



מערכות מסדי נתונים לפתחי תוכנה

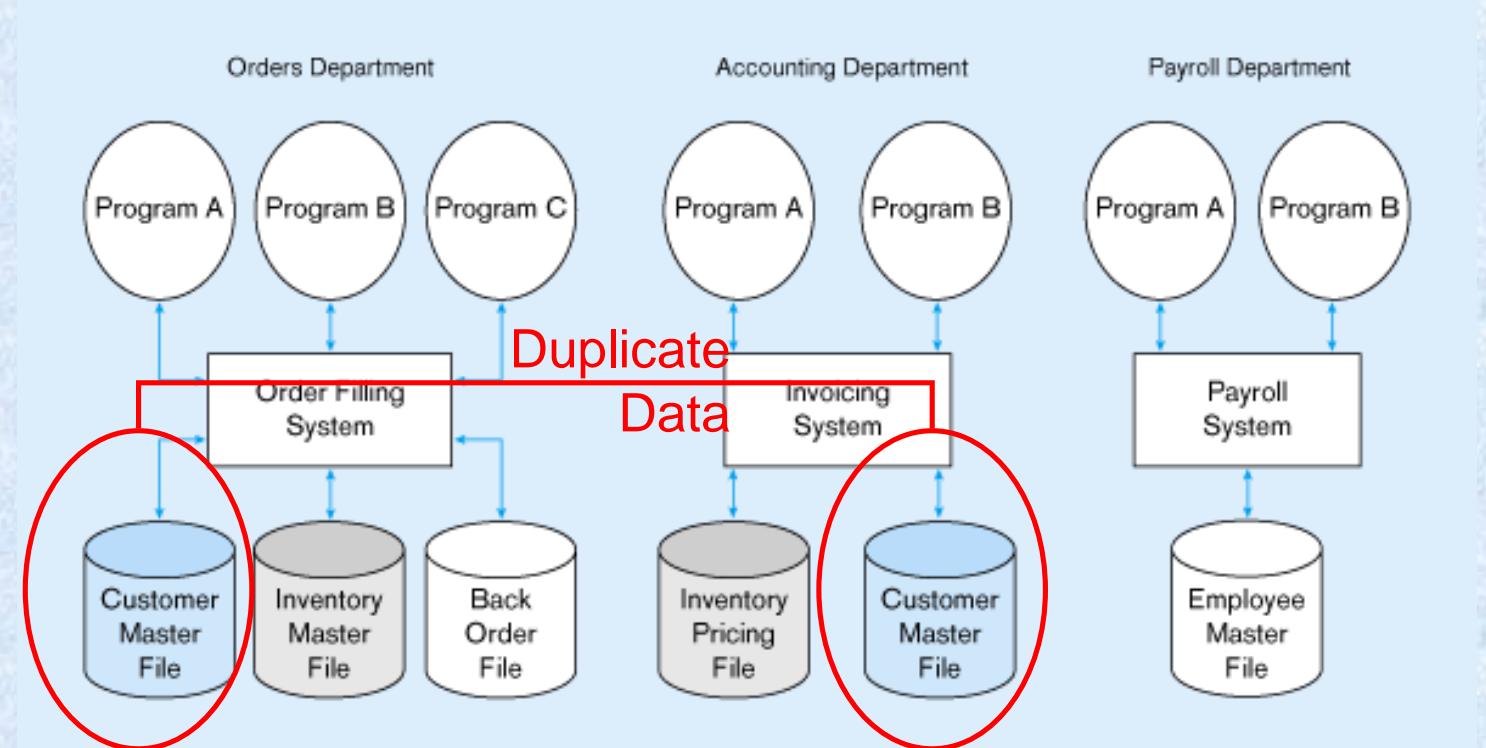
מרצה: שקד לב

מייל: levshakked@gmail.com

תוקן הרצאות מbasics בחלוקת על הרצאות של גב' ערבה צורי

מערכות מבוססות קבצים

- בשנות ה 50 ו ה 60 נתוניים אוכסנו בקבצים
- **מערכת מבוססת קבצים (File-based System)** הינה אוסף של תוכניות יישום המבצעות שירותים עבור משתמש קטן
 - כל תוכנית מגדרה ומנהלת את הנתוניים שלה



חסרונות שימוש במערכות מבוססות קבצים

- **תלות בין תכנית לנדרנים**
 - על כל תכנית לנדרן מטא נתונים על כל הקבצים בהם משתמש
- **כפילות נתונים**
 - מערכות/תכניות שונות מחזיקות עותקים שונים של אותם נתונים – בעיות מקום, עקביות, שלמות
- **שיתוף נתונים מוגבל**
 - אין בקירה מרכזית של הנתונים
- **זמן פיתוח ארוכים יותר**
 - תכניות חיבים לתוכן ולממש את פורמטי הקבצים שלהם
- **אחזקה יקרה של התוכניות**
 - 80% מתקציב מערכת המידע

מהו בסיס נתונים?

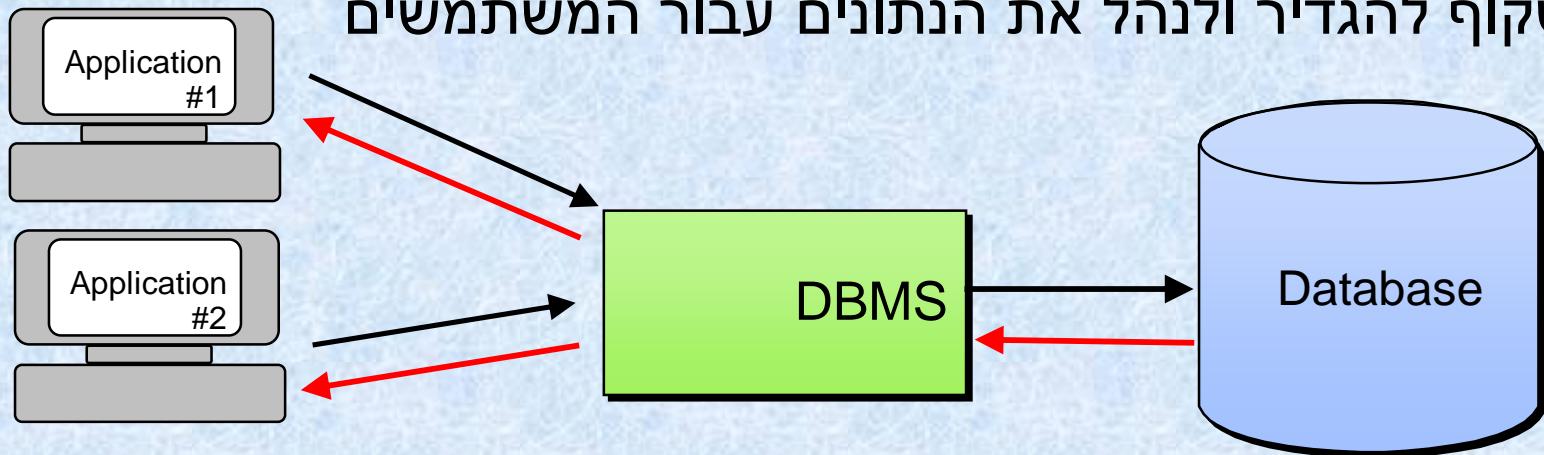
- **בסיס נתונים (Data Base)** הוא אוסף של נתונים הקשורים ביניהם
 - היחidot הבסיסיות המרכיבות נתונים מוכנות רכיבי נתונים או שדות נתונים (Data Field)
 - לדוגמה: ת"ז, שם, תאריך לידה ותובתם רכיבי נתונים של אנשים
- **מערכת ניהול בסיס נתונים (Data Base Management System)** היא אוסף של תכניות המאפשרות ליצור ולנהל בסיס נתונים:
 - הגדרת הנתונים: הצהרה והגדרה של סוגים נתונים, המבנה שלהם, הקשרים ביניהם ואילוצים
 - הקמת הנתונים: שמירה ראשונית של נתונים
 - פעולה הנתונים: שליה, עדכון, הוספה ומחיקה של נתונים
 - ניהול הנתונים: הרשות, גיבויים, ועוד'

גישה מסדי הנתונים

- **מסד נתונים (Database)** הינו אוסף מאורגן של נתונים הקשורים לוגית

– נתונים אלו מתוכננים כך שייעמדו בדרישות המידע של הארגון

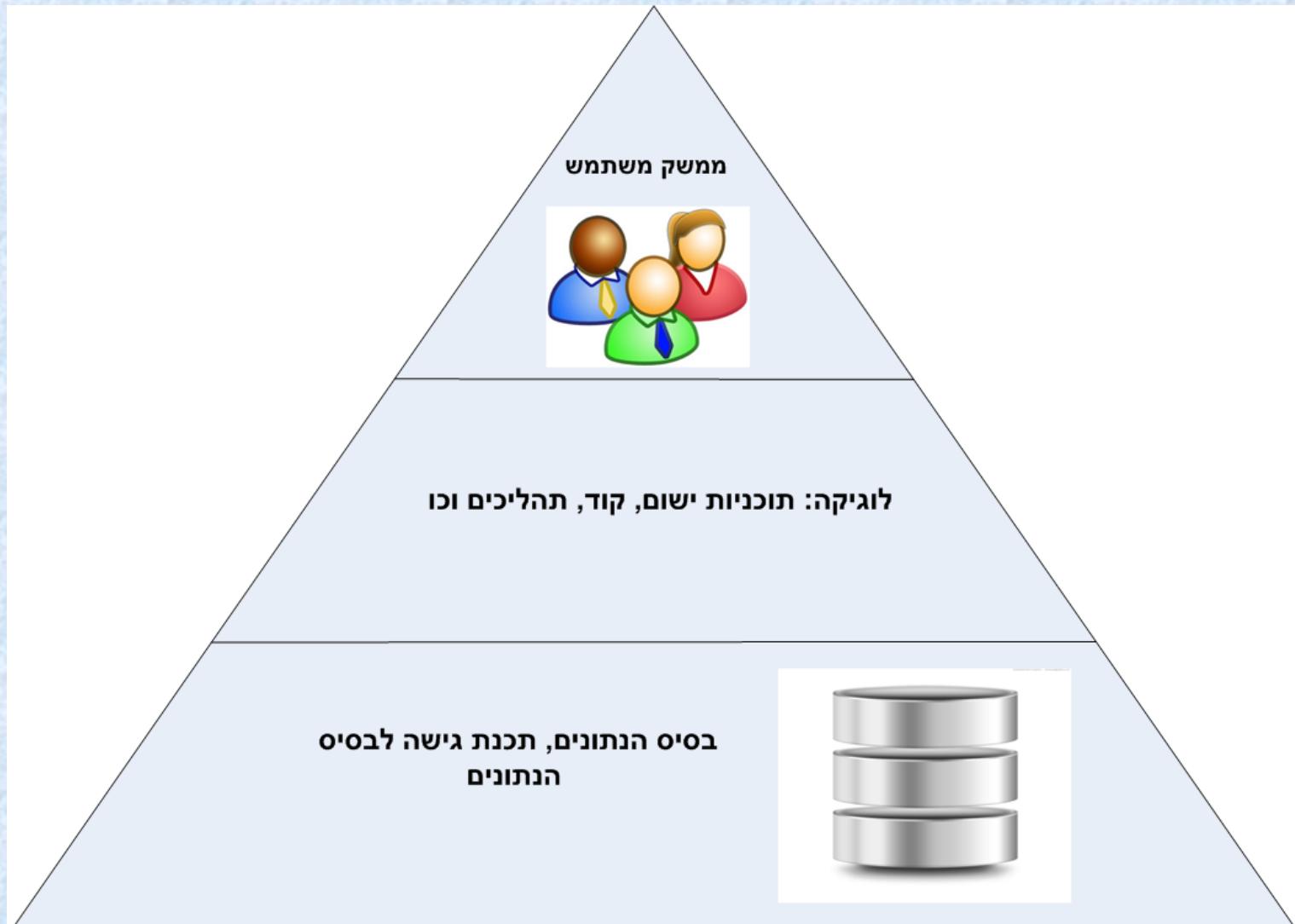
- **מערכת לניהול מסדי נתונים (DBMS)** הינה מערכת תוכנה (אוסף תוכניות) המאפשרת באופן שקוף להגדיר ולנהל את הנתונים עבור המשתמשים



מערכת לניהול בסיסי נתונים מספקת...

