① 23.10.06 DB

אם בא לך לדלג

בהרצאה

# מסדי נתונים (הקדמה)

לקורס יש אתר ושקפים מאוד מפורטים אם להתכונן ואם להתראות.
השקפים כ"כ מפורטים כך שיש תקנון מלא של כל מה שנאמר.
⇐ יש אנשים שמגחכים לא לבוא ולה בסדר גמור. מי שבא
בדרך הבא - רק שישמרו על שקט מופתי.
מישהו להרצאה - רבי שבראו קדם את השקפים כי אבל יהיה ברור
יותר. לא תמיד תהיה הרצאה המעוכבת כמו אבל דרך לעשות
לאתר של שנה שלברה.

המתרגל - גיל רוטשילד - מתרגל מנוסה וטוב.

## הקדמה כללית שמסבירה מה זה מסדי נתונים ולמה דרך אותם

מסד נתונים - אוסף של נתונים שמנוהל בעזרת כלי תוכנה שנקראים
(DBMS) data base management system
בכלים האלה התפתחו מצמן מרכ שהיו כלים מסיים לנהל מע'
קבצים. הלכו והרחיבו את הכלים האלה וקיבלו אשכרה כלי
אמצעי מבחינת מורכבות.
השאלה הבסיסית היא - מדוע עדיף מע' קבצים בלבד שנתנו מכירים
איך מסביבה? - כי DBMS מאפשרת לעשות כ עבור
בלבד הרבה יותר פשוטה. למשל, שיפה קלה של נתונים. הדבר
השני (ותשלם מאוד) הוא שמע' מסדי נתונים שומרת את הנתונים
בלבד יעילה כלל האפשר. זה לא מפתיע כי מבור שרשמתכנים
תוכנית מנסים לעשות אותה יעילה. אבל כ מה שמדברנו עד היום
מתבסס ⇐ זה שכל החישובים נעשים בזיכרון הפנימי. למשל
בשביל למין קוראים את כל הנתונים לזיכרון, ממיינים מאד מהר.

אבל מה אם יש יותר נתונים מזיכרון? - אז יש אשכול שנקרא זיכרון וירטואלי, וזה יכול להתאבד של בעיות בעבודה. אבל מה קורה לשליטה - ואשר לא כל הנתונים נמצאים כל הזמן בזיכרון הפנימי? זאת בעיה בשרה היתרון של מס' נתוני נתונים הוא שהיא מאפשרת להתליכ רתיבת תכנית יעילה לביות בשוק יותר.

בקורס נלמד איך להשתמש במסדי נתונים ומהם העקרונות שעל פיהם הם נבנו.

דוגמה

יש רשומות של ת"ז ושם

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| כהן | 202 | ... | אברהמי | 103 | לוי | 102 | כהן | 100 |

אומרים לרתום תכנית בבאווה שמאפשרת שאילתות לפי שני סוגים - לפי ת"ז ולפי שם (המון יכולים להיות כמה אנשים עם אותו שם).

יש כמה אינדקסים, אבל הכי אפקטיבי הוא טוב.

לשמור על חיפוש מאוזן (עץ B+tree) לפי שם ובו פוינטרים לרשומות. ומשבר לחפש לפי ת"ז אפשר לעשות חיפוש בינארי גם הרשומות.

אבל שבנו וכתבנו את כל זה נגמ אין יותר מדי נתונים הדבר הישנן שהתוכנית עושה הוא שהיא מעלה הכל לזיכרון ואז הכל טוב ויפה.

אבל אם יש המון - המון נתונים, האם התכנית עדיין יעילה ורלוונטית? חיפוש בינארי הוא בכל לא טוב אם חלק מהנתונים לא נמצא בזיכרון בפעם אחת. מה על החיפוש הוא לא הכי יעיל. מאוסן אפתיים לפעמים יכול להיות יותר יעיל לקרוא את כל הנתונים מאוסן סדרתי ולא להשתמש בעץ.

DBMS זה הדבר היחיד שעוזר לנו לעשות את הדברים האלה יותר בקלות.

הנה דוגמה ותיבה - אפשר לנהל טיסות. יש מאגר נתונים מרכזי
של החברה והרבה אנשים במקביל עושים פעולות. או כמו
ב- AMAZON שהרבה אנשים בונים בוט. למשל אם נשאר
תותב אחד במלאי ובעדכן באותו זמן שני אנשים שונים מתלוטים
לקנות אותו. המע' דרושה לדאוג שהספר לא יימכר פעמיים.
DBMS מאפשר לנו לטפל בבעיות רבות. מאחורי כל אתר מכירות
באינטרנט יש מסד נתונים. וזה מאפשר למי שכותב את התוכנית
לחשוב ראשון רק השו קיים בעולם והמע' מאפשרת לדברים
לעבוד גם רשר כמה תוכניות רצות במקביל.

מה עם בעיית שמירת הנתונים? למשל במקרה של אתר מכירות
דרוש לבחות מספר כרטיס אשראי ולשמור אותו במקום אחר.
ואת הכתובת ושם הספר דרוש לשמור במקום אחר.
אופן כוכמן פנימי לא עמיד בפני נפילות חשמל. אבל מה אם
בדיוק ברגע כתיבת הכתובת ושם הספר התשמל נופל?
אבל המע' דרושה גם לדאוג שהנתונים עמידים בפני כל סוג
של נפילה, שהנתונים לא יוכלו לאבוד. ויש דרכים לעשות את זה!

השמדברים על נתונים דרוש להבדיל בין מבנה לוגי למבנה פיסי.
המבנה הפיסי משפיע על הביצועים - אם יש אינדקס, אם
הם ממוינים וכו'. המבנה הלוגי זה הקשרים בין
הנתונים - או באופן יותר מופשט הדרך שמבטאים את
המשמעות של הנתונים. המהות של הנתונים מנקודת ראותו
של המשתמש הפשוט. אין דרך אחת ויחידה לבנות מבנה
לוגי. המבנה הלוגי משפיע על הדרך שבה המשתמש מסתכל
את הבקשות שלו לשליפת הנתונים, אבל מאידך מסויימת הוא
יותר חשוב ממבנה פיסי. שני המבנים יכולים להשתנות מבלי
להשפיע על השני.

מס' ניתוק מסדי נתונים אחת מאוחלת לנהל מספר מסדי נתונים בו"ל.
למשל באוניברסיטה יש מס' ניתוקי משכורות ומס' ניתוקי קבלות של הספריה ובאופן עקרוני. מס' הניתוק יכולה לנהל את שניהם ביחד.
עוד חשוב להבין שמסד הנתונים קיים תמיד בלי בשר לתוכנית שרצה עליו.

הסכמה הפיסית מתארת את המבנה הפיסי והסכמה הלוגית מתארת את המבנה הלוגי. היום נדבר על הסכמה הלוגית. המודל הנפוץ ביותר הוא המודל היחסי.

הסכמה הלוגית מתארת סמנטיקה שמשורה לסד הנתונים.
יש שתי רמות הפשטה - פיסית ומושגית. אבל יש עוד רמה שמעדפת אחראית על הגישה לנתונים ואז האפליקציות כ"כ נכתבות לרמה הבאה.

הסכמה הפיסית והסכמה המושגית נכתבות ע"י שפה מתאימה שנקראת DDL ואחרי המבטים נכתבים ב- SQL.
במס' מסד נתונים אם משנים משהו במבנה הפיסי לא צריך לשנות את האפליקציות שעדיין מאוד טוב. אבל ההפרדה בין רמות ההפשטה מאוד חשובה.

## - המודל היחסי בקצור -

אינטואיטיבית כל מודל פשוט וברור. הנתונים מוצגים בטבלאות (יחסים) שורה בטבלה נקראת רשומה ( record ) ושם של עמודה נקרא אטריבוט. למשל

| sid | name | year |
|---|---|---|
| 240 | white | 3 |
| ⋮ | | |

רשומה כאן בעלת 3 שדות

האטריבוטים בעזרם מייצגים את המבנה הלוגי. אפשר לשנות אותם אולי מה שמשתנה בדג'ל זה הערכים שרשומים בר שומות.

הדרך הפורמלית היא שהסכמה הלוגית היא קבוצת האטריבוטים. במבנה הקודם {sid, name, year}.

באופן מעשי מגדירים עוד כל מיני דברים, כמו הטיפוס של האטריבוט, שם ליחס, אילוצים שהיחס דרוש לקיים (למשל גיל ירול להיות רק בין 0 ל-100).

הסכמה הפיסית שלו יחס מגדירה איך היחס מאוחסן פיסית על הדיסק - B+-tree, hashing או אחרת. הבטים נוספים של המבנה הפיסי הם למשל אם יש אינדקס נוסף כמו בדוגמה עם הרשומות שלתג ושם.

## מעבר לתכנון סכמות

אם רוצים ליצור מסד נתונים חדש קודם כל צריך ליצור סכמה לוגית וסכמה פיסית. אחרי אפשר להכניס רשומות לטבלאות אפליקציות. המבנה הפיסי משפיע על יעילות ומדדים (נדבר על זה אח"כ) והסכמה הלוגית משפיעה על האפליקציות והשימושים. הסכמה הלוגית מאוד חשובה ולכן בהרבה כל פשוט לתכנן אותה. תוך-מקרה אם מוצאים בה טעות יותר קשה לתקן מאשר טעות בסכמה הפיסית. לקיום השינוי רבה לשנות הסכמה הלוגית בלבד נדרשת עלות תחזוקה ולכן לא קל כלל ועיקר.

יש גישות שונות לתכנון סכמות. אנחנו נלמד בהרצאה גישה שמפיה מתכנתים קודם סכמה לוגית במודל עשיר יותר מהמודל היחסי ואחר"כ לתרגם אותה למודל היחסי.

מה זה בעצם לתכנן סכמה לוגית? קודם כל צריך לעשות system analysis, לומר להבין מה הלקוח מבקש, להבין את מהות הנתונים. אל המודל הזה יותר קל לנו לבדוק מודל הישות והקשרים והוא קל יותר מהמודל היחסי. הוא מבדיל

בין שני דברים - ישויות וקשרים. אלה שני דברים מאוד מסוימים ברמה הלוגית. המודל אומר שהם בעולם מתחלקים לשני סוגים -

- ישויות - למשל סטודנטים, מרצים, קורסים, חוגי לימוד
- קשרים - למשל - סטודנטים ⟷ קורסים
  - קורסים ⟷ מרצים
  - קורסים ⟷ חוגי לימוד

לישות יש 3 מאפיינים מרכזיים -

- כל מישהו שקיים בעולם המציאותי ויש לו תכונות כלשהן
- יש אופן מזהה ייחודי לכל ישות לפי תכונה או תכונות מסוימות. (למשל מספר זהות של סטודנט)
- כל הישויות מאותו סוג מקבלות קבוצה אחת של ישויות.

כשמתכננים מסד נתונים בפועל הדברים לא רק חד משמעיים. למשל - האם חוגי לימוד הם תכונה של סטודנט או ישות בפני עצמה? זה תלוי באופן השימוש והמטרה של מסד הנתונים. אבל מסד נתונים צריך להתאים להקשר שלו!!!

כמובן ישנה קלות מאוד לייצג באמצעות רשומה ואז אם יש לנו הרבה ישויות מאותו סוג קל לתאר לו יחס.

תכונה או קבוצת תכונות שמזהה ישות באופן ייחודי נקראת מפתח.

בין מספר כל שהו של ישויות יתכן לראות קשר. למשל

הסטודנט "משה כהן" לוקח את הקורס "מערכות הפעלה" שמלמד המרצה "אמנון פרק". זה קשר משקשר.

והנה שני קשרים בינריים:

הסטודנט "משה כהן" לוקח את הקורס "מערכות הפעלה"

המרצה "אמנון פרק" מלמד את הקורס "מערכות הפעלה".

שני הקשרים הבינריים לא בהכרח שקולים לקשר המשולש כי ובכל לחלוטין שיש כמה קבוצות של מע' הפעלה" אבל קבוצה יש מרצה אחר.

ניתן לייצג קשר ע"י השמה. למשל מהדוגמאות בעמוד הקודם ניתן לייצג

(אשת כהן, מערכות הפעלה, אמנון ברק)
(אשת כהן, מערכות הפעלה).
(אמנון ברק, מערכות הפעלה)

סוגים שונים של קשרים

כל סוג קשר אפשר לייצג ע"י יחס. בלומר קבוצה של קשרים מאותו סוג אפשר לייצג ביחס.

כאן רק התחלה. בתרגול נתעמק בלה הרבה יותר.

הערה לגבי מינוח - המונח attribute משמש גם במודל היחסי וגם במודל הישויות והקשרים, אבל בשני המודלים זה לא בדיוק אותו הדבר.

קשר מאוד חשוב הוא קשר ה IS.A שאותו אנחנו מכירים כבר מאז שמדברנו ירושה. קשר זה מבטא הירכיה ירושה בין הישויות. במודל היחסי אין ירושה. לכן כשמשתמשים בירושה צריך לדעת איך לתרגם את זה למודל היחסי (כל כאן ועוד בתרגול)

## מדוע תכנון סכמה של מסד נתונים הוא בעיה קשה?

- מעבר מעולם לא פורמלי למודל פורמלי. גם הבעיה מוגדרת היטב מבחינה מתמטית והיחס פשוטים יותר (למשל כאלו שנראה באלג'). אבל לא תמיד זה כך. הרבה פעמים מתחילים בעולם האמשי והוא לא מוגדר ע"י מתמטיקאים. והוא מלא סתירות ופניות ונתון לפרשנויות. לכן לייצג אותו באופן פורמלי זה קשה. אין לזה פתרון אלגוריתמי פורמלי. לכן חשוב גם להבין את המגבלות של המודלים שלנו.
- המודל של סכמות. בפורמט אנחנו רואים קומבינציות מאוד מאוד פשוטות. אבל בעולם האמיתי הדברים הרבה יותר מורכבים.

- ירושה - יש דורך ב ירושה מרובה, כי לה קיים בתי הים ים. אבל זה לא אישרו שכל לתמוך בו. לאשל מגאווה בכלל אין את זה. הקושי הגדול הוא ליילה נתן דברים מאוד דומים אחד לשני, אבל בכל זאת שונים.

- ישות פרטנית לעומת ישות אחרת - לאנשי לקורס יש מאפיינים שקבועים לאורך השנים (מס' קורס, לאיזו פקולטה הוא שייך וכו') ויש מאפיינים שמשתנים שנה (מקום, זמן, הסטודנטים שרשומים)

- אילוצים - דרוש ליילה נתן אילוצים ולחלק מתהליך הבניה הוא לשלוט את כל האילוצים

- ערכים חסרים או לא ידועים -

את השאר יש לקרוא בשקפים!

30.10.06
DB

# אלגברת היחסים

תכונה של המודל היחסי: יחס כזו בעצם טבלה. השורות הן רשומות ושמות העמודות נקראים אטריבוטים. האטריבוטים הם הגבנה הקבוע של היחס ואולו הרשומות משתנות עם הזמן.
כל זמן שיש שמות לעמודות הסדר אינו חשוב ויחסים עם אותם אטריבוטים אבל עם סדר שונה נחשבים להיות. אם אין שמות, נותנים לעמודות מספרים ואז הסדר כן משפיע.
סכמה של יחס כבר אוסף האטריבוטים וכן קבוצה לא סדורה.
לפעמים גם נותנים שם לסכמה ליחס.
פורמלית, במודל היחסי יחס הוא קבוצה סופית של רשומות כך חשוב גם שכן קבוצה, כך אומר שאותה רשומה לא יכולה להופיע פעמיים (מניעה למה שבודה ב- SQL)

אלגברת היחסים היא אוסף פעולות על יחסים שהתוצאה שהן היו גם יחס => ניתן להרכיב פעולות ל

הפעולות מתחלקות באופן ל-3 קבוצות
- פעולות שמבדלים על קבוצות - איחוד, חיתוך, הפרש
- פעולות שמאפשרות לקבל חלק מהיחס - חלק מהשורות או חלק מהעמודות
- פעולות שמאפשרות לבנות רשומות מיחסים שונים.

