

SUMMARIZED BY NOAM KIMHI
noam.kimhi@mail.huji.ac.il

DATABASES

YEAR 2

SEMESTER B

2024-2025



Course Summary

COURSE NUMBER 67506

Week 1: ER Diagrams

Introduction 1.1

What is a Database?

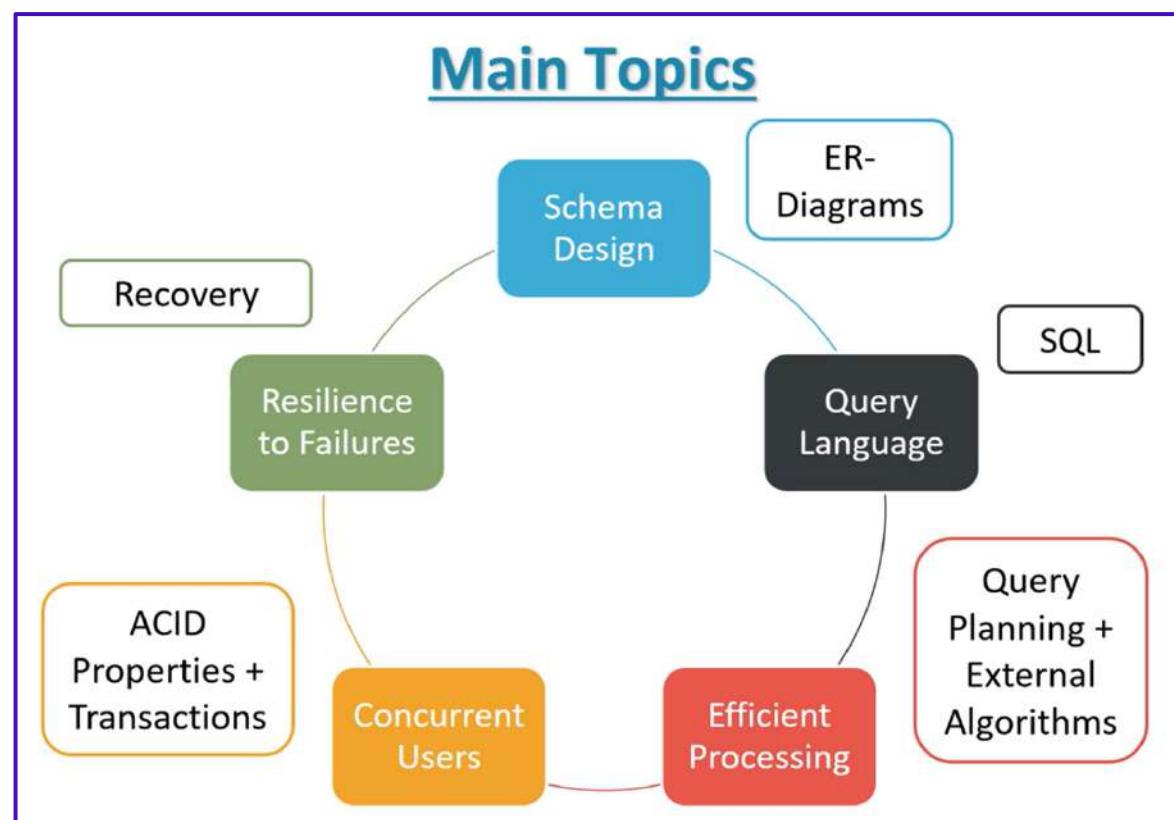
միջավայրում կատարվություն կատարելու համար:

օ – օպերատոր մասնակիցներ

օ – օպերատոր մասնակիցներ

Main Topics

Եթե առաջարկությունը կատարված է առաջարկություն կատարելու համար:



What do we learn in the course?

Այս դասընթացում կատարված է առաջարկություն կատարելու համար:

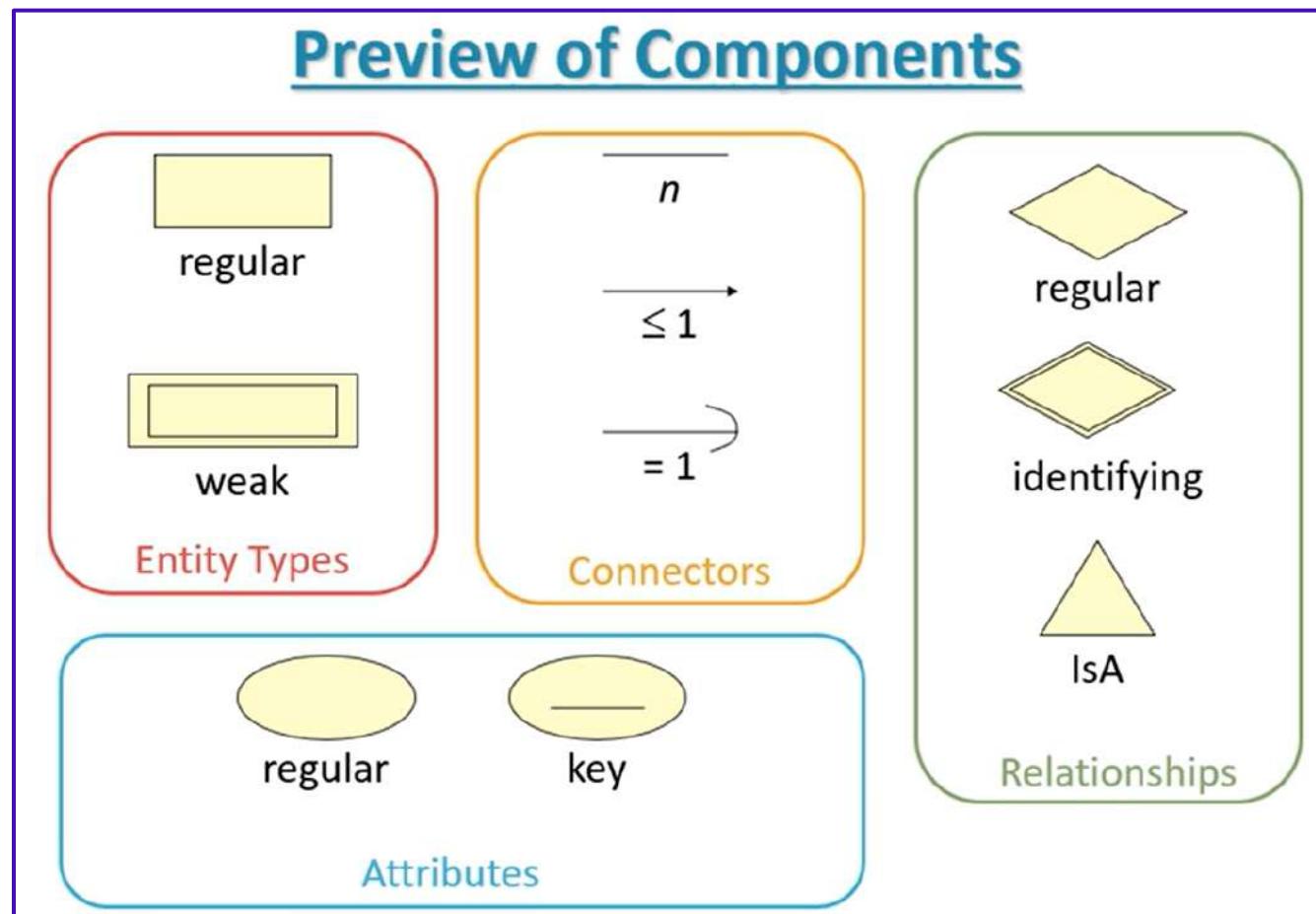
Բայց այս դասընթացում կատարված է առաջարկություն կատարելու համար:

Այս դասընթացում կատարված է առաջարկություն կատարելու համար:

ERD 1: Entities, Attributes, Relationships 1.2

Steps in building a database

1. Նշում պահանջվող գործությունների մասին – այս գործությունները պահանջվում են պահանջվող գործությունների մասին:
2. Կոնսեպտուալ նշում – պահանջվում են պահանջվող գործությունների մասին:
3. Տեսական նշում – պահանջվում են պահանջվող գործությունների մասին:
4. Տեսական նշում – պահանջվում են պահանջվող գործությունների մասին:



כאן יש את כל המרכיבים בדיאגרמת יישיות קשרים.

ישיות, תכונות וקשרים

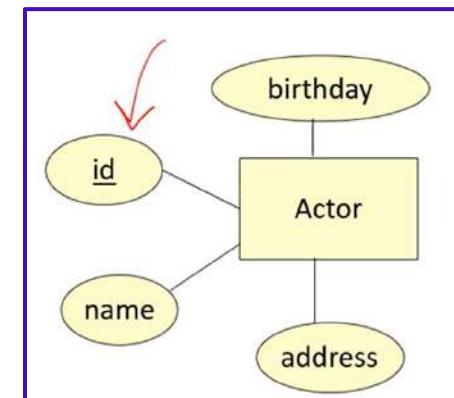
ישיות Entity - אובייקט בעולם שאפשר להבדיל ביןו לבין אובייקט אחר באמצעות התכונות שלו (למשל תלמיד, קורס). מצד שני קצת יותר קשה לחשב על עלה ספציפי באובייקט, כי קשה לתאר אותו ובולם ידעו באיזה עלה ספציפי מדובר).

קובוצת יישיות מאותו סוג (קובוצת סטודנטים, קבוצת מרצים, קבוצת קורסים). נציין אותם בדיאגרמה בעזרת מלבן.

$$\text{Person} = \{\text{p1}, \text{p2}, \text{p3}\}$$

תכונות Attributes – נשתמש בהן כדי לתאר את הישיות בתחום הקבוצה. למשל עבור קבוצת יישיות של *Actors* נוכל לקבוע תכונות כגון:Tарик, שם, גיל, מקום, נסיעה. לכל הישיות בקבוצה חיבר ליהות **ערך אמייני** עבור כל תכונה (כלומר לא זהה). כך נגיד מה יכול ומה לא יכול להיחשב בתכונה. למשל במקרה של זוג לא יכול להיות תכונה בקבוצה הזה, כי תכנו שחקנים לא בן זוג. כמו כן, לכל תכונה חיבר ליהות **ערך ייחידי**. למשל email יכול להיות תכונה רק אם כל שחקן יש בדיק בהתובת מיל אחת. אם קיים שחקן עבור נרצה לשמור כמה תכונות מיל, או שחקן ללא מיל בכלל – לא נוכל להגיד מיל בתכונה.

מפתח Key – סט מינימלי של תכונות שיכולים להבדיל באופן ייחידי יישיות בין יישיות בתחום הקבוצה. למשל לשחקנים יהיה מפתח ID מספר זהה. לכל קבוצת יישיות חיבר להיות מפתח, אותו נזכיר עם קו תחתון מתחת לשם התכונה:



האם לחופין הזוג שם וכותבת יכול להיות מפתח? תלוי מה אנחנו יודעים על העולם. נניח שכן – ומה מתקיים תנאי המינימליות? כי שם בלבד או בתובת בלבד לא יזהו שחקן, אבל הבחירה שלהם כן (אם זה מתקיים בעולם זהה). האם הזוג ID ושם יכולים להיות מפתח? לא, כי ID בלבד כבר מזהה את השחקן ומהו הוא מפתח עצמו.

קשר Relationship – אסוציאציה בין שתים או יותר יישיות. למשל, יוסף לומד את הקורס DB – אז (יוסף, DB) יכול להיות קשר.

קובוצת קשרים Relationship Set – קבוצה של קשרים מאותו סוג. את קבוצת הקשרים נסמן בדיאגרמה באמצעות מעוין:



השחקן מכבב בסרט (צריכה להיות בדיאגרמה גם תכונות לשחקן ולסרט, זה הושמט לשם הקיצור).

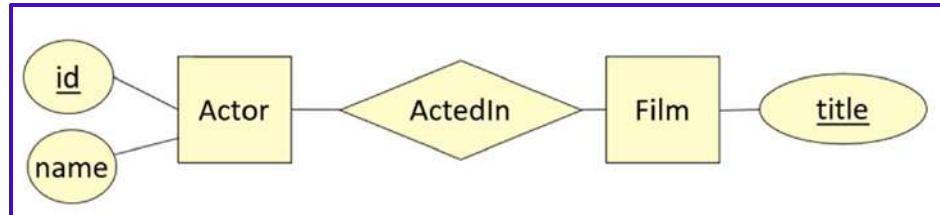
נבחן כי באופן מתמטי:

$$\text{Actor} = \{\text{a}_1, \text{a}_2, \dots, \text{a}_n\}$$

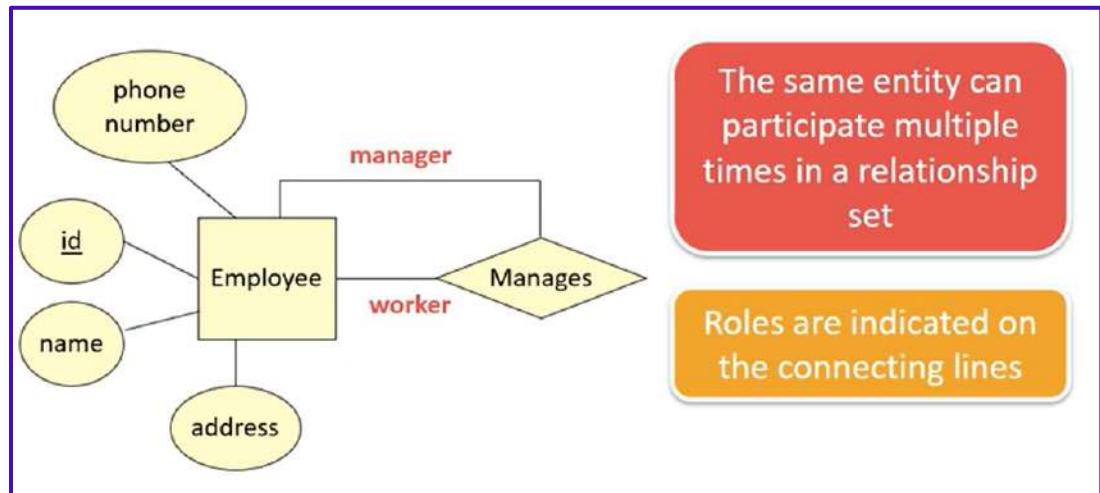
$$\text{Film} = \{\text{f}_1, \text{f}_2, \dots, \text{f}_m\}$$

$$\text{StarsIn} \subseteq \text{Actor} \times \text{Film}$$

לכן כמות הזוגות שיכולות להיות בקובוצת הקשרים StarsIn הוא לפחות כמות השחקנים כפול כמות הסרטים.

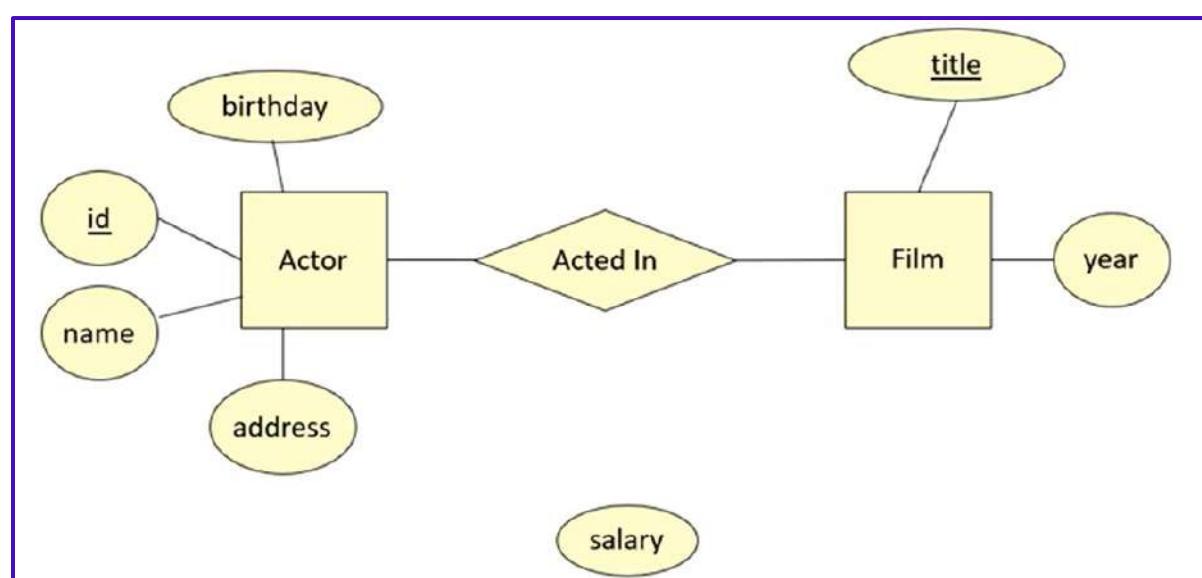


קובצת קשרים וקורסיביות Set Recursive Relationship – קבוצת קשרים שמקשרת בין אותה ישות לעצמה (אותה ישות משתתפת בה יותר מפעם אחת).



את התפקידים השונים שהעובד משמש בתפקיד משתתף בקבוצת הקשרים Manages נציג ליד הקשרים המחברים (מוגיע באדום) נבחן כי אכן קבוצת הקשרים מקיימת: $\text{Manages} \subseteq \text{Employee} \times \text{Employee}$.

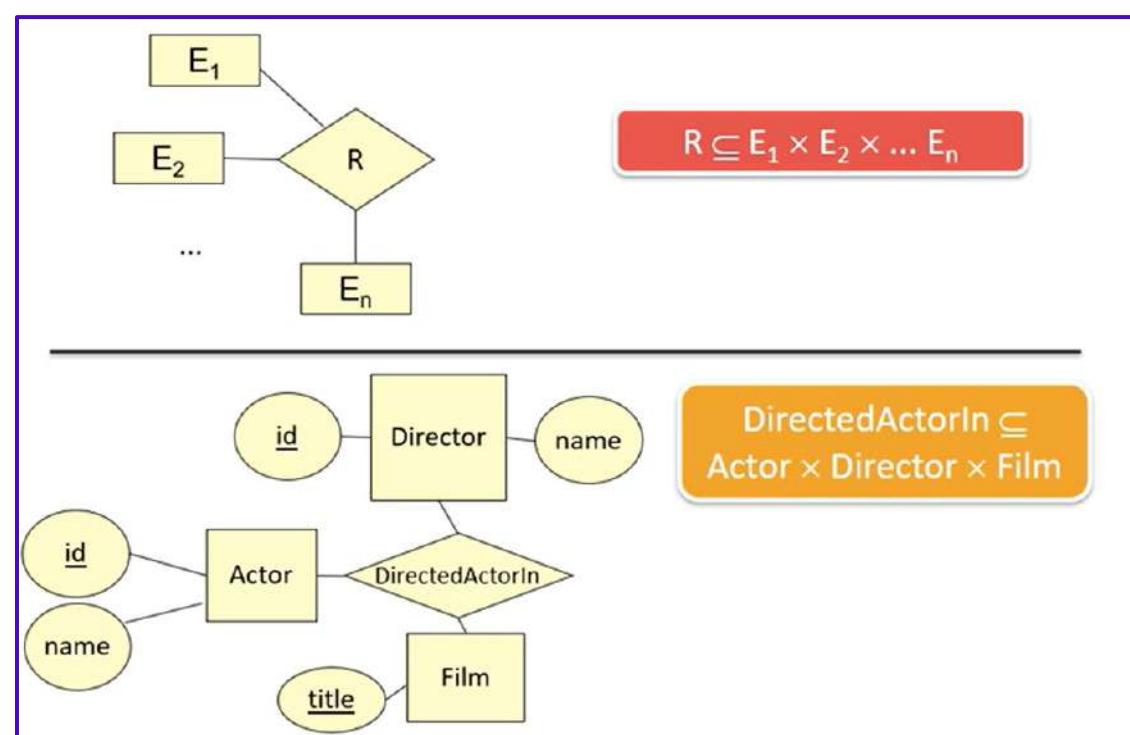
[דוגמא לדיאגרמה](#)



לאן נחבר את תכונת salary? אם נחבר אותה לישות השחקן, המשמעות היא שלשחקן יש משכורת קבועה בילי קשר לסרט בו משחקים, זהה לא מה שקרה בדרך כלל. אופציה נוספת היא לחבר את המשכורת לישות הסרט – וזה **לסרט יש משכורת קבועה** עבור כל השחקנים, זה גם לא מאד סביר. האופציה שיכולה הנראת אידיאלית במקרה זה, היא לחבר את תכונת המשכורת לקבוצת הקשרים **In** – גם לקבוצת קשרים יכולים להיות תכונות, והן מתארות את הזוגות בתחום הקשר. ועכשו לכל זוג Actor, Film התווסף עוד ערך (תכונה) שהוא המשכורת, המשכורת צריכה לקבל ערך אחד, ובديוק אחד.

[n-ary Relationship Sets](#)

עד עכשו ראיינו קבוצות קשרים ביןaries, אבל יכולים להיות מבוסן קבוצות קשרים של 3,4,...,n ערכיהם:



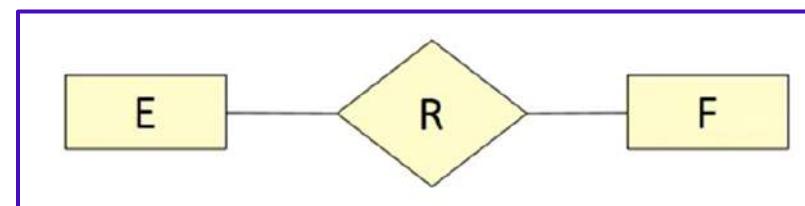
תרגיל: נניח שיש 2 שחקנים, 3 במאים ו-4 סרטים. מהו טווח מספר השלשות בקשר **DirectedActorIn**? המינימום הוא 0 כי זו תת קבוצה והיא יכולה להיות ריקה. הכמות המקסימלית היא $4 \cdot 3 \cdot 2 = 24$, ולכן יש לפחות 24 זוגות. תשובה סופית: [0,24].

כמה זוגות יכולים להיות? התשובה היא רק **0** – זה מבלבל סטודנטים. לא יכולות להיות זוגות, כי **DirectedActorIn** הוא אוסף של **שלשות** בלבד.

בללי

כלומר כמה פעמים אפשר שאוֹתָה ישות תשתתף בקבוצת קשרים.

למשל בדיאגרמה הבאה, אין שום אילוץ זהה:

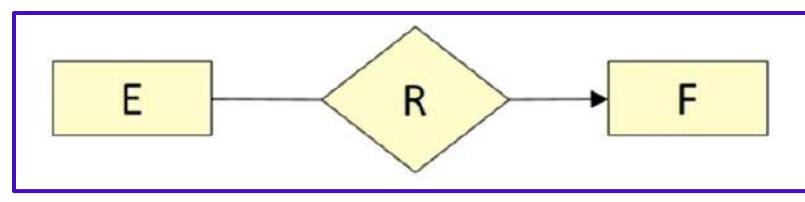


מצב זה נקרא Many-to-Many.

לכן בהינתן ח' ישות ב-E, ו-ט' ישות ב-F.

בקבוצת הקשרים R יש $[0, n] \times [m, n]$ ישות.

אם נרצה להגביל שישות ב-E יכולה להשתתף בכל היותר פעם אחת ב-R, נוציא ח' לכיוֹן F כר':

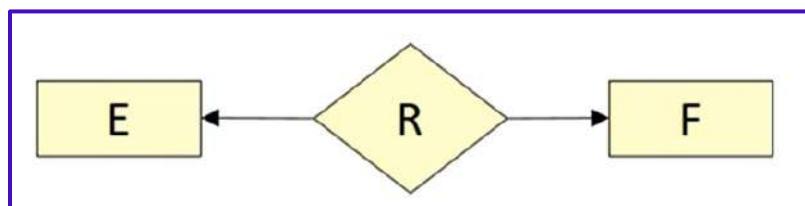


מצב זה נקרא One-to-Many.

כל' ישות ב-E יש לכל היותר F אחד שמתאים לו. במצב זה, עדין כל' ישות ב-F יכולה להשתתף ב-R כמה פעמים שתרצה.

בקבוצת הקשרים R יש $[n, 0]$ ישות.

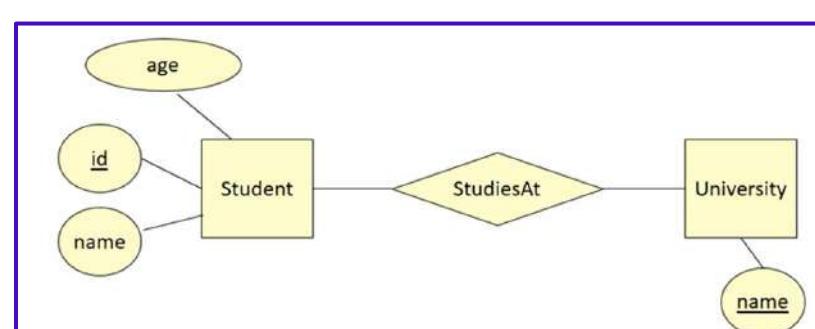
לבסוף, נוכל להגביל את הקשרים להיות אחד לאחד:



מצב זה נקרא One-to-One.

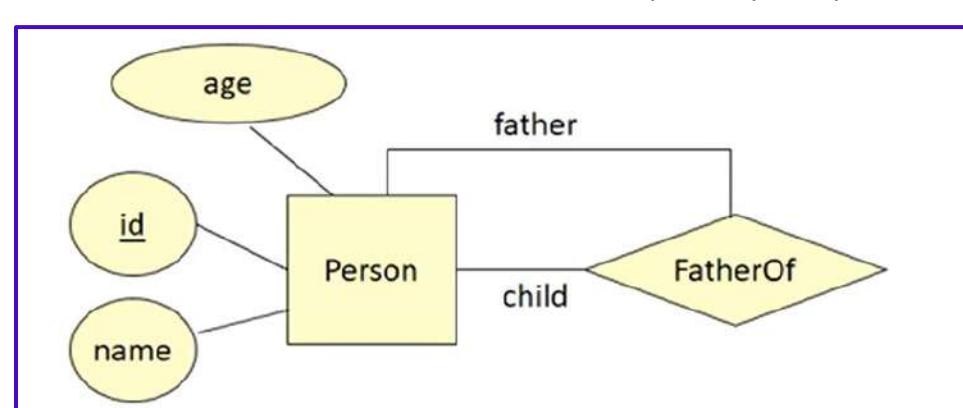
כלומר כל' ישות ב-E יכולה להשתתף ב-R בכל היותר פעם אחת, וכל' ישות ב-F יכולה להשתתף בכל היותר פעם אחת.

בקבוצת הקשרים R יש $\min\{n, m\}$ ישות.

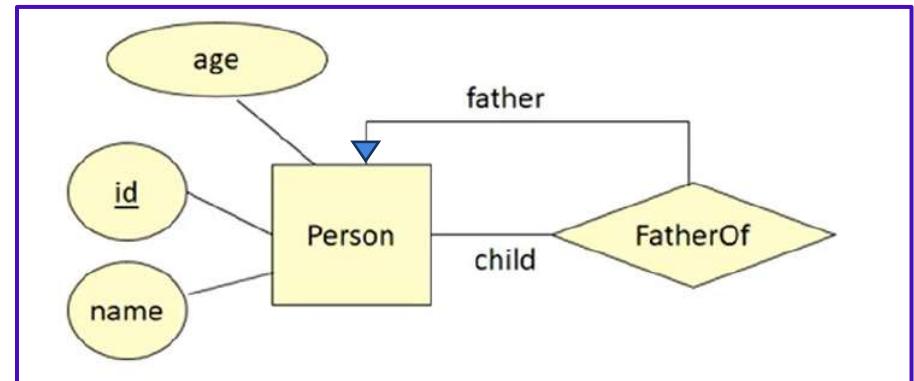
דוגמאות בלליות

איפה נשים כאן את החץ? לשים ח' לכיוֹן סטודנט אומר שלכל אוניברסיטה בקשר יש בדיק סטודנט אחד – לא ממש הגיוני. אם נשים לכיוֹן אוניברסיטה, זה אומר שכל סטודנט לומד בכל היותר באוניברסיטה אחת (טיפ לזכור: הסטודנט מצביע על האוניברסיטה בה לומד, لكن החץ לכיוֹן האוניברסיטה).

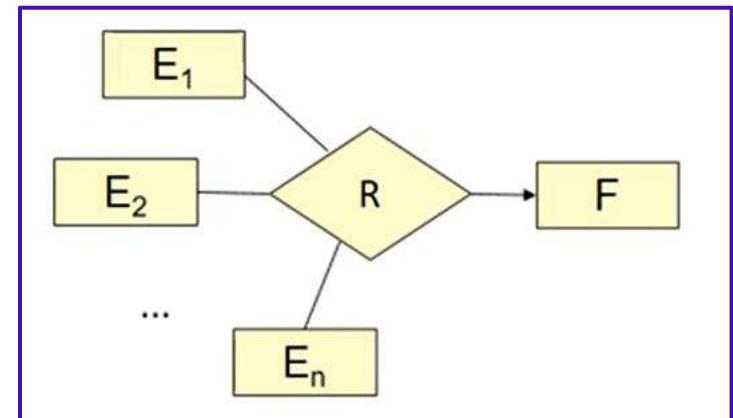
דוגמא עם קבוצת קשרים רקורסיבית:



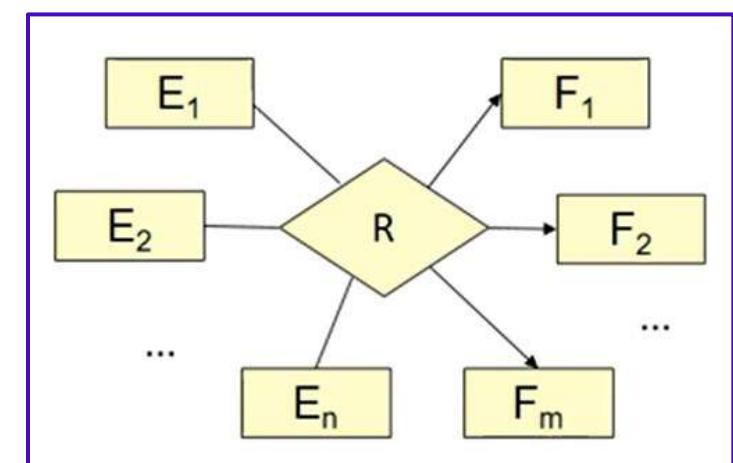
איפה כאן נשים את החז'ן לכל יلد יש אבא אחד לכל היותר, החז'ן יהיה לביוון האבא כלומר כך:



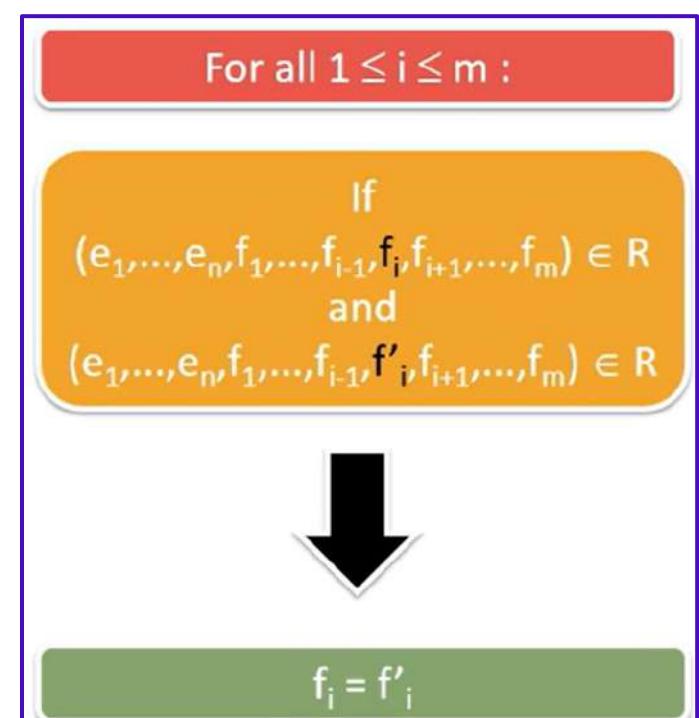
אפשר כמובן להשתמש בחיצים גם בקבוצת קשרים לא ביןארית. בדוגמה הבאה יש קבוצת קשרים של $1 + n$ קבוצות ישיות, עם חז'ן F:



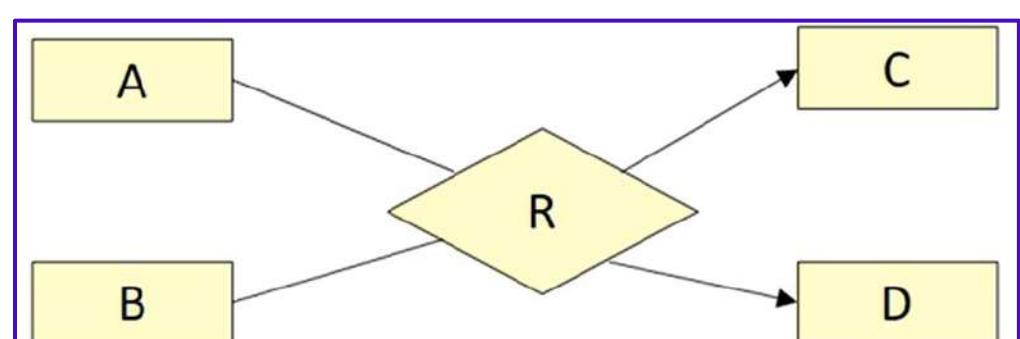
מה זה אומר במצב זה? קודם כל, $R \subseteq \{E_1 \times E_2 \times \dots \times E_n \times F\}$. הדיאגרמה זו אומרת שבוינטן אותן e_1, \dots, e_n אם יש קשר (e_1, \dots, e_n, f) וגם קשר (e'_1, \dots, e'_n, f') אז בהכרח $f = f'$. בוא נסביר עוד:



במצב זה, לכל אחד מ- m החיצים צריך להתקיים:



[דוגמא ספציפית](#)

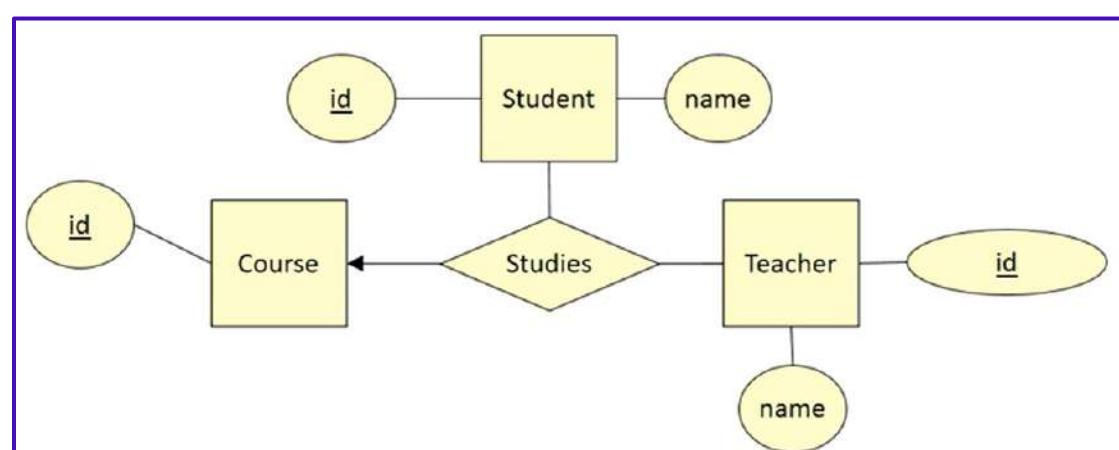


$R \subseteq A \times B \times C \times D$ מתקיים

מה אומר החץ לתוך C? אם $a \in R$ וגם $c' \in R$ אז $c = c'$. באופן דומה החץ לתוך D ייתן תכונה דומה. מה הנסיבות המקסימלית של ריבועיות- R ? מהחץ ל-C אנחנו יודעים שיש לכל היותר $d \times b$ ריבועיות. למה? כי כל שלישיה צדו של d, b, a נוכן להתאים לכל היותר ל- c אחד.

מהחץ ל-D אנחנו יודעים שיש לכל היותר $c \times b$ ריבועיות. לכן המינימום ביניהם הוא בנסיבות הריבועיות המקסימליות. (הנסיבות המינימלית עדין 0).

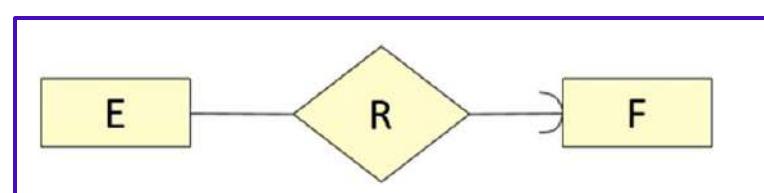
מה אפשר להסיק מהדיאגרמה הבאה?



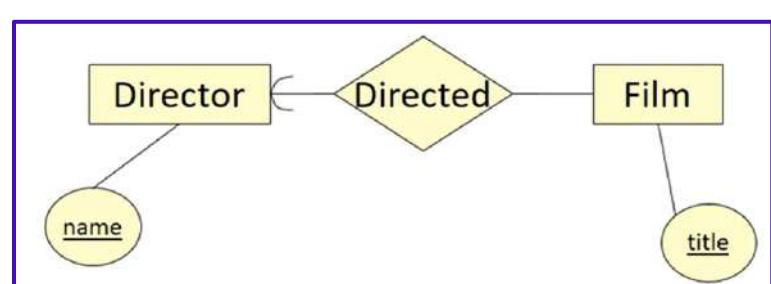
מרצה מלמד סטודנט בכל היותר קורס אחד בלבד. אם נוסיף חץ לתוך סטודנט, מה זה אומר בעצם? קודם כל, מה שהיא לפני עדין תקף – כל מרצה מלמד סטודנט בקורס אחד בלבד. עבשו קיבלנו מידע נוסף, שבכל מרצה מלמד בכל קורס שלו בכל היותר סטודנט אחד. הלקח פה הוא שכאשר יש קשר α -ה צריך לדון במשמעות של כל חץ בנפרד.

Referential Integrity 1.4

נושא זה עוסק בשאלת האם ישות **חייבת** להשתתף בקבוצות קשרים.
נסמן זאת באמצעות חץ עגול, למשל:



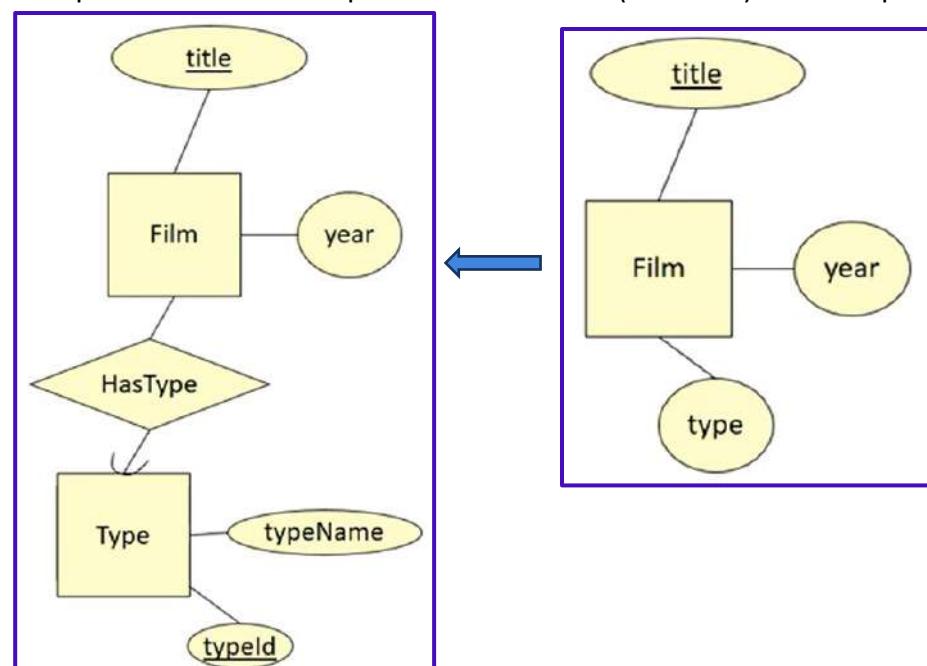
זה מסמן שככל ישות ב-E **חייבת** להשתתף בבדיקה פעם אחת בקשר R עם ישות מ-F. דוגמה ספציפית:



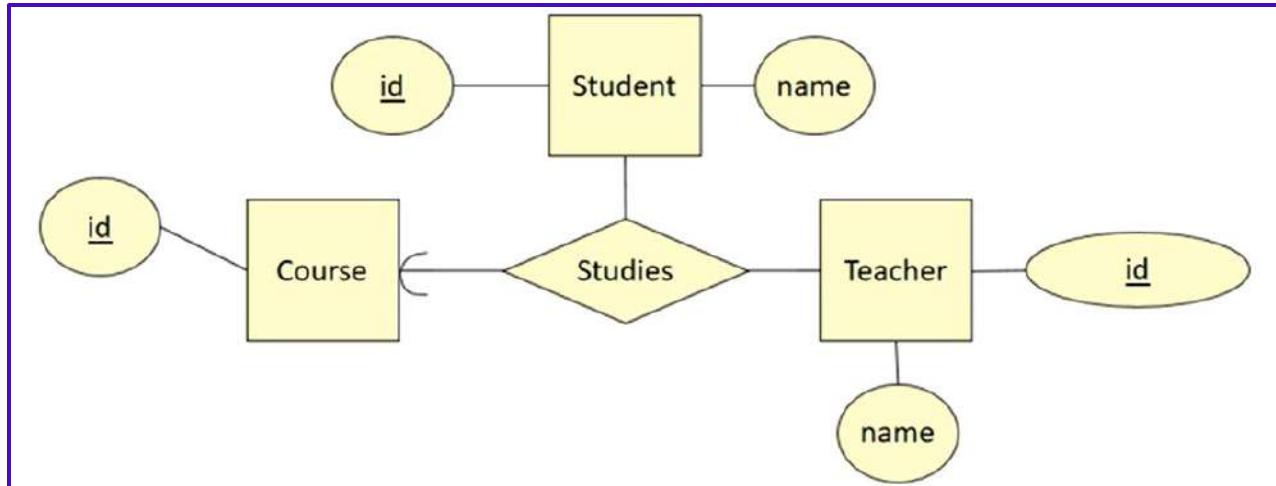
לכל סרט יש **בדיקה** במאי אחד.

Attributes vs. Entities

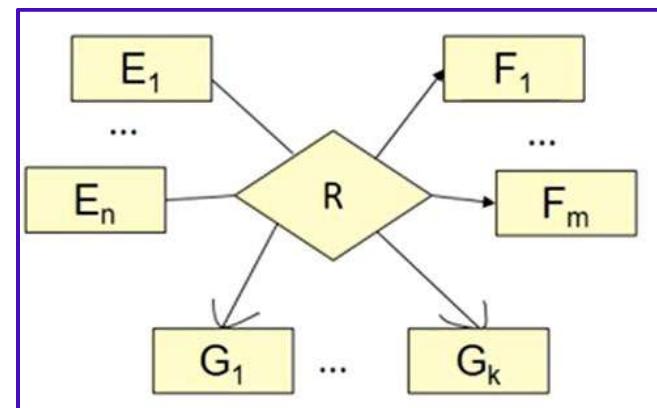
עם התווסףת האפשרות לציין את העובדה שישות **חייבת** להשתתף בבדיקה פעם אחת, נוצרת לנו אפשרות לאפשרות להשתמש לפעם בקבוצות ישויות במקומם בתכונה (attribute). זה מאפשר לנו ליצור קשרים יותר מורכבים כגון:



מקובל שכשתוכנה מגיעה מתוך קבוצת ערכים קטנה, נכניס אותה בתוך entity, כמו ז'אנרים של סרטים. לעומת זאת, התכונה title לא בא מתוך קבוצה קטנה של ערכים, ולכן היא נשארת בתכונה. בוא נראה דוגמה נוספת:

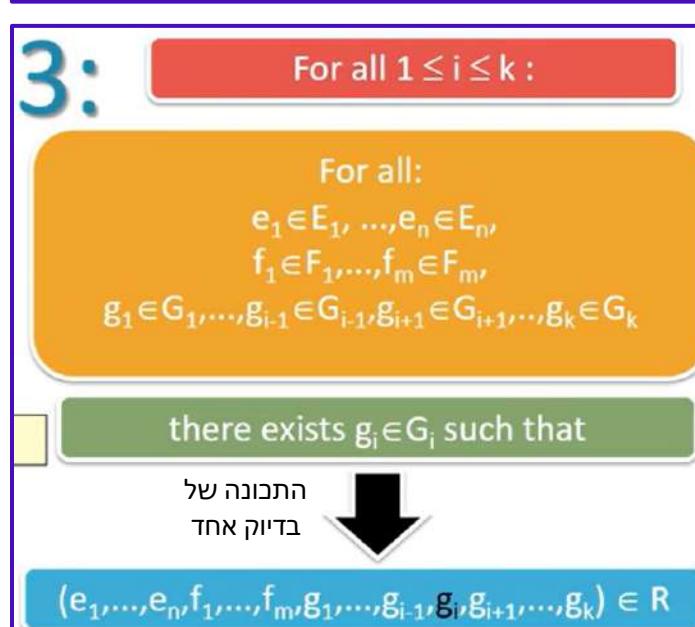
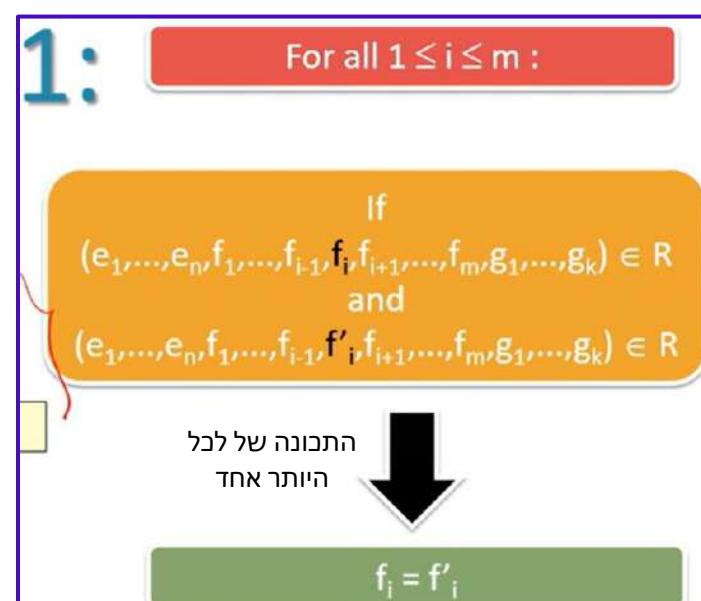
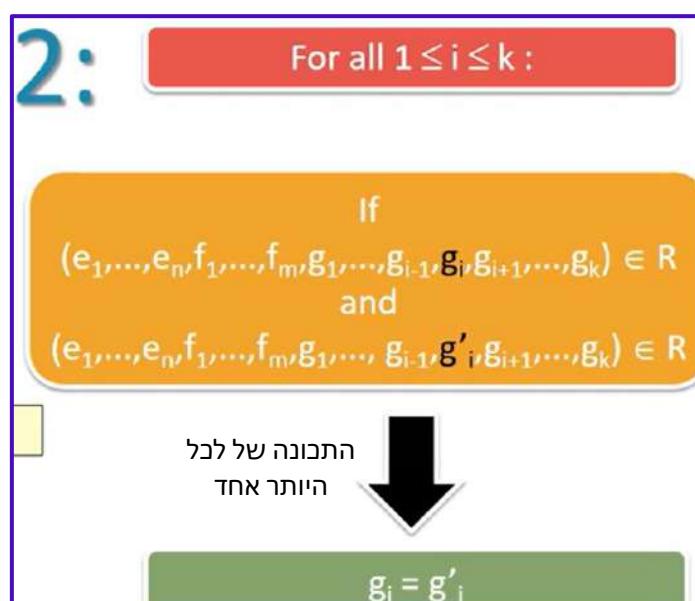


מה אומר החץ העגול לתוך קורס? לכל זוג סטודנט ומרצה, יש לבדוק קורס אחד שבו המרצה מלמד את הסטודנט. בוא נסתכל על מקרה כללי יותר:

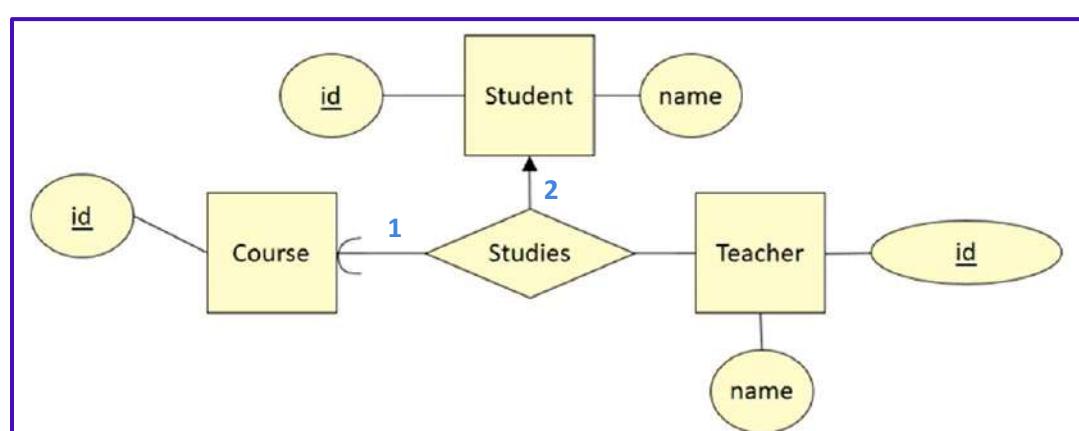


בוחיצים העגולים מתקיים:

בחיצים הרגילים, זה אומר בדיקת כמה שראינו בסרטונים הקודמים



דוגמה קונקרטית

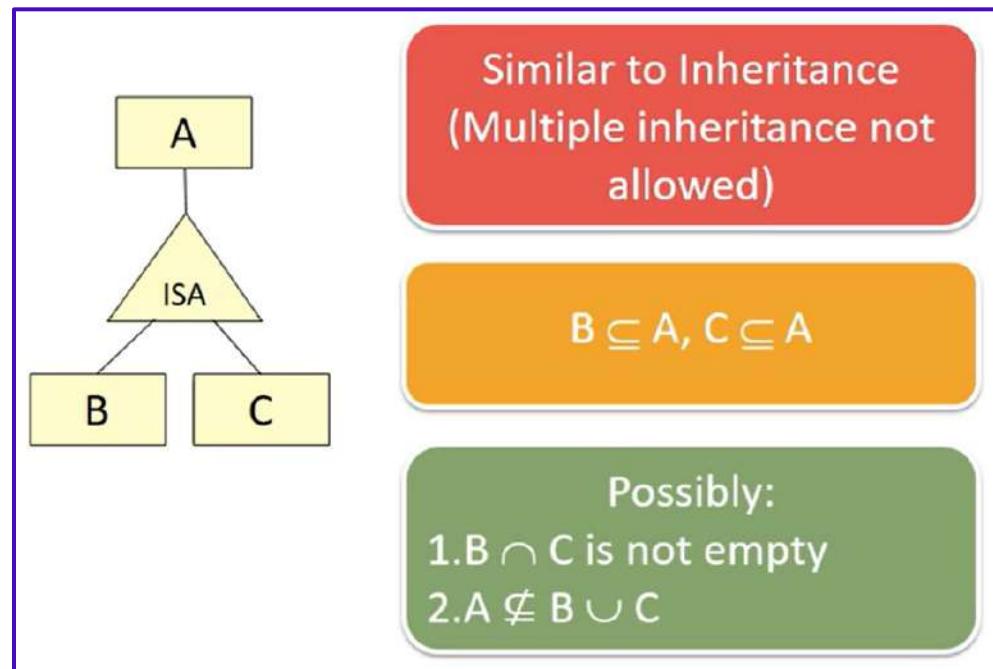


- 1.משמעות החז' היא: $\forall s \in Student, \forall t \in Teacher \exists c \in Course (a \text{ single } c!) s.t (s, t, c) \in Studies$
- 2.משמעות החז' היא: $\forall t \in Teacher, \forall c \in Course: if (s, t, c) \in Studies \wedge (s', t, c) \in Studies \Rightarrow s = s'$

ירושה 1.5 Inheritance

ירושה ב-ER

דומה לו שראים בשפות תכנות, אבל לישיות שלנו אין פעולות – אז הירשה תהיה רק על התוכנות.



משמעות ירשה בדיאגרמה באמצעות מושל ובהכו ISA בולם מקיים יחס של a is. כאן למשל B הוא סוג של A , ולכן ירשה את התכונות של A .

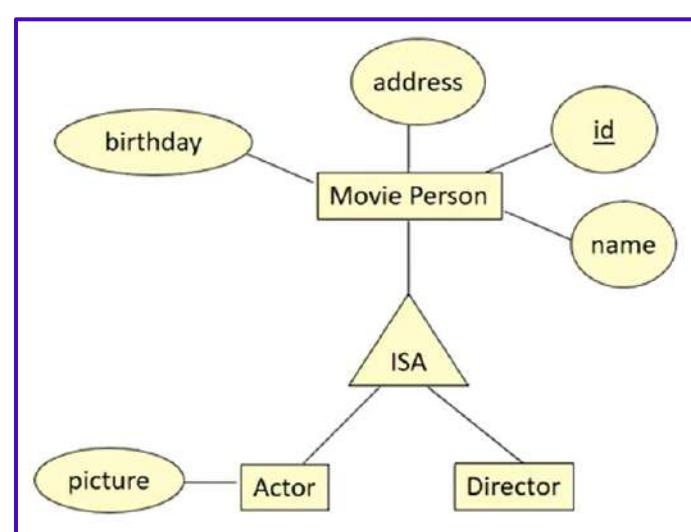
נשים לב לכמה נקודות:

o B מוכלת ב- A (למשל A קבוצת אנשים ו- B קבוצת ילדים).

o החיתוך של B ו- C לא בהכרח ריק, למרות איך שהדיאגרמה נראה.

o לא מחייבים ש- B ו- C מבסים את כל A .

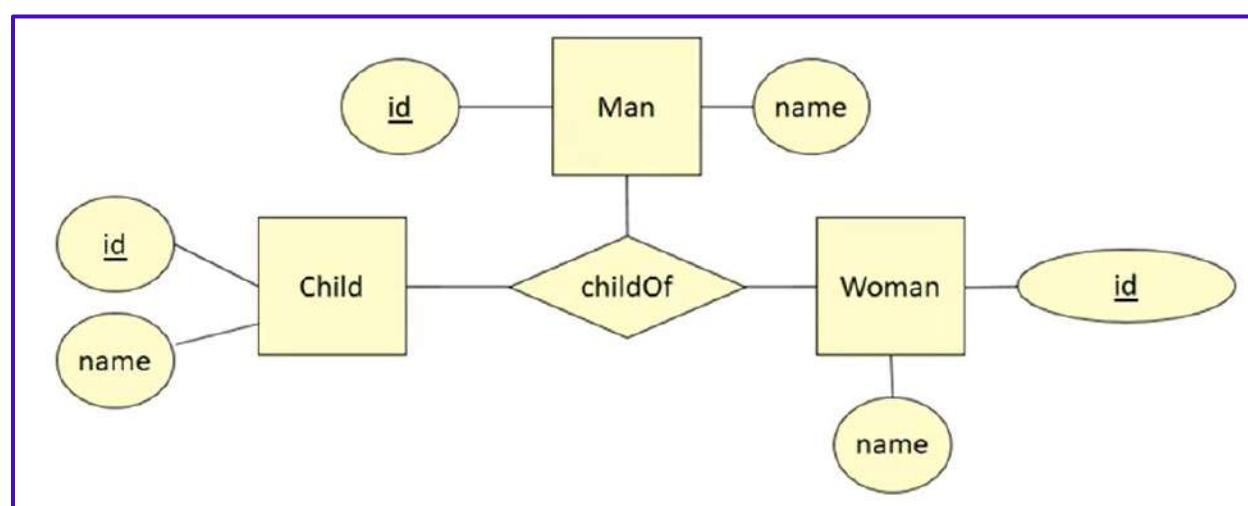
דוגמאות קונקרטיות



מה הפתח של Movie Person? כמובן $\langle \rangle$, לפי הuko בדיאגרמה.

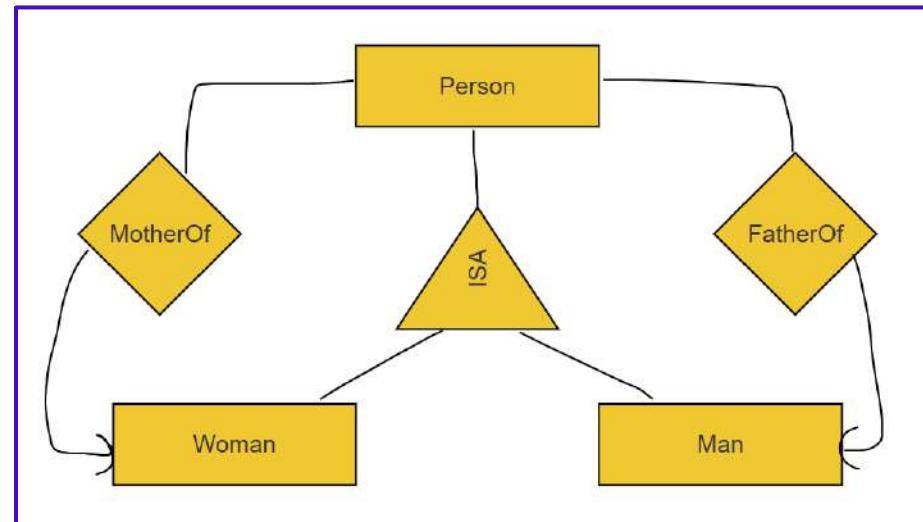
마וחר שגם Actor וגם Director שניהם סוג של Movie Person, יש להם את אותן התכונות, ולכן המפתח של שניהם הוא $\langle \rangle$ גם כן.

דיאגרמה בעיינית



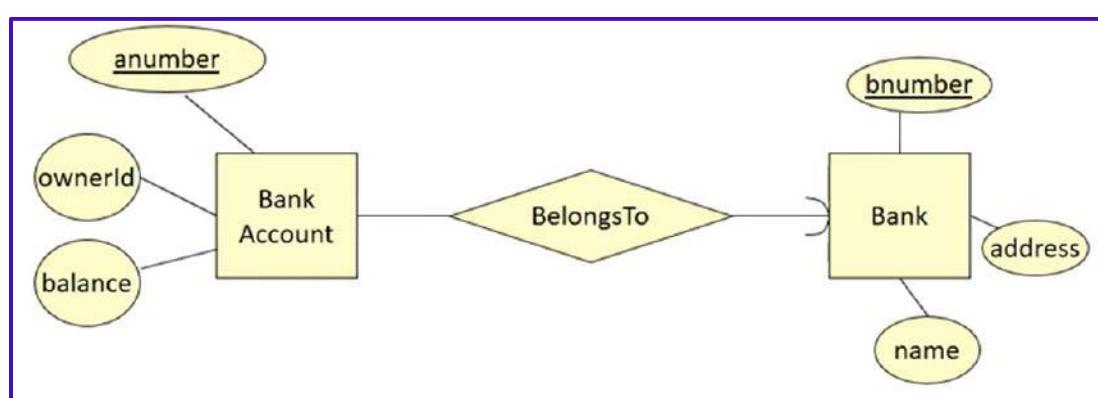
- דיאגרמה זו מנסה למדל את הקשר של הורות, אבל יש מספר בעיות:
- לא הצלחנו להראות באן ש-child הוא ילד של זוג אחד בדיק של הורים. חצים לא בדיק יעצרו פה, גם אם נשים חץ על Man ועל Woman זה יגיד למושל "לכל זוג Child,Woman יש לכל היוטר Man אחד מתאים" וזה לא נכון.
 - יש כאן חוסר זהות בין הקבוצות, כל woman היא child וגם כל man הוא child.

air נתכן?

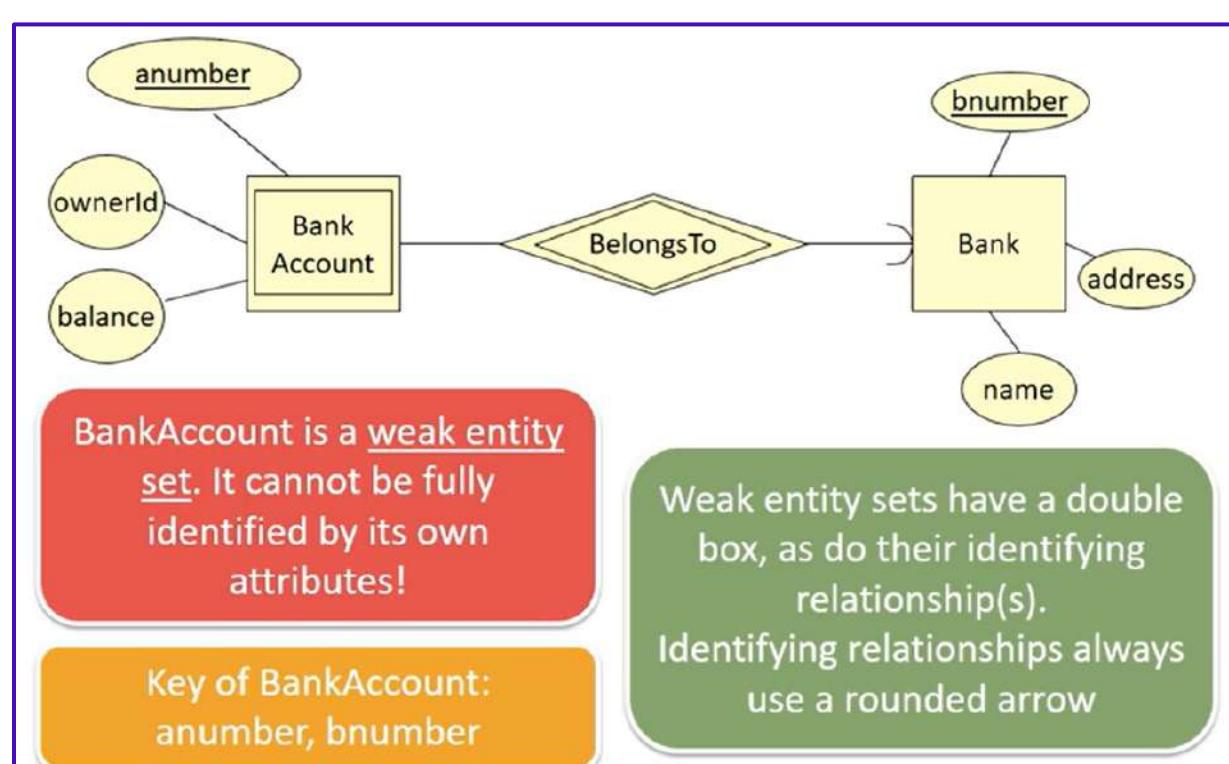


Weak Entity Sets 1.6

[דיאגרמה להמחשת הבעיה](#)

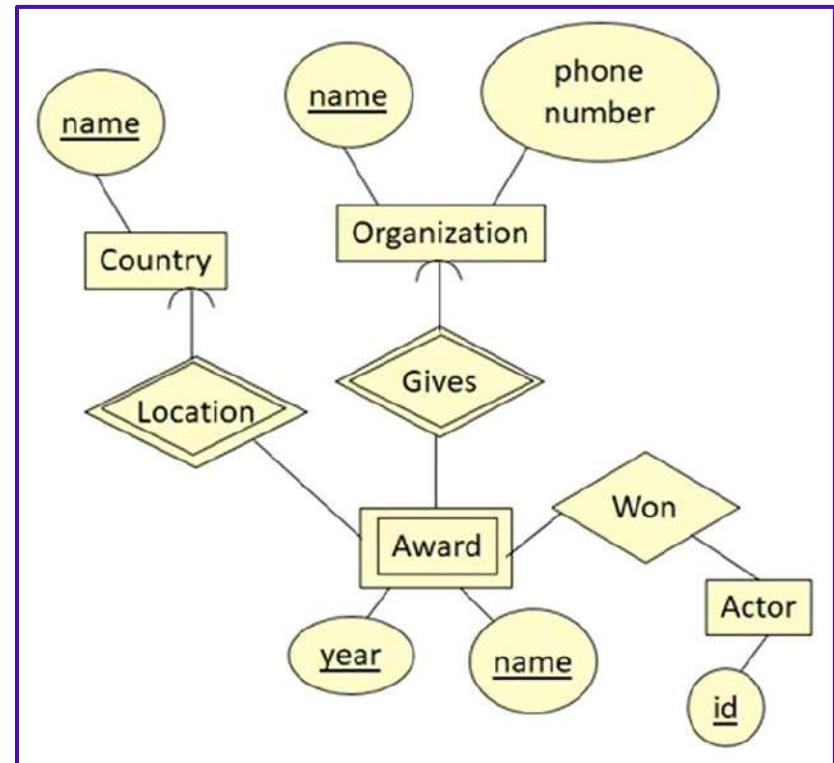


נבחן שבdíagramה המפתח של חשבון בנק הוא מסטר חשבון, והמפתח של בנק הוא מסטר הבנק.
בהינתן מסטר חשבון בנק בלבד, אני לא אוכל לדעת לשיך אותו גם לבנק, כי אני יודע רק להבטיח שבתוון אותו במסטר החשבון הוא ייחודי, אבל לא בין מספרי בנקים שונים. לכן, הדיאגרמה לא באמת משקפת את המציאות, כי anumber לא באמת מזיהה חשבון בנק, הוא מזיהה רק בצויר עם מסטר בנק.
לכן זו קבוצת ישות חרשה, בגלל שהמפתח שלה בלבד לא מספיק להזיהות אותה, אלא נדרש גם מפתח של קבוצת ישות נוספת היא משוכבת.



air נסמן קבוצת ישות חרשה? קודם כל, מרובע כפול סביב Bank Account, מצין קבוצת ישות חרשה.
מעוין כפול מסביב לקבוצת הקשרים, מצין את העובדה שקבוצת הקשרים הספציפית הזאת היא זו שבאמצעותה נזהה את חשבון הבנק.
תמיד יהיה חץ מעוגל כלפי קבוצת היחסות שמצויה אותה, ומציין את העובדה שככל חשבון בנק אמייתי בעולם שיר אך ורק לאחד.
כדי להזיהות קבוצת ישות חרשה, משתמש בצויר של המפתח שלה עם המפתח של קבוצת היחסות דרך מזיהם את היחסות החרשה.

