASTW(A)).
PRED(T) קבוצת התנועות שכותבות נתונים שתנועה T קראה.
<u>קריאה מ – A ע"י תנועה T:</u>
אם TS(T)<TS(LASTW(A)) אז ABORT(T).
אחרת, READ A
הוסף את LASTW(A) ל – PRED(T).
אם T מאוחרת מ-LASTWR(A) אזי T = LASTWR(A).
<u>כתיבה ל – A ע"י תנועה T:</u>
אם LASTWR(A) מאוחרת מ – T אזי ABORT(T).
אחרת, WRITE A .
LASTW(A)=LASTWR(A)=T.
<u>סיום תנועה T:</u>
אם PRED(T) מכילה תנועות שהופסקו אז ABORT(T).
אם PRED(T) מכילה תנועות שטרם הסתיימו אזי T תחכה עד שהן יסתיימו ואם אחת מתנועות אלו הופסקה אז גם T תופסק.

Strict Timestamps: בקריאה מאובייקט A תחכה עד שהתנועה שכתבה אחרונה לאובייקט תסיים. בכתיבה ל – A, תחכה עד שהתנועה שכתבה אחרונה ל – A וכל התנועות שקראו אותו יסיימו. (מונע את מפולת ההפסקות).

Recovery

לכל תנועה נגדיר נקודת <u>התחייבות</u> (commit).
אם תנועה נפלה או הופסקה <u>לפני ההתחייבות</u>, כאילו שלא נעשתה כלל. <u>אחרי ההתחייבות</u> התנועה לא תופסק, ותוצאותיה ישמרו במסד הנתונים.
סוגי נפילות:
<u>נפילת דיסק:</u> נפילת חומרה.
<u>נפילת חומרה:</u> נניח שנתונים שנכתבו לדיסק נשארים תקינים.

Strict 2 phase locking:

1. התנועה מבצעת את כל הנעילות שלה לפני ההתחייבות.
2. התנועה משחררת את כל המנעולים שלה אחרי ההתחייבות

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

אלגוריתמי התאוששות:

Redo מתוחכם:

1.כשתנועה T מתחילה נרשום ביומן (T Begin).
2.כאשר T כותבת ערך v לכתובת A: נרשום (T,A,after=v) ביומן, נכתוב את v להעתק של A אך לא ל – A עצמה.
3. אם תנועה T מופסקת נרשום (T abort) ביומן. (אפשרי רק לפני ההתחייבות).
4. כשתנועה T מבצעת התחייבות נרשום (T Commit) ביומן, ונוודא שהיומן נכתב על הדיסק – זהו מימוש ההתחייבות.
5. אחרי ההתחייבות: נבצע את כל הכתיבות של התנועה על הדיסק ולאחר הכתיבות נשחרר את המנעולים.
6.**ביצוע השחזור** חוקים מהיומן תנועות שלא התחייבו
עוברים על היומן מההתחלה לסוף ומבצעים את כל פעולות הכתיבה

<u>הערות:</u>
1. תנועות שטרם גמרו בזמן הנפילה לא נרשמות לקובץ.
2. אם ישנה נפילה תוך כדי התאוששות נפעיל את אלגוריתם ההתאוששות שוב
3. בסוף ההתאוששות אף תנועה לא מחזיקה במנעול.
4. באלגוריתם redo אפשר לשמור את היומן בחוצץ ולכתוב אותו רק כאשר יש התחייבות, לפעמים דוחים התחייבויות עד שדף היומן מתמלא.

תכונות:
1. מאפשר להתאושש מנפילת מדיה
2. לא משתמש ב – before information.
3. אידמפוטנטי.
4. אם יש כתובת שכתבו אליה כמה תנועות שהתחייבו, אפשר לשמור רק את הכתיבה האחרונה של התנועה שהתחייבה.
5.אפשר לנצל checkpoints.
6. כל הכתיבות לדיסק של תנועה מבוצעות בבת אחת.