כל הפעולות הן: הטלה, בחירה, מכפלה קרטזית, איחוד, הפרש, שינוי שם. אלה הן הפעולות הבסיסיות וההכרחיות - אי אפשר להגדיר אף אחת מהן ע"י האחרות (אבל זה ניתן להוכיח פורמלית וזה תרגיל מעניין)

הטלה - פעולה שמוחקת מספר עמודות. ממומש אותה ב-$\Pi$
והתוצאה היא יחס ללא העמודות שמוחקים.
רושמים כך: $\Pi_{\text{רשימה של אטריבוטים למחיקה}}(R)$
יש בעיה בלהה: אם מוחקים עמודה המלות להיות כפלויות בין הרשומות שנשארות. אז בתוצאה לא מוחקים את הכפלויות (ב SQL לשון מוחקים את הכפלויות. למה מעשית לא מוחקים כפלויות? - כי בשביל לעשות את זה צריך לסדר את הנתונים ומאחר שמדובר בהרבה נתונים צריך יהיה להעביר אותם מהזיכרון הפנימי לזיכרון החיצוני ולכן הלאה. כשאמרנו שמיין לוקח $O(n\log n)$ בעצם ספרנו השוואות. אבל הקריאה והכתיבה בין ה RAM לדיסק לוקחות המון זמן. קריאה מהדיסק איטית לפחות פי 10000 מהקריאה ב-RAM ומאחר שבשביל לבדוק מיון חיצוני חייבים לקרוא ולכתוב בדיסק המון פעמים זה יוצא מלתי יעיל בשלמה וזאת הסיבה שב-SQL לא מבטלים כפלויות אלא אם המשתמש - מי שנכתב את השאילתא - מבקש זאת במפורש. חוץ מזה - לפעמים רוצים להשאיר את הכפלויות - למשל אם רוצים לספור את מספר האנשים בכל קומה.)

מחיצה - זאת פעולה נוספת להטלה. בעוד שהטלה מוחקת עמודות, מחיצה מוחקת שורות. ממומש $\sigma_{\text{תנאי}}(R)$ למשל התנאי יכול להיות שם=tid ואז מתקבלות בתוצאה כל העמודות של הרשומות שבהן שם=tid.

כמובן אפשר לשלב בין שתי הפעולות, אבל זה לא חוקי: למשל אם התנאי הוא שם=tid, אם עושים $\sigma_{tid=20}(\Pi_{tname}(R))$ אז רשומות לעשות מחיצה tid לא מופיע בכלל.

התנאים של הבחירה מבוטאים ע"י שימוש באיזוי של תנאים אריתמטיים - פעולות השוואה. הן השוואות הן $<, \leq, >, \geq, =, \neq$. אפשר להשוות גם בין מחרוזות וגם בין מחרוזות לקבועים.

לפי הגדרה הטהורה אין פעולות אריתמטיות באלגברת היחסים (ב SQL לעומת זאת אפשר לרשום דברים כמו $A < B + C$).

שב
לעומק של מבנה של דיסק

* הבדל חשוב בין RAM לדיסק הוא שב- RAM בשביל לבצע הפעולה תמיד לוקחת אותו זמן לעומת השר למקום שבו נמצא המידע ב-RAM ואילו בדיסק הזמן לבצוע תלוי גם במיקום הנתונים בדיסק. לכן חשוב למקם אותם בצורה נחמדה לקריאה.

למשל אם רוצים $\sigma_{year=3}(S)$ היה נוח לארגן את הנתונים בדיסק בצורה ממוינת לפי שנה. ומה אם אח"כ נרצה $\sigma_{name='levi'}(S)$ - ברור שהמיון לפי שנים כבר לא מתאים! אבל אפשר לסדר את הנתונים בצורה נוחה לאות ד' האטריביוטים אבל זה לא בהכרח יהיה טוב לאחרים. כאן נכנס נושא המערכות. צריך לבחון מהו האטריביוט שמתפשים הכי הרבה אפשר גם לבחון את בידוקי המערכת שלנו בדיבוגים תקופתיות ולבחון את הפעולות לשנות בהתאם.

לרתכב שאותו צד ממשקל. אבל ל DBMS יש עבודה קשה- היא צריכה לבחון את השאילתא ואת המבנה של הנתונים ולבחור באלגוריתם אופטימלי לביצוע התשובה.

אבל בשביל שפעולת בחירה תהיה יעילה רצוי האפשר צריכים להתקיים שני תנאים

- שלפי הדיסק הנתונים ממוינים
- שיהיה אינדקס שאומר אפה מתחיל כל סוג נתון (בדי שלא נצטרך כל פעם לבצוע הכל מהתחלה)

מכפלה קרטזית - שילוב של כל רשומה מיחס אחד עם כל רשומה מיחס אחר. בתוצאה מספר האטריביוטים הוא סכום מספרי האטריביוטים של היחסים ואילו מספר הרשומות הוא מכפלת מספרי הרשומות ביחסים.
יש בעיה של כפילות של שמות ואז משנים את השמות לאשם ל- R.tid ו- S.tid.

המכפלה הקרטזית לא נותנת מידע מעניין - הרי אנחנו סתם מוסיפים רשומות ומאבדים לגמרי העניין. אנחנו צריכים להוסיף הגבלות ומסירות.

צת מביאו אותנו לדירוג הטבעי. מומר לוקחים רק את הרשומות שבהן יש התאמה בין האטריביוטים המשותפים וצה יצא הגיוני כי צה אומר שיש התאמה בין המידע. בשפה השני מוחקים את המידע המיותר.
וצה בעיקר הדירוג הטבעי. מסמנים אותו ב RAS.
צאר לא פעולה אלתי תלויה - מבאים אותה ע"י הפעולות המסוימות שהגדרנו.
מכפלה קרטזית בלא דירוג טבעי בפרט הן פעולות לא יעילות כי מעבר משום שהן בזבוז המון המון זיכרון (יחס רבועי לגודל הקלט). לכן אפשר עדיף קודם לעשות בחירות והטלות ורק אחר כך דירוג טבעי - כי בהתחלה נוריד את רמות המידע עוד הדירוג יתבלע של יחסים קטנים יותר.
דירוג טבעי הוא אידיאלי וקלמוטטבי ותכנות אלה אם מאפשרות לנו לעשות שאילתות.
אם מלקחים יחסים שיש להם יותר מעמודה משותפת אחת דורשים שוויון בין כל צמד עמודות ואחר מוחקים את העמודות המשותפות.

איחוד - יחסים הם קבוצות אבל אפשר לעשות איחוד.
בגלל כאן אנחנו דורשים שהסכמות יהיו זהות - כלומר
אותו מספר עמודות ואותם שמות לעמודות (אם יש שמות
בכלל) זה כאן בתוצאה אין השמות הפעלות (בניגוד ל SQL)

הפרש - גם מוגדר ליחסים עם סכמות זהות. S-T
נותן לנו את כל מה שיש ב-S ואין ב-T. באופן
טבעי לאו יתלות להיות כאן רפלציה בכלל.

---

לפעמים יש צורך להשתמש במספר עמודות. למשל, אם עושים
מרפלה ברטציה ונוצרות שתי עמודות עם אותו שם. קודם
פתרנו את זה ע"י שינוי השמות של העמודות. אבל אפשר גם
להתייחס לעמודות לפי מספרים וזה מבטל בעיות.
במציאות מסבר באופן אמורטי איך משתמשים במספרי עמודות
ואת יצירים הבדל בין קבועים מספרים למספרים שהם שמות
עמודות.

לפעמים עושים מרפלה ברטציה אלא רוצים לשנות עמודות רפלות
אבל בשביל שהסכמה של התוצאה תהיה תקינה צריך לעשות
renaming לפחות לאחת מהעמודות הרפלות
במציאות יש את התחביר המדויק שמשתמשים בו.

הצירוף הטבעי הוא הצירוף הנפוץ ביותר. אבל יש עוד שני צירופים
כלליים יותר. במציאות יש את הגרסות הפרוצדורליות שהם.
בצירוף מותנה אנחנו מגדירים את התנאי שצריך להתקיים
בשביל לקחת חלק מהרשומות. לדוגמא הצירוף הטבעי היה
בעצם $R \bowtie_{R.tid = T.tid} T$. כל תנאי כללי שאפשר לרשום
ע"י השוואות אריתמטיות והקשרים $\wedge, \vee$ אפשר לרשום שם.
כלומר עושים מרפלה ברטציה ומוחקים את השורות שמקיימות
את התנאי.

עוד כמה דברים לקחנו:

יש דבר כזה יחס ריק - זה יחס שאין לו אף השומה.
אבל יש שם יחס מלא קבוצה ריקה של אטריבוטים -
יחס ללא עמודות. במקרה כזה קיימת בדיוק השומה אחת
ביחס הזה - זו ההשומה הריקה והיא אמיתית.
אפשר להגדיר הטלה של השומה ריקה של אטריבוטים
ואז מתקבלת השומה אחת ויחידה והיא ההשומה הריקה.
זו דרך להבין אם היחס המקורי היה ריק או לא ריק.

מתמלה ברוטציות לא סכמה ריקה מקבלים קבוצה ריקה -
פשוט כי אין לה מה לחבר. אבל לתוצאה יש סכמה
והיא כל האטריבוטים של R וכל האטריבוטים של T.

אם הסכמה של T ריקה אז לתוצאה יש סכמה R.
והתוצאה עדיין תלויה ב-T. אם $T=\emptyset$ אז התוצאה
היא $\emptyset$. ואם $T=\{t\}$ אז התוצאה היא R.

אפשר להשתמש בהטלה בשביל לשנות סדר עמודות - כי
ביחס החדש העמודות מופיעות בסדר שבו הן השומות מהטלה.

פעולות מונוטוניות הפעולות הן מונוטוניות אם לא
מונוטוניות במובן הבא: נניח שיש 3 יחסים $R_1, T_1, S$
שמתאימים ביניהם ע"י הפעולות שלמדנו. פעולה מונוטונית
זה אומר שאם ל $R_1, T_1, S$ נוסיף עוד נתונים (בלי למחוק)
אז התוצאה או תיגדל או לא תשתנה, אבל בכל אופן לא תקטן.
למשל, מתמלה ברוטציות היא פעולה מונוטונית.
חיסור עמודות אינו מונוטוני. תרגיל: מה עם הפעולות האחרות?

שקילות של שאילתות - אלה ביטויים שתמיד נותנים אותה תוצאה -
כלומר לא משנה מה אוסף היחסים שנבחר לשים בביטוי, התוצאות יהיו זהות.
זה חשוב לאופטימיזציה של שאילתות.

(8) 6.11.06

# תיאוריית הכנון סכמות

למה בכלל צריך תיאוריה? הרי יש אישה שעובדת לפי השכל הישר כאדם אישה טובה ומושלמת, אבל היא מבוססת רק על אינטואיציה ולא תמיד נותנת תשובה חד משמעית. התאוריה נותנת הגדרות מדויקות ואלגוריתמים מדויקים שמאופשרים לענות באופן חד משמעי על שאלות כמו למשל - איך לדעת לייצר קשר בין מורים, בורסים ותלמידים - בקשר משולש או שניים כפולים.

השאלה הבסיסית היא מהי סכמה - הוא לו סכמה טובה?
בשביל להתלוות לא מספיק להסתכל רק על יחס נתון אחד, אלא לבחון את כל היחסים שכולל בתשבון. למשל - שרפל מידע לה משהו שאותנו רוצים למנוע וזה לא בהכרח בא לידי ביטוי בכל יחס.
למה לה משהו שרוצים למנוע?
- אם בעיה כיבוי (למרות שלה צניות היום)
- ואם בעיבור בעיה עדכון - כאשר יש המון מבומות שצריך לעדכן - כה ממש מלמין שמישהו ישכח לעדכן משהו לאט תהיה שעור בסיס הנתונים שלנו - המידע לא יהיה עקבי.
אז השבותנים סכמה להתלוות אם היא טובה או לא צריך לשאול את עדמנו אם היא מאפשרת שרפל מידע (= רע) או לא (= טוב). מה זה אומר "מאפשרת"? כמו אמרנו שביחס אחד הבעיה ורצה לבטל לידי ביטוי ביחס אחר לא. אז צריך לבדוק אם ורצה להיות יחס משהו שיגרום לבעיה, לבחון את כל היחסים האפשריים לבור הסכמה ולפי זה להתלוות אם ורצה להיות יחס שבו מידע משרפל וכו בעיה.
אבל מהם כל היחסים האפשריים? מה בכלל אותם? קודם כל יחס חייב להתאים לסכמה (דה!) אבל מאוד חשוב הוא שיחס חייב להתאים לאילוצים של הסכמה!

את האלוגים צריך למדוא בתהליך תכנון הסכמה ולא מספיק רק לזהות את הלמידים. האלוגים הם אלה שקבלו מספרו של דבר, לא רק אלו יחסים אפשריים, אלא אם יש בכלל יחס לדיווח שרפאל מידע.

לקרוניות אפשר להביא בחשבון הרבה אלוגים שונים, אבל אנחנו נדבר רק על תלויות פונקציונליות שקיימות ונדבר עוד מעט.

דוגמה: מורים (T) – תלמידים (S) – קורסים (C)

לכל קורס יש רק מורה אחד אבל הרבה תלמידים.
הסכמה {S, C, T} אינה טובה כי יש שכפול מידע של המורים. למשל

| S | C | T |
|---|---|---|
| Levy | DB | SMITH |
| Levy | CS | BING |
| Cohen | DB | SMITH |

מה הפתרון? אפשר לפרק לשני יחסים:

| S | C |
|---|---|
| Levy | DB |
| Levy | CS |
| Cohen | DB |

| C | T |
|---|---|
| DB | SMITH |
| CS | BING |

הבעיה תמיד נובעת לקחת סכמה אחת ולפרק אותה לקטנות יותר? לא בהכרח! רשמנו לפרק צריך לדרוש לא רק שהיחסים התקבלו לא יאפשרו שכפול מידע, אלא גם שניתן לשחזר מהם את המידע המקורי.

בשביל לקבל יחסים קטנים מיחסים גדולים משתמשים בהטלה מרווח ההפצה אם לוקחים פירוק ועושים שזו מכפלה טבעית – הבעיה תמיד מקבלים את היחס המקורי? – במקרה הזה כן,

אבל זה לא תמיד נכון. במקרה הזה, זה עובד רק בגלל שכל קורס יכול להיות רק מורה אחד.

ובאמת קל לראות שאם את S, C, T לא היינו מפרקים ל- S, C ו- S, T אתה יכול טבעי היינו מקבלים השמות נוספות שגויות ולא תהיה לנו שום דרך לדעת אילו השמות נכונות ואילו לא. חוץ מזה לא מתקיים האילוץ שכל קורס יש מורה יחיד. אז הפירוק דרך אם אפשר לנו לוודאן שכל האילוצים כן מתקיימים.