יתכן מצב שבו ישות חלשה מזוהה באמצעות **שתי** קבוצות ישות נוספת ונוספות:



במקרה זה, המפתח של Award מורכב מ-4 תכונות: שם הפרס, שנת הפרס, שם המדינה בה ניתן הפרס, שם הארגון מתן הפרס.

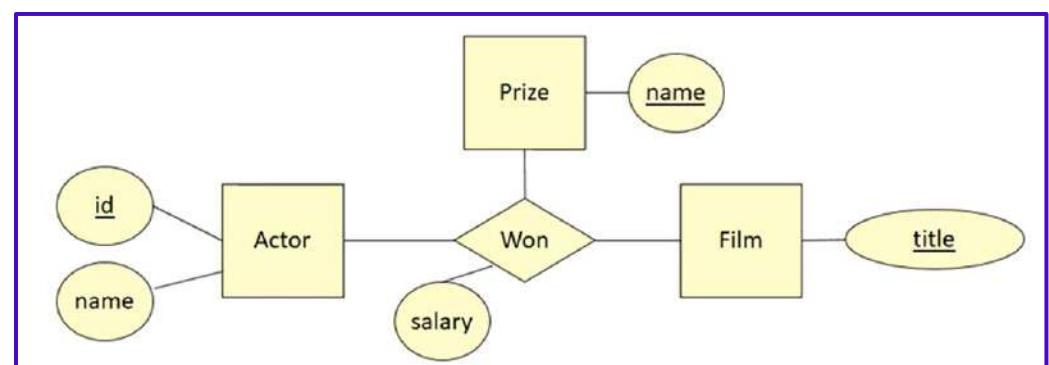
דוגמה מורכבת

נניח שהמטרה שלנו היא לשמר מידע על:

1. איזה שחקן שיחק באיזה סרט

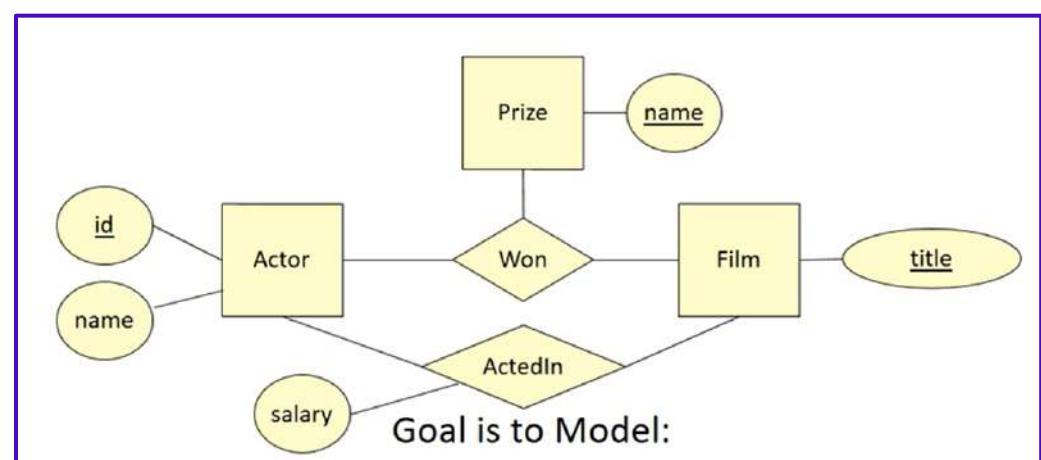
2. מי ניצח באיזה פרס עבור תפקידו בסרט

נסתכל באופציה זו:

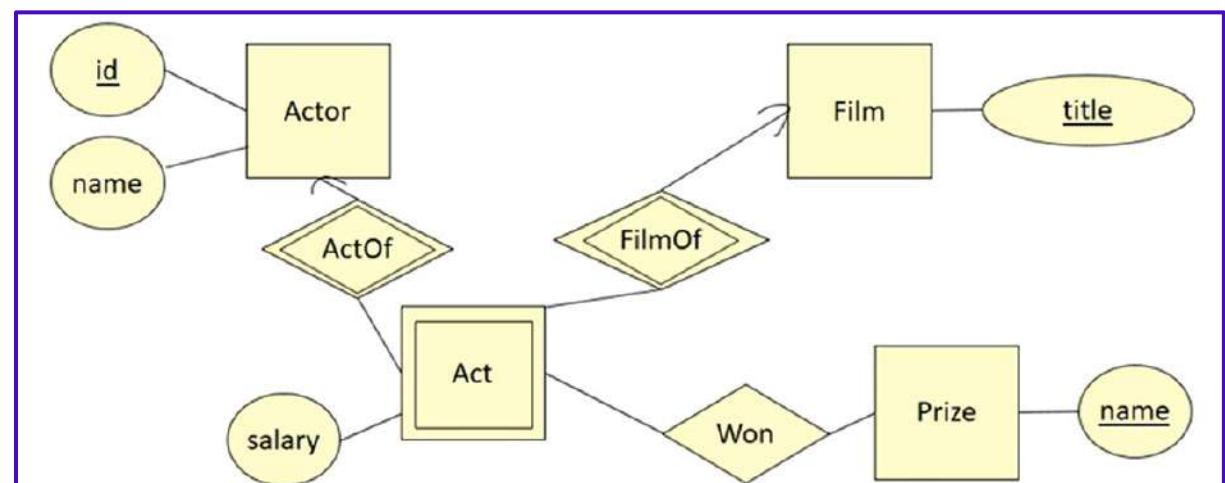


מהחר ש-Won שומר רק שלשות, אנחנו לא נצליח להשיג את מטרת (1), כי שחקן ששיחק בסרט אך לא זכה בשום פרס לא יופיע.

לכօරה הדיאגרמה הבאה מתקנת את הבעיה:



ואז יש לנו קבוצת קשרים עבור כל אחת משתי הדרישות. הבעיה היא שזה מידול לא תקין, כי אין שום קשר בין Won ל-ActedIn, כי אז נוכל בטיעות להציג למצב שבו יש שלשת Won של (שחקן, סרט, פרס) כאשר הזוג (שחקן, סרט) לא מופיע ב-ActedIn. איך נפתרו? עם קבוצת ישות נוספת ונוספות:



יכרנו ישות חלשה Act שמצויה על ידי שתי זוגות קשרים ActOf ו-Film. נבחן שבל Act מצויה על ידי המפתח `actor id & film title`. אותו Act יכול להשתתף בקבוצת קשרים `Won` שם יקשר לפרס. תנאי (1) נשמר באמצעות קבוצת היחסות החלה Act, תנאי (2) נשמר באמצעות קבוצת הקשרים (הרגילה) `Won`.

1.7 תרגום דיאגרמת קשרים

התכנון הלוגי

כיצד נعبر מdiagرام ישויות קשרים למבנה שאפשר לשמר בתוך מסד נתונים?

From ER Diagrams to DBMS

מתחילת בדיאגרמה, ממנה עוברים יצירה מבנים של יחסים (עליו מדובר בפרק זה), ובסיום מתרגמים אותן לפקודות בסינטקס של מערכת DB.

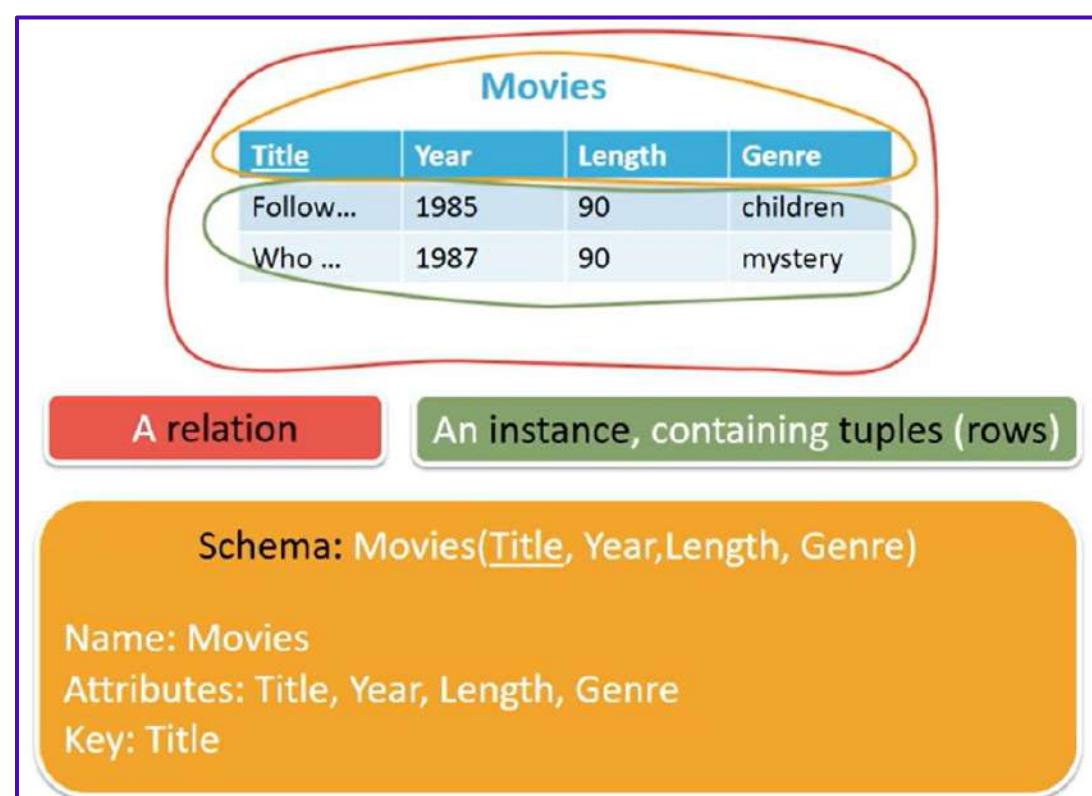
מודל רלוונטי

דרך "מפותחת" לדבר על טבלאות. מושגி בסיס:

יחס – זו טבלה, נקרא בה מຕור המושג מעולם הלוגיקה.

مفعע של היחס – אוסף השורות שנמצאות בתחום היחס. רישימת המפעעים שמקיימים את תוכנות היחס.

סבימה – אוסף העמודות של היחס. תיאור שם היחס והתכונות שלו. את מפתח היחס נציין עם קו תחתון.

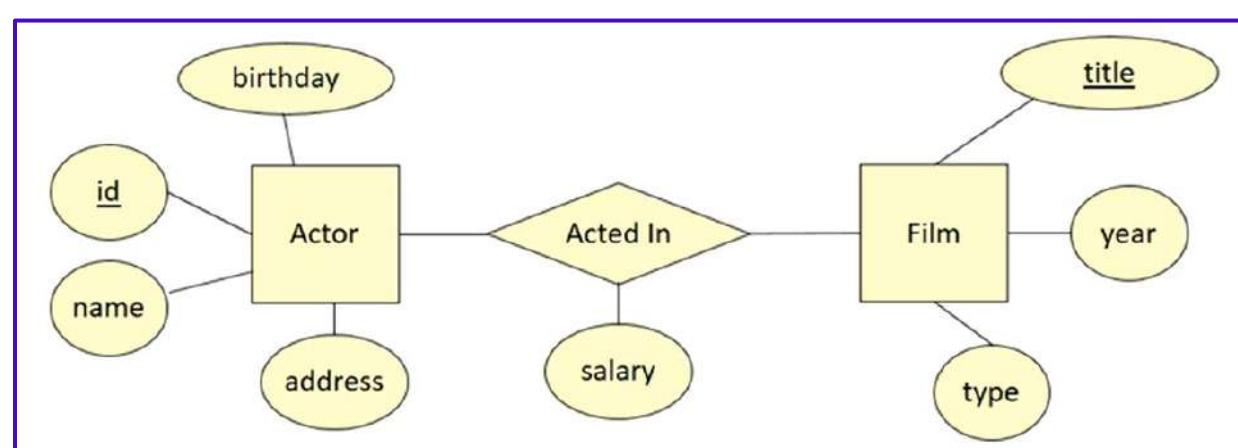


מפתח היחס – קבוצת תוכנות אם לא יכולים להיות 2 שורות שונות עם אותם הרכים עבור כל האטראיביטים של המפתח (כמו בדיאגרמה).

תרגום דיאגרמת ישויות קשרים לרלוונטי

בשנתרגם, נצורך להסתכל על בל אחד מהרכיבים של הדיאגרמה ולתרגם אותו. פעמים נסטה מהדיאגרמה אם יש לנו מידע נוסף שלא הצלחנו לקודד בתחום הדיאגרמה.

מה נתרגם? קבוצות ישויות, וקבוצת קשרים. איך נתרגם את הדיאגרמה הבאה?

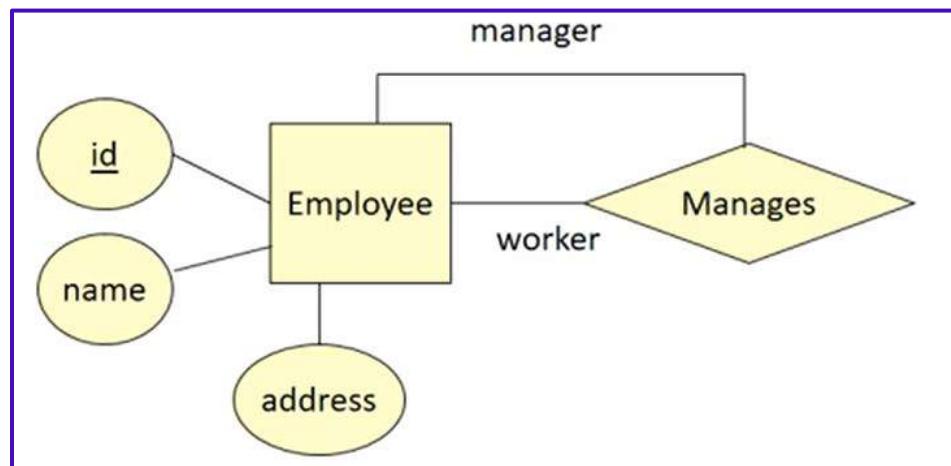


`Actor(birthday, id, name, address)`

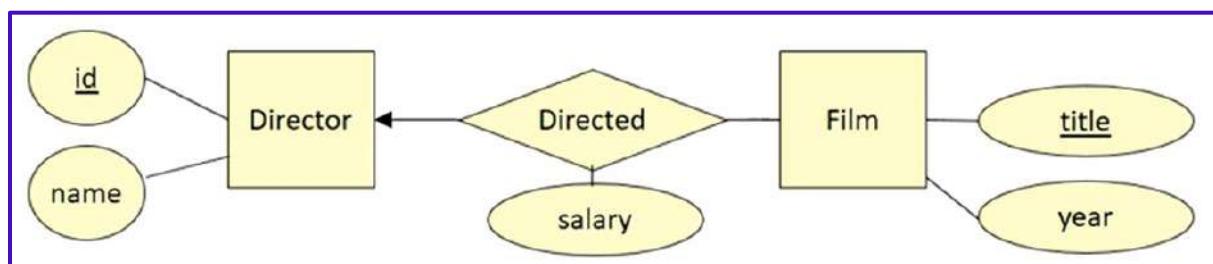
`Film(title, year, type)`

`ActedIn(id, title, salary)`

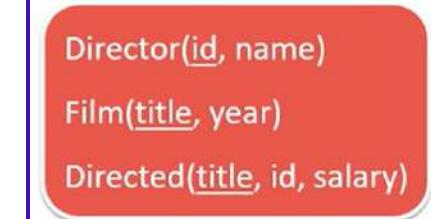
כלומר בקבוצת קשרים ניקח את המפתחות של כל קבוצות היחסות הקשורות אליו, ואת האטראיביטים שלו.

עבור Employee(id, name, address) נקבל: Employee(id, name, address)עבור Manages(managerId, workerId) נקבל: Manages(managerId, workerId)[Translating Relationships \(one-to-many\)](#)

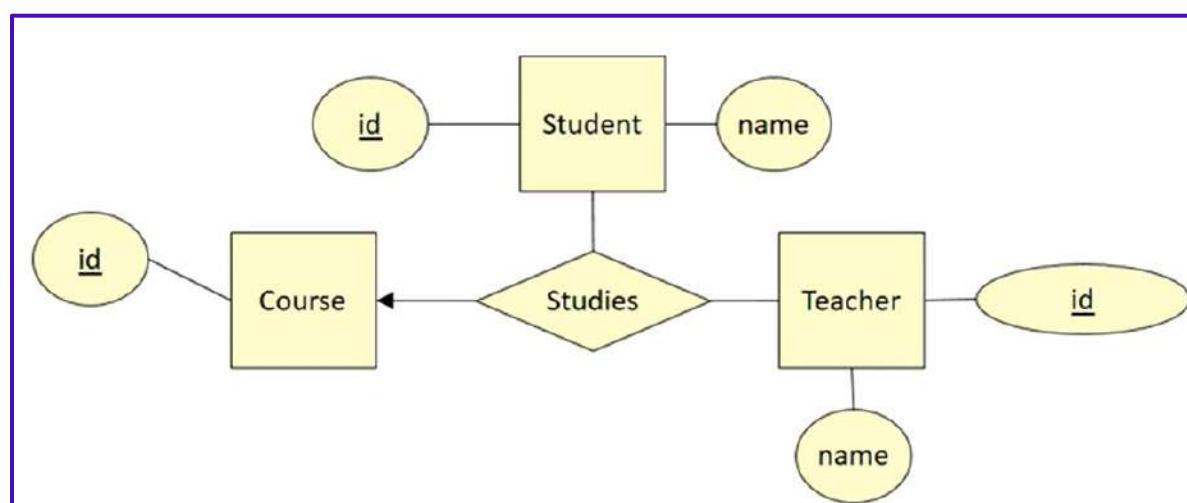
עבור הדיאגרמה הבאה:



נתחיל מלבנות את היחס בדוק במו תמייד, אבל השוני יהיה במפתח של Directed, שייקבע רק לפי Film. לכל סרט יש לפחות אחד במאי אחד, ולכן לא יתכן שתי שורות ביחס Directed עם אותו title.



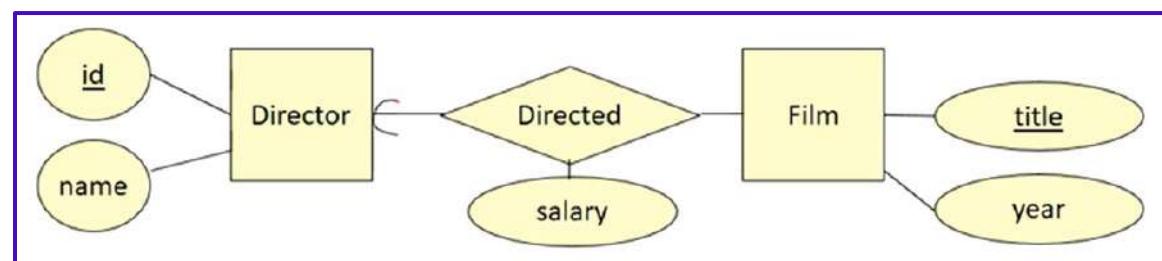
از השלבים לתרגם Directed כאשר יש גם חץ: קודם כל לרשום את כל המפתחות של מי שמקשור אליו, ועוד את התוכנות של הקשר עצמו. בסוף, בחירת המפתח תהיה לפי היחסות שמהן לא נבנה חץ.

[Trinary Relationship Set](#)במצב זהה כדי לתרגם את Studies:(cid, sid, tid) Studies(cid, sid, tid) בລומר הקשר Studies נקבע על פי ת.ז של סטודנט ומרצה.

אם נוסיף גם חץ לתוך Teacher, נקבל אופציה נוספת לתוך course, או course בغالל החץ Studies(cid, sid, tid) Studies(cid, sid, tid) בغالל החץ לתוך teacher.

טיפול: במקרים מסוימים בדיאגרמה היא כמות האופציות השונות שבנה אפשר לבחור!

מה עושים כאשר יש חץ מעוגל?

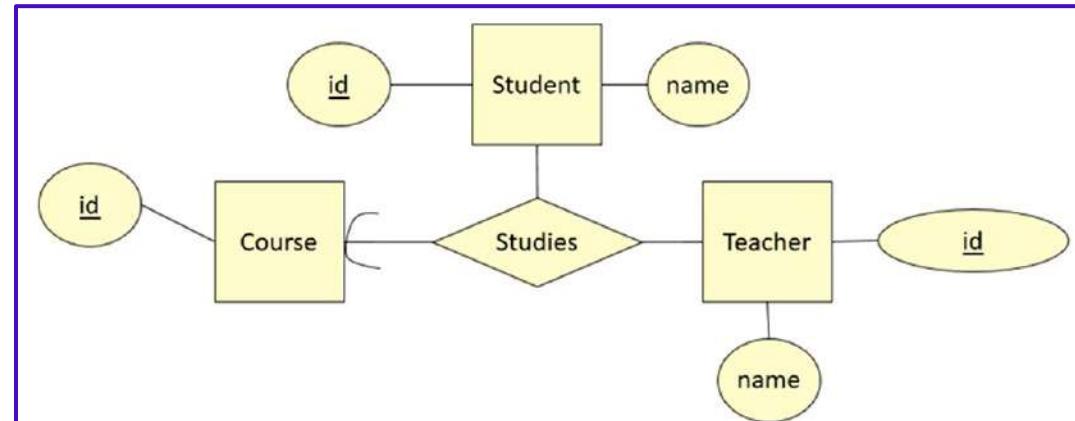


נידבר שזה אומר שלכל סרט יש בדיק במאו אחד. לכן את המידע על מי הבמאי של הסרט, אנחנו לא חייבים לשמר ביחס נפרד עבור **Directed**, אלא אפשר להוסיף אותו ישירות לתוך היחס של **Film** הבא:

Director(id, name)
Film(title, year, salary, id)

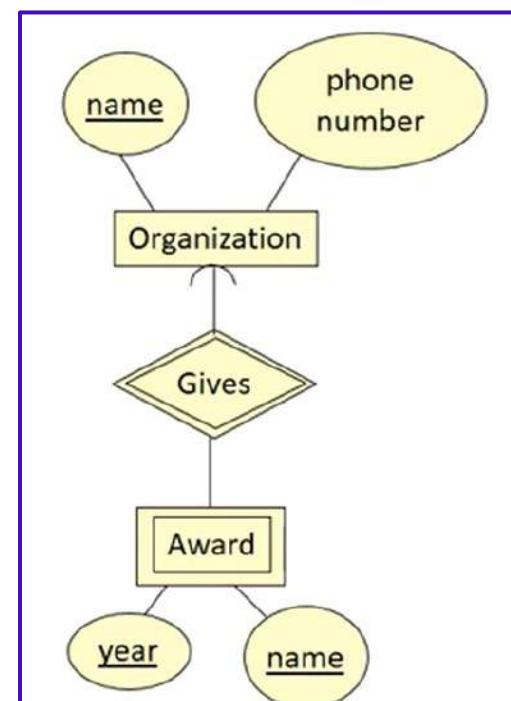
[הובצת קשרים טרינארית עם חץ מעוגל](#)

נתבונן בדיאגרמה הבאה:



כאן בשונה מהדוגמה הקודמת, אנחנו כבר לא יכולים לשמר את המידע על הקבוצת הקשרים בתוך אחד מקבוצות היחסיות, כי החץ המוגול כאן אומר שלכל צירוף (student, teacher) יש בדיק קורס אחד מתאים (בשונה מהמקרה הקודם). לכן ניצור במקרה זה 4 יחסים, ברגיל.

[תרגום קבוצת ישות שלשה](#)



מפתח היחס הוא המפתח שלו עצמו, בתוספת המפתח של קבוצת היחסות שדרךה אנחנו מזינים את היחסות בקבוצה.

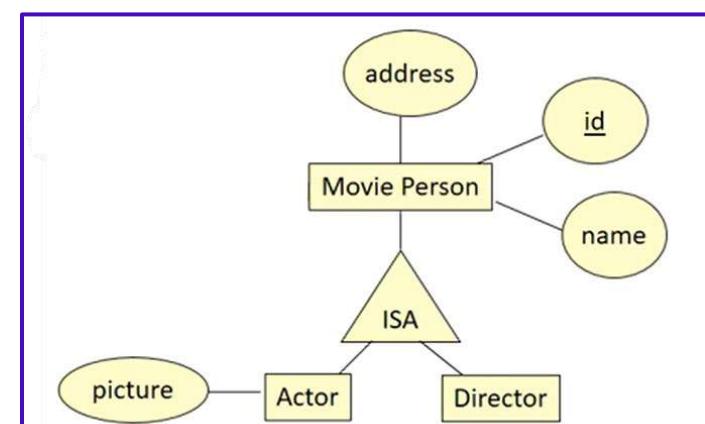
כאן ניצור יחס עבור **Award**, **Organization** (שמכיל גם את השם של הארגון שנוטן את הפרס), אבל **לא ניצור יחס** עבור קבוצת הקשרים Gives:

Organization(name, phoneNumber)

Award(name, year, oname)

[תרגום מרכיב הירושה](#)

ישנן 3 דרכים לתרגם את מרכיב הירושה בדיאגרמה. בפועל, נבחר בדרך שנראית לנו הבי מותאמת לסייעות.



אפשרות ראשונה E/R Style Conversion, מציעה ליצור יחס עבר או אחד מקבוצות היחסיות, ושות ספציפית יכולה בפועל שהמידע שלה יהיה מפוזר על כמה מהיחסים. אם נטבל על שחקן – המידע על השחקן (id, picture) יהיה בשדה Actor, והכותרת והשם בטבלה של MoviePerson. כלומר, לקבוצות ישויות שיורשות ניתן את התכונות שלהן, וגם את המפתח (של קבוצת היחסיות ממנה יוצרו).

MoviePerson(<u>id</u> , address, name)
Actor(<u>id</u> , picture)
Director(<u>id</u>)

גישה נוספת היא Object Oriented, שבה ניתן יחס עבר כל קומבינציה אפשרית של ישויות בתוך הקבוצות:

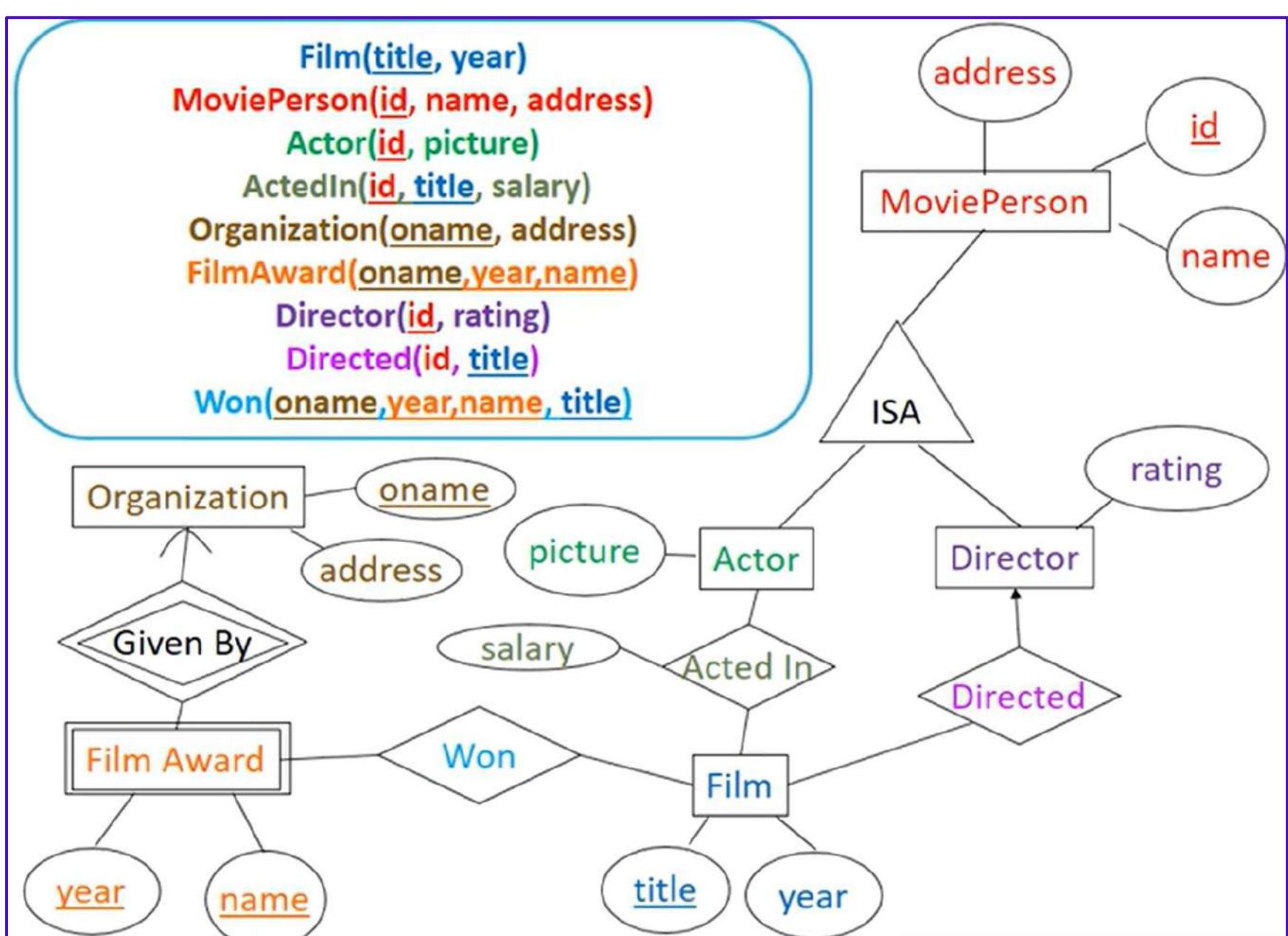
MoviePerson(<u>id</u> , address, name)
MoviePersonActor(<u>id</u> , address, name, picture)
MoviePersonDirector(<u>id</u> , address, name)
MoviePersonActorDirector(<u>id</u> , address, name, picture)

* אם יש לנו מידע נוסף, כמו למשל שלא יתרון MoviePerson שהוא גם במאי וגם שחקן, זה יכול להשפיע על בנות היחסים.

גישה אחרת היא גישת Null Value Approach
יחס אחד שמכיל את כל התכונות, ואם יש לנו ישות שחשר לה חלק מהתכונות (כמו director בלי תמונה) שם יהיה ערך Null.

MoviePerson(<u>id</u> , address, name, picture)
--

[שאוף מסכם עם דוגמאות](#)



הסביר מלא על כל היחסים נמצא בשבוע 1 סרטון 6, מתחילה ב-18:50.

Week 2: Relational Algebra

SQL 1: Creating Tables 2.1

פקודות לייצור טבלאות

כדי ליצור טבלה חדשה משתמשים בפקודה CREATE TABLE בצויר שם הטבלה. בסוגרים נוסף שורה שמתאר את שמות העמודות:

```
CREATE TABLE TableName(  
    Column1 DataType1 ColConstraint, ...  
    ColumnN DataTypeN ColConstraint,  
    TableConstraint1, ...  
    TableConstraintM  
)
```

אם יש אילוצים על הטבלה כולה, נשים אותם בסוף (מוסיע בורוד). נקודות חשובות:

- כל פקודה מסתיימת ב- ;
- SQL היא שפה case insensitive, אין משמעות ל-case .

דוגמה

```
CREATE TABLE Employee(  
    ID      INTEGER NOT NULL,  
    Fname   VARCHAR(20),  
    Lname   VARCHAR(20),  
    Gender  CHAR(1),  
    Salary  INTEGER NOT NULL,  
    Dept    INTEGER  
)
```

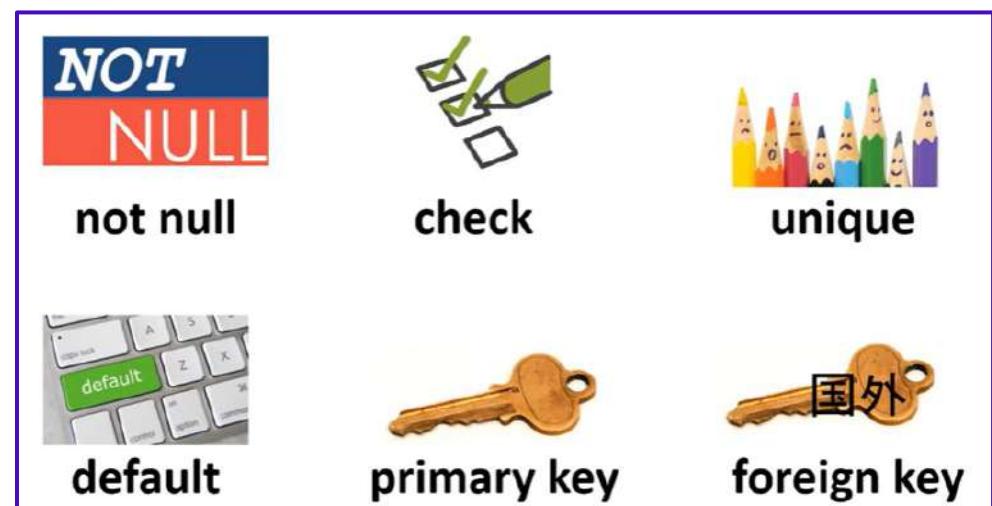
.string VARCHAR(20) – זה string שיכל לקבל כל אורך עד גודל 20.

.CHAR(1) – זה תו, בדיק באורך שצינו (לא כמו לעלה שזה בטוח עד 20).

אילוצים בפקודת CREATE TABLE

אם נכניס אילוצים לטבלה, כל פעולה שיצרת ניגוד לאילוץ זהה תיחסם על ידי מערכת DB, لكن נרצה לשימוש כמה שיותר אילוצים כדי למנוע טעויות. זה כן גורר שפעולות insert או update יהיו מעט איטיות יותר, כי צריך תמיד לוודא שהאילוצים מתקבים.

בעולם האמיתי, עדין נרצה לשימוש הרבה אילוצים אלא אם כן אנחנו מגיעים למצב שבו יש צוואר בקבוק שנגרם בעקבות האילוצים. סוג אילוצים:



NOT NULL

השדה לא יכול לקבל ערך ריק.

default(value)

נכבל לתת ערך דיפולטיבי לשדה, ואז אם לא נציין במפורש מה השדה – ייכנס הערך הדיפולטיבי.

Check Constraints

אילוץ שמודד את התנאי שהכניסנו, למשל בדוגמה הבאה:

```

CREATE TABLE Employee(
    ID      INTEGER primary key,
    Fname   VARCHAR(20),
    Lname   VARCHAR(20),
    ● Gender CHAR(1) CHECK(gender='F' or
                           gender='M'),
    Salary  INTEGER NOT NULL,
    DeptNumber INTEGER
    ● CHECK (Gender = 'M' or Salary > 10000)
);

```

אפשר להגדיר אילוצי בדיקה על שדה מסוים (בצהוב) או על הטבלה כולה (בירוק).

אילוץ בולאיוני שנבדק על ערך של attribute או על ערך של שורה.

Unique

אילוץ שמנדרים לשורה בטבלה שמנדר שלא יכולות להיות שתי שורות בטבלה עם אותו ערך בעמודה זו. הגיוני לשימוש המילה UNIQUE למושל על שורה של ת.ז. אפשר גם לשימוש אילוץ על טבלה באופן הבא:

```

CREATE TABLE Employee(
    ID      INTEGER NOT NULL,
    Fname   VARCHAR(20),
    Lname   VARCHAR(20),
    Gender  CHAR(1) default('F'),
    Salary  INTEGER NOT NULL,
    Dept    INTEGER,
    UNIQUE(FNAME,LNAME)
);

```

זה אילוץ בטבלה על הזוג שם פרטי ושם משפחה, שאומר שיכולים להיות שני אנשים עם אותו שם פרטי, שני אנשים עם אותו שם משפחה – אבל לא יכולים להיות 2 עובדים עם אותו שם פרטי ושם משפחה.

Primary Key

זה מה שהוגדר במבנה בדיאגרמת ER. כאשר עמודה או זוג עמודות ייחד הן primary key הן יוכלו להיות ערך NULL והן UNIQUE. למעשה זה שקול לסימן העמודה בזוג זהה. מבנה הנתונים ייצור גישה יעילה לשדה הזה, כי הוא מניה שנדרצה לגשת אליו הרבה. יש שתי דרכים לעשות זאת, אילוץ על עמודה (ミミン) או באילוץ על הטבלה (משמאל):

```

CREATE TABLE Employee(
    ID      INTEGER,
    Fname   VARCHAR(20),
    Lname   VARCHAR(20),
    Gender  CHAR(1),
    Salary  INTEGER NOT NULL,
    Dept    INTEGER,
    PRIMARY KEY(ID)
);

```

```

CREATE TABLE Employee(
    ID      INTEGER PRIMARY KEY,
    Fname   VARCHAR(20),
    Lname   VARCHAR(20),
    Gender  CHAR(1),
    Salary  INTEGER NOT NULL,
    Dept    INTEGER,
    UNIQUE(FNAME,LNAME)
);

```

Foreign Key

אינטואיציה: לעיתים נרצה לשימוש בטבלה שלנו ערך שמצויר בדיקה של קיום ערך בטבלה אחרת (למשל אם יש טבלה של Employee שבה נרצה לשימוש מחלקה של העובד, ויש טבלה Department שבה יש מספרי מחלקות במבנה – נרצה לוודא שיבשומנו מספר מחלקה לעובד, זה אכן מספר מחלקה שקיים בטבלה Department). נעשה זאת כך:

```

CREATE TABLE Employee(
    ID      INTEGER PRIMARY KEY,
    Fname   VARCHAR(20),
    Lname   VARCHAR(20),
    Gender  CHAR(1) DEFAULT('F'),
    Salary  INTEGER NOT NULL,
    DeptNumber INTEGER,
    FOREIGN KEY (DeptNumber)
                REFERENCES Department(DeptNumber)
);

```

במילים אחרות: "כל מה שמופיע אצלנו בשדה DeptNumber צריך להופיע (להצביע על) שדה DeptNumber בטבלה Department".

כדי שיהיה מותר לנו להגיד אותו, השדה שאחנו מביצים אליו בטבלה השנייה יהיה או UNIQUE או Primary Key.

דרך לקרוא את השורה שמסומנת באדום:

"The field **DeptNumber** in **Employee** is a foreign key that references the field **DeptNumber** in **Department**"

כל ערך שהוא לא null בשדה DeptNumber בטבלה Employee חייב להופיע בשדה DeptNumber בטבלה Department.

מה יקרה אם ננסה למחוק מחלקה בטבלה Department שיש עובדים בטבלה Employee המשיכים למחלקה זו?

אם לא הגדרנו אחרת, לא יוכל למחוק את המחלקה זו אם יש עובדים המשיכים לה. זה יגרור שגיאה, כי הפעולה סותרת אילוץ. באופן כללי, זה כלל אכבע של ה-system database לכל פעולה שסותרת אילוץ.

יכולנו להוסיף בסוף השורה **ON DELETE CASCADE** ואז אם נמחק את המחלקה זו, כל עובד שמחובר למחלקה יימחק גם כן.

מה קורה אם יש תלות Foreign Key בין שתי טבלאות בזורה ציקלית?

זה יגרום לכך שלא יוכל להכניס נתונים, כי בעת הכנסה לטבלה אחת נדרש שיהיה ערך מתאים בטבלה השנייה, אבל כדי להכניס את הערך בטבלה השנייה צריך לטבלת הראשונה. יש לה פתרון שיאפשר שבאופן רגעי האילוצים יהיו מופרים, נראה אותו בהמשך הקורס במנגנון הטרנדקציה.

מחיקת טבלה

כדי למחוק את הטבלה TableName נכתב `DROP TABLE TableName;`

אם יש אילוצי מפתח זר, צריך לשימוש לבסדר מחיקת הטבלאות כי אנחנו עלולים להפר אילוץ מפתח זה. כדי להיפטר מהבעיה זו אנחנו יכולים לכתוב את הפקודה כך: `DROP TABLE TableName cascade;`. במקרה זה, כל אילוץ מפתח זר שהצביע על הטבלה שמחקנו, ימחק את **הイルוץ** הזה במקביל למחיקת הטבלה. כלומר, בשונה ממחיקת שורה בטבלה שמקושרת לטבלה אחרת – במקרה לא ימחק שורות שהו מקשורות לטבלה, אלא ימחק את העמודה שהייתה מקושרת לטבלה.

הכנסת שורות לטבלה

מתבצע באמצעות: `INSERT INTO TableName(column1,...,columnN) VALUES (val1,...,valN)`

אפשר גם בלי לציין את שמות העמודות, כאשר אנחנו מכניםם לפי סדר העמודות שצינו בעת יצירת הטבלה.

אפשר גם להוסיף שורה ולציין רק חלק מהערכים אותם נרצה למלא. במקרה זה שאר הערכים יוגדרו להיות NULL. כמובן, אם יש עמודה בטבלה שהוגדרה כך שלא מקבלת את הערך NULL, נקבל שגיאה. אם בעמודה הוגדר ערך דיפולטיבי, השורה תקבל את ערך זה.

דוגמיה:

Sailors			
sid	sname	rating	age
22	Alice	7	45
31	Barbara	8	55.5
58	Carol	10	35
70	Alice	10	25

? `INSERT INTO Sailors (sname, rating)
VALUES ('Danna',10)`

מה יקרה בכך? נשים לב ש-sid הוא **מפתח**, כלומר null NOT + UNIQUE ולכן פקודה זו לא תאפשר執行.

יש גם דרכים להכניס שורות רבות תוך טבלה באמצעות Bulk Loader.

RA1: Projection, Selection 2.2

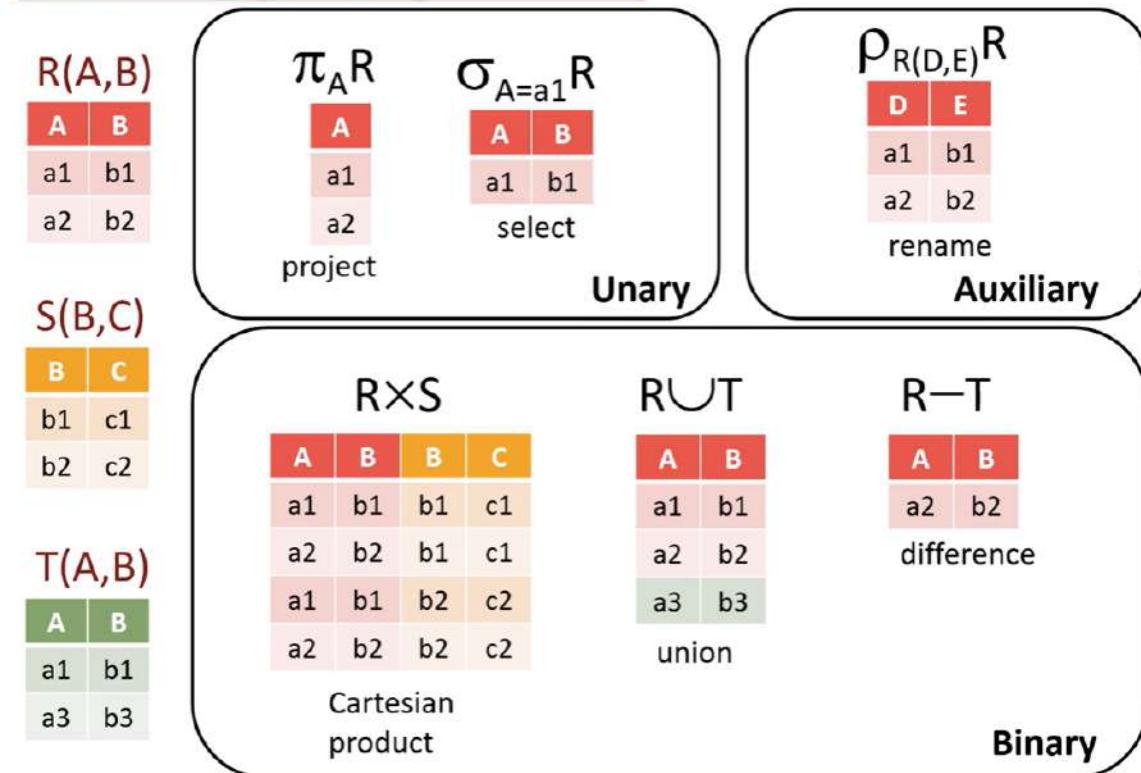
המודול הרלציוני

רלציה היא **קובוצה** של שורות, כלומר אין שורות זרות, ונכון שאין ערכי null. לאחר שמדובר בקובוצה, סדר השורות לא משנה. כמו כן, גם סדר העמודות לא משנה. אפשר עבשו גם לדבר על הפעלת פעולות אלגבריות מעל היחסים, שייצרו לנו יחס חדש.

קשר בין אלגברה רלציונית ל-SQL

שפה שמקבלת טבלאות ומוציאת טבלאות. הסיבה שאלגברה רלציונית היא חשובה זה מאחר שבבסיס נתונים אמייתי אנחנו מתחילה עם טבלאות, שאליתת SQL מוגדרת עליון ומאתורי הקלעים מתרגמת לביטויinalgברה רלציונית שעובר אופטימיזציה של המערכת, הוא מחושב ומוחזר בטבלה. כדי להבין את התהליך כמו שצריך, חייב להבין אלגברה רלציונית.

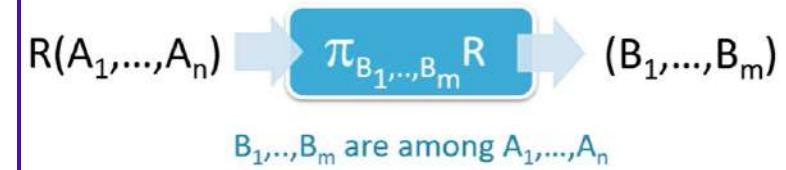
Summary of Operators



יחסים אונאריים – Unary

1. הטלה

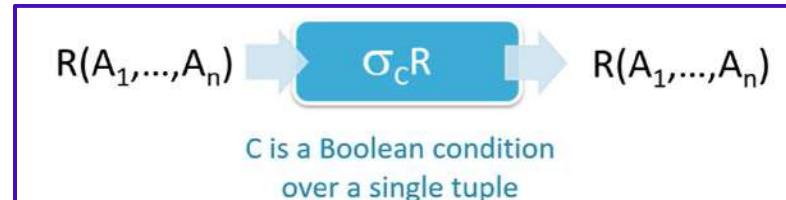
פעולת הטללה מקבלת יחס עם n עמודות. ליד פעולה הטללה π נציין בתת-אינדקס את שמות העמודות שאחננו חיצים להטיל עליהם, ובהתוצאה קיבל יחס שיש לו את אותן עמודות שהזכרנו. הפעולה מחזירה את אותו יחס, אבל רק את העמודות שהזכרנו:



נשים לב שגם אחרי הטללה קיבל שורות זהות (כיו לפניה הטללה שורות נבדלו בערכים שבחרנו לא לחת), אך אחרי הטללה קיבל רק את השורות הייחודיות, וכפיפות ידו כי מדובר ביחס.

2. בחירה

פעולת הבחירה מקבלת יחס R ורושמים אותה כ- σ עם תת-אינדקס שמתאר תנאי בוליאני מעל שורה בודדת. התוצאה של השאלה היא אותן שורות ב- R ubeון התנאי הבוליאני C מתקיים.

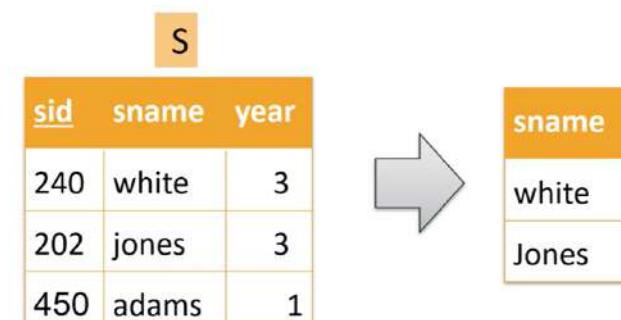


למשל אם העמודות שלמו הן sid, course, tid, id, sid, course, tid ומבצע רק את אלה שבהם tid=20 אז המורה הוא 20 נקבע: $R_{tid=20}$. התנאים בבחירה יכולים להיות מורכבים ולבול השוואות $\neq, =, <, >$, אבל תמיד יתיחסו לשורה אחת בכל פעם

דוגמא לשילוב בין פעולה בחירה להטלה:

באשר נתונה פעולה כמו $(\sigma_{course=tid=20} R) \pi$ החישוב צריך להיות מבוצע מבפנים החוצה.

שאלה: איך נמצא את השמות של כל התלמידים בשנה השלישי?



הפעולה המתאימה היא $\pi_{sname}(\sigma_{year=3} S)$

מה יחזיר כאן?

$$\pi_{\text{sid}} (\sigma_{\text{sname}=\text{jones}' \wedge (\text{year} = 1 \vee \text{year} = 3)} (S))$$

	sid	sname	year
S	240	white	3
	202	jones	3
	450	jones	1

ת החזר הテーブל:

sid
202
450

כיוון קודם הסרנו את השורה הראשונה בגלל שם הסטודנט, אז הטלים לפי העמודה sid.

RA2: Union, Difference, Cartesian Product 2.3

[יחסים ביןaries – Binary](#)

3. איחוד

פעולה שמקבלת שני יחסי סכימה עם אותה סכימה (עמודות), ומחזירה את איחוד השורות שלהן. אם יש שורות זרות, הן יופיעו בתוצאה רק פעם אחת.



4. נטול

בדומה לפעולה האיחוד, אבל אם יש 2 טבלאות G,S ונרצה את G-S זה יחזיר את השורות שנמצאות ב-S אבל לא נמצאות ב-G.



דוגמה

נתון יחסי, איך נציג את sid של סטודנטים שלא לומדים את הקורס db?

tid	sid	course
20	202	os
10	450	calculus
20	202	db
20	240	db

לכוארה נראה שפתרון יהיה $\pi_{\text{sid}} (\sigma_{\text{course} \neq \text{db}} R)$ אבל נשים לב שהוא יוריד את השורה השלישית והרביעית, ואז ייתן לנו את 202 ו-450. 202 הוא סטודנט שבן לומד db. נכון: $\pi_{\text{sid}} (\sigma_{\text{course} = \text{db}} R)$. לקחנו תז' של כל הסטודנטים, והודנו מהן את תז' של אלו שŁומדים db.

5. מכפלה קרטזית

בהתאם יחסי עם עמודות כמו בתמונה למטה, במכפלה הקרטזית נקבל שורות שיש להן $n + m$ עמודות. בתוכן נקבל את כל הזוגיות האפשריות של שורות מ-R עם שורות מ-S. אם יש שמות עמודות שונים שמופיעים גם ב-R וגם ב-S נוכל להשתמש בנותיציות R.A, S.A. מספר הטעafilim בתוצאה הוא תמיד התוצאה של מספר הטעafilim האפשריים ב-R וב-S. למשל אם ב-R 4 שורות וב-S 2 שורות, בתוצאה נקבל 8 שורות. נבחן האם מהטבלאות ריקה (בלי שורות) נקבל בטבלת התוצאה 0 תוצאות.

דוגמה לשימוש במכפלה קרטזית עם הטלה ובחירה:

$$\pi_{\text{course}}(\sigma_{R.\text{tid} = T.\text{tid} \wedge \text{tname} = 'levy'} (R \times T))$$

(tid)	sid	course	(tid)	tname	dept
20	202	os	10	cohen	math
10	450	calculus	10	cohen	math
20	202	db	10	cohen	math
20	240	db	10	cohen	math
20	202	os	20	levy	cs
10	450	calculus	20	levy	cs
20	202	db	20	levy	cs
20	240	db	20	levy	cs

ביצענו קודם מכפלה קרטזית על היחסים R ו-S, רצינו את השורות שבנה $R.\text{tid} = T.\text{tid}$ וגם שם המרצה הוא לוי, זה מביא לנו את השורות שמסומנות באדום. לבסוף, נרצה לבצע הטלה על שם הקורס ונקבל את הטבלה הקטנה בהצד.

עד כאן אלה 5 האופרטורים החשובים.

6. אופרטור שיבוי שם – Renaming

מקבל יחס, וונתן לו שם חדש, סכימה חדשה. שימושי לשבירת ביוטי ארוך לביטויים קצרים יותר, לתת שם לתוצאות ביניים. כמו כן, זה יפתרו קונפליקטים בשמות תכונות.

$$R(A_1, \dots, A_n) \xrightarrow{\rho_{S(B_1, \dots, B_n)}} R \xrightarrow{\rho_{S(B_1, \dots, B_n)}} S(B_1, \dots, B_n)$$

דוגמה

$$\pi_{sn1,sn2}(\sigma_{y1=y2 \wedge si1 \neq si2} (\rho_{R(si1,sn1,y1,si2,sn2,y2)} (S \times S)))$$

S	sid	sname	year	sn1	sn2
	240	white	3	white	jones
	202	jones	3	jones	white
	450	adams	1		

מה השאלה עשויה? קודם כל, מכפלה קרטזית של S עם עצמו. יהיו לנו 9 שורות בתוצאה, ונקבל 6 עמודות כל הזמן עם אותו השם. בשביל זה נשתמש בשינוי שם ב� שבסמוך העמודות: sid, sname, year, sid, sname, year, sid, sname, year נקבע: y1, y2, si1, sn1, si2, sn2, y1, y2, si1, sn1, si2, sn2, y1, y2. אחר ב�, תנאי בחירה לאלה שלומדים באותה שנה, ולא מזמין באותו סטודנט. לבסוף, לבצע הטלה על שמות הסטודנטים. זה למשל עבור דרך לצוות סטודנטים באותה שנה. אפשר גם לחלק ל-2 חלקים באופן הבא עם שינוי שם:

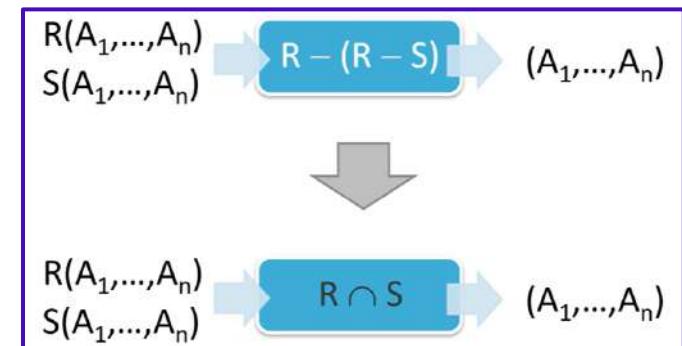
$$\begin{aligned} &\rho_{R(si1,sn1,y1,si2,sn2,y2)} (S \times S) \\ &\pi_{sn1,sn2}(\sigma_{y1=y2 \wedge si1 \neq si2} (R)) \end{aligned}$$

אופרטורים נוספים

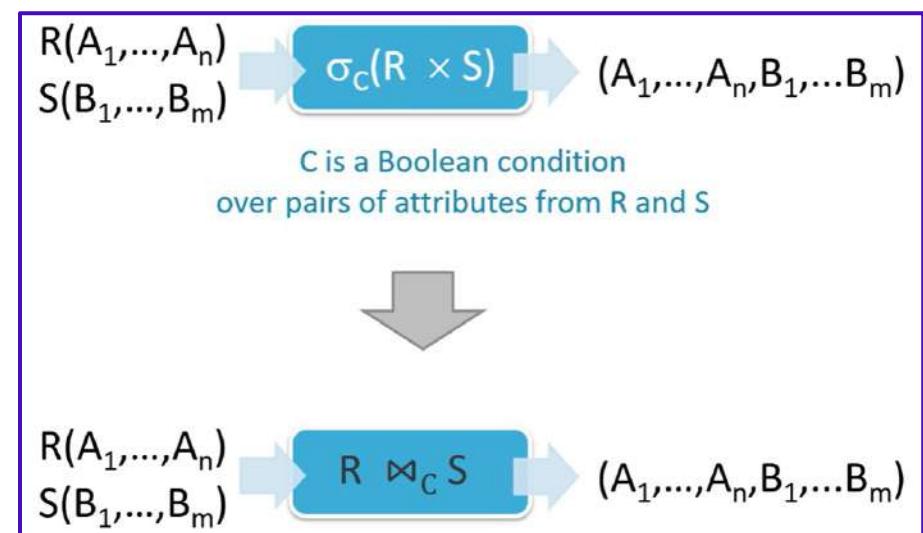
נראה בעת כמה אופרטורים שהם "סוכר סינטקטי", כלומר אנחנו לא זוקקים להם, אפשר להסתדר עם 5 הפעולות הבסיסיות. אבל יש פעולות שאנו צריכים עושים שוב ושוב, אז יצרו להם סינטקס מוקוצר.

1. חיתוך – Intersection – Intersection

אפשר לבטא את $(R - S) \cap R$ באמצעות $S \cap R$. גם כאן, אפשר לבצע את הפעולה רק אם לשני היחסים יש את אותן עמודות (אותה סכימה). התוצאה תהיה כל השורות שモופיעות גם ב-R וגם ב-S.

2. צירוף על תנאי – Join – Conditional Join

לפעמים נרצה להכפיל שני יחסים זה בזה (מכפלה קרטזית) ואז להתנוט את התוצאה.

דוגמה

למשל אם נרצה למצוא יחסים הבאים את כל המרצים שמלמדים את סטודנט 202:

R	<u>tid</u>	<u>sid</u>	course
20	202		os
10	450		calculus
20	202		db
20	240		db

T	<u>tid</u>	tname	dept
10	cohen	math	
20	levy	cs	

ונכל לרשום זאת בשני אופנים, הפעם השנייה משתמשת באיחוד על תנאי:

$$\pi_{\text{tname}}(R \bowtie_{R.\text{tid} = T.\text{tid} \text{ and } \text{sid} = '202'} T) \quad \pi_{\text{tname}}(\sigma_{R.\text{tid} = T.\text{tid} \text{ and } \text{sid} = '202'}(R \times T))$$

3. צירוף טבעי – Natural Join – Natural Join

כאן אנחנו ממעוניינים להכפיל את R עם S ושהתנאי שנדרש יהיה שוויון על כל האтриビוטים המשותפים.



למעשה **מסתורות בסימן זה 3 פעולות**: מכפלה קרטזית על R עם S, בחירת השורות של R-S עם אותו ערך על אטריבוטים משותפים, ולבסוף הטלה כדי שנשאר עם העתק אחד ולא סט כפול של עמודות. הפעולה זו אסוציאטיבית וקומוטטיבית, אך אפשר לשרר אותה ללא סוגרים. עבשו אפשר לרשום את השאלה מהלכה מוקדם כך:

$$\pi_{\text{tname}}(\sigma_{\text{sid} = '202'}(R \bowtie T))$$

לשימים לב שהתנאי שבו אנחנו מעוניינים להסתבל על סטודנט 202 לא נתפס בצירוף הטבעי, צריך לציין אותו מפורש. אפשר היה גם קודם את סטודנט 202 ביחס R ואז לבצע צירוף טבעי, זה שקול למה שעשינו קודם:

$$\pi_{\text{tname}}(\sigma_{\text{sid} = '202'}(R) \bowtie T)$$

דוגמה לצירוף טבעי

R			S		
A	B	C	B	C	D
1	2	3	2	3	5
1	2	4	2	3	6
2	3	3	3	3	1

R ו-S יש שתי עמודות משותפות. אם נרצה לעשות ביניהם צירוף טבעי, למשל עבור השורה הראשונה נדרוש שגם $B=2$ וגם $C=3$ (יש קווים בין העמודות המתאימות). לבן ייחס התוצאה יהיה:

$$R \bowtie S$$

A	B	C	D
1	2	3	5
1	2	3	6
2	3	3	1

(העתק אחד של B והעתק אחד של C)

מהו גודל תוצאת הצירוף הטבעי?

נכיח של-R יש n שורות, ול-T יש m שורות.

- 1) כמה שורות **לכל היותר** יש ביחס התוצאה? $m \cdot n$, כי יכול להיות שיש עמודה משותפת עם בדיק אותה ערך לכל אורך הטבלה.
- 2) כמה שורות **לכל הפחות** יש ביחס התוצאה? 0, כי יכול להיות שאף שורה ב-R לא שווה לאף שורה ב-T על האטראיביטים המשותפים.
- 3) מה קורה **אם הסכמתות זהות?** מבצעים מכפלה קרטזית ומשארים רק את השורות שבנה יש שוויון על כל האטראיביטים, והעתק אחד על כל עמודה, لكن נקבל את $T \cap R$.
- 4) מה קורה **אם אין אטראיביטים משותפים?** ההגדרה של צירוף טבעי הוא מכפלה קרטזית ואז שוויון על כל העמודות המשותפות. אם אין עמודות משותפות, אז יש תמיד שוויון. לכן זה שקול ל- $T \times R$.

RA4: Division 2.54. אופרטור חילוק Division

באופרטור זה יש תנאי מיוחד על הסכמתות על מנת שייהי ניתן להשתמש בו. בהינתן יחסים R, S אם נרצה לבצע $S \div R$ אז הסכימה של S צריכה להיות מוכבלת בסכימה של R. מה ש意義ר בתוצאה הם האטראיביטים שנמצאים ב-R ולא נמצאים ב-S. השורות בתוצאה יכולים את אוסף השורות עם ערכי a_1, \dots, a_n אך שלכל (a_1, \dots, a_n) ב- S , (b_1, \dots, b_m) ב- R (a_i, b_j)

דוגמה שנועדה להסביר את השורות בתוצאה

R			S
A	B	C	C
1	2	3	3
1	2	4	
2	3	3	

חלוקת מוגדרת כי הסכימה של S מוכבלת בסכימה של R. עמודות התוצאה יהיו A,B,C כי הם אלו שלא נמצאים ב-S. מה לגבי השורות? אלו שMOVEDות ביחד עם כל אחד מערכיו S. יש רק את הערך 3, לכן נקבל את כל השורות שיש להן הרחבה עם הערך 3:

$$R \div S$$

A	B
1	2
2	3

אם בטבלה S יהיה בעמודה C גם את הערך 4, נקבל את כל השורות של R, עם העמודות A,B.

דוגמה נוספת

R			S	
A	B	C	A	C
1	2	3	1	3
1	2	4	1	4
2	3	3		

עבשו נקבל ביחס התוצאה רק את העמודה B. אילו שורות נקבל את כל אלה שיש להן המשך ל-3, או ל-1,4 (בעמודות C,A בהתאם). لكن התוצאות האפשריות הן רק 2,2 ומאחר שהן זהות נקבל:

$$R \div S$$

B
2

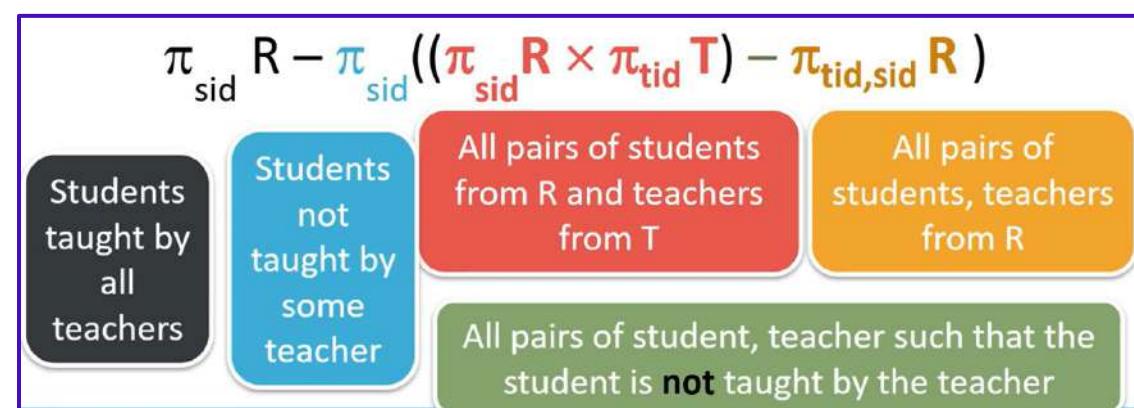
דוגמה נוספת עם ממשמעות ליחסים

R	tid	sid	course
20	202	os	
30	450	calculus	
10	202	db	
10	240	db	

T	tid	tname	dept
10	cohen	math	
20	levy	cs	

$$\pi_{tid,sid} R \div \pi_{tid} T$$

ההטלות מבוצעות לפני החילוק, לכן אפשר להתחיל מלחשב אותן:
הטלה של T על tid משaira לנו את כל המזהים של המרצים. הטלה של R על sid, tid מותנה לנו זוגות של מרצה עם סטודנט שהוא מלמד.
בפועל חילוק נשאר עם העמודה sid, ונשאר עם הערכות sid שמופיעים ביחד עם כלtid של המרצים. בדוגמה שלנו זה רק הסטודנט 202. מה
השאילתת מחזירה? הסטודנטים שלומדים אצל **כל** המרצים. איך יוכלו לבטא את השאלה בלי פועלות חילוק?