...אחסון וגישה של כמויות מאסיביות של נתונים באופן יעיל, אמין, נוח
ובטוח ומתאים לריבוי משתמשים

מקום בסיס הנתונים במערכת המידע



דוגמאות למערכות מידע מבוססות DB

- **מערכת בנקאית**
 - ניהול מידע על לקוחות, חשבונות, הלוואות, תוכניות חסコン וכו'
 - מבצעת פעולות על חשבון וכו'
- **מערכת ניהול מודד להשכלה גבוהה**
 - ניהול מידע אודוות קורסים, סטודנטים, מרצים
 - מבצעת פעולות רישום לקורסים, מעקב אחר ציונים וכו'
- **רשות חברתית**
 - ניהול מידע על משתמשים, פוטטים, תמונות
 - מבצעת פעולות של העלאת תמונה/פוט, תגובה וכו'

דיאגרמת ישות – קשרים (ERD)

- Entity Relationship Diagram
- מודל תפיסתי המאפשר בניית מודל של המיציאות שאוותה רוצים לייצג במערכת המידע באופן גרافي ותור שימוש במספר מושגים קטן
- מטרות:
 - לייצג מה ישמר בבסיס הנתונים (אילו נתונים)
 - להגדיר מהם הקשרים בין הנתונים השונים וכי怎 באים לידי ביטוי אילוצים עשיים

עקרונות מוחים לתרשים ER

- **דיוק – Faithfulness**
 - תיאור המציאות באופן מהימן
- **פשתות – Relevance**
 - התמקדות רק בפרטים הנוגעים למערכת
- **תמציתיות – Redundancy Avoidance**
 - המנעот מיתירות פרטים
 - המנעוט מכפילות נתונים
 - רישום פרטים פעם אחת בלבד (במידת האפשר)

מהו ישות?

- **ישות (Entity)** היא אובייקט ("דבר") שקיים וניתן לאבחנה מאובייקטים אחרים
 - אובייקט יכול להיות מוחשי או לא מוחשי
 - לדוגמה, אדם, חברה, אירוע, מכשיר חשמלי
 - ישות מייצגת כל דבר שעבورو נרצה לשמר מידע בסיסי הנתונים (במערכת המידע)
- **לישות יש תכונות**
 - המאפיינים המגדירים את הישות עליה נרצה לשמר נתונים
 - למשל, עבור עובד נרצה לשמר ת"ז, שם, כתובות, טלפון ושכר

מהי קבוצת ישוות?

- קבוצת ישוות או סוג ישות (Entity set) היא קבוצה של ישוות מאותו הסוג ובעלות אותן התכונות
 - למשל קבוצת כל הלקוחות, קבוצת כל המוצרים, קבוצת כל הזמנות
- ישות (אובייקט) יכולה להשתир לכמה קבוצות של ישוות
 - למשל, אדם יכול להיות מרצה, סטודנט, שניהם
- שם קבוצת הישויות יקבע באופן חד ערכי – לא יתכנו שתי קבוצות ישוות בעלות אותו שם באותו התרשים

מהי קבוצת ישיות? (המשך)

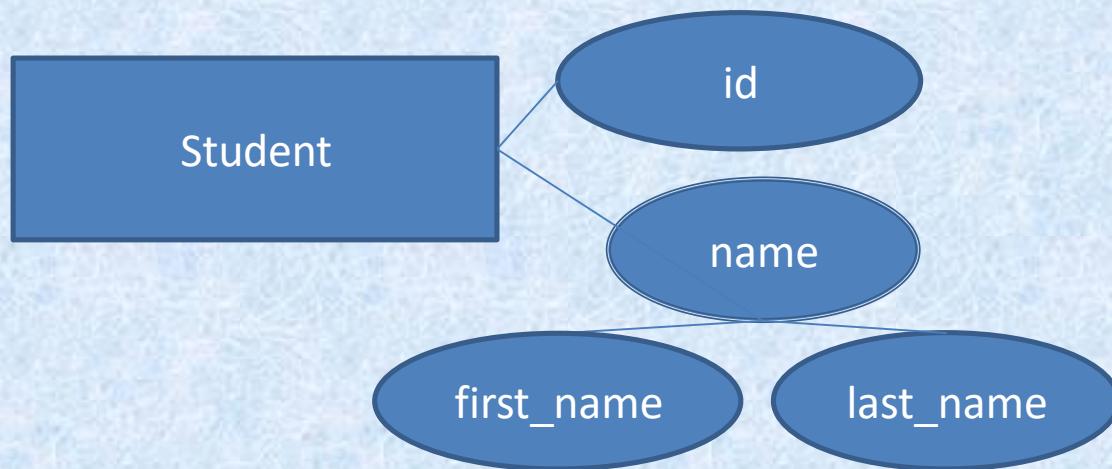
- **קבוצת הסטודנטים**
 - איזה מידע נרצה לשמור עליהם במערכת המידע האוניברסיטאית?
- **פרטים אישיים:** ת"ז, שם, כתובות, טלפונים, כתובות דוא"ל, מגדר, תאריך לידה
- **פרטים אקדמיים:** ממוצע מצטרר, מספר נקודות זכות – הסימן המקובל:

Student

תכונות של יסויות

- **ישות מיוצגת ע"י קבוצה של ערכים שמתארים מאפיינים המשותפים (הנשמרים) לכל החברים בקבוצת היסוד**

– הסימון המקובל:



– **סוגי תכונות:**

- **פשוטה או מורכבת**

- פשוטה – בעלת ערך פשוט שלא ניתן לפירוק. למשל, ת"ז, ציון ממוצע
- מורכבת – בעלת ערך הנitin לפירוק. למשל: כתובה (רחוב, מספר בית, עיר, מיקוד). תאריך לידה (יום, חודש, שנה)

תכונות של ישות (הmarsh)



– סוגי תכונות:

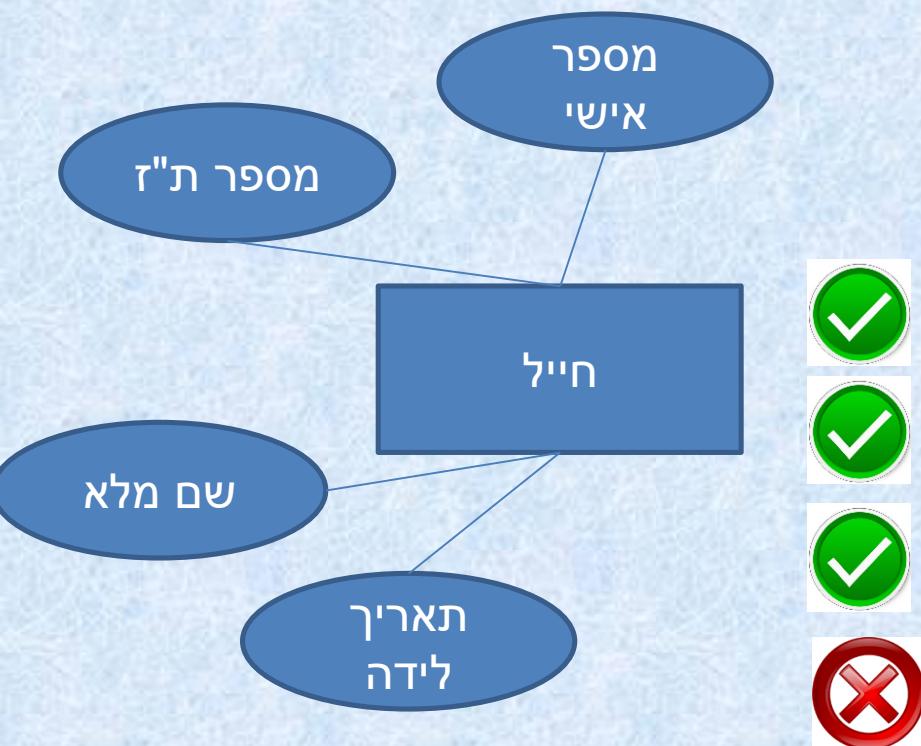
- ערך יחיד או מרובה ערכים
 - ערך יחיד – ערך שלא חוזר על עצמו (באופן רב) עבור ישות, למשל, ת"ז, שם
 - ערך מרובה ערכים – ערך שיכול לחזור על עצמו (ומספר המופיעים בד"כ לא ידוע) עבור ישות, למשל, כתובת (אם מעוניינים לשמר את כל הכתובות הקיימות של ישות), מספר טלפון (אם מעוניינים לשמר את כל מספרי הטלפון הקיימים של ישות)
- בסיסית או נגזרת
 - בסיסית – מתקבלת באופן ישיר מהנתון
 - נגזרת – מתקבלת ע"י פועלות על נתון אחר. למשל גיל, שכר. מסומנת בקו מקווקו

תכונות של י纠וית (המשר)

- סוג את התכונות הבאות:
 - כתובת מיל של ל Koh, תאריך לידה של סטודנט, גיל של עובד
 - פשוטה או מורכבת
 - מרובה ערכאים או ערך יחיד
 - בסיסית או נגזרת

פתרונות של קבוצת ישות

- מפתח על (Super key) היא תכונה/ אוסף של תכונות המבדילות בין הישויות בתוך הקבוצה באופן חד ערכי



- כל התכונות של הישות
- חלק מן התכונות של הישות
- מספר אישי, מספר ת"ז, שם מלא, תאריך לידיה
 - ✓
 - ✓
 - ✓
- מספר אישי, שם מלא, תאריך לידיה
 - ✓
- מספר ת"ז, שם מלא
 - ✓
- שם מלא, תאריך לידיה?
 - ✗

מפתחות של קבוצת ישות (המשר)

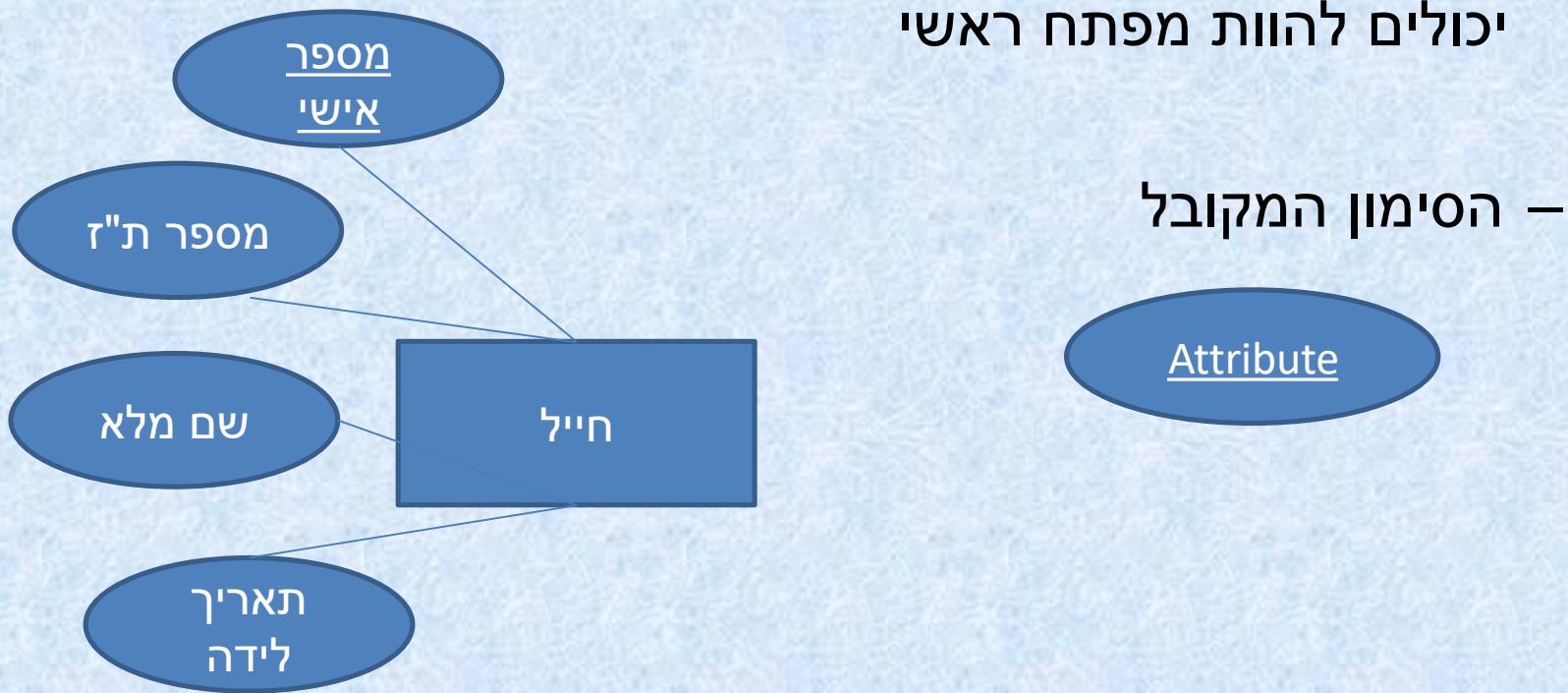
- מפתח קביל (candidate key) היא תכונה/ אוסף של תכונות שהינו מפתח על מינימאלי
 - אינם מכיל תת קבוצה של תכונות שהיא מפתח על עצמה



