מערכות קבצים

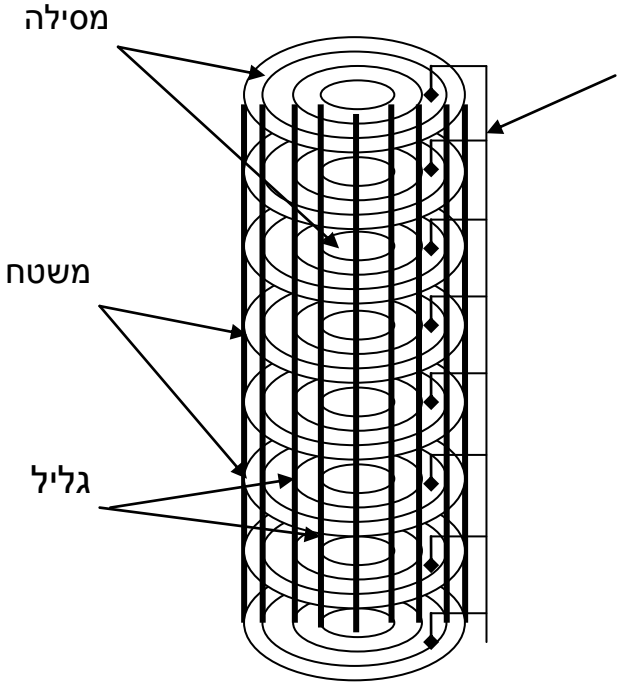
סיכום החומר בקורס "מערכות קבצים" בטכניון

סיכם: דוד ארינזון

שיפץ: אייל מוסקוביץ'

המסמך נכתב על ידי **דוד ארינזון ואייל מוסקוביץ'**

מבנה הדיסק וזמני גישות



זמן הגישה מורכב משלושה מרכיבים :

- **זמן תזוזת זרוע** (seek time)
- **זמן השהיית סיבוב** (rotation delay)
- **זמן גישה** (transfer time)

נוסחא לחישוב זמן תזוזת זרוע במעבר בין גלילים :

$$s(d) = \frac{d-2}{v} + s(2) \quad for \quad d \geq 2$$

d – מספר הגלילים שיש לעבור

v – מהירות תזוזת הזרוע (גלילים למילי שנייה)

- כאשר לא ידוע מיקום הזרוע הנוכחי או מהו היעד, אזי נקבע $d = \frac{n}{3}$, כאשר n הוא מספר הגלילים בדיסק.
- T_{rot} - **זמן סיבוב שלם**, כלומר, זמן קריאת מסילה.
- כאשר המיקום הנוכחי לא ידוע, על מנת להגיע לסקטור ספציפי במסילה, ואין ידיעה באיזה סקטור הזרוע נמצאת כרגע, נוסיף לנוסחא את הערך $\frac{T_{rot}}{2}$ (מסיבות הסתברותיות).
- נוסחא לחישוב **זמן מעבר על k סקטורים רצופים** : $\frac{k}{total\ \ sectors\ \ per\ \ track} * T_{rot}$.

בעית ארגון של קובץ סדרתי

בהינתן קובץ סדרתי, כלומר, קובץ שהקריאה והכתיבה אליו היא מההתחלה לסוף. קיימת בעיה של אחסון הקובץ בדיסק. באם הקובץ גדול מכדי להיכנס, למשל לסקטור אחד, או למסילה אחת, יש למצוא דרך לשמור אותו בצורה יעילה, על מנת לנסות לחסוך את זמן הגישה הכולל.

זמן מיתוג - הזמן שלוקח עד שהראש במשטח הבא "מתיישב מעל המסילה" ויכול להתחיל לקרוא (לכתוב).

מעבר בין מסילות

בפתרון נאיבי, אשר גורס כי יש פשוט לשמור את הקובץ מסילה אחר מסילה, בצורה רציפה, יהיה קיים זמן המיתוג (במעבר בין משטחים). זמן המיתוג הזה הוא בדר״כ בעל ערך נמוך מאוד, אולם, במקרים מסוימים הזמן הזה יכול להיות הגורם לאיבוד סיבוב שלם.

לכן, הפתרון המועדף הוא ביצוע דירוג בין מסילות עוקבות באותו הגליל. בדר״כ, נתבונן על זמן המיתוג, ולפיו נוכל להחליט, מהו הדירוג המבוקש (זאת, לפי חישוב של זמן המעבר על סקטור יחיד, ובדיקה מהי הכפולה הקטנה ביותר של הזמן הנ״ל אשר מכילה את זמן המיתוג).

מעבר בין גלילים

בעת מעבר בין שני גלילים עוקבים, יש להתייחס לזמן תזוזת הזרוע המינימלי.

קיימות שתי אפשרויות

- <u>גלילים מיושרים</u>, כלומר, נקודות ההתחלה של כל המסילות באותו משטח הן על אותו רדיוס (אחת מול השנייה).
במקרה הזה, ״נשלם״ מחיר נמוך יותר מזמן תזוזת זרוע מינימלית במעבר בין הגלילים.

- <u>דירוג בין גלילים</u>, כך שתהיה ביניהם תזוזה קבועה של סקטורים.
האפשרות הנ״ל ממזערת עוד יותר את המחיר שיש ״לשלם״ במעבר בין הגלילים, אך בדר״כ לא מאפסת אותו. בדר״כ, נתבונן על זמן תזוזת הזרוע המינימלי, ולפיו נוכל להחליט מהו הדירוג המבוקש (זאת, לפי חישוב של זמן המעבר על סקטור יחיד, ובדיקה מהי הכפולה הקטנה ביותר של הזמן הנ״ל אשר מכילה את זמן תזוזת הזרוע המינימלי).