שאלה נוספת

R(tid, sid, course)
S(sid, sname, year)
T(tid, tname, dept)

Find sid-s of students who studied all courses taught by 'Gil'

המیدע על המרצים מופיע ב-T, והמیدע על מי מלמד איזה קורס ואת איזה סטודנט מופיע ב-S. יכולנו לחושב שפתרון לבעה יהיה:

$$\pi_{sid} \sigma_{tname='Gil'}(T \bowtie R)$$

אבל זה לא הפתרון. השאלה זו מחייבת את כל הסטודנטים שיש איזשהו קורס של גיל שאוטו הם לומדים (פחות אחד) ואנחנו רצינו הסטודנטים שלומדים את **כל** הקורסים של גיל. המילה // מرمצת הרבה פעמים שצריך להשתמש בפועלות חילוק.

בצד ימין של החילוק נרצה לשים את כל הקורסים של גיל, נקבל: $(\sigma_{tname='Gil'} T \bowtie R) \div \pi_{course}$

בצד שמאל נרצה לשים את זוגות של סטודנט וקורס בר שסטודנט לומד את הקורס: $\pi_{sid,course} R$.

סה"כ נקבל: $\pi_{sid,course} R \div \pi_{course} (\sigma_{tname='Gil'} T \bowtie R)$

הערה: חשב לעשות את הטלה **לפניהם** פועלות חילוק, כי רק את הזוגות של sid, course נרצה לוודא שהם מכילים את כל הקורסים בצד ימין. אם לא נעשה זאת, נקבל תשובה לא נכונה.

מה זו שקולות בביטויים?

נגיד שבביטויים E_1, E_2 שקולים כאשר הם מחזירים את אותה התוצאה, ללא תלות בתוכן היחס. נסמן \equiv .

$$E1 = \pi_A R$$

$$E2 = \pi_A \pi_{A,B} R$$

$$E3 = \pi_A \pi_{A,C} R$$

כל שלושת הביטויים שקולים, אבל E_1 יהיה הביטוי הכייעיל.

הובחת שקולות

$$\pi_A R \cup \pi_A S \equiv \pi_A (R \cup S)$$

איך נוכיח את שקולות הביטויים? באמצעות הכללה זו פיוונית. נתחיל מהכיוון האדום:

$$a \in \pi_A R \cup \pi_A S \Leftrightarrow a \in \pi_A R \vee a \in \pi_A S \Leftrightarrow \exists b \text{ s.t } (a, b) \in R \text{ or } (a, b) \in S \Leftrightarrow \exists b \text{ s.t } (a, b) \in R \cup S \Leftrightarrow a \in \pi_A (R \cup S)$$

הפרצת שקולות

$$\pi_A R \cap \pi_A S \not\equiv \pi_A (R \cap S)$$

איך נפרק את שקולות הביטויים? באמצעות דוגמה נגדית:

R	
A	B
1	2

S	
A	B
1	3

בהתוצאה הביטוי האדום קיבל את השורה עם הערך אחד. לעומת זאת בהתוצאה הצהובה לא קיבל אף שורה, כי אין אף שורה שנמצאת גם ב-R וגם ב-S. לכן הביטויים לא שקולים.

דוגמה

בاهינתן היחסים (C, R, S) , מה אפשר להציג על היחסים הבאים?

- 1) $\sigma_{C>5} R - \sigma_{C>5} S$
- 2) $\sigma_{C>5} (R - S)$

האם הם שקולים? האם יש הכללה באחד הצדדים? באך צד? נתחיל מلنשות להוכיח שקולות:

$$\begin{aligned} \sigma_{C>5} R - \sigma_{C>5} S &\Leftrightarrow (a, b, c) \in \sigma_{C>5} R - \sigma_{(C>5)} S \Leftrightarrow (a, b, c) \in \sigma_{C>5} R \wedge (a, b, c) \notin \sigma_{C>5} S \Leftrightarrow \\ &\Leftrightarrow (a, b, c) \in R \wedge C > 5 \wedge ((a, b, c) \notin S \vee C \leq 5) \Leftrightarrow (a, b, c) \in R \wedge C > 5 \wedge (a, b, c) \notin S \Leftrightarrow \\ &\Leftrightarrow C > 5 \wedge (a, b, c) \in R \setminus S \Leftrightarrow (a, b, c) \in \sigma_{C>5} (R - S) \end{aligned}$$

דוגמה נוספת

בاهינתן היחס (A, B, C, R) , מה אפשר להציג על היחסים הבאים?

- 1) R
- 2) $\pi_{A,B} R \bowtie \pi_{B,C} R$

האם הם שקולים? האם יש הכללה באחד הצדדים? באך צד?

נמצא דוגמה נגדית לשקולות, למשל R:

A	B	C
1	2	1
2	2	3

נבדוק את יחס 2:

הצירוף הטבעי שלהם R ייתן לנו:

A	B	C
1	2	1
1	2	3

הצירוף הטבעי שלהם R- ייתן לנו:

B	C
2	1
2	3

הצירוף הטבעי שלהם R+ ייתן לנו:

A	B
1	2
2	2

אפשר לראות שהתוצאה הסופית לא שקופה ליחס R . אפשר אבל להראות שהביטוי הראשוני מוכל בשני. נראה זאת:

$$(a, b, c) \in R \Rightarrow (a, b) \in \pi_{A,B}R \wedge (b, c) \in \pi_{B,C}R \Rightarrow (a, b, c) \in \pi_{A,B}R \bowtie \pi_{B,C}R$$

דוגמה נוספת

נתונים היחסים $R(A, B, C), S(C)$. מה אפשר להגיד על היחסים הבאים?

- 1) $\pi_A(R \div S)$
- 2) $(\pi_{A,C}R) \div S$

האם הם שקולים? האם יש הכליה באחד הצדדים? באף צד?

נראה באמצעות דוגמה נגדית שאין שקולות:

יחס S יחס R

A	B	C	
1	1	1	
1	0	0	

C
0
1

ubo של השאלה הראשונה:

נחשב $S \div R$, אילו עמודות יש בשמאלי שאין בימין? A, B . נרצה את הזוגות b, a שמוופיעים עם כל אחד מהערבים $-S$. אין אף זוג שמוופיע עם שני הערבים. הזוג $(1,1)$ מופיע רק עם $c = 1$, והזוג $(1,0)$ מופיע רק עם $c = 0$. לכן תוצאות תוצאה ריקה.

ubo של השאלה השנייה:

נחשב את $R \div S$ ונקבל את הטבלה:

A	C
1	1
1	0

בחילוק נחשב את ערבי A שמוופיעים עם כל אחד מערבי C של היחס S . נראה בעמודה A שהערך 1 אכן מקיים את תנאי זה.

נרצה להראות שהביטוי הראשוני מוכל בשני, בולם: $S \div (\pi_{A,C}R) \subseteq \pi_A(R \div S)$

$$(a) \in \pi_A(R \div S) \Rightarrow \exists b s.t (a, b) \in R \div S \Rightarrow \exists b s.t \forall c if (c) \in S \text{ then } (a, b, c) \in R \Rightarrow \forall c if (c) \in S \text{ then } (a, c) \in \pi_{A,C}R \Rightarrow (a) \in \pi_{A,C}R \div S$$

שימושים באופטימיזציה ביטויים

מערכת-h database מפעילה מאחוריה הקלעים שקוליות שיצמצמו כמה שיותר תוצאות ביןיהם אבל לשומר על נוכנות השאלה.

איך מיעלים חישוב?

- 1) להפעיל מוקדם פעולות של בחירה (מצמצם את היחסים).
- 2) להפעיל מוקדם פעולות של הטלה (מצמצם את היחסים).
- 3) לשימוש לב לסדר ביצוע הצירופים.

דוגמה

נדרשנו לחשב את השאלה: $((T \bowtie S) \bowtie (R \bowtie (\sigma_{tname='levy'}(S)))$

מה לא טוב בכך?

- 1) צירוף טבעי של S, T כאשר בין היחסים t ו- s Teachers-students אין אף שדה משותף. פעולה הצירוף הטבעי זו פועלות מכפלה קרטזית והוא בזבזנית.
- 2) אנחנו מעוניינים רק בשורות שבן ששם המרצה הוא 'levy', ואנחנו עושים לפניה חישובים מאוד יקרים.
- 3) בסוף אנחנו מעוניינים רק בשנה, אם יש אטריבואיטים נוספים ביחסים היה אפשר להיפטר מהם מוקדם יותר.

פתרון:

ניקח קודם ורק לפי שם מרצה לוי: $\pi_{year}(\sigma_{tname='levy'}(T)) \bowtie ((\pi_{year}(S) \bowtie (\pi_{year}(R)))$

מבצע פעולות הטלה מוקדם יותר, מה שיישאיר אותנו עם אטריבואיטים שימושתיים בצירוף הטבעי:

$((\pi_{year}(\sigma_{tname='levy'}(T)) \bowtie (\pi_{sid,year}(S) \bowtie (\pi_{tid,sid}(R))))$

מבצע קודם צירוף טבעי בין יחסים שיש להם אטריבואיטים משותפים, במקום להתחילה עם צירוף טבעי שאין בו אטריבואיטים משותפים מה שיוצר מכפלה קרטזית:

$\pi_{year}(((\pi_{tid,sid}(R) \bowtie (\pi_{sid,year}(S))) \bowtie \pi_{tid}(\sigma_{tname='levy'}(T)))$

Week 3: SQL

SQL 2: Intro to Queries 3.1

Structured Query Language

אנחנו נקבל בשאלתה טבלאות, ונחזיר טבלת פלט. תוצאה הפלט מודפסת למסך, הן לא נשמרות (אלא אם נבקש) ולא משנות את המסלע עצמו. אנחנו נקבל שאלתה מהמשתמש, שתורגם על ידי מסד הנתונים לאלגברה רלציונית, יעבור אופטימיזציה על ידי המרה לביטוי שקול ויעיל יותר לחישוב, אותו המערכת מחשבת עם ערכים של המסלע ונdelivr בסוף טבלה.

SQL שונה מכל שאר שפות התוכנות, היא שפה דקלרטיבית, הכוללת מילים מושגיות מה רצים לראות בתוצאה אבל לא אומרם איך לחשב את זה במפורש.

SQL vs. Relational Algebra

ב-RA-Anchored מניחים שבכל טבלה יש קבוצה של שורות בלי חוזרות, לעומת זאת ב-SQL יכולים להיות שקים של שורות, כלומר כל שורה יכולה להופיע מספר פעמים.

ב-RA-Anchored אין ערכי NULL, ב-SQL הם כן יכולים להופיע.

RA מופלת בלוגיקה של אמת או שקר, לעומת זאת SQL שמקובל בנוסף גם ערך שלא ידוע מה תוצאתו.

RA הוא Turing Complete, כלומר אפשר לכתוב בתוכנה כל תוכנית אפשרית. SQL הוא כן כן.

שאילתת SQL בסיסית

מבנה בצורה הבאה:

```
SELECT Attributes  
      FROM relations  
      WHERE condition
```

שלושה חלקים: SELECT אוסף של תכונות, FROM אוסף של יחסים, WHERE איזשהו תנאי מתקיים.

דוגמת הרצתן

Sailors				Boats		
sid	sname	rating	age	bid	bname	color
22	Alice	7	45	101	Iron Man	red
31	Barbara	8	55.5	103	Moonlight	green
58	Carol	10	35			
70	Alice	10	25			

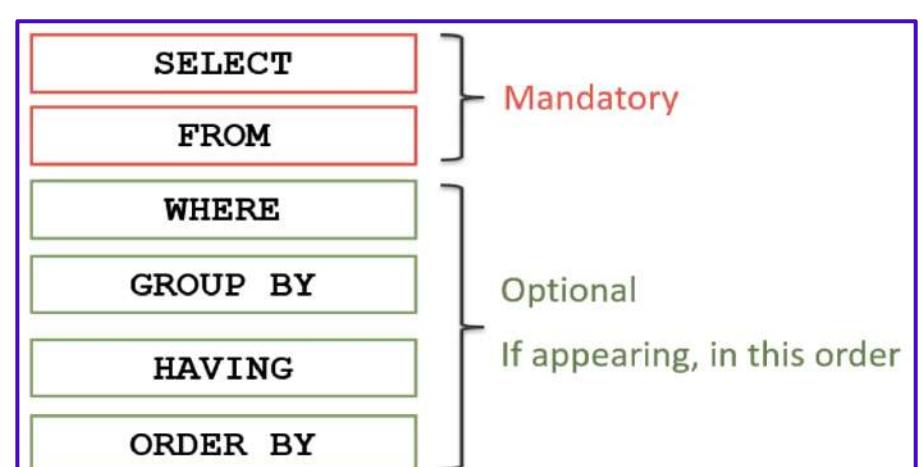
Reserves		
sid	bid	day
22	101	1/1/2017
58	103	2/3/2017

למשל:

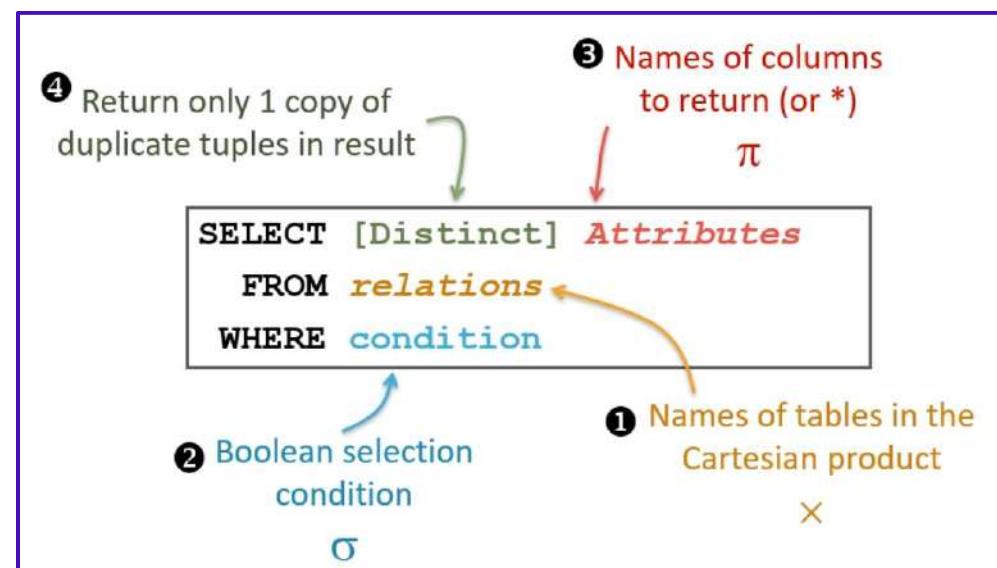
```
SELECT sid,sname  
      FROM Sailors  
      WHERE sid=1122
```

SQL 3: SELECT, WHERE 3.2

מבנה בסיסי של שאלה

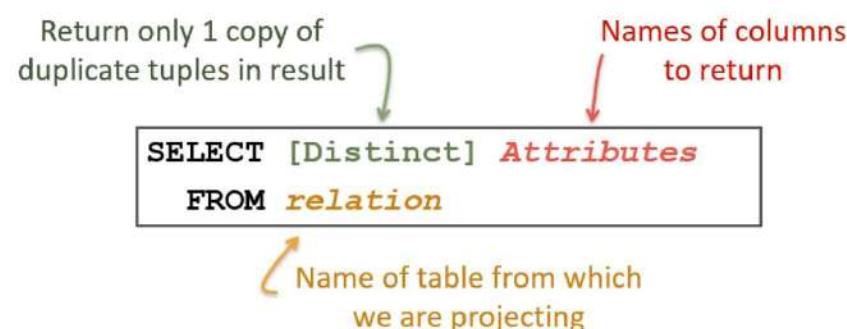


SQL היא שפה case insensitive. המוקם היחיד שבו יש משמעות ל-case הוא בין quote marks ("). הפורמט הבסיסי מכיל בדרך כלל SELECT, FROM, WHERE:



דוגמה

שאילתת הטלה – זו למשל דוגמה שבה לא נראה מכפלה קרטזית, כי נרשם רק יחס אחד ואין פה בחירה (לא מופיע WHERE):



למשל –

Sailors

sid	sname	rating	age
22	Alice	7	45
31	Barbara	8	55.5
58	Carol	10	35
70	Alice	10	25

```
SELECT sname  
FROM Sailors;
```

ונקבל את הטבלה:

sname
Alice
Barbara
Carol
Alice

לכן זה כמו הטלה, כי אפשרנו כאן שורות כפולות (Alice). אם נוסיף את המילה DISTINCTSELECT DISTINCT נקבל בדיקת הטלה π_{sname} .

דוגמה נוספת –

Sailors

sid	sname	rating	age
22	Alice	7	45
31	Barbara	8	55.5
58	Carol	10	35
70	Alice	10	25

```
SELECT sid, sname  
FROM Sailors;
```

! Can there be duplicates in the result?

sid	sname
22	Alice
31	Barbara
58	Carol
70	Alice

נשים לב לכך לא משנה אם נכתוב DISTINCT או לא, כי sid הוא מפתח, ולכן בכל מקרה לא ניתן שורות כפולות. אם נוסיף, זה לא ישנה את התוצאה אבל זה יכvide על המסד ללא סיבה.

אם לא נרצה לבצע בחירה נוכל לבצע * SELECT כדי להציג את כל העמודות (או בשילוב DISTINCT כדי להציג ייחודיים). למשל:

sid	sname	rating	age
22	Alice	7	45
31	Barbara	8	55.5
58	Carol	10	35
70	Alice	10	25

```
SELECT *
FROM Sailors
WHERE rating < 10;
```

sid	sname	rating	age
22	Alice	7	45
31	Barbara	8	55.5

זה כמו הביטוי $\sigma_{rating < 10} Sailor$.

פונקציות ב-SQL

נוכל לבקש לבצע פונקציות בחלק מהבחירה, למשל:

Sailors

sid	sname	rating	age
22	Alice	7	45
31	Barbara	8	55.5
58	Carol	10	35
70	Alice	10	25

```
SELECT sid, age/rating
FROM Sailors
WHERE rating = 10;
```

sid	?column?
58	3.5
70	2.5

נשים לב שם העמודה לא מוגדר כי ביקשנו עמודה שלא מופיעה בטבלה המקורית. אם נרצה להוסיף לה שם: `ar` אזי SELECT sid, age/rating AS ar יחזיר את ערכיו sid ועבור כל אחד מהם את הפונקציה הקבועה 0.

תנאים ב-WHERE

```
SELECT sid
FROM Sailors
WHERE (rating != 10 and age is NOT NULL) or
      sname LIKE 'A%_e' ;
```

עם האפשרויות:

Numerical and
string comparisons:
`!=,<>,=,<,>,>=,<=`

Logical
components:
AND, OR, NOT

NULL checking:
Is NULL, Is NOT NULL

Text Pattern matching: LIKE
_ any single character
% 0 or more characters

לගבי LIKE – למשל מה שבתו בשאלתה 'e'%' מתחילה ב-e, מסתיים ב-e ויש בו לפחות אחת באמצע.

מיון תוצאות

אפשר גם לציין באיזה סדר אנחנו רוצים לראות את התוצאה באמצעות הוספת ORDER BY. המיון בצורה דיפולטיבית הוא בסדר עולה, אבל אפשר לציין מפורשת בדרך הבאה:

Sailors			
sid	sname	rating	age
22	Alice	7	45
31	Barbara	8	55.5
46	Danna	7	60
58	Carol	10	35
70	Alice	10	25

```
SELECT sid
FROM Sailors
ORDER BY rating ASC,
          age DESC
```

sid
46
22
31
58
70

יש כאן מיון לפי שתי עמודות, קודם נמיין לפי rating בסדר עולה – ואם נדרש לשזור שווין אנחנו נמיין לפי age בסדר יורד.

שאילתת מכפלה קרטזית

אם נרצה לבצע מכפלה קרטזית, זה יבוצע דרך חלק ה-FROM.

– דוגמה –

Sailors

sid	sname	rating	age
22	Alice	7	45
31	Barbara	8	55.5
58	Carol	10	35
70	Alice	10	25

Reserves

sid	bid	day
22	101	1/1/2017
58	103	2/3/2017

```
SELECT *
FROM Sailors, Reserves;
```

נשים לב שב-Sailors יש 4 שורות, וב-Reserves יש 2 שורות. לכן בתוצאה תהיה 8 שורות, ותראה כך:

sid	sname	rating	age	sid	bid	day
22	Alice	7	45	22	101	1/1/2017
31	Barbara	8	55.5	22	101	1/1/2017
58	Carol	10	35	22	101	1/1/2017
70	Alice	10	25	22	101	1/1/2017
22	Alice	7	45	58	103	2/3/2017
31	Barbara	8	55.5	58	103	2/3/2017
58	Carol	10	35	58	103	2/3/2017
70	Alice	10	25	58	103	2/3/2017

דוגמה שימושת בבחירה הטלה ומכפלה קרטזית:

Sailors

sid	sname	rating	age
22	Alice	7	45
31	Barbara	8	55.5
58	Carol	10	35
70	Alice	10	25

Boats

bid	bname	color
101	Iron Man	red
103	Moonlight	green

Reserves

sid	bid	day
22	101	1/1/2017
58	103	2/3/2017

```
SELECT bname
```

```
FROM Sailors, Reserves, Boats
WHERE Sailors.sid = Reserves.sid and
      Reserves.bid = Boats.bid and
      Sailors.sname = 'Alice';
```

What does
this return?

נשים לב ששתי השורות הראשונות ב-WHERE הן כמו צירוף טבעי בין Sailors, Reserves, Boats, כי זו עמודה שמופיעה רק בטבלה אחת. אפשר גם כך:

```
SELECT bname
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid and
      R.bid = B.bid and
      sname = 'Alice';
```

צירוף על תנאי

```
SELECT S1.sname, S2.sname
FROM Sailors S1, Sailors S2
WHERE S1.sid != S2.sid and
      S1.sname = 'Rusty'
```

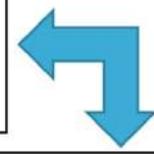
קיים דרך מעט מהירה יותר לקיים צירוף, וזה באמצעות JOIN INNER בצורה שבתמונה. זה יביא לנו זוגות של שמות מלחים כך שהראשון הוא Rusty והשני הוא Mischa אחר.

```
SELECT S1.sname, S2.sname
FROM Sailors S1 INNER JOIN Sailors S2 on
      (S1.sid != S2.sid)
WHERE S1.sname = 'Rusty'
```

צירוף טבעי

צירוף טבעי דוחש שווין על כל האטሪביוטים המשותפים.

```
SELECT S.sname
FROM Sailors S, Reserves R
WHERE S.sid = R.sid and
      S.age > 20
```



```
SELECT S.sname
FROM Sailors S NATURAL JOIN
      Reserves R
WHERE S.age > 20
```

פעולת ה-DISTINCT

מתי נרצה להשתמש ב-DISTINCT? הזכרנו שהזיהוי משורט התוצאה, לעיתים לא יכולות להיות כפליות בתוצאה ואז אין סען לציין זאת.
דוגמאות:

```
SELECT sid
FROM Sailors;
```



```
SELECT DISTINCT sid
FROM Sailors;
```

```
SELECT sid
FROM Reserves;
```



```
SELECT DISTINCT sid
FROM Reserves;
```

כאן מכיון שה sid זה מפתח, אין צורך לכתב DISTINCT כי ככל מקרה לא יתכנו כפליות
שבצד ימין נוכל להגיד רק אילו ימאים הזמינים לפחות ספינה אחת.

```
SELECT sid
FROM Sailors Natural Join Reserves
```



```
SELECT DISTINCT sid
FROM Sailors Natural Join Reserves
```

אומנם sid הוא מפתח ב-sailors, אבל הוא לא sid בתוצאות הצירוף הטבעי. לעומת שאלהה העילונה, כמו בדוגמה הקודמת מצד ימין, נקבל sid בכמות
ההזמנות שבוצעו לעומת בשאלתה התחתונה (עם ה-DISTINCT) נקבל sid בכל היותר.

SQL 5: UNION, EXCEPT, INTERSECT 3.4

Three SET Operators

```
[Query]
UNION
[Query];
```

```
[Query]
EXCEPT
[Query];
```

```
[Query]
INTERSECT
[Query];
```

הפורמט הוא שאלתה שלמה, לאחר בר הפעולה, ואז שאלתה נוספת.

פעולות כאלה הן פעולה תקינה רק אם השאלות המשתתפות מחזירות את אותו מספר עמודות ויש להם טיפוסים מתאימים (בדיקה האם טיפוסים או
מספר דומים כמו float-integer).

דוגמאות:

```
SELECT sid
FROM Sailors
UNION
SELECT age
FROM Sailors;
```

```
SELECT *
FROM Sailors
UNION
SELECT *
FROM Reserves;
```

השאלתה השמאלית תקינה, כי sid הוא מספר וagem age הוא מספר, ושתיهن מחזירות את אותו מספר עמודות.
השאלתה הימנית לא תקינה, כי sid ב-sailors יש 4 עמודות בעוד sid ב-reserves יש 3.

הערות:

- שם העמודה בשאלת התוצאה נקבע על פי השאלה העילונה.
- בכל שלושת השאלות לא יהיו כפליות בתוצאה (כמו בפעולות על קבוצה מתמטית). אם נרצה בכל זאת נוסיף את המילה ALL אחרי הפעולה:

[Query]
UNION ALL
[Query];

[Query]
EXCEPT ALL
[Query];

[Query]
INTERSECT ALL
[Query];

אבל בשניים הימניים אף מערכת db לא תומכת.

במota הפעמים ב-ALL UNION שכל שורה תופיע היא סכום הפעמים שהוא מופיע בשאלת העילונה + בשאלת התछטונה.

דוגמה –

```
SELECT sid
  FROM Reserves R, Boats B,
 WHERE R.bid = B.bid AND color = 'red'
INTERSECT
SELECT sid
  FROM Reserves R, Boats B,
 WHERE R.bid = B.bid AND color = 'green'
```

מה השאלה זו מחייבת?

העלונה מחייבת את sid מ-Reserves ו-boats reseves עם התנאים שהזמינים, וכן יש צירוף טבעי בין שתי הטבלאות האלה ולאחר מכן הטלה:
 $\pi_{sid} \sigma_{color='red'}(Reserves \bowtie Boats)$

השאילתה התछטונה עשויה כמעט אותו דבר, רק עברו צבע ירוק.

מציעים חיתוך וכן נקבל את sids של ימאים שהזמינים גם ספינות אדומות וגם ספינות ירוקות. האם יכולנו לכתב את זה גם ללא חיתוך? כן:

```
SELECT R1.sid
  FROM Reserves R1, Reserves R2
    X Boats B1, Boats B2
 WHERE R1.sid = R2.sid and
       R1.bid = B1.bid and
       B1.color = 'red' and
       R2.bid = B2.bid and
       B2.color = 'green';
```

leshim lib לשורות הממורקרות בצבעים, ואיזה צירופים طبيعيים כל אחת מהן יוצרת.

דוגמה –

```
SELECT sid
  FROM Sailors
EXCEPT
SELECT sid
  FROM Reserves
 WHERE bid = 103;
```

בשאילתה הראשונה קיבל את כל הימאים, בתछטונה את אלו שהזמינים את ספינה 103, ואז מציעים חיסור, כלומר את אלו שלא זמינים את הספינה.

דוגמה –

```
SELECT sid
  FROM Reserves R
 WHERE bid <> 103;
```

כאן יש שאלה בודדה שמחזירה sid מהטבלה Reserves עבורם bid שונה מ-103.

האם שתי הדוגמאות מחייבות את אותן התוצאות? **וזו נושא למבחן –**

השאילתה הירוקה מחייבת את כל הימאים שהזמינים ספינה כלשהי שהיא לא 103. אבל אם ימאי החיזיר למשל את ספינה 100 ואת ספינה 103, הוא עדין יופיע בטבלה, וכך נקבל אותו למרות שרצינו רק את אלה שלא זמינים את הספינה.

דוגמיה –

```

SELECT      sname
  FROM      Sailors
EXCEPT
SELECT      sname
  FROM      Reserves R, Sailors S
 WHERE     R.sid = S.sid and bid = 103;

```

כאן יש שאלתה של שמות ימאים פחות שמות של ימאים שהזמין את ספינה 103? לא בדיק. הסיבה לכך היא ש-sname הוא לא מפתח ב-Sailors. לעומת זאת, השם יופיע פעמיים את אותו השם, ולמעט נפחית את אלה שהזמין את ספינה 103, ככלור נקבל שמות של ימאים עוברים אין ימאי עם אותו שם שהזמין את ספינה 103. לא בדיק מה שרצינו.

באופן כללי – להיזהר עם פעולות INTERSECT/EXCEPT עם ערכים שהם לא מפותחים

דוגמיה –

Sailors			
sid	sname	rating	age
22	Alice	7	45
31	Barbara	8	55.5
58	Carol	10	35
70	Alice	10	25

```

SELECT      DISTINCT sname
  FROM      Sailors S
UNION ALL
SELECT      DISTINCT sname
  FROM      Sailors S;

```

השאילתה הראשונה תחזיר לנו Alice, Barbara, Carol ואגם השאילתה התחתונה. ב-ALL Union Union נקבל את כל 6 השורות.

SQL6: Subqueries 3.5

[הרכבת שאילתות](#)

$$\pi_{\text{sid}} S - \pi_{\text{sid}} ((\pi_{\text{sid}} S \times \pi_{\text{bid}} B) - \pi_{\text{sid,bid}} R)$$

האם אפשר לבטא את השאילתה זו ב-SQL? מה היא ממחזרה?



בקיצור, זה כל הימאים שהזמין את כל הספינות.

באלבגרה רלציונית, יכולים לבתוב פשוט את פעולה \setminus , אבל היא לא קיימת ב-SQL, ולכן חיברים דרך לכתוב השאילתה כזו.

[שאילתות מkonnect](#)

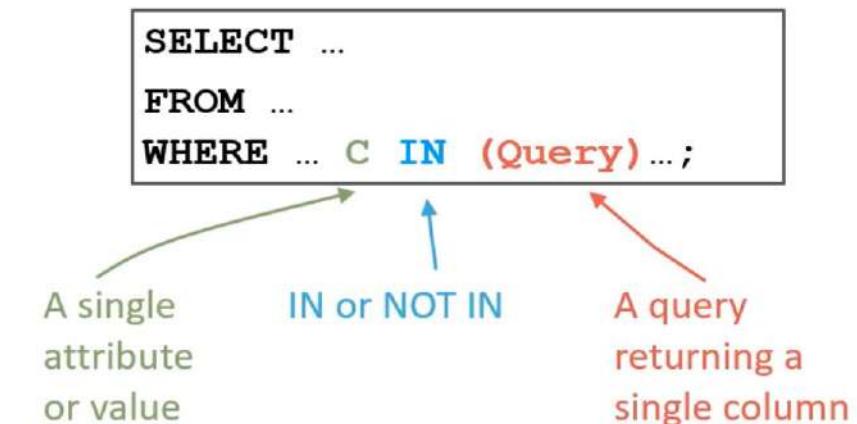
SELECT	Can contain subqueries
FROM	
WHERE	
GROUP BY	Cannot contain subqueries
HAVING	
ORDER BY	

ב-SQL בשונה מ-RA, תת שאילתות יכולות להיות מתואימות, ככלור מתייחסת לשאילתה שמחוצה לה.

תתי שאלות WHERE

בשאלה WHERE כתובים בתה שאלתה WHERE, אנחנו חייבים לכתוב אותה בחלק מביטוי בוליאני (כיו WHERE עובד על תנאים בוליאניים). אך השימוש בתה שאלתה בן הוא כדי להגדיר ביטוי בוליאני מורכב.

אחת מה דרכים לשימוש בתה שאלתה WHERE היא באמצעות IN או NOT IN:



הביטוי הבוליאני זהה יחזיר לנו ערך אמת אם הערך של C נמצא (או לא נמצא) בתוצאת השאלתה. נוח לחשב על זה כאשר שאלת השאלתה מחושבת עבור שוב ושוב לכל תוצאה של המכפלת הקרטזית JOIN (בפועל יש להז אופטימיזציה).

דוגמה –

```

SELECT S.sname
  FROM Sailors S
 WHERE S.sid NOT IN (SELECT R.sid
                        FROM Reserves R
                       WHERE R.bid = 103);
  
```

בתוך השאלתה הפנימית קיבל את sid הימאים שהזמין את ספינה 103. لكن בשילוב NOT-IN קיבל את sid הימאים שלא הזמין את ספינה 103, ואז נבחר את שמותיהם.

דוגמה –

```

SELECT S.sname
  FROM Sailors S
 WHERE S.sid NOT IN
      (SELECT R.sid
        FROM Reserves R
       WHERE R.bid IN
            (SELECT B.bid
              FROM Boats B
             WHERE B.color='red'))
  
```

כאן יש שתי שאלות בתוך אחת. נבחן מבפנים החוצה. הפנימית מחזירה bid של ספינות אדומות, אחרת יחזיר sid עברון R.bid בקבוצה – כלומר ימאים שהזמין ספינות אדומות. השאלת כולה מחזירה שמות של ימאים שלא הזמין ספינה אדומה.

דוגמה –

```

SELECT S.sname
  FROM Sailors S
 WHERE S.sid in
      (SELECT R1.sid
        FROM Reserves R1, Boats B1
       WHERE R1.bid=B1.bid and B1.color='red'
     EXCEPT
      SELECT R2.sid
        FROM Reserves R2, Boats B2
       WHERE R2.bid=B2.bid and B2.color='green');
  
```

מה השאלת הזו מחזירה?

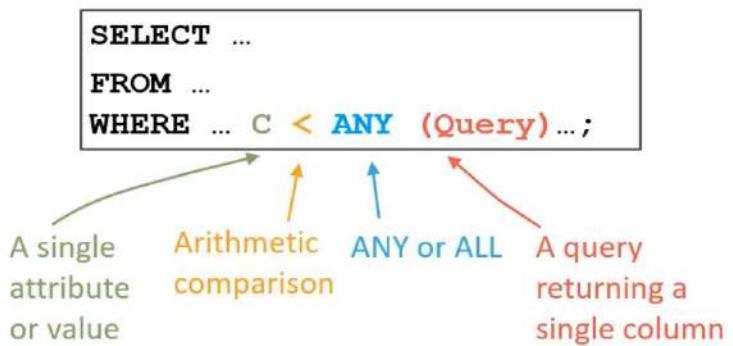
צירוף טبعי בין reserves ל-boats והורשת שהצבע יהיה אדום, לכן קיבל sid של ימאים שהזמין ספינה אדומה.

במעט אותו דבר, sid של ימאים שהזמין ספינה ירוקה.

sid של ימאים שהזמין ספינה אדומה ולא הזמין ספינה ירוקה.

מחזירים שמות של ימאים שה-sid שלהם נמצא בקבוצה זו.

דרך נוספת להשתמש בתת שאלתה ב-WHERE היא באמצעות ANY או ALL :



בוחרים ערך בודד, תנאי השוואת אריתמטי, מולוּה ב-ANY או ALL. זה ייחזיר ערך אמת אם הערך של C עונה על האופרטור האריתמטי ייחסית ל-ANY או ל-ALL תוצאות השאלה.

דוגמה –

תת השאלה תחזיר את גילאי הימאים, והחיזוקית תחזיר ערך אמת עבור שורות בהן ה-age גדול ממספר ימאי אחד בטבלה. כלומר שורות הימאים שאינם הצביעו.

```

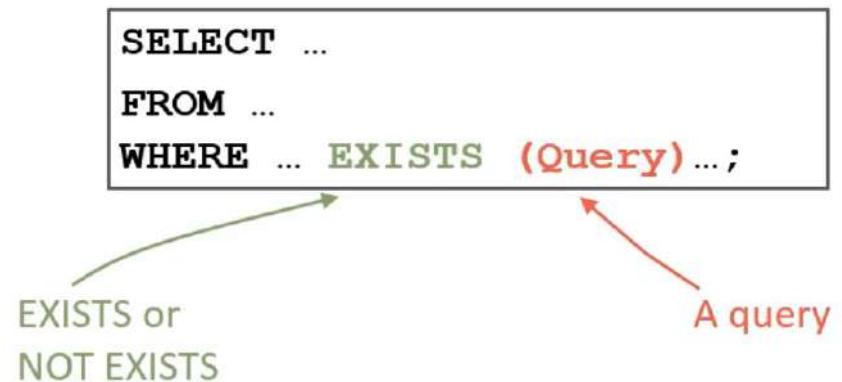
SELECT *
  FROM Sailors S1
 WHERE S1.age > ANY (SELECT S2.age
                        FROM Sailors S2);
  
```

שמות ימאים שהגיל שלהם גדול מכל הגילאים בטבלה Sailors, מכיוון שאף ימאי לא יכול להיות גדול מכלם, כי בפרט אינו גדול עצמו, השאלה זו תחזיר ערך ריק. אפשר לתקן אותה באמצעות $=>$ ועוד נקבל שורות של ימאים שגדלים שווים מכל הימאים, כלומר ימאים הצביעו.

```

SELECT *
  FROM Sailors S1
 WHERE S1.age > ALL (SELECT S2.age
                        FROM Sailors S2);
  
```

דרך נוספת להשתמש בתת שאלתה ב-WHERE היא באמצעות EXISTS :



הביטוי עם EXISTS מחייב ערך אמת בתת השאלה לא ריקה, ובעור NOT EXISTS נקבל ערך אמת רק עבור שאלה ריקה. כשבודקים NOT EXISTS אפשר פשוט לשום ב-SQL את התו * כי זה לא משנה מה חוזר.

דוגמה –

```

SELECT S.sname
  FROM Sailors S
 WHERE EXISTS (SELECT *
                  FROM Reserves R
                 WHERE R.bid = 103 and
                       S.sid = R.sid);
  
```

S not in subquery,
refers to outer query

נשים לב שכאן יש תת שאלה מתואמת, כי S מוגדר בתשאלה החיזוקית (מחוץ לסוגרים). מתחילה מ-FROM ועוברם שורה שורה על Sailors. בכל שורה נחשב את תת השאלה ובזדקים האם התוצאה אינה ריקה. כאשר אנחנו מחשבים את תת השאלה עבור שורה מסוימת, נקבל ערך מסויים של sid אותו נמוך במקום sid.S, אנחנו מחפשים שורות של reserves של ספינה 103 שבהם מדובר בבדיקה בימאי שנחקרו בעברם עלי. השאלה תחזיר שמות של ימאים עבורם קיימת תוצאה על השאלה, כלומר הם הזמינו את ספינה 103.

גרסאות שונות לביצוע שאלות של פועלות החילוק

נראה 3 גרסאות. מומלץ למודד אחת ולהשתמש בה.

: גרסה ראשונה:

```
SELECT sid
FROM Sailors S
WHERE NOT EXISTS
(SELECT B.bid
FROM Boats B
WHERE B.bid NOT IN
(SELECT R.bid
FROM Reserves R
WHERE R.sid = S.sid));
```

Sailors who
Reserved all
Boats

את הסגנון זהה קל להבין באמצעות העברת השאלה לשפה טבעית. גם כאן יש שאלתה מתואמת.
כאן נזכיר sid של ימאים עבורם לא קיימת ספינה שאיננה בקבוצת הספינות שהזמן הימאי, כלומר הזמן את כל הספינות.

: גרסה שנייה:

```
SELECT S.sid
FROM Sailors S
WHERE NOT EXISTS(
SELECT B.bid
FROM Boats B
WHERE NOT EXISTS(
SELECT R.bid
FROM Reserves R
WHERE R.bid=B.bid and
R.sid=S.sid))
```

Sailors who
Reserved all
Boats

כאן השאלתה מתואמת פעמיים. היא נותנת לנו sid של ימאים עבורם לא קיימת ספינה, עבורם לא קיימת הזמן – כלומר sid של ימאים שהזמין את כל הספינות.

: גרסה שלישית – מומלץ:

```
SELECT S.sid
FROM Sailors S
WHERE NOT EXISTS((SELECT B.bid
FROM Boats B)
EXCEPT
(SELECT R.bid
FROM Reserves R
WHERE R.sid = S.sid));
```

Sailors who
Reserved all
Boats

ימאים שהזמין את כל הספינות, אבל **באן מופיעה בתת השאלתה כל הספינות**, ובפנים **הספינות שהימאי הספציפי שאנו חזו עוברים עליו הזמן**.
אחרי פועלת EXCEPT-EXCEPT קיבל את הספינות שהימאי לא הזמן, ונדרש שלא תהיה ספינה כזו – בambilם אחרות, הוא הזמן את כל.

```

SELECT      sname
FROM        Sailors S,
            (
                SELECT      sid
                FROM        Sailors
                EXCEPT
                SELECT      sid
                FROM        Reserves
                WHERE       bid = 103) T
WHERE       S.sid = T.sid
  
```

A subquery in the FROM must be given an alias

בתוך ה-FROM יש טבלה נוספת נספחת יש שאלתה שלמה. מבחינת החישוב, אנחנו צריכים להתחיל ממכפלה קרטזית של היחסים ב-FROM. נחשב את השאלות שמויפות שם. ב痼ע הווורד, יוחזרו sid של ימאים שלא הזמינו את ספינה 103. כדי לנצל להשתמש בה, **חייב** לתת לה שם, הפעם T. אחרי שעשינו מכפלה קרטזית בין Sailors לבין sid לטבלה שמכילה את sid של ימאים שלא הזמינו את ספינה 103, דרשו עליה שווין – ומשם נחזיר את השמות שלהם.

Week 4: SQL

SQL 7: Aggregation 4.1

פונקציות הקבוצה

מקבלת מולטי-סט של ערכים ומחזירה ערך יחיד. הפונקציות הסטנדרטיות הן:
COUNT – סופר את מספר השורות.

COUNT(*) – בכמה ערכים יש באтриビוט ספציפי A (אם מוסיפים DISTINCT זה יספור רק את היחודיים).
COUNT(DISTINCT A) – סומם את הערכים ב-A.
SUM(DISTINCT A) – ממוצע הערכים ב-A.

MIN(A), MAX(A) – ערך מינימלי ומינימלי (יכול לעבד גם על ערכים לא מספריים כל עוד מוגדרת מעליים אופציית השוואה, למשל במחזרות מינימום יჩיר את המחרוזת עם האות המוקדמת ביותר בא"ב).

דוגמאות

```
SELECT COUNT(*)  
FROM Sailors S
```

```
SELECT COUNT(sid)  
FROM Sailors S
```

בשמאל סופרים את כמה השורות ב-S, בימין סופרים ערכים בשדה sid. מתי זה יהיה שונה? אם sid יכול לקבל ערכי NULL, אז נתעלם מהם בספירה.

```
SELECT AVG(S.age)  
FROM Sailors S  
WHERE S.rating=10
```

```
SELECT COUNT(distinct color)  
FROM Boats
```

Computing Aggregate Values for Groups

```
SELECT  
FROM  
WHERE  
GROUP BY  
HAVING  
ORDER BY
```

Used to form the groups
Rows are in the same group if they are equal on all grouping attributes

בשאנו רוצים להשתמש בפונקציות הקבוצה לעתים אנחנו רוצים להפעיל אותן על כל הטרבלה. לעיתים אנחנו רוצים לחשב רק עבור קבוצות ערכים. במקרה זה נשתמש ב-GROUP BY.

```
SELECT  
FROM  
WHERE  
GROUP BY  
HAVING  
ORDER BY
```

Boolean expression used to qualify groups
Only groups for which the condition holds are used to create output

לעומת זאת, HAVING זה תנאי בוליאני שייצרנו מעל הקבוצות שלנו. אם WHERE היה תנאי בוליאני על שורות, GROUP BY היה תנאי בוליאני על קבוצות. כל קבוצה שייצרנו בעזרת GROUP BY מאפשרת לתנאי על HAVING. כל קבוצה תיצור לנו שורה אחת בפלט.

דוגמה 1

Sailors

sid	sname	rating	age
22	Alice	7	45
31	Barbara	8	55.5
58	Carol	10	35
70	Alice	10	25

```
SELECT rating, AVG(age)
FROM Sailors
GROUP BY rating;
```

השאילתה מבקשת להחזיר לכל דירוג מהו הגיל הממוצע של ימאים עם אותו הדירוג. אנחנו רואים בטבלה 3 ערכי rating שונים ולקן קיבל 3 קבוצות, 3 שורות בתוצאות:

rating	avg
7	45
8	55.5
10	30

דוגמה 2

Sailors

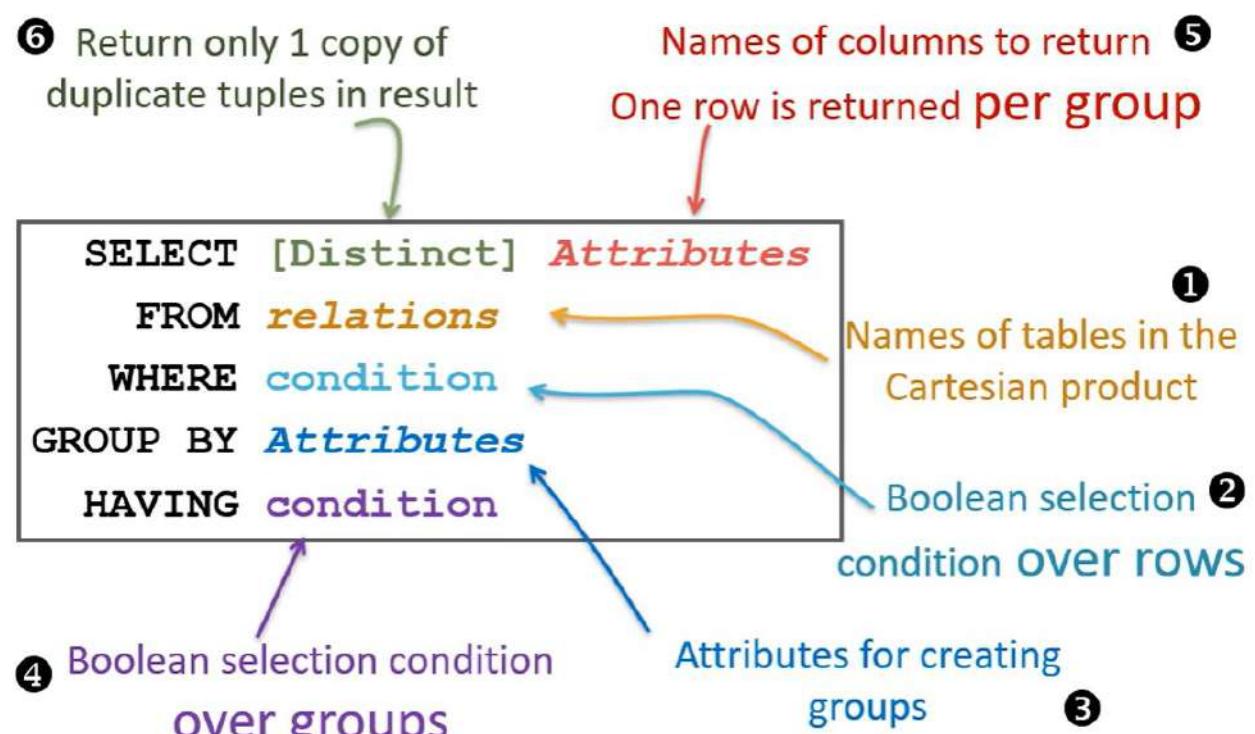
sid	sname	rating	age
22	Alice	7	45
31	Barbara	8	37
58	Carol	10	35
70	Alice	10	25
54	Danna	7	55
99	Erica	10	15

```
SELECT rating, AVG(age)
FROM Sailors
WHERE age < 50
GROUP BY rating
HAVING count(*) > 1;
```

השאילתה זו מבקשת לכל rating את הגיל הממוצע של הימאים, אבל רק עבור ימאים מתחת לגיל 50, וגם רק ב-rating שבו במות השורות בקבוצה (summation by COUNT(*)) גדול מ-1.

נתחיל למלחוק כל שורה שבה גיל הימאי גדול מ-50. לאחר מכן ניצור את הקבוצות לפי הדירוג, ואז נשאיר רק את הקבוצות שבהן מספר הימאים גדול ממש מ-1. זה נשאיר אותנו רק עם הקבוצה של הימאים המודרגים 10. לנוכח התוצאות:

rating	avg
10	25

שබולונה כללית

דוגמה 3

Sailors

<u>sid</u>	<u>sname</u>	<u>rating</u>	<u>age</u>
22	Alice	7	45
31	Barbara	8	37
58	Carol	10	35
70	Alice	10	25

```
SELECT sid, AVG(age)
FROM Sailors
GROUP BY rating
```

בשאילתתנו זו יש בעיה. אומנם אין בעיה לחשב את AVG(age), אבל מה אמרו לחזור בערך sid? הרי בקבוצה אחת יכולים להיות כמה ימאים עם ערכי sid שונים, אז מה מהם אמרורים להחזיר? זה יחזיר שגיאה.

מכאן נולד כלל אצבוע – **כל תכונה שמופיעה ב-GROUP BY מוחז לפונקציית הקבוצה, צריכה להופיע גם ב-GROUP BY**

דוגמה 4

Sailors

<u>sid</u>	<u>sname</u>	<u>rating</u>	<u>age</u>
22	Alice	7	45
31	Barbara	8	37
58	Carol	10	35
70	Alice	10	25

```
SELECT rating, AVG(age)
FROM Sailors
GROUP BY rating
HAVING sname <> 'Alice'
```

נבחן שעבור הקבוצה בעלת rating 10 ניתקל בקורסוי, כי מופיעים בקבוצה גם Carol וגם Alice, אך יכולות להיות גם Carol וגם Alice, מאותן כל שראים מקודם.

לעומת זאת, אם נשנה התנאי ל-'Alice' <> (min(sname) <> 'Alice') בסדר, כי זה מופיע בתוך פונקציית הקבוצה. אם זה היה התנאי הקבוצה עם rating 10 לא הייתה עוברת, כי השם המינימלי בקבוצה זו הוא sid Alice.

דוגמה 5

Sailors

<u>sid</u>	<u>sname</u>	<u>rating</u>	<u>age</u>
22	Alice	7	45
31	Barbara	8	37
58	Carol	10	35
70	Alice	10	25

```
SELECT S.sname,
       MAX(S.age)
  FROM Sailors S
```

מה הכוונה בשאילתתנו זו? בнерאה התכוונו למצוא את השם והגיל של הימאי המבוגר ביותר. אם אין לנו BY GROUP מתקיים בכל הטרבלה קבוצה אחת. MAX(s.age) מוגדר היטב מעל הטרבלה כולה, אבל S.name לא. כדי לתקן אפשר:

```
SELECT sname,
       MAX(age)
  FROM Sailors
 GROUP BY sname
```

ואז מקבל מכל קבוצת שם את הימאי עם השם זהה והגיל המקסימלי. אם בכל זאת הינו רצים לעשות זאת:

```
SELECT S.sname, S.age
  FROM Sailors S
 WHERE S.age =
    (SELECT MAX(S2.age)
     FROM Sailors S2)
```

דוגמה 6

```

SELECT B.bid, COUNT(*)
FROM Boats B, Reserves R
WHERE R.bid=B.bid and B.color='red'
GROUP BY B.bid

```

זה ייצור קבוצה לכל ספינה בצבע אדום, ונמצא החוצה את ה-*id* וב모ת השורות מכל קבוצה – נקבל את כמהות ההזמנות שיש לכל ספינה אדומה.

דוגמה 7

```

SELECT bname
FROM Boats B, Reserves R
WHERE R.bid=B.bid
GROUP BY bid, bname
HAVING count(DISTINCT day) <= 5

```

יצרנו קבוצות לפי *bid* ושם הספינה, ואנחנו משאירים רק את הקבוצות בהן $\text{count}(\text{DISTINCT day}) \leq 5$ כלומר מחפשים את שמות הספינות שהוזמנו בכלל היתר 5 ימים.

דוגמה 8

```

SELECT color
FROM Boats B
GROUP BY color
HAVING max(count(bid)) 

```

זה ניסיון בושל למצוא את הצבע שיש בו הכי הרבה ספינות. זו שאלה לא תקינה, היא לא תעבור ותוציא שגיאה. אנחנו מחלקים את הספינות לקבוצות לפי צבע, ומספרים כמה ספינות יש בכל צבע, לצורך הדוגמה נניח שיש 5 בכל צבע. פונקציית *max* מצפה לקבל כמה ערכים ולהחזיר ערך אחד, אבל נתנו לה ערך אחד, לכן אין משמעות להפעלת פונקציית הקבוצה על פונקציית הקבוצה. כמו כן, *HAVING* אמרו להיות תנאי בוליאני על קבוצות, וגם אם היה מחושב משהו – היה מחושב מספר ולא תנאי בוליאני. כדי לתקן:

```

SELECT color
FROM Boats B
GROUP BY color
HAVING count(bid) >= ALL
    (SELECT count(bid)
     FROM Boats
     GROUP BY Color)

```

דוגמה 9

Sailors

<u>sid</u>	sname	rating	age
22	Alice	7	45
31	Barbara	8	37
58	Carol	10	35
70	Alice	10	25

```

SELECT S.sid, S.age, (SELECT MAX(S2.age)
                        FROM Sailors S2
                        WHERE S2.age < S.age)
                FROM Sailors S;

```

עוד דוגמה לפונקציית הקבוצה, הפעם עם תת-שאילתה ב-SQL. מותר לעשות זאת אך ורק אם היא מחזירה ערך בודד. במקרה זה אנחנו מחשבים את הגיל המקסימלי של הימאים הצעירים מהשורה שאנו ברגע עוברים עליה. תוצאה:

sid	age	max
31	37	37
22	45	37
31	37	35
58	35	25
70	25	NULL

[האם אפשר לחשב חילוק באמצעות פונקציות הקבוצה?](#)

נניח שנרצה למצוא את שמות הימאים שהזמיןו את כל הספינות. אנחנו בעצם רוצים את הימאים שמספר הספינות שהם הזמין שווה למספר של כל הספינות:

```
SELECT sname
FROM Sailors S NATURAL JOIN Reserves R
GROUP BY sname, sid
HAVING COUNT(DISTINCT BID) = (SELECT COUNT (*)
                               FROM Boats)
```

הערה: יש מקרה קיצון עם טבלאות ריקות בהן זה יכשל. כל עוד הטבלאות אינן ריקות, זה עובד בבדיקה כמו פועלות החילוק.

SQL 8: Null Values 4.2

[ערבי NULL בטבלאות](#)

למשל השאלה הבאה עברו טבלת הימאים זו:

The diagram shows a table with columns sid, sname, rating, and age. The rows are:

- sid: 22, sname: Alice, rating: 7, age: NULL
- sid: 31, sname: Barbara, rating: 8, age: 37
- sid: 58, sname: Carol, rating: NULL, age: 35
- sid: 70, sname: Alice, rating: 10, age: 25

Two queries are shown below the table:

- Left Box:** `SELECT sname, rating + 1 FROM Sailors;`
- Right Box:** `SELECT * FROM Sailors WHERE age < 100;`

A red question mark is placed above the age column header in the table, and another red question mark is placed above the first 'NULL' value in the age column.

מה קורה לגבי אליס? האם היא תחזור בתוצאה? הגיל שלה קטן מ-100 או שלא?

מה לגבי קארול? מה זה `?NULL+1` ומה זה `?NULL < 100`?

אלו דברים שצורך להגדיר בצורה מדויקת.

[3 Valued Logic](#)

ב-SQL אנחנו משתמשים בלוגיקה של 3 ערכים: **אמת, שקר ולא ידוע**.

A	B	A and B	A or B
True	True	True	True
True	False	False	True
True	Unknown	Unknown	True
False	True	False	True
False	False	False	False
False	Unknown	False	Unknown
Unknown	True	Unknown	True
Unknown	False	False	Unknown
Unknown	Unknown	Unknown	Unknown

A	Not A
True	False
False	True
Unknown	Unknown

למען האמת, טבלאות האמת מתנהגות עם Unknown כמו שהיינו מצפים שהם יתנהגו. למשל פעולה AND דורשת את שני הערבים כדי להחליט מה התוצאה. אם יש לנו ביד אחת אמת, זה משנה מאוד אם ביד השנייה יש לנו שקר או אמת. לעומת זאת ב-OR זה לא משנה, כי מספיק אמת פעם אחת כדי שההתוצאה תהיה אמת.

חזרה לשאלתה – בתרע WHERE יש ביטוי בוליאני, אמת שקר או לא ידוע. רק שורות שעבורן חישבemo ערך אמת יוחשבו בתוצאה. מה שייצא שקר/לא ידוע לא יכנס לתוצאה.

[NULL Values in Expression](#)

- Expressions operating on NULL return NULL

NULL + 1 NULL * 7

- Expressions comparing NULL to another value return UNKNOWN

NULL > 10 NULL <> NULL NULL = NULL

- The correct way to determine if an attribute x has value NULL is

x IS NULL x IS NOT NULL

(1) כל ביטוי מתמטי שימושי על NULL מחזיר NULL

(2) כל השוואה שבוצעת על NULL מחרירה את הערך הבוליאני Unknown.

(3) הדרך הנכונה להבין האם לאייזהו אטሪビוט יש ערך NULL זה לשאול NULL IS x או NOT NULL IN x? זה יחזיר לנו את הערך שאנו צריכים.

דוגמה 1

```
SELECT *
FROM Sailors
WHERE sname = sname;
```

השאלתה זו למשה מחרירה את כל השורות בהן sname איטו NULL, כי הרו NULL=NULL מחייבownUnknown ולבן לא נקבל שורות כאלה בתוצאה.

```
SELECT *
FROM Sailors
WHERE rating > 5 or rating <= 5;
```

אותו עקרון, כי Unknown או Unknown זה Unknown or Unknown.

```
SELECT sname, rating * 0
FROM Sailors;
```

כאן, מבעד ל垦וקשים של שרה אפשר לראות שאלה שתחזיר לנו 0 rating שלהם היה NULL כי כל פעולה מתמטית על NULL תחזיר NULL, לבן מי שהוא לו NULL rating ישאר עם NULL.

[פונקציות הקבוצה והערך NULL](#)

כללים:

- **count(*)**: Counts all rows
- **count(A)**: Counts non-null A-s. Returns 0 if all As are null
- **sum(A), avg(A), min(A), max(A)**: Ignores null values of A. Returns null if all of As are null
- For the purpose of GROUP BY and DISTINCT, nulls in different rows are equal

B	C
1	null
2	null
3	4
3	5

```
SELECT count(*), count(c),
       min(c), sum(c)
FROM  (SELECT c
       FROM R
      WHERE c IS NULL or
            c <> NULL
     GROUP BY c) T
```

<http://sqlfiddle.com/#!9/a5d8ee/2>

כל תחת השאלה מהזירה טבלה עם עמודה אחת בשם C, ובה שורה אחת עם הערך NULL. מדוע? שורות 1 ו-2 עומדות בתנאי הראשון כי C IS NULL. שורות 3 ו-4 לא עומדות בתנאי, כי הביטוי למשתף NULL <> 4 והביטוי למשתף NULL <> 5 מחזירים שניהם UNKNOWN ולכן לא יוכנסו לתוצאה.

השאילתה החיצונית:

COUNT(*) סופרת כמה שורות יש בתוצאה, ויש שורה אחת.

COUNT(C) סופרת כמה ערכים שונים מ-NULL יש בתוצאה, ולכן התשובה היא 0.

min(c), sum(c) מעריכים מערך ה-NULL אלא אם כן זה הערך היחיד, וזה קורה לכן ייחדו שניהם NULL.

סך הכל תחזיר השורה:

1	0	NULL	NULL
---	---	------	------

[Left Outer Join](#)

Sailors

sid	sname	rating	age
22	Alice	7	45
31	Barbara	8	55.5
58	Carol	10	35
70	Alice	10	25

Reserves

sid	bid	day
22	101	1/1/2017
58	103	2/3/2017

When joining Sailors and Reserves, some of the rows from Sailors will not appear as part of the result

ב>Showcase זה אם נעשה JOIN נאבד חלק מהשורות (אלו שמסומנות באדום), אחרי פעולה הצירוף הם פשוט יעלמו, ולפעמים זה לא מה שאנו רוצים. לשם כך נועד **NATURAL LEFT OUTER JOIN**

```
SELECT Sailors.sid, Reserves.bid
FROM Sailors NATURAL LEFT OUTER JOIN Reserves
```

כמו צירוף טבעי, אבל חיצוני שמאלית – בתוצאה יהיה את כל השורות של הצירוף של הטבלה השמאלית עם הימנית, ובונוסף כל שורה בטבלה השמאלית שלא הטרפה עם שורה של Reserves נתקבלות עם ערכי NULL בעמודות הקשורות לטבלת Sailors:Reserves

sid	bid
22	101
31	NULL
58	103
70	NULL

[Right Outer Join, Full Outer Join](#)

Left outer join of S and R contains:

- 1) all the tuples in the join of S and R
- 2) all the tuples in **S** that did not join padded with null values

Right outer join of S and R contains:

- 1) all the tuples in the join of S and R
- 2) all the tuples in **R** that did not join padded with null values

Full outer join of S and R contains:

- 1) all the tuples in the join of S and R
- 2) all the tuples in **S** that did not join padded with null values
- 2) all the tuples in **R** that did not join padded with null values

```
SELECT S.sid, count(bid)
FROM Sailors S NATURAL LEFT
OUTER JOIN Reserves R
GROUP BY S.sid
```

מה השאלה זו מחשבת? צירוף טבעי בין Sailors ו-Reserves ומשארה לנו שורות הטרפו. אחר כך מחלוקת לקבוצות לפי sid וספירת כמה ערכי bid יש. עבור ימאים שלא הגיעו אף ספינה נקלט NULL, ובוורם נספר אתכמות השורות ונקבל 0 אז השאלה תחזיר לכל ימאי בדיק כמה ספינות הוא החזיר, ועבור אלו שלא הגיעו אף ספינה נקלט 0.

leshim lab – אם הינו עושים רק ח�וֹיָן Natural לא הינו מקבלים את אלה שלא הגיעו אף ספינה, כי הם היו נמחקים בצירוף הטבעי מאחר שלא מופיעים בטבלה R.

SQL 9: Modifications 4.3

שינויים לשורות קיימות ב-DB

השאלות לא משנות את הטבלאות, אבל לפחות נרצה לעדכן את הנתונים בטבלאות.

INSERT INTO

אם נרצה להכניס שורה חדשה לטבלה SAILORS נעשה זאת כך:

Sailors			
<u>sid</u>	sname	rating	age
22	Alice	7	45
31	Barbara	8	55.5
58	Carol	10	35
70	Alice	10	25
84	NULL	10	NULL

```
INSERT INTO Sailors (sid, rating)
VALUES (84,10)
```

בשודות שבhem לא הגדרנו איזה ערך נכנס, ייכנס NULL באופן דיפולטיבי. מה יקרה עבור השאלה הבאה?

?

```
INSERT INTO Sailors (sname, rating)
VALUES ('Danna',10)
```

מערכת ה-DB תנסה להכניס ערך NULL עבור sid ועבור age. לגיל אין בעיה, אבל לאחר ש-sid זה מפתח, אסור להכניס שם ערך NULL.

DELETE FROM

דיברנו על פקודה DROP TABLE שמוחקת טבלה שלמה, אבל מה אם נרצה למחוק רק חלק מהשורות?

Sailors			
<u>sid</u>	sname	rating	age
22	Alice	7	45
31	Barbara	8	55.5
58	Carol	10	35
70	Alice	10	25

```
DELETE FROM Sailors
WHERE rating > 9
```

DELETE FROM Sailors
WHERE sid not in (SELECT sid
FROM Reserves)

אפשר לבתוב ב-WHERE כל תנאי שורצים, מורכב בכל שהוא. למשל השאלה האדומה תמחק מ-Sailors את כל מי שהזמין ספינה. אם נרצה למחוק את כל השורות בטבלה, אפשר לבתוב ; DELETE FROM Sailors ; וזה ימחק את כל השורות, בלי למחוק את הטבלה עצמה (כמו DROP).