Undo מתוחכם:

1. כשתנועה T מתחילה – נרשום (T begin) ביומן
2. כאשר-T רוצה לכתוב ערך v לכתובת A:
   נקרא את הערך הישן u
   נרשום (T,A,before=v) ביומן, ונכתוב את היומן בדיסק
   נכתוב את הערך החדש v לכתובת A.
3. כשתנועה מבצעת abort
   נרשום (T abort) ביומן, ונבטל את התיבות של T.
4. כשתנועה T מבצעת התחייבות –

נוודא שכל הכתיבות של T בוצעו
נרשום (T Commit) ביומן
נוודא שהיומן נכתב על הדיסק, בסוף נשחרר את המנעולים של T.
5. **ביצוע שחזור:** קבוצת התנועות שהתחייבו commited – ריקה.
עבור על היומן מהסוף להתחלה:
עבור כניסה (T Commit), הוסף את T ל commited.
עבור כניסה (T,x,V-old)
אם T שייך ל-commited אל תעשה דבר.
אחרת, כלוב את V-old ל – x בדיסק.

תכונות:
1. מניחים שכל תנועה מתחייבת רק אחרי שכתיבותיה בוצעו לדיסק.
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.
5. אין שימוש ב- after information.
6. אידמפוטנטיות
7. מחייב קריאה לפני כתיבה.
8. האלגוריתם אינו מאפשר שחזור דיסק מגיבוי.

אלגוריתם FUZZY - שילוב בין שני האלגוריתמים:
הכתיבה תהיה לחוצצים אשר יורדו לדיסק מאוחר יותר.
באלגוריתם החדש רושמים לכל פעולה ביומן גם את אינפורמציית before וגם את after.
את המידע נכתוב אל החוצצים (כמו redo).
מדי פעם נבצע checkpoint.

<u>הפעולות שמתבצעות ב – checkpoint ה- i:</u>
1. כתוב לדיסק את כל החוצצים שהשתנו בין ה – chkp ה i-2 ל chkp ה-i1.
2. כתוב chkp i לדיסק.

<u>ביצוע שחזור (בוצעו i checkpoints)</u>
1. מבטלים פעולות של תנועות שלא התחייבו ע"י מעבר אחור מה-checkpoints ה – i-1 (כמו ב – undo).
2. מוחקים מהיומן תנועות שלא התחייבו.
3. עוברים על היומן מ – i-1 checkpoint לסוף, ומבצעים את כל פעולות הכתיבה לדיסק (כמו ב – redo).

<u>הפעולות שמתבצעות אחרי שהתנועה מבצעת התחייבות:</u>
1. מוודאים שהכניסה (T Commit) הגיעה ליומן על הדיסק. בזה מסתיים מימוש ההתחייבות
2.מבצעים את כל הכתיבות לקבצים
3. משחררים מנעולים

	התאוששות בפני נפילות	התאוששות מנפילת דיסק	bursty
Redo	כן	כן	כן
Undo	כן	לא	לא
Undo/redo	כן	כן	לא

Checkpoints:
גיבוי מלא: משהים את כל התנועות החדשות, מחכים עד שכל התנועות הפעילות ייגמרו. כותבים את כל החוצצים לדיסק, מצבעים גיבוי מלא של הדיסק .
כתיבת כל החוצצים: מאפשר להתגבר על נפילת מערכת – לא נפילת מדיה
בצע אלגוריתם undo מנקודת הנפילה אחורה.
בצע אלגוריתם redo מנקודת הביקורת עד לנפילה.
כתיבת החוצצים חדשים:
<u>Fuzzy Checkpoints</u>: נכתוב פיזית את כל החוצצים שחודשו בין נקודת הביקורת לפני הקודמת לנקודת הביקורת הקודמת ושלא נכתבו

שיטות שונות להפיכה כתיבה אחת לפעולה אטומית:
1. קריאה לאחר כתיבה
2. כתיבה כפולה – הכתיבה השנייה נעשית רק אחרי שווידאנו שהראשונה הצליחה, בקריאה נקרא מהמקום הראשון אם נכשל-מהשני
3. כתוב לסירוגין ב – 2 מקומות, בקירה נקרא משניה המקומות ונקבל את זה עם מספר גרסה גדול יותר.