- **מעבר אקראי על הקובץ** משמעו סדרה של גישות (לצורך קריאה/כתיבה) למיקומים בקובץ אשר אינם בהכרח קשורים אחד לשני (מבחינה רציפות). במקרה כזה, הנוסחא **לחישוב זמן קריאה של k בלוקים מתוך קובץ** הוא (בהנחה כי נקרא בלוק אחד בכל פעם):

$$num\ block(seek\ time + rotation\ delay + block\ transfer\ time)$$

RAID

- בהינתן הסתברות p לכך שדיסק כלשהו יפול בשעה, **הזמן הממוצע לנפילה לדיסק** הוא:

$$MTTF = \frac{1}{p}$$

- במערכת בעלת G דיסקים, והסתברות נפילת דיסק p, נחשב ה**סתברות** p_i **עבור נפילה של בדיוק i דיסקים**:

$$p_0 = (1-p)^G = 1 - Gp + \binom{G}{2} p^2 + O(G^3 p^3)$$

$$p_1 = \binom{G}{1} p(1-p)^{G-1} = Gp[1-(G-1)p + O(G^2 p^2)] =$$

$$Gp - G(G-1)p^2 + O(G^3 p^3)$$

$$p_2 = \binom{G}{2} p^2 (1-p)^{G-2} = \frac{G(G-1)}{2} p^2 [1 + O(Gp)] =$$

$$\frac{G(G-1)}{2} p^2 + O(G^3 p^3)$$

$$p_{\geq 2} = 1 - p_0 - p_1 \cong p_2$$

- **הסתברות נפילת מערכת בעלת N דיסקים אשר מחולקים לקבוצות של G דיסקים**:

$$P \cong \frac{N}{G} p_{\leq 2:G} \cong \frac{N}{G} \frac{G(G-1)}{2} p^2 = \frac{N(G-1)}{2} p^2$$

- **הסתברות לאובדן נתונים**: $P_{Data\ \ Loss} = \frac{1}{2} P_{System\ \ Failure} N(G-1)p^2$

- **זמן צפוי לאובדן נתונים**: $MTTDL(disk + system) = \frac{1}{P_{Data\ \ Loss}}$

- בהינתן k דיסקים, נחשב (בעזרת סטטיסטי הסדר):
  - תוחלת **מינימום השהיית הסיבוב**: $\frac{1}{k+1} T_{rot}$
  - תוחלת **מקסימום השהיית הסיבוב**: $\frac{k}{k+1} T_{rot}$

רמות RAID:

(0) אין יתירות. כלל הנתונים נכתבים לדיסקים, ולא עוברים עיבוד נוסף.

יתרון: ניצול זיכרון אופטימלי.

חסרון: אי עמידה בפני נפילות.

(1) שכפול (Mirroring). בשיטה זו, לכל דיסק A, יהיה קיים דיסק משוכפל 'A, אשר יכיל בדיוק את אותו המידע.

- בעת קריאה, זמן הגישה יהיה המינימלי מבין הגישה ל-A ול-'A.
- בעת כתיבה, זמן הגישה יהיה המקסימלי בין הגישה ל-A ול-'A (זאת, מכיוון שיש לכתוב את האינפורמציה בשניהם).

יתרון: התאוששות מהירה מתקלה בודדת.

חיסרון: ניצול זיכרון נמוך.

(2) בקודים לתיקון שגיאות בזיכרון.

(3) בזוגיות ברמת הסיבית. הדיסקים מתחלקים לקבוצות של G דיסקים, ולכל קבוצה מוסיפים דיסק זוגיות. יחידת האינפורמציה אשר תהיה כתובה בדיסק הנ"ל היא ביט.

- בעת קריאה, ניגש לכל ה-G+1 דיסקים, אך הקריאה תיגמר לאחר שייקראו ה-G הראשונים, לכן השהיית הסיבוב תהיה $\frac{G}{G+2}T_{rot}$ (חישוב בעזרת סטטיסטי הסדר).

חשוב לציין, כי בכל פעם שנרצה לקרוא מידע (אפילו בגודל byte יחיד) נצטרך לגשת לכל הדיסקים ולקרוא את אותה כתובת, לכן הזרועות של כל הדיסקים נמצאים תמיד מעל אותו גליל.

- בעת כתיבה, נצטרך לעדכן גם את דיסק הזוגיות, ולכן השהיית הסיבוב תהיה $\frac{G+1}{G+2}T_{rot}$ (חישוב בעזרת סטטיסטי הסדר).

שיטת הכתיבה לדיסק הזוגיות היא בעזרת ביצוע פעולת XOR על הביטים של שאר הדיסקים.

יתרון: קצב העברה גבוהה, ואלגוריתם פשוט.

חיסרון: זמן גישה ארוך.

(4) זוגיות ברמת הבלוק. בשונה מ-RAID 3, רמת הגרעיניות כאן היא גדולה יותר (בדר"כ סקטור או מסילה).

- <u>בעת קריאה</u>, הגישה תהיה לדיסק בודד.
- <u>בעת כתיבה</u>, הגישה תהיה בנוסף גם לדיסק הזוגיות. הכתיבה דורשת 4 פעולות ק"פ, ולפי הנוסחא הבאה: $new\quad parity = [old\quad parity] \otimes [old\quad data] \otimes [new\quad data]$.

<u>יתרון</u>: אפשרות התאוששות מוגברת, וביצוע קריאות במקביל.

<u>חסרון</u>: כתיבה איטית, וקריטיות דיסק הזוגיות (בעת כתיבה, חובה לגשת אליו).

(5) נוסף על RAID4, מתבצע דירוג של בלוקי הזוגיות בין הדיסקים.

בהינתן חלוקת RAID בעלת G דיסקים, ודיסק זוגיות אחד:

- **מציאת כתובת ומס' הדיסק שבו נמצא בלוק הזוגיות של הבלוק ה-n**:
  - **מס' מסילה**: $\boxed{k = \left\lfloor \frac{n}{G} \right\rfloor}$.
  - **מס' גליל**: $\boxed{\left\lfloor \frac{k}{num\quad tracks} \right\rfloor = \left\lfloor \left\lfloor \frac{n}{G} \right\rfloor \Big/ num\quad tracks \right\rfloor}$.
  - **מספר משטח**: $\boxed{k \bmod (num\quad tracks) = \left\lfloor \frac{n}{G} \right\rfloor \bmod (num\quad tracks)}$.
  - **מספר דיסק**: $\boxed{m = G - \left(\left\lfloor \frac{n}{G} \right\rfloor \bmod (G+1) \right)}$.
- **מציאת כתובת ומס' הדיסק שבו נמצא הבלוק ה-n**:
  - **מס' דיסק**: $\boxed{m = \begin{cases} n \bmod G & n \bmod G < \left(G - \left(\left\lfloor \frac{n}{G} \right\rfloor \bmod (G+1) \right) \right) \\ (n \bmod G) + 1 & n \bmod G \geq \left(G - \left(\left\lfloor \frac{n}{G} \right\rfloor \bmod (G+1) \right) \right) \end{cases}}$.
  - היות ולכל הבלוקים באותה שורה, אותה כתובת (רק בדיסקים שונים) הכתובת תחושב כמו בסעיף קודם.