Sailors

sid	sname	rating	age
22	Alice	7	45
31	Barbara	8	55.5
58	Carol	10	35
70	Alice	10	25

```
UPDATE Sailors
SET rating = rating - 1
WHERE sid = 31
```

```
UPDATE Sailors
SET rating = 0
WHERE sid not in (SELECT sid
                   FROM Reserves)
```

אם נרצה לעדכן ערכים בטבלה, נשתמש ב-UPDATE, ונקבע באמצעות SET את הערך החדש.

SQL 10: Views 4.4

[מה זה View](#)

A **view** is a **virtual table** defined by a query.
It can be used anywhere that a table is used.
It is computed every time it is used

אפשר להגיד שזו טבלה וירטואלית שמוגדרת על ידי שאלתה, בפועל היא לא באמת קיימת בטבלה, אלא במאקרו – הגדרה של שאלתה שנשמרת על ידי הdb. כל פעם שנתייחס לשם זהה, השאלתה זו תבוצע מחדש. זה מתרחש על ידי:

```
CREATE VIEW <view-name>
AS <query>;
```

למשל:

```
CREATE VIEW GreatSailors AS
SELECT sid, sname
FROM Sailors
WHERE rating >= 9
```

ואז יוכל להשתמש בה כמו בטבלה רגילה:

```
SELECT sid
FROM GreatSailors
WHERE sname = 'Joe'
```

אבל מאחורי הקלעים מערכת-h-db מחשבת את השאלתה שモקפת בצחוב.

שימושים ל-View

מתי משתמש בזה?

- (1) שאלתה שאנו מעתה מושתמשים בה שוב ושוב והיא מורכבת לבתיה, אפשר להגיד אותה פעם אחת ואז להשתמש בה הרבה פעמים.
- (2) אבטחת מידע – אפשר לתת למשתמשים לגשת לתוצאות של View שהגדרכנו במקום לתת להם לגשת לטבלה כולה.

דוגמה ל-(1)

```
CREATE VIEW RedBoats AS
SELECT bid, bname
FROM Boats
WHERE color = 'red'
```

```
SELECT DISTINCT sid
FROM Reserves
WHERE bid in (SELECT bid from RedBoats)
```

```
SELECT DISTINCT count(*)
FROM RedBoats
```

בכל פעם שנתענין בספינות האדומות נוכל להשתמש ב-RedBoats.

דוגמה ל-(2)

```
CREATE VIEW SailorInfo
SELECT sname, sid, age
FROM Sailors
```

```
grant SELECT on SailorInfo to shimon;
```

נותנים הרשות בעזרת פקודה grant, מה שבירוק יתן הרשות לשמעון לבצע פקודות SELECT על SailorInfo.

Modifying Views

לפעמים מתאפשר גם לבצע שינויים על Views שהגדכנו:

Sometimes it is possible to insert into, delete from, or update, a view !!!

Modifications are actually performed on the underlying base table

Modifications are possible only when the view is **updatable**

פעולה עדכון על View מתבצעים **על הטרבלה שעליה מוגדר ה-View**. אפשר לעשות זאת רק כאשר ה-View מוגדר להיות updatable. באלו תנאים השאלה של ה-view צריכה לעמוד כדי שהוא יהיה ?updatable?

The defining query must have exactly one table in the FROM clause

The defining query must not contain one of the following clauses at top level: GROUP BY, HAVING, DISTINCT, UNION, INTERSECT, and EXCEPT.

The selection list must not contain any aggregate function

דוגמה למחיקה מתוך View

```
CREATE VIEW GreatSailors AS
SELECT sid, sname
FROM Sailors
WHERE rating>=9
```

```
DELETE FROM GreatSailors
WHERE sname = 'Alice'
```

sid	sname	rating	age
22	Alice	7	45
31	Barbara	8	55.5
58	Carol	10	35
70	Alice	10	25

```
DELETE FROM Sailors
WHERE sname = 'Alice' and rating>=9
```

מה שיש בירוק זה התרגום שמתבצע מאחורי הקלעים על ידי מערכת db.

דוגמה לעדכון מתוך View

```
CREATE VIEW GreatSailors AS
SELECT sid, sname
FROM Sailors
WHERE rating>=9
```

```
Update GreatSailors
Set sname = 'Aviva'
WHERE sname = 'Alice'
```

sid	sname	rating	age
22	Alice	7	45
31	Barbara	8	55.5
58	Carol	10	35
70	Alice	10	25

```
Update Sailors
Set sname = 'Aviva'
WHERE sname = 'Alice' and rating>=9
```

דוגמה להכנסה לתוך View

```
CREATE VIEW GreatSailors AS
SELECT sid, sname
FROM Sailors
WHERE rating>=9
```

```
INSERT INTO GreatSailors
VALUES(113, 'Danna')
```

sid	sname	rating	age
22	Alice	7	45
31	Barbara	8	55.5
58	Carol	10	35
70	Alice	10	25
113	Danna	NULL	NULL

```
INSERT INTO Sailors(sid, sname)
VALUES(113, 'Danna')
```

דרך ה-view אנחנו לא חשופים ל rating, age ולקן שודות אלו יקבלו את הערך NULL. נציין שזה קצת שונה, כי הכנסנו את דנה לתוך sid, GreatSailor, אבל נשים לב שהיא לא תחשב GreatSailor כי ערך rating שלו הוא NULL ולבן בפרט לא 9 או גבוה יותר. אם רוצים אפשר לבצע הגבלות על db כך שלא יאפשר הכנסות מהסוג הזה.

טבלה זמנית

טבלה שגדירים לצורך חישוב שאלתה מסוימת, מיד בסוף החישוב היא תמחק מה-db.
כדי להגדיר אותה נשתמש בפורמט הבא עם המילה השמורה WITH:

**WITH <table-name> as (<query>
<another-query>)**

הערה: בשונה מ-View שבה מגדירים אותו פעמי אחד ב-db והוא לא מחושבת בטבלה אלה כאמור של שאלתה שנוכל להשתמש בה פעמי נוסף, כאן הטבלה הזמנית נוצרת רק עד לסיום חישוב השאלה [<another-query>](#).

דוגמה 1

```
WITH GS(sid) as (SELECT sid from Sailors where rating=10)
SELECT bid
FROM GS, Reserves R
WHERE GS.sid = R.sid
```

מכילה את הימאים עם דירוג מעל 10, ואז מחשבים צירוף טבעי על התנאים ב-WHERE. מקבלים ספינות שהוזמנו על ידי ימאים עם דירוג 10.

דוגמה 2

```
SELECT color
FROM Boats B
GROUP BY color
HAVING count(bid) >= ALL (SELECT count(bid)
                             FROM Boats
                             GROUP BY Color)
```

קודם כל מחלקים את השורות בתוצאה לפי צבע, כל הספינות באותו צבע יופיעו נמצאות ביחד בקובץ.
לאחר מכן מופיע תנאי על הקבוצה בצהוב – זה מחייב לכל אחד מהצבעים אתכמות הספינות שיש לו צבע. התנאי בירוק דורש שכמות הספינות באותו הספציפי יהיה גדול או שווה לכמות הספינות בכל אחד מהצבעים האחרים – ככלMORE יחייב את הצבעים הכל פופולריים (YSIS מכבי הרבה ספינות).

דרך נוספת וסקולה לבנות את השאלה:

```
WITH ColorCnt(color, num) as
  (SELECT color, count(bid)
   FROM Boats
   GROUP BY color)
SELECT color
FROM ColorCnt
WHERE num = (SELECT max(num) from ColorCnt)
```

הגדרנו טבלה זמנית ColorCnt שמכילה לכל צבע אתכמות הספינות באותו צבע, ואז ביצענו שאלה על הטבלה הזמנית שיצרנו.

הגדרה רקורסיבית של טבלאות זמניות

כשאנחנו מגדירים טבלה רקורסיבית אנחנו כתבים במפורש :

```
WITH RECURSIVE <table-name> as
  (SELECT ...
   ...
   UNION [ALL]
   SELECT ...
   ...
   <another-query>)
```

The diagram illustrates the recursive part of a query. It shows a green arrow pointing to the first part of the query (the non-recursive term) labeled "Non-recursive term". A red arrow points to the UNION or UNION ALL operator labeled "UNION or UNION ALL". A purple arrow points to the recursive part of the query labeled "Recursive term".

החלק הראשון (בירוק) יהיה לא רקורסיבי, ולא מתייחסת בחזרה לשם הטבלה הזמנית שאנו מגדירים, לאחר מכן UNION או ALL UNION או ALL וודאי שאליה נספה שיכולה בבר להתייחס לשם הטבלה הזמנית שעובדיו הגדרנו.

דוגמה 3

```
WITH RECURSIVE t(n) AS (
    VALUES (1)
    UNION ALL
        SELECT n+1 FROM t WHERE n<5)
SELECT sum(n) FROM t
```

באן בירוק למעשה ניצור שורה עם הערך 1.

בסגול יש את ההגדרה הרקורסיבית.

מה השאלה מחשבת?

ביצוע שאלה רקורסיבית

שלב 1 – מחשבים את החלק הלא רקורסיבי, אם בתוכו UNION ואם החלק הרקורסיבי חוזר לנו בלי כפליות נמחק את הכפליות. את השורות שקיבliśmy נשים בשתי טבלאות זמניות. אחת זו הטבלה הסופית שם יהיו כל השורות שנছזר בסוף, ואחת "טבלת עבודה". עברו הדוגמה לעיל:

Table t, final result	Table t, working table
n 1	n 1

שלב 2 – כל עוד טבלת העבודה לא ריקה, נחשב את החלק הרקורסיבי, ובאשר היא מתייחסת לטבלה הזמנית אנחנו משתמשים בתוכן שבכתב ב-table working table בשביל הגדרת הטבלה הזמנית. אחרי שהיחסנו אותה, אם יש UNION נמחק כפליות (הן במאשנוצר, והן עם התוצאות הקודמות שייצרנו) ומה שמחוברים נשים גם ב-table final result וגם ב-table working table נמחק את מה שהוא ובמקרה נשים את התוכן.

למשל בדוגמה שלנו, נסתכל על טבלת העבודה – יש בה את הערך 1. מתקיים $1 > 5$ ולכן נוסיף לטבלה הסופית את הערך $1+1=2$. נוסיף את 2 שוב לטבלת העבודה ונמחק את הערך הקודם שהוא בה (1). נחשב שוב עד שיפסיק לקיים את התנאי $5 < n$.

n 1	n 1
2	1
3	1 2

וכן הלאה..

דוגמה 4

Edge(U,V)

U	V
1	2
2	3
3	4

```
WITH RECURSIVE Reach(U,V) AS (
    SELECT * FROM Edge
    UNION ALL
        SELECT Reach.U, Edge.V
        FROM Reach, Edge
        WHERE Reach.V = Edge.U)
SELECT * FROM REACH
```

Reach,
final result

U	V
1	2
2	3
3	4

Reach,
working
table

U	V
1	2
2	3
3	4

אחרי מעבר אחד קיבל:

Edge(U,V)	
U	V
1	2
2	3
3	4

```
WITH RECURSIVE Reach(U,V) AS (
    SELECT * FROM Edge
    UNION ALL
    SELECT Reach.U, Edge.V
    FROM Reach, Edge
    WHERE Reach.V = Edge.U)
SELECT * FROM REACH
```

Reach,
final result

U	V
1	2
2	3
3	4

Reach,
working
table

U	V
1	2
2	3
3	4

עבדיו נמוך את כל מה שהוא לפניו בטבלת העבודה (מה שלא כתוב בכתב יד אדום) ונשאר רק עם הערכים החדשניים שהגדרכנו. נמשיך לבדוק גם עבורם. הזוג החדש היחיד שנוסף הוא 4 1 ונקבל את כל הזוגות האפשריים:

U	V
1	2
2	3
3	4

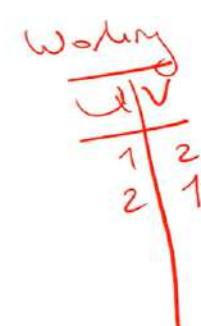
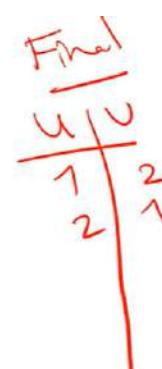
דוגמה 5

Edge(U,V)	
U	V
1	2
2	1

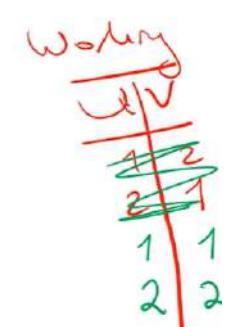
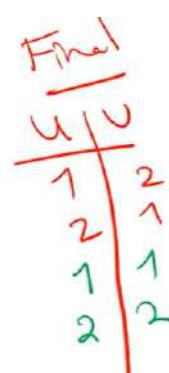
```
WITH RECURSIVE Reach(U,V) AS (
    SELECT * FROM Edge
    UNION ALL
    SELECT Reach.U, Edge.V
    FROM Reach, Edge
    WHERE Reach.V = Edge.U)
SELECT * FROM REACH
```

כאן יש מעגל בגרף. מה קיבל בתוצאה?

בהתחלתה:



אחרי מעבר אחד קיבל:



עבשו נמשיך ונקבל:

Final	
1	2
2	1
1	1
2	2
1	2
2	1

Working	
1	2
2	1
1	1
2	2
1	2

נשים לב שהשתמשנו ב-ALL UNION ולכן מושאים כפליות. אפשרobar לשימוש ב-UNION לולאה אינסופית כי חזרנו למצב ההתחלתי.
נקבל כאן שגיאה מה-dp.

מה היה קורה אם היה בתוכן UNION ולא ALL UNION?

בשחינו באים להוסיף את 2 ו-1 ו-2 שכתובים בתכלה, הינו מוחקים אותם כי הם כפולים ולכן גם מטבלת העבודה, אז היא הייתה ריקה והיינו מסיים, ומתקבלים את הטבלה FINAL שמכילה רק את השורות שכתובות באדום ובירוק.

Week 5: Indexes

IND 1: Intro to Query Processing 5.1

היררכיות מידע

זיכרון הדיסק הוא גדול יותר ונשמר לפחות זמן. הזיכרון הראשי מהיר יותר בהרבה. בשימוש בשתי זיכרונות בדיסק מודע בדיסק מעבירים אותו לזיכרון הראשי ביחידות של page ותמיד נעביר דף שלם.

נכיה שלא ניתן לחתת ל-tuples בשורות להיות ביוטר מבלוק אחד. דוגמה:

Example

Suppose that

1. a block contains **1024 bytes**
2. table T has **100,000 tuples**
3. each tuple requires **100 bytes**
4. tuples may not span blocks

How many blocks
are needed for T?

$$100,000 / \text{floor}(1024 / 100) \\ = \\ 10,000 \text{ blocks}$$

How much space
is "wasted"?

$$(1024 - \text{floor}(1024 / 100) * 100) \\ * 10,000 = \\ 240,000 \text{ wasted bytes}$$

הרבה פעמים הזמן להעביר את המידע מהזיכרון לדיסק הוא שימושית יותר ארוך זמן העיבוד שלו בזיכרון הראשי. מתקבל לשימוש במודל חישוביות שתמודוד רק את הסיבוכיות של פעולות O(/) בתוכנית.

I/O Complexity

אנחנו נספר כמה בלוקים קראנו מהדיסק לזיכרון הראשי, וכמה בלוקים כתבנו מהזיכרון הראשי לדיסק. לא נספר כמה פעולות O(/) נדרשו לבכנת התוצאה הסופית (מאחר שהיא לרוב מודפסת למסך ולא נשמרת בדיסק). דוגמה:

Example

Suppose that

1. a block contains **1024 bytes**
2. table T has **100,000 tuples**
3. each tuple requires **100 bytes**
4. tuples may not span blocks

How many blocks
are needed for T?

$$100,000 / \text{floor}(1024 / 100) \\ = \\ 10,000 \text{ blocks}$$

What is the I/O Complexity of
SELECT * FROM T?

cost of read = **10,000**

נשים לב שלא החשבנו גם את 10,000 פעולות הבכנה, כי לא שמרנו אותן בדיסק אלא רק הדפינו אותן למסך. דוגמה נוספת:

Suppose we want to sort the data in
pages 1, 2, 3, given a 4 page buffer.
What would be the I/O complexity of this sort?

אנחנו רגילים שמיון n ערכים עולה $n \log n$, אבל כאן זה לא מה שאנו מוחפשים, רק בודקים כמה עולה להעביר 4 דפים מהדיסק לזיכרון הראשי, ולכן המחיר הוא 4 פעולות. נשים לב שהפעולה של מיין חלקים גדולים מטור הדיסק היא קשה יותר במקרה שבזיכרון הראשי אין מספיק מקום לכל הערכים, ונענה על השאלה הזאת בהמשך.

[ארגון טבלאות בדיסק: קובץ עריםה](#)

קובץ עריםה הוא אוסף של קובצי טבלה חדשים שוריה מוגעה לטבלה אנחנו מוסיף אותה לסוף קובץ העריםה. למשל:

Sailors							
sid	sname	age	rating	sid	sname	age	rating
123456	Alice	20	10	234695	Danna	44	6
666666	Barbara	54	7	988327	Francis	37	8
234876	Carol	37	3	293894	June	89	5
287843	Alice	89	10	777883	Erica	65	10

Page 1

Page 2

בכל דף בדוגמה יש מקום ל-4 שורות, אם תתווסף עוד שורה נצטרך בלוק חדש. לצורך פשטות נניח שככל השורות בטבלה לוקחות את אותו מקום. אם שמרנו את הטבלה שלנו בקובץ עריםה, כמה זמן ייקח לעשות פעולות בסיסיות מול הטבלה?

(1) **אם נרצה למצוא שורה עם ערך SID מסוים?** במקרה הנוכחי נצטרך לקרוא את כל הדפים. אנחנו מודדים זמן O(N) אך אם הטבלה שמורה ב- N דפים אנחנו צריכים N פעולות קריאה.

(2) **כמה זמן ייקח להוסיף שורה לטבלה?** אנחנו צריכים לרשום אותה לתוכן הבלוק האחרון, או אם נגמר בו המקום לפתח בלוק חדש. כדי לבתוב שורה בבלוק האחרון צריך קודם לקרוא את הבלוק האחרון ליצורו המרבי, להוסיף את השורה החדשה, ואז לבתוב לדיסק. נשים לב שלמרות שהוספנו רשותה אחת לבלוק זהה – אין לנו אפשרות לבתוב רק שורה לדיסק, צריך לבתוב את כל הבלוק מחדש. אם הכנסנו מפתח, צריך גם לוודא שלא סתרנו את אילוץ היחידות שלו, אז זה דורש גם קריאה של כל הבלוקים, ורק אחרי מוכן פנימה. לכן אם יש מפתח כמות פעולות ה-O($N+1$) היא 1+N.

(3) **כמה זמן ייקח למחוק שורה מהטבלה?** כדי למצוא אותה אנחנו צריכים לקרוא את הבלוקים, זה עולה N פעולות, ואז לבתוב את הבלוק ללא השורה שרצינו למחוק זו ועוד פעולה אחת. סה"כ $O(N+1)$ פעולות. נשים לב שגם למשל הטבלה שלנו מקושרת לטבלה אחרת, בשאיי מוחק שורה מהטבלה שלו אני צריך לוודא גם מה קורה בטבלה המקושרת.

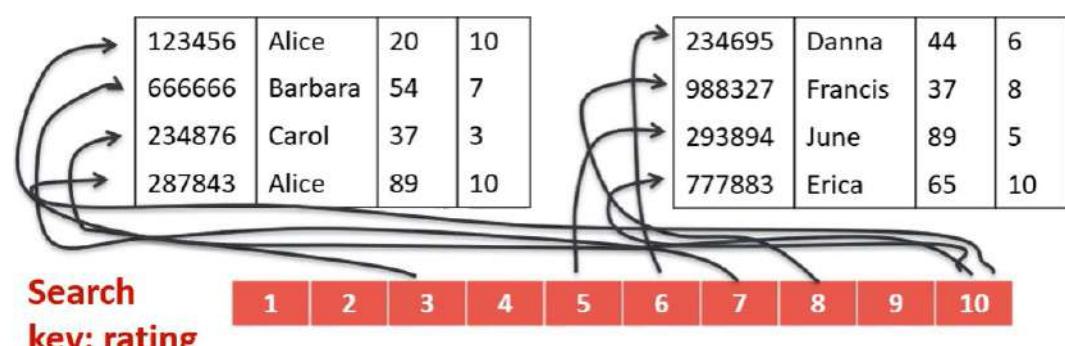
לכן אם אנחנו משתמשים בקובץ עריםה, כל פעולה שנרצה לעשות צריכה את קרואת כל הטבלה. ב-data גודל קריאה של כל הקובץ זו פעולה איטית. אם היינו שומרים את הקובץ ממיון זה היה יכול ליעל את הזמן, כי בעת פעולה יכולנו לחסוך את חלק מהקריאה ולהגיע מהר יותר לאזרור שאנו חסרים. הבעיה היא שעבודה עם קבועים ממויינים מבקשת על הכנסת מידע לטבלה, כי צריך לדוחף שורות חדשות בך שלא יופר המילוי, וזה ידרש לפחות לבתוב מחדש את כל הבלוקים כי נדרש לדוחף את המידע שורה קדימה. לכן בפועל הקבועים לא שומרים בצורה ממויינת חוץ מבארש:

(1) אם יש מילוי טבוי שנוצר בעת הכנסה כמו למשל Log של מערכת, שבאופן טבעי נספנותו לו שורות רק עם מספר סידורי גדול יותר (מוסיפות שורות בכל שעובר זמן).

Index organized files (2)

[Index: Intuition](#)

בכל זאת נרצה לגשת לצורה מהירה. לשם כך יש **מבנה אינדקס**, מבנה נתונים שמוגדר מעל הטבלה ונוטן לנו דרך מהירה לגשת לשורות בהן אנחנו מעוניינים בלי לגשת לטבלה כולה. מבנה האינדקס הוא למשל hash table או עצ' חיפוש בינארי, והמבנה מצביע על השורות בטבלה. מפתח החיפוש במבנה הוא השדה שלפיו אנחנו רוצים לגשת לטבלה. בדוגמה שלפנינו, נרצה למשל לחפש ומאים לפי דירוג:

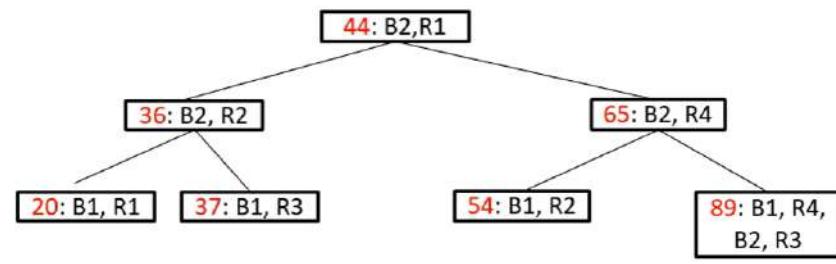


דוגמא נוספת היא עצ' חיפוש בינארי עם מפתח חיפוש לפי גיל:

123456	Alice	20	10
666666	Barbara	54	7
234876	Carol	37	3
287843	Alice	89	10
234695	Danna	44	6
988327	Francis	36	8
293894	June	89	5
777883	Erica	65	10

Block 1

Block 2

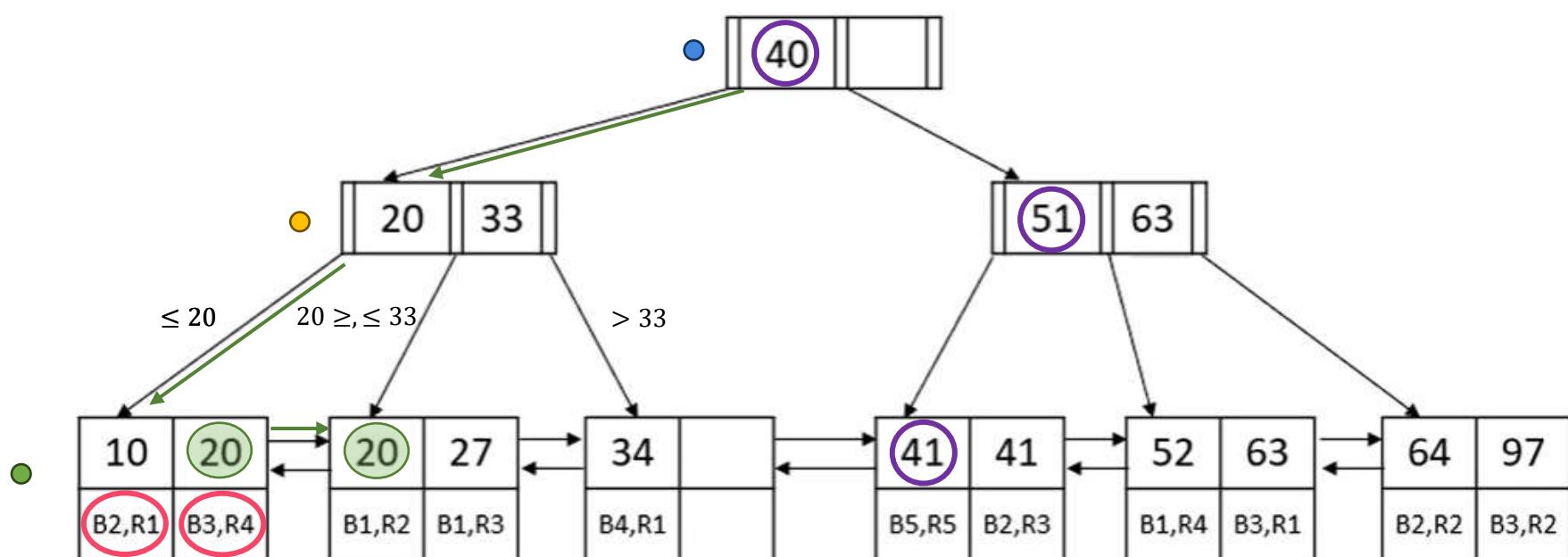


הצior מטעה, כי מבנה אינדקס רגיל הוא לא עז בינהר, וגם אנחנו שומרים את כל הכתובות בעליים, ולא נשמר כתובות לכל אורך הדרכ. כמו כן, נשים לב לעלה עם הערך 89 שמצביע לשתי שורות – גם זה לא יקרה, בפועל יהיה פיצול של ערכים כפולים.

IND 3: B+ Trees 5.3

[Index Structure: B+ Trees](#)

עג+B זו הגרסת הפופולארית ביותר של מבנה אינדקס:



אנחנו רואים **שורש**, **עלים** ו**קוודקודים פנימיים**. זה עג מאוזן, המרחיק מהשורש לכל אחד מהעלים זהה. העלים הם רשימה משורשתה דו-ביוונית. בכל קוודקוד יש לנו מספר ערכים והעלים ממשויים. יש לנו **מפתח חיפוש**, **ערכים/שורות שעליהם מצביעים**.

בשכבה העלים, לכל שורה בטבלה יש ערך מתאים לו בעלה. איך נחפש ערך בעץ? נניח שאנו ממעוניינים בערך 20. מתחילה בשורש שהוא 40, והעג מאוזן אז לאחר שמחפשים מספר קטן יותר נלך שמאליה. בקוודקוד הבא שהוא (20,33) אנחנו נרצה להגיע ל-20. החץ השמאלי הוא כל מה שקטן/שווה מ-20, החץ הימני בין 20 ל-33, והימני גדול 33. נחפש את המצביע הראשון שמתאים לנו לחיפוש. זה החץ השמאלי, ואז נטיל ימינה ונמצא את השורה הבאה שבאה נמצאת 20. המסלול מסומן בעיגול.

אם הערך מופיע רק פעם אחת הוא בהכרח יופיע הצד השמאלי, ומובטח לנו להגיע לערך אם הוא נמצא. אם יש כמה מופעים שלו נמשיך ימינה בשכבה העלים.

כמה זמן עולה לחפש בעץ? בגובה העץ.

[תבניות של עג+B](#)

A B+ Tree of **branching factor** d , has the following properties:

1. All leaf nodes are at the same level
2. All nodes have at most d children
3. All non-root nodes have at least $[d/2]$ children
4. A non-leaf node with k children, has $k-1$ values
5. All keys values within a node are in ascending order

(1) כל העלים באותו רמה.

(2) לכל קודקוד יש לכל היותר p ילדים כאשר דרגת הפיצול היא p .

(3) מובטח לנו שלכל קודקוד מלבד השורש יש לפחות $\left\lceil \frac{d}{2} \right\rceil$ ילדים.

(4) כל קודקוד שיש לו k ילדים יהיו לו $1 - k$ ערכים.

(5) כל המפתחות בתחום הקודקוד מסודרים בסדר עולה.

איך מחשבים את המספר המינימלי והמקסימלי של עליים?

למשל עבור דרגת עומק $d = 683$, עם גודל entry בכל עלה עם מפתח בגודל 4 ומצוין בגודל 8 (כלומר כל entry בגודל 12). בהנחה שגודל הבלוק הוא 2192 נקבל:

$$\text{block size} = 8192 \text{ bytes}$$

$$\text{size per entry} = 12 \text{ bytes}$$

$$\text{entries per leaf node} = 683 - 1 = 682$$

$$\text{ולכן מספר המינימלי של עליים הוא } 15 = \left\lceil \frac{10,000}{682} \right\rceil = [14.66]$$

$$\text{המספר המקסימלי של עליים הוא כאשר כל קודקוד מכיל אתכמות המינימלית של ילדים, שהוא } 341 = \left\lceil \frac{683}{2} \right\rceil = [341]$$

$$\text{ואז מספר העליים } 30 = \left\lceil \frac{10,000}{341} \right\rceil = [29.32]$$

איך נבחרת דרגת הפיצול?

אין לנו שליטה עליון, זה נבחר על ידי מערכת ה-DB. העץ שומר על הדיסק. בכל פעם שנעבור דרך מצביע בעץ, אנחנו מבצעים קריאת דיסק (אך מצביע העץ בתובת של בלוק בדיסק ששם נמצא הקודקוד הבא). אם יש דרגת פיצול גבוהה, העץ פחות עמוק וכן פחות קריאות דיסק ובכך העבודה היא גבוהה יותר. מערכת ה-DB תבחר את דרגת הפיצול המקסימלית שתאפשר לה עדין לשמר קודקוד בבלוק אחד.

דוגמא

Suppose

- Integer search key: 4 bytes
- Pointer: 8 bytes
- Block size 1024

איך נבחר את דרגת הפיצול d ? נסתכל על הקודקוד הגדל ביותר שבו ניתן שיכל להיות לנו ונוויד ששהוא יכול להיבtnס בתחום הבלוק. בקודקוד הגדל ביותר יהיה לנו d ילדים, ככלומר $8d$ בתים עבור מצביעים ילדים. יהיו לנו $1 - d$ ערכי חיפוש, ככלומר $(1 - d)4$ בתים לערך החיפוש.

צריך שיתקיים: $1024 \leq 8d + 4(d - 1)$. נקבל $85 \leq d$. אך נבחר שלכל קודקוד יהיה לפחות 85 בתים, ולכן הפחות 43 = $\left\lceil \frac{85}{2} \right\rceil$ בתים.

עלות O/I שנשלם בחיפוש

Step 1: Find first relevant leaf

Cost 1: Depth of tree

Step 2: Read all relevant leaves

Cost 2: Depends on number of relevant values + number of values per leaf node

עלות ראשונה היא עומק העץ כדי למצוא את העלה הראשון הרלוונטי.

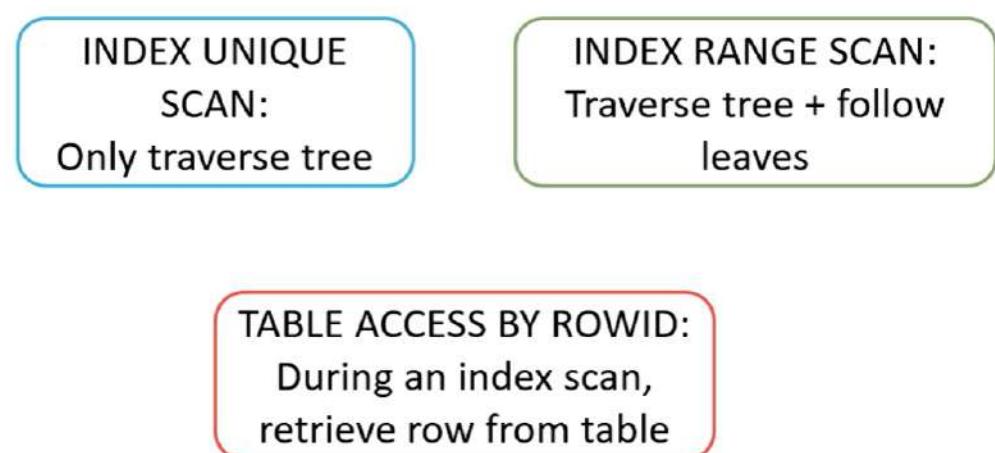
עלות שנייה היא חיפוש ימינה על כל העליים שיש להם את הערך הנדרש.

Index Maintenance

כדי שאינדקס יהיה יעיל צריך לשמר עליו מאזור בהכנסות/עדכונים/מחיקות. אפשר לעשות את כל עבודות התחזוק בזמן פרופורציונלי לגובה העץ.

השוואת שאילותות

יש 3 דרכי שונות להשתמש במבנה אינדקס במהלך חישוב שאלתה:

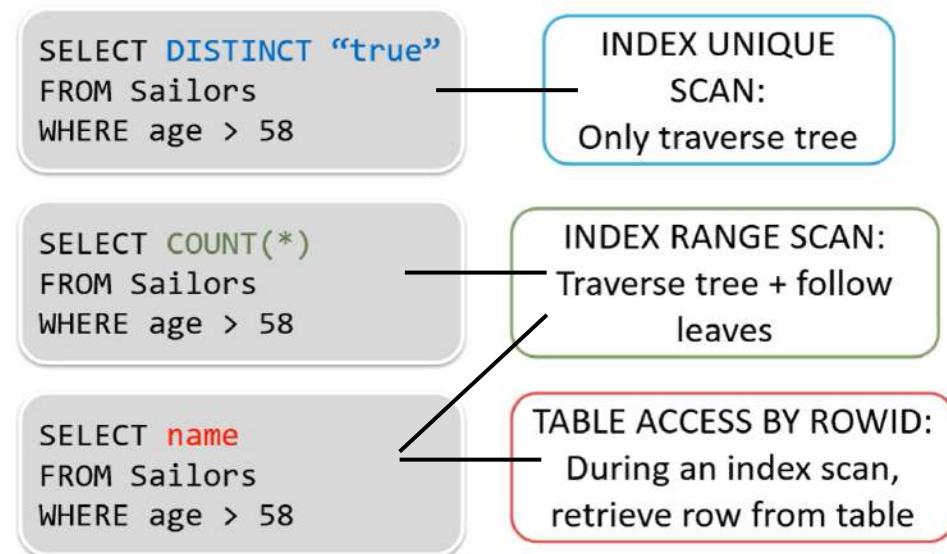


בשיטת הכחולה, אנחנו לכל היותר צריכים מעבר אחד משורש האינדקס עד לעלה אחד, ולא נדרש לעבור ימינה/שמאליה בעלים נוספים. למשל לחפש מפתח של טבלה באינדקס, הוא יופיע פעם אחת.

בשיטת **הירקה**, נבעור גם מהשורש לעלה, אבל אחר כך נמשיך שמאליה או ימינה בrama של העלים. למלש כל הימאים שגולם מעל 20, נctrיך לטיעל על הרבה עליים.

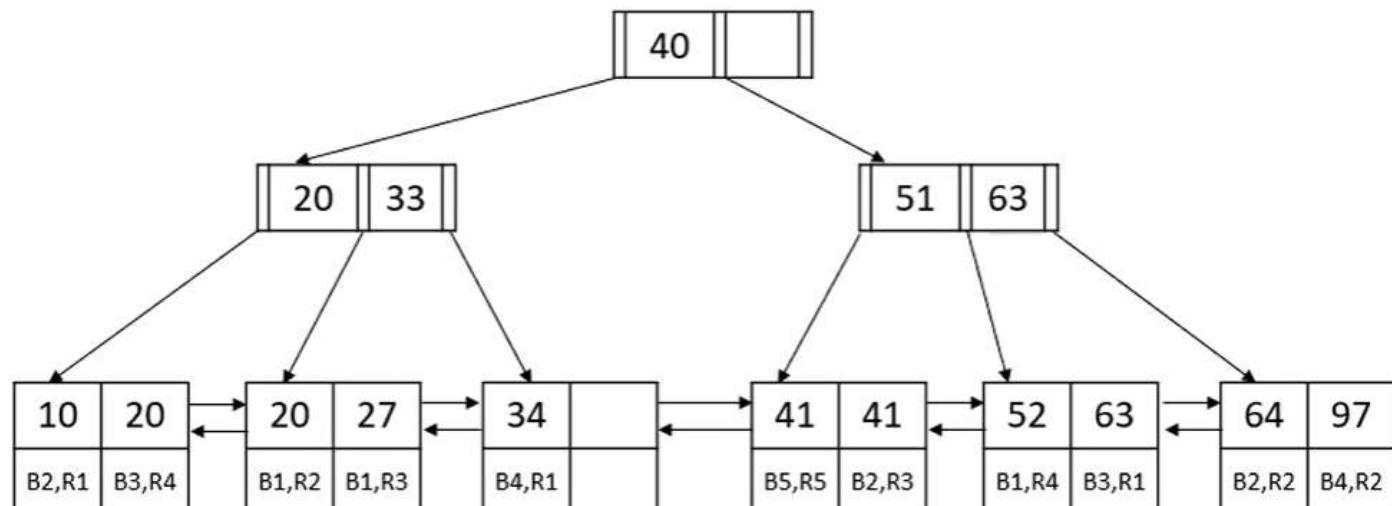
בשיטת האדומה, זה מגע בדרך כלל עם `index unique scan` או עם `index range scan`. נניח שהגענו לעלה, בו יש בתובת של שורה בדיסק. בשיטה זו נצטרך ממש ללבת לטבלה עצמה, לבלוק הספציפי ולבדוק את השורה הרלוונטית ואת הערכים בה. מבנה ה-`B+` שומר בדיסק בקובץ אחד, והטבלה בקובץ אחר. כשנចטריך לשולח שווה שלמה, נדרש לקרוא עוד בלוק מאותו ערך מהדיסק.

מתי נשתמש בכל סוג של סריקה?



דוגמָה

נכיה שזה עז מ-db על ימאים הממוני לפי age:



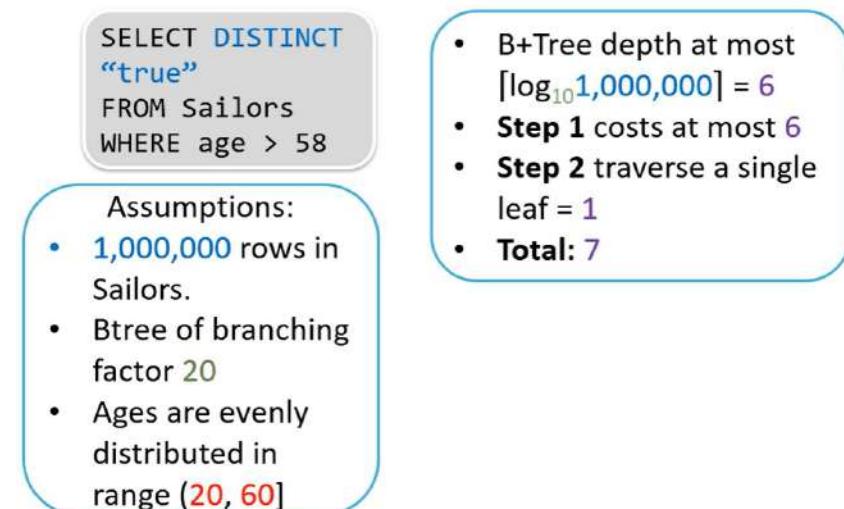
אם נרצה את שמות הימאים מעל גיל 58, נצטרך לרדת בעז, לקרוא 2 עליים, וגם לשלוֹף את בלוקים B3,B2,B4. 2 פעולות לירידה בעז, 2 פעולות לקריאת העליים, 3 פעולות לקריאת הבלוקים שמקיימים את התנאי. סה"כ 7 פעולות O/I. יתכן מצב שבו כמה מהשורות שאנו חזו צרכיים נמצאים כבר באותו בלוק, אבל בפרק הוגרע הם בבלוקים שונים ממו בדוגמה שלボן.

- Table with N rows
- B+Tree of branching factor d
 - each node has at least $\lceil d/2 \rceil$ children
 - each node has at least $\lceil d/2 \rceil - 1$ values
- k matching rows

בעץ כזה, העומק של העץ (במאות השכבות לא כולל העלים) היא לכל היותר $\left\lceil \log_{\frac{d}{2}} N \right\rceil$.

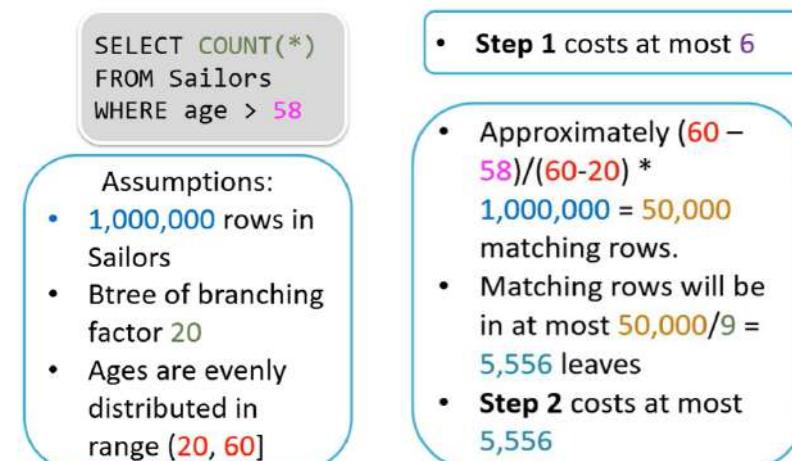
נניח שאנו רוצים לענות על שאלתה שמצריכה למצוא k שורות. נדרש לדמת את עומק העץ זהה לעלה לנו $\left\lceil \log_{\frac{d}{2}} N \right\rceil$. אחר כך נעבור על עלים שמכילים את k השורות התואמות. כמה עלים נמצאים אוטם? בכל עלה יש לפחות $1 - \left\lceil \frac{d}{2} \right\rceil$ ערכים, המידע שלו יייחו ב- $\frac{k}{\left\lceil \frac{d}{2} \right\rceil - 1}$ עלים לכל היותר. לכן טויל על העלים עלה $\frac{k}{\left\lceil \frac{d}{2} \right\rceil - 1}$. לבסוף, אם נרצה לקרוא את השורות המתאימות מתוך ה-db, זה עולה עד k פעולות O(1).

דוגמה 1

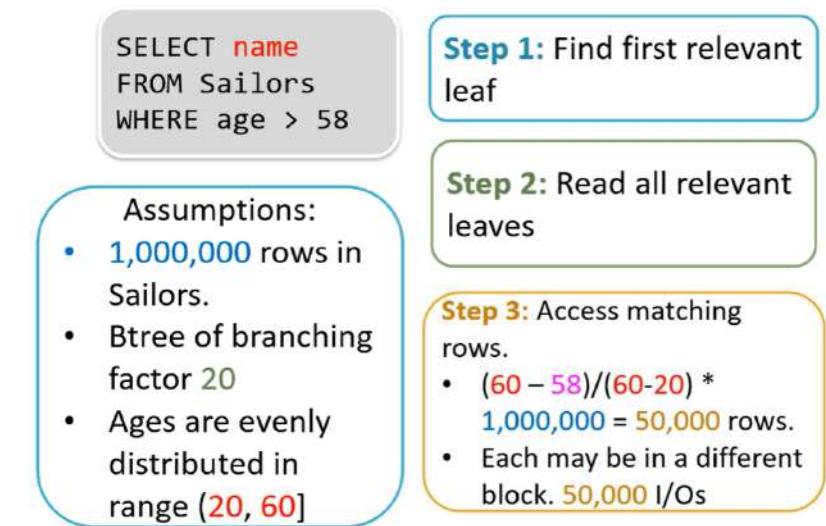


כמה עולה לנו לבדוק האם יש ימי שגילו מעל 58?
לדמת את עומק העץ יהיה $6 = \lceil \log_{10} 1,000,000 \rceil$. למה? לוג של גודל הטבלה על בסיס branching factor חלק 2.
נ לצורך להסתכל על עלה 1.
סה"כ 7 פעולות O(1).

דוגמה 2



כמו קודם, לדמת את העץ עולה 6 פעולות.
נעבור על כל העלים שיש להם ערך מעל 58. כדי להעיר את זה, נוכל להיעזר בכמויות השורות והתפלגות ערכי הגיל בשורות. אם נניח התפלגות אחידה בין 40 גילאים בין 20 ל-60, ואנו מעריכים ב-2 גילאים מתוכם ($59,60$) $\frac{2}{40}$ מילון שורות הם אלו שייתאימו, כלומר 50,000 שורות.
כעת נבון כמה עלים מכילים את 50,000 השורות הללו. מאחר שדרגת הפיזול היא 20, בפועל דרגת הפיזול המינימלית היא 10 וכאן כמות הערכים השונים שאנו מעריכים למצוא בעלה הוא לפחות $9 = \frac{50,000}{9}$ עלים לכל היותר. וזה המחיר של שלב זה.
סה"כ עלות $5,556 + 6 = 5,562$ פעולות O(1).



כמו מקודם, עולה לנו 6 פעולות לרוחט בעט, ו- 5,556 עליים שעשויים להכיל ערכים מתאימים. השוני הוא שעכשו לבל ערך מתאים אנחנו צריכים ללבת ולשלוף את השורה המתאימה מהבלוקים. כמה זה יעלה? יש לנו בערך 50,000 שורות מתאימות לתנאי, ובכל עשייתו להיות ב- 50,000 בלוקים שונים, סה"כ 50,000 פעולות. סה"כ עלות: $50,000 + 5,556 + 6 = 55,562$ פעולות.

האם תמיד כדאי להשתמש במבנה אינדקס?

הנה דוגמה:

```
SELECT avg(price)
FROM Orders
WHERE itemId = 'i242'
```

What is the best way to answer the query?

Suppose:

- Orders has 10,000 rows with 10 rows per block
- There is an index on itemId of branching factor 100
- Item i242 appears in 20 orders

אם לא משתמש באינדקס בכלל, וביצע סריקה מלאה של הטבלה זה יעלה $\frac{10,000}{10} = 1,000$. יש 10,000 שורות ובכל בלוק נכנסות 10, צריך לקרוא 1,000 בלוקים ולכן זו הูลות.

אם משתמש במבנה האינדקס נחשב:

רוחט למטה בעז

לעבור על כל העליים שמכילים את id itemId הנכון, ואפשר להעריך שהו יהיה אחד בלבד, כי יש 20 הופעות של id itemId שאנו מכחשים, ובכל עלה יהיו לפחות 49 ערכיהם, הם יכולים להיות בכל בלוק.

כל אחת מ-20 השורות נctrar לגשת לטבלה לשולוף את המחיר, זה 20 פעולות נוספת.

Using the index cost:

- Step 1: $\lceil \log_{50} 10,000 \rceil = 3$
- Step 2: 1, as all values of i242 will be in the same leaf
- Step 3: 20 (to access relevant table blocks)
- Total: 24

סה"כ יצא עדיף שימוש באינדקס. בואו נראה איך שינוי קטן משנה את המצב – נניח שה-*itemId* שונה מופיע ב-1,000 הזמינות. עבשו נקבל:

Using the index cost:

- Step 1: $\lceil \log_{50} 10,000 \rceil = 3$
- Step 2: $1,000/49 = 21$ leaves to traverse
- Step 3: 1,000 (to access relevant blocks)
- Total: 1,024

וכבר היה עדיף לא להשתמש באינדקס כלל.

מערכת ה-DB צrica להחליט באיזו שיטה להשתמש, וזה חלק מעבודת ה-DB שנדבר עליו בהמשך.

בחירה דרגת פיצול של העץ

Search key: s bytes
 Pointer: p bytes
 Block size: b bytes

Best branching factor:

$$\left\lceil \frac{b+s}{p+s} \right\rceil$$

כך יבטיח לנו שכל קודקוד יוכל להיבנס לבlok אחד של זיכרון, ואם נבחר מספר גדול יותר אז קודקוד יוכל להיות גדול מדי בשבייל בלוק.

גובה העץ

Number of table rows: n
 Branching factor: d

B+Tree height (at most):
 $\log_{\lceil d/2 \rceil} n$

תזכורת: כאשר דרגת הפיצול היא d לכל קודקוד חייב להיות לפחות $\lceil \frac{d}{2} \rceil$ בנים.

כמויות העלים שיכילו ערכים שמתאימים לתנאי החיפוש שלנו

Number of matching table rows: m
 Branching factor: d

Leaves containing matching values (at most):

$$\left\lceil \frac{m}{\lceil d/2 \rceil - 1} \right\rceil$$

כל קודקוד יכול לפחות $1 - \lceil \frac{d}{2} \rceil$ מפתחות שונים, ולכן m מפתחות זהה ייתן את כמות העלים המתאימה.

IND 5: Execution Plans 5.5

[AIR מייצרים אינדקסים?](#)

CREATE INDEX ON table_name ({ column_name, ... })

CREATE INDEX ON Sailors(age)

CREATE INDEX ON Sailors(age, rating)

ובהרבה:

Requires index entries to have unique values

Provides a name for the index

CREATE [UNIQUE] INDEX [name] ON table_name
 ({ column_name } | (expression)) [ASC|DESC]
 [, ...])

Allows the indexing of expressions, and not only columns

Only indexes tuples satisfying predicate

Determines ordering of index (ASC is default)

```
CREATE INDEX on Sailors(UPPER(sname) DESC)
WHERE rating > 5
```

יצרנו אינדקס על ה-UPPER CASE של השמות, סידרנו בסדר יורד, ושמרנו את שמות המשפחה עבורם הדירוג גדול מ-5.

יש אינדקסים שנבנו בצורה אוטומטית:

```
CREATE TABLE Employee(
    ID      INTEGER PRIMARY KEY,
    Fname   VARCHAR(20),
    Lname   VARCHAR(20),
    Gender  CHAR(1),
    Salary  INTEGER NOT NULL,
    Dept    INTEGER,
    UNIQUE(FNAME, LNAME)
);
```

בכל פעם שנגדירם מפתח על טבלה, מערכת ה-DB מייצרת אינדקס שבוצעתה היא מודדת שהמפתח נשאר ייחודי. כך כאשר נכניס למשל עובד חדש לטבלה זהו, מערכת ה-DB לא תסrox את בולה, אלא תשתמש באינדקס כדי לבדוק האם ID של העובד שהכנסנו כבר קיים ויזרק שגיאה אם כן. באופן דומה, אם נכרייז על שדה או אוסף שדות כ-UNIQUE, מערכת ה-DB תיצור אינדקס כדי לשמר את האילוץ הזה. קיבל הודעה:

```
NOTICE: CREATE TABLE / PRIMARY KEY will
create implicit index "employee_pkey" for
table "employee"

NOTICE: CREATE TABLE / UNIQUE will create
implicit index "employee_fname_lname_key"
for table "employee"
```

דרך לקבוע את השם שיינתן לאינדקס של מפתח

```
CREATE TABLE Sailors(
    sid      integer,
    name     varchar(20),
    age      integer,
    rating   integer,
    CONSTRAINT sailors_pk PRIMARY KEY(sid)
)
```

במקום להכרייז על sid כמפתח בצורה רגילה, הכרזנו עליו בשורה האחורה בדרך חדשה. כך השגנו את העבודה שיש אינדקס בשם sailors_pk שומר על התכונה sid הוא primary key. בכיה בעצם קבענו את השם של האינדקס.

דוגמה

```
SELECT name
FROM Sailors
WHERE sid = 22
```

INDEX UNIQUE
SCAN:
Only traverse tree

Which of these
are needed?

INDEX RANGE SCAN:
Traverse tree + follow
leaves

TABLE ACCESS BY ROWID:
After an index scan,
retrieve row from table

בשביל השאלה שאנו רואים ברגע נרצה: TABLE ACCESS BY ROW ID וגם INDEX UNIQUE SCAN. כשמייצרת ה-DB רוצה לחשב את השאלה יש לה 2 אפשרויות. או להשתמש במבנה האינדקס שבנו, או לעבור על הטבלה בוליה להשתמש באינדקס. אם נרצה לראות באיזו שיטה המערכת בחירה להשתמש, נוסיף את הפקודה EXPLAIN לפני השאלה. היא לא מבצעת שאלתה, רק מבקשת להסביר איך הפקודה שהבננו עומדת להיות מוצעת:

```
QUERY PLAN
-----
Index Scan using sailors_pk on sailors
(cost=0.15..8.17 rows=1 width=58)
Index Cond: (sid = 22)
```

נשים לב שהזיה נראה כאילו אין גם גישה לטבלה SAILORS, אבל זה כן מבוצע למחרות שזה לא בתוב (כ噫 חיב את name). יש שם גם נתונים על cost, ה-0.15 – זה ממה זמן ייקח למערכת להתקון לבצע השאלתה. המספרים אינם בשניות, זו סוג של הערכה יחסית. אם המספר גבוה תהיה בזירה עלות גבוהה. ה-8.17 זה עלות ביצוע הפעולה. הערך rows זה במתה השורות שהוא מצעה שיוחזרו, רוחב השורות width בbytes.

אפשר לבקש שהוא גם ממש תבצע את השאלה באופן הבא:

```

EXPLAIN ANALYZE
SELECT name
FROM Sailors
WHERE sid = 22

```

```

QUERY PLAN
-----
Index Scan using sailors_pk on sailors
(cost=0.15..8.17 rows=1 width=58)
(actual time=0.005..0.005 rows=0 loops=1)
Index Cond: (sid = 22)
Planning time: 0.123 ms
Execution time: 0.024 ms

```

מציא גם תוכנית ביצוע, וגם מחשב את השאלה ומນתח כמה זמן זה לוקח בפועל.

מערכת-DB יכולה להשתמש באינדקס גם אם השאלה יותר מורכבת:

```

EXPLAIN
SELECT name
FROM Sailors
WHERE sid = 22 and age = 35

```

```

QUERY PLAN
-----
Index Scan using sailors_pk on sailors
(cost=0.15..8.17 rows=1 width=58)
(actual time=0.005..0.005 rows=0 loops=1)
Index Cond: (sid = 22)
Filter: (age = 35)

```

המערכת עדין בוחרת להשתמש באינדקס ונוסף תנאי פילטר על השורת ששלפנו.

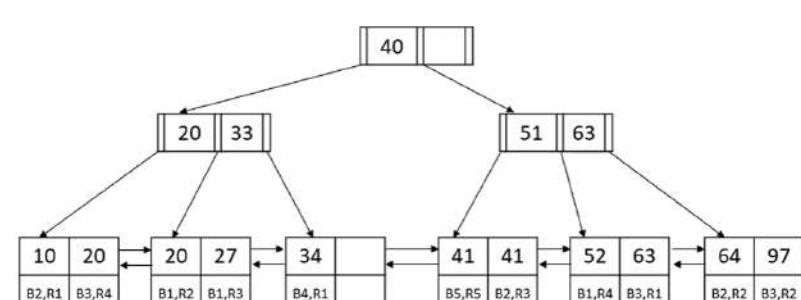
הערות חשובות

- (1) מערכת-DB לא יוצרת אינדקסים בעת הכנסת שאלתה. מי שמתכוון את מערכת-DB צריך להחליט איזה אינדקסים ליצר, ומערכת-DB אחר כך תבחר אם להשתמש או לא.
- (2) כל אינדקס נוסף שנוצר מגביל את כמות הזיכרון שנצרך כדי לשמור את האינדקס, אבל אינדקס נוסף שמייצר גורר גם עליות תחזוק בשnenkins/נוריד/נדכן שורות. כדאי לייצר אינדקסים שיזרו את השאלות הנפוצות והחשובות ביותר.
- (3) ככל שנשים יותר עמודות באינדקס שלנו, גדלה גם כמות המקום שהוא צריך וגם עליות התחזוק שלו.
- (4) לא לייצר אינדקסים מיותרים – אם ייצרנו אינדקס על הזוג rating,age אין טעם לייצר אינדקס רק על age.

IND 6: Multicolumn Indexes 5.6

תזכורת – אינדקס על ערך יחיד

בר נראה אינדקס על ערך יחיד:



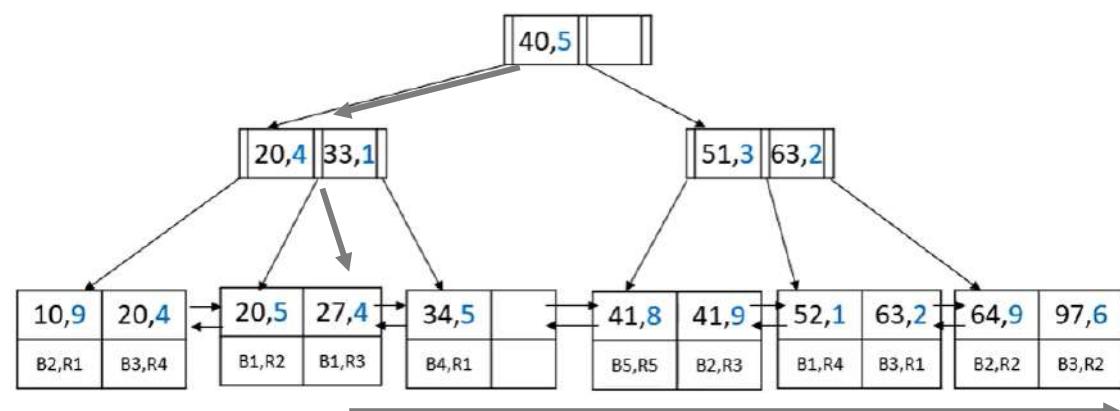
Index on age

```

CREATE INDEX ind_age
ON Sailors(age)

```

במבנה הבא רואים אינדקס על זוג עמודות age ו-gm rating:



Index on (age, rating)

```
CREATE INDEX ind_age_rating
ON Sailors(age, rating)
```

בכל הקודקודים הפנימיים יש זוג ערכים, וכך גם בעלים. באינדקס יש מין ראשוני לפי גיל כי הוא הראשון שהכנסנו, ושניוני לפי דירוג. כך למשל בשיש ערכים קבועים כמו בערך 20, יופיע קודם זה בעל דירוג 4, ואז זה בעל דירוג 5.

אם נחפש לפי age לא תהיה בעיה כמו שעשינו עד עבשו. אבל מה קורה אם נבקש את השאלה:

```
SELECT name
FROM Sailors
WHERE rating = 6
```

נשים לב שהימאים עם דירוג 6 מפוזרים, כי אנחנו ממינים קודם לפיקולו. לא נוכל ביעילות למצאו את מה שרצינו,>Kצת כמו לחפש שם פרטี้ בספר טלפונים שמנוי לפי שמות משפחה.

דוגמאות נוספת

כאן מופיעות שאלות עם תנאי גם על age ו-gm rating על:

```
SELECT name
FROM Sailors
WHERE age > 20
and rating = 7
```

```
SELECT name
FROM Sailors
WHERE age = 20
and rating > 7
```

```
SELECT name
FROM Sailors
WHERE age > 20
and rating > 7
```

שאילתת אפורה: אנחנו רוצחים ימאים שגילם מעל 20 עם דירוג 7. נרד בעץ השמאלי ונגיע לעלה עם גיל 27, ואז נלך ימינה. נשים לב שבדרך נבדוק המונע עליים מיוחדים שבהם הדירוג קטן מ-7. נctrיך לעבר ערך ערך. מסלול באפור ביצור למלחה.

שאילתת **צהובה**: אותה דזוקא אפשר לבצע ביעילות. נגיע לעלה של גיל 20 ונחפש ימאי עם דירוג גובה מ-7, כל השירות שעומת על התנאי יהיה רצופות בעלים, ולא נctrיך להמשיך לחפש.

שאילתת **ירוקה**: ימאים שגילם מעל 20 עם דירוג גובה מ-7, שוב נctrיך כמו בדוגמה הראשונה לחפש ימינה.

מסקנה: האינדקס הוא היעיל ביותר כאשר יש תנאי שווין על השדה הראשון באינדקס, ותנאי השוואת על השדה השני.

דוגמאות

```
CREATE INDEX
ON Orders(itemId, price)
```

```
SELECT avg(price)
FROM Orders
WHERE itemId = 'i242'
```

Full table scan cost:
 $10,000/10=1,000$

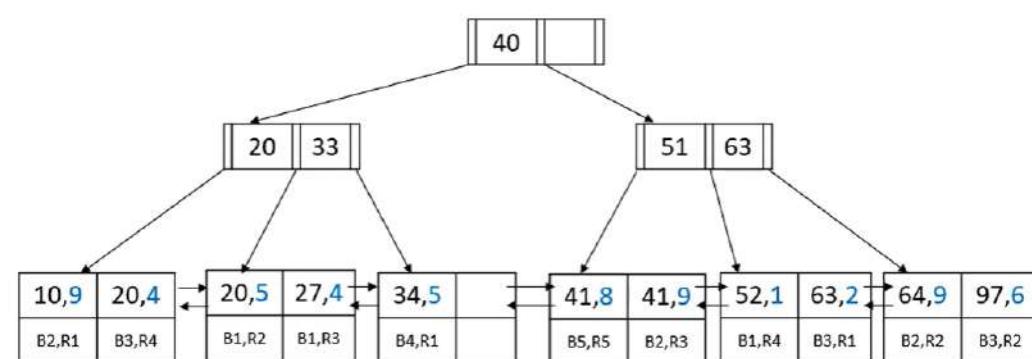
Suppose:

- Orders has 10,000 rows with 10 rows per block
- The index is of branching factor 100
- Item i242 appears in 1,000 orders

Using the index cost:
 Step 1: $\log_{50} 10,000 = 3$
 Step 2: $1,000/49 = 21$ leaves to traverse
 Step 3: Not needed. All information is in index!
 Total: 24

באן אנחנו רוצחים למצוא את המחיר הממוצע של פריט עם itemId,price ספציפי. הפעם בנו אינדקס על הזוג itemId,price. כמה יעלה לחשב אותה? מופיע ביריבוע האדום.

אפשריה נוספת בהמשך היא לבצע אינדקס על שדה מסוים, ורק בעלים להוסיף שדה נוסף:



Index on (age) include rating

```
CREATE INDEX ind_age
ON Sailors(age)
include (rating)
```

למה זה עוזר? אם יש עמודות נוספות באינדקס, זה יכול לחשוך את הצורך בלבלה עצמה כדי לשלווף את הערכים. יש להה יתירונות מעל לבנות אינדקס על זוג העמודות כי מפתח החיפוש הוא רק age ולבן תופס פחות מקום מאשר הזוג age,rating. מכיוון שהוא תופס פחות מקום בדיסק, דרגת הפיצול שנובע למת לעצם בשחайнדקס הוא רק על age תהיה גבוהה יותר מאשר אם נבצע אינדקס על age,rating. זה מאפשר לנו להגיע ל-B Tree עם גובה נמוך יותר, ולכך עם עליונות נוספת.

דוגמאות

```
CREATE UNIQUE INDEX
ON Sailors(sid)
```

```
CREATE INDEX
ON Sailors(name)
```

```
CREATE UNIQUE INDEX
ON Sailors(sid)
include (name)
```

```
CREATE INDEX
ON Sailors(name)
include (sid)
```

```
SELECT name
FROM Sailors
WHERE sid = 22
```

באיזה אינדקס cocci כדי להשתמש עבור השאלה באפור?
 האינדקס **בתקינות** מבון עוזר, יוכל להגיע בנסיבות ל-pid המתאים ולשלוף את השם מהשורה המתאימה.
 האינדקס **בצחוב** לא עוזר לנו בכלל להציגו לתוכה, האינדקס הזה מכיל שמות ומשם לשורות.
 האינדקס **הירוק** עוזר לנו, כי יוכל להגיע ל-pid הנכון ובעהobar נראה את השם – כך גם נחשוך את ההגעה לטבלה. הוא היעיל ביותר מביניהם.
 האינדקס **האדום** לא עוזר לנו בכלל, מאותה סיבה כמו האינדקס הצחוב.