בפירוק S, C ו- S, T הוה אפשר לפצוח באופן עובאלי את קיום האילוץ - פשוט מסתכלים על S, T ורואים שכל קורס יש רק מורה אחד.

לסיכום, הדרישות מפירוק הן

- שלא יאפשר שרשרת מידע
- שיאפשר שחזור של המידע המקורי
- שיאפשר בדיקה קלה של קיום אילוצים.

## תלויות פונקציונליות

איך אילוצים אנחנו מכירים?

- אילוץ השתתפות
- אילוץ מפתח
- אילוץ מפתח זר
- ועוד

אנחנו לא יכולים לדון בכל האילוצים וזה לא מסובך. אנחנו נדבר על אילוץ שנקרא <u>תלות פונקציונלית</u>. למשל "לחלקה יש רק מנהל אחד" נקרא תלות פונקציונלית ומסומן $H \rightarrow D$.

כאומר D קובע את H. - אם שתי השורות שוות ב- D אז הן שוות גם ב- H. אבל זה לא אילוץ מפתח כי

המתיקה לא מהברת קובעת את הסטוקטים.
באופן מעניין, ראשי ושרמה תפ שמתבימים יש עוד תפ שמתבימים.
למשל אם $D \to S$ ו $D \to H$ אז אסוציטיבית של יחס השוויון נבע זה ש- $S \to H$.
אבל אם $D \to S$ ולא $D \to H$ זה לא מהברת אומר שלא נשון ש- $S \to H$.

נסכם, בתהליך תכנון סכמה אנחנו קובעים את האטריבוטים ואת האילוצים- את התלויות הפונקציונליות בין האטריבוטים.
אולם אפשר להתחיל לבדוק אם יש אפשרות לשרשל מידע-
אם יש אטריבוט D שקובע פונקציונלית תלק מהאטריבוטים האחרים אז כדאי לעשות.
בשביל להחליט אם יש שרשרת מידע לא מספיק להסתכל על התלויות שבינו במפורש, אלא יש גם כל אלה שנובעות מהן.
רק לדעת שיש קבוצה של אטריבוטים קובעת אטריבוט אחר, ואז מסמנים $SC \to G$. לא מהברת אם $G \to SC$ אז $S \to G$ או $C \to G$. מה שחשוב הוא שלה ה ברור של S ו- C.

לרשום נפרד את כל הקבוצות והאינטואיציה שבבסיס.

סימונים-
- $X, Y, Z, \ldots$ - קבוצות של אטריבוטים
- $A, B, C, \ldots$ - אטריבוטים.

תמיד בצורה הכללית ביותר היא מהצורה $X \to Y$ - קבוצה של אטריבוטים קובעת קבוצה אחרת של אטריבוטים.
קצת פחות כללי זה תלות מהצורה $X \to A$ - קבוצה של אטריבוטים קובעת אטריבוט בודד.

בשביל לסמל שוויונים ואי שוויונים בין רשומות משתמשים בסימון $r[X]=t[X]$ - הרשומות $r$ ו- $t$ שוות על כל האטריביוטים ב- $X$. אם $r[X]\neq t[X]$ זה אומר שקיים אטריביוט ב- $X$ לא מקיים שוויון.

אז מה זה אומר שיחס מקיים תלות פונקציונלית?
יחס $r$ מעל הסכמה של $R$ מקיים תפ $X\to Y$ אם לכל שתי רשומות $s,t$ ב- $r$ מתקיים

$$s[X]=t[X] \Rightarrow s[Y]=t[Y]$$

(כך הגדרה זו סימטרית ואינטואיטיבית אבל התשובות היו אם שותרמל לתרגיל שורתלר מסומן לה).
השקולה היא: $r$ מעל $R$ לא מקיים תפ $X\to Y$ אם קיימות שתי רשומות $s,t$ ב- $r$ כך ש- $s[X]=t[X]$ אבל $s[Y]\neq t[Y]$.

התכונות הפונקציונליות קובעות בין היתר מהם היחסים האפשריים.

יש תכונות פונקציונליות טריוויאליות - שמתקיימות בכל יחס. למשל $A\to A$ או $AB\to B$. ובאופן כללי $X\to Y$ כאשר $Y\subseteq X$ זה תמיד מתקיים. ~~אבל בכל הנושאים~~
טענה: תפ $X\to Y$ היא טריוויאלית אמ"מ $Y\subseteq X$
הוכחה: במבחן.

אירוניה - למשל אם $A\to B$ ו- $B\to C$ אז זה אומר ש- $A\to C$.
מסמנים $A\to B, B\to C \models A\to C$.
קבוצת $F$ של תפ אמורות תפ $X\to Y$ ומסמנים $F\models X\to Y$ אם כל יחס $r$ שמקיים את $F$ מקיים גם את $X\to Y$.

אבל בודקים אם יש אירוניה נפתח אלגוריתם לינארי שמגלה אם יש אירוניה או אין אירוניה!

(11) 13.11.06
DB

אותו מדברים על תכונות פונקציונליות. אם תון יתם קיים לבדוק את כל ההשמות בטבלה כדי לדעת אם תכ"מ. אבל בדיונים תכ"מ יש גם תכ"מ שנובעות מאנד ואז אותן לא קיים לבדוק.

אם F הבולה של תכ"מ ו- R הבולה של אטריבוטים אז מן הסתם מעניינים אותנו רק היחסים שמבוימים את F. כל אחד מהם יכול להיות בנפרד.

אמרנו שיש תכ"מ טריוויאלי - זו תכ"מ לכל יחס:
$Y \to X$ טריוויאלי אם $Y \subseteq X$. למשל:

$$AB \to A$$
$$AB \to B$$
$$AB \to AB$$

הכיוון $Y \subseteq X \Rightarrow X \to Y$ טריוויאלי הוא די ברור.
הכיוון השני, לעומת זאת, הוא פחות ברור:
$X \to Y$ לכל יחס $\Rightarrow Y \subseteq X$
אם (לכל יחס $X \to Y$) $\neg \Rightarrow \neg(Y \subseteq X)$.
אז נניח ש- $Y \not\subseteq X$, ונראה ש- $X \to Y$ לא מתקיים בכל יחס.
צריך להראות שיש לנו הגדרות של X ואי שוויון של הגדרות של Y.
אבל $Y \not\subseteq X \Rightarrow$ קיים אטריביוט $A \in Y \setminus X$.

| X | | A | שאר עמודות |
|---|---|---|---|
| 1 | 1 | 1 | לא משנה |
| 1 | 1 | 0 | |

(11)

בדיוק הבולה תכ"מ F יעניין אותנו כל התכ"מ שנובעות מ-F.
למשל אם $F=\{A\to B, B\to C\}$ אז כל מה שמקיים את F
יקיים גם את $A \to C$. לעומת זאת, לא בהכרח יתקיים $C \to A$.
אם כל היחסים שמבוימים את F מבוימים גם את $A \to C$
מסמנים $F \models A \to C$. אבל $F \not\models C \to A$. אין
מרחיב שאין יכולה - מבוים דוגמה נגדית!

לדוגמא

| A | B | C |
|---|---|---|
| 1 | 1 | 1 |
| 0 | 0 | 1 |

מתקיים $A \to B$ וגם $B \to C$. וכמובן $A \to C$.
אבל לא נשון ש- $C \to A$. וזאת פונקציה נכונה טובה.

איך בודקים אם יש אפשרות לראות יש לבדוק את כל הדרכים האפשריות בשביל לבדוק אם יש פונקציה נכונה. אבל יש אלגוריתם נוסף אך זה לא מתמטי.
אך אפשר לבדוק בשיטות לוגיות.

האקסיומות של ארמסטרונג

- רפלקסיביות $X \supseteq Y \Rightarrow X \to Y$ (תלויות טריוויאליות)
- אוגמנטציה $X \to Y \Rightarrow XZ \to YZ$
- טרנזיטיביות $X \to Y \wedge Y \to Z \Rightarrow X \to Z$

בעזרת האקסיומות אפשר לגזור תכ"ס נוסף מתוך F.
לדוגמא: $F = \{A \to C, B \to D\}$ ונגזור את $AB \to ABCD$

| | | |
|---|---|---|
| 1 | $A \to C$ | נתון |
| 2 | $AB \to ABC$ | אוגמנטציה עם AB |
| 3 | $B \to D$ | נתון |
| 4 | $ABC \to ABCD$ | אוגמנטציה עם ABC |
| 5 | $AB \to ABCD$ | טרנזיטיביות של 2 ו-4 |

זה טוב - זה נותן לנו דרך לבדוק אם סגירה כלשהי של תכ"ס היא נכונה - פשוט צריך לבדוק את כל האקסיומות - אם ניתן ואיך להחיל אותן. אך אלו זה לא יעיל במיוחד, אבל כדי לדעת F וסגירה של תכ"ס ניתן לבדוק אם הסגירה היא נכונה של F או לא. מסמנים אותה $F \vdash X \to Y$

עכשיו יש לנו שני מושגים - גזירה ואמיתות ומן הסתם נרצה להראות ששני המושגים האלה הם שקולים.

$F \vdash X \to Y \iff F \models X \to Y$.

<u>נאותות של גזירות</u>: $F \vdash X \to Y \Rightarrow F \models X \to Y$.

<u>הגדרה</u>: בעזרת דרכים להראות רק את הנאותות של האקסיומות ו אז באינדוקציה זה יהיה נכון לכל גזירה של גזירות.

- רפלקסיביות - זה משהו שנכון בכל שנדרשת תכונת קדמות. אבל יש להראות שזה מתבים לכל יחס ואת זה כבר ראינו - רפלקטיביות נותנת לנו תכונה טריוויאלית.

- אינטנסיביות - דרך להראות ש- $XZ \to YZ \models X \to Y$.
אז נניח ש- $Z$ מבים את $X \to Y$. ונראה ש- $Z$ מבים גם את $XZ \to YZ$ אבל זה ברור. נתון שהם יש שוויון של $X$ אבל יש שוויון של $Y$. עכשיו, נניח שיש שוויון גם של $X$ וגם של $Z$ אבל יש שוויון של $Y$ (בגלל הנחה) וגם של $Z$ - טוב פשוט כי $Z=Z$. בגלל זה רתבה באופן מתמטי פורמלי.

- טרנזיטיביות - הנאותות פחות נובעת מהטרנזיטיביות של יחס השוויון.

(:)

רפלקסיביות, אינטנסיביות וטרנזיטיביות הן אקסיומות בלתי תלויות - כלומר, אף אחת מהן לא נובעת מה אחרות.
אבל יש עוד אקסיומות שנובעות מהן ולהן נאותות כי הן גזירה מאקסיומות של ארמסטרונג

- איחוד: $X \to YZ \Leftarrow X \to Y \wedge X \to Z$
- פירוק: $X \to Z \Leftarrow X \to Y \wedge X \to YZ$
- פסאודו-טרנזיטיביות: $XW \to Z \Leftarrow YW \to Z \wedge X \to Y$

מהאיחוד נובע משהו חשוב. נניח יש קבוצה של אטריביוטים $X$ שנובעת מתמנה כל מיני קבוצות. $X \to Y_1$; ..., $X \to Y_n$.
אז קיימת קבוצה מקסימלית אחת ויחידה של אטריביוטים שנובעת מ- $X$: $X \to Y_1 \dots Y_n$.

סגור של קבוצת אטריביוטים X - קבוצת כל התכ"
שנגזרות מ-F. מסמנים כאן ע"י $X_F^+$.
למשל אם $F = \{A \to B, B \to C\}$ אז

$$A^+ = ABC$$
$$B^+ = BC$$
$$C^+ = C$$
$$(AC)^+ = ABC$$

טענה: $(AB)^+ \supseteq A^+ \cup B^+$

הוכחה: הכלה היא ברורה, אבל לא בהכרח יש שוויון. למשל $F = \{AB \to C\}$ אז $A^+ = A$
$B^+ = B$

אז $ABC = (AB)^+ \neq A^+ \cup B^+ = AB$

בגלל הרפלקטיביות ברור שתמיד $X \subseteq X^+$

שלמות של האקסיומות: $F \vdash X \to Y \Leftarrow F \models X \to Y$

נניח שיש תכ $X \to Y$ שאינה נגזרת מ-F. נראה שגם ורלה לא היה מקיים אי"ש שנבנה דוגמה נגדית.

| $X$ | $X^+$ | שאר |
|---|---|---|
| 1 1 | 1 1 | 1 1 1 |
| 1 1 | 1 1 | 0 0 0 |

בגלל שמשתמשים ב"רק" הפרים לי, ההוכחה לוקה מתסר.
מתקיים יש הכל מתפרק.

(:)

אלגוריתם יכול לבדיקה אם $F \vdash X \to Y$ -
אם נמצא דרך יכולה לחשב את $X^+$ גם פשוט לבדוק
אם $Y \subseteq X^+$ וכן הלאה. בשביל לעשות את זה פשוט נסבור
מ F וגוסף באופן חמדני כל אטריביוט שנמצא
מהקבוצה שאספנו עד כה.
בשביל להוכיח את נכונות האלגוריתם צריך להוכיח שני
דברים:
- נאותות - להראות שנוספים לסאור רק אטריביוטים שאומנם
בסאור לפי ההגדרה - אבל לא ברור
- שלמות - להראות שנוספים לסאור כל האטריביוטים
שאומרים להיות בסאור לפי ההגדרה. - מאוד דומה להוכחה
של השלמות של האקסיומות

אם $V \subseteq W$ אז ברור ש- $V^+ \subseteq W^+$.

## מפתח ומפתח על

מפתח של - X מפתח על של הסכמה R שיש לה קב' תל
F אם $F \vdash X \to R$.
מפתח הוא מינימלי מבחינת הכלה - אם $F \vdash X \to R$ ואם אין
$Z \subsetneq X$ כך ש- $F \vdash Z \to R$ אז X מפתח.
אבל מפתח הוא מפתח על אבל לא כל מפתח על הוא מפתח.

איך בודקים אם X הוא מפתח? מורידים כל פעם אטריביוט
אחד ובודקים אם זה עדיין.

(14) 20.11.06
DB

אותנו מעניינים של תלויות פונקציונליות. $X \to Y$ - X קובע את Y פירושו שאם ביחס יש שורות על X אז יש שווין גם ב-Y. סכמה אקימות תקפה אם לא קיים יחס שמהווה דוגמה נגדית - כלומר לא קיים יחס שבו שתי שורות שוות ב-X אבל שונות ב-Y.

קיבלנו של אכירה של תלויות. אומרים ש- $F \vdash X \to Y$ לה אומר שאפשר לקחת את התלויות ב-F ולהפעיל עליהן את האקסיומות של ארמסטרונג וזה שמתקבל $X \to Y$.

לעומת זאת אומרים ש- $F \models X \to Y$, כלומר F גורר את $X \to Y$ אם לא קיימת דוגמה נגדית שבה מתקיימים התנאים ב-F אבל לא מתקיים $X \to Y$. למשל אם $F = \{A \to B, B \to C\}$ אז $F \vdash A \to C$ אבל $F \not\models C \to A$. והנה דוגמה נגדית:

| A | B | C |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 1 |

רשים לנו סכמה R וקבוצה של תנאים F. אנו מעניינים אותנו רק ביחסים שמוגדרים על R ומקיימים את F. מתוך כל אלה בכל רגע נתון, מאוד קיים רק יחס אחד, אבל לה משהו דינאמי שמשתנה, אבל בכל רגע נתון היחס צריך לקיים את הכל. אבל הינו רוצים לדעת מהן כל התנאים שנגזרות מ-F ואז כל היחסים צריכים לקיים את כל התנאים האלה.