(6) P+Q redundancy

- יש שימוש בקוד Reed-Solomon לתיקון שגיאות.
- כל כתיבה מחייבת 6 פעולות קלט/פלט.

מימוש/שמירת קבצים סדרתיים על הדיסק

שיקולים בבחירת צורת השמירה:

- צורך בקריאה מהירה של נתונים באופן סדרתי.
- צורך בקריאה מהירה של נתונים בגישה ישירה.
- ניצול יעיל של שטח הדיסק (מניעת שברור).
  - שברור חיצוני – שטח הדיסק לא מנוצל בין השטחים אשר מוקצים לקבצים שונים.
  - שברור פנימי- שטח דיסק לא מנוצל בתוך השטחים אשר מוקצים לקבצים שונים.

אפשרויות הקצאה עבור קובץ סדרתי

1. **הקצאה רציפה**

יתרונות:

- גישה אקראית לפי המספר הסידורי של הרשומה.
- קריאה רציפה ויעילה.

חסרונות:

- הוספת רשומות (אפילו בסוף קובץ), מחייבות העתקה של כל הקובץ.
- קושי בעדכון רשימות.
- יש צורך למצוא מקום גדול ורציף בדיסק לטובת הקובץ.
- יש לדעת את גודל הקובץ בעת ההקצאה לדיסק.

2. **הקצאה בגושים**

חלוקה של הקובץ לגושים בעלי גודל קבוע. כל גוש יוקצה באופן רציף, אולם כלל הגושים יוקצו באופן אקראי.

יתרונות:

- מניעת שברור חיצוני ופנימי (בעזרת גודל מתאים של גוש).

חסרונות:

- בזבוז מקום עבור קבצים קטנים

אפשרויות מימוש

- רשימה מקושרת – כל גוש יכיל מצביע לגוש הבא.

  יתרונות:

  - אין שברור חיצוני.
  - אין צורך בהקצאה מראש עבור הקובץ.

חסרונות:

- איבוד נצילות עקב מצביעים.
- אין אפשרות גישה אקראית לקובץ.

- תקלה בבלוק כלשהו מייצרת בעיה עבור כלל הקובץ.

- הקצאה מבוססת אינדקס – לכל קובץ יהיה גוש אינדקס אשר יכיל מצביעים לכלל הגושים המוקצים עבורו.

יתרונות:

- אין שברור חיצוני.
- אין צורך בהקצאה מראש עבור הקובץ.
- קיימת גישה ישירה עבור הקובץ.

חסרונות:

- איבוד נצילות עקב הקצאת גוש נוסף ומצביעים.

- יש צורך בטיפול מיוחד בקבצים גדולים (למשל במקרה שבו מס' האינדקסים לא נכנסים בגוש אחד).

שיטות ארגון המקום החופשי בדיסק

1. **bitmap** – הקצאת סיבית עבור כל גוש, כאשר 1 מייצג גוש מנוצל.
2. **רשימה מקושרת** – כל גוש חופשי יכיל מצביע לגוש החופשי הבא.
3. **טבלה** אשר תכיל מצביעים לכלל הגושים החופשיים.

שיטות ארגון גושים מנוצלים

1. רשימות מקושרות
2. טבלת הקצאות FAT (טבלה נפרדת עבור המצביעים לגושים)
3. מבנה עץ (ה-inode המוכר מ-UNIX).
4. NTFS

אלגוריתם R/S ליצירת מעלות התחלתיים ומיון מיזוג

- **מעלה** – סדרה עולה מקסימלית.

אלגוריתם ליצירת מעלות התחלתיים

רוצים למצוא אלגוריתם אשר במעבר סידרתי יחיד על הקובץ בונה מעלות בגודל ממוצע מרבי.

נניח לרשותנו זיכרון ראשי של M רשומות.

(RS) Replacement Selection

אתחול: ממלאים את הזיכרון ב-M רשומות ראשונות.

בכל שלב מוציאים לפלט את הרשומה עם המפתח הקטן ביותר שעדיין גדול מהמפתח האחרון שהוצא. אם אין כזאת – מוציאים את הרשומה עם המפתח המינימלי.

- **עבור קלט אקראי – תוחלת אורך המעלות בפלט הינו 2M רשומות.**

- **רשומה קפואה** – רשומה אשר המפתח שלה קטן יותר מהמפתח האחרון אשר הוצא מהזיכרון.

כלל אצבע: אם עבור זיכרון בגודל M נקבל K מעלות אז עבור זיכרון גדול מ- M נקבל לכל היותר K מעלות.

מבנה הזיכרון ב-RS

- בהינתן שני דיסקים, אחד לקלט והשני לפלט, וזיכרון בגודל M. נשתמש בשיטת double buffering, ונחלק את הזיכרון כך שיהיו לו שני חוצצי קלט ושני חוצצי פלט. בכל איטרציה של האלגוריתם, נקרא מאחד מחוצצי הקלט, ונכתוב לאחד מחוצצי הפלט.
- עבור מיזוג מסדר k, מספר מעברי המיזוג הוא $\lceil \log_k n \rceil$, כאשר n הוא מספר המעלות.
- מיון מסדר k הינו חילוק הקובץ ל-k חלקים בכל איטרציה, ומיזוג בין התלמים המקבילים מבין כלל הקבוצות (מקבילים מבחינת אינדקסים).

מבנה הזיכרון והקצאת חוצצים עבור מיון המיזוג

- בהינתן שני דיסקים, אחד לקלט והשני לפלט, עלינו לבצע מיזוג מסדר k. לשם כך, נזדקק לשני חוצצי פלט, חוצץ אחד עבור כל אחד מ-k המעלות, ועוד k חוצצים נוספים אשר אותם יוכל דיסק הקלט למלא בזמן שה-k חוצצים הראשונים מעובדים. בצורה הזאת נגרום לשני הדיסקים לעבוד בצורה רציפה.
- מאיזה מעלה, על הדיסק, נקרא אל החוצץ הריק?
נקצה חוצצים לפי **עיקרון החיזוי** , לפי שיטה זו קוראים חוצץ עבור המעלה שצפוי ל"הגמר" ראשון.