- מספר אישי, שם מלא, תאריך לידה
- מספר ת"ז, שם מלא
- מספר אישי, שם מלא
-
- מספר אישי
- ת"ז

מפתחות של קבוצת ישיות (הmarsh)

- מפתח ראשי (primary key) הנו אחד המפתחות הקבילים של קבוצת הishiות
 - לקבוצת ישיות מפתח ראשי יחיד
 - בקבוצת הishiות "חיל" מספר אישי או מספר ת"ז יכולים להוות מפתח ראשי



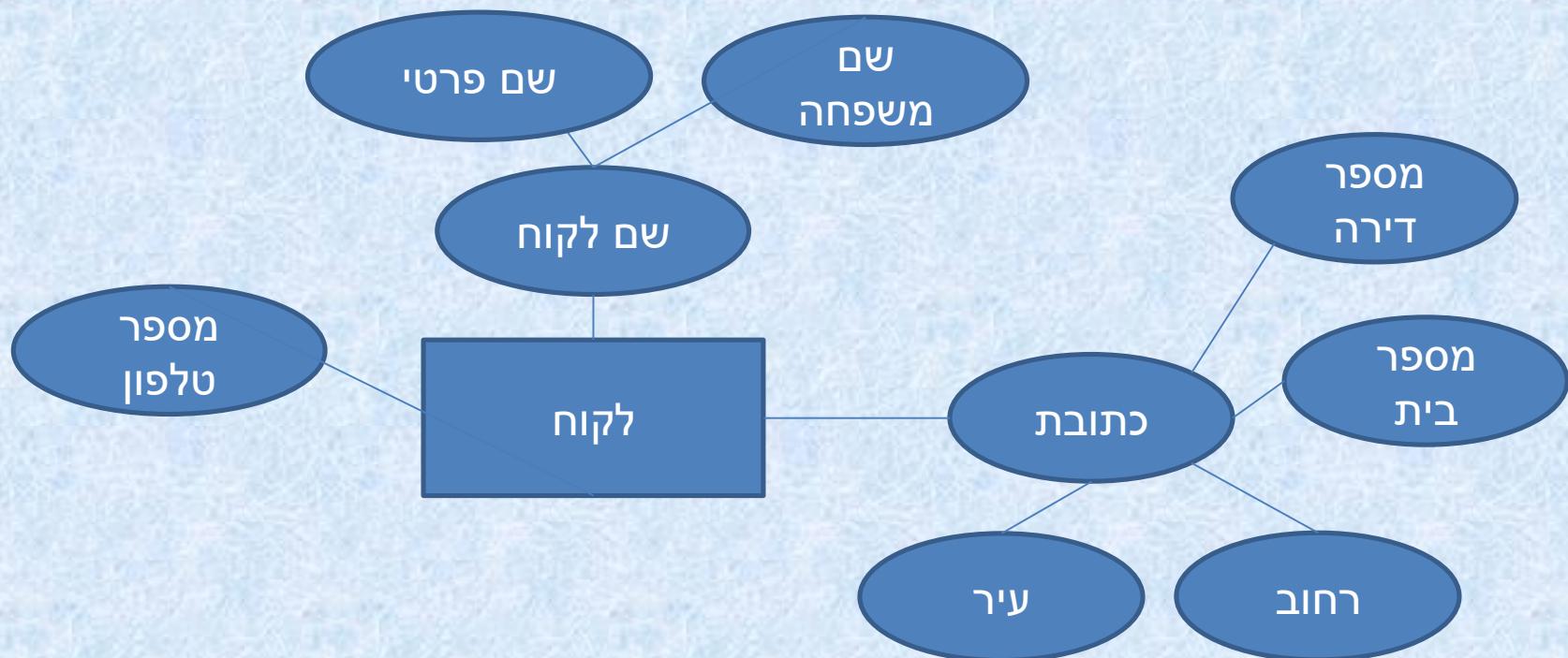
מפתחות של קבוצת ישיות (המשר)



- **תכונות רצויות של מפתח ראשי:**
 - ערךן של השדות במפתח קבוע ולא משתנה לאורך הזמן
 - אי אפשר לזהות ישות עפ"י ערך שהשתנה
 - השדות במפתח אינן מקבלות ערכי *null*
 - חייב להיות ערך מוגדר כדי לזהות באמצעותו את הישות
 - המפתח אינו מורכב מדי
 - מפתח משמש לקישור בין ישיות ולא נרצה שה קישור יהיה מסורבל
 - מפתח המורכב משדות רבים יחולף במפתח פשוט (למשל מספור אוטומטי)
- **עדיף שדה מספרי על טקסט**
 - מחפשים את ישות ע"י השוואת הערך המבוקש לערך בשדה המפתח. השוואת מספרים עיליה לאין ערוך על השוואת מחרוזות, או סוג מידע אחרים!

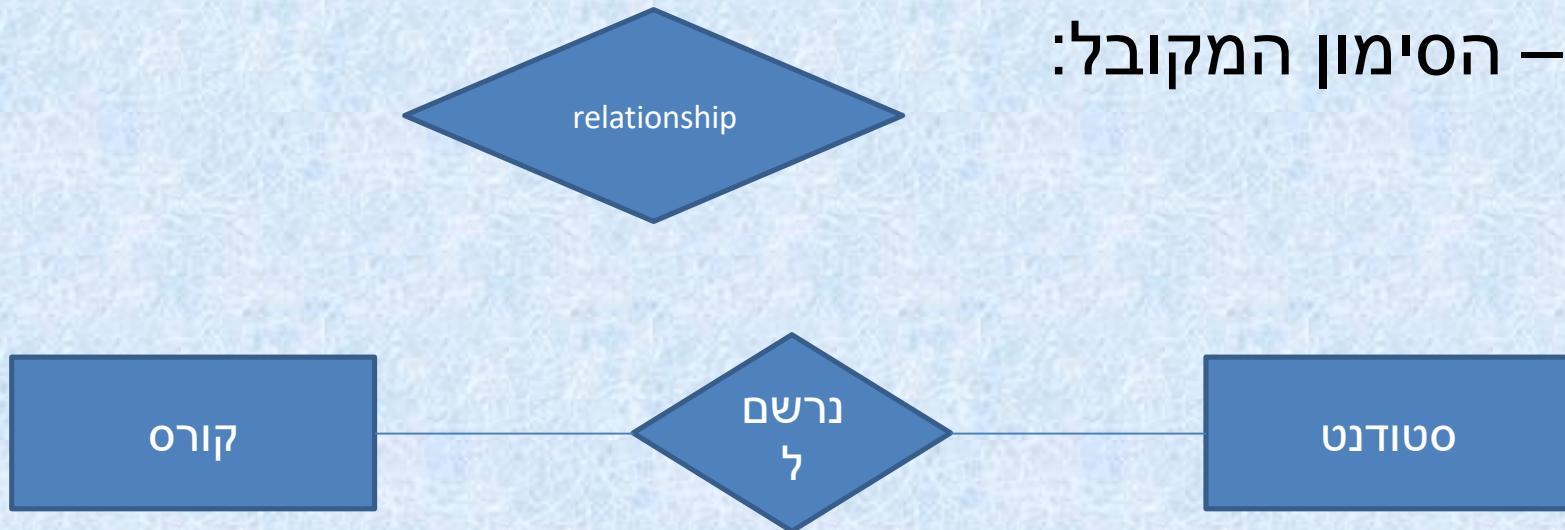
מפתחות של קבוצת ישיות (המשר)

– אתר את המפתחות הקבילים, ובחר מ当中ם מפתח ראשי מתאים



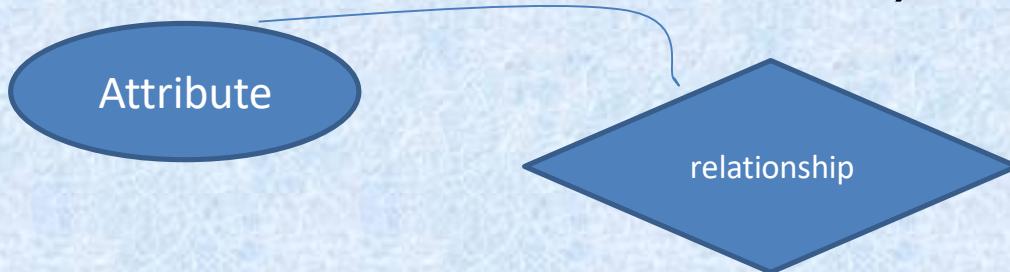
מהו קשר?

- קשר (Relationship) מתאר יחס (חיבור) בין קבוצות ישויות
 - למשל, סטודנט נרשם לקורס
 - הסימן המקובל:

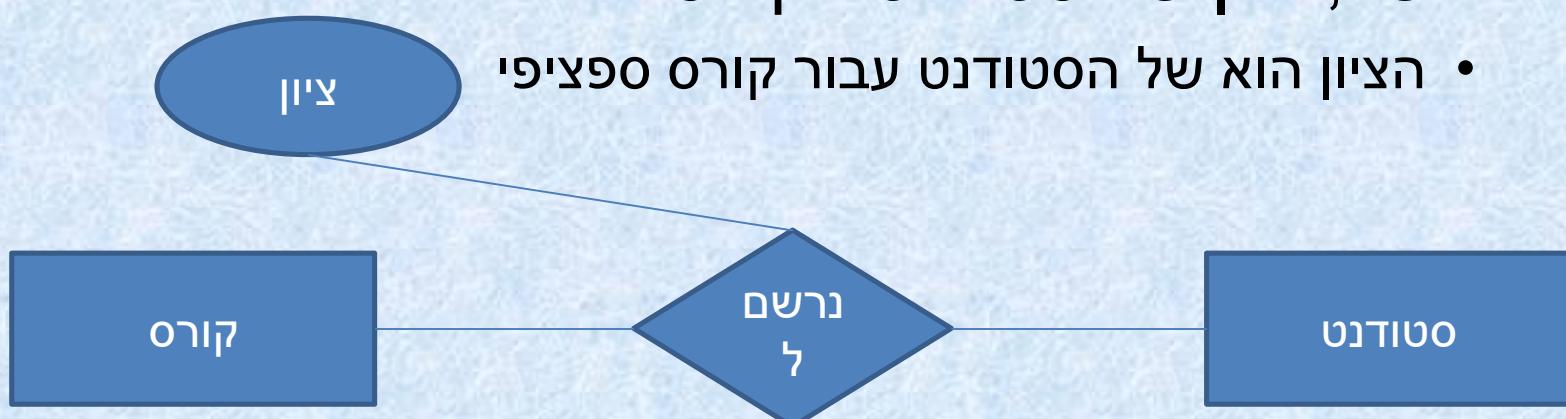


מהו קשר? מהי תכונה של קשר?

- **תכונה של קשר** הינה ערך המאפיין את החיבור בין היחסיות (ולא של היחסיות המעורבות בפניהם עצמן)