ראינו שיכירה כזה משהו ששקלנו לאכירה. אמנם לה טובים לפני שהייתה לנו אכירה, רצו להפרק קיים תנאים הינו צריכים לבחון את כל היחסים ת אם שרים בשבלם לאוגו דוגמה נגדית וזה לא מעניין. כי מושג האכירה אפשר לנו לפתח אלגוריתם שמוצא את כל התנאים הנגזרים מ-F.

הגדרנו סגור של קבוצת אטריביוטים:

$$X_F^+ = \{A : F \vdash X \to A\}$$

כלומר קבוצת כל האטריביוטים - הקבוצה המקסימלית של אטריביוטים - שנקבעת ע"י X.

בשביל לחשב סגור של X עוברים על כל הקשתות ב-F ומוסיפים אם אפשר להוסיף את הצד הימני שלה - לה אם שרוי אם הצד השמאלי כבר קיים.

דוגמה: $F = \{A \to C,\ B \to D,\ DE \to H,\ DK \to G\}$

$(ABK)^+_F = ?$

1) ABK
2) ABKC $(A \to C)$
3) ABKCD $(B \to D)$
4) ABKCDG $(DK \to G)$

כלומר ערשיו סיימנו - אי אפשר עוד להוסיף כלום.

ערשיו אפשר לתת הגדרה מדויקת של מפתח ומפתח על.

מפתח על של R הוא קבוצת אטריביוטים $X \subseteq R$ כך ש- $F \models X \to R$.

נאמר X מפתח על אם קיים יחס שיש בו 2 שורות ששוות על X אבל אין שווה על איזה אטריביוט A.

לדוגמה

| X | A |
|---|---|
| 0 | 0 |
| 0 | 1 |

כאן X הוא לא מפתח על.

בגלל זה אפשר מגדירים אותו רק בעזרת היחסים שמקיימים את F. ואז X אינו מפתח על אם קיימת דוגמה נגדית לכך ש- X גורר את R.

עכשיו הינו רוצים שזה להיות מספיק אם הסתכל על כל היחסים האפשריים (כי לזה הרבה סבירות) אבל הינו רוצים לדעת איפיון למפתח על שמשתמשים בתכונה.

$A$ הוא מפתח אם $F \models A \to R$. אבל אם $A^+_F \subseteq R$ אבל $A$ הוא מפתח על.

דוגמה: $R = ABCD$ $F = \{A \to B, B \to D, C \to D\}$

אבל $A^+_F = ABD$ אבל $A$ אינו מפתח על.

לעומת זאת $(AC)^+_F = ABCD$ אבל $AC$ דווקא כן מפתח.

- מפתח - מפתח הוא מפתח על אשר מינימלי מבחינת הכלה. כלומר אם תת קבוצה ממש שלו אינה מפתח על.

למשל בדוגמה הקודמת $AC$ הינו מפתח כי לא $A$ ולא $C$ אינם מפתחות על.

הרעיון לחפש תמיד יש מפתח? בוודאי - $R$ הוא תמיד מפתח על של $R$. אבל אפילו אם לא נדע לחזיר מינימלי וקטנים ש-$R$ הוא מפתח.

אם בודקים אם $X$ הוא מפתח של סכמה $R$ עם קבוצת תלות $F$ לרוורה פרק לבדוק את כל תתי הקבוצות ממש, אפילו זה לא יעיל מאחר כי יש $2^n - 1$ רשימה. למעשה מספיק לבדוק רק קבוצות מאפס ל-$n$ (ורשימה יש בדיוק $n$) כי אם הוריד אטריבוט אחד ולא נמצא, בוודאי שאם נוריד יותר אטריבוטים אז זה לא "גם מפתח על". שזה אם $X \subseteq Y$ אבל $X^+_F \subseteq Y^+_F$.

---

סכמה שמאפשרת שרשרת מידע הוא סכמה בעייתית משתי סיבות:

- כפילות
- (ומעבר) זה יכול להביא לשאילתות שמאן עדכון היחסים.

אפשר להתייחס לבעיה הבאה מבחינת מבט של תלויות פונקציונליות
למשל אם יש סטודנטים וראשי מחלקות והטבלה היא
SDH - סטודנטים, קורסים וראשי מחלקות.
אם ידוע להיות רק ראש מחלקה אחד אז $D \to H$.
ואם בפעם שמופיע סטודנט ומישהו לה מופיע מספר פעמים
מופיע אותו ראש המחלקה שלו.
אז מבחינה הבעיה היא התנאי $D \to H$.
באופן כללי, סכמה היא בעייתית אם קיימת תנאי $X \to Y$ שהיא
לא טריוויאלית (כלומר $Y \not\subseteq X$) וגם ש-
- $F \models X \to Y$
- $X$ איננו מפתח על (כלומר $X_F^+ \not\subseteq R$)

במקרה שלנו, לסכמה SDH שתי תנאים: $S \to D$, $D \to H$
אבל $S \to D$, $D \to H \not\models D \to S$
לכן $D$ אינו מפתח על של הסכמה $\Leftarrow$ הסכמה היא
בעייתית.

בהמשך נראה דרכים טובה בשביל שיש לנו אלגוריתמים
פולינומיאליים לבדיקת כל הדברים האלה.

---

סיור של קבוצת תנאים מוגדר באופן מאוד טבעי ע"י:
$F^+ = \{X \to Y : F \models X \to Y\}$
כלומר כל התנאים שאפשר לגזור מ-F.
$F^+$ הוא מאוד גדול מאוד כי רק תלויות טריוויאליות יש בו n!
וראשי n הגודל של R. אז להלן מדיאגרמה לחלוא
את כל $F^+$.
בשקפים הוסבר איך אפשר לייצג את הכלל ה"יותר מעניין" של
$F^+$ - דברים שנובעים מטרנזיטיביות.

שקילות של תאוריות פורמליות -

מעבר לשאמרים ששני דברים הם שקולים אם הם שווים, אפשר גם לא הגדרה מעניינת כי אם שתי קבוצות תו"פ שקולות אזי הן שוות אבל לא אמרנו שום דבר חדש.

מן הסתם היינו רוצים להגיד שקבוצות תו"פ שקולות אם אחת תו"פ נגזרת מכל אחת מהן. או במילים אחרות, $F$ ו- $G$ שקולות אם $G^+ = F^+$.

להגדיר אם $G^+ = F^+$ עובדת מן אקסיומטיזציה כי יכול להיות כמה מספר אקסיומטיזציה של אמרים.

אפשר אפשר לנסח אותו באלגוריתם:

- בהינתן $F$ ו- $G$ נוכל לבדוק אם $G^+ = F^+$ ע"י:
  - לכל $X \to Y \in F$ בודקים אם $G \vdash X \to Y$
  - לכל $X \to Y \in G$ בודקים אם $F \vdash X \to Y$

אם $F \vdash G$ וגם $G \vdash F$ אז $G^+ = F^+$.

כפי שהוכחנו שכך אלגוריתם נכון נסתכל מה לה אומר שאין שקולות.

$F^+ \neq G^+$ אם מתקיים $F \vdash X \to Y$ אבל $G \nvdash X \to Y$

או $G \vdash X \to Y$ אבל $F \nvdash X \to Y$

אבל אלה מקרים סימטריים אז נטפל רק במקרה הראשון.

אז נראה שאם קיים $X \to Y$ כך ש- $F \vdash X \to Y$ ו- $G \nvdash X \to Y$

אז יש לפחות תלות אחת ב- $F$ שאינה נגזרת מ- $G$:

ניתן בשיטה שאין ב- $F$ אחת כזאת. אז $G \vdash F$ - כלומר מ- $G$ אפשר לגזור כל תלות שמופיעה ב- $F$. אבל $G \vdash F^+$. לכן $G \vdash X \to Y$ אבל זאת סתירה!

לראות $G \vdash X \to Y$?

- בונים גזירה מ- $F$ ל- $X \to Y$
- בונים גזירה מ- $G$ ל תלות הראשונה שהשתמשנו בה בסעיף הקודם
- משרשרים את ההרכבות.

דוגמה:

$F = \{A \to B,\ B \to C,\ A \to C\}$

$G = \{A \to B,\ B \to C\}$

המשמעות לבדיקת שקילות מספיק לבדוק אם $G \vdash A \to C$ (כי כל השאר הן מבדיהות), אבל ברור שלא כך.

לכן $F^+ = G^+$. אבל בעדפה ב-F יש תלות מיותרת - כלומר אפשר למחוק את $A \to C$ ועדיין יש שקילות.

דוגמה:

$F = \{A \to B,\ AB \to C\}$

$G = \{A \to B,\ A \to C\}$

ב-G יש לנו תלות יותר חלשה - $A \to C$ כי תמיד אפשר להוסיף לה דברים אבל עדיין אימפליקציה. אבל $AB \to C$ חזק יותר. ולכן לא טריוויאלי ש- $F \vdash A \to C$, כל השאר ברור.

אבל ראה שלא כך:

$A^+_F = ABC$

$\Leftarrow$ $F \vdash A \to C$. אבל ב-F יש משהו מיותר - האטריביוט B בחלק $AB \to C$.

אם נתבונן בדוגמאות אלו נראה שיש דברים שיתלים להיות מיותרים ואז

- אם מוחקים חלק מ-F, אז F יהפוך לקטן
- אם מוחקים אטריביוט מחלק ב-F אז F יהפוך לגדול

נגדיר:

- חלק של F הנה מיותר אם מחיקתה אינה משנה את $F^+$
- אטריביוט בדף שמאל של חלק של F הוא מיותר אם מחיקתו אינה משנה את F.

בהינתן F - קבוצה חלק, ניסוי חסר כפילות הוא G שאין בה לא חלק מיותר ולא אטריביוטים מיותרים אבל $F^+ = G^+$.

איך מוכיחים כוסי חסר כפילויות?

- עוברים על כל התת־קבוצות ב־F ובודקים אם אפשר להוריד אחת ולדרוש הסגור לא ושתנה. פשוט מורידים את כל הדברים המיותרים מ־F

באלגברה יש בעיה להישאר.

מחיקה של אטרבוטים יכולה להכניס תלויות חדשות למחיקה שלא היו קודם.

אז עוברים על כל האטרבוטים בלבד שמאל ובודקים אם הם ניתנים למחיקה ומוחקים אם אפשר.

אחרי עוברים על כל התלויות ובודקים אם הן ניתנות למחיקה.

הסדר הזה לא יוכל הכניס תלויות חדשות למחיקה שלא היו מלכתחילה. הסבר פורמלי לזה יש באלגברה.

שמאל אינטואיטיבית, מחיקה של אטרבוטים מגדילה את הסגור ואילו מחיקה של תלויות מקטינה. אבל רגע להיות שפני שמחקנו אטרבוטים F לא היה משפיע תלוי בשמאל להראות שניתן למחוק תלויות.

(18) 27.11.06
DB

# תיאוריית תכנון סכמות למסדי נתונים

נניח, לה הלוקח שאותנו מדברים עליו

עכשיו שיש לנו R ו- F אותנו יכולים לעשות כל מיני דברים מעניינים.
למשל, הייתה לנו מטרה לבדוק אחרי סכמה היא בעייתית - כלומר שהיא
מאפשרת שרפלס מידע, כלומרבחלק מהיחסים שמעניינים אותנו
יתלו להיות שרפלס מידע.

הפדיגמה הקלאסית - סטודנטים S, מחלקות D ורושי מחלקות H.
ונתונה התלות $D \to H$ אבל לא $D \to S$.

הנה יחס עם שרפול מידע

| S | D | H |
|---|---|---|
| Levy | CS | Rubin |
| Cohen | Math | Bush |
| Barak | CS | Rubin |

אבל לא חייב להיות שרפול! למשל ביחס הריק אל פעם אין שרפול מידע.
ואם היינו מחקים את השורה השישית, עדיין לא היה שרפול.
בכל מקרה נתון יש באסף יחס אחד בפיק, אבל הוא משתנה עם
הזמן בעקבות שינויים.

מה הבעיה בפדיגמה? יש תלות $D \to H$ אבל D לא מגדיר את
כל שאר האטריבוטים

באופן פורמלי, יש בעיה אם יש תכ"מ שנגזרת מ- F:
$F \vdash X \to Y$ אבל X לא מפתח של R.

הוכחה נגזרת של פתרון הבעיה -
הפיתרון הוא בעיקרו פשוט. אם יש סכמה בעייתית, נפרק אותה לשתי סכמות קטנות יותר שמכילות רק חלק מהאטריביוטים והתקווה היא שבסכמות הקטנות לא יהיה שרשור מידע ואם נתן יהיה לשחזר מהן את כל המידע המקורי.

<u>דוגמה</u>: סטודנטים S, קורסים C ומורים T
כאשר $C \to T$ אבל לא $S \to C$
לאחר סכמה בעייתית כי C אינו מפתח שלם.
נפרק את SCT לשתי סכמות:
SC - לה יתרם רבים רבים אל $F = \emptyset$
CT - יתרם רבים לאחד! $F = C \to T$
ברור ששתי סכמות אלה אינן מאפשרות שרשור מידע.
אבל הבעיה לה מספיק? - לא! יש רבדוק שאותנו מסוגלים לקבל את המידע המקורי שלה בדיוק הדרוש הטבעי.
במציאות יש דוגמה שבה הפירוק הלה בעומת מאפשר שתצור ל המידע, כלומר היחס המקורי מתהבל מדרוש טבעי של היחסים הקטנים יותר. הבעיה לה מספיק? - לא! יש רבדוק שלה נשין לכל יחס אפשרי!
ואכן במציאות יש לוד דוגמה שבה ומתי הדרוש הטבעי נושבות שורות שלא היו בבדבה. אם לה נחשב אובדן מידע כי אותנו לא יודעים אילו מן הרשומות היו במקור ואילו לא.

טוב, ברור שבלתי אפשרי לבדוק את כל היחסים האפשריים כי לה יבת כמן אינסופי.
אבל המטרה שלנו היא למצוא אלגוריתם יעיל לבדיקה אם פירוק הוא טוב.

<u>הפרדה ללא אבידה</u> -

- בפירוק אין שכפול מידע
- הפירוק מאפשר שחזור מידע ואין אובדן מידע
- יש דרך פשוטה לבדוק שהתכונות הפונקציונליות והמקוריות מתקיימות בעזרת הפירוק.

<u>הגדרה של פירוק</u> - תהי סכמה $R$ עם קבוצת תכ"פ $F$.
הסכמות $R_1, \dots, R_n$ יהוו פירוק של $R$ אם $R_i \subseteq R$
! - $\bigcup_{i=1}^{n} R_i = R$ .

בגלל ההנחות של היחס המקורי על הפירוק צריך להיות שווה ליחס המקורי - כלומר הרשומות של פירוק ללא אובדן.
באופן פורמלי, לכל יחס $r$ עבור הסכמה $R$ שמקיים את כל התכ"פ של $F$ צריך להתקיים $r = \pi_{R_i}(r) \bowtie_{i=1}^{n}$ .
ההגדרה הזו מוגדרת בהינתן $R$, $F$ ופירוק $R_1, \dots, R_n$.
אבל השבוע להראות שהתכונה איננה מתקיימת צריך לא לבדוק דוגמה נגדית - כלומר $r$ ספציפי ש עבורו אין שוויון.
אבל אתנו בודים אלגוריתם שבודק אם יכולה להיות דוגמה נגדית.