עבור כל מעלה נסמן ב- $last_i$ את המפתח הגדול ביותר שלו שנמצא בזיכרון. נקרא חוצץ עבור המעלה המקיים $\min_i\{last_i\}$.

עצי +B

- **m – דרגת העץ המקסימלית**, קרי, **מספר הבנים המקסימלי לכל צומת**.
- בהינתן דרגה d של צומת יתקיימו הטענות הבאות:
  - עבור השורש $2 \le d \le m$.
  - עבור כל צומת פנימי יתקיים $\left\lceil \frac{m}{2} \right\rceil \le d \le m$.
- **B – מספר הרשומות המקסימלי בעלה**.
- בהינתן **מס' רשומות m בעלה**, תמיד יתקיים $\left\lceil \frac{B}{2} \right\rceil \le b \le B$.
- הרשומות ישבו תמיד בעלים, כאשר הצמתים הפנימיים (כולל השורש) יחזיקו אינדסקים.

  בהינתן דרגה d של צומת, הוא יחזיק:
  - d-1 אינדקסים.
  - d מצביעים לרמה מתחת.

- **בחישוב של גובה העץ**, נהוג להשתמש **בתפוסה ממוצעת**, אשר מתבטאת בהכפלה בפקטור $\ln(2) \cong 0.7$.

- בהינתן עץ בגובה h, מספר העלים L מקיים:
  - $L_{\min} = 2\left\lceil \frac{m}{2} \right\rceil^{h-1}$
  - $L_{\max} = m^h$.
- בהינתן L עלים, גובה העץ מקיים:
  - $h_{\min} \lceil \log_m L \rceil = \Leftarrow m^{h_{\min}-1} < L \le m^{h_{\min}}$
  - $h_{\max} \le 2 + \frac{\log_2 L - 1}{\log_2 m - 1} \cong 2 + \frac{\log_2 L}{\log_2 m} \Leftarrow 2\left\lceil \frac{m}{2} \right\rceil^{h_{\max}-1} \le L$

כאשר L ו-m גדולים נקבל את הקירוב $h_{\max} - h_{\min} \le 2$.

- כאשר יש קריאות רבות, כדאי להגדיר עלים קטנים.
- כאשר יש קריאות מעטות, כדאי להגדיר עלים גדולים.

- **שדרוג נפוץ לעלים הוא יצירת רשימה מקושרת**, הדבר מייעל קריאה של כל רשומות הקובץ בצורה סדרתית.

אינדקס משני

	נעוץ	בלתי-נעוץ	משולב (נעוץ ובלתי-נעוץ)
מבנה רשומה בעלה	אינדקס משני + מצביע לעלה המתאים בעץ הממוין לפי אינדקס ראשי	אינדקס משני + אינדקס ראשי	אינדקס משני + אינדקס ראשי + מצביע לעלה המתאים בעץ הממוין לפי אינדקס ראשי
אופן הכנסה	▪ הכנס את הרשומה החדשה בעזרת האינדקס הראשי לעלה L. ▪ הוסף לאינדקס המשני מיני-רשומה המכילה מפתח משני וכתובת של L. אם העלה של האינדקס הראשי פוצל, עדכן את כל המפתחות באינדקס המשני שהצביעו לרשומות שנדדו לעלה אחר.	▪ הכנס את הרשומה החדשה בעזרת האינדקס הראשי. ▪ הוסף לאינדקס המשני מיני-רשומה המכילה מפתח משני ומפתח ראשי.	▪ הכנס את הרשומה החדשה בעזרת האינדקס הראשי לעלה L. ▪ הוסף לאינדקס המשני מיני-רשומה המכילה מפתח משני, מפתח ראשי וכתובת לעלה L.
אופן חיפוש	▪ חפש באינדקס המשני, קבל כתובת של עלה. ▪ קרא את העלה וחפש את הרשומה בו.	▪ חפש באינדקס המשני, קבל מפתח ראשי k. ▪ חפש את k באינדקס הראשי.	▪ חפש באינדקס המשני, קבל כתובת של עלה ומפתח ראשי k. ▪ קרא את העלה וחפש את הרשומה בו. ▪ אם לא נמצאה - חפש את k באינדקס

			הראשי ועדכן את כתובת העלה באינדקס המשני.
חסרונות	כשמעבירים רשימות ממקומם בקובץ (למשל, באיחוד או פיצול) יש לעדכן את המיני-רשומות הרלוונטיות בכל האינדקסים המשניים. יתכן שיש הרבה אינדקסים משניים (חלקם אפילו זמניים).	זמן החיפוש כפול, כי מהאינדקס המשני מקבלים את המפתח הראשי ולא את מיקום הרשומה.	

<u>יתרונות הפיתרון המשולב</u>:

- לא מבזבזים זמן מיותר לעדכן כל האינדקסים המשניים בזמן העברת רשומה ממקום למקום.
- בזמן חיפוש רשומה משלמים מחיר חיפוש כפול (או אפילו קצת יותר) רק בפעם הראשונה שניגשים אליה דרך אותו אינדקס משני לאחר העברתה.
- כשיש הרבה רשומות עם אותו מפתח משני ניתן לעתים לזהות את הרשומה המבוקשת ע״פ המפתח הראשי מבלי לגשת ולבדוק את כל הרשומות האפשריות.

אלגברה רלציונית

פעולות אפשריות:

- **איחוד** $R \cup S$ - מספר המופעים של שורה באיחוד שווה לסכום המופעים שלה בשני היחסים.
- **חיתוך** $R \cap S$ - מספר המופעים של שורה בחיתוך שווה למינימום של מספר המופעים שלה בשני היחסים.
- **הפרש** $R \setminus S$ - מספר המופעים של שורה בהפרש שווה להפרש בין מספר המופעים שלה בשני היחסים. (רק שורות מתוך R).
- **מכפלה** $R \times S$ - שרשור של כל השורות ב-R עם כל השורות ב-S.
- **Join** $R \rhd\lhd S$ - ב-join של שני היחסים תופיע שורה לכל זוג (r,s) המקיים:
  - $r \in R, s \in S$
  - r ו- s מסכימים על כל התכונות המשותפות להם.

  ה-join יכיל רק עמודה אחת לכל תכונה משותפת.