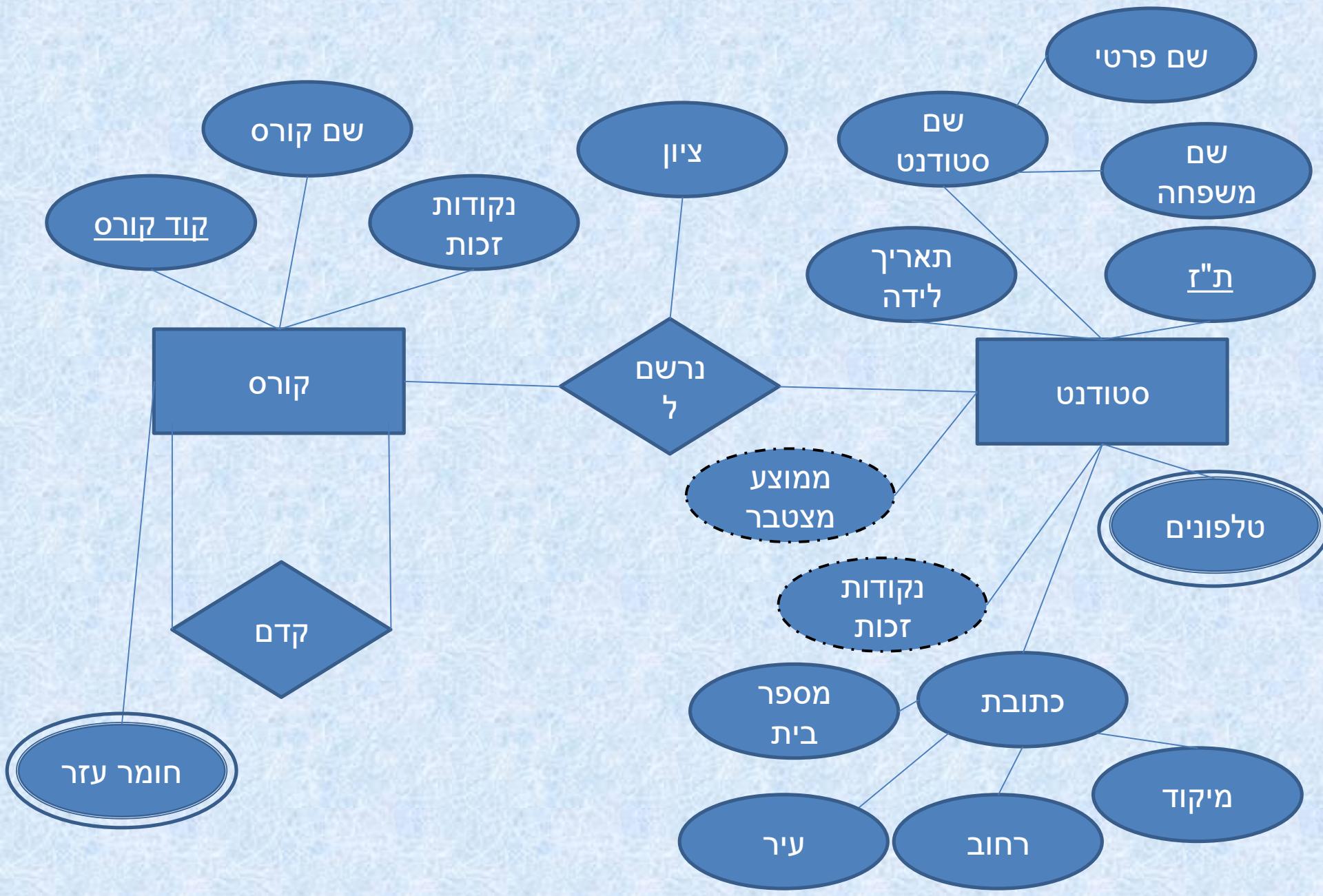
– **למשל, ציון של סטודנט בקורס**



דרגה של קשר

- דרגה (degree) של קשר מתייחסת למספר קבוצת הישיות המשתתפות בקשר
 - קשר רפלקסיבי – לחבר שני ישיות **אותו סוג** (קורס מהו
קדם ל קורס)
 - למשל קורס "הסתברות" מהו קורס קדם לקורס "סטטיסטיקה"
 - קשר ביני – מחברים שני ישיות **מקבוצות ישות שונות** (סטודנט **רשום ל** קורס), הקשרים הנפוצים ביותר
 - למשל הסטודנט "משה זוכmir" רשום לקורס "חקר ביצועים"
 - קשר משולש – לחבר בין 3 קבוצות ישיות (**רופא רושם עבור חוליה תרופה**)
 - למשל "דר' ספר" רושם תרופה "רפאן" לחולה "משה זוכmir"
 - מה לגבי המינון של התרופה?

דיאגרמת יישויות – קשרים (ERD)



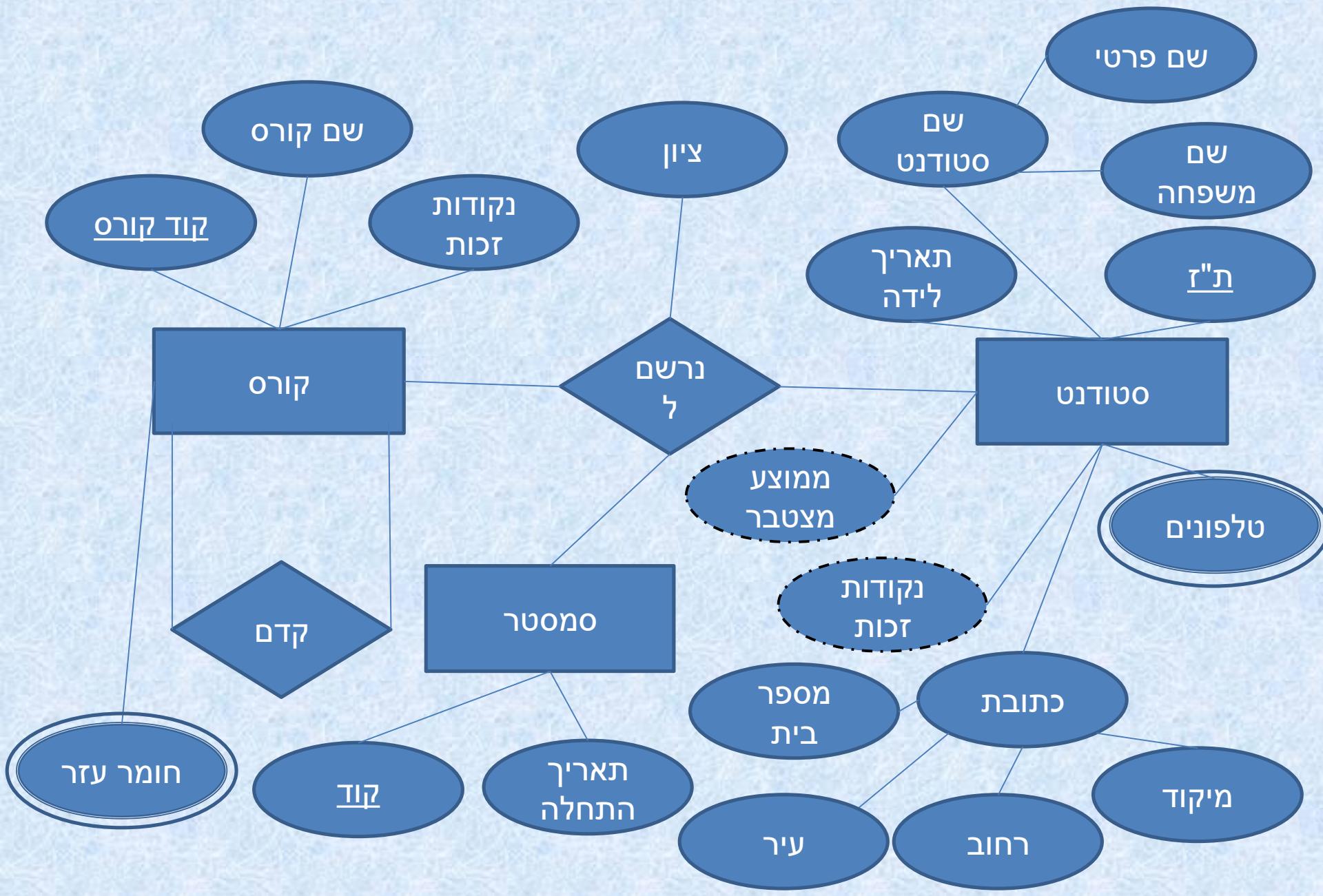
דיאגרמת ישות – קשרים (ERD)

- סטודנטים באוניברסיטה ניגשים לעיתים לגשת לקורס פעם נוספת כדי לשפר ציון או במקרה של כישלון
- הסטודנט יכול לגשת לקורס פעם נוספת בסמסטר אחר (כלומר בכל סמסטר יכול הסטודנט להירשם מחדש לקורס) ולקבל ציון חדש
- כיצד נקבע את האילוץ החדש?



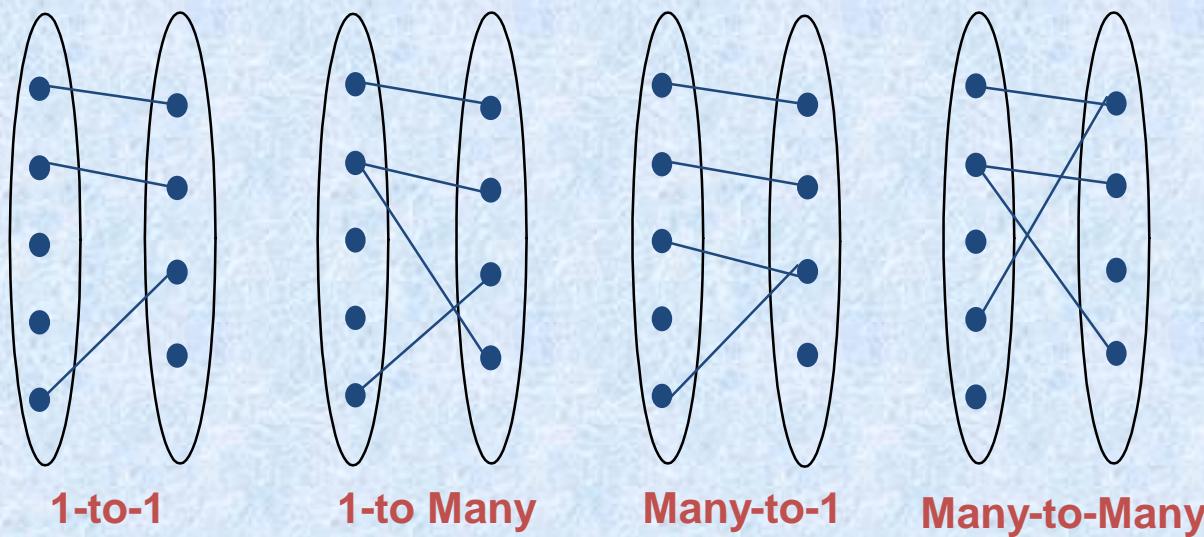
כיצד נוכל להבדיל בין שני רישומים
של אותו סטודנט לאותו קורס?

דיאגרמת יישויות – קשרים (ERD)



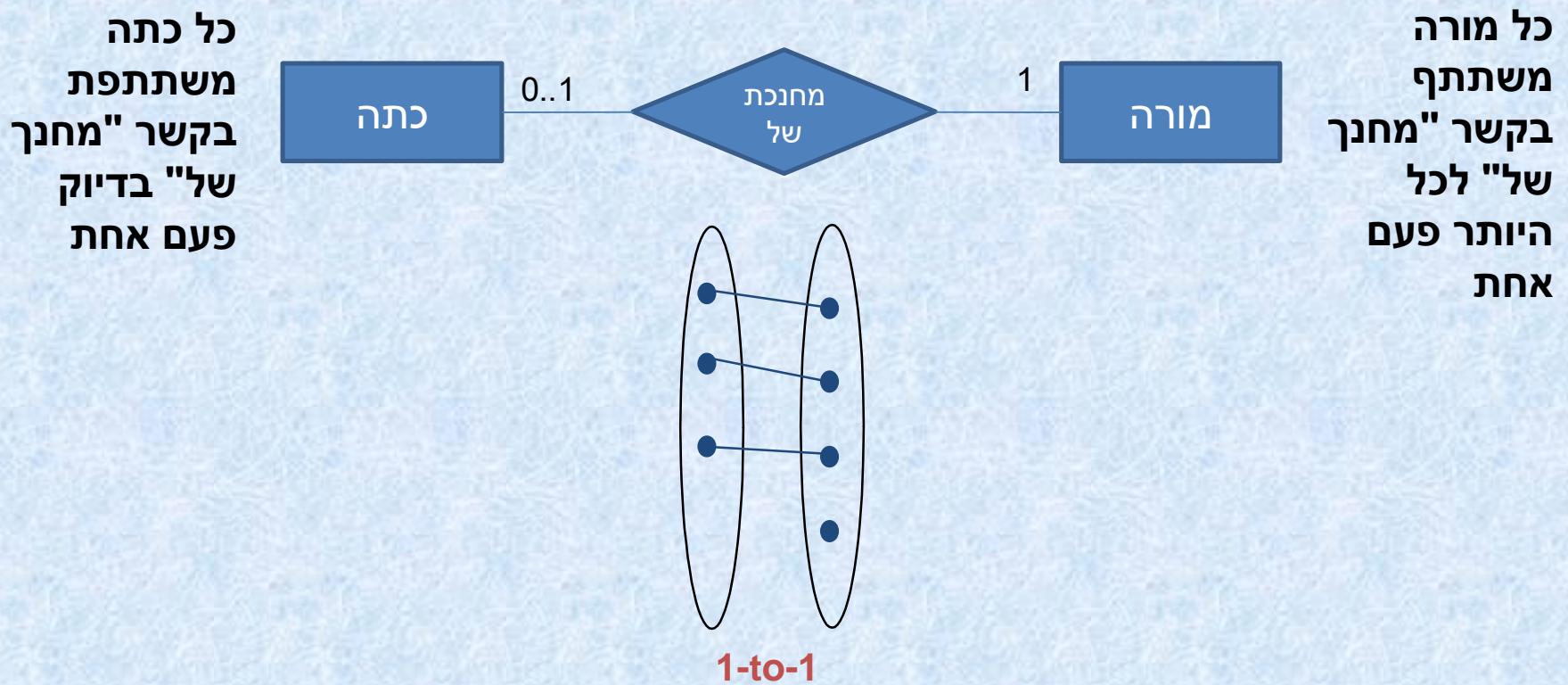
ריבוי של קבוצת קשרים (אלוצי השתפות)

- ריבוי (cardinality, multiplicity) מבטא את מספר הפעמים שישיות יכולות להשתתף בקשר