נשים ✓ שאם היחס החדש לא שווה ליחס המקורי אז חייב להיות שהיחס החדש <u>מכיל</u> את היחס המקורי. וזה ברור כי בפירוק מופיעים כל האטריבוטים ובשבוע היחס החדש אנחנו עושים פירוק טבעי, אבל כל הרשומות המקוריות תמיד יתקבלו.
הבעיה היא שיכולים לקבל רשומות שלא היו בהתחלה - כלומר קורה כשבפירוק הטבעי מתאפשר לפרק רשומות קטנות שלא היו במקור מאותה הרשומה.

תמיד מתקיים $r \subseteq \pi_{R_i}(r) \bowtie_{i=1}^{n}$ !

האלגוריתם שלנו ינסה לבנות פונקציה נאמנה. אם היא תצליח - יופי, אם לא, זה אומר שאין פונקציה נאמנה מלא אובדן מידע.
הרעיון הוא לנסות לקחת הטלות שהגיעו מרשומות שונות של היחס המקורי ולראות אם אפשר לחבר אותן יחד.

למשל, הסכמה SCT עם $F=\{C\to T\}$
נמצאים אם SC, CT הטלות אובדן?
ננסה לבנות פונקציה נאמנה:

| S | C | T |
|---|---|---|
| $\underline{a_1}$ | $\underline{a_2}$ | $b_{1,3}$ |
| $b_{2,1}$ | $\underline{a_2}$ | $\underline{a_3}$ |

כל פונקציה נאמנה בהכרחה הטלות ביחד כי אנחנו צריכים שרשרת של הבדיקה של שוויון על C נקבל שאפשר לקבל רשומות שבאו ממקורות שונים.
אבל!! צריך להתקיים $C\to T$ ואם $b_{1,3}\neq a_3$ אז לא מתקיים F. ⇐ אין בכירה וחייב להיות

| S | C | T |
|---|---|---|
| $\underline{a_1}$ | $\underline{a_2}$ | $a_3$ |
| $b_{2,1}$ | $\underline{a_2}$ | $\underline{a_3}$ |

עכשיו, האם כל עדיין פונקציה נאמנה? לא! כי הרשומה "חדשה" $a_1\ a_2\ b_{1,3}$ הגיעה מעלים $a_1\ a_2\ a_3$ אבל כל לא בעיה.
המסקנה היא שהפירוק הוא טוב: התחלנו מהבכירה הטלה ביחד של פונקציה נאמנה, תיקנו אותה לפי התנאים שאמורות להתקיים ומצאנו שאז ירדנו להיות שכל פונקציה נאמנה!

## האלגוריתם הכללי:

נתונה סכמה $R = A_1 \ldots A_k$

ופירוק $R_1, \ldots, R_n$

בדוגמה הנ"ל היו לנו ביתר יהיו n שורות - שורה לכל סכמה בפירוק, k-! עמודות - אחת לכל אטריביוט ב-R.

במקום ה- $i,j$ בטבלה כזו יש:

- $a_i$ אם $A_i$ מופיע ב- $R_j$
- $b_{ij}$ אחרת

עכשיו מתחילים להחליף את היחס ואם בסוף יגלו שאחת הרשומות היא $a_1 \ldots a_k$ אז לא יכולה להיות דו"מה נ"ל.

אם אין שורה כזאת אז מגיעים לדו"מה נ"ל

(~ דו"מה מפורטת במצגת).

הוכחה פורמלית של האלגוריתם יש בספר של Ulman.

עכשיו נתבונן בפירוק רק שהמס התחדש מקיים את התלויות המקוריות. חשוב להבין שגם שהפירוק הזה ללא אבדן, זה לא אומר שבהכרח התלויות מקיים את כל מה שבדרוש.

הרי כל מה שעשינו קודם היה מבוסס על כך שקחנו חס מקורי, פרקנו אותו ובדקנו אם הפירוק מתאים למה שהיה קודם.

אבל במסד נתונים, באופן מנשי, אחרי שמפרקים חס הוא לא קיים יותר, קיים רק הפירוק ולכל חס קטן ניתן להוסיף דברים. אז בעצם כך יכול שאנחנו שומרים מידע חלקי - אם נדרש את היחסים הקטנים יכולים להיווצר ערכי NULL, שגורם בעיה כי המשתמש צריך להוסיף מידע לכך. אז פירוק מאפשר לנו לשמור מידע חלקי ללא שימוש ב- NULL. אבל כל אופן צריך לבדוק שגם אפשרנו בפירוק להבדקים רשומות שלא מקיימות את התלויות המקוריות.

צריך לבדוק אולי תלויות מתקיימות בפירוק. ויש לבדוק אם הן מקיימות יתקיים את כל $F^+$. אבל לא בעיה כי משמרים לתשם את $F^+$

פרוק לא תלוי (כי $F^+$ מאוד אקספוננציאלי).

הגדרה <u>הטלה של תלות</u>: $S \subseteq R$ תת־קבוצה.

הטלת התלות שלתת־קבוצה $S$ = $\Pi_S(F)$

$= \{X \to Y : X \to Y \in F^+ \wedge XY \subseteq S\}$

נתבונן למידע התלוי - לא משנה אם מפרקים את $S \subset T$, $F = \{C \to T\}$ ל- $SC$ ו- $ST$ אבל אחד מהם אין שום תלויות אם כן נשנה רק אותם אין שום דבר שמבטיח $C \to T$.
ואילו אם מפרקים ל- $CT$ ו- $SC$ אז היינו להתקיים $C \to T$ כי זה מתקיים ב- $CT$ ובגלל הכלל רשומה האטריביוטים $C$ ו- $T$ מופיעים מהימין של הסכמה $CT$ ושם מתקיים $C \to T$.
$\Leftarrow$ זה מתקיים אם קבוע.

הגדרת <u>פירוק שמשמר תלויות</u>: $R$ סכמה עם קב' תל' $F$ ו- $R_1, \dots, R_n$ פירוק של $R$, אז $\{R_i\}$ משמר את התל' אם

$$\Big(\underbrace{\bigcup_{i=1}^{n} \Pi_{R_i}(F)}_{\text{ס' התל' שם ס' הסכמות שהפירוק}}\Big)^+ = F^+$$

זה אומר שכל תלות מ- $F$ מתקיימת בפירוק ולהפך.

אותנו מדברים על סכמות בעיתיות - סכמות שמאפשרות שכפול מידע בתוך מתיחסים האפשריים. לה בדומה הרשר יש הבדלה של אטרבוטים X שקובעת תוך מהאטרבוטים אבל לא את כולם, כלומר X לא מפתח על.

סכמה R עם הבדלה הזו בצורה BCNF אם לכל $X \to A \in F^+$ מתקיים $X \to A$ טריוויאלית או X מפתח על.

ברור שסכמה כזאת הינו לא בעיתית כי כו בדיוק השלולה של הגדרת סכמה בעיתית.

נזכור מה המטרה שלנו: בהינתן סכמה R לחלק פרוק $R_1, \dots, R_n$ כך ש-

- הפירוק חסר אובדן
- הפירוק משמר את התלויות
- כל סכמה $R_i$ הינה בצורה נורמלית

כל זמן שמתקיימים הדברים הבאים:

- הפירוק הוא ללא אובדן
- מתיחסים באמצע הם הטלות של יחס אחד מעל כל האטרבוטים שמביים את התכ"ם

כפי לא מבטיח לי בכלל של כל התכ"ם. כי בכמה שיש פירוק אל כל יחס הטלה יכול להיות עדכן באופן בלתי תלוי באחרים ואז רשנעשה פירוש עדיין יתקיימו כל התכ"ם המבורות.

$\Leftarrow$ חייבים לדרוש גם שהפירוק ישמר את התכ"ם. שזה אומר

$$(\cup \pi_{R_i}(F))^+ = F^+$$

גם אם הפירוק לא משמר תכ"ם, תמיד מתקיים $(\cup \pi_{R_i}(F))^+ \subseteq F^+$ כי לא יכולות להיווצר לנו יש מאין תלויות פונקציונליות חדשות.

נתפש אלגוריתם לפירוק

- מתחילים עם סכמה R ובקבוצת תל"פ F.
- קודם מתפשים הפרה של BCNF, נאומר $X \to A \in F^+$ הן ש- $A \notin X$ אם $X \to R \ \forall F$. כי אם אין הפרה אז אין שום בעיה ונסיימנו.

ניתן להוריד שאם יש הפרה ב-$F^+$ אז יש הפרה ב-F. לחפש הפרה ב-F אפשר בזמן פולינומיאלי. אז למציאו הפרה של BCNF אפשר בזמן פולינומיאלי. (אם יש כזאת).

- אפשר לפרק לסכמה שכוללת את האטריביוטים של ה"ת המפרה. וסכמה שכוללת את כל האטריביוטים של הסכמה המקורית פחות מהאטריביוט בצד ימין של התלות המפרה.
- מחשבים באופן רקורסיבי לכל אחת מהסכמות הקטנות. בכל אחת מהסכמות יהיה שוב $X \to A$ הא"ל הפרה של BCNF לכן יש פחות הפרות של BCNF מאשר בסכמה המקורית.

האלגוריתם הזה נותן לנו פירוק שהוא בטוח בלוסי הנורמלי. אבל הוא לא מבטיח שימור של ה"ת. אמנם נראה שה סכמות יכם תמיד בלוסי שאין אובדן מידע. כזה בלי הלמה הבאה: פירוק של R לשתי סכמות $R_1$ ו $R_2$ הוא ללא אובדן אמ"מ $F \models R_1 \cap R_2 \to R_1$ או $F \models R_1 \cap R_2 \to R_2$. (הוכחת הלמה בתרגיל).

אפשר לראות שאלגוריתם שלנו כזה תמיד מתקיים. לכן האלגוריתם מוביל פירוק ללא אובדן.

הוא תמיד אפשר לחזור פירוק שמקיים את כל מה שראינו רוצים לעשות לחשוב הסכמה ABC זם $F=\{AB \to C, C \to A\}$. $C \to A$ הפרה כי $C \not\to B$. ⇐ חייבים לפרק. אבל רוצים שימור תלויות. יש 3 אופציות לפירוק - לבחור מתוך AB, BC, AC. ב- AB, BC לא באות לידי ביטוי שום תלויות. ב- AC יש $C \to A$ אבל לא לזה משמר את התלויות. אז אם היינו לוקחים ABC

היה שימור תלויות אבל לא צורה נורמלית. ואם מתקבלים
על צורה נורמלית אז אין שימור תלויות.

הבעיה כאן נמצאת מצה כשכשפירקנו לקחנו את המפתח
AB של הטבלה המקורית. והשמפקים את המפתח התלות
שהוא היה ביניהם השאלה שלה הלכה לאיבוד

צורה נורמלית שלישית 3NF - * סכמה היא בצורת 3NF
אם לכל $X \to A \in F^+$ מתקיים
- $X \to A$ טריוויאלית, X מפתח על, או האטריביוט A שייך
למפתח כלשהו של R.

כל הבעיה של BCNF ובשמלה יש אלגוריתם פשוט לפירוק.
מתבוננים עם הסכמה R וקבוצת התנאים F. מוצאים בסוף חסר
רפלוקציה של F ומכל תלות שם יוצרים סכמה. וגם בכל
תלות שהפכה לסכמה, צריך השאלה הוא מפתח אם לא רק מפתח
על (כי אין רפלוקציה) בגלל שלא מגיים שימור תלויות כי
הפירוק התבסס על התנאי. אבל צריך שיהיה שאחת מהסכמות
הינו מפתח על ואם אין אחת מהן אינה מפתח אז צריך להוסיף.
במצב יש הוכחה שלה באמת מפרק לצורת 3NF...

(23) 11.12.06
DB

# חלק חדש בקורס: אופטימיזציה

במסדי נתונים יש רמות גדולה של נתונים. יותר מדי גדולה מדי שנוכל להכניס את כולה בבת אחת ל RAM לכן צריכים למצוא שיטה (וזה לא טריוויאלי איך) שבה קוראים כל פעם רק חלק מהנתונים, מבצעים חישוב רק לאותו חלק מהתשובה, כותבים לדיסק חלק של התוצאה וחוזר חלילה.
כי הגודל שבה עובדים עם כמות גדולה של נתונים.

כמן החישוב של שאילתא מעיקר מושפע מהקריאה והכתיבה לדיסק כי זה לוקח לפחות פי $10^3$ מהפעולות שמתבצעות בזיכרון הפנימי. אבל בחישובים שלנו אותנו ענין שפעולות ב CPU על נתונים בזיכרון הפנימי הן כניחות ולספור רק את מספר פעולות הקריאה והכתיבה לדיסק.
חוץ מזה ה CPU יודע לעבוד במקביל לקריאה מהדיסק. כולאו מקבילות טהורה. הרבה פעמים ה CPU צריך לחכות בכל זאת אבל זה עוד זמן שמאצר את השפעת הפעולות בזיכרון הפנימי.

גורמים עיקריים שמשפיעים על הזמן הנדרש ל I/O
- אכיון הנתונים של הדיסק. הזיכרון הפנימי RAM רשמו בן הוא אין בעיה לגשת לכל חלק שרוצים. הוא עובד מיחידות של מילים, ענין 32 ביט ואם רוצים לקרוא 4 מילים זה לוקח פי 4 זמן מאשר מילה אחת.
דיסק, לעומת זאת, פונה מיחידות של בלוקים (או דפים) שהגודל הטיפוסי שלהם הוא 1K-4K שלכן כמו הרבה יותר מ- RAM. חוץ מזה אי אפשר כבר סתם לגשת לכל מקום ואם רוצים לקרוא 4 בלוקים

שנמצאים במקומות שונים בדיסק לא יכול לקחת יותר
מפי 4 זמן מאשר לקרוא בלוק אחד.
אם הנתונים נתונים ברצף אחד אז אפשר לקרוא
אותם מהר יותר (לכן רצוי לעשות איחוי דיסק)
- השיטה שבה מבצעים כל אחת מהפעולות - ישל
אינו שיטות לביצוע צירוף
- הסדר שבו מבצעים את הפעולות.

דוגמה רוצים לצרף $R(A,B) \bowtie S(B,C) \bowtie T(C,D)$
צירוף טבעי הוא אסוציאטיבי וקומוטטיבי אז יש הרבה
אופציות לחשב את זה.
- $(R(A,B) \bowtie T(C,D)) \bowtie S(B,C)$
זה לא כפוי כי ב-R ו-T אין אטריבוטים
משותפים. ושום דבר לא יוצא בצירוף הטבעי.
סה"כ אנחנו שואפים שבסוף תהיה
הרי קטנה שאפשר.

האופטימיזציה של שאילתות זה חשוב מאוד. לא יכול להיות
הבדל בין חישוב של שעות לחישוב של שניות. אוכ
אנחנו לא מדברת. נחפש את הפיתרון הכי טוב אבל בשום
פנים ואופן לא נרצה להגיע לתסריט שבו טוקה לנו שעות
תוך מצה נניח. שזמן הקריאה/כתיבה של בלוק בודד הוא קבוע.