- **Select** $\sigma_C(R)$ - בוחר את כל השורות ב-R המקיימות את התנאי c.
- **הטלה (projection)** $\pi(R)$ - בחירת חלק מהעמודות. מאפשר חישוב תכונות ושינוי שם: לדוגמה, הביטוי $\pi_{a,b+c\to d}(R(a,b,c))$ - יחזיר יחס שיכיל את עמודה a ועמודה חדשה d שתהיה הסכום של העמודות b ו- c ב-R.
- **Distinct** $\delta(R)$ - הפיכת יחס לקבוצה, משאירה רק שורה אחת מכל סוג.

שיוויונות של אלגברה רלציונית:

- אסוסיטיביות של שילוב (join): $(R \rhd\lhd S) \rhd\lhd T = R \rhd\lhd (S \rhd\lhd T)$.
- קומוטטיביות של שילוב (join): $R \rhd\lhd S = S \rhd\lhd R$.
- הורדת בחירה (select): $\sigma_C(R \cap S) = \sigma_C(R) \cap S = R \cap \sigma_C(S) = \sigma_C(R) \cap \sigma_C(S)$.
- **אם** תנאי הבחירה רלוונטי ליחסים התאימים:

  $\sigma_C(R \rhd\lhd S) = \sigma_C(R) \rhd\lhd S = R \rhd\lhd \sigma_C(S) = \sigma_C(R) \rhd\lhd \sigma_C(S)$.
- הורדת δ: $\delta(R \rhd\lhd S) = \delta(\delta(R) \rhd\lhd S) = \delta(R \rhd\lhd \delta(S)) = \delta(R) \rhd\lhd \delta(S)$.

הגדרות גדלים

תהי R רלציה, נגדיר את הסימונים הבאים:

- $B(R)$ - מספר הבלוקים ש-R תופסת.
- $T(R)$ - מספר הרשומות של R (מספר השורות בטבלה).
- $V(R)$ - מספר הרשומות השונות של R.

- $V(R,A)$ - מספר הערכים השונים של התכונה A ביחס R.
- M -מספר הבלוקים העומדים לרשותנו בזיכרון הראשי.

שיטות מימוש פעולות

<u>מעבר יחיד</u>

- קל לבצע פעולות כמו max, min, average במעבר יחיד (דורש O(1) זיכרון).
- *פעולות אונריות המחזירות יחס* : קל לבצע Select והטלה (דורש O(1) זיכרון). מימוש של במעבר יחיד דורש ש- $\delta(R)$ ייכנס כולו בזיכרון הראשי.
- *פעולות בינאריות על יחסים* : קל לבצע איחוד (כותבים לפלט את שני היחסים אחד אחרי השני). כדי לבצע את יתר הפעולות : חיתוך,הפרש, מכפלה ו- join דרוש שלפחות אחד מהיחסים R ו- S (עליהם מתבצעת הפעולה) ייכנס לזיכרון.
- *כדי לאפשר קריאה רציפה של הרלציות, נזדקק לשני חוצצי קלט. וכדי לאפשר כתיבה רציפה של הפלט גם כן נשתמש בשני חוצצים (double buffering).*

לצורך זאת, נדרוש כי יתקיים $B(R) \le M-4$.

זהו מעבר יחיד, ולכן מן הסתם, מספר הגישות לדיסק הוא $T(R)$.

אלגוריתמי מעבר יחיד

הפעולה	דרישות זיכרון	גישות דיסק
ממוצע, π , σ	O(1)	$B(R)$
איחוד	O(1)	$B(R)+B(S)$
δ במעבר יחיד	$\delta(R)$	$B(R)$
הפרש, חיתוך, $\times$, $\rhd\lhd$	$\delta(S)$	$B(R)+B(S)$

<u>לולאות מקוננות</u>

- מכיוון שקוראים רשומה רשומה, מספר הגישות לדיסק הוא בערך $T(S)\cdot[T(R)+1]$.
- במקום להשתמש בשתי לולאות (בזבזניות), ניתן לבצע שיפור, ובכל פעם לקרוא $M-4$ חוצצים, ואז מספר האיטרציות יירד ל- $\left\lceil \frac{B(S)}{M-4} \right\rceil$.

- אזי נוכל לקבל סה"כ $\left\lceil \frac{B(S)}{M-4} \right\rceil \cdot \left[B(R)+\left(M-4\right) \right]=\left\lceil \frac{B(S)}{M-4} \right\rceil \cdot B(R)+B(S)$ גישות לדיסק.

<u>שיטות מבוססות מיון</u>

- לעיתים, מיון מקל על חלק מהפעולות הבינאריות/אונאריות להתבצע. לצורך המיון, ניתן להשתמש באלגוריתם RS + מיון מיזוג, כפי שנלמד בכיתה.

אלגוריתמי מעבר יחיד

הפעולה	מספר הבלוקים בזיכרון הראשי	מספר גישות דיסק לקלט
δ	1	$B(R)$
Min, Max	1	1
איחוד, חיתוך	2	$B(R)+B(S)$
$\triangleright\triangleleft$	2	$B(R)+B(S)$

<u>שיטות מבוססות ערבול</u>

- הרעיון: הערבול מאפשר לרכז את כל הרשומות שרוצים לעבד יחד באותו דלי.
  - למשל כל המופעים של אותה רשומה כאשר מבצעים חיתוך
- אם הדלי נכנס לזיכרון, נוכל לבצע את הפעולה במעבר יחיד (לאחר החלוקה לדליים).
- בשלב הראשון נחלק כל יחס לדליים.
  - נזדקק לשני חוצצים לכל דלי פלט, ועוד זוג חוצצים לקלט.

- **מספר דליים מקסימלי**: $k=\frac{M-2}{2} \Leftarrow 2k+2=M$.
- בהינתן גודל קובץ $B\left(R\right)$, **גודל דלי ממוצע** יהיה $\frac{B\left(R\right)}{k}=\frac{2\cdot B\left(R\right)}{M-2}$.
- **על מנת שדלי כלשהו ייכנס לזיכרון**, נדרוש $\frac{2}{M-2}\cdot B\left(R\right)\le M-4$.

חישוב מספר הגישות לדיסק:

- בשלב החלוקה לדליים של שני היחסים $2\cdot B\left(R\right)+2\cdot B\left(S\right)$ (קריאה וכתיבה של כל הבלוקים).