ריבוי של קבוצת קשרים (אילוצי השתתפות)

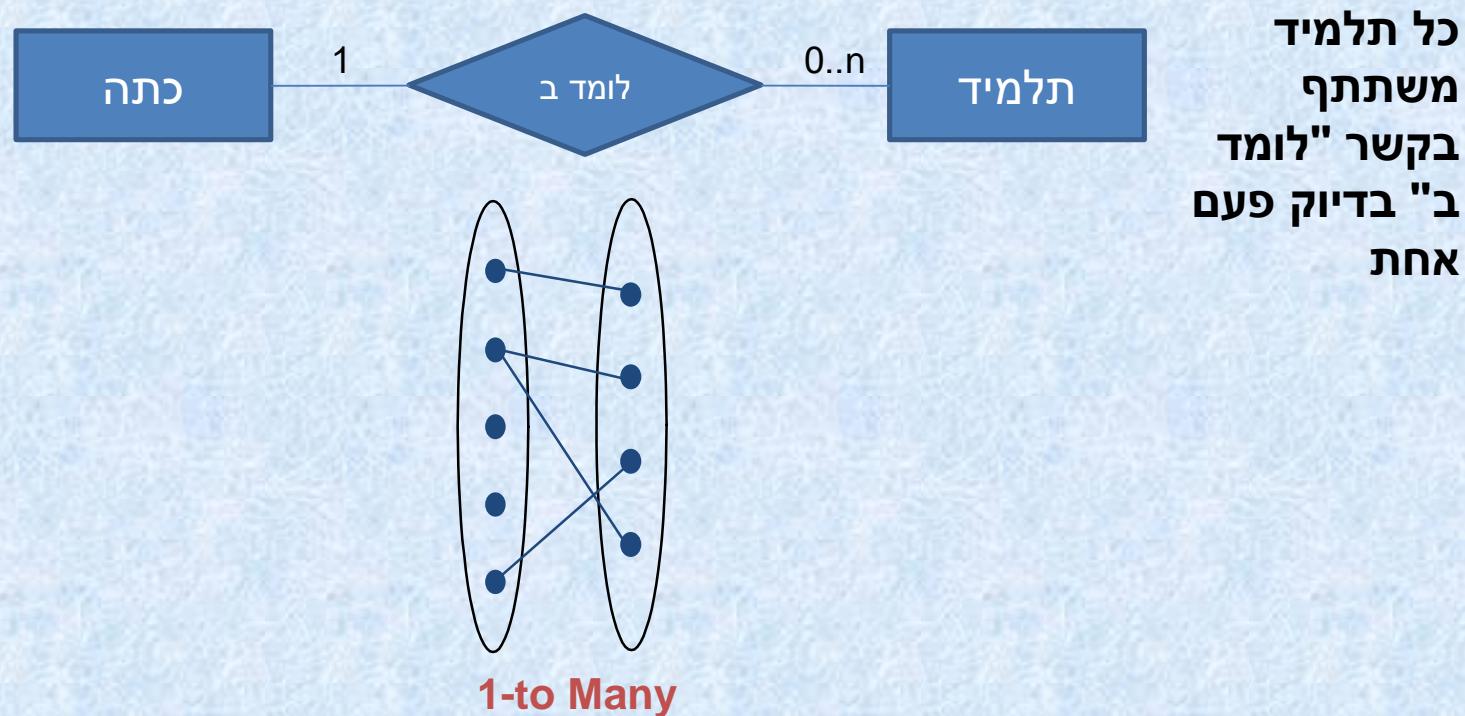
- אחד לאחד (one to one) (1:1)
 - דוגמא, מורה מחנכת של כיתה. לכתחה תוגדר מורה מחנכת אחת, למורה מחנכת תהיה רק כיתה אחת שהיא מוגדרת כמחנכת כל כיתה



ריבוי של קבוצת קשרים (אלוציא השתפות)

- אחד לרבים (one to many) (1:m)
 - דוגמא, תלמיד לומד בכיתה. בכתה לימוד (בי"ס יסודי) משובצים תלמידים רבים, תלמיד משובץ לכיתה לימוד אחת

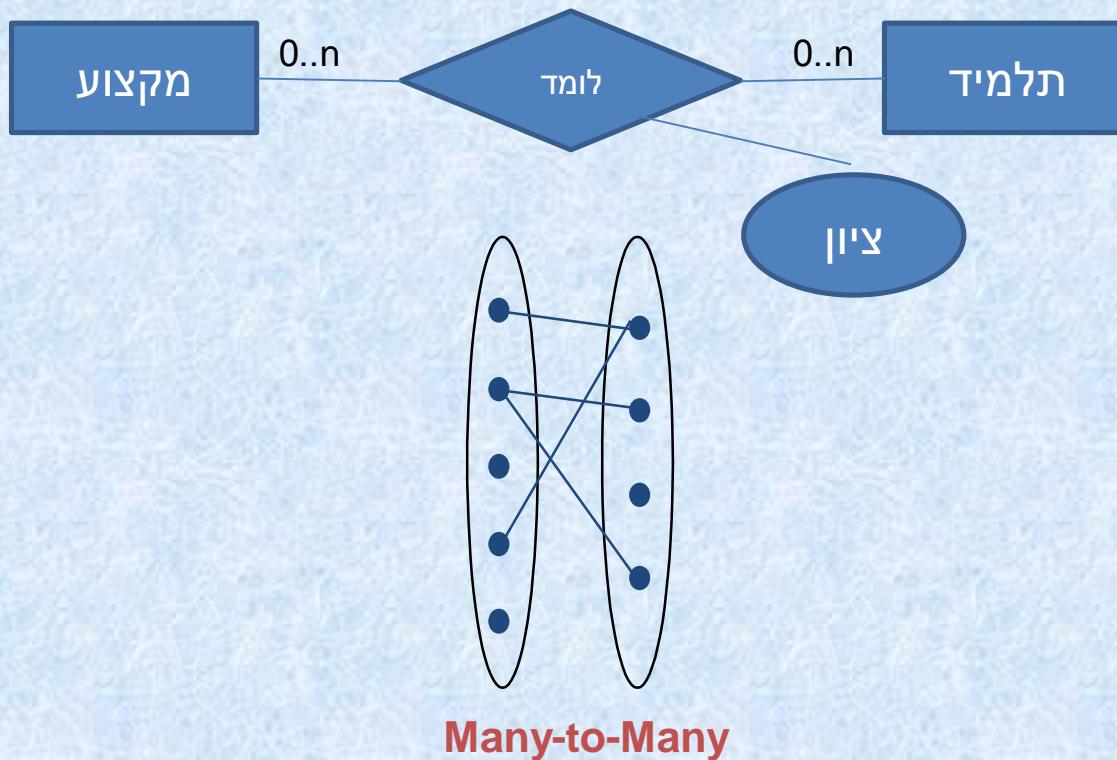
כל כיתה יכולה להשתתף בקשר "לומד ב" מספר רב של פעמים, אך מצד שני אינה חייבת כלל להשתתף בקשר



ריבוי של קבוצת קשרים (אלוציא השותפות)

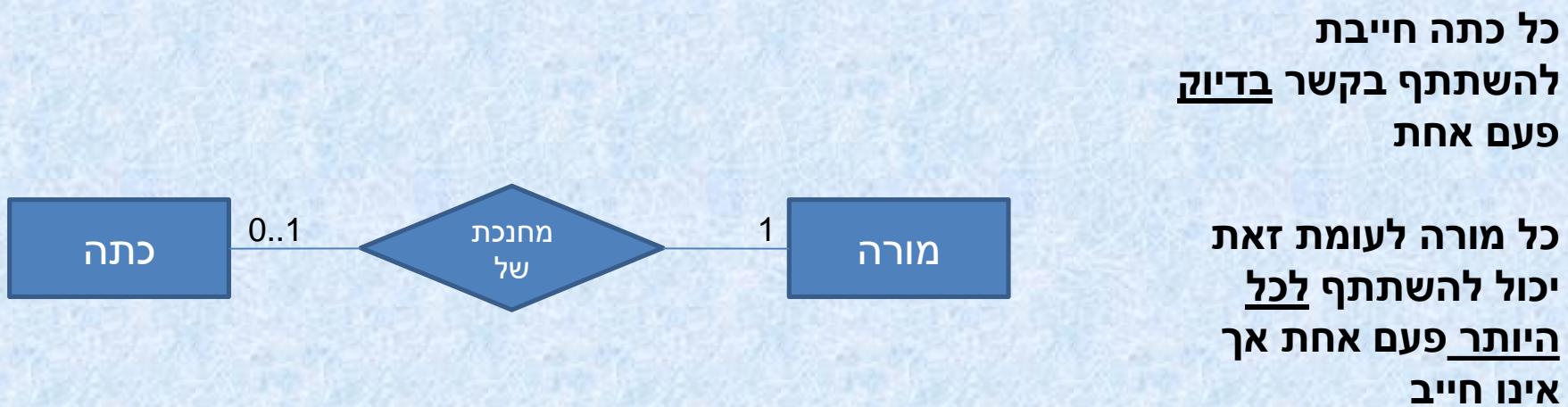
- רבים לרבים (many to many) (m:m)
 - דוגמה, תלמיד לומד הרבה מקצועות. כל מקצוע נלמד על ידי הרבה תלמידים

אין מגבלת
מינימום ואין
מגבלה
מקסימום על
השתנות ישות
בקשר



ריבוי של קבוצת קשרים (אלוציא השותפות)

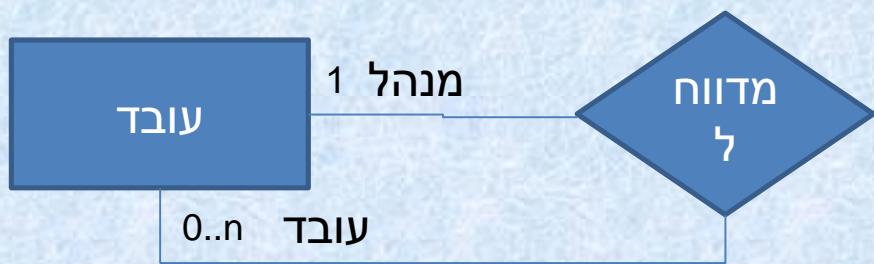
- השותפות חובה מול השותפות אפשרית (אופציונאלית)



- לכטה **חייב** להיות מחניך 1 (כתה **חייבת** להשתתף בקשר)
- מורה **יכול** (אך **אינו חייב**...) לחנוך לכל היותר כטה אחת

תפקידי (קבוצות) ישות בקשר

- **תפקידים נחוצים (לעתים כדי להבחין בין קבוצות ישות המחברות ע"י קשרים**
 - אפשרים קריית היחסים בכוון הנכון
 - תפקידים הם אופציונליים
 - תפקידים נרשמיים על גבי הקשר
 - נפוץ בעיקר בקשרים רפלקסיביים (Self associations)

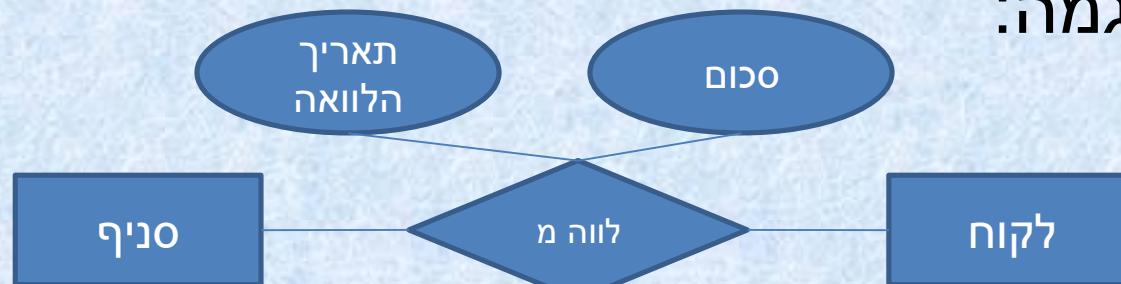


- עובד יכול לנוהל 0 או יותר עובדים
- לעובד יש מנהל אחד בדיוק

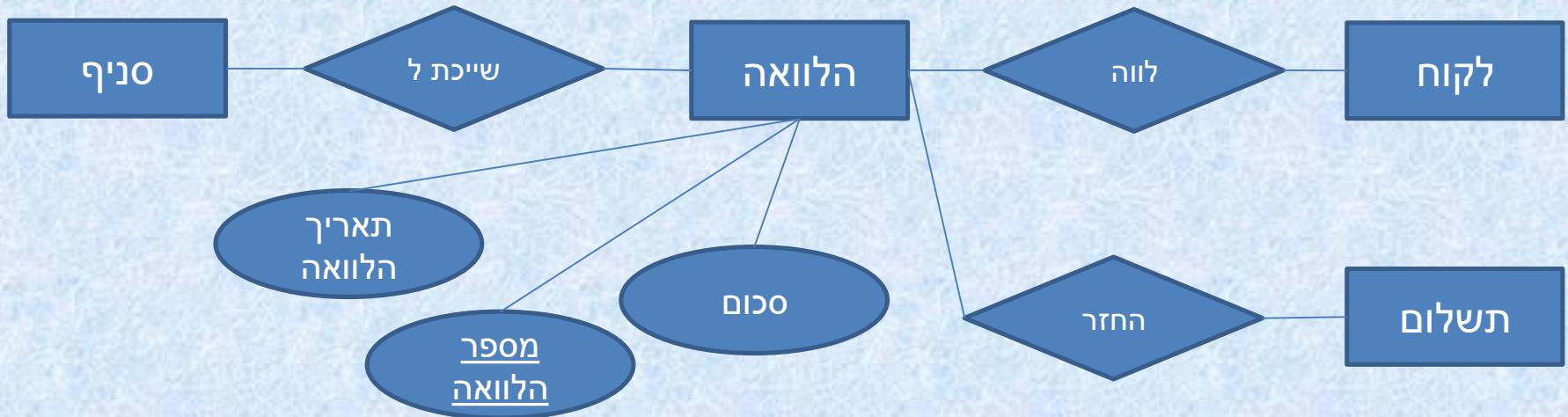
שימוש בקבוצת ישות לעומת שימוש בקשרים

- איך נחליט אם למדל כישות או חלק מקשר?