Week 6: Joins and Sorting

JOIN 1: Intro 6.1

[Evaluating Join Queries](#)

פעולת צירוף היא פעולה שבה יש מכפלה קרטזית בין שני יחסים ואז דרישת של תנאי. למשל:

Sailors			
	sid	sname	rating age
B1	22	Alice	7 45
	31	Barbara	8 55
B2	58	Carol	10 35
	70	Danna	10 25

Reserves		
	sid	bid day
B3	22	101 1/1/2018
	58	103 2/1/2018
	31	101 1/2/2018
	22	102 1/2/2018
B4	70	104 2/2/2018
	31	103 2/2/2018
	22	102 3/2/2018
	58	104 4/2/2018

Sailors ⚠ Reserves					
	sid	sname	rating	age	bid day
	22	Alice	7	45	101 1/1/2018
	58	Carol	10	35	103 2/1/2018
	31	Barbara	8	55	101 1/2/2018
	22	Alice	7	45	102 1/2/2018
	70	Danna	10	25	104 2/2/2018
	31	Barbara	8	55	103 2/2/2018
	22	Alice	7	45	102 3/2/2018
	58	Carol	10	35	104 4/2/2018

כאן הצירוף הטבעי הוא על sid, כלומר לכל שורה ב-Sailors ולבב שורה ב-Reserves אנחנו בודקים האם יש להם את אותו sid ואז מייצרים להם שורה ארכבה. נרצה להבין איך מערכת ה-DB מחשבת שאילתות מהסוג הזה בצורה יעילה. הטבלה של sailors שמורה על הדיסק, נניח שהבלוקים מחולקים במתואר בתמונה המצורפת.

הטבלה שאנו מיצרים חלק מהשאילתה גם תופסת מקום, בין אם נכתבו אותה לדיסק ובין אם נשמרת בזיכרון המרכזי ופולטים אותה אחר כך למשתמש, כל חלק מההשובה תופס חלק מהזיכרון. לכן בדוגמה זו יש סה"כ 12 בלוקים (input+output).

[עיבוד מידע](#)

המידע נשמר על הדיסק, אבל כדי לעבד אותו חייבים להעביר אותו לזיכרון המרכזי. הוא מועבר ביחידות של בלוקים. נניח שבזיכרון המרכזי יש 12 בלוקים, אנחנו מקבלים דוגמה מאוד פשוטה. הזיכרון המרכזי יכול לתפוס את כל הבלוקים (גם של הקלט וגם של הפלט):

Main Memory

כאן החישוב יהיה בצורה יעילה. אחרי שביצמנו את החישובים הנדרשים יוכל לעשות כל פעולה שהמשתמש מבקש.

1	Sailors (B1)
2	Sailors (B2)
3	Reserves (B3)
4	Reserves (B4)
5	Output
6	Output
7	Output
8	Output
9	Output
10	Output
11	Output
12	Output

מה קורה אם היזכרון המרכז' לא יכול להכיל הכל ביחד? למשל מקום רק ל-5 בЛОקים?

Main Memory	
1	Sailors (B1)
2	Sailors (B2)
3	Reserves (B3)
4	Reserves (B4)
5	Output

אנחנו מיד נעשה flush לפט (לדיסק או למסך) בכל פעם שמחושב בЛОק תוצאה. בעצם אפשר לדרכס את בЛОק 5 עם הבלוק הבא של התוצאה.

בוא נצמצם אפילו יותר. מה קורה אם יש 4 בLOCKים? אין מספיק מקום גם לקרוא את כל הטבלאות וגם כדי לכתוב פט. בוא ננסה בר:

Main Memory	
1	Sailors (B1)
2	Sailors (B2)
3	Reserves (B3)
4	Output

איך נוכל לתקדם במצב זה? אפשר לצרף את השורות של בלוק מס' 1 ו-2 של Sailors, ולכתוב את התוצאה ל-Output כל פעם שהבלוק מתמלא. אחרי שצירפנו את B3 עם כל הטבלה Sailors נוכל לדרכס אותו, להביא את B4 ולצרף אותו עם כל השורות ב-Sailors. זה נקרא אלגוריתם חיצוני – הוא מבצע חישוב תחת ההנחהuai אפשר להכניס את ה-*Input* ליזכרון המרכזי במלואו.

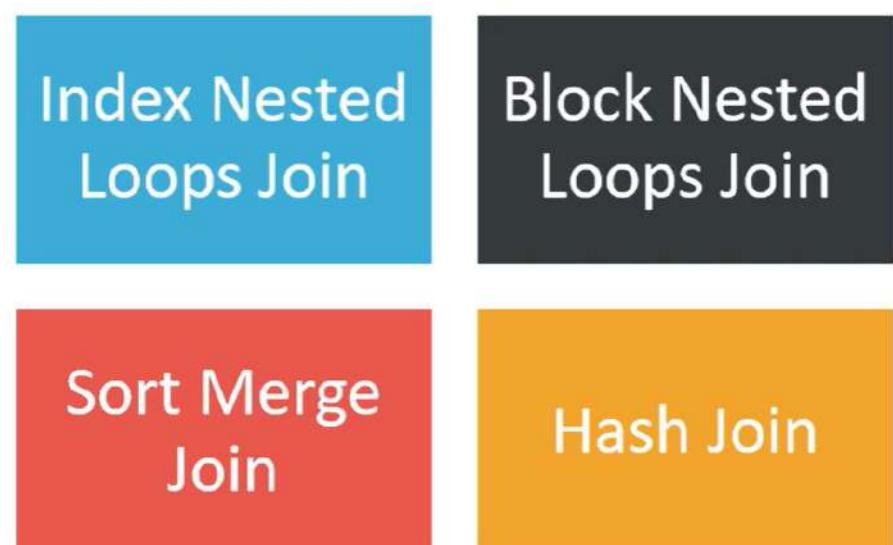
מה אם יש רק 3 בLOCKים?

Main Memory	
1	Sailors (B1)
2	Reserves (B3)
3	Output

לא כל ברור איך נחשב ביעילות את כל היצירוף במצב זה.

אלגוריתמים לחישוב פועלות Join

אנחנו נפגש 4 אלגוריתמים:



Query optimizer chooses algorithm estimated to be most efficient for a given query.

בכל פעם שמערכת ה-DB מקבלת בקשה ל-JOIN היא מנסה לחשב כמה יעלה בקריאות וכתיבות O/I עם כל אחד מהאלגוריתמים, ומפעיל את הזולה ביותר.

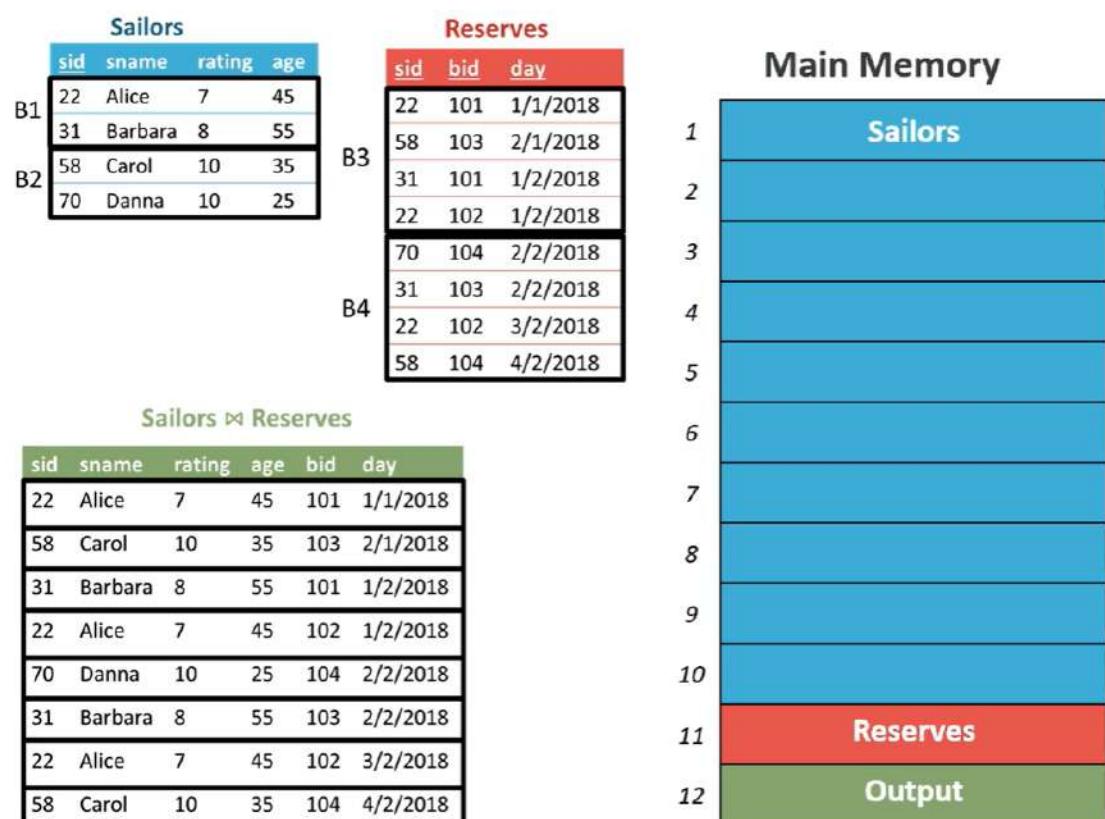
נוסיציות:

M
Number of blocks of internal memory available

B(S)
Number of blocks in S

B(R)
Number of blocks in R

נחוור לדוגמה:



מה בעצם האלגוריתם? האלגוריתם פועל בעדרת לולאות מקוננות. מחליטים מי יהיה היחס הפנימי וממי יהיה היחס החיצוני. לאחר שקיבלו את ההחלטה, נקרא את היחס החיצוני ביחידות של 2-M בЛОקים:

Internal Memory



M-2

לאחר מכן, נקרא את היחס הפנימי בЛОק, בכל פעם נעשה צירוף בין הBLOCK הפנימי (האדום) וBLOCK היחס החיצוני (הכחולים), ואת התוצאה נכתב בFILE (בירוק), ונעשה flush. אחר כך נעלה את הBLOCK האדום השני, השלישי וכן הלאה. אחרי שנס'ים לעבור על כל היחס הפנימי, נעלה את ה-2-M בЛОקים הכתולים הבאים, ושוב נעשה את אותה הלולאה עם הBLOCKים האדומים אחד בכל פעם.

פואdon:

Inner R(D, E) *Outer* S(E, F)

R stored in blocks $b_0, \dots, b_{B(R)-1}$
 S stored in blocks $b'_0, \dots, b'_{B(S)-1}$

```

left = 0
while left < B(S):
    right = min(left + M-2, B(S))
    read  $b_{left}, \dots, b_{right-1}$  to memory
    for i = 0 to B(R)-1:
        read  $b'_i$  to memory
        for each tuple (e,f) of S in memory
            for each tuple (d,e') of R in memory
                if (e=e') add (d,e,f) to output
    left = left + M-2
    
```

דוגמא רצאה בסרטון.

עלות

במה עלה לבצע את האלגוריתם?
 את היחס החיצוני קראנו רק פעם אחת, אך העלויות שלו היויה כמות הבלוקים שהוא מופס, כלומר $B(S)$.
 את היחס הפנימי קראנו $\left\lceil \frac{B(S)}{M-2} \right\rceil$ פעמים, וככפוף בעלות של קריאת הבלוקים: $B(R) \cdot \left\lceil \frac{B(S)}{M-2} \right\rceil$ פעולות.
 לכן סה"כ:

$$B(S) + B(R) \left\lceil \frac{B(S)}{M-2} \right\rceil$$

נשים לב! כדי לבחור את היחס הקטן יותר בתורה היחס החיצוני, זה יחסור פעולות.

דוגמה 1

```
SELECT *
FROM Sailors S, Reserves R
WHERE S.sid = R.sid
```

Assumptions:

- 1,000,000 rows in Reserves, 100 rows per block
- 10,000 rows in Sailors, 10 rows per block
- 102 buffer blocks

נבחר מי יהיה היחס החיצוני, לשם כך נבדוק מי קטן יותר: $B(R)$ או $B(S)$.

Size of Sailors: 10,000 / 10 = 1,000 blocks
 Size of Reserves:
 $1,000,000 / 100 = 10,000$
Sailors is smaller

זמן ריצה:

Run time:
 Read Sailors once: 1,000
 Read Reserves 1,000 / (102 - 2) = 10 times, at cost 10 * 10,000 = 100,000

סה"כ: 101,000 פעולות O/I. לשים לב, אנחנו לא מחשבים את עלות כתיבת התוצאה של הבחירה אנחנו לא מחשבים חלק מחישוב פעולות ה-O/I.

דוגמה 2

```
SELECT *
FROM Sailors S, Reserves R
WHERE S.sid = R.sid and
      date = '1/1/01'
```

$\sigma_{date='1/1/01'}(Sailors \bowtie Reserves)$

$Sailors \bowtie \sigma_{date='1/1/01'}Reserves$

Option 1:

Compute query as before, then **filter out** non-matching rows

Option 2:

Push selection, computing it *before* the join

Total: 101,000

לצורך הפשטה נניח שאין כאן מבנה אינדקס, זהה נתעסק בהמשך.
אופציה ראשונה היא לחשב את השאלתה כמו שכתוב בביטוי האלגברי השמאלי. נחשב את הצירוף ברגיל, ובכל פעם שמייצר תוצאה של הצירוף נדרש שורות שלא עוננות על התנאי. באן ב모ת הפעולות תהיה בדיקת כמו בדוגמה הקודמת (אין עלות על הפילטר).
אפשרות אחרת, היא לנסות לדוחוף את התנאי **לפני** חישוב הצירוף. איך?

```
SELECT *
FROM Sailors S, Reserves R
WHERE S.sid = R.sid and
      date = '1/1/01'
```

Assumptions:

- 1,000,000 rows in Reserves, spanning 50 dates
- 100 rows per block
- 10,000 rows in Sailors, 10 rows per block
- 102 buffer blocks

נניח התפלגות אחידה על 50 תאריכים, נחשב שוב את הגודל של הבלוקים:

Size of Sailors: 10,000 / 10 = 1,000 blocks
 Size of Reserves:
 $1,000,000 / 100 = 10,000$
 Reserves after selection:
 $1,000,000 / 50 / 100 = 200$

הגודל של Reserves עבשוי חולק ב-50, כי מתווך 50 תאריכים אפשריים רק אחד מעניין אותנו. אחרי הבחירה, מתקיים $B(R) < B(S)$. אנחנו משתמשים ב-Reserves ביחס החיצוני. זמן ריצה חדש:

Run time:
 Read Reserves once (10,000) only keeping rows matching selection.
 Read Sailors $200 / (102 - 2)$ = 2 times, at cost 2
 $* 1,000 = 2000$

את היחס החיצוני קוראים רק פעם אחת, 10,000 פעולות, אבל נשמר רק את ה-200 שורות שמתאימות לתנאי הבחירה. נשים לב שעבשוי נדרש לקרוא את Sailors רק פעמיים (פעם עם 100 בלוקים ראשונים של Reserves, ופעם עם 100 בלוקים אחרים). עלות הקראיה של Sailors תהיה 2,000. סה"כ עלות כל החישוב: 12,000 פעולות/א. שיפור משמעותי לעומת החישוב הקודם.

[Index Nested Loop \(INL\) Join](#)

אלגוריתם שעבוד בולאה מקוננת, ומוכיח קיום של אינדקס. נוטציות:

$T(R)$
Number of tuples in R

$B(R)$
Number of blocks in R

לפתרון זה נדרש רק ל-4 בלוקים בזיכרון המרכזי:

Sailors				Reserves			
	sid	sname	rating	age	sid	bid	day
B1	22	Alice	7	45	22	101	1/1/2018
B2	31	Barbara	8	55	58	103	2/1/2018
	58	Carol	10	35	31	101	1/2/2018
	70	Danna	10	25	22	102	1/2/2018
B3					70	104	2/2/2018
B4					31	103	2/2/2018
					22	102	3/2/2018
					58	104	4/2/2018

Sailors \bowtie Reserves						
	sid	sname	rating	age	bid	day
	22	Alice	7	45	101	1/1/2018
	58	Carol	10	35	103	2/1/2018
	31	Barbara	8	55	101	1/2/2018
	22	Alice	7	45	102	1/2/2018
	70	Danna	10	25	104	2/2/2018
	31	Barbara	8	55	103	2/2/2018
	22	Alice	7	45	102	3/2/2018
	58	Carol	10	35	104	4/2/2018

Main Memory	
1	Sailors
2	Index
3	Reserves
4	Output
5	
6	
7	
8	
9	
10	
11	
12	

על מנת לחשב את היצירוף הטבעי בין שתי הטבלאות, נבחר מי יהיה היחס החיצוני ומי הפנימי. נבעור על היחס החיצוני (מושך בלוק אחד בכל פעם), ונבעור שורה שורה ביחס הפנימי (שוב, בלוק אחד בכל פעם). לכל שורה ביחס הפנימי משתמש במבנה אינדקס על שדה הה-join של היחס הפנימי, כדי למצוא שורות מתאימות. פסאודו קוד:

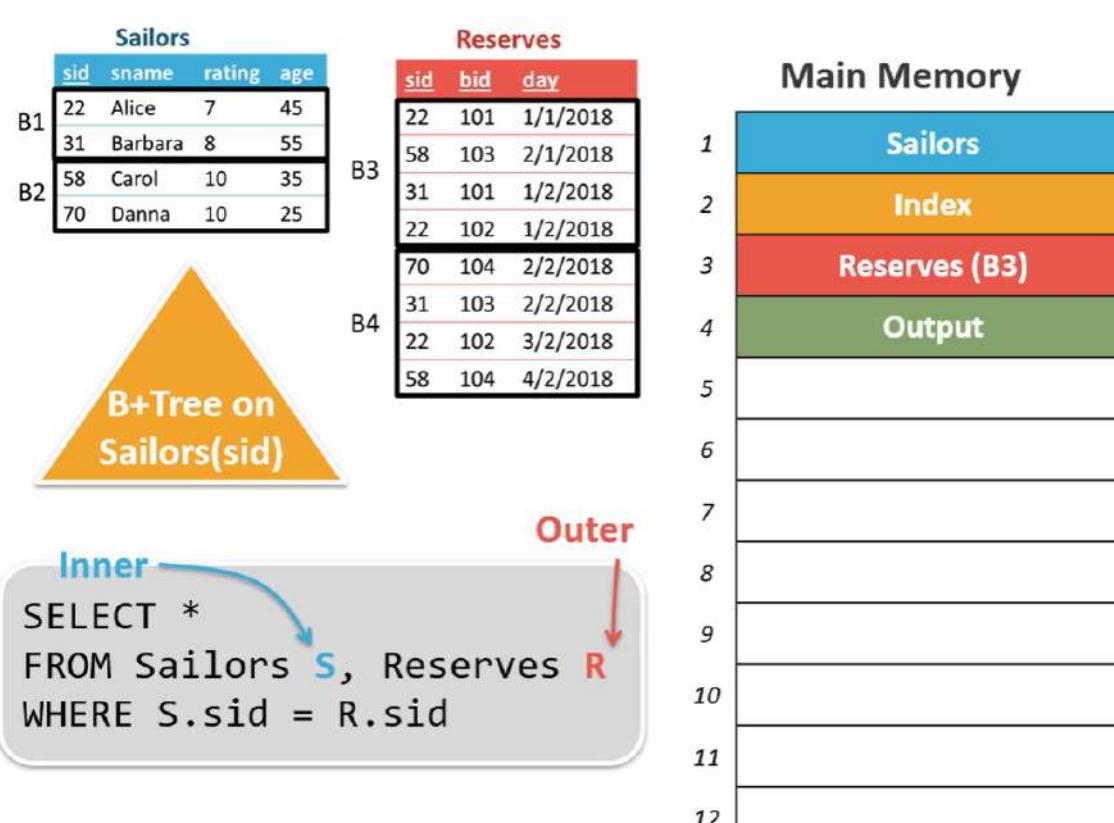
outer
 $R(D, E)$ *inner*
 $S(E, F)$
Assuming index on S.E

```
for each block b of R:  
    read b to memory  
    for each tuple (d,e) in b:  
        Search the index on S.E for e  
        for each index entry (e,tupleId):  
            access S at tupleId to retrieve (e,f)  
            add (d,e,f) to the output
```

הערה חשובה: על מנת להשתמש באלגוריתם INL חיברים שייהי למ אינדקס על שדה היחס הפנימי!

באן, יש לנו אינדקס על E.S. לכן אפשר לבחור את S להיות היחס הפנימי.

דוגמא



מכיוון ש-sid הוא מפתח בטבלה Sailors אנחנו יודעים שקיים אינדקס ל-key (מערכת DB יוצרת אינדקס על שדה sid primary key באופן אוטומטי). נקרא את הבלוק הראשון של Reserves ומעבר אליו שורה שורה.

עלות חישוב

הדגשה: אפשר להשתמש בשיטה רק כשייש אינדקס על שדה ה-join של היחס הפנימי.

$B(R)$ = Number of blocks of R (outer relation)

$T(R)$ = Number of tuples of R

- Outer relation R is read once: $B(R)$
- For each tuple in R, apply selection condition on join attribute: $T(R) * \text{cost_of_select}$

נשים לב ש-cost_of_select זה העלות לחפש ב-B-Tree ושליפת השורה המתאימה. מחיר זה תלוי בחישובים שראיםו ביחידה הקודמת. תוצאות:



סה"כ: $B(R) + T(R) \cdot \text{cost_of_select}$

דוגמה 1

```
SELECT *
FROM Sailors S, Reserves R
WHERE S.sid = R.sid
```

Read Reserves. Cost:
 $1,000,000 / 100 =$
10,000

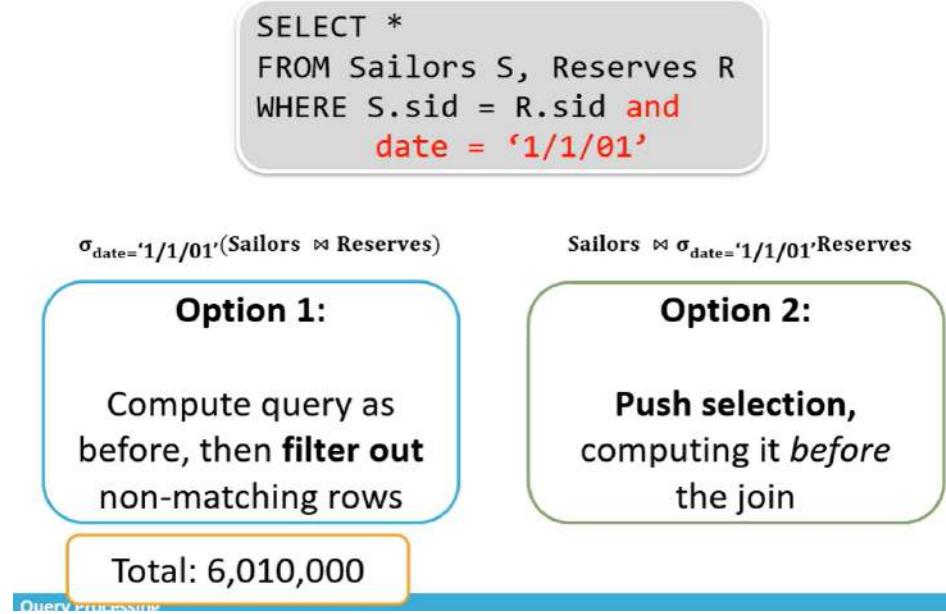
Assumptions:

- 1,000,000 rows in Reserves, 100 rows of Reserves per block
- 10,000 rows in Sailors
- Btree of branching factor 20 over Sailors

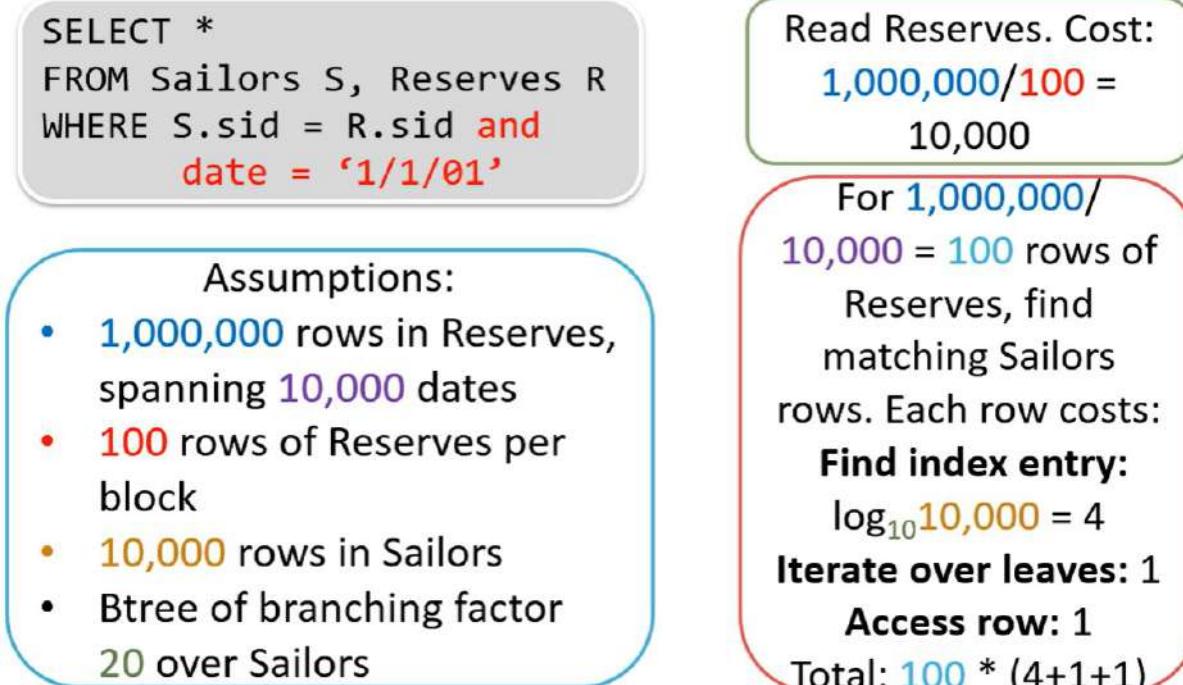
For 1,000,000 rows of Reserves, find matching Sailors rows. Each row costs:
Find index entry:
 $\log_{10} 10,000 = 4$
Iterate over relevant leaves:
1
Access row: 1
Total: $1,000,000 * (4+1+1)$

마חר sid הוא מפתח, נדרש לעבור על עלה אחד בכל פעם (לא יתכו חזרות), וכן נדרש לגשת גם לשורה אחת בהתאם.

Total: 6,010,000



כאן אנחנו מוסיפים תנאי. כמו קודם, יכוליםו או לסקן את התוצאות רק לאחר החישוב ואז העלות לא תשתנה, או שננסן לפני ה-join. נחשב:



הוספנו הנחה של התפלגות אחידה של 10,000 תאריכים. עדין נדרש לקרוא את כל Reserves וזה יעלה לנו 10,000. הפעם, רק לשורות של Reserves שעונות על התנאי נדרש לבצע את `.cost_of_select`.

Total: 10,600

גם כאן, ביצוע הבחירה לפני פעולת הצירוף מייעלת את השאלה ממשמעותית.

זה כלל – תמיד אם אפשר לדוחף את פעולה הבחירה לפני הצירוף מומלץ לעשות זאת כדי לחסוך בעליות.

נשים לב שלא תמיד אפשר, למשל:

```

    SELECT *
    FROM Sailors S, Reserves R
    WHERE S.sid = R.sid and
          rating = 10
  
```

אם אפשר לבצע את פעולה הבחירה לפני פעולה הצירוף? התשובה היא לא. אנחנו צריכים להשתמש באינדקס על היחס הפנימי כדי למצוא את השורות המתאימות. אין לנו אינדקס על השורות של *Sailors* בהן *rating=10*, מען האמת אין מידע על *rating* בכלל. לכן נהיה חייבים לצרף ורק אז לסקן. אילו היה לנו אינדקס על *sid,rating* יכולנו לעשות זאת.

מיון נתונים במסד נתונים

לפעמים בשאינו מבקש למין לפי תכונה, לא באמת תבוצע פעולה מיון. זה יכול לקרות למשל אם יש לנו אינדקס:

```
SELECT name
FROM Sailors
ORDER BY name
```

```
CREATE INDEX ON
Sailors(name)
```

QUERY PLAN

Index Only Scan using sailors_name_idx on sailors
(cost=0.28..67.26 rows=1000 width=5)

הצורה שבה נחזיר את התוצאה היא בעזרת Index Only Scan על השם. כל המידע שאנו צריכים לשאליתה נמצא שם, ואם נקרא את האינדקס לפי הסדר מההתחלה עד הסוף, נקבל בכל מקרה את הסדר המקורי. לרוב זה לא המקרה, ונctrkr Sort כדי להשיג מיון. בדוגמה הבאה זה Index Scan כי אחר כך ניגשים לטבלה לשולף את age:

```
SELECT name, age
FROM Sailors
ORDER BY name
```

```
CREATE INDEX ON
Sailors(name)
```

QUERY PLAN

Index Scan using sailors_name_idx on sailors
(cost=0.28..67.26 rows=1000 width=9)

בשאלתה הזאת לעומת זאת, אנחנו נctrkr למין לפי שם וגיל בשាឦאינדקס רק לפי שם, ולכן יהיה כאן מיון:

```
SELECT name
FROM Sailors
ORDER BY name, age
```

```
CREATE INDEX ON
Sailors(name)
```

QUERY PLAN

Sort (cost=67.83..70.33 rows=1000 width=9)
Sort Key: name, age
-> **Seq Scan** on sailors
(cost=0.00..18.00 rows=1000 width=9)

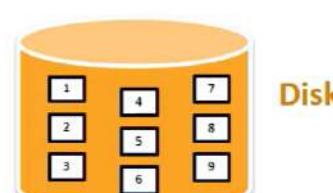
מיון יכול להיות שימושי למערכת-h-DB לא רק כאשר המשמש מבקש בסדר לפי תכונה. המערכת תשתחם בהז גם כאשר יש DISTINCT בשאליתה וצריך למצוא כפליות בתוצאה, אחת הדריכים שלו למצוא כפליות היא בעזרת מיון הנתונים. כאשר עושים GROUP BY אחת מהצורך לבצע זאת היא על פי מיון בסדרות ה-GROUP BY. אם יש לנו פעולה איחוד, חיתוך, מינוס – אלו פעולות בהן מוחקים כפליות וגם דורשים מאיתנו להשווות תוצאות של שאלותות שונות. לפעמים נשיג את המבוקש בעזרת hashing ולא מיון.

AIR MIION MMOMASH?

Typically
Main Memory << Data File

Sorting with a standard **internal sort** method
will be very inefficient since it will make
MANY disk reads and writes

Main
Memory

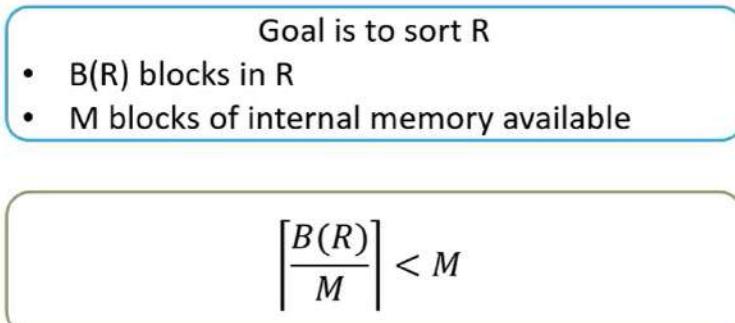


אנו יודעים איך למיין נתונים (למשל quicksort). הבעיה היא שהמיעד שאנו צריכים למניין הרבה יותר גדול מהזיכרון המרכזי. לא ניתן להחזיק מידע על כל הנתונים ולמיין אותם. אם ננסה לעשות זאת, או שנקל שגיאה על חוסר מקום או אם יש הרבה זיכרון וירטואלי נצליח להקצות אותם, אבל הניסיונות להחליפ' נתונים ידרשו קריאות ובתיות רבות – מה שיגרום לתוכנית להיות איטית.

הפתרון – אלגוריתם חיצוני

אלגוריתם חיצוני הוא אלגוריתם שמטרתו לבעד נתונים שגדולים מדי עבור הזיכרון המרכזי של המחשב. ננתח את הסיבות שלו על פי מספר הקריאה והבתיות לדיסק (לא כולל כמה זמן לוקח לבחוב את התוצאה הממונית לדיסק).

נלמד מקרה פרטי של מיון חיצוני שבו מתקיים התנאי המוקף בירוק:



We will see **2-phase external merge sort**

- 1) Create sorted sequences
- 2) Merge sorted sequences

2-Phase External Merge Sort

נשתמש במניין מיוג' חיצוני בשני שלבים. בשלב ראשון ניצור סדרות ממונות קצרות יחסית ביחס R . בשלב שני, נמזג את הסדרות הקצרות כדי לקבל סדרה אחת ממונית.

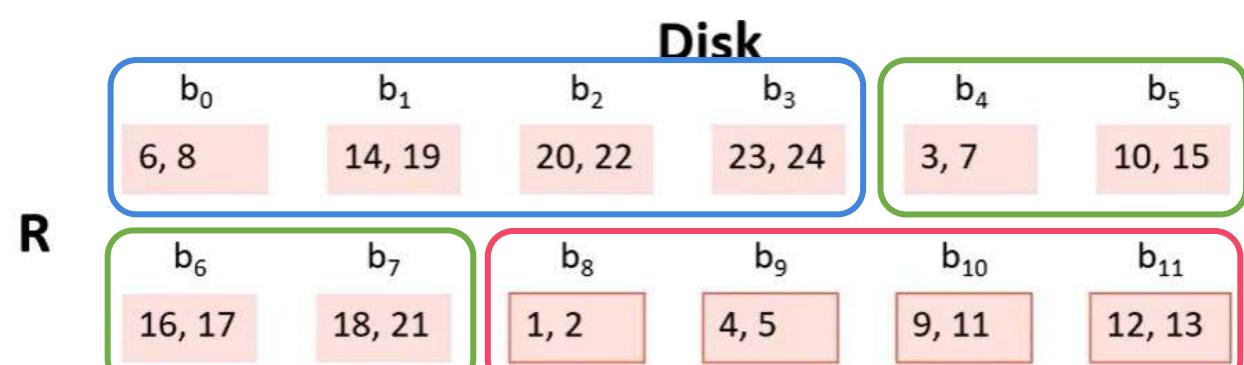
שלב 1



```
left = 0
while left < B(R):
    right = min(left + M, B(R))-1
    read blocks  $b_{left}, \dots, b_{right}$  to main memory
    sort  $b_{left}, \dots, b_{right}$  in place
    write sorted sequence to disk
    left = left + M
```

נכיה ש- R מאוחסן בבלוקים $b_{B(R)-1}, \dots, b_0$. נזכיר את הצד השמאלי של האינטראול שנרצה למיין, וכל עוד הוא קטן מ- (R) נבחר את הצד ימין להיות צד שמאל ועוד M. כמובן, נמיין את M הבלוקים הראשונים. נעביר את M הבלוקים לזכרון וממיין אותם שם in. למשל, עם quicksort. נכתוב את הסדרה הממונית חזרה לדיסק, ונגדיל את left בעוד M. **דוגמת הרצה בסרטון בדקה 10:00.**

אחרי המניון של שלב זה נקבל את המערכת ממויין בחלקים כר:

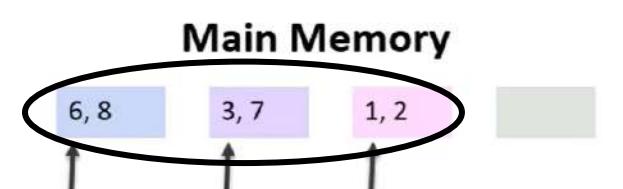
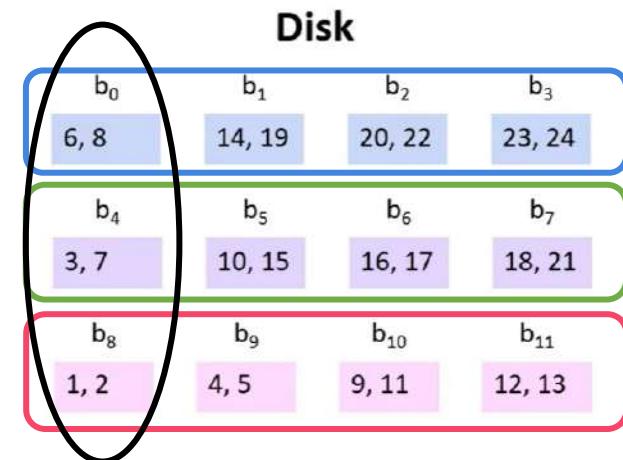


עבשו יש לנו $\left\lceil \frac{B(R)}{M} \right\rceil$ סדרות ממונות. כמה עלה שלב זה? קראנו את R פעמי' אחד: עלות $(R) \cdot B(R)$. סה"כ $(R) \cdot 2B(R)$.

עבור הפסאודו, נניח שיש n שורות בכל בלוק של R , ו- N שורותסה"ב.

```
for i = 1 to  $\left\lceil \frac{B(R)}{M} \right\rceil$ :
    read the first block of the i-th sorted
    sequence into memory
    set  $p_i$  to point to the first tuple in
    this block
```

לכל אחת מ- $\left\lceil \frac{B(R)}{M} \right\rceil$ הסדרות הממויניות, נקראת הבלוק הראשון של כל אחת מהסדרות ל זיכרון, ונקבע לשורה הראשונה בבלוק עם המצביע i . **דוגמא**
הרצה בסרטון בדקה 13:00. זה בעצם מעורבב את הסדרות:

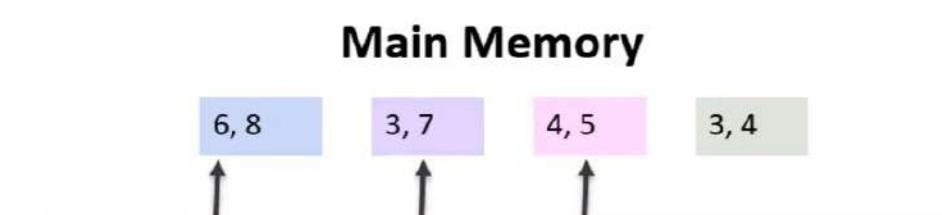


נשים לב שתמיד נשאר לנו בלוק אחד בזיכרון המרכזי אליו נכתוב את הפלט. למה מובטח לנו שיש זהה? בזכות ההנחה ממקודם:

$$\left\lceil \frac{B(R)}{M} \right\rceil < M$$

```
k = 0
while k < N:
    let  $p_i$  be the pointer to the minimal
        tuple among all pointers
    copy the tuple of  $p_i$  to the output block
    increment k
    if  $k \bmod n = 0$ 
        flush the output block to screen/disk
    if  $p_i$  points to the last tuple of the
        block, read the next block of the
        sequence and set  $p_i$  to its first tuple
    else increment  $p_i$ 
```

אחרי ששמננו את כל הבלוק הראשון מכל סדרה בזיכרון המרכזי, נמיין את הערכים. בהתאם קטן מכਮות השורות שא"ב אනחנו מוצאים את המצביע לשורה המינימלית מבין כלום. אනחנו בעתיק את הערך שהוא מצביע אליו לבלוק הפלט, ונגדיל את k . אם $0 \bmod n = 0$ כלומר אם k מילאנו את בלוק הפלט, נכתוב את הפלט לאן שרצינו (מסך/דיסק). אם i מצביע לשורה האחורונה בבלוק שלו, נקראת הבלוק הבא בתור של אותה סדרה, ו- i יעבור להצביע לשורה הראשונה שם. אחרת, נגדיל את i להצביע לשורה הבאה. **דוגמא הרצה ב-15:15.**



כמה עלה השלב הזה? $B(R)$ (בהתבה שלא ספכנו את העלות של הכתיבה לדיסק).

סה"כ כל האלגוריתם:

Total: $3B(R)$

דוגמא 1

האם אפשר לבצע כאן מיון בשני שלבים?

```
SELECT *\nFROM R\nORDER BY A
```

- R has **6,000,000** rows
- 10 rows fit in one block
- 6 available buffer blocks

Is 2-phase external merge-sort applicable? If so, what is the I/O cost?

No!

$$B(R) = \frac{6,000,000}{10} = 600,000$$

$600,000 / 6 > 6$

לא. התנאי שדרשנו שיתקיים, $\left\lceil \frac{B(R)}{M} \right\rceil$ לא מתקיים.

דוגמא 2

```
SELECT *\nFROM R\nORDER BY A
```

- R has **6,000,000** rows
- 1000 rows fit in one block
- 100 available buffer blocks

Is 2-phase external merge-sort applicable? If so, what is the I/O cost?

Yes!

$$B(R) = \frac{6,000,000}{1000} = 6,000$$

$6,000 / 100 < 100$

I/O Cost: $3 * 6,000 = 18,000$

כאן כן אפשר.

מה אם התנאי לא מתקיים?

אי אפשר למזג בחלק אחד, נctrarף בכל פעם למזג 1-M שוב ושוב עד שסימנו הכל.

Week 7: Query Plans and Optimization

OPT: Intro 7.1

השבוע נלמד איך מערכת-h-dp בוחרת תוכנית חישוב שלמה.

[?Select](#)

נתונה השאלה הבאה:

```
SELECT *  
FROM Sailors S  
WHERE age > 90
```

ונניח שיש לנו אינדקס על age. האם היינו עוכבים על השאלה אם האינדקס או עם הטבלה כולה? אנחנו דורשים ימאים מעל גיל 90 ואנחנו לא מצפים לקבל הרבה באלו, שכן בנסיבות גודף לעבוד עם האינדקס. אם נרצה למשל $0 < \text{age} < 90$ עבור דרך האינדקס יהיה מיותר, כי בנסיבות יקימו את התנאי זה לא יחסור לנו קריאות.

למערכת-h-dp אין את המידע הזה, אז איך היא תחליטה? כדי שמערכת-h-dp תוכל לבחור שיטה נקבעת בה, היא צריכה דרך להעיר במה אchod מהטבלה יתאים לתנאי ובמה לא – לשם כך היא תאסוף סטטיסטיות על הנתונים.

[?Join](#)

Which join order would you prefer?

1. $(\sigma_{\text{age} > 90} \text{Sailors} \bowtie \text{Reserves}) \bowtie \text{Boats}$
2. $\sigma_{\text{age} > 90} \text{Sailors} \bowtie (\text{Reserves} \bowtie \text{Boats})$

```
SELECT *  
FROM Sailors S, Reserves R, Boats B  
WHERE S.sid = R.sid and R.bid = B.bid  
and age > 90
```

אם עדיף לנו קודם לסנן גיל ואז לעשות צירוף טבעי עם Reserves ולאחר מכן עם Boats? שיטה 1 עדיפה, כי יהיו פחות שורות אחרי הסימון של הגיל ואז פחות עבודה בצירופים הבאים.

אבל מה נעשה בדוגמה הבאה:

Which join order would you prefer?

1. $(\sigma_{\text{age} < 30} \text{Sailors} \bowtie \text{Reserves}) \bowtie \sigma_{\text{color} = \text{'pink'}} \text{Boats}$
2. $\sigma_{\text{age} < 30} \text{Sailors} \bowtie (\text{Reserves} \bowtie \sigma_{\text{color} = \text{'pink'}} \text{Boats})$

```
SELECT *  
FROM Sailors S, Reserves R, Boats B  
WHERE S.sid = R.sid and R.bid = B.bid  
and age < 30 and color = 'pink'
```

אם עדיף קודם לסנן את צבע הספינה או קודם את גיל הימאי? זה קשור בכך שמספר ספינות בצבע ורוד לעומת מספר ימאים מתחת לגיל 30. לשם כך נדרשות סטטיסטיות איתן לבצע את ההשוואה.

סטטיסטיות

כדי לבחור באיזה סדר למסג אנחנו חייבים לקבל שיטה להעריך את גודל תוצאות הבינים. לשם כך יש סטטיסטיות:

Statistics are needed to choose efficient query plans

The database computes and stores statistical information

In Postgres, statistics are created and updated upon index creation, or when running ANALYZE or VACUUM

מערכת ה-db שומרת מידע סטטיסטי על הדטה שלנו, והוא מעדכנת את הנתונים בזמן יצירת אינדקס או כאשר מרייצים את הפקודה `ANALYZE` או `VACUUM` (למשל ב-`postgres`). נשלף נתונים מהמערכת:

```
public=> \x
Expanded display is on.
public=> select * from pg_stats where tablename = 'movies';
```

ונקבל:

schemaname	sara
tablename	movies
attname	movieid
inherited	f
null_frac	0
avg_width	4
n_distinct	-1
most_common_vals	
most_common_freqs	
histogram_bounds	{9,1711,2525,3100,3619,3888,4205,4449,4710,4945,5151,5370,5573,5761,5945,6129,6329,6495,6680,6840,7030,7169,7317,7485,7630,7794,7944,8070,8217,8347,8471,8607,8741,8880,9014,9142,9278,9400,9526,9652,9782,9907,10055,10180,10313,10438,10572,10711,10837,10963,11095,11219,11345,11457,11591,11717,11836,11952,12082,12202,12324,12444,12562,12675,12799,12912,13036,13157,13283,13401,13523,13646,13760,13878,14004,14126,14239,14356,14479,14597,14718,14841,14955,15065,15177,15291,15412,15527,15658,15777,15896,16036,16158,16275,16390,16510,16642,16753,16882,17004,17119}
correlation	0.9999983
most_common_elems	
most_common_elem_freqs	
elem_count_histogram	
-[RECORD 2]-----+	

למשל אם `inherited` האם חלק מהעמודות באוט בטבלה אחרת, מה שיאפשר לנו להשיג סטטיסיות ממש (במקרה זה `false`).

`null_frac` אחוז השורות בטבלה עם ערכי `null`

`avg_width` רוחב ממוצע של הנתונים בעמודה זו (בביטים)

`n_distinct` כמות הערכים השונים שיש בעמודה. אם המספר חיובי זה כמות הערכים שיש בעמודה, אם המספר שלילי זה אחוז הערכים השונים מתווך כל השורות של הטבלה. כל השורות שנותן אז קיבלו נ-1.

`most_common_vals`, `most_common_freqs` עמודות שמכילות ערכים נפוצים ותדירות שלהם.

`histogram_bounds` חלוקה של הערכים בקובוצה כך שבכל תח חילק יש בערך אותה כמות של שורות

`correlation` קורלציה בין הערך בעמודה זו והמקום הפיזי שלו בדיסק (האם ערכים קרובים מבחינה ערך האם קרובים גם פיזי בדיסק) חשוב למערכת כדי להבין האם כשהוא יגש לטבלה באינדקס הוא ישלו שורות שקרובות אחת לשנייה, מה שיאפשר פחות קריאות דיסק.

OPT 2: Size Estimation 7.2

שיעור גודל תוצאה של שאלתה

אנחנו נדבר על כמה שיטות פשוטות להעריך את גודל תוצאה השאלתה בהינתן סטטיסיות על הטבלאות.

נוטציות

T(R) is the number of tuples of R

B(R) is the number of blocks R occupies on disk

V(R,A) is the number of distinct values in attribute A of R

הערכה: עבור A אם A הוא מפתח אז $V(R, A) = T(R)$

הערכת תוצאה שאלתה

```
SELECT *
FROM R
WHERE A = 'a'
```

$\sigma_{A='a'}(R)$

Number of rows in result,
assuming uniform distribution:

$$\frac{T(R)}{V(R, A)}$$

בהרצתה זו אנחנו נתעניין בכמות השורות בתוצאה של מר (R), אבל לפעמים נדבר גם על כמות הבלוקים.

[מה קורה אם יש תנאי השוואת?](#)

```
SELECT *
FROM R
WHERE A <= x
```

 $\sigma_{A \leq x}(R)$

Range of A is known to be $[y, z]$

Number of **rows** in result
(percentage of values in range satisfying condition)

$$T(R) \frac{x - y + 1}{z - y + 1}$$

אם אנחנו יודעים שהטווח של A הוא בין $[z, y]$ אנחנו מוצאים שהערכים שיעם על התנאי יהיו $1 + y - x$ שזה במתה הערכים שמקיימים את התנאי, חלקי $1 + y - z$ שזה הטווח כולו.

[מה עושים באשר הטווח של A לא ידוע?](#)

```
SELECT *
FROM R
WHERE A < x
```

 $\sigma_{A < x}(R)$

Range of A is unknown

Number of **rows** in result:

$$\frac{T(R)}{3}$$

מספר השורות בתוצאה יהיה מספר השורות ביחס חלקו 3. למה? כובע.

[מה קורה אם יש שני תנאים?](#)

```
SELECT *
FROM R
WHERE A = 'a' and B = 'b'
```

 $\sigma_{A='a' \text{ and } B='b'}(R)$

Assume independence of conditions.
Number of **rows** in the result:

$$\frac{T(R)}{V(R, A) \times V(R, B)}$$

אנחנו נניח אי תלות בין התנאים, ולכן נכפול במכנה (כמו גימום בהסתברות).
אם יש תנאים שבהם אחד מהאטריבואיטים הוא $=$ ואחד עם $<$ אנחנו נמשיך להשתמש באותו שיטות, גם למשל כאשר אחד הטווחים אינו ידוע:

```
SELECT *
FROM R
WHERE A = 'a' and B < 'b'
```

 $\sigma_{A='a' \text{ and } B < b}(R)$

Assume independence of conditions, e.g., if,
range of B is unknown:

$$\frac{T(R)}{V(R, A) \times 3}$$

אם יש שתי השוואות ואנחנו לא יודעים את הטווח של אף אחד מהם, נעריך את הגודל כ- $\frac{T(R)}{3}$

מה קורה אם יש or ?

```
SELECT *
FROM R
WHERE A = 'a' or B = 'b'
```

$\sigma_{A='a' \text{ and } B='b'}(R)$

Assume independence of conditions.

Number of rows in the result:

$$A \neq 'a' \quad \wedge \quad B \neq 'b'$$

$$T(R) \left(1 - \left(1 - \frac{1}{V(R,A)} \right) \times \left(1 - \frac{1}{V(R,B)} \right) \right)$$

עדין נכח אי תלות בין התכונות, ונעשה 1 פחתה ה-"וגם" של המשלימים.

פעולת צירוף

```
SELECT *
FROM R, S
WHERE R.A = S.A
```

$R \bowtie S$

Assume a uniform distribution of values of one table into those of the other.

Number of rows in the result:

$$\frac{T(R) \times T(S)}{\max\{V(R,A), V(S,A)\}}$$

בפעולת צירוף, איך נעריך את הגודל? נשים לב שגודל צירוף הוא למעשה הפעלה של תנאי בחירה הרבה פעמים. אם נסתכל על התוצאה מנוקדת מבט של היחס R, כל שורה ב-R אナンנו צופים שהיא תתאים לכמה שורות ב-S. אם יש ערך ספציפי של A ב-R שאנחנו רצimos ש-A.S. יהיה שווה לה, אナンנו מצפים לקבל $\frac{T(S)}{V(S,A)}$ שורות מתאימות ב-R. זה נכון לכל שורה ב-R, ולכן מבחןינו אナンנו נראה $\frac{T(S)}{V(S,A)} \times T(R)$ שורות בלבד. אותו דבר מנוקדת המבט של S נצפה לקבל $T(S) \times \frac{T(R)}{V(R,A)}$. אナンנו נרצה לתקן את הביטוי הכללי הקטן יותר, ולכן ניקח מקסימום במבנה.

מה קורה אם A הוא מפתח ב-R ומפתח זר ב-S?

```
SELECT *
FROM R, S
WHERE R.A = S.A
```

$R \bowtie S$

What if A is a key in R and a foreign key in S?

הערכים ב-A.R לא חזרים על עצם, וכל ערך שמוופיע ב-A.R מופיע כבר ב-A.S. אナンנו צופים שככל שורה ב-A.R תצטרף לשורה אחת בבדיקה של R, כי יש בדיקת אחת שתתאים. לכן נצפה לראות בבדיקה (S) T(S) שורות מתאימות:

בנוסחה מוקדם, במבנה שהגדול יותר הוא $(R, A) (R, A) T(R) \times T(S) \frac{T(R) \times T(S)}{V(R,A)}$. אבל מאחר ש-A הוא מפתח, נקבל $\frac{T(R) \times T(S)}{T(R)} = T(S)$.

תנאי על Join וגם תנאי בחירה

```
SELECT *
FROM R, S
WHERE R.A = S.A and R.B = 'b'
```

$\sigma_{B='b'}(R \bowtie S) = (\sigma_{B='b'}R) \bowtie S$

$$\frac{T(R) \times T(S)}{\max\{V(R,A), V(S,A)\} \times V(R,B)}$$

נחשב בצורה של אי-תלות בין הדברים. נצמצם את R לפי התנאי שלו על B על ידי חילוקה ב- $(R, B) \setminus A$, וגם מצמצמים את הצורך על ידי חילוקה בביטוי המקסימום.

ziehof על שתי עמודות

```
SELECT *
FROM R, S
WHERE R.A = S.A and R.B = S.B
```

$R \bowtie S$

Assume a uniform distribution of values of one table into those of the other.

Number of **rows** in the result:

$$\frac{T(R) \times T(S)}{\max\{V(R,A), V(S,A)\} \times \max\{V(R,B), V(S,B)\}}$$

נדרש שווין בין R ל-S גם על A ו-S וגם על B. וכרגע מניחים אי-תלות, אז מחלקים גם בתנאי שמקבלים בגלל הצורך על A, וגם בתנאי שמקבלים בגלל הצורך על B.

הערכת גודל של הטלה

זה דזוקא פשוט וחסית. אם השאלה שלנו היא הטלה ללא מחיקת בפיזיות זה יהיה ($R \setminus T$). אם נרצה הטלה עם מחיקת בפיזיות (Distinct) אז נעריך את גודל התוצאה כ- $V(R, A)$.

OPT 3: Query Plans 7.3

בנייה תוכנית ביצוע שאליתה

To create a query plan for an SQL query:

1. Translate the query to algebra
2. Choose evaluation methods for each operator
3. Push projections and selections where possible

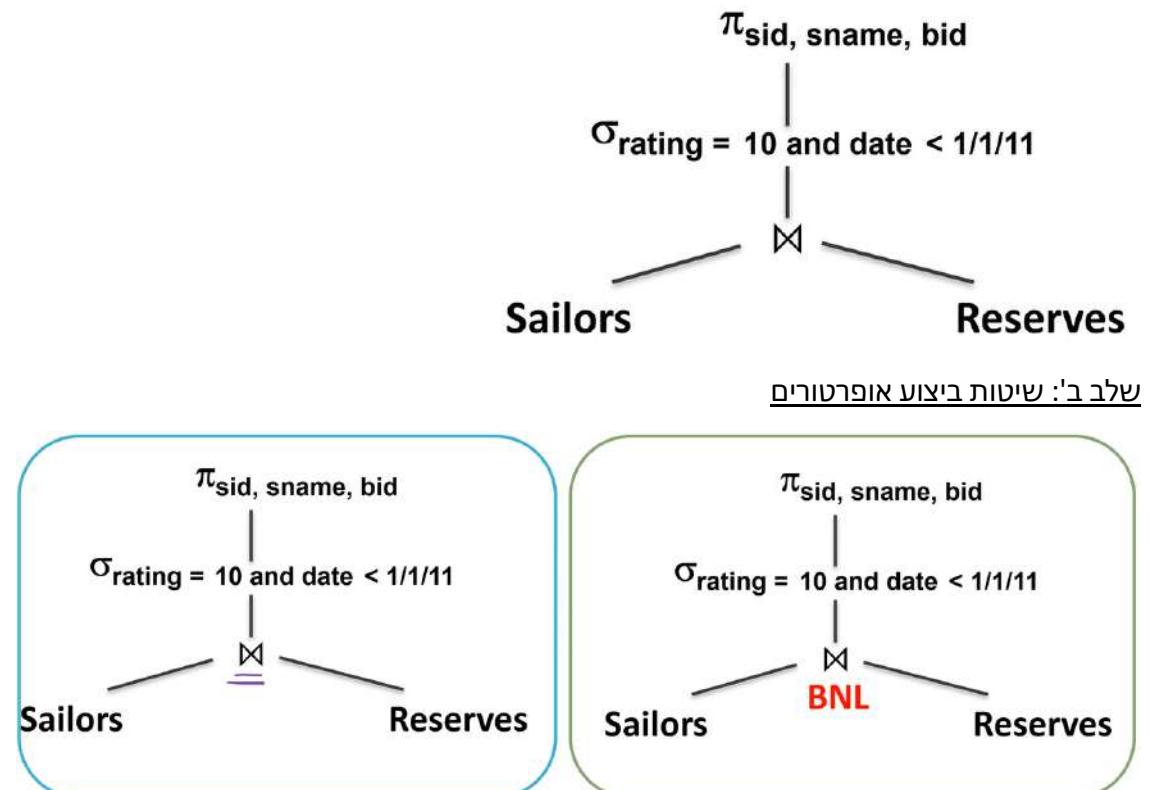
דוגמה 1

```
SELECT S.sid, S.sname, R.bid
FROM Sailors S, Reserves R
WHERE S.sid = R.sid and
      rating = 10 and
      date < 1/1/11
```

שלב א': תרגום לאלגברה רלציונית

$\pi_{\text{sid, sname, bid}}(\sigma_{\text{rating} = 10 \text{ and } \text{date} < 1/1/11}(\text{Sailors} \bowtie \text{Reserves}))$

אבל נציג עבשוי שיטה חדשה לרשום את הביטוי בצורה עז:

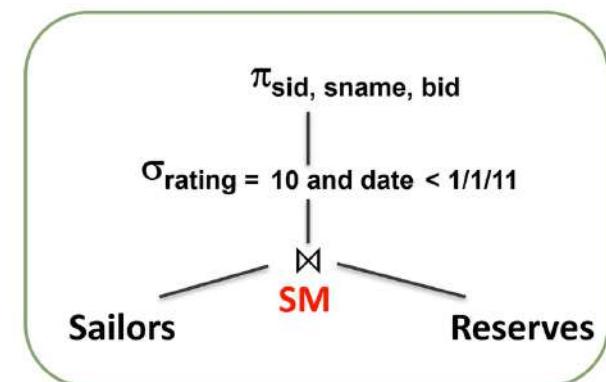


עבור BNL קיימת מוסכמה שהחיצוני הוא השמאלי. אותה מוסכמה קיימת גם עבור IN.

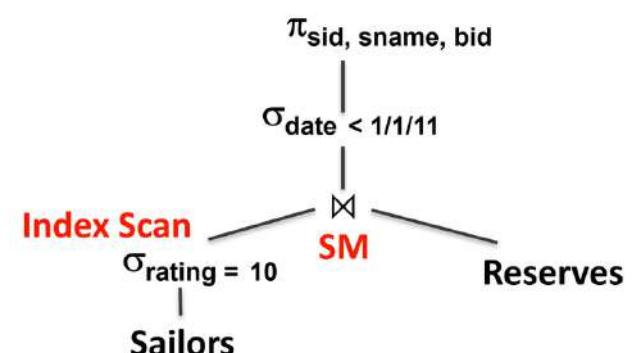
לא ניתן אין לבצע את פעולה הבחירה ואת פעולה ההטלה. אם לא צינו מפורשות, אנחנו מניחים שתוצאות הפעולה הקדמת מועברות בצורה ישירה לפעולה הבאה בשיטה שנקרה pipeline בלי כתיבת תוצאות ביןיהם לדיסק. בשוציאה שורה שמתאימה לצירוף, כל שורה שיצא מ-BNL נכנסת לשירות לפעולות הבחירה, ואם מקיימת את התנאי היא תעבור להלאה לפעולות ההטלה ותיצור שורת תוצאה.

שלב ג': דחיפת פעולה הבחירה

כאן יש דוגמה נוספת:

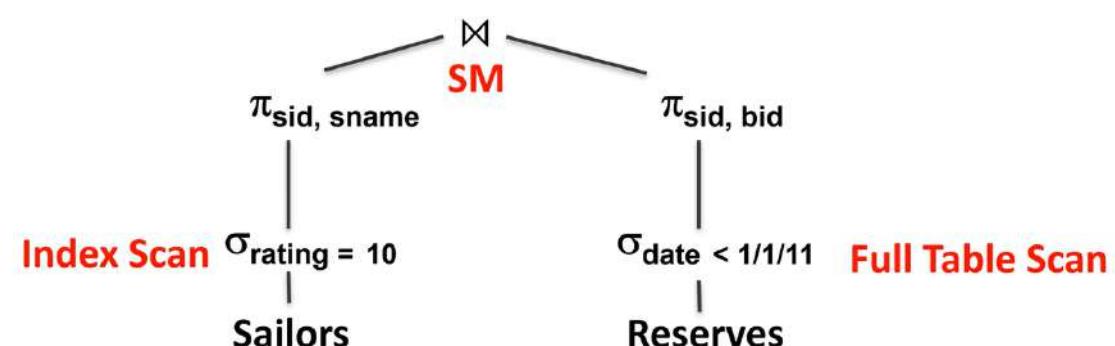


כאן משתמשים ב-*Join Sort Merge*. כאן אין משמעותו ליחס חיצוני פנימי אך לא משנה איפה יחס שמננו באיזה צד. דחף את פעולה הבחירה מוקדם יותר:



מכיוון שדחיפנו אותה למטה אנחנו מבצעים אותה תוך קריית *Table Scan* של Sailors אוניברסלי לא מושגנו באותו אופן. אם יש Index והוא יותר מושלם נרשום *Index Scan*, אחריה נרשום *Full Table Scan*. גם כאן עובדים ב-*pipeline Full*. לא נחשב את כל תוצאה הבחירה של Sailors נרשום לדיסק ואחר כן צירוף עם Reserves אלא תוך כדי צירוף החישוב. בעצם השורות שלא מקיימות את התנאי לא נמנים בכלל.

באופן כללי, נרצה לדחוף כמה שיותר בחירות והטלות למטה בעז:



במה תוכניות שונות אפשר לראות עבור אותה שאלתה?

(דבר כאן על שאלות עם פועלות צירוף אחת)

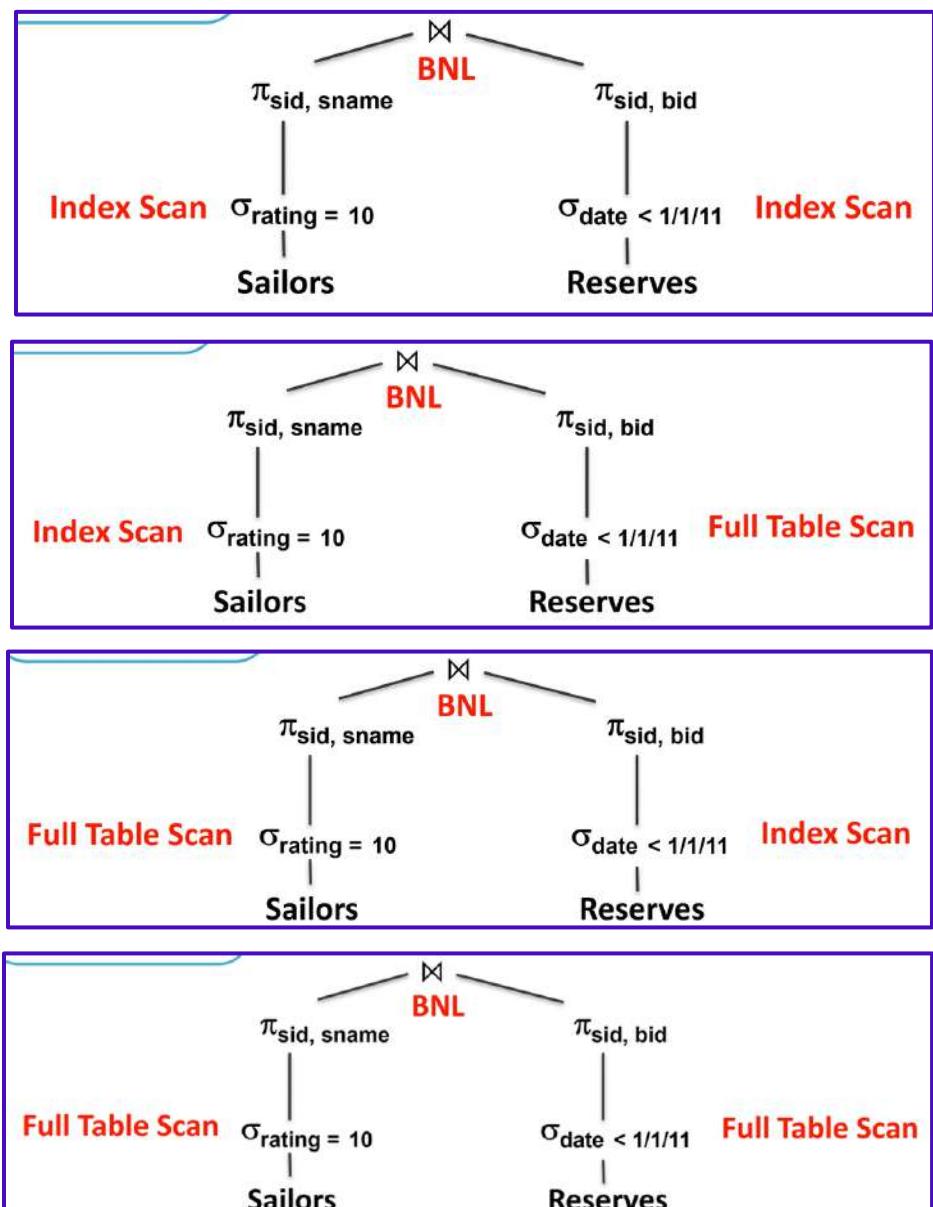
דוגמא

```
SELECT S.sid, S.sname, R.bid
FROM Sailors S, Reserves R
WHERE S.sid = R.sid and
      rating = 10 and
      date < 1/1/11
```

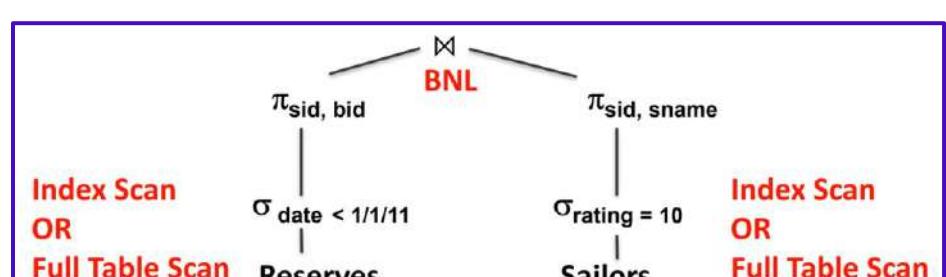
- 10,000 tuples of Sailors
- 100,000 tuples of Reserves
- sid, sname, age, rating, bid each require 10 bytes. Date requires 5
- Only 1% of ratings are 10
- A block has 1,000 bytes
- 50 buffer pages
- There are indexes on rating, date, and R.sid of branching factor 100

נתמך בנתונים על מבנה האינדקס (בצהוב). יש אינדקס על rating ועל date וזה מעניין כי יש עליהם בחירה, ויש אינדקס על R.sid וזה גם מעניין כי הוא שדה הצירוף.