- לאחר מכן עוד מעבר יחיד: $B(R)+B(S)$ . .

סה"כ $3B(R)+3B(S)$ [החישוב הנ"ל אינו כולל את הכתיבה לדיסק!]

- עבור $B>\frac{M^2}{2}$, נחליף את המקדם 3 ל-5.
- עבור קבצים גדולים יותר ניתן לבצע מעבר ערבול נוסף שבו נחלק כל דלי ל- $(M-2)/2$ דליים. אם ישנו צורך, נמשיך לבצע מעברים, כאשר בכל מעבר נוסף נחלק כל דלי ל-k דליים. באם התבצעו h מעברים כאלה (כולל המעבר הראשון) נקבל:

מספר המעברים: $h=\lceil \log_k(B(R)/M) \rceil$, **מספר גישות לדיסק**: $(2h+1)B(R)$.

- בהינתן ביטוי, נרצה לקבל אומדן על הדרך הטובה (מהירה/חסכונית) ביותר לחשב את הביטוי. נעשה זאת תו"כ הסתמכות על אומדן ותוצאות ביניים. (בעזרת הטבלה הבאה), וכך, נבנה עץ חישוב אופטימלי.

טבלת נוסחאות לחישוב גודל התוצאה

הפעולה	נוסחא	הערות והסברים
$\sigma_{A=a}(R)$	$E\left(T\left(\sigma_{A=a}(R)\right)\right)=T(R)/V(R,A)$	לא תלוי בהתפלגות של A. מניחים רק ש- a נבחר מבין הערכים האפשריים של A בהסתברות שווה.
$\sigma_{A<a}(R)$	אם ידוע ההתפלגות: $E\left(T\left(\sigma_{A<a}(R)\right)\right)=P(A<a)*T(R)$ אחרת, נניח שההתפלגות היא יוניפורמית במקרה הרציף אם ידוע התחום $[a_0,b_0]$ אז, $E\left(T\left(\sigma_{A<a}(R)\right)\right)=\frac{a-a_0}{b-b_0}*T(R)$ ובאופן דומה במקרה הדיסקרטי אם התחום לא ידוע, נניח באופן שרירותי $E\left(T\left(\sigma_{A<a}(R)\right)\right)=T(R)/3$.	
$\pi(R)$	$T(R)*f$, כאשר גודל השדות שנבחרו f בתים.	

$R(X,Y) \triangleright\triangleleft S(Y,Z)$	$E\left(T\left(R \triangleright\triangleleft S\right)\right) = \\ = \frac{T\left(R\right) * T\left(S\right)}{\max\left\{V\left(S,Y\right), V\left(R,Y\right)\right\}}$	תחת ההנחות הבאות: • Y מכיל תכונה בודדת. • אם Y_R ו Y_S הן קבוצות הערכים של התכונה Y ב R וב S אז $Y_R \supseteq Y_S$ או $Y_S \supseteq Y_R$. • שמירת ערכים: $V\left(R \triangleright\triangleleft S, A\right) = V\left(R, A\right)$.
$\delta(R)$	$T\left(\delta\left(R\right)\right) = V\left(R\right) = \\ = \min\left\{\prod_{1 \le i \le n} V\left(R, a_i\right), \frac{1}{2} T\left(R\right)\right\}$	"כלל אצבע"

בקרת מקבילות

- **תנועה** – קבוצת פעולות שיש ביניהן קשר לוגי מסוים.
- **תהליך הביצוע** של מספר תנועות נתונות הוא הסדר בו בוצעו הפעולות המרכיבות אותן על ידי המחשב.
- תהליך ביצוע הוא **בר-סידור** אם הוא *שקול* לתהליך ביצוע סדרתי.
- בקר מקבילות הוא **נכון** אם כל תהליכי הביצוע שלו הם ברי סידור.
- שני תהליכי ביצוע הם **שקולי מצב סופי** אם לכל מצב התחלתי ולכל *סמנטיקה (חישוב)* של התנועות, שני התהליכים מביאים את מערכת הקבצים לאותו מצב סופי.
- תהליך ביצוע הוא **בר-סידור מצב סופי** אם הוא **שקול מצב סופי לתהליך סדרתי**.
- שתי פעולות הן **מתנגשות** אם הן מתייחסות לאותו נתון (אותה כתובת) ולפחות אחת מהן היא כתיבה. סוגי התנגשויות:

- R\W – write after read
- W\W – write after write
- W\R – read after write

- שני תהליכי ביצוע הם **שקולי קונפליקט** אם:
1. הם מוגדרים על אותה קבוצת תנועות.

2. הם מסכימים על הסדר בין פעולות מתנגשות.

כלומר, אם פעולה P_1 מתנגשת בפעולה P_2 ומקדימה אותה ב-H_1, אזי גם ב-H_2 P_1 מקדימה את P_2.

- תהליך ביצוע הוא **בר-סידור קונפליקט** אם הוא **שקול קונפליקט לתהליך סדרתי**.
- אם תהליך ביצוע הוא **בר סידור קונפליקט** אז הוא גם **בר-סידור מצב סופי**.
- תהליך ביצוע הוא **בר סידור קונפליקט** אם"ם גרף התלויות שלו חסר מעגלים מכוונים.
- תנועה מכילה **קריאה מיותרת** אם התנועה מבצעת קריאה שאין אחריה אף כתיבה.
- תנועה מכילה **כתיבה עיוורת** אם התנועה מבצעת כתיבה לכתובת שלא קראה ממנה קודם לכן. למשל, T = R(X)W(X)W(Y), הכתיבה ל-Y היא עיוורת.
- אם תהליך ביצוע **בר-סידור מצב סופי** אינו מכיל כתיבות עוורות וקריאות מיותרות אז תהליך הביצוע הוא גם **בר סידור קונפליקט**.
מכאן נסיק כי:

אם תהליך ביצוע אינו מכיל כתיבות עוורות וקריאות מיותרות אז, הוא בר סידור קונפליקט אמ"מ הוא בר סידור מצב סופי.