ההחלטה בהתאם
לחשיבות הישות
לישות יש תוכנות
משל עצמן וקשרים
לקבוצות ישות
נוספות (תשלומיים,
כנסות)



- דוגמה:

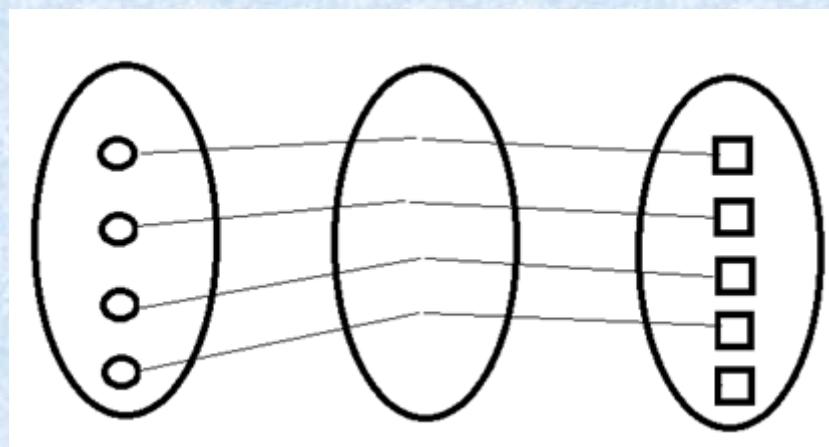
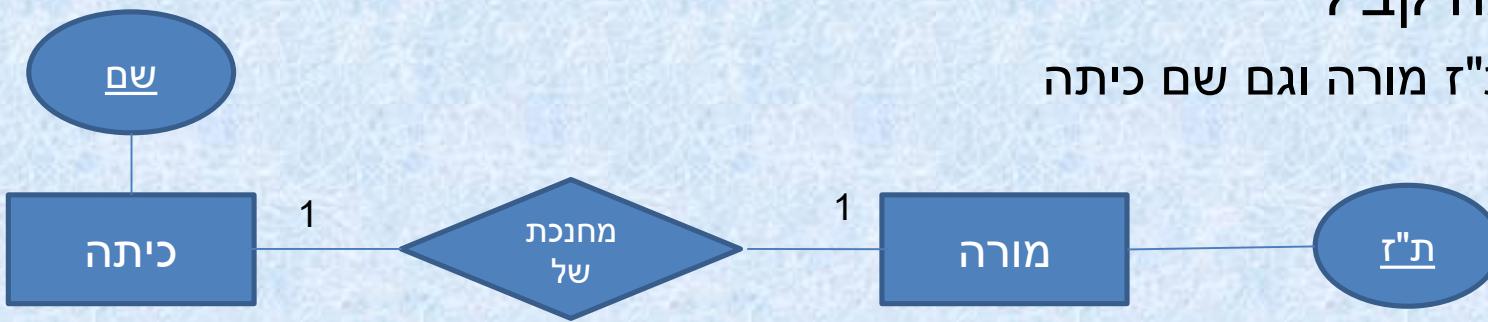


פתרונות של קשר

- גם לקשרים יש מפתח
- מפתח של קשר מראה את הזיהוי (החיבור) עברו כל ישות המשתתפת בקשר
 - לדוגמה, הקשר של סטודנט **רשום ל** קורס צריך להזות גם את הסטודנט וגם את הקורס
- כדי להחליט מהו המפתח הקביל של קשר יש צורך להתייחס לאילוצי ההשתתפות

מפתחות של קשר (המשר)

- קשר 1:1 (אחד לאחד)
 - מורה יכולה לחבר כיתה אחת, לכל כיתה יש מחנכת יחידה
 - ת"ז מורה לא חוזרת פעמיים, וכך גם שם כיתה
 - מפתח קביל
 - ת"ז מורה וגם שם כיתה



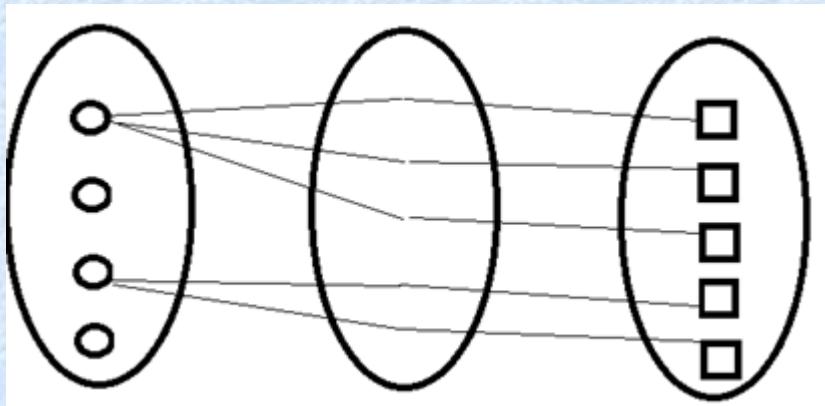
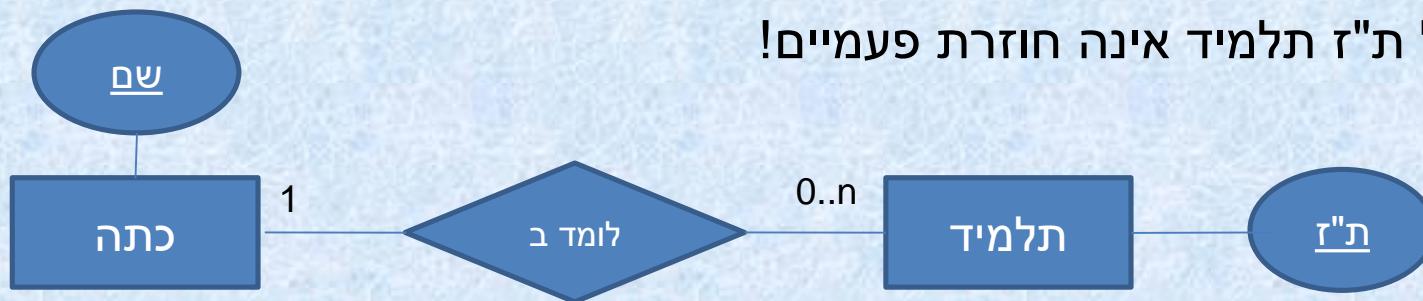
| ת"ז מורה | שם כיתה |
|----------|----------|
| 1א | 11111111 |
| 2א | 22222222 |
| 3א | 33333333 |

מפתחות של קשר (המשר)

- **קשר מ:1 (אחד לרבים)**

- תלמיד יכול ללמידה בכתה אחת, בכטה יכולים ללמידה הרבה תלמידים
- מפתח על הוא צירוף המפתחות של היחסות הקשורות, אך יתכן שיש מפתח קביל שאינו כולל את צירוף המפתחות

- **למשל ת"ז תלמיד אינה חוזרת פעמיים!**



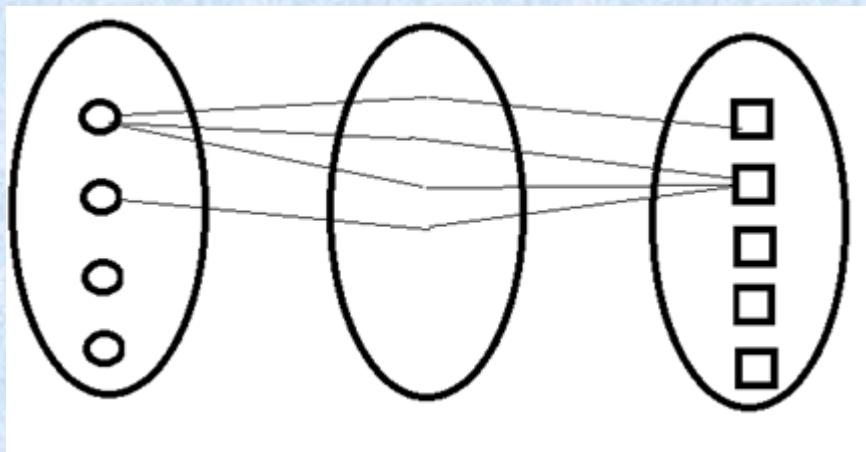
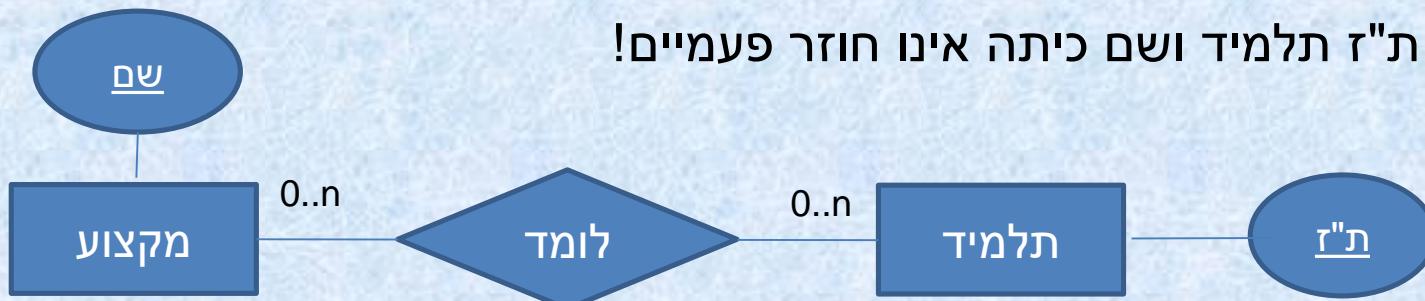
| ת"ז תלמיד | שם כיתה |
|-----------|------------|
| 1א | 1212121212 |
| 1א | 1313131313 |
| 1א | 1414141414 |
| 2א | 1717171717 |
| 3א | 1818181818 |

מפתחות של קשר (המשר)

• קשר m:n (רבים לרבים)

- תלמיד יכול ללמידה מספר מקצועות, מקצוע נלמד ע"י הרבה תלמידים
- המפתח הקביל של הקשר הוא איחוד של המפתחות משתי הקבוצות

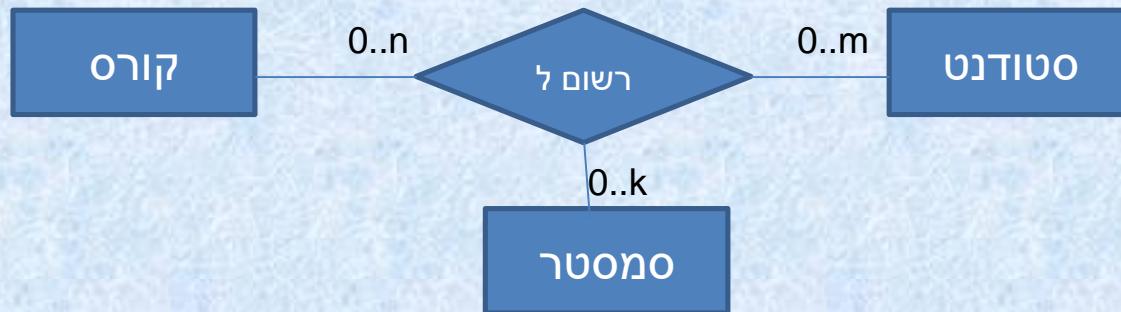
- הצירוף ת"ז תלמיד ושם כיתה אינו חוזר פעמיים!



| ת"ז תלמיד | שם מקצוע |
|-----------|------------|
| חשבון | 1212121212 |
| שפה | 1212121212 |
| חשבון | 1414141414 |
| שפה | 1414141414 |
| חשבון | 1818181818 |

קשר משולש בתרשימים ERD

- מהם הריבויים האפשריים לקשרים משולשים?