לא יחס שמור בקובץ והקבצ מנוי מבלוקים. בכל בלוק יכולות
להיות הרבה רשומות. פעולה על רשומה דורשת קריאת בלוק
ומרול אותה לזיכרון הפנימי, מילוי הפעולה וכתיבה חזרה -
סה"כ 2 פעולות קריאה כתיבה.
רשומות יכולות להיות באינדקס קבוע או משתנה. בכל דיסק לחשוב
על שיטת אחסון. שמאפשרת מציאת רשומה מתוך בלוק ונ"ל
מקום פנוי שנותר אחרי מחיקת רשומה.

רשומה מזוהה ע"י RID שמורכב ממספר בלתי וממספר רשומה בתוכן הבלוק. בסוף הבלוק יש מערך שאומר איזה מתחילה כל רשומה שנמצאת בבלוק, מערך של פוינטרים רשומות קיים רשומה שמים בתא הנתאים במערך ALL אחרי שנקבע שכן רשומה שרבר לא קיימת רבר. אנחנו לא נרצה לתת את אותו RID לרשומה חדשה כי יכול להיות שתהיה איפשהו פנייה לרשומה הראת ואת והיה בלבול.

בשביל אשות פעולות על בלוק צריך לקרוא אות מהדיסק ל RAM. אבל ב RAM יש מאגר של חוצצים שאליהם נקראים הבלוקים והם נשארים שם לעוד זמן. אבל אם נאמר הנקום צריך לפנות. אבל לפני שמוחקים מה RAM צריך לרשום את זה חלרה לדיסק. בשביל זה לוברים אם הבלוק הזה שונה או לא, כי אם הוא לא שונה אז חבל לבזבז זמן ברתיבה. אפשר להשתמש במערכת הפעלה לניהול מאגר החוצצים אבל מס' ניהול מסדי נתונים מעדיפות לעשות את זה לבד כי יותר קל להשיג תאימות למספר מע' הפעלה שונות ויש גם חיבטים שלא קיימים במע' הפעלה ומלוחים טיפול מיוחד (תיבנות שרלות ממקבילי, קריאה מקדימה של מלבים נכון).

קריאה מקדימה - הרבה פעמים נשקראנו חלק מהקובץ הגדול ה מעל רבר יכול להתחיל לעבוד את המידע אבל אפשר להמשיך לקרוא למאגרי החוצצים בלי לחכות שה מעל יסיים. זה לא תמיד אפשרי אבל בתיאוריה זה טוב מאוד.

יש 3 סוגים של קבצים לאחסון יחסים.

- קובץ ערמה - אין שום מיון. הרשומות מאוכסנות בסדר שבו הן הוכנסו.
- קובץ ממוין - מאורגן לפי סדר כלשהו שקבענו (לפי איזה שדה מסוים או שרשרת שדות ברשומות)
- קובץ ערבול - כל בלוק הוא דלי בפני עצמו ואנחנו עושים hash על איזשהו שדה של הרשומות. התוצאה שפונקציית הערבול היא הבלוק שבו הרשומה נמצאת. ובלשון הרי צריך לקרוא את כל הבלוק אבל זה לא משנה שהוא לא נתנו את המיקום בתוך הבלוק.

בקובץ ערמה מוסיפים רשומות בסוף הקובץ והכל טוב ויפה.
בקובץ ממוין מנסים שאין מקומות ריקים אבל כשמוחקים צריך לכווץ את הרשומות למעלה וגם מוסיפים רשומה לאמצע צריך לה צאת את הרשומות כדי לפנות מקום.
בקובץ ערבול אותנו מעניין שאין גלישה. מתכננים כך שלכל דלת יהיה אסף תפוסה ואז תמיד אפשר להוסיף והכל טוב.

במציאות יש בעיה שמפרקת כשהזמן לוקח לעשות פעולות שונות סוגי הקבצים.

## שב"ל

- לסיים לקרוא את המצגת
- לקרוא באוקטרנט איך עובד דיסק

(25) 18.12.06
ג'בי

בשיעור הקודם פגשנו שני סוגי קבצים שונים.
היום נכיר סוג אינדקסים. אינדקס מעל קובץ נבנה לפי מפתח חיפוש - למשל ת"ז, שם, מדינה וכו'.

הנתונים יושבים באופן קבוע על הדיסק.
אבל מה האינדקס צריך להיות קבוע. כי לבנות אינדקס לוקח הרבה זמן (כי צריך לבנות את כל הנתונים). אז כדי לא לבנות שוב ושוב תבנית אינדקס בכל רגע שהיא מתחולה ותמחק אותו השתמשים.
האינדקס צריך להיות זמין תמיד לכל אפליקציה כדי שלא נצטרך לבקרו את כל הנתונים. אבל אם מתפשרים לפי שפה שאין עבורו אינדקס אז אין ברירה ונצטרך לבקרא הכל.
מה אותנו רוצים שהאינדקס יכיל?
- רצוי שיהיה שמור באופן רציף על הדיסק
- סביר שעושים עץ, אבל רוצים שהרמות יהיו שמנים מאוד, כדי שבכל בלוק יהיה כמה שיותר מידע. כי בלאו הכי חייבים לקרוא בלוק שלם.
דוגמה נפוצה למבנה נתונים כזה הוא עץ $B^+$:

העץ הזה הוא יפה עץ מאוזן, אבל הבעיה שלו הוא תמיד נמוך לגמרי. בעץ כזה כל המידע עדיין נמצא ברמה התחתונה של העץ והיא בלינקד מה יפה מאוד אבל לרוב מסתכלים כבר שהרמה התחתונה תהיה שמורה באופן רציף על הדיסק. אבל כל שאומרים נתונים הרבה פעמים יכולה להתקבל בשלבי בקרוא צריך לארגן

מתרגש על הדיסק. אם אוספים מלא רשומות ואז מורקים את
ומן יכול להיווצר מצב רע שלה לא טוב. אבל אנחנו
מניחים שלה לא קורה כי באמצע נתונים בדר"כ מוספים נתונים
הרבה יותר משמוחקים נתונים.

אין אפשר לעשות אינדקסים לפי כמה שדות? במחרוזת מפתח
תפוש אחד שלפי מאיזה הרשומות האמיתיות. לה האינדקס הראשי.
ובצדי החפוש האחרים, מצביעים שלהם יהיו רק פוינטרים
לרשומות האמיתיות - נתוני כניסה. נתוני כניסה הם מהצורה
(k, rid) ואשר k ערך ו- rid מצביע לרשומה שיש היתה
עם הערך k. ה rid עצמו מורכב משני חלקים - מספר
בלוק ומספר רשומה בתוך הבלוק (לה טוב כי לה מאפשר לארגן
מחדש את הרשומות בתוך הבלוק באופן נוח).
כמה בריאות אם ערך משבש לקבוצה 1000 רשומות שמפוזרות על
600 בלוקים? לה תלוי באיזה החוצץ. אם יש בו מקום
לבלוק אחד והרשומות כל פעם מגיעות מבלוק אחר אז כל
פעם צריך לחזור לקרוא מחדש. אבל אם יש 600
מקומות אז אנחנו נקרא את כל הבלוקים בפעם אחת, כלומר
600 בריאות ויותר לא נצטרך.
עכשיו היתה יכולה להיות אינדקס מקובץ אחד במסד נתונים כי פיסיות
הרשומות יכולות להיות מאוחסנות רק לפי סדר אחד. בכל אופן
אבל כך משפיע רק כשהתיקייה היא רשומות רבות ולא כשהחיפוש
הוא על ערך שהוא מפתח. לכן הגיוני למיין פיסית לא
לפי מפתח אלא לפי משהו שחלק לתחומים - למשל לפי
שם משפחה ולא לפי ת"ז.
כשעושים חיפוש מורכב שכולל כמה שדות כדאי למיין את השדות
במיון לבסיקלי לפי, וגם אפשר לעשות חפוש יעיל בטווח של חלק
מהשדות.

הגורמים שמשפיעים על ההחלטה אם להשתמש
באינהקס או לקרוא באופן סדרתי
- אם האינהקס מקובל
- הערכת יעדל התוצאה
- גודל החוג.

למשל אם מערכים שיעדל התוצאה הינו 90% מהמידע
שבים אז עדיף לעבור פשוט על הקובל כי בלאו הכי
צריך לקרוא רמעט הכל. אבל אם ידוע שהתוצאה בטוח
עדיף להגיע אליה באופן ממוקד דרך אינהקס.

הטלה כזו משהו שמאוד קל לעשות באופן
אבדוני, אלא אם כן אנחנו רוצים לבטל
כפילויות. משבים לעשות את כה
צריך לבדוק מין. משבים לבדוק איזן
תיזוני צריך כמות מינימלית לא צמומה
של מקום בתוכל, וצריכלה להיות בעיה.

הודעה חשובה:

השיעור עבר באופן
קבוע למתמטיקה ב

(27) 25.12.06
DB
יום הזכרון לשמעון אבן (לוי) שרון

אנחנו מדברים על חישוב שאילתות.
צריכים לראות כמה הפעולה היקרה ביותר.

פשוטה

```
select *
from reserves R, sailors S
where r.sid = s.sid
```

מהם האלגוריתמים שמשפיעים על זמן החישוב?

- לפי איזה אטריביוט מחפשים
- אם כן מאמין
- אם יש אינדקס מקומי } ארגון הנתונים על הדיסק
- כמה גדול ה- Buffer Pool. כל שהוא יותר גדול אפשר לבדוק יותר בבת אחת לצורכי הפנימי ואז החישוב מהיר יותר.
- גודל תוצאות הביניים ובפרט גודל התוצאה הסופית.

בשביל לחשב צירוף של שני חסים צריך לסרוק ב דיסק ב חוצץ - בל ל- R, בל ל- S ובל לתוצאות ביניים (בהנחה שהוא מתמלא כותבים אותו לדיסק ומתחילים למלא מחדש).
במציאות יש אינטליגנציה שמראה שפי צריכים לחשב את הצבר הנכון - אחת יעולה ואחת לא יעולה.

השיטה הראשונית - nested loops

```
foreach tuple r of R
    foreach tuple s of S
        if r.sid = s.sid
            add (r,s) to result
```

הנוסחה הכי טובה עכשיו

$$M + p_R \cdot M \cdot N$$

$M$ ← בלוקים ב-R

$p_R$ ← רשומות בבלוק של R

$N$ ← בלוקים ב-S

כי יכולת אמשיך אותך לא אשתמש בכה.
(מה שמשתמשים בו כה Block-Nested Loops)
בשיטה השניה שבה לא מתלבטים את S כל הזמן אלא רק לכל בלוק של R כי יבחר פי $p_R$ פחות קריאות. שכי עדיין לא אשהו.

איך אפשר לשפר? אם החוצץ מספיק שגדל כך שאפשר לקרוא את כל R בבת אחת מתקבל הזמן $B_R + B_S + outputsize$

$B_R$ ← בלוקים ב-R

$B_S$ ← בלוקים ב-S

בשביל לדעת מה יוצא חייבים לה עריך את גודל התוצאה.
אותנו דבר לא כל מינו שיטות לחשב את הערכה.
כל אחת מהשיטות האלה הגודל של התוצאה בסוף כהה אפשר להתקדם ממנו. אבל בעדם הזמן הוא $B_R + B_S$.

מה קורה אם יש B בלוקים ב מאגר החוצצים אבל כי לא מספיק לכל אחד מהיתרים $B < B_R$ , $B < B_S$.
נניח $B_S > B_R$. ניתן ל- $B_R$ B-2 בלוקים והשאר בלוק ל-S. ובלוק לתוצאה. ואז הזמן יוצא

$$B_R + B_S \left\lceil \frac{B_R}{B-2} \right\rceil$$

אם $B_R \le B-2$ אז $\left\lceil \frac{B_R}{B-2} \right\rceil = 1$ ומתקבל זמן לינארי ב- $B_R$, $B_S$. אחרת הזמן הוא ריבועי בהם. מהנסתה אם ברור למה עדיף לתת לחלק הקטן יותר את $B-2$ הבלוקים.
במדגם יש פסאודו-קוד של האלגוריתם הזה.

אז השבאים לחשב שאילתא אתנו רוצים להיעזר מלעשות את החישוב הכי מהרה. לזה לא נורא אם החישוב לוקח דקה או שעתיים כשהשרת בחלבי דקה. אבל אם החישוב לוקח עשרות שעות זה כבר רציני.

מה היתרון של השיטה שראינו עד כה?
- היא לא מניחה שום מבנה של הנתונים ואין צורך במבנה נתונים נוסף
- היא צריכה רק את המינימום ההכרחי של בלוקים במאגר התוך זכרון

מה החסרונות?
- אם יש מעט בלוקים במאגר אז הזמן גדל באופן ריבועי
- אם יש הרבה מאוד מקום בתוך אז הדבר לא יועדף לממש את אלגו הפוטנציאלי, למשל וכל להיות שעשר להכניס יותר מבלוק אחד של S בכל פעם.

index nested loops join - רעיון שיש אינדקס של S לפי אחד האטריבוטים שלפיהם מבצעים את הצירוף. רעיון שיש 4 בלוקים במאגר התוך זכרון.
אז יש בלוק ל-R, בלוק ל-S, בלוק לאינדקס ובלוק לתוצאה. ואז את הצירוף מחשבים בעזרת האינדקס - כדי לצמצם את הרשומות המתאימות.

הנוסחה עבור המחיר ~~[illegible]~~ שיצא הינו

$B_R + t_R X$

$t_R$ – מספר הרשומות ב-R

$X$ – זמן לאיתור הרשומות המתאימות של עבור רשומה ב-X

$X$ תלוי בדברים אחרים – עומק האינדקס
– אם הוא מקובץ או לא
– מספר הרשומות המתאימות
– גודל ה Buffer

Hash Join – יש פונקציה שמעבירה אטריבוט לבלוק. ואז השרותים לחשב דיסק יודעים בדיוק מה הבלוק שצריך לחפש בו. אך בשלב הראשון צריך לקרוא את כל היחסים ושמים כל רשומה בבלוק שהיא שייכת לו לפי פונקציית הערבול. בשביל זה מספיק כל אחד באופן שאלו קראים כל אחד מהיחס. חוץ מזה נשארים B-1 דפים שאליהם הולך הכל המקורי לפי פונקציית הערבול ומשם הם עוברים לדיסק רשלריק. ואז נוצרים partitions בדיסק.
אז לכל אחד משני היחסים קוראים אותו פעם אחת ומחלקים אותו פעם אחת.
בשלב הבא מבצעים את הדיחול עצמו. לכל ערך של פונקציית הערבול קוראים את הבלוקים של R שמתאימים לערך זה (בשלמותם) ואז אפשר לקחת בלוק בלוק מ-S, לבדוק ל-R ולהחזיר ל output.
בשביל לייעל את החישוב יש מתב הכיבוי הפנימי, סאיים כפנים מתלבים לפי פונקציית ערבול פנימית ומתשבים בעצרתה.

אם יש סה"כ B מקומות במאגר הרוצדים
אז יש B-1 מקומות לצלמים שונים.
⇐ הגודל של כל Partition הוא $\frac{B_R}{B-1}$
אם מתשיבים את התישוב רואים שצריך להיות $B > \sqrt{B_R}$
ואם זה כך אפשר להשתמש בשיטה הזאת שהיא
יעילה מאוד.