- תרחיש H_1 **שקול מבט** לתרחיש H_2 אם:

- אם בתרחיש H_1 תנועה T קוראת ערך v למשתנה x אזי גם בתרחיש H_2 התנועה T קרואת ערך v למשתנה x.
- אם בתרחיש H_1 תנועה T_1 קוראת ל-x ערך שכתבה תנועה T_2, אזי גם בתרחיש H_2 התנועה T_1 קוראת ל-x ערך שכתבה התנועה T_2.
- אם בתרחיש H_1 תנועה T היא האחרונה שכותבת לערך ל-x, אזי גם בתרחיש H_2 התנועה T היא האחרונה הכותבת ערך ל-x.

- תרחיש הינו **בר-סידור מבט** אם הוא **שקול מבט לתרחיש סדרתי**.
- אם תרחיש H_1 **שקול מבט לתרחיש** H_2, אזי תרחיש H_1 **שקול מצב סופי לתרחיש** H_2.
- אם תרחיש A הינו **בר-סידור מבט**, אזי הוא **גם בר-סידור סופי**.
- אם תרחיש הוא **בר-סידור קונפליקט**, אזי הוא **בר-סידור מבט**.

מנעולים

- תנועה שרוצה לקרוא פריט במבנה הנתונים צריכה להשיג קודם מנעול מסוג Slock על אותו פריט.
  (ניתן לאפשר מספר קריאות במקביל, ולכן נאפשר למספר תזמונים לתפוס מנעול Slock).
- תנועה שרוצה לשנות (לכתוב) פריט במבנה הנתונים צריכה להשיג קודם מנעול Xlock על אותו פריט.
  (המנעול הנ"ל הוא אקסקלוסיבי, ולכן רק תזמון יחיד יכול לתפוס אותו ברגע זמן כלשהו)

פרוטוקול נעילה דו שלבית 2 Phase Locking (2PL)

- תנועה מקיימת את פרוטוקול 2PL אם:
  - התנועה שומרת על כללי הגישה והנעילה.
  - כל הנעילות שהתנועה מבצעת מופיעות לפני השחרור הראשון.
- תהליך ביצוע מקיים את פרוטוקול 2PL אם:
  - **כל** התנועות שלו מקיימות את פרוטוקול 2PL.
- אם תהליך ביצוע מקיים 2PL אזי התהליך הוא **בר-סידור קונפליקט**.

גרסאות מיוחדות של 2PL

- **Strict 2PL** – שחרור המנעולים נעשה רק בשלב האחרון כחלק מסיום הריצה, כלומר לאחר שהסתיימו כל פעולות הקריאה והכתיבה (מפורט בהמשך בהתאוששות מנפילות).
- **Conservative 2PL** – כל בקשות המנעולים וקבלתם נעשים בתחילת ריצת התנועה לפני כל פעולות הקריאה והכתיבה, מונע חבק מפני שתנועה מתחילה ריצה רק כאשר יש לה את כל המנעולים המתאימים לכל הפעולות שהיא רוצה לבצע.

בקרת מקביליות על עצים

פרוטוקול העץ

תנועה מקיימת את פרוטוקול העץ אם:

- פרט לצומת הראשון שהתנועה נועלת, תנועה יכולה לנעול צומת **רק** אם היא מחזיקה מנעול על אביו.
- **תנועה אינה נועלת אף צומת פעם שנייה.** כלומר, ברגע שתנועה שחררה מנעול מצומת היא לא מנסה להשיג שוב מנעול על אותו הצומת.

- תהליך ביצוע מקיים את פרוטוקול העץ אם:
  - **כל** התנועות שלו מקיימות את פרוטוקול העץ.
- תהליך ביצוע המקיים את פרוטוקול העץ הוא **בר-סידור קונפליקט**.
- בתהליך ביצוע המקיים את פרוטוקול העץ **לא יתכן חבק**.

פרוטוקול Timestamps

- כל תנועה T מקבלת timestamp ייחודי, TS(T) בהתחלה.
- תנועה T2 מאוחרת מתנועה T1 אם TS(T2)>TS(T1) (זהו יחס סדר מלא)
- תנועה **מבוטלת** במקרים הבאים:
  - קוראת אובייקט שכבר נכתב ע"י תנועה מאוחרת יותר
  - מנסה לכתוב לאובייקט שכבר נקרא או נכתב ע"י תנועה מאוחרת יותר.
  - קראה נתון שכבר נכתב ע"י תנועה שהופסקה.
- תרחיש של הפרוטוקול הזה הינו **שקול קונפליקט לתרחיש הסדרתי המסודר לפי חותמות זמן**.

מימוש

- לכל אובייקט A נשמור שני משתנים:
  - LastW(A) – התנועה המאוחרת ביותר שכתבה ל-A.
  - LastWR(A) – התנועה המאוחרת ביותר שכתבה/קראה מ-A.

  מההגדרה נובע כי TS(LastWR(A)) ≥ TS(LastW(A)).

- לכל תנועה T נשמור את המשתנה
- Pred(T) – קבוצת התנועות שכתבו נתונים שתנועה T קראה.

<u>קריאה מ-A ע"י תנועה T</u>

1. אם TS(T) < TS(LastW(A)) אזי נבצע ABORT(T)
2. אחרת
   1.2. READ(A)
   2.2. הוסף את LastW(A) ל-Pred(T)
   3.2. אם T מאוחרת מ-LastWR(A) אזי LastWR(A) = T

<u>כתיבה ל-A ע"י תנועה T</u>

1. אם LastWR(A) מאוחרת מ-T אזי ABORT(T)
2. אחרת
   1.2. WRITE A
   2.2. LastW(A) = LastWR(A) = T

סיום תנועה T

- אם Pred(T) מכילה תנועות שהופסקו אזי נבצע ABORT(T)
- אם Pred(T) מכילה תנועות שטרם הסתיימו, אזי T תחכה עד שכולן תסתיימנה בהצלחה. אם אחת מהתנועות הופסקה, גם T תופסק.

- יהי H תרחיש **המקיים את פרוטוקול Timestamp**, אם נסלק מ-H את התנועות שהופסקו אז נקבל **תרחיש בר-סידור קונפליקט**.

Strict Timestamps

בקריאה מאובייקט A, חכה עד שהתנועה שכתבה אחרונה לאובייקט תסיים. בכתיבה ל-A, חכה עד שהתנועה שכתבה אחרונה ל-A וכל התנועות שקראו אותו תסיימה (מונע מפולת של הפסקות).