- $n:m$ (רבים-רבים-רבים)

- כל סטודנט יכול להירשם לכל קורס בכל סמסטר
- בכל סמסטר ניתנים הרבה קורסים אליהם יכולים להירשם הרבה סטודנטים
- כל קורס יכול להינתן בכל סמסטר ואלו יכולים להירשם הרבה סטודנטים

קשר משולש בתרשים ERD

- מהם הריבויים האפשריים לקשרים משולשים?



- מ: n: 1 (יחיד-רבים-רבים)
 - תלוי מייהו היחיד (בדוגמה שלעיל התפקיד)
 - עובד מסוים בסניף מסוים יכול לבצע תפקיד אחד בלבד
 - « משה משתמש כמאנטח בסניף אחוזה. באותו סניף הוא לא יכול לשמש בשום תפקיד אחר...
 - « אין מניעה שבסניף אחר משה יעבד בתפקיד אחר
 - « אין מניעה שבסניף אחוזה יהיו עוד עובדים בתפקיד מאנטח

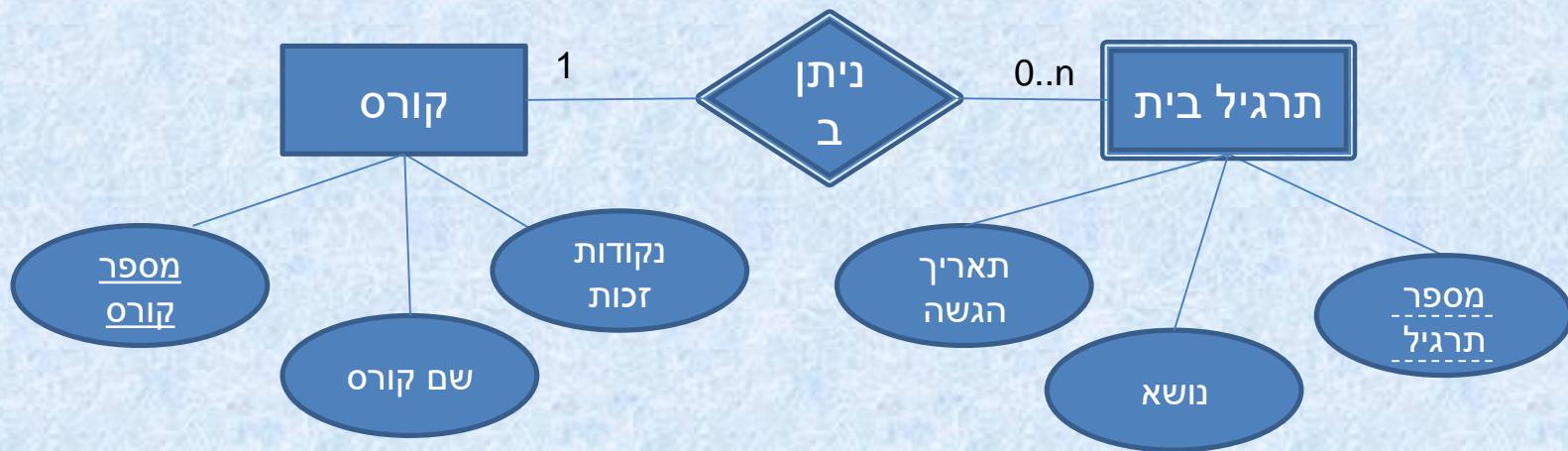
קבוצת ישוות חלשה

- לקבוצת ישוות חלשה שאין לה מפתח ראשי מתייחסים
כאל קבוצת ישוות חלשה (week entity set)
- אז מהו ישות חלשה?
 - ישות שלא ניתן להזדהות אליה ללא קשרו לשות אחרת (חזקה).
לדוגמה תרגיל בית הניתן בקורס לא ניתן להזדהות עפ"י מספרו
(בכל קורס ניתנים כמה תרגילים בית הממוספרים בסדר רצ...)
- ניתן לדבר על תרגיל בית מסוים רק בהקשר של קורס

מוסויים
 - נניח תרגיל בית מס' 1. באיזה קורס?
 - תרגיל בית מס' 2. באיזה קורס?
 - מבלי לדעת מהו הקורס לא ניתן לדעת לאיזה תרגיל מתכוונים,
ולכן חייבים להזדהות את תרגיל הבית גם ע"י הקורס

קובוצת ישוות חלשה (המשר)

- הקיום של קבוצת ישוות חלשה תלוי בקיום של קבוצת
ישוות חזקה המגדירה אותה



- מהו המפתח של הישות הchlשה?
 - מספר הקורס + תכונה השhicת לישות הchlשה ומצהה אותה
בין הישויות השhicת אותה ישות חזקה (mbdil -
(discriminator

קבוצת ישוות חלשה (המשר)

- המבדיל (discriminator) של קבוצת ישוות חלשה הוא קבוצת תכונות המבדילה באופן חלקי בין הישויות של קבוצת הישות החלשה
 - למשל בדוגמה של תרגיל בית בקורס, המבדיל הוא מספר תרגיל. כך נוכל להבדיל בין תרגילי הבית השווים לאותו קורס
- המשמעות של ישות חלשה היא שאין לה זכות קיום ללא הישות החזקה בה היא תלולה, כלומר מחייבת הישות החזקה גוררת מחייבת הישויות החלשות התלוויות בה
 - מחייבת קורס אלגברה מקבוצת הקורסים תגרור את מחייבת כל תרגילי הבית של הקורס מקבוצת תרגילי הבית

קבוצת ישוiot חלשה (המשר)

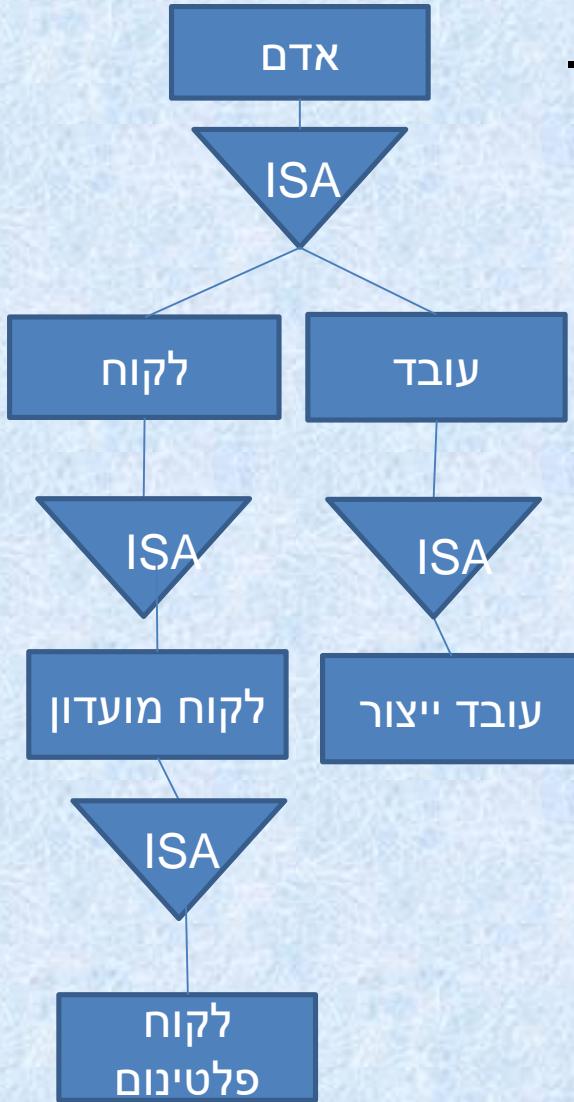
- עוד דוגמאות
 - דירה במבנה
- דירה מזוהה ע"י הבניין אליו היא שייכת (המפתח - גוש/חלה), והבדיל (מספר דירה)
 - שורה בחשבונית
- שורה מזוהה ע"י החשבונית אליה היא שייכת (מספר חשבונית), והבדיל מספר שורה
 - עותק של ספר בספריה
- עותק מזוהה ע"י הספר (isbn) והבדיל הוא מספר עותק

קשרי הכללה - התמחות

- קשרי הכללה התמחות מתייחסים להורשה
- בהורשה, הישות "הירושת" מקבלת את כל התכונות והקשרים של ישות "האב" ומוסיפה תכונות ו/או קשרים ייחודיים משל עצמה
 - חסכו בתכנון / מימוש
 - אפשרות לשימוש חוזר (reuse)
- קשר הכללה – התמחות מצוין ע"י תווית ISA ("is a")
 - לקוח הוא (a-is) אדם
 - עובד יוצר הוא עובד



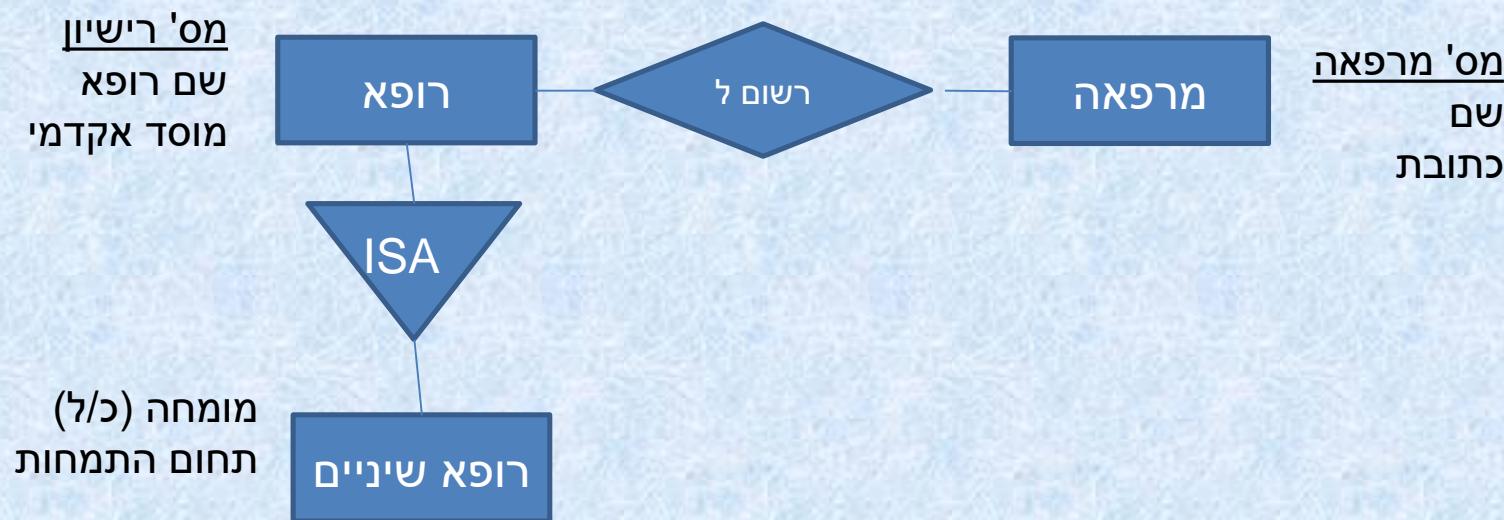
קשרי הכללה – התמחות (המשר)



- ניתן לבצע כמה רמות של קשרי הכללה -
התמחות
- עובד הוא אדם
- ל Koh הוא אדם
- עובד ייצור הוא עובד
- Koh מועדון הוא Koh
- Koh פלטינום הוא Koh מועדון