אך לרשיון קיבוץ של תישוב פעולות בודדות. לרשיון נמור
לשאילתות שלמות.
* מה הסיבוכיות של שאילתא? זה תלוי באלגוריתם
היתוסים והאלגוריתם השאילתא.
התוצאה וכלה לצאת באלגוריתם מסדרי אלכן לא
קיים אלגוריתם פולינומיאלי שמחשב את זה.
למשל אם $R = \boxed{0 \mid 1}$ ומחשבים את $R^n$
אז האלגוריתם הוא $n^2$. אם, אז יש לנו באן פלאו
קטן (זה ביטוט). אין אלגוריתם פולינומיאלי
ומכל צורות מסדרי נתונים רבים כל הזמן בכל מקום
ומובדים היטב ומהר.
למה? רוב השאילתות המעשיות הן קטנות
ולכן זה מאפשר לנו להניח שהוא קבוע ולחשב
סיבוכיות רק לפי גודל הנתונים.

## ש"ב

לגבור של הנמצאות שהיו אמורות
להיות היום כי אנחנו בשיעור ונמבור
עליהן מהר.

(30) 1/1/07
בס"ד
שנה טובה!

אותנו מדברים על אופטימיזציה של שאילתות.
דיברנו בשיעור הקודם על 4 שיטות לחשב צירוף.
שפיים מהן לא הסתמכו שיש איזה מסדר של טבלאות
בתוך אבל הן עבדו בזמן ריבועי. לעומתן השפיים
האחרות הן צריכות יותר טבלאות בתוך אבל עבדו
בזמן לינארי.

איך לוקחים שאילתא שלמה ומוצאים את הדרך הכי
יעילה לחשב אותה?
לשם כך יש מנוע סטנדרטי של optimizer שמותר
לא בהכרח את הדרך הכי יעילה אבל דרך שהיא
לפחות לא איטית.
קודם כל משתמשים (rewriter) את השאילתא המקורית.
זה מכונת ה SQL. למשל אם אפשר הופכים
שאילתות מקוננות לשאילתות.
אח"כ ה-planner הופך את השאילתא לרכבה
ביטויים אלגבריים שקולים (algebraic space).
לכל אחד מהביטויים האלה צריך לבדוק את המחיר
המינימלי שלו (מהתאם לביטוי עדמו לאופציות התישוב
שמומשו במערכת).

* כל הדברים האלה מתבצעים בצורה הפנימית. לכן, אופלו שהם לא יעילים בכלל, עדיף לנו לחשב לבד את מה באופטימיזציה מחסום והבהקפעלות בדיוק לתבה לפיסם

רק לדעת מה המחיר של ביטוי צריך
לדעת להעריך את הגודל של תוצאות
הביניים.
המרחב האלגברי יכול להיות גדול מאוד
ואותנו רוצים לבדוק אותו, רק שנצטרך
לעשות פחות חישובים.
נבחל את עדמנו רק לשאילתות ההצירה
select ... from ... where

שאלתות אלה אפשר לבדוק בכמה דרכים שונות
במסדר ובמספר הפעולות.
כדי שנוכל לדמות לאמת כל האפשר את תנאי ההחירה,
נוח להפעיל אותם מהר כל האפשר כי אז פעולות
הדיגיטל (שהן הבקרות) יהיו על חמש בקנה יותר אלכן יבחו
פחות זמן.
אם הטלות מבטיחות את אופן הוחסים, לכן אם הטלות, כמה
שניתן, כדאי לעשות כמה שיותר מוקדם.
בידוק בחירה והטלה מוקדם כל האפשר לא מיבר את זמן
הריצה לכן זה דבר שתמיד כדאי לעשות.
הסיבה שזה לא מיבר היא שעושים אותן on the fly,
כלומר מדפיסים אותן לפעולות אחרות. קריאה/כתיבה
לדיסק אחרון עושים בכל תוך כדי פעולת דיגיטל
ובזמן שעושים את זה אל אם תוך כדי עושים בחירה
והטלה. לכן זה בחינם.
אל כבה מדמדמים את המרחב האלגברי. רשום
רק דיגיט אחד לה די פשוט.
הענינים מסתבכים קצת כשיש שני דיגיטים כי אפשר
לחשב אותם במסדר שונה. בכלל אבחנה כדאי לפעמל
מראש אופציה שיש בה ממש מרפלה קרטזית (לא לא
שום תנאי) כי אל הגבל של התוצאה ממש גדול.
חל מצה נפסול אופציות שמכריחות אותנו לכתוב תוצאות ביניים
לדיסק כי זה לוקח הרבה זמן ואל אחד לא מבטיח לנו
שיש מקום לתוצאות הביניים בליכאן הפנימי.
איך בכלל אפשר לעשות חישוב בלי לכתוב שום תוצאות ביניים
לדיסק? נניח שרוצים דיגיט של 3 וחמישים. אז בתוכל
מבקשים מלוח לכל וחמש ומלוח לתוצאה. ואז פשוט מחשבים
ע"י 3 לולאות אקוונות. שבל להבל מאות לא פותר אותנו
מבעיה של איזה דיגיט גדול לעשות קודם.

רפי להתשב אז יתר נשון למצמן את האופציה הכי טובה משתמשים בתכנון דינמי (בדומה למה שעשינו באלגו עם נפל מטריצות)

אבל, צריך להתחשב בעוד דברים. למשל, יכול להיות שאופציה בדרך פחות מלאה נותנת תוצאה ממוצעת של אטרקציות אחר וגם כה מוציל את שאר הפעולות. לכן לכל תת הבעיה צריך לשמור גם את המחיר הכולל ביותר ומס מחירים כלליים ביותר שנותנים תוצאות ממוצעות.

נעב: לעבור במצגת של שעורכם אבל תוצאות ביניים.

# ניהול תנועות

איך תוכניות פונות של מסד נתונים?

- קריאת נתונים מהמסד
- מבצעת חישובים בהיכון הפנימי המוקצה לה
- כותבת נתונים חדשים למסד

איך תוכניות רבות במקביל של המסד? רשכתבים אפליקציה מנתים כאולו רק האפליקציה הנוכחית בזמן בעולם וה- dbms אחראי של כה שתוכניות יכלו לעבוד נשון במקביל. תוך מכה המתכנת מניח שאו שהתוכניות מתבצעות בזמן אמת או שרצים לא מתבצעות בכלל

דוגמה תוכנית להעברת מקום בטסה.

נניח שבמסד יש פרים A שאומר כמה מקומות פנויים יש בטוס. ברור איך התכנית אמורה לעבוד. היא קוראת את A, בודקת אם יש מקום או אין, אם יש,

היא מבטיחה את A, אחרת מודיעה שאין מקום פנוי.
פעולות קריאה וכתיבה מתבצעות במסך הנתונים. כל השאר
נעשה בכיבוי הפרטי של התוכנית. מה שבכיבוי הפרטי,
רמבן לא יכול להתנגש בשום דבר. לעומת זאת בקריאה
ובכתיבה וכלולות להיות התנגשויות עם אפשר בלי אחרות.
אם יש רמה תוכנית אפשר לעשות אותן ברמה דברים
- באופן סדרתי - כל תוכנית מתבצעת בשלמות אחת
אחרי השנייה ובכך אין התנגשויות. הסדר השונה יכול
לגרום לכך שסדרת הפעולות בכל מקרה יהיה שונה.
- בידוע מקבילי - אותנו מעניינים שה מעבד יכול לבצע
פעולה אחת בלבד בכל יחידת זמן ואז פעולות
מתוכניות שונות מתבצעות לסירוגין.
המערכת צריכה להחליט אם בידוע מקבילי הוא נכון או לא.
מה זה בידוע מקבילי נכון? - בידוע מקבילי הוא נכון אם הוא
נותן אותה תוצאה כמו בידוע סדרתי. אותנו מעניינים שבידוע
סדרתי הוא תמיד נכון.
אותנו עושים אמסטרדם ביה - אסטרטגיה רק על פעולות קריאה
וכתיבה. בשביל להבין אם האמסטרדם ביה הוא נכונה
אם לכל חישוב שיכול להיעשות בין פעולות הקריאה והכתיבה
ה תוצאה שקולה לאיזשהו בידוע סדרתי.

<u>תנועה</u> - סדרה של פעולות קריאה וכתיבה שמתבצעות על
פריטים (רשומה, יחס, וכו') מהמסד צריכים לה לה.
לשם הפשטה נניח ש תנועה לא קוראת פריט יותר מפעם
אחת ולא כותבת לפריט יותר מפעם אחת (בשאר הכי
כל החישובים מתבצעים בכיבוי הפנימי שלה).
<u>תזמון</u> - בידוע של מספר פעולות במקבילי. כל נקודת זמן
רק פעולה אחת יכולה להיות מבוצעת ב מעבד.

(22) 8/1/07
בוקר

אותנו מדברים על תנועות. מה שמעניין אותנו הוא לדעת אם בהינתן בידוע מקבלי הוא שקול לבידוע סדרתי. מבחינתנו המבנה לקבוץ אם בדקת הוא נכון או לא כך האם התוצאה שהתקבלה בפועל וכלה להתקבל ע"י איזשהו בידוע סדרתי (ראו משפט סדר התכניות).

במודלי שנו נתמקד רק בפעולות שמתבצעות מול המסך. הסיבה שאותנו יכולים להתעלם משאר הפעולות היא שהן מתבצעות בזיכרון הפרטי של התכניות ואין לא משפיעות על התכניות האחרות. אבל הסיבה שאנחנו מתעלמים מהן היא שאחרת כך יהיו מסובכים. אבל אנו צריך להדגיר שיש שקילות אם יש שקילות לכל חישוב אפשרי. ואז רק להראות שאין שקילות צריך לעשות פורמליזציה אחת לחישוב שמפקח את הטבע.

תצמון סדרתי הוא תצמון שבו כל טרנזקציה מתבצעת ביחד. בתצמון מקבילי כמה טרנזקציות מתרחשות במקביל (בכל נקודת זמן מתבצעת פעולה אחת בלבד אבל יכולות להתבצע פעולות של טרנזקציה ב' לפני שגמרה טרנזקציה א').

הגדרת השקילות - שני תצמונים שקולים מתקיים אם:

- שני התצמונים מורכבים מאותם תנועות
- בשני התצמונים כל טרנזקציה קוראת אותו ערך לכל פרוט
- לשני התצמונים אותה השפעה על המסך

* והשקילות חייבת להתקיים לכל חישוב אפשרי.

במציאות יש הגדרה סינטקטית שמאפשרת (מגדירה) אלגוריתם שבודק אם שני תצמונים הם שקולים.

תיבה זיורת - כתיבה לרבים לפרוטוקול מבלי לבדוק מה יש בו, הלקח שנותר מן הסיפור תלוי באיך שהם חישבו פנימיים שתלויים בפרוטוקול אחרים. אבל בכל זאת לא לגו תלוי ב- B לכן נקרא כתיבה זיורת.

כה מיקרו תנועות חסרות השפעה - תנועות שאפשר פשוט לאחסן אותן לא תהיה שום השפעה על המשך. כל תנועה שבנתיבות לפרוטוקול שאל תנועה אחרת לא יקרה אחרי. אפשר לדעת שתנועה היא חסרת השפעה רק בדיעבד כשכל התצמון הסתיים.

<u>קונפליקטים בין תנועות</u> - אינטואיטיבית, ירצה להיות בזיה אם יש לסבאות שינוי שמבדלות פעולות של אותו פרוטוקול לפתחות אחת מהפעולות הוא תיבה (ברור שאם רק קוראות אז כה לא משנה באיזה סדר לה נעשה). סדר של קונפליקטים חיים לרישמר בתצמונים שבלים.

שני תצמונים הם <u>שבולי קונפליקטים</u> אם
- הם מורכבים מאותן תנועות (ובכל תנועה אותה סדרת פעולות)
- כל זמן פעולות שיש ביניהם קונפליקט מופיע בשני התצמונים באותו סדר.

מה הקשר בין שבולות מבטים לשבולות קונפליקטים?
אם הקונפליקטים נשמרים אז יש אם שבולות מבטים (ק) לחלוטין ומה ליבי הכיוון ההפוך? שבולות מבטים לא מורות שבולות מבטים. (למשל אפשר להראות שבתנאי מסוים הן הגדרות שבולות (נגיע לזה בהמשך)) בשקפים מסתבר לכם פוגמה. (עמ' 34).

תצמודים בין סדרתיות - זאת כל הכח מעניינת אותנו שקלות בין שני תצמודים ספציפיים, אלא אם בהינתן תצמוד שאינו סדרתי אם קיים איזשהו תצמוד סדרתי ששקול לתצמוד הנתון.

בר-סדרתיות אמיתית - קיבת תצמוד סדרתי ששקול אמיתי לו.
בר-סדרתיות קונפליקטית - קיים תצמוד סדרתי ששקול קונפליקטי לו.

בשביל להבדיל אם תצמוד היה בר סדרתיות קונפליקטית - יודעים אבל קדימות שאומר מי צריך לבוא לפני מי בתצמוד סדרתי שקול קונפליקטים. הבעיות הן אסקאות והבשתות הן קונפליקטים. יש בלגן בין $T_1$ ל- $T_2$ אם קיים קונפליקט שמייב ש- $T_1$ יהיה לפני $T_2$.
ואז אם מראש יש מספיק אז לא יכול שיש תצמוד שקול. ואם יש מספיק אז יודעים תצמוד סדרתי שקול ע"י מיון טופולוגי.

לעומת זאת, להבדיל אם יש בר סדרתיות אמיתית זו בעיה משה- קמ שלמה.

<u>אבל</u>, שתי הבעיות הן שקלות אם אין תיבות עיורות.

בגלל שבר סדרתיות קונפליקטית אות בר שקולות אמיתית (כי אותה סדרה ששקולה קונפליקטית תהיה גם שקולה אמיתית).

אבל, זה אותנו בכלל מעניין מה כל זה? באמצע נתונים אותנו רוצים לאפשר בידוק ~~[illegible]~~ מקבילי אבל שיהיה שון.
אותנו צריכים מנגנון שמאפשר אבל רק בידוק לעומת שמה שקורה הוא בר- סדרתיות (לא משנה לפי איזו הגדרה).

נראה מנגנון כזה שמשתמש בבר סדרתיות קונפליקטית.
לכל פריט יש שני סמי מנעולים - מנעול משותף S שמאפשר לקרוא מישהו ומנעול בלעדי X שמאפשר לכתוב אליו.

אם עסקה כלשהי לבדה פעולה הינו דרישה להשיג מנעול מתאים.
יש מקרים נוספים שאלו פונים כדי לקבל מנעולים והם אחראים
לוודא שהכל נעשה בסדר לדוגמא שאנו שתי עסקאות שיש לכן בדיוק
מנעול אם סקלוסיבי לאותו פרוט.

2PL
פרוטוקול נעילות בשתי פאזות - כל עסקה מתחלקת בשתי
פאזות - פאזת של הגשת מנעולים ופאזה של שחרור
מנעולים. כל עוד היא בפאזה של קבלה היא רשאי לבקש
כמה מנעולים שבא לה אבל ברגע שהוא שחרר ולו
מנעול אחד אסור לה לבקש עוד מנעולים.

אם כל העסקאות בתצמון מתבצעות ב 2PL אז התצמון הוא
בר סדרתיות קונפליקטיות. (הוכחה בחלק).

אבל רואים בתרגול את כל הבעיות. אבל יש סכנה בעיה של
מבוי בסתום (dead lock). התהליך רצה להמתין
באינדק עלאו כל אפשרות להתקדם. למשל

| $T_1$ | $T_2$ |
|---|---|
| R(A) | |
| W(A) | |
| | R(B) |
| | W(B) |
| | R(A) |
| | R(A) |
| R(B) | |
| W(B) | |

אז אין הכל בסדר

אבל ברשיון הן דרישות להשיג מנעולים לפרוט בשני כדי לבדוק ולרתוב אבל זה בלתי אפשרי כי המנעולים עוד לא הושתו תחרו.