התאוששות מנפילות

- עבור כל תנועה נגדיר נקודת התחייבות (COMMIT), כאשר אם פעולה עוברת את הנקודה הזאת בהצלחה, אזי נגיד שהפעולה הסתיימה בהצלחה.
- סיבות להפסקה/אי השלמה של פעולה כלשהי:
  - Aborts (כתוצאה מהחלטה פנימית, התערבות של משתמש או החלטה של מ"ה).
  - Failures (נפילת חומרה או מערכת).

פרוטוקול Strict 2PL (S2PL)

- תנועה מקיימת פרוטוקול S2PL אם:
  - היא מקיימת את פרוטוקול 2PL
  - היא משחררת את כל המנעולים מייד אחרי ביצוע ההתחייבות, ורק אז. (כדי שלא נצטרך להתיר (לבטל) תנועות שכבר התחייבו, בגלל שהן השתמשו במידע שנכתב ע"י תנועות שעוד לא התחייבו)
- תהליך ביצוע מקיים את פרוטוקול S2PL אם כל התנועות שלו מקיימות את פרוטוקול S2PL.

שימוש ביומן (Log)

- מקובל להשתמש ביומן כדי לאפשר התאוששות.
- היומן הוא קובץ סדרתי הנמצא בדרך כלל על דיסק נפרד.
- מערכת הקבצים כותבת (כמעט) ברצף לסוף היומן.
  - מעט המתנה בכתיבה ליומן, משמעה כתיבה יעילה.

אלגוריתם Redo

- לא כותבים לקבצים עדכונים שלא התחייבו עליהם.
- את העדכונים האלה כותבים ביומן, ומעדכנים את הקבצים רק לאחר שהתנועה התחייבה.

המידע שנכתב ביומן עבור תנועה T:

- תחילת תנועה (T, Begin)
- התחייבות (T, Commit)
- סיום לא מוצלח (T, Abort)
- כתיבה של ערך V לפריט X (T,X,V)

הפעולות שמתבצעות אחרי שהתנועה מבצעת התחייבות:

- מוודאים שהכניסה (T, Commit) הגיעה ליומן על הדיסק.
  - בזה מסתיים **מימוש ההתחייבות** – אחרי פעולה זו מובטח שהתנועה תסתיים בהצלחה, גם במקרה של נפילה.
- מבצעים את כל הכתיבות לקבצים.
- משחררים מנעולים.

ביצוע שיחזור:

- מוחקים מהיומן תנועות שלא התחייבו.
- עוברים על היומן **מההתחלה לסוף**, ומבצעים את כל פעולות הכתיבה.

יתרון:

מימוש התחייבות מהיר.

חסרון:

- כל פעולות הכתיבה מרוכזות לאותה נקודת זמן.
- השחזור הוא תהליך ארוך – יש לבצע מחדש את כל הפעולות שכתובות ביומן.

אלגוריתם Undo

- רוב התנועות מסתיימות בהצלחה, ולכן כדאי לבטל רק תנועות שנכשלו.
- כותבים עדכונים ישירות לקבצים.
- את הערכים הקודמים של הפריטים כותבים ליומן **לפני** שכותבים את הערך החדש לפריט (לדיסק).

המידע שנכתב ביומן עבור תנועה T:

- תחילת תנועה (T, Begin)
- התחייבות (T, Commit)
- סיום לא מוצלח (T, Abort)
- כתיבה לפריט X שערכו הקודם V-old (T, X, V-old)

הפעולות שמתבצעות אחרי שהתנועה מבצעת התחייבות:

- מוודאים שכל הכתיבות של התנועה הגיעו לדיסק.
- מוודאים שהכניסה (T, Commit) הגיעה ליומן על הדיסק.
  - בזה מסתיים **מימוש ההתחייבות** – אחרי פעולה זו מובטח שהתנועה תסתיים בהצלחה, גם במקרה של נפילה.
- משחררים מנעולים.

ביצוע שיחזור:

- עוברים על היומן **מהסוף להתחלה**, ועבור כל תנועה שלא התחייבה כותבים לקבצים את הערכים הישנים.

יתרון:

- פעולות הכתיבה מבוצעות במשך כל זמן ביצוע התנועה ולא מרוכזות רק בסוף.

חסרון:

- מימוש התחייבות איטי.

אלגוריתם FUZZY

- כדי לחסוך בזמן, אנחנו רוצים שהכתיבה ב-online תהיה לחוצצים אשר "יורדו" לדיסק מאוחר יותר.
- רוצים לשחזר את הדיסק מבלי לעבור על כל היומן מההתחלה.
- נשתמש באלגוריתם המשלב בין redo ו- undo.
- ביומן נחזיק אינפורמציית before ו- after
- את המידע נכתוב לחוצצים (כמו ב-redo)
- מדי פעם נבצע checkpoint:
  - במהלכו נעביר חלק מהחוצצים לדיסק, כך שלא יהיו תנועות "מוקדמות" שלא נכתבו לדיסק.

המידע שנכתב ביומן עבור תנועה T:

- תחילת תנועה (T, Begin)
- התחייבות (T, Commit)

- סיום לא מוצלח (T, Abort)
- כתיבה לפריט X שערכו הקודם V-old (T, X, before=V-old, after=V)

שומרים רשימה של כל הפעולות שלא ביצעו commit

<u>הפעולות שמתבצעות אחרי שהתנועה מבצעת התחייבות:</u>

- מוודאים שהכניסה (T, Commit) הגיעה ליומן על הדיסק. בזה מסתיים **מימוש ההתחייבות.**
  - אחרי פעולה זו מובטח שהתנועה תסתיים בהצלחה, גם במקרה של נפילה.
- מבצעים את כל הכתיבות לקבצים.
- משחררים מנעולים.

<u>הפעולות שמתבצעות ב- checkpoint ה-i:</u>

- כתוב לדיסק את כל החוצצים שהשתנו בין ה-checkpoint ה- i-2 ל-checkpoint ה- i-1 ושלא נכתבו בינתיים.
- כתוב checkpoint i לדיסק.

<u>ביצוע שיחזור (בוצעו i checkpoints):</u>

- מבטלים פעולות של תנועות שלא התחייבו ע"י מעבר אחורה מה-checkpoint ה- i-1 (כמו ב- undo).
- מוחקים מהיומן תנועות שלא התחייבו.
- עוברים על היומן **מ-i-1 checkpoint לסוף**, ומבצעים את כל פעולות הכתיבה לדיסק (כמו ב- redo).