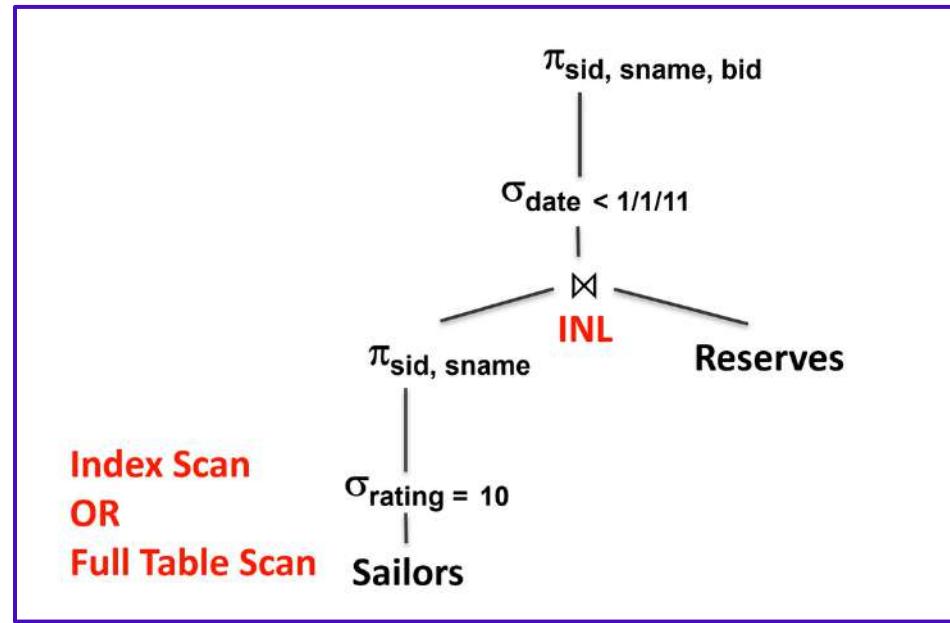
יש לנו 4 אפשרויות ל-BNL כאשר Sailors הוא היחס החיצוני.



יש 4 תוכניות ביצוע נוספת ב-BNL, כאשר Reserves הוא היחס החיצוני:



קיבלו שיש אינדקס על השדה pid . R. ולכן נוכל לבצע גם JIN באופן הבא, ויש 2 דרכים שモוצגות כאן:



(בנהנזה שיש אינדקס על $\text{R.sid, rating, date}$. הערא – מניחים כאן שאין אינדקס על pid . זה קצת לא הגיוני כי זה מפתח, אבל זה רק לשם הדוגמה) במקורה הזה Reserves חייב להיות היחס הפנימי ב- JIN , כי אנחנו חייבים לבחור את היחס הפנימי להיות זה שיש לו אינדקס על יחס הצירוף. למה כאן פועלות הבחירה מופיע מעל הצירוף? (date מופיע מעל JIN). הסיבה לכך היא שכשאנו מבצעים JIN זה לא אפשרי להכין את פועלות הבחירה לפני שדה הцירוף אלא אם כן האינדקס שיש על Reserves מאפשר ביצוע פועלות הבחירה כשהאנו מבצעים JIN אנחנו צריכים לגשת ליחס הפנימי דרך האינדקס על שדה הцירוף (pid) כשהאנו קוראים שורה דרך דרך אינדקס על pid היא מצטרפת לשורה מ- Sailors ורק לאחר מכן אנחנו יכולים לסנן את התוצאות לפי ערך התאריך. לא היינו יכולים לבחור שורות עם התאריך המתאים דרך מבנה האינדקס כי האינדקס לא מספק לנו את השדה הזה. אם היה לנו אינדקס על date, pid יכולם לדוחוף את הבחירה לפני.

[AIR נבחרים את התוכנית הטובה ביותר?](#)

We have 10 Different Options!

(The recording states that there are 18 options, which would be correct if we also considered sort-merge join and hash join, which were omitted from the material this year.)

Observe that many options are due to the fact that we can access each relation either with a full table scan or with an index

The best choice depends on what is known about the data

בסה"כ ראיינו 18 תוכניות ביצוע שונות. 8 עבור JIN , 2 עבור $\text{Sort-Merge hash join}$ שיד מוחומר). הבחירה הטובה ביותר תלויה במאגר הנתונים.

OPT 4: Choosing Selection 7.4

[AIR נבחר את התוכנית הטובה ביותר?](#)

1. ננסה להבין מה הדרך הזולה ביותר לחשב כל אחת מפעולות הבחירה? (או אינדקס אם זמין)
2. למצוא את שיטת ה-join הזולה ביותר שמתאימה לקרה
3. תוך כדי הבדיקה, מניחים דחיפפה מקסימלית של פעולות בחירה והטלה. (לא נסתכל על תוכניות בהן היה אפשר לדוחף ולא דחפנו).

נתמוך ברגע בחלק 1. בבר לפני שבועיים בחרנו עליונות אינדקס לעומת full table scan , אז עבשו רק נעשה חזרה.

```
SELECT S.sid, S.sname, R.bid
FROM Sailors S, Reserves R
WHERE S.sid = R.sid and
rating = 10 and
date < 1/1/11
```

- 10,000 tuples of Sailors
- sid, sname, age, rating, each require 10 bytes.
- Only 1% of ratings are 10
- A block has 1,000 bytes
- 50 buffer pages
- Index on rating of branching factor 100

נבדוק כמה יעלה Full Table Scan לעומת אינדקס על rating.
כל שורה תופסת 40 בתים, יש 10,000 שורות 1-1000 בתים בכל בLOC.

Using Index on Rating:

- Index traversal costs $\log_{50} 10,000 = 3$
- $\%1 * 10,000 = 100$ matching tuples
- Leaf traversal costs $100/49 = 3$
- Accessing rows costs 100

I/O Cost: 106

Full Table Scan:

- Row requires 10 * 4 = 40 bytes.
- $1,000 / 40 = 25$ rows per block
- $10,000 / 25 = 400$ blocks

I/O Cost: 400

אנו רואים שככל תוכנית שכוללת Full Table Scan מומלץ לשולול, כי היא פשוט יקרה יותר. מתוך 18 תוכניות ביצוע נשארנו עם 9.

מה הדרך היולה ביותר לגשת לטבלה Reserves?

25 בתים לשורה, בכל בלוק 1,000 בתים. אין לנו מידע על האופן בו תאריכים מתפלגים – נשתמש בכל האצבוע של $\frac{1}{3}$ מהשורות מקיימות את התנאי.

Using Index on Date:

- Index traversal costs $\log_{50} 100,000 = 3$
- $100,000 / 3 = 33,334$ matching tuples
- Leaf traversal costs $33,334/49 = 681$
- Accessing rows costs 2,500

I/O Cost: $3 + 2,500 + 681 = 3,184$

Full Table Scan:

- Row requires 10 * 2 + 5 = 25 bytes.
- $1,000 / 25 = 40$ rows per block
- $100,000 / 40 = 2,500$ blocks

I/O Cost: 2,500

לכן לא ננצל את מבנה האינדקס.

We can rule out all plans using an index on date to access Reserves

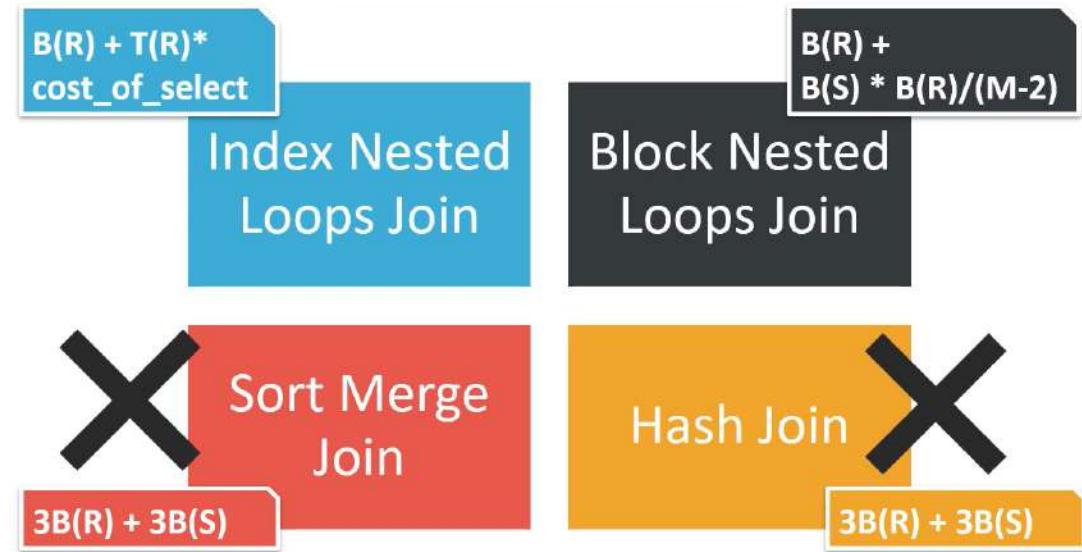
We are left with 5 plans:

1. Block Nested Loops, Sailors as outer
2. Block Nested Loops, Reserves as outer
3. Sort Merge Join
4. Hash Join
5. Index Nested Loops join, Sailors as outer

*Need to compare all these options!
(This year: only options 1,2,5)*

[הערבת עלות הציגוף](#)

בשבוע שעבר ראיינו 4 אלגוריתמים שונים של צירוף וחישבנו את עלות הביצוע. יש לנו נסחאות לבן.

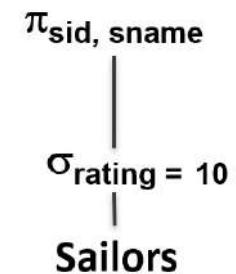


אנחנו חישבנו את העלות מעלה יחסים, אבל אנחנו רוצים לחשב מעל תוכנות חישובי בחירה באלגברה – זה קצת משנה את התמונה.

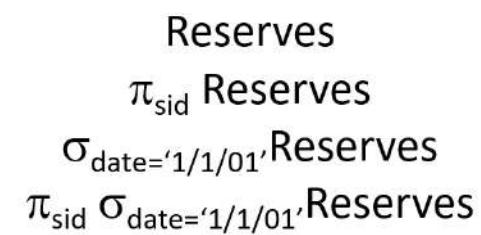
[האם הנוסחאות יעבדו אותו דבר?](#)

הנחנו שאנו מצרפים לרוב יחסים R ו- S , אבל בפועל אנחנו מחשבים תוכנות ביוטיים, E_R ו- E_S . בחישובים שנעשה נראה שזה משנה את הזמן.

בשחישבנו את הנוסחאות בעבר השתמשנו ב- $B(R)$ זה היה הגדל של R וגם מחיר קריית R . אם אנחנו מעוניינים לצרף את R עם S (בל' תנאים על R) עולה לנו $(B(R) + B(S))$ והוא $B(R+S)$. בשיש לנו איזשהו ביוטי העולות של קריית הביטוי וגודל הביטוי יכולם דוווקא להיות שונים. למשל:



כאן $B(R+S)$ הוא הגדל של $Sailors$ אבל לא גודל הביטוי כולם. אנחנו מדברים על ביוטיים שיכולים להכיל פעולות של בחירה או הטלה (או שניהם).



אנחנו נגדיר:

Define:		
Read(E)	I/O Cost of Reading E from Disk	
B(E)	Number of Blocks of E	
T(E)	Number of Tuples of E	

אם יש לנו למשל $\text{Read}(E) = E$ ונניח שאין לנו אינדקס על $Sailors$. כמה עולה לנו (E) ?

$\text{Read}(E) = B(\text{Sailors})$ על $Sailors$ full table scan וכן $\text{Read}(E) = B(\text{Sailors})$.

$$T(E) = \frac{T(\text{Sailors})}{1000} \quad B(E) = \frac{B(\text{Sailors})}{1000}$$

מה זה (E) ? אם נניח שрак $\frac{1}{1000}$ מהימאים מקיימים את התנאי אז

אנחנו רואים ש- $B(E) = \text{Read}(E)$ יכולם להיות שונים.

BNL: Reminder

Sailors			
sid	sname	rating	age
22	Alice	7	45
31	Barbara	8	55
58	Carol	10	35
70	Danna	10	25

B1 B2 B3 B4 Sailors \bowtie Reserves

sid	sname	rating	age	bid	day
22	Alice	7	45	101	1/1/2018
58	Carol	10	35	103	2/1/2018
31	Barbara	8	55	101	1/2/2018
22	Alice	7	45	102	1/2/2018
70	Danna	10	25	104	2/2/2018
31	Barbara	8	55	103	2/2/2018
22	Alice	7	45	102	3/2/2018
58	Carol	10	35	104	4/2/2018

Reserves		
sid	bid	day
22	101	1/1/2018
58	103	2/1/2018
31	101	1/2/2018
22	102	1/2/2018
70	104	2/2/2018
31	103	2/2/2018
22	102	3/2/2018
58	104	4/2/2018

Main Memory			
1	Sailors	2	3
4			
5			
6			
7			
8			
9			
10			
11	Reserves		
12	Output		

תזכורת: בוחרים יחס חיצוני בלולאה, משתמשים ב-2-M בלוקים מהזיכרון המركزي לקרוא את היחס החיצוני, ובעבור הבלוקים האלה נקרא בלוק בלוק של היחס הפנימי, נחשב את ה-הוֹלְדָן ווצוא בהתוצאה. כנסים עם 2-M הבלוקים הראשונים נעבור לבאים ונורץשוב בלוק של היחס החיצוני, וכך עד שנסים.

איך המידע החדש משנה את הנוסחה?

$$E_R \bowtie E_S$$

Read(E)
B(E)
T(E)

I/O Cost of Reading E from Disk
Number of Blocks of E
Number of Tuples of E

$$B(R) + B(S) * \lceil B(R)/(M-2) \rceil$$

$$\text{Read}(E_R) + \text{Read}(E_S) * \lceil B(E_R)/(M-2) \rceil$$

אם R הוא היחס החיצוני, קראנו את R פעם אחת בעלות $B(R)$ ועזרנו את S $\lceil \frac{B(R)}{M-2} \rceil$ פעמים. איך זה יתרגם הפעם? אם מדובר בבייטוי E_R הוא הביטוי החיצוני ונקרה אותו פעם אחת. לאחר מכן אנחנו קוראים את S הרבה פעמים, ולכן $\text{Read}(E_S)$ בכל פעם, בעוד מספר השורות שמקיימות את הביטוי, וכן הכמות היא $\lceil \frac{B(E_R)}{M-2} \rceil$ כי לכל $2 - M$ בלוקים של E_R נדרש ללבת שוב לקרוא את E_S .

INL: Reminder

Sailors			
sid	sname	rating	age
22	Alice	7	45
31	Barbara	8	55
58	Carol	10	35
70	Danna	10	25

Reserves		
sid	bid	day
22	101	1/1/2018
58	103	2/1/2018
31	101	1/2/2018
22	102	1/2/2018
70	104	2/2/2018
31	103	2/2/2018
22	102	3/2/2018
58	104	4/2/2018

B3 B4 Sailors \bowtie Reserves

sid	sname	rating	age	bid	day
22	Alice	7	45	101	1/1/2018
58	Carol	10	35	103	2/1/2018
31	Barbara	8	55	101	1/2/2018
22	Alice	7	45	102	1/2/2018
70	Danna	10	25	104	2/2/2018
31	Barbara	8	55	103	2/2/2018
22	Alice	7	45	102	3/2/2018
58	Carol	10	35	104	4/2/2018

Main Memory

1	Sailors
2	Index
3	Reserves
4	Output
5	
6	
7	
8	
9	
10	
11	
12	

תזכורת: בוחרים יחס חיצוני ופנימי, מעל שדה הצורך של היחס הפנימי חייב להיות מבנה אינדקס. מה עושים? קראנו את כל היחס החיצוני וכל שורה ביחס החיצוני אנחנו קראנו דרך האינדקס את השורה המתאימה ביחס הפנימי. קראנו את כל Sailors Reserves על הטבלה Reserves לכל שורה בס-*s*. מצאנו את השורות המתאימות ב-Reserves.

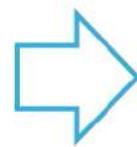
בדוגמה הבאה R הוא היחס החיצוני:

$$E_R \bowtie E_s$$

Read(E)
B(E)
T(E)

I/O Cost of Reading E from Disk
Number of Blocks of E
Number of Tuples of E

**$B(R) + T(R)*$
cost_of_select**



Read(E_R) + $T(E_R)*$ cost_of_select

קראנו פעם אחת את R ואז לכל שורה של R חיפשנו את השורות המתאימות ביחס הפנימי דרך האינדקס. לכל שורה של R חישבנו את עלות הבחירה דרך מבנה האינדקס. מה קורה אם אין יחס R אלא ביטוי ? E_R ?
את כל הביטוי נצטרך לקרוא מהדיסק ולכן אנחנו נשלם (E_R) T ואל כל שורה שמתאימה לתנאי (E_R) 我们将计算所有满足条件的元组数。

סיכום

**Read(E_R) +
 $T(E_R)*$ cost_of_select**

Index Nested Loops Join

**Read(E_R) +
Read(E_S) * $\lceil B(E_R)/(M-2) \rceil$**

Block Nested Loops Join

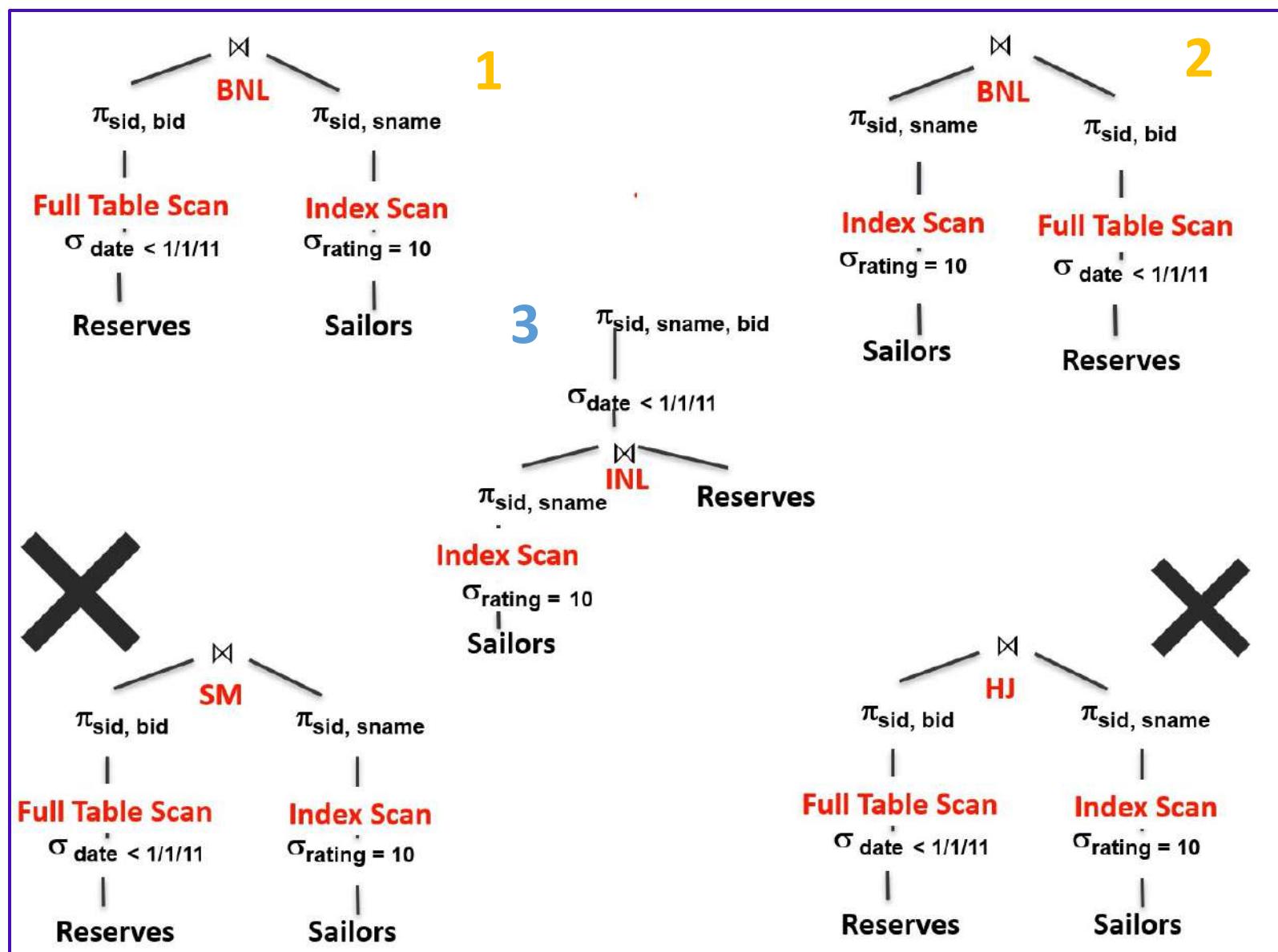
**Read(E_R) + Read(E_S) +
 $2(B(E_R)+B(E_S))$**

Sort Merge Join

Hash Join

**Read(E_R) + Read(E_S) +
 $2(B(E_R)+B(E_S))$**

הנה 5 תוכניות הביצוע ואנחנו צריכים למצוא את הזול מביניהם (אנו חושב ש-2 מתוכם ירדו מהחומר):



נשים לב שבכל תוכנית הביצוע אנחנו הتمקדמו בשימוש ב-full scan בעקבות הבחירה על Sailors ו-index scan על Reserves, כי חישבנו שזו הדרך היעילה יותר.

יש 5 תוכניות: 2 אופציות של **BNL** (לפי מי היחס החיצוני או sailors או reserves), אופציה אחת של **INL** כאשר reserves הוא היחס הפנימי כי היה לנו אינדקס רק על שדה הבחירה של Reserves (ולא היה על של Sailors). אפשרות של Sort-Merge (Sailors) ושל Hash-Join (Reserves).

עלויות וגדלים

$$\begin{aligned} E_R &= \pi_{\text{sid}, \text{bid}} \sigma_{\text{date} < '1/1/01'} \text{Reserves} \\ E_S &= \pi_{\text{sid}, \text{sname}} \sigma_{\text{rating} = 10} \text{Sailors} \end{aligned}$$

$$\begin{aligned} \text{Read}(E_S) &= 106 \\ B(E_S) &= \\ T(E_S) &= \end{aligned}$$

$$\begin{aligned} \text{Read}(E_R) &= 2,500 \\ B(E_R) &= \\ T(E_R) &= \end{aligned}$$

So far, we have calculated the above values...

כפי שראינו בהרצאה הקודמת, כדי שנוכל לחשב את עלויות אלגוריתמי ה-join עבור כל אחד מהביטויים שיש לנו (ביטוי בחירה של reserves עם הטלה על bid ו-pid, וביטוי בחירה על sailors עם הטלה על sname,sid) אנחנו צריכים לחשב את עלות הקראיה של הביטוי ואת גדי התוצאות של הביטוי. בינתים בשחישבנו את עלות הבחירה בברור חישבנו כמה עלה הכל אחת מפעולות ה-full (את דרך E_S Read-E และ את דרך full table scan E_R). נשאר לנו להבין את שאר הנתונים שחסרים במלבנים הירוקים.

```
SELECT S.sid, S.sname, R.bid
FROM Sailors S, Reserves R
WHERE S.sid = R.sid and
rating = 10 and
date < '1/1/11'
```

- 10,000 tuples of Sailors
- sid, sname, age, rating, each require 10 bytes.
- Only 1% of ratings are 10
- A block has 1,000 bytes

- 1% * 10,000 = 100 matching rows
- After projection each row requires 10 + 10 bytes
- $100 / (1,000 / 20) = 2$ blocks

$.T(E_S) = 100$ ו $B(E_S) = 2$ בוכן

```
SELECT S.sid, S.sname, R.bid
FROM Sailors S, Reserves R
WHERE S.sid = R.sid and
rating = 10 and
date < '1/1/11'
```

- 100,000 tuples of Reserves
- sid, bid each require 10 bytes. Date requires 5 bytes
- A block has 1,000 bytes

- $100,000 / 3 = 33,334$ matching rows
- After projection each row requires 10 + 10 bytes
- $33,334 / (1,000 / 20) = 667$ blocks

$.T(E_R) = 33,334$ ו $B(E_R) = 667$ בוכן

התמונה היא ברגע זו:

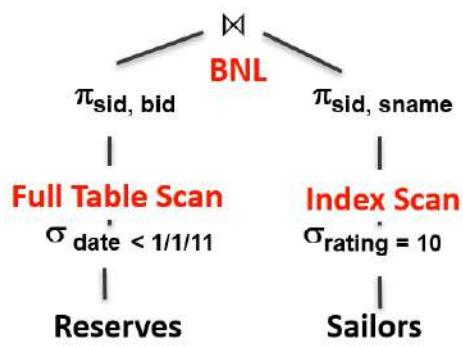
$$E_R = \pi_{sid,bid} \sigma_{date < '1/1/01'} Reserves$$

$$E_S = \pi_{sid,sname} \sigma_{rating=10} Sailors$$

$$\begin{aligned} \text{Read}(E_S) &= 106 \\ B(E_S) &= 2 \\ T(E_S) &= 100 \end{aligned}$$

$$\begin{aligned} \text{Read}(E_R) &= 2,500 \\ B(E_R) &= 667 \\ T(E_R) &= 33,334 \end{aligned}$$

חישוב BNL ראשוני



**Read(E_R) = 2,500
 $B(E_R)$ = 667
 $T(E_R)$ = 33,334**

M=50

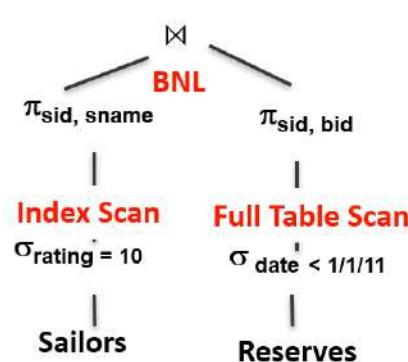
**Read(E_S) = 106
 $B(E_S)$ = 2
 $T(E_S)$ = 100**

**Read(E_R) +
 Read(E_S) * [B(E_R)/(M-2)]**

נציב בנוסחה ונקבל:

$$2500 + 106 * \left\lceil \frac{667}{50 - 2} \right\rceil = 3984$$

חישוב BNL שני



**Read(E_R) = 2,500
 $B(E_R)$ = 667
 $T(E_R)$ = 33,334**

M=50

**Read(E_S) = 106
 $B(E_S)$ = 2
 $T(E_S)$ = 100**

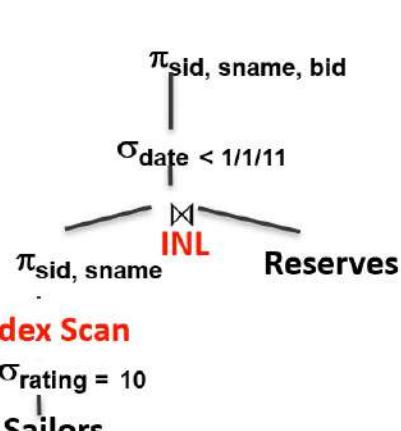
**Read(E_S) +
 Read(E_R) * [B(E_S)/(M-2)]**

נציב ונקבל:

$$106 + 2500 * \left\lceil \frac{2}{50 - 2} \right\rceil = 2606$$

leshim lab: makiun shachri hachira vohatla be l'S nikkas b'shi biutio p'nei blokim, at biutio p'nei anachnu kora'im pum achot belbad! lkn ha'ulot hiya kriah pum achot shel biutio chitzoni, pum achot shel biutio p'nei.

חישוב JNL



**Read(E_R) = 2,500
 $B(E_R)$ = 667
 $T(E_R)$ = 33,334**

**Read(E_S) = 106
 $B(E_S)$ = 2
 $T(E_S)$ = 100**

Read(E_S) + T(E_S) * cost_of_select

$$106 + 100 * ???$$

נשלם את עלות קריית הביטוי החיצוני, ועוד (E_S) T ככלומר כל שורה שמתאימה לתנאי החיצוני (100) נשלם את עלות פעולה הבחירה ב-???. נחשב אותה:

- 10,000 tuples of Sailors
- 100,000 tuples of Reserves
- Only 1% of ratings are 10
- Index on R.sid of branching factor 100

- Each tuple from S will have $100,000 / 10,000 = 10$ matching tuples from R
- Finding them costs $\log_{50} 100,000 + 10 / 49 + 10 = 14$ I/Os

Read(E_S) + T(E_S)* cost_of_select

כל שורה ב-Sailors יהיה לו ממוצע $10 = \frac{100,000}{10,000}$ שורות מתאימות ב-Reserves. איך הגיענו להערכתה זו? ערכו sid ב-Sailors. מכיון שאין לנו מידע מיוחד, נשתמש בהנחה ההטפלגות האחדה (כלומר הערכים השונים ל-sailors מתפלגים באופן אחיד בין השורות ב-Reserves). יש לנו 10,000 ערכי sid שונים ב-Sailors אז נניח שככל שערך sid חוזר 10 פעמים ב-Reserves.

במה זה יעלם? ירידה באינדקס עליה $\log_{50} 100,000$ ועוד קריית העלים המתאימים (יהיה רק אחד מתאימים, כי כל 10 השורות יכנסו בעלה 1) ועוד 10 גישות לדיסק לקרוא השורות המתאימות.סה"כ 14 פעולות I/O.

סה"כ עלות החישוב:

$$106 + 100 * \mathbf{14} = 1506$$

זה יותר מ-7BN, וזה כי לא קראנו את כל Reserves אלא רק את השורות שהתאימו ל-Sailors.

Week 8: Framework (Design Theory)

FD 1: Intro 8.1

תיאוריות התבונן

תחום של נירמול מסד הנתונים. מהו תבונן טוב של מסד נתונים?

נירמול מסד נתונים

מושג שמתיחס לתבונן סכימה שמקיימת עקרונות מסוימים. מהם היתרונות של נירמול?

(1) מהיר יותר לבצע פעולות עדכון על מידע במד מנורמל

(2) פעולות הכנסה מהירות יותר

(3) טבלאות קטנות יותר

(4) קל יותר לשמור על נכונות המידע (כל פריט מידע הוא אמין ונכון)

מה החיסרון? בשנצריך לחשב שאלות בסיס מינימל יהו למ' יותר פעולות צירוף מאשר במקרה לא מנורמל. בגודל נרצה לנורמל כמה שנוכל, עד שנגלה בעיה ביצועים הקשורה לנירמול – רק אז נרצה לחזור אחרונית ולבנות דה-נורמליזציה.

מוטיבציה

Empld	EmpName	Office	OffPhone	Customer1	Customer2	Customer3
1003	Mary Smith	Chicago	312-5551212	Ford	GM	
1004	John Hunt	New York	917-5551784	Dell	HP	Apple
1005	Martin Hap	Chicago	312-5551212	Boeing		

מה הבעה בטבלה זו? הקשיות בכמה הלקחות – אולי נרצה שלעובד יהו למ' יותר מ-3 לקוחות? ולעובד שיש לו פחות אנחנו רואים ערכי נט. ומה אם נרצה לחפש את הנציג שאחראי על הלוקח בשם Ford, איך תראה השאלה?

```
SELECT EmpId
FROM SalesStaff
WHERE Customer1 = 'Ford' or
      Customer2 = 'Ford' or Customer3= 'Ford'
```

בגישה כזו גם לא נוכל להגיד שלכל לקוח יש רק employee אחד שמתפלט בו.

-אלטרנטיבנה-

Empld	EmpName	Office	OffPhone	Customers
1003	Mary Smith	Chicago	312-5551212	{Ford, GM}
1004	John Hunt	New York	917-5551784	{Dell, HP, Apple}
1005	Martin Hap	Chicago	312-5551212	{Boeing}

הפכנו את העמודה customers להיות מטיפוס קבוצה, ומסדי הנתונים בדר"כ מאפשרים זאת. למה גם זה לא טוב? לא אפשר לבטא שיש רק employee אחד שמתפלט בכל customer, בכתבה של שאלות מעלה עמודות של קבוצות או מערכים אין-יעילות.

נציג סדרה של צורות נורמליות שהולכות ומתחזקות. כל אחת היא דרישת על הטבלאות:

צורה נורמללית ראשונה – First Normal Form

כל attribute מכיל רק ערכים אטומיים (אינו קבוצות או מערכים) או אין attribute שחזר על עצמו.

איך נעשה זאת עבור הדוגמה שראינו? נשמר 2 טבלאות:

Empld	Customer
1003	Ford
1003	GM
1004	Dell
1004	HP
1004	Apple
1005	Boeing

בטבלה השמאלית שמרנו מידע על כל עובד, בטבלה הימנית על הלקוחות של העובדים.

נתרכז עבשו בטבלת העובדים (הshmאלית), האם עדין יש כאן בעיות? אנחנו רואים שעובד 1003 ועובד 1005 עובדים באותו משרד בשיקגו, ולכן גם יש להם את אותו מספר. אם אנחנו מניחים שלכל משרד יש טלפון מסויל, אז ביצענו שיכפול מיותר של המידע – אנחנו לא רוצים את זה. זה מגדיל את הצורך באיחסון ואם נרצה לשנות את הנתון נצטרך להיות זהירים.

אם נרצה לשנות את מספר הטלפון במשרד בשיקגו זה יהיה יקר יותר (צריך לעדכן הרבה ערכים, בשכפול רך ערך אחד השתנה), וגם זה חשוף יותר לטעויות. זה נקרא **אנומליות עדכון**.

בעיה נוספת היא **אנומליית הוספה**, קושי בהכנסת נתונים. אם פתחנו משרד חדש ויש לנו מספר טלפון עבורו, אבל אין שם עדיין עובדים. איך נבנис אותו למסד הנתונים?

EmpId	EmpName	Office	OffPhone
1003	Mary Smith	Chicago	312-5551212
1004	John Hunt	New York	917-5551784
1005	Martin Hap	Chicago	312-5551212
???		Atlanta	546-555862

אנומליית מחיקה – מה קורה אם העובד האחרון בניו יורק עזב? איבדנו את מספר הטלפון של המשרד.. מחקנו מידע שלא אמר להיות מחוק רק בגלל הצורה שבה שמרנו את הנתונים.

הינו רוצים להיות מסוגלים לקחת את התהוושה הזאת של מסד נתונים טוב או לא טוב, ומשה להוכיח שהטבלה מתוכננת היטב או לא. איך עושים את זה?

תבונןロー

מה יוצרת הבעיה? ידענו ש-office קבוע מספר טלפון. זה נקרא תלות פונקציונאלית: אם תנתנו לי משרד, אחזיר לכם מספר טלפון אחד ויחיד. אבל במשרדים רבים עובדים, אז נוצר מצב שאחננו שומרם אותו שוב ושוב. זה מבון תלוי, כי אילו זו של עובד היה קבוע מספר טלפון, גם בכל משרד היה עובד יחיד – לא הייתה כאן אף כפילות.

$\text{EmpId} \rightarrow \text{OffPhone}$ $\text{Office} \rightarrow \text{EmpId}$

כדי להכריע, אנחנו חייבים מידע על העולם בו אנחנו עובדים.

מטרות

- (1) לתת הגדרה פורמלית כך שבינהן סכמה בניה נוכל להגיד האם היא מתוכננת היטב או לא – הכרעה בוליאנית.
- (2) אחרי שקבעי אם סכמה לא טובה, איך נוכל לתקן אותה עם סכמה בניה היטב?

FD 2: Functional Dependencies 8.2

תלות פונקציונאלית

נטזיות:

*Write next to each other
to indicate set union*

$$\begin{aligned} XY &= X \cup Y \\ XA &= X \cup \{A\} \\ ABC &= \{ABC\} \\ XX &= X \cup X = X \end{aligned}$$

A, B, C, ...
A single attribute

..., X, Y,
Sets of attributes

R
Name of a relation

R(A,B,C)

r
instance of a relation

A	B	C	s
1	2	3	s
1	2	4	t

s, t
a tuple in an instance

$$\begin{aligned} s[A] &= t[A] \\ s[AB] &= t[AB] \\ s[AC] &\neq t[AC] \end{aligned}$$

נתון לנו:

Given:

Functional dependency

$$X \rightarrow Y$$

holds in r if,
for every two tuples s, t $\in r$:

$$\text{If } s[X] = t[X] \text{ then } s[Y] = t[Y]$$

- $R(A_1, \dots, A_n)$

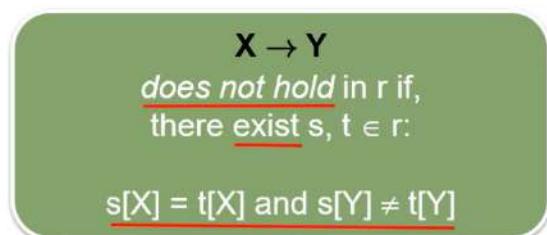
- $X \subseteq A_1, \dots, A_n$

- $Y \subseteq A_1, \dots, A_n$

- Instance r of R

אנו אמר שהתלות הפונקציונלית $Y \rightarrow X$ מתקיים במופיע z אם לכל שתי שורות z, s-B-R, אם $[Z] = t[X] = t[Y]$ אז גם $[Z] = t[Y]$ (אם השורות שוות-ב-X אז הן גם שוות-ב-Y). הערך שנראה ב-X קובע פונקציונלית את הערך שנראה ב-Y. למשל אם אנחנו יודעים שלמשרד יש מספר טלפון בוודד, בכל פעם שנראה שם של משרד נוכל לדעת שבעמודת מספר הטלפון יהיה אותו ערך.

בדרכו שליליה נראה:



כأن מספיק למצאו זוג מופיעים בודד של היחס R שבו אנחנו רואים ששני ערכי X זהים מלאוים בערכי Y שונים (למשל, אותו משרד עם שני מספרי טלפון).

דוגמה 1

Example

A	B	C	D	E	F
a_1	b_1	c_1	d_1	e_1	f_1
a_1	b_1	c_1	d_2	e_2	f_2
a_1	b_1	c_1	d_3	e_3	f_1

Does $A \rightarrow D$ hold in this instance?

התשובה היא לא. כי אנחנו רואים a_1, d_1 וגם a_1, d_2 .

Does $A \rightarrow B$ hold in this instance?

כן. כל זוג אנחנו רואים שיש a_1, b_1 .

Does $ABD \rightarrow CFE$ hold in this instance?

כן. מאחר שהשורות שוות ב- ABD אך ה"אם" של התנאי לא מתקיים, ולכן זה לא משנה מה קורה ב- CFE , הכל קורה באופן ריק.

A	B	C	D	E	F
a_1	b_1	c_1	d_1	e_1	f_1
a_1	b_1	c_1	d_2	e_2	f_2
a_1	b_1	c_1	d_3	e_3	f_1

בכה קורה לכל זוג שורות.

Does $AC \rightarrow BF$ hold in this instance?

לא.

A	B	C	D	E	F
a_1	b_1	c_1	d_1	e_1	f_1
a_1	b_1	c_1	d_2	e_2	f_2
a_1	b_1	c_1	d_3	e_3	f_1

Does $CF \rightarrow F$ hold in this instance?

זה חייב להתקיים, יש כאן טאוטולוגיה.

A	B	C	D	E	F
a_1	b_1	c_1	d_1	e_1	f_1
a_1	b_1	c_1	d_2	e_2	f_2
a_1	b_1	c_1	d_3	e_3	f_1

דוגמה 2

EmpID	EmpName	DeptID	DeptName
0001	John Doe	1	Human Resources
0002	Jane Doe	2	Marketing
0003	John Smith	3	Sales

אילו תלויות פונקציונליות מתקיימות באן?

כל השורות שנות בכל אחד מהאטריבוטים, ולכן כל תלות פונקציונלית תתקיים. בעודם האמיתי זה לא כך, אנחנו צריכים נציג את מתכון הסכמה באן שיתשאל את החברה ויסביר לנו את התלויות.

דוגמה 3

EmpID	EmpName	DeptID	DeptName
0001	John Doe	1	Human Resources
0002	Jane Doe	2	Marketing
0003	John Smith	3	Sales
0004	Jane Goodall	1	Human Resources

באן אנחנו רואים שהתלוות $DeptID \rightarrow EmpID$, כי יש לנו שתי שורות עם מחלקת 1, שבה יש שני עובדים שונים. לעומת כן מתקיים. איך נדע אם זהה מקרי?שוב, נשאל את מתכון הסכמה.

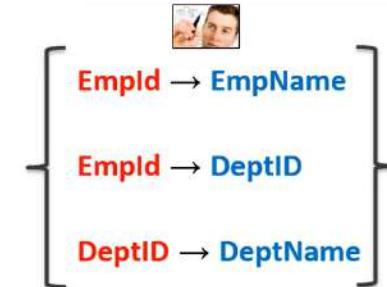
FD 3: Implications 8.3

נביעה לוגית

מתכון הסכמה צריך להגיד אילו תלויות אמורות להתקיים מעל הסכמה. אבל, יתכן שהוא יציין רק חלק מהן. בהינתן קבוצת תלויות, נרצה להיות מסוגלים להסיק מה התלויות הנוספות שמתקיימות.

דוגמה

EmpID	EmpName	DeptID	DeptName
0001	John Doe	1	Human Resources
0002	Jane Doe	2	Marketing
0003	John Smith	1	Human Resources
0004	Jane Goodall	3	Sales



What can we infer from these dependencies?

אנחנו רואים את הנחות מתכון הסכמה בסוגרים. מהתלוות האלו ניתן להסיק גם $EmpID \rightarrow DeptName$. אפשר גם להסיק ... $EmpName, DeptId \rightarrow EmpName$

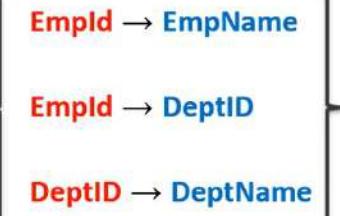
הסקת תלות

מה זה אומר שם-F אפשר להסיק תלות? נסתכל על קבוצת תלויות פונקציונליות F, ותלות ספציפית $Y \rightarrow X$.

נאמר שם-F משתמע $Y \rightarrow X$ אם לכל מופיע z של R אם התלוות ב-F מתקיימות, אז גם $Y \rightarrow X$ מתקיימת. אם זה נכון, נגד גם ש-Y $\rightarrow X$ מובע מ-F.

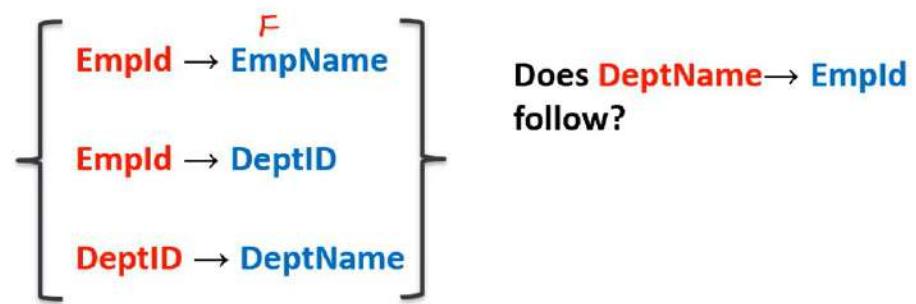
דוגמה

EmpID	EmpName	DeptID	DeptName
		F	



ראינו שהה $Y \rightarrow X$ מתקיימת. אבל איך נוכיח את זה?

יהי r מופיע בו F מתקיימים. נראה שגם $EmpID \rightarrow DeptName$ מתקיים. יהי $t[DeptID] = s[DeptID]$ כך ש- $t[s]$ שורות ב-R. מכיוון ש-F מתקיים נסיק $[DeptName] = t[DeptName]$. שוב, מכיוון ש-F מתקיים, אנחנו יודעים ש- $t[s] = t[DeptName]$. לכן, $t[s]$ לא סותרת את התלוות. אין זוג שורות שסותר את התלוות ולכן היא מקתימת.



נצרך לבנות בaan דוגמה נגדית. צריך לדאוג שהדוגמה תקיים את כל התלותות הנתונות.

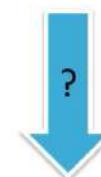
Empld	EmpName	DeptID	DeptName
Empl 1	name1	dep1	Dept1
Empl 2	name2	dep2	Dept1

דוגמאות נוספות

Functional Dependencies: Implication

Student	Dept	Head
---------	------	------

$\text{Student} \rightarrow \text{Dept}$
 $\text{Dept} \rightarrow \text{Head}$



① $\text{Student} \rightarrow \text{Head}$ ② $\text{Head} \rightarrow \text{Student}$

③ $\text{Student, Dept} \rightarrow \text{Head}$ ④ $\text{Student} \rightarrow \text{Student}$

(1) נכון. טריוויאליות.

(2) לא נכון. אין מונע שמנהל אחת מהלקה אחד יהיה במטה סטודנטים.

(3) נכון. נובעemyידית מהתלות $\text{Dept} \rightarrow \text{Head}$.

(4) נכון. זה נובע גם אם קבוצת התלותות הנתונה (F) הייתה ריקה.

[תלות פונקציונאלית טריוויאלית](#)

When $Y \subseteq X$ we say
that
 $X \rightarrow Y$
is a **trivial** functional
dependency

A trivial functional
dependency follows
from every set of
functional
dependencies

Which of the following
are trivial?

- 1) $D \rightarrow D$
- 2) $SD \rightarrow S$
- 3) $S \rightarrow SD$

אם קבוצת האטראיביטים Y מוכלת בקבוצת האטראיביטים X , אז התלות הפונקציונאלית $Y \rightarrow X$ היא טריוויאלית, ולכן תמיד תתקיים (מעל כל יחס).

דוגמאות בכתום:

- (1) טריוויאלי
- (2) טריוויאלי
- (3) לא טריוויאלי

הברעת הבעה

אנחנו יודעים להכريع תלויות טריוויאליות אבל נרצה גם להכريع תלות לא טריוויאלית. בהינתן קבוצה F ותלוות $Y \rightarrow X$, האם $Y \rightarrow X$ נובע מ- F ? כדי להראות ש- $Y \rightarrow X$ לא נובע מ- F מספיק לנו למצוא דוגמה נגדית. אבל איך נראה שהוא כן נובע מ- F ?

דוגמה נגדית

נמצא שתי שורות שסותרות את התלוות ומקיימות את F . שלב ראשון:

$$F = \left\{ \begin{array}{l} AB \rightarrow C \\ A \rightarrow B \\ C \rightarrow D \\ D \rightarrow B \end{array} \right\}$$

A	B	C	D
Q2		C1	
Q1	b1	C1	d1

Can you show that $C \rightarrow A$ does not follow from F ?

נשים בטבלה את מה שנרצה להפריך. אנחנו רואים 2 שורות זהות ב-C ששונות בערכיהן בעמודה A. שלב שני:

A	B	C	D
Q2	b1	C1	d1
Q1	b1	C1	d1

הוכחה

Can you show that $C \rightarrow B$ follows from F ?

יהי Z מופיע, יהיו t, s שורות ב- Z .. זה השלב הבא שנלמד.

FD 4: Closure 8.4

הברעת שאלות בביצה לוגית

בהינתן קבוצה של תלויות פונקציונליות F וסת אטריבוטים X , נגידר את הסגור של X ביחס ל- F , ונסמך X_F^+ בקבוצת האטריבוטים A כך ש- $A \rightarrow X$ נובע מ- F .

כלומר כל האטריבוטים A כך שאם F מתקיים אז גם $A \rightarrow X$ מתקיים. לעיתים בש- F -ברור מההקשר נכתב רק X_F

דוגמה

$$F = \left\{ \begin{array}{l} \text{Student} \rightarrow \text{Dept} \\ \text{Dept} \rightarrow \text{Head} \end{array} \right\}$$

Student	Dept	Head

What is Student^+ ?

במובן שיהיה בסגור את Student עצמו, ומהתלוות עצמן אנחנו רואים במפורש שגם Dept מתקיים, וגם Head בסגור מטריציטיביות. לכן:

$$\text{Student}_F^+ = \{\text{Student}, \text{Dept}, \text{Head}\}$$

אלגוריתם לחישוב סגור

Closure(X, F)

$V := X$

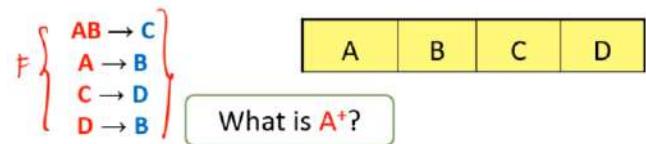
While there is a $Y \rightarrow Z$ in F such that

1. Y is contained in V and
 2. Z is not contained in V
- do** add Z to V

Return V

מתחלים בלשימ אט X עצמו (תלות טריוויאלית). כל עוד יש תלות $Z \rightarrow Y$ ב-Fvr ש-Y מוכל ב-Vvr ו-Z לא, נוסיף את Z ל-Vvr. ההיגיון הוא מהטרדייניטיות. רץ בזמן פולינומיאלי (אפשר לשפר, אבל מבחינתנו עיל).

דוגמא



שלב ראשון: נשים את A. עכשו נחשב תלות בו צד שמאל נמצא ב-Vvr הצד ימין לא. נוסיף את B. עכשו יש AB קובע C וכן נוסיף את C וכן הלאה.

$\checkmark = A B C D$

What is C^+ ?

נתחילה מילשים את C, ולאחר C יגרור D, ומשם גם את B וכן:

האם האלגוריתם נכון?

הוכחה ש- $CLOSURE(X, F) = X_F^+$.

בכיוון אחד נראה $X_F^+ \subseteq CLOSURE(X, F)$ נראה באינדוקציה:

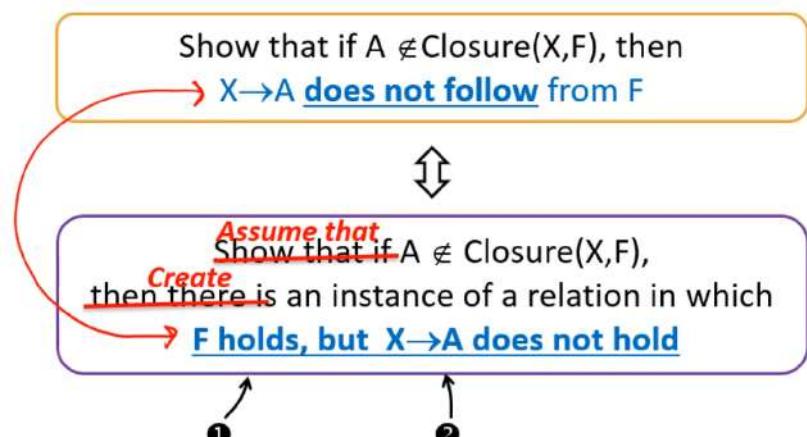
טענה: אם הוספנו את A ל-Vvr לאחר k איטרציות של הלולאה אז $A \rightarrow X$ נובע מ-F.

בסיס: עבור $0 = k$. ברור כי זו תלות טריוויאלית.

צעד האינדוקציה: נניח שהוספנו את A לאחר k איטרציות. אז יש $Z \rightarrow Y$ ב-Fvr ש-Z מוכל ב-Vvr ו-Y מוכל ב-B_ivr. מהתיקים $t[B_i] = t[Z]$ נובע מ-Fvr ש-B_ivr מקיים את Fvr ויהי s שורת ב-zvr. מכיוון ש-Fvr מתקיים $t[X] = t[Z]$ נובע מ-Fvr ש-Xvr מקיים את Fvr. כלומר $t[A] = t[Z]$ נובע מ-Fvr. בפרט $t[A] = t[Z]$ נובע מ-Fvr. כלומר $A \rightarrow X$ נובע מ-Fvr.

בכיוון השני נראה $X_F^+ \supseteq CLOSURE(X, F)$, או במלים אחרות עלינו להראות שגם $A \rightarrow X$ נובע מ-Fvr אם $A \in Closure(X, F)$. אנחנו נראה את השיליה של זה, כלומר שאם $A \notin Closure(X, F)$ אז $A \rightarrow X$ לא נובע מ-Fvr.

כדי להראות את זה אנחנו צריכים להראות שיש מופיע של היחס שבו Fvr מתקיים $A \rightarrow X$ לא מתקיים. מה שנעשה:



איך עושים את זה? נאותחל שתי שורות ריקות. בשורות שנמצאות בתוצאה של חישוב האלגוריתם נשים את אותם ערכים בשתי השורות. בשאר השורות (אלו שלא חזו על ידי האלגוריתם) נשים ערכים שונים בשתי העמודות.

Closure(X,F)	Everything Else
s	0.....0
t	0.....1

איך נראה ש-Fvr מתקיים אבל $A \rightarrow X$ לא מתקיים?

נתחילה בהראות שהמופיע מקיים את Fvr. עבור תלות $Z \rightarrow Y$ ב-Fvr אם Z מוכל בתוצאה. ולכן גם Y וגם Z מופיעים מצד השמאלי של הטעלה למעלה ולבן יהיו עם אותו ערך.

במקרה השני Y לא מוכל בתוצאת האלגוריתם, חלק מ-Y נמצא מצד הימני של הטעלה ואז $[Y]s \neq [Y]t$ ואז השורות שונות ב-Yvr והתלות מתקיימת (כי הן כrüיכות להיות שוות ב-Yvr כדי שייהי על מה לדבר). זה מוכיח ש-Fvr מתקיים.

עכשו נראה שלא מתקיים $A \rightarrow X$. נשים לב שבוואדי (Closure(X,F)) A כי אנחנו מוסיפים אותה באתחול האלגוריתם. A לפ' ההנחה לא הוחזר מהאלגוריתם, אך הוא מצד הימני של הטעלה ומתקיים $[A]s \neq [A]t$. קיבלנו שהשורות שוות ב-Xvr ושונות ב-Avr, כלומר התלות לא מתקיימת.

Definition:

$$X^+ = \{A \mid X \rightarrow A \text{ follows from } F\}$$

Proved!

Theorem: $X^+ = Closure(X, F)$

אם מפה נרצה להתקדם, אנחנו רוצים להסתכל לא רק על אטריבואיטים בודדים, אלא גם האם קבוצות אטריבואיטים מקיימות:

Lemma: $X \rightarrow Y$ follows from F , if and only if $Y \subseteq X^+$

$X \rightarrow Y$ follows from F , if and only if $Y \subseteq \text{Closure}(X, F)$

FD 5: Keys and Superkeys 8.5

מפתח

X is a superkey in R if $X^+ = R$

X is a key in R if

- 1) $X^+ = R$ and
- 2) For all $Y \subset X$, $Y^+ \subset R$

מפתח על – קבוצת האטריבואיטים X הוא מפתח על ב- R אם הסגור שלו הוא R (אם במשמעותם את הסגור מקבלים את כל האטריבואיטים).

מפתח – אם הוא מפתח על, וגם מינימלי: אם נוריד ממנו אטריבואיט נקבל קבוצה שהיא בבר איננה מפתח על. זה מושג חזק יותר ממפתח על.

דוגמה 1

$\text{Student} \rightarrow \text{Dept}$

Student	Dept	Head
---------	------	------

- 1) Find all superkeys
- 2) Find all keys

נרשום את כל תת-הקבוצות האפשרות: H, SDH, SD, SH, DH, S, D, H

מפתחות על:

- כमובן SDH הוא מפתח על.
- האם SD מפתח על? לא, כי לא נוכל להוסיף את H.
- האם SH מפתח על? כן, כי S יגרור D נקבל SHD.
- האם DH מפתח על? לא, כי לא נוכל להוסיף את S.
- כל היחידונים גם לא יעבדו.

מפתחות – איך נמצאו אותן? קבוצות של אטריבואיטים שהם גם מפתחות על וגם מינימליות (או אפשר להוריד משם ועדיין נשאר עם מפתח על).

SDH הוא לא מפתח, כי מצאנו תת-קבוצה שלו שהיא גם מפתח על: SH. לכן רק SH מפתח.

דוגמה 2

$\text{Dept} \rightarrow \text{Head}$
 $\text{Head} \rightarrow \text{Dept}$

Student	Dept	Head
---------	------	------

- 1) Find all superkeys
- 2) Find all keys

מפתחות על: SH, SD, SDH.

מפתחות: SH, SD.

דוגמה 3

$\text{Student, Dept} \rightarrow \text{Head}$
 $\text{Head} \rightarrow \text{Student, Dept}$

Student	Dept	Head
---------	------	------

- 1) Find all superkeys
- 2) Find all keys

מפתחות על: HD, HS, H, SD, SDH.

מפתחות: SD, H.

\emptyset (no functional dependencies)	Student	Dept	Head
1) Find all superkeys 2) Find all keys			

מכיוון שאין לנו תלויות, אז לכל X הסגור שלו יהיה רק הוא עצמו. אז מפתח הול היחיד יהיה SDH והוא יהיה גם מפתח יחיד.

[אלגוריתם למציאת מפתח](#)

Minimize(X, F):

```

for each A ∈ X
  if A ∈ {X - A}+
    then X = X - {A}
Return X
  
```

FindKey(R, F):

```
Return Minimize(R, F)
```

פונקציית Minimize מקבלת קבוצת אטריבואיטים X ומורידה אטריבואיטים מ-X בצורה שהורדת האטריבואיטים לא משנה את הסגור של X. השתמש בה גם בהמשך ولكن היא בקורס.

לכל אטריבואיט A בקבוצת R שנארצנו עליה Minimize בודקים האם הורידה של A מ-X משנה את הסגור, אם A עדין בפנים אז הוא מיותר (השאגנו אותו בלי כולל אותו) אז נוריד אותו. הסדר שבו נורץ את האלגוריתם יכול לשנות את הקבוצה הסופית, אבל אנחנו תמיד נקבל מפתחה.

דוגמת הרצה:

Find a key for
 $R = (A, B, C, D, E)$
 with
 $F = \{A \rightarrow B, BC \rightarrow E, ED \rightarrow A\}$

- ~~A~~
 1) $A \in (BCE)^+ = BCEA$
 2) $B \in (CDE)^+ = CDEAB$
 3) $C \in (DE)^+ = DEAB$
 4) $D \in (CE)^+ = CE$
 5) $E \in (CD)^+ = CD$

מצאנו מפתח אחד עבור הסכימה: CDE, האם יש עוד? כן. איך נמצא אותם? האלגוריתם של מינימיז מחזיר רק אחד. נגענו על זה בחילוק הבא.

Week 9: Normal Forms (Design Theory)

FD 6: All Keys 9.1

[Finding All Keys](#)

אנחנו יודעים למצוא מפתח אחד עבור היחס, וכעת נרצה למצוא את כל המפתחות. ראיינו את האלגוריתם למציאת מפתח בודד.

Minimize(X, F):

for each $A \in X$

if $A \in \{X - A\}^+$

then $X = X - \{A\}$

Return X

FindKey(R, F):

Return Minimize(R, F)

האלגוריתם מקבל קבוצה של אטריבוטים X ומנסה להוריד אטריבוטים מ- X כל עוד הורדתם לא מפריעה לסגור של X . איך נמצא את כל המפתחות?

All Keys(R, F):

$K := \text{FindKey}(R, F)$

KeyQueue := {K}

Keys := {K}

While KeyQueue.isNotEmpty()

$K := \text{KeyQueue.dequeue}()$

Foreach $X \rightarrow Y \in F$ for which $Y \cap K$ is not empty do

$S := K - \{Y\} \cup X$ //S is a superkey!

If S does not contain any $J \in \text{Keys}$ then

$S' := \text{Minimize}(S, F)$ //S' is a new key

Add S' to Keys and to KeyQueue

Return Keys

נתחילה ממצא מפתח אחד K . יש שני מבנים: תור המפתחות שבו יהיו המפתחות שכבר מצאנו ובאמצעותם נמצא מפתחות חדשים. יש את קבוצת המפתחות שנרצה להחזיר בסופו. בשנייהם נשים את K .

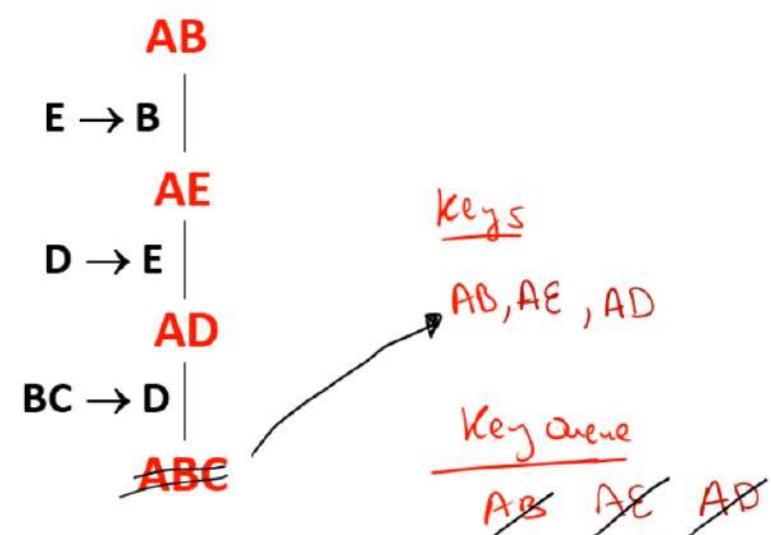
בהמשך, נעבור בלולאה כל עוד המפתחות שנמצאו בהם KeyQueue אינם ריק, נוציא משם מפתח כלשהו ונעבור בלולאה על התלותות הפונקציונליות $Y \rightarrow X$ עבורם צד ימין של התלות יש לו חיתוך לא ריק עם המפתח. אם זה המצב, נגדיר בתור S את הקבוצה $X \cup \{Y \setminus K\}$. הורידנו את צד ימין של התלותות מ- K והוספנו את צד שמאל S כרגע מובטח שהוא מפתח על, כי K היה מפתח לפני כן (הסגור של K הוא כל R). הורידנו מ- K את Y והוספנו לו את X , בודאי ש- Y מוכל בסגור, כי אנחנו מקבלים אותו דרך X . לכן נוכן להמשיך.

יכול להיות ש- S לא מינימלי, או שמצאנו את המפתח שモכל בו. לבדוק: אם הקבוצה S לא מכילה אף מפתח שכבר מצאנו, אז היא מכילה מפתח חדש – נפעיל את פעולה המינימיזציה ונקבל מפתח חדש. נוסיף אותו גם ל- $Keys$ וגם ל- $KeyQueue$. בסוף נחזיר את המפתחות.

זמן ריצה: יתכן של-R יש כמה אקספוננציאלית של מפתחות. לכן, זמן הריצה של האלגוריתם אינו פולינומי. האלגוריתם רץ בזמן פולינומיAli בקלט ובפלט. במקרים בהם יש מעט מפתחות זמן הריצה קצר, אם יש הרבה יותר – אבל אין מנוס כי צריך למצוא את כולם.