פתרון מבוי בסתום - מתבצע ע"י דיבוק אבל. יש דומה לכל
עסקה ובלעו מ- $T_1$ ל- $T_2$ אם $T_1$ מחכה למנעול מ- $T_2$.
אם במקרה הזה יש מעגל אז בעיה. אז צריך לבחור אחת
מהעסקאות על המעגל ולבטל אותה רק שלא יהיה מעגל
ואז לבדוק אותה אחר כך.

אולם להבנות את האלה זה לא מאוד יעיל.
כל עסקה שמתחילה לבצע פעולה ל DBMS ומקבלת
חותמת זמן ייחודית. יש שתי אישורים

- Wait - Die - עסקה בעדיפה שמבקשת מנעול עלולה
להתבטל. נניח $T_1$ בעדיפה מנעול שיש ל $T_2$.
יש 2 אפשרויות:
  - לחכות ל- $T_1$ לחכות
  - לבטל את $T_1$
  - לבטל את $T_2$

באישור הנאות אם $T_1$ התחילה קודם (ויש תחכה.
אם $T_1$ בעדיפה יותר אז מבטלים אותה.

- Wound - Die - אם $T_1$ ותבקה יותר אז מבטלים
את $T_2$.

באל אותה מהשיטות האלה אין אפשרות לקיפאון. אולם
בכל מקרה יכולה להיות הרעבה - זיכי עסקה מסוימת
שכל הזמן מבטלים אותה (לה יכול להיות משתי הגישות)
ואז הוא נכנס להרעבה כי הוא אל פעם לא מצליח
לקבל את המנעולים הדרושים הדי לסיים.
בכה אפשר לטפל בבעיות. בשתי השיטות ברור
שהעסקה הותקבה הוותר אל פעם לא מתבטלת.
ואז כשמבטלים עסקה חותמת הזמן החדשה שהיא
תקבל לא באמת תהיה חדשה אלא היא תישאר עם
החותמת המקורית. ואז כל עסקה באיזשהו שלב תהיה
הותקבה ביותר אלם תתבצע.

נשאר עם הבעיה לבטל עסקאות. הדרך הכי יכולה
לבוא עם רשלנות אופטימית המוחרת ומה מה משתמש
שיכולים להתחרט באמצע הפעולה.

צריך לוודא שעסקה תתבצע באופן אטומי - או שהיא
תתבצע או שהיא לא תתבצע בכלל. לא יכול
להיות שתלך מהפעולות של עסקה יתבצעו ותלך לא.

תכניות מתחיימות בדיסק שנקרא מאפשרות את כל הפעולות.
אם היה התחיימה כל הפעולות נשמרות, אחרת שום
פעולה לא נשמרת מהמסך (בדיסק עדיין לא ב RAM).

(35) 15.1.07
DB

## התאוששות מנפילה

עסקאות בנויות על הזכרון שאו שהן מתבצעות בשלמותן או שאינן מתבצעות בכלל. תכי עסקה לא מבצעת תקין.
עסקה מתחילה לעבוד, מבצעת הרבה פעולות שהן בודאי לא אטומיות ביחד, ומסוף היא מבצעת commit - מודיעה למערכת שהיא מוכנה להתחייב. המערכת קיבלה את ההתחייבות הזאת אבל בינתיים אם לא משנה איזו תקלה קרתה התוצאות של העסקה נשמרות. אם התקלה קרתה בדיסק לכן גם התוצאות של העסקה לא נשמרות.

מניון ההתאוששות אחראי על:
- לבטל את כל הפעולות שעשתה תכנית אם היא נפלה לפני שביצעה commit (לעשות מלא של כישלון)
- לוודא שכל הפעולות של התכנית נשמרות במסד אם התכנית הגיעה commit.

לפעמים כשמבטלים עסקה לא מספיק שצריך לבטל מה עסקאות אחרות שקראו לנתונים שנכתבו ע"י העסקה המבוטלת. אולי יהיה אפשר כי ורק לדרוש שתלך מהעסקאות נבר עשו commit. ואי אפשר לבטל עסקה שעשתה commit
הפתרון לזה הוא 2PL strict וזה אומר שעסקה משתנה משתמשת בנתונים שנכתבו ע"י עסקה שעוד לא עשתה commit לכן יכולה להתבטל.

בעיית הפאנטום - עסקה נכנסה לנעול פריטים שקרם ביניים במסד אולם מתוספים למסד ע"י עסקאות אחרות תוך כדי פעולה של העסקה.

הנה תצמון אפשרי שמקיים Strict 2PL

| | EMP | DEPT | AGE | |
|---|---|---|---|---|
| $T_1$ | Jones | CS | 71 | Read |
| $T_2$ | Cohen | CS | 75 | Insert |
| | Levy | EE | 80 | Delete |
| | White | EE | 63 | Read |

יש לסקאלר $T_1$, $T_2$ שמתפשרת עם הדרישה לא אל אקסיומה ב- CS וב- EE.

ברור שבדוגמה לעיל יש בעיה. רק שלה יבוא שיון הינו דרישה אם לאנות את האפשרות לבדיקה פרוטים תרשים לאחר.

פתרון אפשרי - נעילת פרדיקטים - עסקה צריכה לקבל מנעול על כל הרשומות שמתיחסות פרדיקט מסוים, לדוגמא AGE>50 או DEPT = CS.

אולם זה תהליך יקר אך עושים בדרכים תלויים. לדוגמא אם יש אינדקס על DEPT אז נוכל לבדוק אלו נעלות.

## בקרת מקביליות לעל נעלות

- בקרת מקביליות אופטימית
  - אחסון יעילה
  - עיקרון פתרון יעילה
- בקרת מקביליות המבוססת על שיאושבחות מות זמן

הרעיון הוא שנותנים לכל עסקה חותמת זמן ומשתמשים
מהן רק לבדיקה שלה מי סדרתיות.
אלגוריתם מרכזי שהבידוד המקבילי דורש לדרוש שבכל
לתצמון סדרתי שמסודר לפי חותמות הזמן של העסקאות.
כל פעם שמגבלת פעולת קריאה או כתיבה דורש לבדוק
שלה קונסיסטנטי עם הבידוד הסדרתי. אפשר באופן שהו
שלב הבדיקה נרשם אז מבטלים את העסקה הבלונטית

מכיר שלאו לקב אלא מה שמגיעים בין
הם לא מקבילי לגמרי, כמו יש הרבה הגבלות.

בגרסת מקבילות אופטימית - דוחים את הבדיקה עד לשלב הסיום
של בידוד העסקה, מתוך הנחה שיש רק מיעוט קטן שלה לא
יסתדר. העסקה כותבת את כל מה שדורש בלוקאלי הפרטי
שלה ואז היו עושה את הבדיקה ואם היו עברה אז הוא
עם כותבת לגלובלי. אחרת הוא לא כותבת לגלובלי בכלל.
אבל עסקה מחולקת לשלושה שלבים - קריאה וחישובים,
בדיקה וכתיבה. הבדיקה והכתיבה הן בקטע קריטי.
משלב הזה נמדאים בבשר אם הצמד והשו מתבדר
בדרך סדרתי לגמרי. בגרסה הזו אל עסקה אחרת לא
יכולה לבדוק בדיקה וכתיבה. מה שנשאר למקבילי לה הוא חלק
של הקריאה וחישובים. הנחה היא שרוב זמן העסקה מתבדר
משלב קריאה וחישובים ואולם הבדיקה והכתיבה הן בלבד.
לכן זה בשאר מקבילי (רק חלק קטן הוא סדרתי).

במדגרת מפורט איך בודקים שעסקה באמת יכולה להתבדר,
שיש שבילות קונפליקטים לבידוד הסדרתי של העסקאות.
בשביל לעשות את הבדיקה הזאת שומרים טבלה שרשום
מה אינו עסקה בראה איזה פרוט. ואז כשעושים
בדיקה ~~אם~~ של T1 אז אם T1 כתבה לתוך משהו

ש- $T_2$ קרוא אז צריך לבטל את $T_2$ כי יכול שיש לה
תלויות לא נעולים. הטעם כדי לבטל את $T_2$ איך אנו
לתת לה לרוץ באופן עקיף? - אדל לתת לה להמשיך
כי הדפים שהיו צריכים יצאו ל buffer pool וגם
בפעם הבאה היא תרגל מהר יותר כי כל המידע שהיה
צריך כבר יהיה בזיכרון. הסניף, נשים לב שלפני ש- $T_1$
כותבת לדיסק הוא בודקת כותבת ל-buffer pool לכן
המידע שיימצא שם בשבל הריצה הבאה של $T_2$ הוא
נכון.

## בקרת מקבילות ומהימנות של תנועות זמן

לכל פריט A במסד יש שתי תנועות זמן - הראשונה,
RTS אתי הפריט נקרא לאחרונה ע"י עסקה הצעירה ביותר והשנייה WTS
אתי הפריט נכתב לאחרונה. (ע"י עסקה הצעירה ביותר).
בגדר, אם עסקה מנסה לקרוא את A אחרי שעסקה
צעירה קראה את A אז לא נשנים את תנועת הזמן.
במקרה יש פירוש של הבדיקה אתי אפשר לקרוא/לכתוב.

(37) 22.1.07
DB

# התאוששות

כדי שתתאפשר התאוששות צריכות להיות תכונות ACID.

- אטומיות - כל עסקה מתבצעת כאילו היא פעולה אטומית - או הכל או כלום.
- עקביות - כל עסקה עקבית, כלומר נשיגה, כלומר לה בר סדרתיות.
- בידוד - בידוד עסקה מבוצע מהשפעות של עסקאות אחרות
- ~~המשכיות~~ עמידות - אם עסקה מתחייבת אל מיד אחרי לה כל הנתונים שלה צריכים להישמר.

עקביות ובידוד הן תכונות שמובטחות ע"י כל מנגנון שמבטיח בר סדרתיות (למשל strict 2PL).
אטומיות ועמידות הינן תכונות שאחראי עליהן מנגנון ההתאוששות.

אם עסקה מתבטלת והמתשר ממשיך לעבוד (למשל במקרה של dead lock) צריך לבטל את כל השינויים שלה. ב strict2PL אם עסקה מתבטלת אל צריך לבטל רק אותה כי כל המפתחות אפשר לקח לא ירלו להיות שתוכניות אחרות עשו קריאה מלוכלכת.

מה אם הדיסק עצמו נהרס? לא כאן באה לידי ביטוי עצמו ונדבר קצת בהמשך. אבל ברגע שהדיסק לא נפגע. יש לה יכולת שהרתיבה של כלום לדיסק כן פעולה אטומית כל עוד יש תשתית. אם התשתית נפלה בדיוק תוך כדי רתיבה של כלום אין ודבר להיות שטוב לא תקין. אבל מה התאוששות. יכול להיות שיש צורך לשחזר את הכלום ושתרון (אבל רק אותו).

ה נתונים לא מיד נכתבים לדיסק. כל עוד יש מקום במאגר
התוכנים הנתונים לא נכתבים לדיסק עדיין.
מתי נעשה flush? אם נעשה לפני ה commit
זה לא טוב כי המחשב יכול ליפול בינתיים. באותו
אופן זה לא טוב לעשות flush אחרי ה commit.
Oh my God מה נעשה? יהיה דיסק נופל שמשתמש
log עם רשומות לכל פריט מה היה הערך לפני
העסקה ומה היה הערך אחרי העסקה וכמובן עם אם
נעשה commit או לא.
נניח שהמחשב נפל. רשומה עולה שיש צורך להסתכל
ב-log ולבדוק מה אמור להיות. יש הבדלים שלתבים
בדיסק של ה log כמו נקודה כל נכתבים במאגר התוכנים.
אבל רגע שכל העסק יעבוד שינויים נכתבים למסד של הדיסק
רק אחרי שהם נכתבים ל log של הדיסק. עסקה
חייבת לכתוב את כל השינויים שעשתה ל log של הדיסק
לפני שהיא אותחמת.
רששינויים כתובים ב- log הם לא מוכרח כתובים במסד
אבל כרי הסדר כי אם ב log רשום שעסקה
עשתה commit אבל בשלב מסוים זה מטות יעבור למסד.
אבל נקודת הזמן שבה מובטח לנו שעסקה עשתה commit
זה כשבלוגים של הדיסק רשום שהיא עשתה commit.
אם עסקה שותה משהו במסד זה בוודאות כתוב ב- log. אבל
לא כל מה שכתוב ב log בהכרח נעלם במסד.

נראה ראינו יש סוג בעיה גדול - כל דבר צריך לכתוב בעמיים!
שבלבלים זה דווקא מייעל. ה- log הוא קובץ סדרתי -
כל העסק מוסיפים לו רשומות לסוף. לכן הכתיבה היא מהירה.
ובמקום שכתבים ל log אפשר להשתמש את הכתיבה למסד.
ואז לא צריך להדביק לכתוב מלא כתובים בבלבל

שהפריטים שאותן מעדכנים בהם מפוצלים על בלוקים שונים. אם רתבית קודמת ל log וקרסה בכמה זמן ל מסד אבל לא לוגה פתות זמן. הקריטי שהדיסק של ה-log יהיה שורד.

אתי דרך לרתיבה לדיסק? השיטה המקום בזיכרון המוקצים. לא השמוספים השומרת commit ל-log דרך לעשות flush ומעשה לרתבה הכל ל-log: כי לא מה שאומר שבאמת יש commit. אלה הבלוקים היחידים שבהם יש לרתבה לדיסק. כל רתיבה אחרת אפשר לה שהות לנצח.

טוב, אבל השיטה הזו דרך לפגוע בלוקים, אם נבחר את הקורבן? יש לזה כל מיני שיקולים. לדוגמא, רצוי לפגוע בלוקים שאין עליו מנעולים אבל קודם דרך לרתבה אותו לדיסק. אם כל הבלוקים יש מנעולים אפשר לנקוט בקום של בלוק שנמצא בסיכון הפנימי אבל קודם נרשום את זה לדיסק בלוג ואז לעקרה שנצטרך לבטל את העסקה הכל אפשרי כי כל מה שעשה רשום בלוג בדיסק.

* שאלה שהייתה פעם במבחן - מעקבה שרתיה נפולה לא דרך לעדכן מה נעול.

בשביל לבטל עסקה דרך לעשות undo לכל מה שהיה עשה אם במקום החוצים וגם בדיסק.

אם המחשב נופל אז השיטא עלה מחדש עושים אנליזה - עוברים על כל ה- log ומודקים איזה עסקאות עשו commit ואיזה לא. אחרי המשבר של עשיה רושמים את כל מה שדרך ברור שצה לא מעשי לעבור על כל ה log (זה ורוב לוקח שנים אחורה) אבל פעם בכמה זמן רשאון עומס של המע' אז מפסיקים לקבל עסקאות חדשות, מסיימים את הישנות ועושים checkpoint ואז בהתאם אושרות דרך לחצור רק עד ה checkpoint האחרון.

המבחן -

כמו שפיה קודמות.

14 שאלות קצרות למבחני בלשי הלכות ארוכות.

צריך לדעת את כל התוכן של בורין.

3 שעות ואין הארכות (וגם אין צורך)

תוכן סאר