Can you find all keys for
 $R = ABCDE$
 $F = \{BC \rightarrow D, D \rightarrow E,$
 $A \rightarrow C, E \rightarrow B\}?$

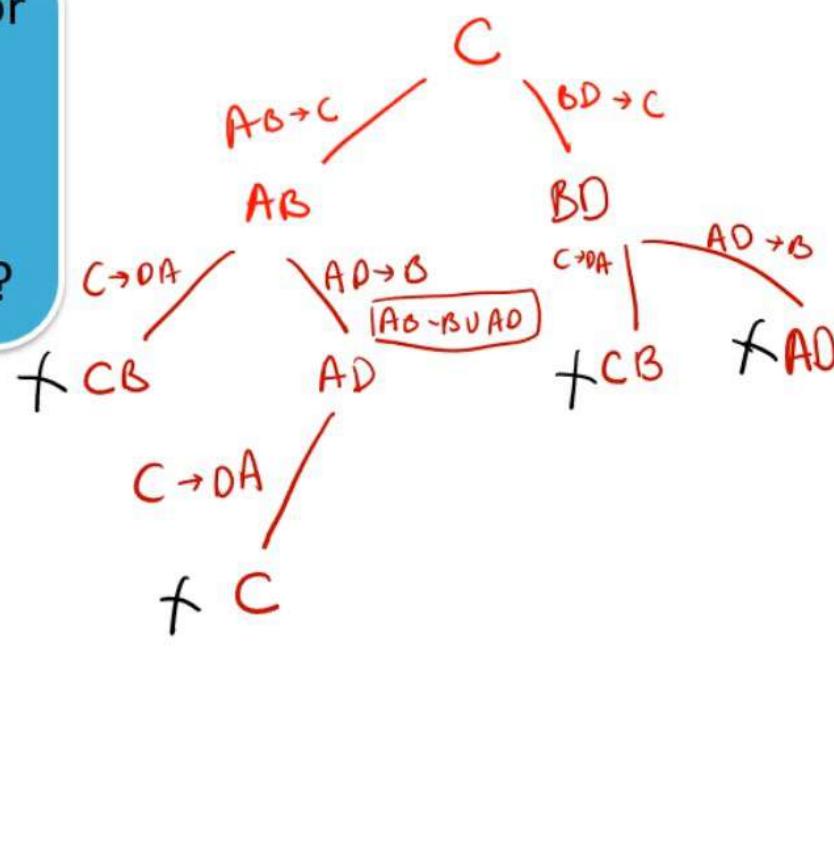
מתחלים ממצא מפתח כלשהו. נתחיל עם AB , נכניס אותו ל-*Keys* ול-*KeyQueue*.
 נוציא מפתח AB , בולגר את AB ונמצא תלויות שצד ימין שלhn יש להן חיתוך לא ריק עם AB . יש תלות אחת כזו, $B \rightarrow E$. נחליף את B בצד שמאל של התלוות E ונקבל את AE . AE לא מכיל מפתח שכבר מצאנו ולכן מכיל מפתח חדש. נפעיל עליו את מינימיז'ינג ונגלה שהוא מינימלי. נוסיף את AE לקבוצת המפתחות וגם לתור.
 נוציא מהטור את AE ונחפש תלויות עם חיתוך לא ריק בצד ימין, יש אחת כזו: $E \rightarrow D$. נחליף את הצד ימין עם הצד שמאל ונקבל את AD . זה לא מכיל אף מפתח שמצאנו, נפעיל עליו מינימיז'ינג ונקבל שהוא מינימלי ולכן נוסיף אותו לשני התורמים. ממשיכים הלאה:
 עבשוי נבחר את התלוות $D \rightarrow BC$, נחליף ונקבל את התלוות ABC , שהיא מכילה בבראשית AB שהוא מפתח שמצאנו. לא נעשה שום דבר עם ABC .
 עבשוי תור המפתחות התרוקן, אז מצאנו את כלם.



לשים לב: כל המפתחות שמצאנו מכילות את האטראיביט A . למה זה לא מפטייע? בתלוות אין אף תלות שיש בה את A בצד ימין, זה אומר ש- A חייב להיות חלק מהמפתח כי איז אפשר להוסיף אותו לסגור בלי זה.

Can you find all keys for
 $R = ABCD$
 $F = \{AB \rightarrow C, C \rightarrow DA,$
 $BD \rightarrow C, AD \rightarrow B\}?$

Keys
 C, AB, BD, AD



צורות נורמלליות

צורות נורמלליות הן צורות בהן אנחנו רוצים לראות את הטעלאות שלנו כדי להיות בטוחים שהן בניות בצורה הגיונית. דיברנו על 1st Normal Form (1NF) שפה אטראטיבית יש ערך בודד ואין חזקה על אטראטיביטם. לדבר היום על שני סוגי נוספים: (2NF, BCNF, 3NF, 4NF).

על הצורה הנורמללית השנייה נזהר.

הצורות הנורמלליות באות לפטור את בעיית היתירות בטעלאות – כלומר כאשר יש ערכים בטבלה שהם שכפולים של ערכים אחרים ויכולנו להסיק אותם בלי שהם יהיו בתובים. ערכים מיותרם כגון דורותים יותר זיכרנו וגם עשויים ליצור בעיות בזמן עדכון ערכים.

ערך בטבלה הוא **מיותר** אם היה אפשר להסיק את הערך בעזרת השורות האחרות בהינתן התלות פונקציונאלית. דוגמה:

A	B	C
a	a	a
b	a	a

הערך הוווד מיותר, יכולנו להסיק אותו בעזרת הערך שמופיע בשורה הראשונה. לעומת זאת, מה עם נתון $C \rightarrow A$, האם הערך בוודוד עדין מיותר? לא! אין לנו דרך להשיג אותו כי ערכי A שונים בין השורות.

יתירות בטعلاות

נתון היחס $R = (A, B, C)$. נסתכל על תלויות פונקציונאליות שונות:

\emptyset

במקרה זה, לא יכולה להיות שום יתירות בטعلاה, כי לא נוכל להסיק ערכים בטvlaה משורות אחרות בלי אף תלות פונקציונאלית.

$A \rightarrow B$

יכול להיות יתירות, למשל:

A	B	C
a	a	a
a	a	a
a	b	c

הערך הממורך מיותר, יכולנו להסיק אותו בלבד (באותה מידה יכולנו להגיד שהוא שווה שמעליו מיותר)

$A \rightarrow B$
 $B \rightarrow C$

A	B	C
a1	b	c
a2	b	c

כאו יש ערך מיותר, יכולנו להסיק אותו בלבד.

$A \rightarrow B$
 $A \rightarrow C$

בשביל יתירות צריך 2 שורות זהות לצד שמאל של התלות ולהסיק משהו על צד ימין. בשתי התלות האטראטיביטם בשמאל הוא A אך נשים את אותו ערך:

A	B	C
a	b	c
a	b	c

כאן למעשה בינו מופיע עם שורה אחת, כי יש כאן חזרה. לא הצליחנו לבנות מקירה עם שתי שורות זהות על צד שמאל ושותות על צד ימין. לא יכולה להיות באן יתירות בכלל. למה זה קרה? כי A הוא **מפתח**, ולכן לא יכולה להיות יתירות בטבלה. בשצד שמאל של התלות הוא מפתח המצביע טוב, בשצד שמאל אינו מפתח – יכולה להיות יתירות.

[Boyce-Codd Normal Form \(BCNF/4NF\)](#)

R is in **Boyce-Codd Normal Form (BCNF)** if for every $X \rightarrow Y$ implied by F one of the following holds

- $Y \subseteq X$, i.e., $X \rightarrow Y$ is trivial or
- X is a superkey of R

יחס R יהיה בצורה נורמלית BCNF אם לכל תלות $Y \rightarrow X$ שנובעת מ-F מתקיימת אחת משתי תכונות: או שהתלות טריוויאלית (צד ימין מוכל מצד שמאל) או הצד שמאל הוא מפתח על של R. זה לא נותן לנו דרך נוחה לבדוק מה הצורה הנורמלית של היחס, כי אנחנו צריכים בשайл זה לבחון כל תלות שנובעת מ-F. נראה הגדרה שקולה:

R is in **Boyce-Codd Normal Form (BCNF)** if for every $X \rightarrow Y$ in F one of the following holds

- $Y \subseteq X$, i.e., $X \rightarrow Y$ is trivial or
- X is a superkey of R

הגדרה זו אומرت ש-R הוא BCNF אם "מכל תלות $Y \rightarrow X$ ב-F עצמו, או הצד שמאל X הוא מפתח על. זו בעיה פולינומיאלית. לבדוק אם תלות היא טריוויאלית זה פשוט מאד. לבדוק האם הצד שמאל הוא מפתח על, נחשב את הסגור של X עם אלגוריתם-hClosure שרצה בזמן פול".

טיפים:

1. אם יש רק 2 אטראיביטים, היחס מיד ב-BCNF.
2. אם כל מפתח הוא אטראיביט בודד אז $R \in BCNF \Leftrightarrow R \in 3NF$.
3. אם כל האטראיביטים מופיעים במפתח כלשהו אנחנו לפחות לפחות ב-3NF ועוד רק נותר לבדוק האם אנחנו ב-BCNF.

[Third Normal Form \(3NF\)](#)

צורה נוספת נספה, חלשה יותר מ-BCNF.

R is in **Third Normal Form (3NF)** if for every $X \rightarrow Y$ implied by F, one of the following holds

- X is a superkey of R or
- for each $A \in Y$, either $A \in X$ or A is an attribute in a key

R יהיה בצורה נורמלית שלישית אם לכל $Y \rightarrow X$ שנובע מ-F מתקיים אחד משני התנאים: או הצד שמאל X הוא מפתח על, או שלכל אטראיביט הצד ימין הוא או בברצד שמאל, או ש-A הוא אטראיביט שופיע באיזשהו מפתח.שוב, זה לא נותן לנו דרך נוחה לבדוק אם R כלשהו הוא ב-3NF כי היא מדברת על מה שנובע מ-F. הגדרה שקולה:

R is in **Third Normal Form (3NF)** if for every $X \rightarrow Y$ in F, one of the following holds

- X is a superkey of R or
- for each $A \in Y$, either $A \in X$ or A is an attribute in a key

מספיק לנו לבדוק לכל תלות האם הצד שמאל הוא מפתח על, או שלכל אטראיביט הצד ימין מתקיים שהוא מופיע הצד שמאל או הוא מפתח שמכיל אותו. מבחינת זמן ריצה לבדוק האם X מפתח על זה בזמן פול" (ראינו קודם), לבדוק אם A שיר ל-X הוא בעיה פולינומיאלית. לבדוק האם A הוא אטראיביט במפתח זו בעיה NP-שלמה. לכן, הכרעה זו באופן כללי היא בעיה NP-שלמה.

[הקשר בין הצורות הנורמלליות](#)

נשים לב שגם $R \in 3NF$, אבל ההפך לא מתקיים.

נראה עבשו מספר דוגמאות, ובכל דוגמה נזכיר האם היחס הוא ב-BCNF או שהוא ב-3NF או אף אחד מהם. סוג זה של שאלה יחוור הרבה.

דוגמה 1

Empld	EmpName	Office	OffPhone
-------	---------	--------	----------

Empld → EmpName, Office
Office → OffPhone

נבדוק קודם אם היחס הוא ב-BCNF. נעבור על כל התלותות.

$$Empld \rightarrow EmpName, Office$$

- זו לא תלות טריוויאלית.

$$(Empld)^+ = Empld, EmpName, Office, OffPhone$$

- בין היחסים אין מפתח עליון.

$$Office \rightarrow OffPhone$$

- זו לא תלות טריוויאלית.

$$(Office)^+ = Office, OfficePhone$$

- בין היחסים אין מפתח עליון.

נבדוק האם היחס הוא ב-3NF. לשכם כך, נctrיך את כל המפתחות ביחס. יש רק מפתח אחד ליחס זה $Empld$. עברו התלות $Office \rightarrow OffPhone$ לא מוכל באף מפתח של היחס, ולכן היחס גם **לא ב-3NF**.

דוגמה 2

Empld	EmpName	Office	OffPhone
-------	---------	--------	----------

Empld → EmpName, Office
Office → Empld, OffPhone

האם היחס הוא ב-BCNF? נעבור על כל התלותות. נשים לב שאף אחת מהן לא טריוויאלית אז נבדוק אם מופיעה בתחום מפתח.

נחשב סגור:

$$Empld^+ = \{Empld, EmpName, Office, OffPhone\}$$

לכן זה מפתח עליון.

$$Office^+ = \{Office, Empld, OffPhone, EmpName\}$$

גם כאן צד שמאל הוא מפתח עליון.

לכן היחס הוא ב-BCNF (שרה לא אמרה בהרצאה, אבל זה גורר גם שהוא ב-3NF מהטענה שראינו קודם).

דוגמה 3

What is the normal form (BCNF / 3NF / neither) of a relation $R=(A,B,C)$ if the functional dependencies F are:

\emptyset

מה קורה באשר $F = \emptyset$? התנאי שדרשנו מתקיים באופן ריק! לכל תלות התלות היא או טריוויאלית או מפתח עליון, בקבוצה ריקה של תלותות בכל התנאים של BCNF מתקיימים באופן ריק. לכן היחס **ב-BCNF**.

What is the normal form (BCNF / 3NF / neither) of a relation $R=(A,B,C)$ if the functional dependencies F are:

$A \rightarrow B$

יש לנו תלות בודדת. זו תלות לא טריוויאלית, אבל A לא מפתח עליון (חסר את C) ולכן תלות לא מקיימת את תנאי ה-BCNF. האם מקיימת את תנאי $3NF$? אכן קל לראות שפתח יחיד הוא AC . במקרה זה, B הוא לא יחס בפתח, ולכן גם $3NF$ לא מתקיים.

What is the normal form (BCNF / 3NF / neither) of a relation $R=(A,B,C)$ if the functional dependencies F are:

A → B, B → C

נסתכל האם זה ב-BCNF. נחשב את הסגור של A: $A^+ = \{A, B, C\}$ וזה אכן מפתח על. לעומת זאת, B אינו מפתח על כי $\{B, C\} = B^+$. לכן לא ב-BCNF.

האם זה ב-3NF? יש מפתח אחד שהוא A. לכן בתלות $C \rightarrow B$ האטריבואט C לא מופיע באך מפתח, והפרנו את התנאי. גם לא ב-3NF.

What is the normal form (BCNF / 3NF / neither) of a relation $R=(A,B,C)$ if the functional dependencies F are:

A → B, A → C

נשים לב שהסגור של A הוא A, B, C ולבן היחס הוא ב-BCNF.

סיכום הדוגמאות: נשים לב ש-4 הדוגמאות שראינו עבשו מתאיםות ל-4 שראינו בהתחלה – איפה שיש יתירות אלו וחסמים שהם לא ב-BCNF, ואיפה שלא היה הם כן היו ב-BCNF.

דוגמה 4

What is the normal form (BCNF / 3NF / neither) of a relation $R=(A,B,C)$ if the functional dependencies F are:

A → B, B → A

תלות ראשונה: התלות אינה טריואילית. מתקיים: $\{A, B\} = A^+$ ולכן שמאלו שלה אינה מפתח על. כדי למצאו האם היחס הוא ב-3NF צריך את כל המפתחות. בכך יש שני מפתחות: AC, BC.

בשתי התלותות צד ימין הוא שדה במפתח, ולכן נגיד שהיחס הוא ב-3NF. הוא לא ב-BCNF כי צד ימין לא מפתח על, אבל הוא כן ב-3NF כי צד ימין הוא שדה במפתח.

סיכום

	BCNF	3NF
Complexity	PTIME	NP-complete
Redundancy	none	some
Achievable with Lossless Join Decompositions	yes	yes
Achievable with Lossless Join, Dependency Preserving Decompositions	no	yes

אם היחס הוא ב-BCNF לא תהיה אף יתירות ביחס – זה מובטח לנו. ב-3NF יכול להיות מעט יתירות, כי התלותות הפונקציונאליות יש להם הצד ימין אטריבואטים ממפתח שכונראה לא ייחזרו הרבה, אבל יתכנו.

שתי השורות האחרונות עוסקות בשאלת האם תמיד ניתן להשיג סכימות כאלה, ובין אותן טוב יותר בהמשך. מה שקרה לנו כאן זה ה-3NF ב-BCNF בשורה الأخيرة. זה נותן מוטיבציה לשימוש ב-3NF. בשנרגча לבנות סכימות נרצה שלא תהיה בהם יתירות ועוד תכונות נוספות. ב-3NF תמיד ניתן להבטיח את התכונות הנוספות.

בשלב זה אנחנו יודעים בהינתן סכימה ותלות פונקציונאליות להכريع האם הסכימה מתוכנתה היטב. במקרה סכימה שלא עומדת בקריטריונים הללו.

הימן רצים שככל אחד מהיחסים שלנו יהיה ב-BCNF או לפחות ב-3NF. מה נעשה אם זה לא המצב?

[מהו פירוק?](#)

SDH	Student	Dept	Head
	Levy	CS	Rubin
	Cohen	Math	Bush
	Barak	CS	Rubin

מהי הצורה הנורמללית של היחס?

האם היחס ב-BCNF? לשם כך נצטרך לבדוק כל אחד מהתליות האם היא טריוויאלית או האם צד שמאל הוא מפתח על התלוות אינה טריוויאלית. $S, D, H = S^+ = D^+ = H^+$ ולכן מתקיים תנאי ה-BCNF. תלוות שנייה אינה טריוויאלית, וקייםנו $D \rightarrow H$ וזה סותר את דרישת BCNF. האם במקרה קיבלנו יחס ב-3NF? נמצא את המפתחות, המפתח היחיד הוא S . לבן התלוות השנייה מפירה גם את תנאי 3NF. היא לא טריוויאלית, צד שמאל לא מפתח על, צד ימין לא שדה במפתח.

נרצה לשפר את הטעלה כדי שהיא תהיה בצורה נורמללית טוביה יותר. נעשה זאת על ידי פירוק היחס לתתי היחסים, כדי להקטין את כמה היתירות של המידע בטבלה:

SDH	Student	Dept	Head
	Levy	CS	Rubin
	Cohen	Math	Bush
	Barak	CS	Rubin

Student	Dept
Levy	CS
Cohen	Math
Barak	CS

 $\pi_{\text{Student}, \text{Dept}} \text{ SDH}$

Dept	Head
CS	Rubin
Math	Bush

 $\pi_{\text{Dept}, \text{Head}} \text{ SDH}$

במקום היחס המקורי נשמר שני תת-יחסים. האם שיפורנו את מצבינו מבחינה הצורה הנורמללית של הסכימה? שמי תת-היחסים הם ב-BCNF. איך אנחנו יודעים את זה? כי כל יחס עם שני אטריבואיטים הוא תמיד ב-BCNF.

איך נשחרר את היחס המקורי? אנחנו יכולים לשחרר אותו על ידי צירוף טבעי בין שני היחסים.

Student	Dept	Dept	Head
Levy	CS		Rubin
Cohen	Math		Bush
Barak	CS		

איך אפשר לוודא שהתליות הפונקציונליות שהוגדרו מעל היחס המקורי ממשיכות להתקיים גם אחרי הפירוק לשני היחסים, וגם אחרי צירופם חזרה ליחס אחד גדול. בדוגמה שלנו זה מאד פשוט, אפשר לוודא $D \rightarrow S$ בתת הסכימה השמאלית, ו- $S \rightarrow H$ בתת הסכימה הימנית. מה ראיינו שעבור יפה?

- (1) כל תת סכימה היא ב-BCNF.
- (2) אפשר לשחרר את היחס המקורי באמצעות צירוף טבעי.
- (3) אפשר לוודא שתליות פונקציונליות עדין מתקיימות בזמן הבדיקה.

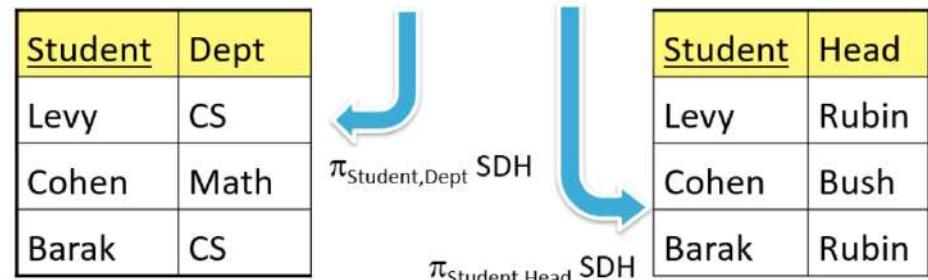
האם זה היה מקרי? יכולנו לבחור כל פירוק וזה היה טוב באותה מידת?

נפרק את אותו יחס, הפעם כר' :

What about decomposing to (Student, Dept) (Student, Head)?

נקבל:

SDH		
Student	Dept	Head
Levy	CS	Rubin
Cohen	Math	Bush
Barak	CS	Rubin



שני היחסים ב-BCNF כי כל אחד מהם מכיל רק 2 אטראיביטים.

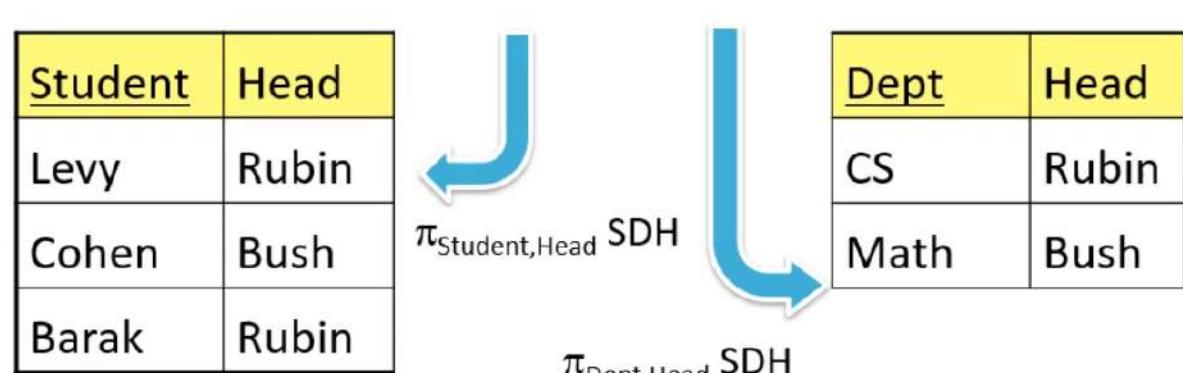
אם נחשב צירוף טבעי על שני תתי היחסים, נקבל זירה את היחס המקורי. אפשר לשחזר את השלשות על ידי ✎ . מה לגבי בדיקת נכונות של תלויות פונקציונליות? אם ננסה את הה收敛ות הבאות:

Student	Dept	What problems can you see with this?	Student	Head
Levy	CS		Levy	Rubin
Cohen	Math	What happens if we try to insert: (Smith, Math) (Smith, Rubin)	Cohen	Bush
Barak	CS		Barak	Rubin

בכל טבלה בנפרד לא תהיה בעיה. אבל בשנכרף את הטעויות נקבל ביחס המצורף את: $Dept \rightarrow Head$ בסתירה לתלות ש-Head. יש לנו אז רואים $Math \rightarrow Rubin$, $Math \rightarrow Bush$ וגם $Math \rightarrow Bush$ בסתירה לתלות. הפירוק הזה לא טוב.

What about decomposing to (Student, Head) (Dept, Head)?

SDH		
Student	Dept	Head
Levy	CS	Rubin
Cohen	Math	Bush
Barak	CS	Rubin



שוב, כל אחת מתתי הסכימות היא ב-BCNF (כי יש רק 2 אטריבואיטים). מה יקרה בשנפער צירוף? נקבל את היחס המקורי. אבל, זה לא בהכרח נכון!

SDH	Student	Dept	Head
$\text{Student} \rightarrow \text{Dept}$	Levy	CS	Rubin
$\text{Dept} \rightarrow \text{Head}$	Cohen	Math	Bush
	Barak	CS Physics	Rubin

Student	Head	Dept	Head
Levy	Rubin	CS	Rubin
Cohen	Bush	Math	Bush
Barak	Rubin	Physics	Rubin

What problems can you see with this?

עבשו בשנעשה צירוף טבעי משהו מוזר.. נקבל את השורה *Levy, Physics, Rubin Levy, CS, Rubin* כי הצירוף הוא על שדה *Head*. אותו דבר יהיה גם ב-CS וגם ב-Physics עם *Rubin* במקום *Barak*. הפירוק בעייתי כי הוא מאבד את המידע על השלשות, ולא יוכל לשזר.

תבונות פירוק

A **decomposition** of R is a set of relations R_1, \dots, R_n such that the set of attributes in R_1, \dots, R_n is the same as in R

פירוק של R הוא קבוצת יחסים R_n, \dots, R_1 , כך שסקא אטראיבואיטים ב- R_1, \dots, R_n זהה ל-R.

למשל, אם $R = (A, B, C, D)$ אז יכול להיות $R_1 = (A, B), R_2 = (B, D), R_3 = (C, D, A)$ כל עוד ביחיד יש את כל האטראיבואיטים בייחס R.

אנחנו נתעניין ב-3 תכונות:

Necessary Property:

Lossless = Can reconstruct R by natural joins

Desired Property:

Dependency Preserving = Can locally verify FDs

Desired Property:

In BCNF or 3NF

1. **פירוק ללא אובדן** – נוכל לשזר את היחס המקורי על ידי צירוף טבעי. רוצים להיות נתונים שלא משתנה מה תוכן היחס, צירוף טבעי יחזק לנו את היחס המקורי.

2. **שמור תלויות** – רוצים להיות מסוגלים לבדוק שככל אחת מהתלוויות הפונקציונאליות מתקיימות על תת-הסכימות, ולהבטיח את קיומן על היחס המלא אחרי הצירוף.

3. **צורה נורמללית** – כל תת-סכמה ביחס תהיה מצורה BCNF או 3NF.

פירוק ללא אובדן

תכונה הכרחית שאומרת שאפשר לשחזר את היחס המקורי מיחס הפירוק על ידי חישוב הצירוף הטבעי, ומובטח לנו לקבל את התוצאה הנכונה חוזרת. הגדרה פורמלית:

A decomposition of R into R_1, \dots, R_n is a **lossless join decomposition** with respect to F if, for every instance r of R that satisfies F :

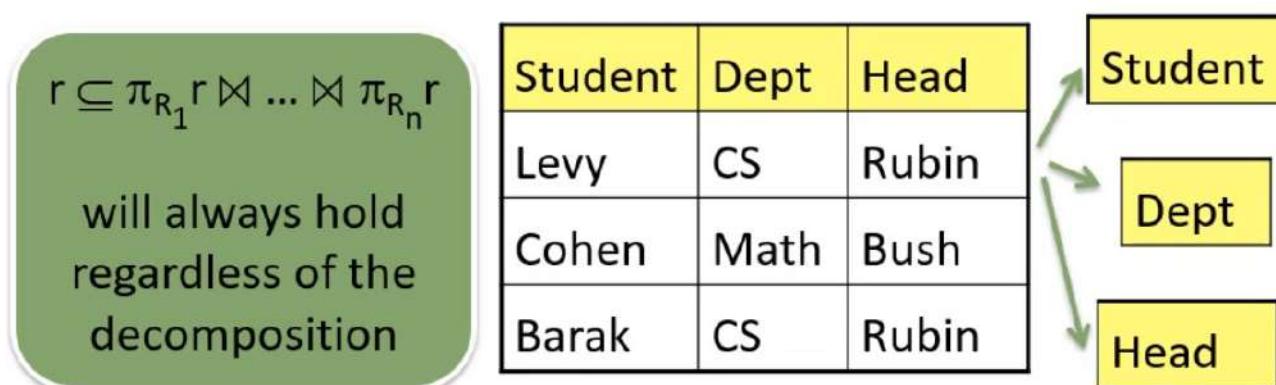
$$r = \pi_{R_1} r \bowtie \dots \bowtie \pi_{R_n} r$$

פירוק של R לתתי יחסים R_1, \dots, R_n הוא **לא אובדן** ביחס לקבוצת התלות הפונקציונליות F אם לכל מופיע r שמקיים את F מתקיים:

$$r = \pi_{R_1} r \bowtie \dots \bowtie \pi_{R_n} r$$

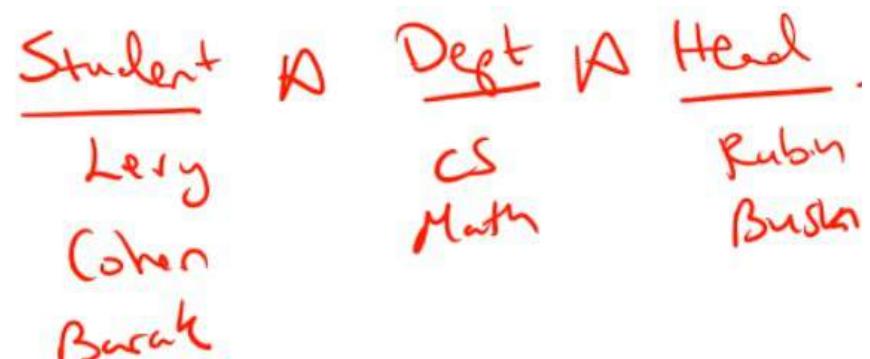
המשמעות היא שאם נטיל את r על כל אחד מתתי היחסים בסכימה, ונחשב את הצירוף הטבעי, נקבל חוזרת את r . אם זה מתקיים, במקומות לשמר את היחס המקורי, אפשר לשמור את ההטלות של R על כל אחד מתתי היחסים ומכל לשחזר את R המקורי. לא נרצה להשתמש בפירוק עם אובדן לעולם.

נשים לב שכיוון אחד של התנאי הזה תמיד מתקיים:



זו טענה שהוכחנו כשדרנו על שקוליות בין ביטויים אלגבריים. לא משנהאיזה פירוק ניקח, היחס המקורי יהיה מוכל בצירוף ההטלות. הכיוון השני לאו דווקא מתקיים.

באן החלטנו לפרק כל אטריבוט לסכמה נפרדת. איך זה יראה?



מה יקרה אם נעשה צירוף טבעי? אין ערכים חוזרים لكن זה כמו מכפלת קרטזית. היחס המקורי בוודאי מוכל ביחס החדש שבנינו, אבל מוכל ממש, כי יש הרבה שורות שלא היו ביחס המקורי. לכן יש $r \bowtie \pi_{R_1} r \bowtie \dots \bowtie \pi_{R_n} r$.

מה קורה בצד השני?

A decomposition of R into R_1, R_2 is a **lossless join decomposition** with respect to F if, at least one of the following holds:

- 1) $R_1 \subseteq (R_1 \cap R_2)^+$
- 2) $R_2 \subseteq (R_1 \cap R_2)^+$

יש מקרה מיוחד שבו הפירוק הוא לשני יחסים. במקרה הכללי אפשר לפרק את היחס לכמה בלשיי של יחס, אבל המקרה של שני יחסים הוא קל במיוחד וכאן נתעבב עליו. פירוק של R ל- R_1, R_2 הוא **לא אובדן** יחסית ל- F אם לפחות אחד משני התנאים מתקיימים.

דוגמה 1

Is this a lossless join decomposition of $R = (A, B, C)$ when $F = \{A \rightarrow B, B \rightarrow C\}$?

$$\begin{aligned} R_1 &= (A, B) \\ R_2 &= (B, C) \end{aligned}$$

$$(R_1 \cap R_2)^+ = (B)^+ = BC$$

מתקיים $\subseteq R_2 \subseteq BC$ ולכן הפירוק לא אובדן.

דוגמה 2

Is this a lossless join decomposition of $R = (A, B, C)$ when $F = \{A \rightarrow B, B \rightarrow C\}$?

$$\begin{aligned} R_1 &= (A, B) \\ R_2 &= (A, C) \end{aligned}$$

$$(R_1 \cap R_2)^+ = (A)^+ = ABC$$

מתקיים $\subseteq R_2 \subseteq ABC$ והאמת גם $R_1 \subseteq ABC$ (לא נדרש, אבל נכון). ולכן הפירוק לא אובדן.

דוגמה 3

Is this a lossless join decomposition of $R = (A, B, C)$ when $F = \{A \rightarrow B, B \rightarrow C\}$?

$$\begin{aligned} R_1 &= (A, C) \\ R_2 &= (B, C) \end{aligned}$$

$$(R_1 \cap R_2)^+ = (C)^+ = C$$

מתקיים $C \not\subseteq R_2$ וגם $R_1 \not\subseteq C$. ולכן הפירוק הוא עם אובדן.

למה זה עובד ב**ן**? למה אם אחד משני התנאים מתקיים אז לכל מופיע r שקיימים את F מתקיים:

$$\pi_{R_1}r \bowtie \pi_{R_2}r \subseteq r$$

(כבר רأינו שהצד השני מובטח לנו)

נכיה שיש לנו את היחס:

$$R = (A_1, A_2, A_3) \quad R_1 = (A_1, A_2) \quad R_2 = (A_2, A_3)$$

ויש שורה:

$$(a_1, a_2, a_3) \in \pi_{R_1} \wedge \pi_{R_2}$$

נרצה להראות שהשורה הזאת מופיעה גם ביחס R . נמשיך:

$$\begin{aligned} \Rightarrow (a_1, a_2) &\in \pi_{R_1} \wedge (a_2, a_3) \in \pi_{R_2} \\ \Rightarrow a_1, a_2 &\in r \wedge (a_2, a_3) \in r \end{aligned}$$

ונניח בה"ב שהתנאי הראשון מתקיים, כלומר $a_1, a_2 \in r$. אם שני השורות זהות ב- a_2 אז חיבות להיות זהות בכל $R_1 \subseteq (R_1 \cap R_2)^+$ גם בעמודה של a_1 שכן הערך b_2 חייב להיות a_1 מהנחה זו. ולכן:

$$(a_1, a_2, a_3) \in r$$

כנדרש.

פירוק של יותר מ-2 תת-יחסים

פירוק תקין יכול להיות ליותר מ-2 תת-יחסים, כל עוד איחודו כולל את כל האטראיביטים שהיו ב-R המקורי.
איך לבדוק אבל שפירוק כזה הוא lossless?

CreateTable($R=(A_1, \dots, A_k), R_1, \dots, R_n$):

```
T := new Table[n,k]
for i = 1 to n
    for j = 1 to k
        if  $A_j \in R_i$ 
            then T[i,j] =  $a_j$ 
        else T[i,j] =  $b_{ij}$ 
return T
```

נתון לנו יחס R עם k אטראיביטים, ופירוק ל- n תת-יחסים. כדי לבדוק אם הפירוק הוא ללא אובדן בניית טבלה בגודל $k \times n$. שורה לכל אחת מתתי הסכימות ועומודה לכל אטראיביט.

נמלא את הטבלה כך: בשגיגע לתא המתאים לשורה ה- i והעמודה ה- j אם האטראיביט A_i נמצא בתת הסכימה R_j נשים בתא את הערך a_j אחרת נשים בתא את הערך b_{ij} .

דוגמה לטבלה מלאה

	A	B	C	D
AB	a_1	a_2	$b_{1,3}$	$b_{1,4}$
BC	$b_{2,1}$	a_2	a_3	$b_{2,4}$
CD	$b_{3,1}$	$b_{3,2}$	a_3	a_4

המשך האלגוריתםChaseTable(T, F):

While there are rows t, s in T and $X \rightarrow Y$ in F such that

$$t[X] = s[X] \text{ and } t[Y] \neq s[Y]$$

for each A_i in Y do

if $t[A_i] = a_i$ then replace $s[A_i]$ with a_i

else if $s[A_i] = a_i$ then replace $t[A_i]$ with a_i

else replace $t[A_i]$ with $s[A_i]$

כל עוד יש שורות t, s בטבלה שיצרנו ואיזשהי תלות פונקציונלית כך שהשורות סותרות את התלות הפונקציונלית אנחנו נתקן את התלוויות. איך? נסתכל על כל אטראיביט $-Y$ שבו סתרנו את התלות הפונקציונלית ונdag לשווין. אם באחת מהשורות היה ערך a_i נשים שם a_i אחרת נעשה החלפה.

דוגמה

	A	B	C	D
AB	a_1	a_2	$b_{1,3}$	$b_{1,4}$
BC	$b_{2,1}$	a_2	a_3	$b_{2,4}$
CD	$b_{3,1}$	$b_{3,2}$	a_3	a_4

יש סתירה לתלות, השורות זהות ב- B אבל שונות ב- C :

	A	B	C	D
AB	a_1	a_2	$b_{1,3}$	$b_{1,4}$
BC	$b_{2,1}$	a_2	a_3	$b_{2,4}$
CD	$b_{3,1}$	$b_{3,2}$	a_3	a_4

לכן נעשה שינוי ונקבע:

	A	B	C	D
AB	a_1	a_2	a_3	$b_{1,4}$
BC	$b_{2,1}$	a_2	a_3	$b_{2,4}$
CD	$b_{3,1}$	$b_{3,2}$	a_3	a_4

נמשיך הלאה, אבחנו רואים סתירה לתלות $C \rightarrow D$:

	A	B	C	D
AB	a_1	a_2	a_3	$b_{1,4}$
BC	$b_{2,1}$	a_2	a_3	$b_{2,4}$
CD	$b_{3,1}$	$b_{3,2}$	a_3	a_4

נשוויה:

	A	B	C	D
AB	a_1	a_2	a_3	$b_{2,4}$
BC	$b_{2,1}$	a_2	a_3	$b_{2,4}$
CD	$b_{3,1}$	$b_{3,2}$	a_3	a_4

עוד סתירה:

	A	B	C	D
AB	a_1	a_2	a_3	$b_{2,4}$
BC	$b_{2,1}$	a_2	a_3	$b_{2,4}$
CD	$b_{3,1}$	$b_{3,2}$	a_3	a_4

נשוויה על ידי החלפת $b_{2,4}$ ל- a_4 , אבל בכל מקום בו מופיע $b_{2,4}$ ולא רק בסתירה המסומנת:

	A	B	C	D
AB	a_1	a_2	a_3	a_4
BC	$b_{2,1}$	a_2	a_3	a_4
CD	$b_{3,1}$	$b_{3,2}$	a_3	a_4

עבשו אין סתירות.

המשך האלגוריתם

TestDecomposition(R,R₁,...,R_n,F):

T = CreateTable(R,R₁,...,R_n)

ChaseTable(T,F)

If T contains a row with only “a” values

return “lossless”

else return “not lossless”

בדוק תנאי פשוט – אם יש שורה שבה יש רק ערך a אז נגיד שהפירוק הוא ללא אובדן. בדוגמה שלנו:

	A	B	C	D
AB	a_1	a_2	a_3	a_4
BC	$b_{2,1}$	a_2	a_3	a_4
CD	$b_{3,1}$	$b_{3,2}$	a_3	a_4

לכן הפירוק שלנו הוא ללא אובדן.

דוגמה נגדית

מה שמיוחד בשיטה זו, זה שהיא מייצרת דוגמה נגדית להיווטו של הפירוק ללא אובדן. אם הוא לא יהווה דוגמה נגדית, הוא מוכיח שהיחס הוא ללא אובדן.

	A	B	C	D
AB	a_1	a_2	$b_{1,3}$	$b_{1,4}$
BC	$b_{2,1}$	a_2	a_3	$b_{2,4}$
CD	$b_{3,1}$	$b_{3,2}$	a_3	a_4

$R = (A, B, C, D)$
 $R_1 = (A, B)$
 $R_2 = (B, C)$
 $R_3 = (C, D)$

	A	B
AB	a_1	a_2
BC	$b_{2,1}$	a_2
CD	$b_{3,1}$	$b_{3,2}$

	B	C
AB	a_2	$b_{1,3}$
BC	a_2	a_3
CD	$b_{3,2}$	a_3

	C	D
AB	$b_{1,3}$	$b_{1,4}$
BC	a_3	$b_{2,4}$
CD	a_3	a_4

אם נטיל את היחס הזה על תת-הסכימות ולאחר מכן נצוף חזרה אנחנו נקבל את השורה $a_1 a_2 a_3 a_4$. איך? מסומן. היחס שבנו הוא הכללי ביותר שיכול לייצר תוצאה שambil את השורה הספציפית זו, כי הייתה חייבת להיות שורה ביחס המקורי בר:

	A	B	C	D
AB	a_1	a_2	$b_{1,3}$	$b_{1,4}$
BC	$b_{2,1}$	a_2	a_3	$b_{2,4}$
CD	$b_{3,1}$	$b_{3,2}$	a_3	a_4

אנו יודעים שם נעשה הטלה ונצוף חזרה נקבל את השורה המבוקשת. נראה שגם שורה שלא הייתה ביחס המקורי, אבל עבשו נראה שגם למעשה שורה שהייתה חייבת להיות שם בגל התלות הפונקציונליות.

למה האלגוריתם עובד? (יוטר חשוב לדעת להריץ את האלגוריתם)

אם סיימנו את האלגוריתם עם יחס שבו לא הייתה שורה שכולה a אז מצאנו דוגמה ל-instance שמקיימת את התליות, לא מכילה את השורה a_1, \dots, a_n . אבל כצפוי עליה על תת-הסכימות ונצוף בחזרה נמצא שורה שלא הייתה ביחס המקורי – הדוגמה זו היא דוגמה נגדית שמרה ש-

$$r \neq \pi_{R_1} r \bowtie \dots \bowtie \pi_{R_n} r$$

אם לעומת זאת היחס שבנו יש בו שורה שכולה a אז תהיליך שהאלגוריתם עובר בתיקון התליות, אפשר להשתמש בו כדי לסייע הוכחה.

דוגמה 1

Is this a lossless join decomposition?

$$\begin{aligned} R &= (A, B, C, D) \\ R_1 &= (A, D) \\ R_2 &= (B, D) \\ R_3 &= (A, C) \end{aligned}$$

$$\begin{aligned} F &= \\ &\{B \rightarrow C, \\ &C \rightarrow D\} \end{aligned}$$

אתחול טבלה:

	A	B	C	D
AD				
BD				
AC				

מילוי טבלה, נתחל מילמלא a :

	A	B	C	D
AD	a_1			a_4
BD		a_2		a_4
AC	a_1		a_3	

: b

	A	B	C	D
AD	a_1	b_{12}	b_{13}	a_4
BD	b_{21}	a_2	b_{23}	a_4
AC	a_1	b_{32}	a_3	b_{34}

נתכן לפי תלויות פונקציונאליות:

התלות $C \rightarrow B$ obar מתקיימת, וגם התלות $D \rightarrow C$. לכן סימנו להריץ את האלגוריתם. האם יש שורה שכולה a ? לא. לכן זה פירוק עם אובדן.

דוגמה 2

נשנה לאותו הפירוק אבל עם תלויות פונקציונליות חדשות:

Is this a lossless join decomposition?

$$\begin{aligned} R &= (A, B, C, D) \\ R_1 &= (A, D) \\ R_2 &= (B, D) \\ R_3 &= (A, C) \end{aligned}$$

$$\begin{aligned} F &= \{C \rightarrow B, \\ &D \rightarrow C, \\ &A \rightarrow C, \\ &AD \rightarrow C\} \end{aligned}$$

	A	B	C	D
AD	a_1	b_{12}	b_{13}	a_4
BD	b_{21}	a_2	b_{23}	a_4
AC	a_1	b_{32}	a_3	b_{34}

נתkan את הסתיירות לתלויות:

בבר מתקיים. $C \rightarrow B$

לא מתקיים: $C \rightarrow D$

	A	B	C	D
AD	a_1	b_{12}	b_{13}	a_4
BD	b_{21}	a_2	$\cancel{b_{23}}$	a_4
AC	a_1	b_{32}	a_3	b_{34}

התלויות $C \rightarrow A$ עם סטיירה, נתkan:

אחד מהערכים ב-C שאנו חנכו צריכים לתקן הוא ערך a אז נתkan לערך a . נחליף את $b_{1,3}$ לכל מקום בו הוא מופיע ל- a_3 :

	A	B	C	D
AD	a_1	b_{12}	$\cancel{b_{13}}$	a_4
BD	b_{21}	a_2	a_3	$\cancel{b_{23}}$
AC	a_1	b_{32}	a_3	b_{34}

לשים לב - קיבלנו סטיירה לתלוית $B \rightarrow C$ שבתחלתה כן התקיימה. נתkan:

	A	B	C	D
AD	a_1	$\cancel{b_{12}}$	$\cancel{b_{13}}$	a_4
BD	b_{21}	a_2	a_3	$\cancel{b_{23}}$
AC	a_1	b_{32}	a_3	b_{34}

תיקנו את השורה הראשונה. בעקרון האלגוריתם דורש להמשיך לתקן עד שנסרים, אבל נבחין שכבר קיבלנו שורה שכולה a אז אפשר לעזoor ולהכريع שבפירוק אין אובדן.

Week 10: Decompositions

FD 11: Dependency Preservation 10.1

בפעם הקודמת בדקנו האם פירוק הוא ללא אובדן. בעת נשוב לתוכנית שימור תלויות – נוכל לבדוק שתליות נשמרות בתמי הסכימות ולא נדרש לצרף אותן כדי לוודא את שימור התליות.

[שימור תלויות](#)

$$\begin{aligned} R &= (A,B,C,D) \\ R_1 &= (A,D) \\ R_2 &= (B,D) \\ R_3 &= (A,C) \end{aligned}$$

$$\begin{aligned} F &= \{C \rightarrow B, \\ &\quad D \rightarrow C, \\ &\quad A \rightarrow C, \\ &\quad AD \rightarrow C\} \end{aligned}$$

If we decompose R to R_1, R_2, R_3 , how we make sure that $D \rightarrow C$ continues to hold?

D and C no longer appear together in a relation!

air נודא שההכנות ל- R_1, R_2, R_3 שהתלות הפונקציונלית למשל $C \rightarrow D$ ממשיכה להתקיים ביחס הגדל. אנחנו רוחים לוודא האם נעשה $R_1 \bowtie R_2 \bowtie R_3 \bowtie R$ עדין נקבל בחזרה את היחס R כאשר כל התליות בו מתקיימות. התלות $C \rightarrow D$ לא מוכלת במלואה באף אחד מתתי היחסים.

נגיד שפירוק משמר תלויות אם מספיק לבדוק שככל אחד מהתליות המוגדרות לוקאלית על תת היחסים מתקיימים כדי שהתליות יתקיימו על כלל היחס.

[הטלה](#)

Given: Relation R , decomposition $R_1 \dots R_n$ and dependencies F

The **projection** of F onto R_i is the set of all functional dependencies following from F , that involve only attributes in R_i :

$$F_{R_i} = \{V \rightarrow W \text{ s.t. } V, W \subseteq R_i \text{ and } V \rightarrow W \text{ follows from } F\}$$

הטלה של F על R_i הוא אוסף התליות הפונקציונאליות שנובעות מ- F - שמדוברים רק על אט�יביטים ב- R_i .

[דוגמיה](#)

Given: $R = (A,B,C)$, $R_1 = (A,B)$ and $F = \{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$, what is the projection of F on R_1 ?

במקרה זה: $F_{R_1} = \{A \rightarrow B, B \rightarrow A, A \rightarrow A, B \rightarrow B, AB \rightarrow A, AB \rightarrow B, B \rightarrow BA, A \rightarrow BA\}$ לא מופיעה ישירות ב- F - שמדוברים רק על אטטיביטים ב- R_1 . אבל כן Nobut ממנה.

[מתי פירוק משמר תלויות?](#)

The decomposition of R into R_1, \dots, R_n is **dependency preserving** if for all $X \rightarrow Y$:
 $X \rightarrow Y$ follows from F
if and only if
 $X \rightarrow Y$ follows from $(F_{R_1} \cup \dots \cup F_{R_n})$

פירוק של R לתתי היחסים R_1, \dots, R_n משמר תלויות אם לכל $Y \rightarrow X$:
 $Y \rightarrow X$ נובע מ- $F \Rightarrow Y \rightarrow X$ נובע מאיחוד הטלות של F על כל תת היחסים.
מספיק לוודא שאין מקיים את האיחוד הזה כדי לוודא שאין מקיים כל תלות שנובעת מ- F .

Given: $R = (A, B, C)$, $R_1 = (A, B)$, $R_2 = (B, C)$ and $F = \{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$, is this decomposition dependency preserving?

מספיק לנו לבדוק את התלות ב- F עצמה, ולא גם אתה מה שנבע מ- F .
 האם $A \rightarrow B$ נובע מהקבוצה $F_{R_1} \cup F_{R_2}$? כן. $A \rightarrow B$ נמצא ב- F_{R_1} .
 האם $C \rightarrow B$ נובע מהקבוצה $F_{R_1} \cup F_{R_2}$? כן. $C \rightarrow B$ נמצא ב- F_{R_2} .
 האם $A \rightarrow C$ נובע מהקבוצה $F_{R_1} \cup F_{R_2}$? כן. נובע מאיחוד היחסים.
 לכן זה פירוק משמר תלויות. בשנכנים מידע לנו $B \rightarrow A$ וגם $A \rightarrow B$ נודע $A \rightarrow B$ ואז יובטח לנו שאנו שמרנו גם את התלות בטבלה הגדולה.

המקרה הכללי

אם Y נמצאים ב- R_i אז בוודאי $Y \rightarrow X$ מתקיים. לכן נוכל להסתפק בלבד בבדיקה את התלות של אוסף מולאות במלואן באחד מהתי היחסים.

אלגוריתם לבדיקת שימור תלויות

IsDependencyPreserving(R, R_1, \dots, R_n, F):

For each $X \rightarrow Y$ in F do:

$Z := X$

repeat

$Z' := Z$

for $i = 1$ to n do

$Z := Z \cup ((Z \cap R_i)^+ \cap R_i)$

until $Z = Z'$

if Y is not contained in Z then return "NO"

Return "YES"

רץ בזמן פול. נמנע מחישוב של איחוד הטלות של F .

עובדרים תלות תלות על התלות ב- F ומוגדים שכל אחת מהן נשמרת. נתחל את Z לקבל את הערך X . קובעים $Z' = Z$, עוברים בלולאה על כל אחת מהתי הסכומות ומחשבים את $(Z \cap R_i)^+ \cap R_i$ ($Z \cap R_i$) ש- Z זה מחשב את הסגור של צד שמאל יחסית לתת הסכימה R_i , מוביל לחשב את F_{R_i} . לא מאד חשוב שבין איך זה עבד – בפועל זה ברור יותר. אחרי שסימנו בודקים האם $Z \subseteq Y$. אם לא, אנחנו מחדרים את הערך NO, ככלומר לא שימרנו את התלות. אם הצליחנו לשמר את כל התלות נחזיר YES.

$$\begin{aligned} R &= (A, B, C, D) \\ R_1 &= (A, B) \\ R_2 &= (B, C) \\ R_3 &= (C, D) \end{aligned}$$

$$\begin{aligned} F &= \{A \xrightarrow{\checkmark} B, B \xrightarrow{\checkmark} C, C \xrightarrow{\checkmark} \\ &\quad D, \text{ } D \xrightarrow{\text{red circle}} A\} \end{aligned}$$

השלוש הראשונות אוטומטיות. נבדוק $A \rightarrow D$. נפעיל שוב ושוב את השורה: $Z = Z \cup ((Z \cap R_i)^+ \cap R_i)$. Z עם ? שונים.

נתחיל עם $Z = D$. נתחילה R_1 . נקבל $Z = D \cup \left(\left(\overline{D \cap AB} \right)^+ \cap AB \right) = D$ וכאן נשארמו רק עם D . כדי להתקדם אנחנו צריכים לעבוד עם משה שמכיל את Z ככלומר $Z = D \cup (DABC \cap DC)$ הוא $DABC$ ואז נקבל $Z = D \cup ((D \cap DC)^+ \cap DC) = DC$. נמשיך:

$$\begin{aligned} R_2 \quad Z &= D \cup \left(\left(\overline{DC \cap BC} \right)^+ \cap BC \right) = DCB \\ R_3 \quad Z &= DC \cup \left(\left(\overline{DC \cap BC} \right)^+ \cap BC \right) = DCB \\ R_1 \quad Z &= DCB \cup \left(\left(\overline{DCB \cap AB} \right)^+ \cap AB \right) = DCBA \end{aligned}$$

נשים לב שאפשר להשתמש שוב בתת סכימה שכבר עברנו עליה, השתמשנו פעמיים ב- R_1 . קיבלנו $Z \subseteq Y$ ולכן התלות נשמרת.

$$\begin{aligned} R &= (A, B, C, D, E) \\ R_1 &= (A, B, D, E) \\ R_2 &= (D, E, C) \end{aligned}$$

$$\begin{aligned} F &= \{A \rightarrow ABCDE, \\ &\quad BC \rightarrow A, DE \rightarrow C\} \end{aligned}$$

What is the normal form of the original relation R?

נשים לב שהתלות הראשונה צד שמאל A הוא מפתח על, ולכן מקיימת את תכאי BCNF. אבל, הסגור של DE הוא לא R , זו תלות לא טריוויאלית הצד שמאל אינו מפתח על, אך R לא ב-BCNF. אבל, הצד ימין יש אטראיביטוט C שהוא חלק ממפתח ולכן **מתקיים תכאי 3NF**.

Is the decomposition lossless?

יש כאן רק שני אטראיביטוטים לכך קל, נחשב את הסגור $(R_1 \cap R_2)^+$ ונבדוק האם R_1 או R_2 מוכל בסגור.

$$R_2 \subseteq (R_1 \cap R_2)^+ = DEC$$

אכן מתקיים, ולכן הפירוק הוא ללא אובדן.

Is the decomposition dependency preserving?

ונתחל עם לבדוק האם $A \rightarrow BC$, נראה תלות נוספת מורכבת. נפעיל את האלגוריתם:

$$\begin{aligned} Z &= Z \cup ((Z \cap R_i)^+ \cap R_i) \\ Z &= BC \cup ((BC \cap ABDCE)^+ \cap ABDCE) = BC \\ Z &= BC \cup ((BC \cap DEC)^+ \cap DEC) = BC \end{aligned}$$

לא משנה איזה תת יחס הפעלים תמיד נשארנו עם BC ולא הצלחנו להוסיף את A . הפירוק **לא משמר תלויות**.

FD 11: Decomposition Normal Forms 10.2

הצורה הנורמלית של היחסים בפירוק

כל המטריה הייתה להגעה למצב שתתי היחסים הם ב-BCNF או ב-3NF. איך נבדוק זאת? נשים לב שכדי לבדוק מה הצורה הנורמללית של יחס אנחנו רק צריכים לדעת מהן התלותות המוגדרות מעלה היחס. בעת אין לנו אותן, יש לנו את R ואת תת היחסים, ובובוצת תלויות F מעלה R . איך נחליט מהו F_{R_i} לבלי?

לממנו לחשב את זה בהרצאה הקודמת. יש כאן רק קאץ' קטן – חישוב F_{R_i} עלול לקחת זמן אקספוננציאלי. נעשה זאת בכל זאת:

ComputeDependenciesInProjection (R, R_i, F):

$G := \emptyset$

For each $X \subseteq R_i$ do:

Add the dependency $X \rightarrow (X^+ \cap R_i)$ to G

יש כאן פרוצדורה שלא מחשבת את כל הטליה, אבל מספיק ממנה (שколоה להטלה בוליה).
ונתחל בקובוצה ריקה של תלויות. לכל תת קבוצה של אטראיביטוטים מעלה R_i נוסיף את התלות $(X^+ \cap R_i) \rightarrow X$ לקובוצה.

What are the normal forms of the relations in the decomposition?

$$\begin{aligned} R &= (A, B, C, D) \\ R_1 &= (A, C, D) \\ R_2 &= (B, C) \end{aligned}$$

$$\begin{aligned} F &= \{AB \rightarrow C, \\ &\quad C \rightarrow A, C \rightarrow D\} \end{aligned}$$

נתחיל בלייצר תלות לכל תת קבוצה של אטראיביטים ב- R_1 .

$$\begin{array}{ccc} A \rightarrow & AC \rightarrow & ACD \rightarrow \\ C \rightarrow & AD \rightarrow & \\ D \rightarrow & CD \rightarrow & \end{array}$$

בצד ימין של כל תלות נרשום את הסגור של צד שמאל, חיתוך עם R_1 :

$$\begin{array}{ccc} A \rightarrow (A^+ \cap ACD) = A & AC \rightarrow ACD & ACD \rightarrow ACD \\ C \rightarrow CAD & AD \rightarrow AD & \\ D \rightarrow D & CD \rightarrow CDA & \end{array}$$

עכשו נעבור על כל תלות ונבדוק האם היא טריוויאלית או הצד שמאל הוא מפתח על.

ב- $C \rightarrow CAD$ הצד שמאל גורר את כל R_1 (לשים לב, אנחנו משתמשים על מפתחות על ביחס ל- R_1 ולא ביחס ל- R).

$$\begin{array}{ccc} \checkmark A \rightarrow (A^+ \cap ACD) = A & \checkmark AC \rightarrow ACD & \checkmark ACD \rightarrow ACD \\ \checkmark C \rightarrow CAD & \checkmark AD \rightarrow AD & \\ \checkmark D \rightarrow D & \checkmark CD \rightarrow CDA & \end{array}$$

לכן $\in BCNF$. יכלנו קצת לחסוך בזמן, כי זה לא מעניין לחשב $ACD \rightarrow ACD$. גם כן, ברגע שהגילים ש- C הוא מפתח יכולנו לוותר על בדיקה של כל תלות שבה C הצד שמאל, כי בודאי שהיא מפתח על.

צריך לבדוק גם את R_2 , אבל למדנו בתרגול שככל יחס עם 2 אטראיביטים הוא ב- $BCNF$.

דוגמה 2

What are the
normal forms of
the relations in the
decomposition?

$$R = (A, B, C, D)$$

$$R_1 = (A, B, C)$$

$$R_2 = (A, D) \text{ BNF}$$

$$F = \{AB \rightarrow C, \\ C \rightarrow D, D \rightarrow A\}$$

$$\begin{array}{ccc} \checkmark A \rightarrow A & \checkmark AC \rightarrow AC & \checkmark ABC \rightarrow ABC \\ \checkmark B \rightarrow B & \checkmark AB \rightarrow ABC & \\ \text{C } \rightarrow CA & \checkmark CB \rightarrow CBA & \end{array}$$

כל תליות טריוויאליות או הצד שמאל הוא מפתח על. מה קורה ב- $C \rightarrow CA$? התלות לא טריוויאלית הצד ימין הוא לא מפתח על. כבר שולחנו $BCNF$. האם הוא ב- $3NF$? נסתכל על כל אטראיביט הצד ימין ונזדד שאו שהוא מופיע הצד שמאל, או שהוא שדה במפתח. C מופיע הצד שמאל, ו- A מופיע במפתח AB וכן התלות מקיימת את $3NF$. לכן $\in 3NF$.

בעת אנחנו רוצים להיות מסוגלים למצוא פירוק בעל תכונות טובות. לשם כך נצורך להגדיר מושג שנשתמש בו באלגוריתם למציאת פירוק.

בסיסי מינימלי

באשר אנחנו מקבלים קבוצת תלויות, יתכן שחלק מיווצרות. בסיסי מינימלי מכיל רק תלויות פונקציונליות הכרחיות ולא נובעות מהאחרות – אם נוריד אותם נאבד מידע.

דוגמה 1

Can you find a minimal cover for:
 $F = \{A \rightarrow B, B \rightarrow C, A \rightarrow C, A \rightarrow A, CB \rightarrow B\}$

נראה לדלן את F בך שיהיה רק את מה שהכרחי ב-F. אפשר להוריד כל תלות טריוויאלית.

Can you find a minimal cover for:
 $F = \{A \rightarrow B, B \rightarrow C, A \rightarrow C, \cancel{A \rightarrow A}, \cancel{CB \rightarrow B}\}$

אפשר להוריד גם את $C \rightarrow A$ כי היא נובעת משילוב השתיים הראשונות.

דוגמה 2

$$F' = \{AB \rightarrow C, A \rightarrow B\}$$

האם היא מינימלית? לבוארה נראה שאין אפשרות להוריד אף תלות. אם נוריד את $B \rightarrow A$ כבר לא נדע את זה כי זה לא נובע מהשניה, וכן גם לגבי התלוות $C \rightarrow AB$. במקרה יש משהו מיותר, וזה האות B ב- $C \rightarrow AB$. בסיסי מינימלי יהיה:

$$\{A \rightarrow C, A \rightarrow B\}$$

air זיהינו זאת? מהתבוננות בסוגרים:

$$\begin{array}{c} F' = \{AB \rightarrow C, A \rightarrow B\} \\ \hline F'' = \{A \rightarrow C, A \rightarrow B\} \end{array} \quad \begin{array}{l} A^+_{F'} = ABC \\ A^+_{F''} = ABC \end{array} \quad \begin{array}{l} F' \xrightarrow{A \rightarrow C} \\ F'' \xleftarrow{A \rightarrow C} \end{array}$$

הצלחנו ב- F'' לבטא את C דרך AB למורoutes צמצמו את התלוות זו.

הגדרה

A **minimal cover** for a set of functional dependencies F is a set of functional dependencies G such that:

1. **Small Right-hand Side:** Every dependency in G has a single attribute on the right-hand side
2. **Equivalence to F :** Every dependency following from F also follows from G , and vice-versa
3. **Cannot be Made Smaller by Deletions:** If we obtain H from G by deleting one or more dependencies and/or deleting one or more attributes from a dependency, then H is not equivalent to F

בכל תלות צד שמאל הוא אט�יביט בודד, שקיות ל- F , לא ניתן להקטין יותר (אם נקטין – נאבד שקיות).

ComputeMinimalCover(F): $G := \emptyset$ for each $X \rightarrow Y$ in F

for each A in Y

 add $X \rightarrow A$ to Gfor each $X \rightarrow A$ in G

for each B in X

 if $A \in (X-B)^+$ then remove B from $X \rightarrow A$ for each $X \rightarrow A$ in G if $X \rightarrow A$ follows from the other dependencies then remove $X \rightarrow A$

1 {
2 {
3 {

אלגוריתם פול"ב-3 שלבים.

1. נdag שלכל תלות יהיה רק אטריביט אחד בצד ימין. נסיף ל-G את הגרירותים הבודדות של כל אחת מהتلויות. למשל אם היה ב-F את $ABC \rightarrow X$ נוסיף ל-G את $C \rightarrow X, X \rightarrow B, X \rightarrow A$.
2. נמחק אטריבוטים מיותרם מצד שמאל של תלות ב-G. לכל תלות $A \rightarrow X$ ב-G, ננסה להוריד את אחת התלויות ב-X ולבדק האם עדין נוכל להסיק את A גם בלי התלוות שהורדנו. אם כן, מוריד אותה.
3. הורדת תלויות מיותרות – אם $A \rightarrow X$ נובע מהתלוות האחרת, מוריד אותה.

דוגמה מעט מורכבתExample: $F = \{A \rightarrow B, ABCD \rightarrow E, EJ \rightarrow GH, ACDJ \rightarrow EG\}$ **שלב 1**

נכнес את כל התלוות ל-G כך שיהיה אטריביט אחד בלבד בצד ימין:

$$\begin{aligned} A &\rightarrow B \\ AB \cup D \rightarrow E \\ EJ &\rightarrow GH \\ EJ &\rightarrow H \\ A \cup DJ \rightarrow E \\ A \cup DJ \rightarrow G \end{aligned}$$

שלב 2

מוריד אטריבוטים מיותרם מצד שמאל של התלוות:

במובן $B \rightarrow A$ אין איך לדל את צד שמאל. נבדוק עבור $ABCD$ האם אפשר לקבל את E עם פחות אטריבוטים:

$$\begin{aligned} E \& \subset BCD^+ = BCD \\ E \& \subset ACJ^+ = ACD \cup E \\ E \& \subset AD^+ = ADB \\ E \& \subset AC^+ = ACB \end{aligned}$$

הצלחנו להוריד את B.

ב-E אנחנו רואים שהסוגר של כל אחד מהאטריבוטים הוא רק הוא עצמו, ולכן לא ניתן להוריד אף אחד מהאטריבוטים בו. בתלוות $E \rightarrow ACDJ$ קל לראות שאפשר לוותר על J כי בבר ראיינו שבסוגר של ACD יש את E. בתלוות $G \rightarrow ACDJ$ נבחן כי:

$$\begin{aligned} G \& \subset CDJ^+ = CDJ \\ G \& \subset ADJ^+ = ADJ \cup B \\ G \& \subset ACJ^+ = ACJ \cup B \\ G \& \subset ACD = ACD \cup BE \end{aligned}$$

או אפשר להוריד אף אחד.

סיימו את שלב 2 באשר:

$$\begin{array}{l} A \rightarrow B \\ ACD \rightarrow E \\ EJ \rightarrow G \\ EJ \rightarrow H \\ ACD \rightarrow E \\ A \oplus J \rightarrow G \end{array}$$

שלב 3

לכל אחת מהתלוויות נבדוק האם היא מיותרת. בהתחלה נחשב את A^+ ביחס לכל התלוויות הפונקציונליות האחרות בלי התלוות $B \rightarrow A$ ובזאת האם B נמצא בסגור. אחרי שבדקנו ראיון $A = A^+$ וזה לא מופיע כי הפעם היחידה ש- B -מושיע בצד ימין הוא $B \rightarrow A$.

$E \rightarrow ACD$ מושיע פעמיים, لكن בהכרח מיותרת פעמיים אחת.

האם G נמצא בסגור של E ללא התלוות $G \rightarrow JE$? לא. שכן אינה מיותרת.

האם H נמצא בסגור של E ללא התלוות $H \rightarrow JE$? לא. שכן אינה מיותרת.

$E \rightarrow ACD$ זו הפעם היחידה ש- E -מושיע בצד ימין וכן אינה מיותרת.

לבסוף נשים לב ש- $E \rightarrow ACDJ \in G$ גם ללא $G \rightarrow ACDJ$ ולכן היא מיותרת.

כיסוי מינימלי:

$$\begin{array}{l} \checkmark A \rightarrow B \\ \cancel{ACD} \\ \checkmark EJ \rightarrow G \\ \checkmark EJ \rightarrow H \\ \checkmark ACD \rightarrow E \\ \cancel{A \oplus J \rightarrow G} \end{array}$$

FD 14: 3NF Decomposition 10.4

מציאת פירוק

בעת, בהינתן סכמה אנחנו יכולים למצוא פירוק שלה שמקיים:

1. לא אובדן
2. שמירה על תלוויות
3. צורה של 3NF או BCNF

אלגוריתם לפירוק

Find3NFDecomposition(R, F):

$G := \text{ComputeMinimalCover}(F)$

for each $X \rightarrow A$ in G

 add the schema XA

If no schema created contains a key, add a key as a schema

Remove schemas that are contained in other schemas

בשלב ראשון - נחשב כיסוי מינימלי ל- F .

בשלב שני - לכל תלות בכיסוי המינימלי $A \rightarrow X$ מוסיף לפירוק את הסכמה AX

בשלב שלישי - אם אף סכמה שיצרנו לא מכילה מפתח (כלומר כשנחשב את הסגור של כל אחד מהתוי הסכומות – אף אחד לא מפתח על) מוסיף מפתח בסכמה.

בשלב אחרון - אם במקרה יש סכמה שמכולה בסכמה אחרת – גוריד אותה.

mobutu lanu shahatzaa hia pirok l-3NF shamshar taliyot vela' abudan.
am nitnu algoritmim at R sheho a bbar bechora 3NF la mobutu lanu shankel shob at R, yitkan shichzir pirok shel. b'kul mukra, mobutu lanu shbel achd matati hichsim hoa lefchot b-3NF.

דגם

Find a 3NF decomposition for $R = (A,B,C,D,E,G,H,J)$
 $F = \{A \rightarrow B, ABCD \rightarrow E, EJ \rightarrow GH, ACDJ \rightarrow EG\}$

shab rashiun - chisbano l-F cisoi minimali b'shiur ha'kodim:

$$\begin{aligned} A &\rightarrow B \\ EJ &\rightarrow GH \\ EJ &\rightarrow H \\ ACD &\rightarrow EG \end{aligned}$$

shab shni - niyatz tuti yichsim be'pirok:

$$\begin{aligned} R_1 &= AB \\ R_2 &= EJGH \\ R_3 &= EJH \\ R_4 &= ACD \end{aligned}$$

shab shelishi - b'dok ha'm achot matati sckmim mafach ul

$$\begin{aligned} R_1^+ &= AB \\ R_2^+ &= EJGH \\ R_3^+ &= EJH \\ R_4^+ &= ACD \end{aligned}$$

af achot man la mafach ul. nzturk lo hossif bitor teta sckma nospat mafach klesheo. nmazia mafach:

$$\begin{aligned} \cancel{ABCDHJ} \\ ACDJ \end{aligned}$$

$$\begin{aligned} G \in (ACDHJ)^+ \\ H \in (ACDJ)^+ \end{aligned}$$

$$R_5 = ACDJ \quad \text{lakn:}$$

shab rbi'ui - b'dok ha'm yish teta sckma shmolbat batet sckma acheret. ain b'zat.

$$\boxed{\begin{aligned} R_1 &= AB & R_5 &= ACDJ \\ R_2 &= EJGH \\ R_3 &= EJH \\ R_4 &= ACD \end{aligned}}$$

kiyalmo pirok shamshar taliyot la' abudan v-3NF.

1. שימור תלויות – אם נסתכל בדוגמה הקודמת, כל אחת מהTELIOOT נשמרות כי כל תלוות נמצאת ממש בתחום תת סכמה
 2. לא אובדן – נובע מהציוויל שכל תלוות הפכה להיות תת סכמה ושיש לתת סכמה שהיא מפתח עלי. אפשר לבדוק עם האלגוריתם שראינו:
- (השורה שבה נגיעה לה שבל השורה a היא השורה של מפתח העל)

	A	B	C	D	E	F	G	H	J
A	a_1	a_2							
B			a_3	a_4	a_5	a_6	a_7	a_8	
E									
J									
G									
H									
F									
D									
C									
A									

מצאת שורה שהוא כולה a ולכן לא אובדן.

3. כל תת סכמה ב-3NF – אנחנו מוסיפים תת סכמה נוספת תלויות או במפתח (אם אף תת סכמה לא מפתח עלי). אם זה מפתח אז הוא בהכרח ב-BCNF. נוכיח רק את החלק השני לפיו כל תת סכמה שייצרנו מתלות בכיסוי המינימלי הוא לפחות ב-3NF.
- נתון $XA = XA \rightarrow X$. תהא $B \rightarrow Y$ תלוות ב- R_1 . אם $X \in B$. אם ב- R_1 B שדה במפתח. מקרה ב': $X \notin B$, אז במקרה זה $A = A$. אם ב- R_1 $Y = X$ ואחרת אם $X \subset Y$ לא היה מינימלי. Y הוא מפתח ולכן גם כאן מתקיימים התנאי.

FD 15: BCNF Decomposition 10.5

על ידי נראה אלגוריתם למציאת פירוק ללא אובדן וכל אחת מהתנאי הסכומות ב-BCNF אבל לא כולל להבטיח שהפירוק משמר תלויות.

האלגוריתם

זה אלגוריתם לא פולינומייאלי, ורקים אחד פול' – אבל הוא מורכב יותר לבנה.

FindBCNFDecomposition(R, F):

If R is in BCNF

then return R

else let $X \rightarrow Y$ be a BCNF violation

$$R_1 = X^+$$

$$R_2 = X \cup (R - X^+)$$

return FindBNCDFDecomposition(R_1, F_{R_1}) \cup

FindBNCDFDecomposition(R_2, F_{R_2})

אם R הוא ב-BCNF (נבדוק בזמן פול') – נחזיר את R. אחרת, נחפש סטירה לתנאי BCNF כלומר תלוות $Y \rightarrow X$ לא טריוויאלית ו-X אינו מפתח על. מגדירים $R_2 = X^+ \cup (R - X^+)$.

קוראים וקורסיבית לאלגוריתם ושולחים לו פעם אחת את R_1 עם הפעלה F_{R_1} ופעם נוספת את R_2 עם הפעלה F_{R_2} . חישוב ההטלות הוא החלק הלא עיל.

למה זה נותן את הנדרש?

1. מבחינת הצורה – אנחנו נמשיך בקריאות לאלגוריתם עד שנקבל צורה של BCNF.

2. לא אובדן כי $X \rightarrow Y$ הוא מפתח עלי $X^+ \subseteq R_1 \cap R_2$.

דוגמה 1

$$\begin{aligned} R &= (A, B, C, D, E), \\ F &= \{AB \rightarrow C, \\ &\quad DE \rightarrow C, \\ &\quad B \rightarrow E\} \end{aligned}$$

אנו צריכים לבדוק האם R הוא BCNF.

בודוק את הסגור של כל צד שמאל של התלוות:

AB אינו מפתח על כי הסגור שלו הוא $ABCE$. נפרק:

$$\begin{array}{c} ABCD E \\ AB \rightarrow C \\ ABCE \end{array} \quad AB \cup (ABCE - ABCE) = ABD$$

עבשו נבדוק כל אחת מההטלות. ניקח את $ABCE$ ולכל אחד מהarterיביטים נחשב את הסגור שלו.

$$\begin{array}{l} F_{ABCE} \\ A \rightarrow A \\ B \rightarrow BE \end{array}$$

כבר מצאנו תלות לא טריויאלית שבה צד ימין אינו מפתח על. התלות סותרת את תנאי $BCNF$ ונפרק אותה.

$$\begin{array}{c} ABCD E \\ AB \rightarrow C \\ ABCE \\ BE \end{array} \quad AB \cup (ABCE - ABCE) = ABD$$

$$BE \quad B \rightarrow BE \quad B \cup (ABCE - BE) = ABC$$

תת הסכמה BE היא ב- $BCNF$ כי מובילה רק 2 אטריביטים. נבדוק אם ABC הוא ב- $BCNF$. נחשב סגורו:

$$\begin{array}{l} F_{ABC} \\ A \rightarrow A \\ B \rightarrow B \\ C \rightarrow C \\ AB \rightarrow ABC \\ AC \rightarrow AC \\ BC \rightarrow BC \end{array}$$

הכל טריויאלי או מפתח על, ולכן ABC הוא ב- $BCNF$. אך גם עבור ABD אם נבדוק. ולכן הפירוק הוא ABD .

דוגמא 2

$$\begin{array}{l} R = (A, B, C), \\ F = \{AB \rightarrow C, C \rightarrow B\} \end{array}$$

R אינו ב- $BCNF$ כי התלות $B \rightarrow C$ אינה טריויאלית וצד שמאל אינו מפתח על. נפרק לפיה.

$$\begin{array}{c} AB \quad C \\ C \rightarrow B \\ CB \end{array} \quad C \cup (AB - CB) = CA$$

הערה: כל פירוק שננסה לבנות לא יוכל לשמור את $A \rightarrow C$.

Week 11: Framework (Transaction Management)

TM1: Intro 11.1

[מה יכול להשתרבש בשמריםים תוכנה מול מערכת ה-DB?](#)

Partial Execution:
Failure in the middle of
the program

**Bug in the
Program:**
Introduces Errors

**Concurrently Run
Programs
Interfere:**
Unexpected Results

System Failure:
Changes are lost

1. ביצוע חלק – התוכנית מפסיקת באמצע ורצה באופן חלק. זה יכול להשאיר נתונים במצב לא הגיוני.

2. באג בתוכנית – אפשר להכניס שגיאות למערכת ה-DB בצורה כזו.

3. ריצות מקבילות – במות גודלה של ריצות מקבילות עלולות להפריע אחת לשניה.

4. נפילת מערכת – כל השינויים לא נשמרו.

אנחנו נתמקד ב-1-3 באמצעות המונחים הטרנסקציית שיבטיה שלא יתקיים ביצוע חלק, ותשוכנות שרצות במקביל לא יפריעו זו לזו. שתי הביעות האחריות נובעות מהמתכנת, ולגבי התאוששות מנפילה נדבר בשלב מאוחר יותר בקורס. מבחינת Consistency, זה באחריות המתכנת לדאג, במצב זה מערכת ה-DB מבטיח לנו ACID:

Atomicity – כל תוכניות מתבצעת בצורה אוטומטית, או שכל הפעולות מתבצעות או שבלום, לא יהיה חישוב חלק. מתקובל שירות דרך דרך על הטרנסקציה. Consistency – לא יהיו באגים בתוכנית (באחריות המתכנת). Isolation – תוכנית אחת לא תראה נתונים של תוכנית אחרת. Durability – גם במקרה של נפילת מערכת, שינויים במערכת ישמרו.

[אילו בעיות יכולות להתעורר בעקבות חישוב חלק של תוכנית או ריצה מקבילה של תוכניות?](#)

```
CREATE TABLE Accounts(  
    pid int, amount int check (amount>=0));  
  
INSERT INTO Accounts VALUES(1,200),(2,50);
```

טבלה שמכילה מידע על חשבון בנק. לכל שורה יש מספר חשבון amount וחייב להיות אי-שלילי. כניסה שנרצה להעביר 100 ש' מחשבון 2 לחשבון 1. זה לא אפשרי, כי אין מספיק כסף בחשבון 2. אנחנו זוקקים ל-2 פעולות, הוספה בסך לחשבון 1 והורדת בסך מחשבון 2. אם אנחנו נבצע:

```
UPDATE Accounts  
SET amount = amount + 100  
WHERE pid = 1;
```

```
UPDATE Accounts  
SET amount = amount - 100  
WHERE pid = 2;
```

על הפעולה הראשונה קיבל הודעה שהיא המבצעה בהצלחה, אבל על השנייה מערכת ה-DB לא תתן לנו לעשות זאת. אם נסיים את ביצוע הפעולות כאן זה המצב:

```
SELECT * FROM Accounts;  
  
pid | amount  
-----+-----  
2   |    50 ←  
1   |   300
```

הבעיה היא שהסתבלנו על כל אחת מהפעולות כאשר תלויה בשנייה, אבל הן תלויות. הינו רוצים שאו שהן יתבצעו שתיהן, או אף אחת מהן.

טרנסקציית

בשביל להגיד למערכת DB שיש לנו אוסף פעולות שלוגית חייבות לקרות יחד, משתמש במנגנון טרנסקציה. המנגנון זה אומר למערכת שיש יחידות עבודה שצרכות להתבצע כולם יחד, או לא להתבצע כלל כדי שהמערכת תדע מה לעשות בעת שגיאה. המנגנון הוא דרך להגדיר שיש אוסף פעולות שבוצעות יחד ו מבחינת תוכנית הביצוע הם אמורים להתבצע יחד לבד בעולם DB בלבד קשור לפעולות אחרות.

```
BEGIN TRANSACTION;  
...UPDATES/INSERTS/DELETES/SELECTS  
COMMIT;
```

```
BEGIN TRANSACTION;  
...UPDATES/INSERTS/DELETES/SELECTS  
ROLLBACK;
```

כרגע עשויים טרנסקציה, commit נכתב אחרי רצף הפעולות כדי להגיד למערכת שהפעולות מבחןתו הסתיימו והדברים צריכים להיות שמורים במערכת DB. או, שנעשה Rollback אם גילינו שגיאה לא תקין (ערך לא צפוי, או מחיקנו שורות אבל לא כמו שרצינו). אם מערכת DB רצתה ב透ction טרנסקציה ובאמצע קיבלה exception בלשונו, המערכת לבד תבצע Rollback.

אפשר גם לבצע COMMIT באמצעות COMMIT, END TRANSACTION, הדומים שකולות.

חזרה לדוגמה עם חשבונות הבנק

איך הינו אמורים לבתוב את זה?

```
BEGIN TRANSACTION;  
UPDATE Accounts  
SET amount = amount + 100  
WHERE pid = 1;  
  
UPDATE Accounts  
SET amount = amount - 100  
WHERE pid = 2;  
END TRANSACTION;
```

במקרה זה Postgres יעדכן אותנו:

```
ROLLBACK  
  
SELECT * FROM Accounts;  
pid | amount  
-----+-----  
1   |    200  
2   |     50
```

בעיית מקבילות

יכולות להיות כמה תוכניות שניגשות למערכת במקביל. בדרך כלל מדברים על 5 סוגים של בעיות מקבילות:

Remember:

- A *dirty read* occurs when a transaction reads data written by a concurrent uncommitted transaction
- A *dirty write* occurs when a transaction writes data written by a concurrent uncommitted transaction
- A *nonrepeatable read* occurs when a transaction rereads data that it has previously read and discovers that it has been modified by another transaction.

Dirty Writes

Dirty Reads

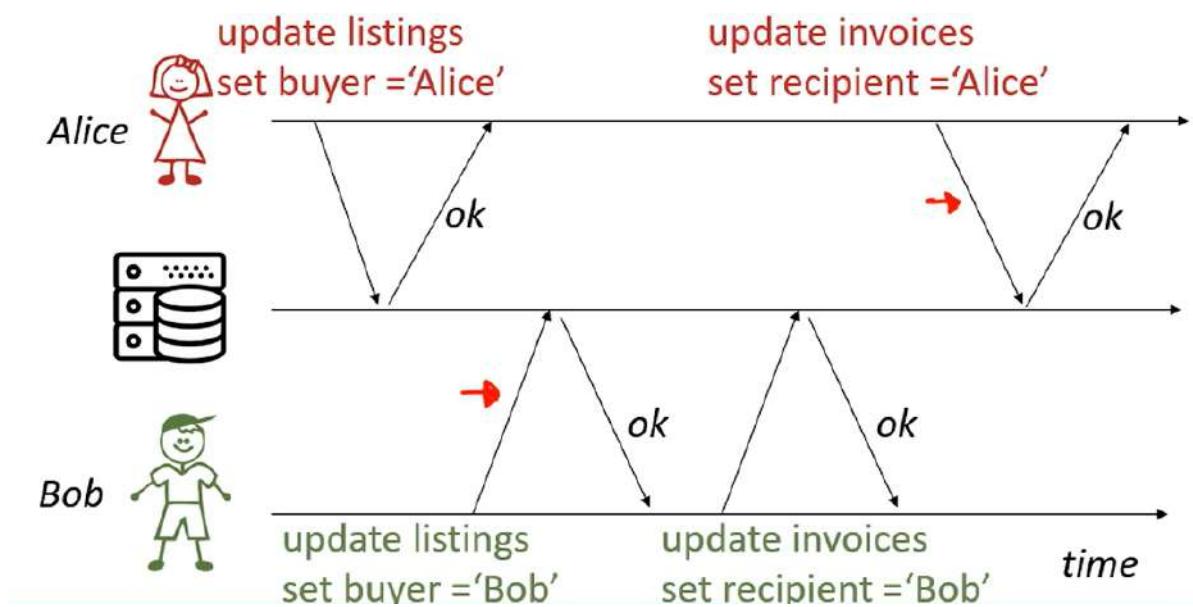
Nonrepeatable Reads

Phantom Reads

Serialization Anomalies

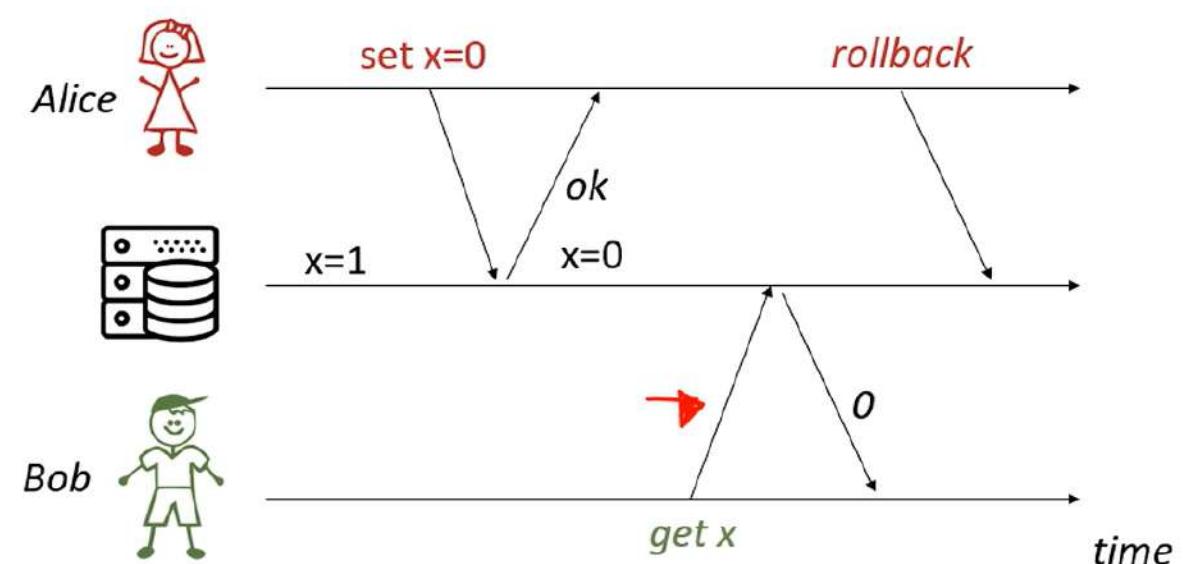
[Dirty Write](#)

ההצטע באשר טרנסקציה בותבת דатаה שנכתב על ידי טרנסקציה אחרת Commit. בעצם, באשר בותבים מעל מידע שטרנסקציה אחרת כתבה אבל לא אישרה.



בחיצים באדום אנחנו רואים Dirty Writes. התרבשו 2 כתיבות כאלו, מה הבעה? בטבלה של מי קונה את הפריט יש ברגע Bob, אבל מי שעומד לשלם על הפריט זו Alice.

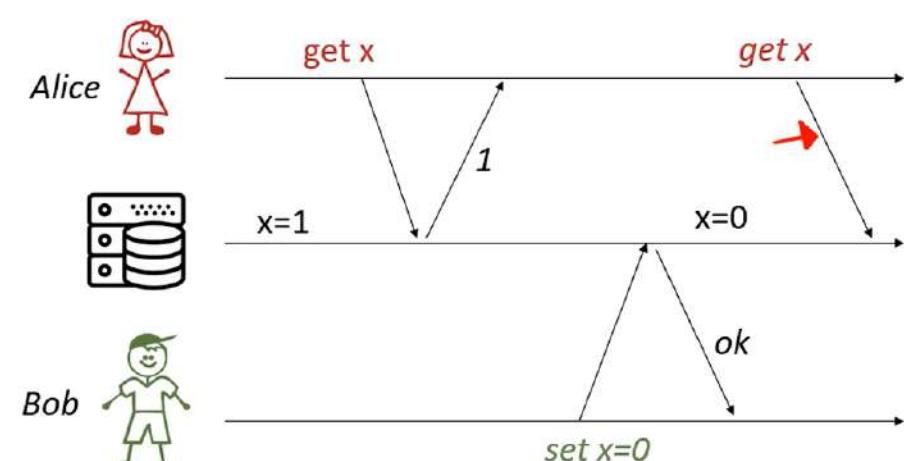
[Dirty Read](#)



כאן בחץ האדום יש dirty read, בו ברגע קרא מידע ממערכת-DB שעוד לא עשה commit וכן קיבל מידע לא נכון.

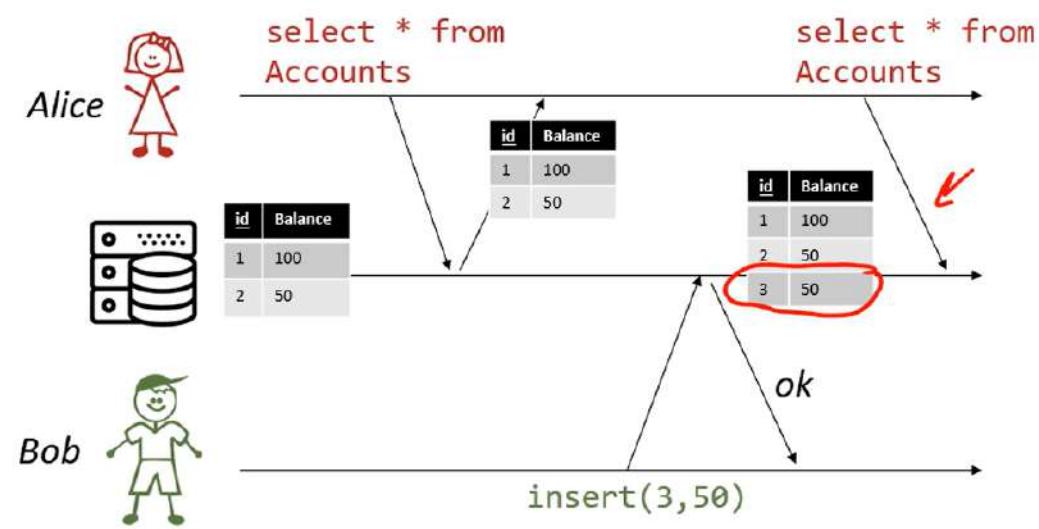
[Nonrepeatable Read](#)

באשר טרנסקציה קוראת ערך שהוא כבר קראה, והוא מגלה שהערך השתנה, ולא על ידה.



[Phantom Read](#)

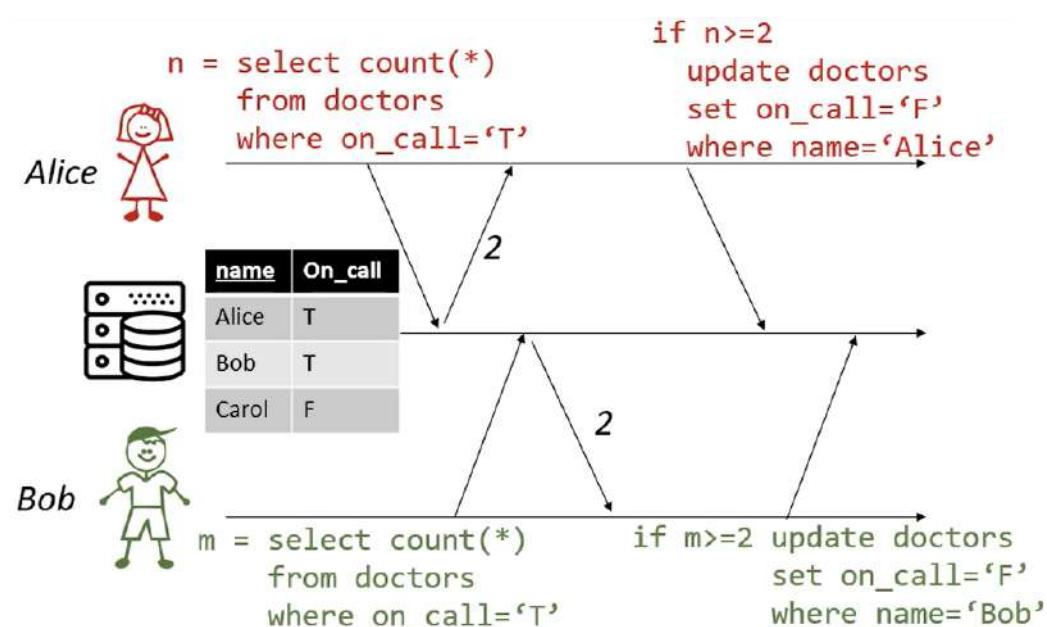
באשר טרנדקציה מבצעת שאלתה, מקבלת קבוצה של ערכים בתשובה אבל בשיהיא מבצעת את השאלתה שוב, היא מקבלת קבוצה שונה של שורות.



כאן בacz האדום סימנו את ה-readphantom. זו בעיה דומה ל-.Nonrepeatable Read

[Serialization Anomaly](#)

פעולה מזורה שקרתית שבה התוצאה שונה מאשר הרצת סדרתית של פעולות. נניח שיש לנו טבלת רופאים ומילוי המשמרת, כאשר יש כל שטميد לפחות רופא אחד במשמרת.



כאן אליס ובוב בדקו במקביל מה מצב הפעלה, ולכן כל אחד מהם הוריד את עצמו ממשמרת. אילו קודם היה מבצעים פעולה ואז הוא בודקים את מצב הפעלה המצביע היה נראה אחרת. קיבלנו תופעה שבה הרצת הפעולות שלהם במקביל יוצרה תופעות שלא היו מקבלים אילו היינו מבצעים אותם בצורה סדרתית.

[TM2: Isolation 11.2](#)

אין מערכת ה-DB גורמת לכך שככל טרנדקציה מרגישה שהיא רצתה בלבד בעולם, למרות שזה לא המצב?

[איך נשיג בידוד במערכת ה-DB?](#)

Actual Serial Execution

Single-Version Concurrency Control

Multi-Version Concurrency Control

- לא להריץ טרנדקציות שונות בו-זמןית. בWOODAI, אם בכלל רגע נתון רק טרנדקציה אחת רצתה יהיה בידיוד מלא. הבעיה, אנחנו נצטרך תמיד לחכות שטרנדקציה אחת תסתיים כדי להתחיל את הבאה בתור, וזה יאט את הביצועים שלנו.
- להשתמש במנגנון לבקרה מקבילים שידאג זה שונרגיש שהטרנדקציות לא יראו אותה את השניה, ועדיין ירצו במקביל. יש לכך שתי אפשרויות: או בעזרת זה שכל ערך עדין ישמר גרסה קודמת שלו, או באמצעות שמירה של הרבה העתקים של אותם פרטיו ידע, וכל טרנדקציה תסתכל על העתק אחר של הפריט ובאיישתו אופן המידע בסוף יעדכן כמו שצריך.

בידוד הוא לא " הכל או כלום", יש ב-SQL 4 רמות בידוד שונות שמאגדירות עד כמה כל טרנסקציה מבוזדת, ואפשר להגדיר כמה חשוב לנו שהוא לא תראה את האפקטים של הטרנסקציות האחרות. ככל שבבודד אותה יותר, המקבילות תפוגן, אבל יוכל לשמור על יתר הפעולות בדאטה ביבס כי נהייה בטוחים שאין השפעות זרות של תוצאות של טרנסקציה אחת על אחרת.

Read Uncommitted

Read Committed

Repeatable Read

Serializable

אלו 4 הרמות שמוגדרות בסטנדרט של SQL. צריך לשים לב שמערכות DB שונות ממשמעות הרמות הללו בצורה שונה. ככל מגדרים את הטרנסקציה להיות **Read Committed**, אבל מה המשמעות של הדבר, מה בדוק המבודד, משתנה ממערכת למערכת.

Isolation Level in Postgres

Level	Dirty Write	Dirty Read	Unrepeatable Read	Phantom	Serialization Anomaly
READ UNCOMMITTED	No	Possible, not in Postgres	Possible	Possible	Possible
READ COMMITTED	No	No	Possible	Possible	Possible
REPEATABLE READ	No	No	No	Possible, not in Postgres	Possible
SERIALIZABLE	No	No	No	No	No

נראה למשל שב-**Read Committed** לא יהיה dirty write, אבל ברמת העקרון המערכת כן תאפשר dirty read – בפוסטgres גם זה לא יתאפשר. בדרגת הבידוד הגבוהה ביותר – **Serializable**, הכל לא מאושר – רמת בידוד מלאה. ה-default הוא שאם אנחנו בותבים טרנסקציה ב-, SQLServer, PostgreSQL, Oracle אנחנו רצים ברמת בידוד **Read Committed** ולא של **Serializable**. MySQL הדיפולט הוא **Repeatable Read**.

במסגרת הקורס אנחנו נלמד רק איך מערכת DB דואגת לרמת בידוד של **Serializable** (כל האחרים הם וריאציות על הפרוטוקולים הללו, מעבר לסקופ שלו).

הגדרת רמת בידוד של טרנסקציית

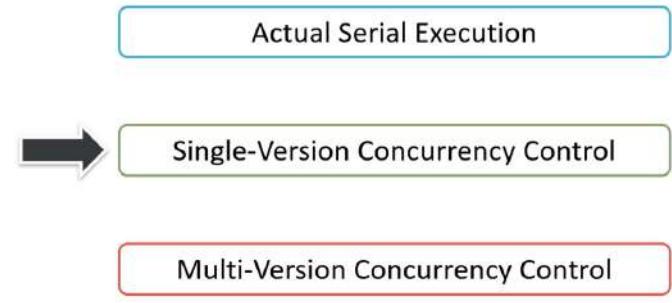
```
BEGIN TRANSACTION
ISOLATION LEVEL ....
queries, updates, ...
COMMIT | ROLLBACK | END TRANSACTION
```

```
SET TRANSACTION
ISOLATION LEVEL { SERIALIZABLE |
REPEATABLE READ |
READ COMMITTED |
READ UNCOMMITTED }
READ WRITE | READ ONLY
```

נכתב ברגיל **BEGIN TRANSACTION** ו**SET TRANSACTION ISOLATION LEVEL**. אפשר גם לעשות **Isolation level**. אפשר גם לציין את רמת הבידוד לכל הטרנסקציות שיבאו אחריו.

איך המערכת מושגה בידוד ברמת Serializable?

נתחיל בلنסוט להבין את זה בעזרת Single Version Concurrency Control



כלומר, נניח שיש העתק אחד בלבד בכל נתון, והמערכת מדאג לכך שהטרםזקציות השונות לא מפריעות אחת לשניה. על מנת לעשות זאת, אנחנו חיבבים אוסף של הגדרות תיאורתיות שמתארות את מה שנרצה להשיג, ורק אחר כך מוכן להבין איך להשיג את זה.

TM3: Schedules 11.3

נתחיל בהגדרות פורמליות:

טרםזקציה

A **transaction** is any one execution of a user program in a database management system

As a convenient **abstraction**, we view any transaction as a sequence of **reads** and **writes** of database **objects**

Can be a data value, a tuple, a page, a table...

טרםזקציה – ביצוע אחד של תוכנית של משתמש במערכת ה-DB. באבストראקציה, נסתכל עליה באוסף פעולות של קריאה וכתיבה של מערכת ה-DB. נניח שאנחנו קוראים וכותבים הם אובייקטים (ערך בשורה, שורה, דף, טבלה..).

דוגמאות

טרםזקציה T תירשם כך:

T: W(A) R(B) R(C) W(B), Commit

בותבת את האובייקט A (ערך לטור A), לקרוא את B, לקרוא את C, לקרוא את B ובסוף commit.

הנחות מפשטות

Transactions interact with each other **only** via reads and writes

A database is a **fixed** collection of **independent** objects

There is a **single copy** of each object, which all transactions read and write

1. נניח שטרםזקציות לא מדברות אחת עם השנייה, אלא רק לראות את האפקטים של הכתיבה אחת של השניה.
2. נניח שה-DB הוא קבועה קבועה של אובייקטים בלתי תלויים אחד בשני (זו הנחה גדולה, כי בפועל זה לא המצב).
3. נניח ברגע שיש רק העתק אחד של כל אובייקט במערכת ה-DB. כל פעם שטרםזקציה קוראת אובייקט היא רואה את הערך האחרון שנכתב בתוך האובייקט. אפילו אם טרםזקציה לא עשתה commit, הערך שהוא כתבה זה הערך שטרםזקציה אחרת תקרה.

A **schedule** of a set of transactions T_1, \dots, T_n is an ordering of the actions in all of T_1, \dots, T_n that is consistent with each transaction

If all transactions in the schedule have commit or abort actions at the end, the schedule is **complete**

If the transactions are not interleaved, the schedule is **serial**

תזמן – בזמןן של קבוצה של טרנסקציות T_1, \dots, T_n נסדר את הפעולות של הקבוצה בצורה קונסיסטנטית עם כל טרנסקציה.

זמןן מלא – אם כל הטרנסקציות בקבוצה עושים בסופו של דבר commit או abort הזמןן הוא **זמןן מלא**.

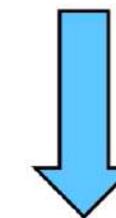
זמןן סדרתי – אם כל טרנסקציה מתחילה ומסיימת בלי הפרעה של אחרת, הזמןן יקרא **זמןן סדרתי**.

דוגמה

T1: R(A) W(A)
T2: R(B) W(B)

את הזמןן נכתב הרבה פעמים בטבלה, באשר הזמןן הוא מלמעלה למטה:

T_1	T_2
R(A)	
	R(B)
	W(B)
	W(A)



Time goes from top to bottom

ולפעמים בסדרת פעולות ואז הזמןן ינוע משמאלי לימין:

R₁(A) R₂(B) W₂(B) W₁(A)



Time goes from left to right

לשם לב לנת אינדקס 2/1 שואמר לאיזה טרנסקציה הפעולה קשורה.

דוגמה לזמןן מלא

T_1	T_2
R(A)	
	R(B)
	W(B)
	Commit
	W(A)
	Abort

T1: R(A) W(A) Abort
T2: R(B) W(B) Commit

כל טרנסקציה עשו בסוף commit/abort. למה השילוב ביניהם נכון? אין סיבה לכך שהוא לא קורא או כותב לאובייקטים שקשורים לו- T_1 .

דוגמה לזמןן מלא וסדרתי

T_1	T_2
R(A)	
W(A)	
Abort	
	R(B)
	W(B)
	Commit

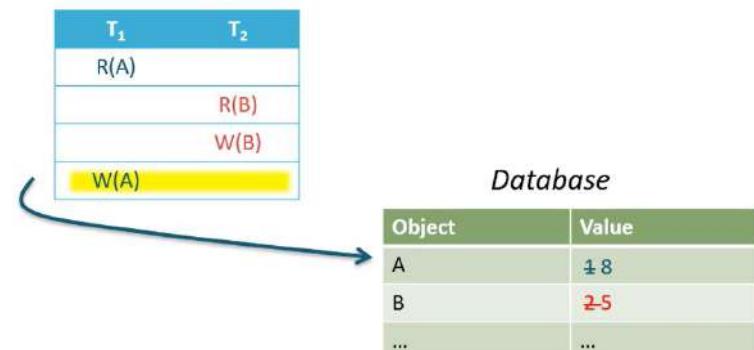
T1: R(A) W(A) Abort
T2: R(B) W(B) Commit

הזמןן מלא כי כל טרנסקציה עשו בסופו של דבר commit/abort והוא סדרתי כי אחת לא מתערבת לשניה במהלך סדר הפעולות. זה נכון נכון, כי במקרה יודעים שהן לא מפריעות אחת לשניה.

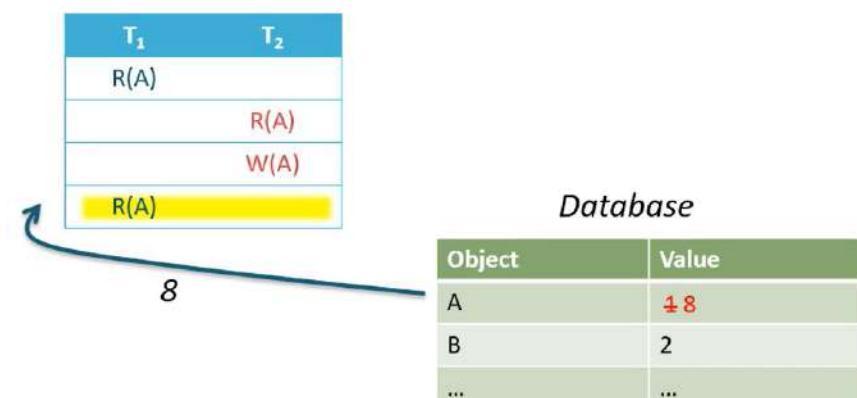
מצד אחד היינו רוצים תזמון **משולב** כדי שיהיה מהיר, מצד שני אנחנו רוצים תזמון **סדרתי** כדי שהוא יהיה מבודד.

איך נשיג מצב שמאז אחד נאפשר ריצה משולבת ומצד שני נקבל תוצאה במערכת שדומה לריצה סדרתית?

דוגמה לתזמון משולב לפרטיטים

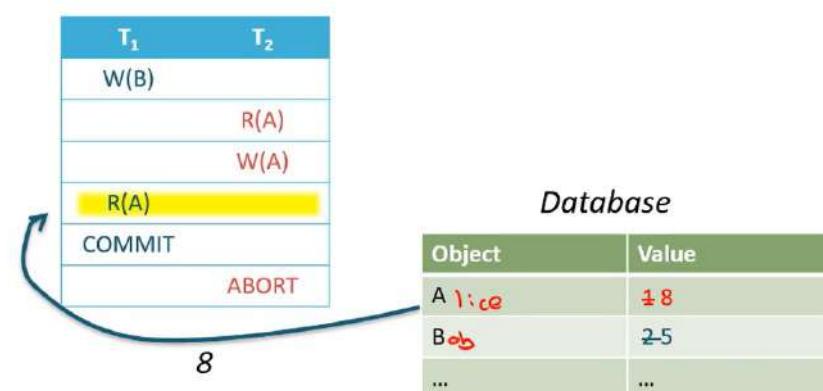


מה קורה אם שתி הטרנסקציות מתעכבות ב-A?



unrepeatable read. T_1 ראה ערך בפעם השנייה שונה מהערך בפעם הראשונה, למרות שלא שינה אותו. זה קרואת של T_1

דוגמה 2



נניח ש- T_1 נועדה להעלות את המשכורת של בוב, אבל רגע לפני האישור היא צריכה לוודא שהמשכורת של Alice גבוהה יותר מאשר בוב. T_1 קוראת את הערך של המשכורת של אליס ורואה שהיא אכן גבוהה יותר ולכן עשויה *dirty read*, אבל אז T_2 עושה *abort*. זה קרא כי הטענה פה *dirty read*, קרנו ערך מלוכל ש- T_2 כתוב ולא עשה לו עדין *commit*.

הנחה הריצה הסריאלית

Schedules are **serial** if the transactions run one after another (with no interleaving)

Assumption: Any **serial** schedule over a database in a **consistent state** leaves the database in a **consistent state**

But, in practice, programs must be run while interleaving to achieve good performance!

דברנו על זה שתזמון הוא סריאלי אם טרנסקציות רצאות אחת אחריה השניה ללא הפרעה. ההנחה שלנו היא שהמשתמש בתוכנית לא באגים, וכך גם נרץ אם נרץ בקורסיה סדרתית מעל מערכת DB וונתחיל במצב שה-DB היה תקין בו, נשאר במצב תקין.

בפועל, ראיינו שאנחנו חייבים להשתמש בשילוב בין פעולות הטרנסקציות כדי להשיג זמן טוב יותר. יש כאן קושי בין ההבטחה על הסריאליות לבין הצורך לשלב טרנסקציות זו בזו.

דוגמא

מה בסדר מבחינת שילוב ומה לא?

T ₁	T ₂
R(A)	
	R(B)
	W(B)
W(A)	

T ₁	T ₂
R(A)	
	W(A)
	R(B)
	W(B)

T ₁	T ₂
	R(B)
	W(B)
R(A)	
	W(A)

Serial Schedule

Serial Schedule

Will all of the above have the same final effect on the database?

Which are guaranteed to yield a desirable effect?

יש לנו 2 טרנסקציות עם 3 תזמים שונים. לפי הנחת הריצה הסדרלית, התזמון האמצעי והימני שכיהם ישאירו את ה-DB במצב קונסיסטנטי. מה לגבי התזמון השמאלי? האם מובטח לנו דבר זהה לגבי התשובה היא כן. מדוע? האפקט הסופי במערכת ה-DB זהה בכל שלושת התזמים. כמובן נראה ב-A את הערך ש-T₁ כתוב, וב-B את הערך ש-T₂ כתוב. לכן, למחרות שהוא לא סדרתי, הוא שකול לתזמון סדרתי.

דוגמא 2

T ₁	T ₂
R(A)	
	R(A)
	W(A)
W(A)	

T ₁	T ₂
R(A)	
	W(A)
	R(A)
	W(A)

T ₁	T ₂
R(A)	
	W(A)
R(A)	
	W(A)

הפעם T₁, T₂ שתיהן עם אותה טרנסקציה. הירוק והאדום תזמים סדרתיים, ולפי הנחת הריצה הסדרתית משאים את מערכת ה-DB במצב תקין. נשים לב שהם משאים את המערכת במצבים שונים. בירוק בשאר עם מה ש-T₂ כתוב, באדום עם מה ש-T₁ כתוב. שכיהם מבחיננתם בסדר. מה קורה עם התזמון הכלול? הוא לא מבטיח לנו שיוכל להיות שקל לסדרתי. שתי הטרנסקציות קוראות את A, וכותבות ל-A. בסופו של דבר נראה מה ש-T₁, בדומה לתזמון האדום. אבל, בתזמון האדום קרא את הערך ש-T₂ כתוב ואחר כך כתוב ערך משלו, בתזמון הכלול הוא לא קרא את הערך הזה.

נשים לב שם שתי הטרנסקציות רוחות להעלות את הערך של A ב-1, נקבל בכך תוצאה שונה בטרנסקציה הכלול:

T ₁	T ₂
0 R(A)	0 R(A)
1 R(A)	0 1 → W(A)
	1 W(A)
W(A)	



T ₁	T ₂
0 R(A)	0 R(A)
1 R(A)	0 1 → W(A)
	1 R(A)
W(A)	W(A)



T ₁	T ₂
0 R(A)	0 R(A)
1 R(A)	0 1 → W(A)
	1 R(A)
W(A)	W(A)



TM4: Serializable 11.4

תזמון בר סידור (Serializable)

Any **serial schedule** of transactions leaves the database in a **consistent state**

A schedule of committed transactions is **Serializable** if its effect on the database is **guaranteed** to be identical to the effect of some serial schedule

כל תזמון סדרתי של טרנסקציות משאיר את המסלד במצב קונסיסטנטי. אנחנו נגיד שתזמון של טרנסקציות שעשוות commit הוא ברת סידור אם האפקט שלו על ה-DB מובטח להיות זהה לאפקט של תזמון סדרתי. התזמון שלנו לאו דויק סדרתי יהיה בר סידור אם התוצאה של הערכיהם שבסופו של דבר כתובים ב-DB היא זהה לריצה סדרתית. זה מה שモבוטח על ידי רמת הבידוד Serializable.

דוגמה 1

האם התזמון הבא בר סידור? שאלת שcolaה, האם האפקט שלו על ה-DB היא כמו להרץ אותו בצורה סדרית?

T ₁	T ₂
R(A)	
	R(B)
	W(B)
	W(A)
	Commit
	Commit

התשובה היא כן. אם נרץ קודם את T₁ במלואו ואז את T₂ במלואו, או הפוך, נקבל את אותו הדבר בבדיקה כמו בתזמון הכהול בטבלה.

דוגמה 2

האם התזמון בר סידור?

T ₁	T ₂
R(A)	
	R(B)
	W(B)
	R(B)
	W(A)
	W(B)
	Commit
	Commit

התשובה היא כן. האפקט הסופי על ה-DB יהיה כמו להרץ קודם T₁ ואז את T₂. בזמן הכהול, T₁ קורא את הערך B ש-T₂ כתוב.

דוגמה 3

T ₁	T ₂
R(A)	
	R(A)
	W(A)
	W(A)
	Commit
	Commit

זהו לא תזמון סדרתי. האפקט של ה-DB הוא לא כמו להרץ את T₁ ואז את T₂ וגם לא כמו מאז את T₂ ואז את T₁.

דוגמה 4

T ₁	T ₂	T ₃
R(A)		
	R(A)	
		W(A)
		W(A)
		Commit
		Commit
		W(A) ← Blind write
		Commit

מה לגבי התזמון זהה? דומה לתזמון הקודם אבל הוספנו טרנדזקציה שלישית. זה כן בר סידור. למה? נשים לב שכן סידור T₁T₂T₃ וסידור T₂T₁T₃ הוא בעל אותו אפקט על מערכת ה-DB. זו כתיבה עיוורת על A, כי הוא לא קורא את הערך לפני, ולכן זה לא משנה מה T₂-T₁ עושים.

מה יכול לגרום לתזמון להיות לא בר סידור?

Transactions that read/write **disjoint sets** of objects do not pose a problem

אם הטרנסקציות קוראות וכותבות קבוצות זרות של אובייקטים, אז אין בעיה.

Transactions that **only read** objects do not pose a problem

אם יש לנו טרנסקציה ש רק קוראת אובייקטים ולא כותבת אף פעם, זה לא משנה את ה-DB ולפנין אין בעיה.

Problems can occur when one transaction **writes** an object A and another transaction **reads or writes** A

מתי עליה בעיה? בשטרנסקציה אחת כותבת אובייקט A וטרנסקציה אחרת קוראת את A או דורסת אותו ואז יכולה להיות תוצאה שונה מרצויה. הבעיה האלו נקראות **קונפליקטים**.

סוגי קונפליקטים

WR Conflict

A **WR conflict** occurs when a transaction T **writes** an object A and T' **reads** A afterwards

T ₁	T ₂
R(A)	
W(A)	R(A)
	W(A)
	R(B)
	W(B)
	R(B)
	W(B)

הkonflikt מתרחש כשה-T כותב אובייקט A, ו-T' קורא את A לאחר מכן. בדוגמה מימין יש 2 קונפליקטים כאלו, הירוק והסגול.

RW Conflict

A **RW conflict** occurs when a transaction T **reads** an object A and T' **writes** A afterwards

T ₁	T ₂
R(A)	
	R(A)
	W(A)
	R(B)
	R(B)
	W(B)

כאשר טרנסקציה T קוראת אובייקט, ו-T' כותבת את A לאחר מכן. כאן בדוגמה מימין שוב יש 2 דוגמאות כאלה.

WW Conflict

A **WW conflict** occurs when a transaction T **writes** an object A and T' **writes** A afterwards

T ₁	T ₂
R(A)	
W(A)	R(A)
	W(A)
	R(B)
	R(B)
	W(B)
	R(B)
	W(B)

מתרחש כאשר טרנסקציה כותבת אובייקט וטרנסקציה אחרת דורשת אותו לאחר מכן. שוב 2 קונפליקטים לדוגמה.

זמןן עם Commit

עד כאן דיברנו על תזמנון עם טרנסקציות שעשוות Commit. הזמןן הוא בר סידור אם הוא שקול מבחינת האפקט שלו על מערכת ה-DB לזמןן סדרתי. אם יכול להיות זמןן שבו חלקם עושים commit וחלקם עושים abort, אז אנחנו אומרים **שהזמןן שלנו הוא בר סידור אם האפקט שלו על מערכת ה-DB זהה ליריצה סדרתית של התזמנונים שעשו commit.**

דוגמאות

T ₁	T ₂
R(A)	R(A)
W(A)	W(A)
R(A)	R(A)
W(A)	W(A)
R(B)	R(B)
W(B)	W(B)
Commit	Commit
Commit	Abort

אם הזמןן האלה בר סידור? הבדיקה בר סידור כי הוא שקול ליריצה של $T_1 T_2$, הזמןן הירוק לעומת זאת לא בר סידור, כי הוא צריך להיות שקול ליריצה של T_2 בלבד, אבל T_2 קודם קורא את השינוי ש- T_1 כתוב ופועל בהתאם (אגב, אם T_2 היה עושה כתיבה עיוורת זה בן היה בר סידור).

TM5: Recoverable 11.5

זמןן בר התואשות

יכולות להיות בעיות חמורות עם טרנסקציות שעשוות Abort.

דוגמאות

T ₁	T ₂
R(A)	T ₂ read a value for A that should never have been there! We cannot undo the effect of this, since T ₂ has committed!
W(A)	Schedule is unrecoverable
R(A)	
W(A)	
R(B)	
W(B)	
Commit	
Abort	

T_1 קוראת וכותבת את A, ואז T_2 קוראת וכותבת את B ועושה Abort. לאחר מכן, T_1 עושה Commit. מתחייבת מעצמה אוטומטית הפעולה שהיא תמחק את כל השינויים ש- T_1 עשתה במערכת. אבל, זה בלתי אפשרי בבר לבצע את זה, כי אם נעשה כך T_2 יהיה אמור לראות ערך אחר ב-A. אנחנו לא מסוגלים לעשותundo כי התבכשו פעולות של הטרנסקציה השנייה בהתקפס הראשונה. זה הזמןן לא בר התואשות. אסור לנו לאפשר תזמנונים כאלה.

הגדרה

נגיד שהזמןן S הוא בר התואשות אם כל הטרנסקציות מבצעות commit רק אחרי שכל הטרנסקציות שמסתמכות עליהם עושים commit.

T ₁	T ₂	T ₃
R(A)		
W(A)		
	R(A)	
	W(A)	
	R(A)	
	W(A)	
	W(B)	
	Commit	
	Commit	
	Commit	

בדוגמה הזאת אנחנו רואים ש- T_1 עושה commit ראשוני ולא קרא שינוי של אף טרנסקציה אחרת ולכן עשה commit בזמן תקין. T_2 קרא את הערך ש- T_1 כתוב וכן עשה commit אחריו וזה בסדר. בנ"ל לגבי T_3 . **לכן זה הזמןן בר התואשות.**

באן יכולים להתרחש cascading aborts, למשל אם T_1 היה עושה abort במקום commit, יכולנו להתואש מזה על ידי כך שמערכת ה-DB הייתה עשוות T_2, T_3 Abort.

הגדרה

זמן **נמנע מ-Abort** כאשר טרנסקציות קוראות רק שינויים של טרנסקציות שעשו commit. בשיש 0 dirty reads המתמן נמנע מ-Abort.

T ₁	T ₂	T ₃
R(A)		
	W(A)	
		R(B)
		W(B)
		Commit
		W(B)
		Commit
		R(A)
		W(A)
		Commit

לא קורא שינוי של אף טרנסקציה ולכן הוא בסדר, וכך גם T₂, אבל T₃ קורא את השינוי שנעשה על ידי T₁, אבל לאחר ש-T₁ עשה commit ולפניהם הזמן זהה נמנע מ-Abort. אולי T₁ היה עוזה abort זה לא היה גורם להפלת אחת אחריה השניה של טרנסקציות.

שאלות מרכזיות

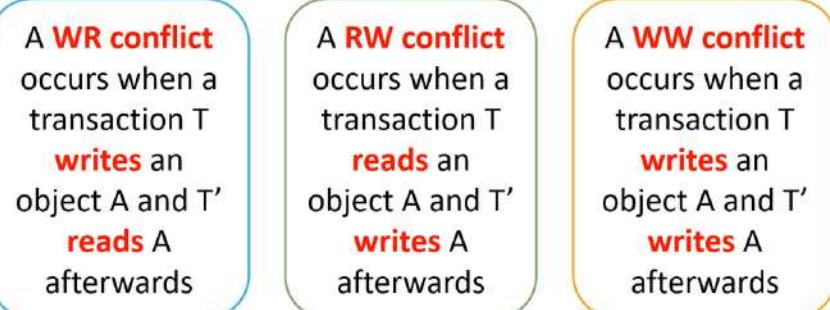
1. איך אנחנו יכולים לזהות בצורה מסודרת האם זמן הוא בר סידור?
2. אנחנו מקבלים בקשות כתיבה וקריאה מהטרנסקציות באופןין, איך מערכת DB יכולה להפעיל פרוטוקול שיגרום לפעולות כתיבת וקריאה ליצור זמן בר סידור, בר התואששות, נמנע מ-Abort?

TM6: Conflict Serializable 11.6

זמן בר סידור

אנחנו רוצים להיות מעוניינים לזמן האם זמן הוא בר סידור, כלומר מהירותה של DB הוא זהה לאם הינו מרים את הטרנסקציות אחת אחריה השניה. לאחר מכן נוכל לעבור לשאלת האם DB יכולה לדאוג שהזמן בפועל יהיה בר סידור.

תזכורת – קונפליקט



קונפליקט קורה כאשר טרנסקציה אחת קוראת/כותבת ערך (אובייקט) וטרנסקציה אחרת קוראת/כותבת את אותו ערך, ולפחות באחד מהם הייתה פעולה בתיבה. הקונפליקטים האלה לאו דווקא אמורים שיש בעיה עם הזמן, אבל אמורים משהו לגבי הסדר שבו פעולות צריכות להתבצע על מנת שהזמן יהיה שקול לתזמן סדרתי.

שקלות תזמננו

בשנותינו ל-2 תזמננו שונים אנחנו יכולים לשאול האם הם שקולים מבחינה הקונפליקטים:

Two actions in a schedule are in **conflict** if:

1. They are on the same object
2. They are performed by different transactions
3. At least one is a write

Two schedules over the same transactions are **conflict equivalent** if they order every pair of conflicting actions of committed transactions in the same way

כלומר, אם נסתכל על כל זוג פעולות בזמן הראשוני (על אותו אובייקט, מבוצעות על ידי שתי טרנסקציות שונות ולפחות אחת מהן כוללת בתיבת) האם הזוג הזה נמצא באותו סדר בשני התזמנויות. אם לכל זוג של פעולות בكونפליקט בזמן הראשוני הם נמצאים באותו סדר בזמן השני – נגיד שהזמן שקולי קונפליקט.

בשלב זהה נניח שכלי הטרנסקציות מבצעות את פעולה commit, וכן לפעמים נשמשו שמות אותה בדוגמאות. העיה: אם יש שני תזמנויות שקולי קונפליקטיים, יהיה להם את אותו אפקט סופי על מערכת ה-DB: כל טרנסקציה תראה את אותם ערכים כשהיא תקרה, כתוב את אותם ערכים כשהיא כתוב, וכן האפקט הסופי על המערכת זהה.

דוגמה 1

Two schedules over the same transactions are **conflict equivalent** if they order every pair of conflicting actions of committed transactions in the same way

T ₁	T ₂	T ₃
R(A)		
	W(A)	
		W(A)
		W(A)
Commit		
	Commit	
		Commit

T ₁	T ₂	T ₃
R(A)		
	W(A)	
		W(A)
		W(A)
Commit		
	Commit	
		Commit

כאן יש שני תזמנויות שונים של אותן הטרנסקציות. T₁ בשני התזמנויות עשו את אותו הדבר. גם T₂, T₃ – רק סדר הפעולות שונה. יש כאן זוג פעולות שנמצאות בكونפליקט:

T ₁	T ₂	T ₃
R(A)		
	W(A)	
		W(A)
		W(A)
Commit		
	Commit	
		Commit

T ₁	T ₂	T ₃
R(A)		
	W(A)	
		W(A)
		W(A)
Commit		
	Commit	
		Commit

Not conflict equivalent!

הזמן אינו שקולי קונפליקט מסיבה זו.

דוגמה 2

T ₁	T ₂	T ₃
R(A)		
	R(B)	
		W(A)
		R(A)
	W(A)	
		R(B)
		W(B)
Commit		
	Commit	
		Commit

T ₁	T ₂	T ₃
	R(B)	
R(A)		
		W(A)
		R(A)
		R(B)
		W(B)
	W(A)	
Commit		
	Commit	
		Commit

נעביר זוג זוג על הפעולות ונבדוק האם הם נמצאים באותו סדר בשני התזמנויות. לאחר מעבר ראיינו שכלי זוג קונפליקטיים באותו סדר. לכן הם שקולי קונפליקט.

בר סידור קונפליקט – Conflict Serializable

אנו נגיד שהזמן S הוא בר סידור קונפליקטיים אם הוא שקול קונפליקטיים לזמן סידורי בלבד.

Schedule S is **conflict serializable** if it is conflict equivalent to some **serial schedule**

מכיוון שאמרנו שזמןונים שקולי קונפליקטיבים יש להם את אותו אפקט על ה-DB בעצם אם יש לנו תזמון בר סידור קונפליקטיבים אז הוא גם בר סידור במובן שדברנו עליו קודם – שהאפקט שלו על ה-DB יהיה כמו של הריצה סדרתית של הטרנסקציות.

Conflict equivalent schedules will have the **same effect** on the database



Conflict serializable schedules are **serializable!**

דוגמא

T ₁	T ₂
R(A)	
	W(B)
W(A)	
Commit	
	Commit

T ₁	T ₂
R(A)	
	W(A)
W(A)	
Commit	
	Commit

T ₁	T ₂
R(A)	
	R(A)
W(A)	
Commit	
	Commit

Which of these are conflict serializable?

בכלול – אין קונפליקטיבים בכלל בתזמון, ולכן אם מרצים תזמון סדרתי במשהו היינו מקבלים זמןונים **שקולי קונפליקטיבים**.

בירוק – יש שני קונפליקטיבים:

T ₁	T ₂
R(A)	
	W(A)
W(A)	
Commit	
	Commit

אם נסתכל על תזמון סדרתי, למשל $T_1 T_2$ – זה לא שקול קונפליקטיבים לשידור שאנחנו רואים עכשווי. אם נשווה ל- $T_2 T_1$ זה שוב לא שקול קונפליקטיבים, כי סידרנו בסדר שונה את הקונפליקט בין (A)R, (A)W בסדר שונה. **אין תזמון סדרתי שקול קונפליקטיבים.**

באדום – יש רק קונפליקט:

T ₁	T ₂
R(A)	
	R(A)
W(A)	
Commit	
	Commit

אם נסתכל על התזמון $T_2 T_1$ התזמון הסדרתי שקול קונפליקטיבים לתזמון האדום, ולכן הוא **בר סידור קונפליקטיבים**.

בדיקה האם תזמון הוא בר סידור קונפליקטיבים

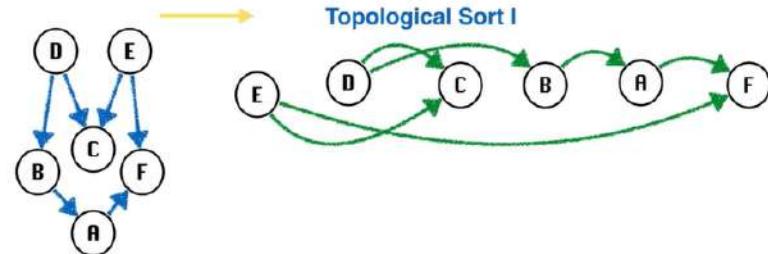
To check if S is conflict serializable:

1. Create a **precedence** graph G with a node for each transaction
2. Add an edge from T_i to T_j if there are conflicting actions in S involving T_i and T_j , in which T_i is the first
3. S is **conflict serializable** if there is no directed cycle in G and is conflict equivalent to any topological ordering of G

אם נתון תזמון S ואנחנו רוצים לבדוק האם הוא בר סידור קונפליקטיבים, אנחנו מייצרים גרף קדימיות לתזמון. כאן יש קודקוד לכל אחת מהטרנסקציות. נסיף צלע מכוננת מ- T_i ל- T_j אם יש זוג פעולות קונפליקטיבים בתזמון שמערבים את T_i ואת T_j והוא הראשון מביניהם הזוג שמתבצע בתזמון S. אחרי שייצרנו את הגרף יוכל להבין ש-S הוא בר סידור קונפליקטיבים אם אין מעגל בגרף הקדימיות שיצרנו. הוא יהיה שקול קונפליקטיבים לכל סדר טופולוגי שנבחר.

A **topological** ordering of a graph $G = (V, E)$ is an ordering of V such that if there is an edge (v_i, v_j) in E , then v_i comes before v_j in the ordering

Topological Sort



בשיש לנו גרף, סידור טופולוגי של קודקודיו הוא סידור שבו אם יש לנו צלע מ- v_i ל- v_j אז v_i בא לפני v_j בסידור.

דוגמת הרצה 1

	T ₁	T ₂	T ₃
	R(A)		
		R(B)	
			W(A)
			R(A)
		W(A)	
			R(B)
			W(B)
		Commit	
			Commit
			Commit

ניציר גרף קדימיות עם קודקוד לכל טרנסזציה, וצלע מכונת עברו כל זוג פעולות בקונפליקט מהטרנסזציה שמבצעת את הפעולה הראשונה לו שמבצעת את השניה:

	T ₁	T ₂	T ₃
	R(A)		
		R(B)	
			W(A)
			R(A)
		W(A)	
			R(B)
			W(B)
		Commit	
			Commit
			Commit



T₃

	T ₁	T ₂	T ₃
	R(A)		
		R(B)	
			W(A)
			R(A)
		W(A)	
			R(B)
			W(B)
		Commit	
			Commit
			Commit



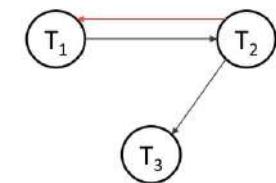
T₃

	T ₁	T ₂	T ₃
	R(A)		
		R(B)	
			W(A)
			R(A)
		W(A)	
			R(B)
			W(B)
		Commit	
			Commit
			Commit

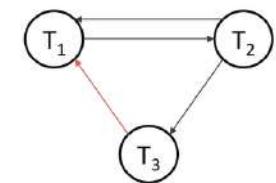


T₃

T ₁	T ₂	T ₃
R(A)		
	R(B)	
	W(A)	
		R(A)
		W(A)
		R(B)
		W(B)
		Commit
		Commit
		Commit



T ₁	T ₂	T ₃
R(A)		
	R(B)	
	W(A)	
		R(A)
		W(A)
		R(B)
		W(B)
		Commit
		Commit
		Commit



אם אנחנו מסתכלים על הגרף שນוצר בבחין שהוא יוצר מעגל, ובעצם אנחנו רואים שהזמן אינו בר סידור קונפליקטיבים.

דוגמה 2

T ₁	T ₂	T ₃
R(A)		
	R(B)	
	W(B)	
		R(A)
		W(A)
		R(B)
		W(B)

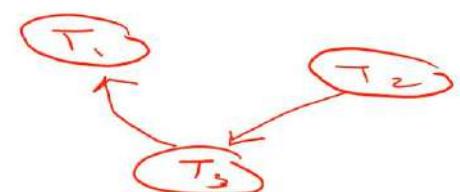
נעביר פעולה אחר פעולה. (A) לא יוצר קונפליקט עם אף פעולה אחרת.

(B) R(B) נמצא בקונפליקט עם (B) של T₃. נסיף צלע בינהם:



W של T₂ נמצא בקונפליקט עם R(B) של T₃ וגם עם W(B) של T₃, אבל בברישתו לנו צלע בזאת.

R(B) של T₃ נמצא בקונפליקט עם W(A) של T₁ אז נסיף צלע:



W של T₁ לא נמצא בקונפליקט עם אף אחד. כך גם (A) R(B) - W(B) של T₃.

סיימנו את הגרף ואין בו מעגל, לכן הוא בר סידור קונפליקטיבים. לאייה תזמון סדרתי הוא שקול קונפליקטיבים? לזמן שמתබן על ידי מין טופולוגי של הגרף, בולם לזמן .T₂T₃T₁

ובשים אנחנו יודעים ליהות האם תזמון הוא בר סידור, או לפחות דרך אחת להזאת זה. אם הוא בר סידור קונפליקטיבים אז הוא בר סידור, אך נדרש לבנות גרף קדימיות. הבעה הקשה היא הבעה השניה שהציגו בתחלת השיעור. טרנסיציות לא מודיעות מראש מה הן צרכות, הכל מתבצע *online*. המערכת צריכה כל הזמן לאשר בלביב האם אפשר לבצע את הפעולות או לא.