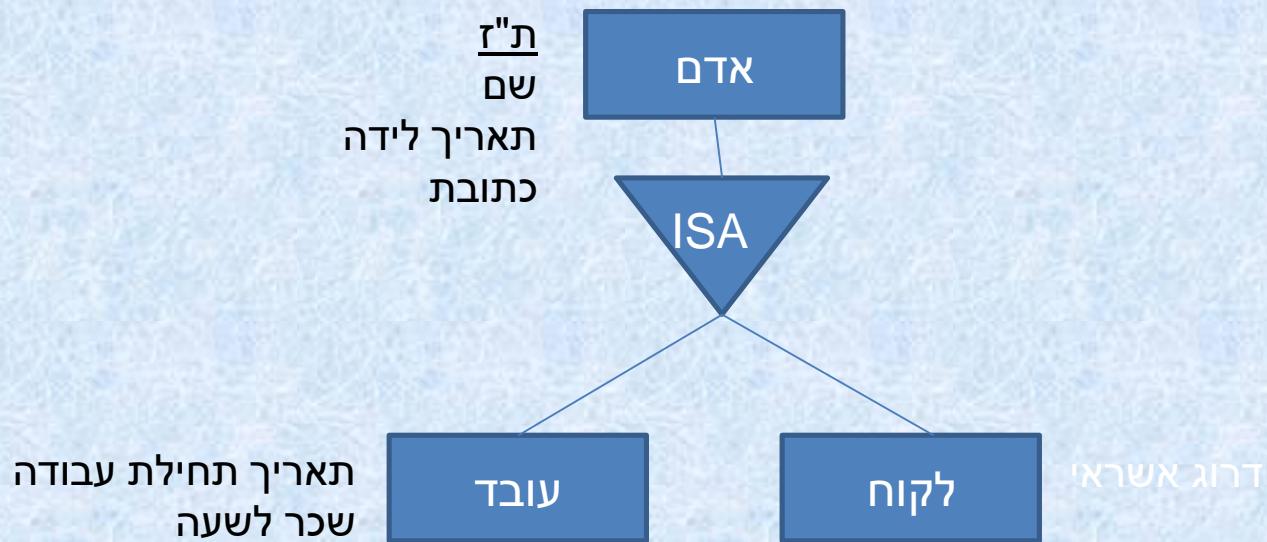
קשרי הכללה – התמחות (המשך)

- התמחות: מבחינים בתתי קבוצות בתוך קבוצה ישיות הנבדלות מישיות אחרות
 - למשל, רופא שניים הוא רופא שומרים עליו פרטים נוספים
 - רופא שניים יורש את כל תכונות הרופא (ואת הקשרים שלו לישיות אחרות)



קשרי הכללה – התמונות (המשך)

- הכללה: מצרפים מספר קבוצות של ישוות בעלות אותם המאפיינים (בחלוקת) לקבוצה ישוות ברמה גבוהה יותר
 - למשל, לך ועובד שניהם אנשים (שנרצה לשמר עבורם מידע)



AILOCIM SHAL KSHRI HACELLA HATMACHOT

1. אילוצים המגדירים האם ישויות יכולות להשתיר ליותר מקבוצת ישויות ברמה הנמוכה

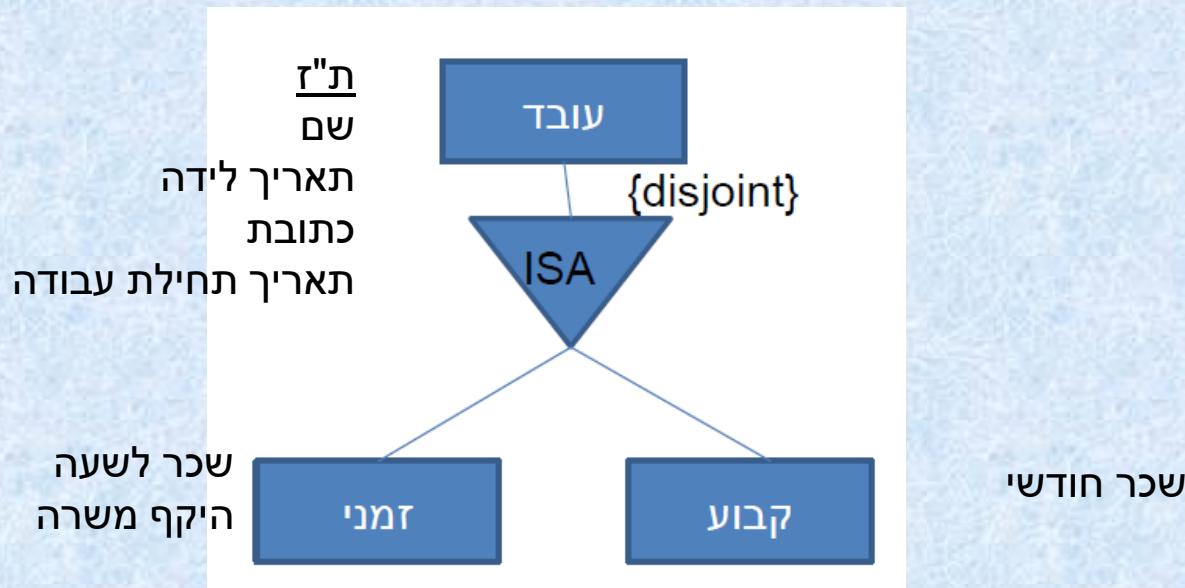
– Disjoint – ישות יכולה להשתיר רק לקבוצת ישויות ברמה הנמוכה

- מצוין בתרשים ERD ע"י כתיבת המילה disjoint ליד מושלש ה ISA

- עובד יכול להיות

- קבוע או זמני

- לא שניהם



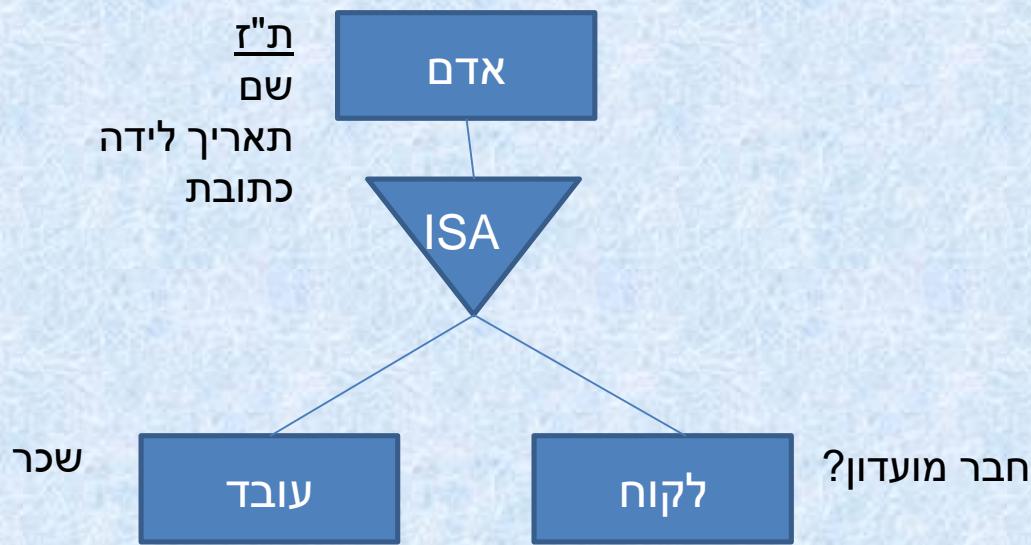
אלוצים של קשרי הכללה התמחות (המשר)

– ישות יכולה להשתир למספר קבוצות ישיות – Overlapping –
ברמה הנמוכה

- זהה ב irritant המclid של קשרי הכללה - התמחות

- אדם יכול להיות

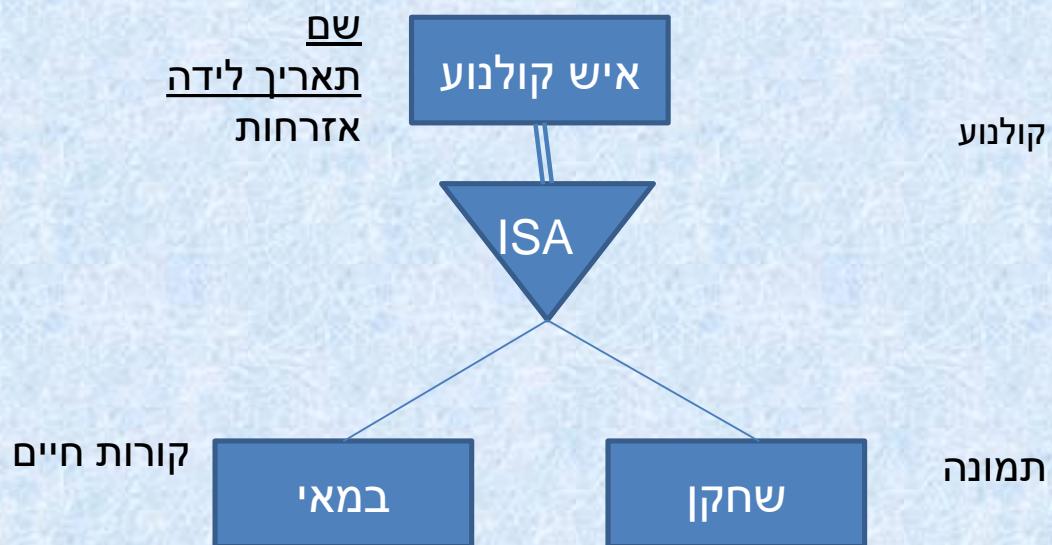
- ל��וח
- עובד
- שניהם



AILOCIM SHAL KSHRI HALLA HTMCHOT (HMSER)

2. אילוצים המגדירים האם ישיות ברמה הגבוהה חייבות להשתיר לפחות לקבוצת ישיות אחת ברמה הנמוכה – Total – ישות מהרמה הגבוהה חייבת להשתיר לפחות לקבוצת ישיות אחת ברמה הנמוכה

- מצין בתרשים ERD ע"י קשר כפול (total)

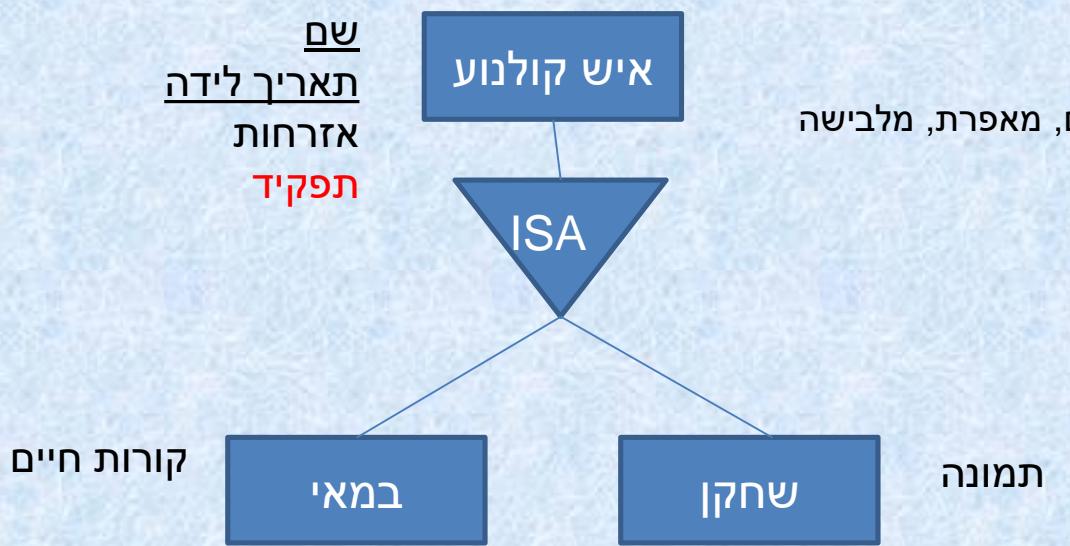


- איש קולנוע חייב להיות
 - שחקן או במאי (או שניהם)
 - לא נשמר מידע עבור איש קולנוע
שהוא לא שחקן או במאי

AILOCIM SHAL KSHRI HACELLA HATMCHOT (HMSH)

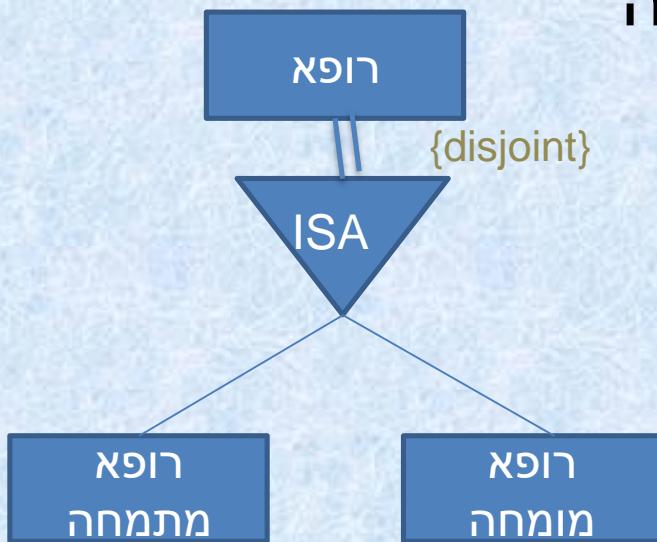
– Partial – ישות מהרמה הגבוהה לא חייבת להשתיר לקבוצת ישיות אחת בرمאה הנמוכה, כלומר יכולות להיות ישיות שהן רק בرمאה הגבוהה ואין משתיקות לקבוצות השיות בرمאות הנמוכות יותר

- זהה בירית המחדל של קשיי הכללה התמחות
- איש קולנוע יכול להיות
 - שחקו
 - במא
 - או כל תפקיד אחר כגון צלם, מאפרת, מלכישת



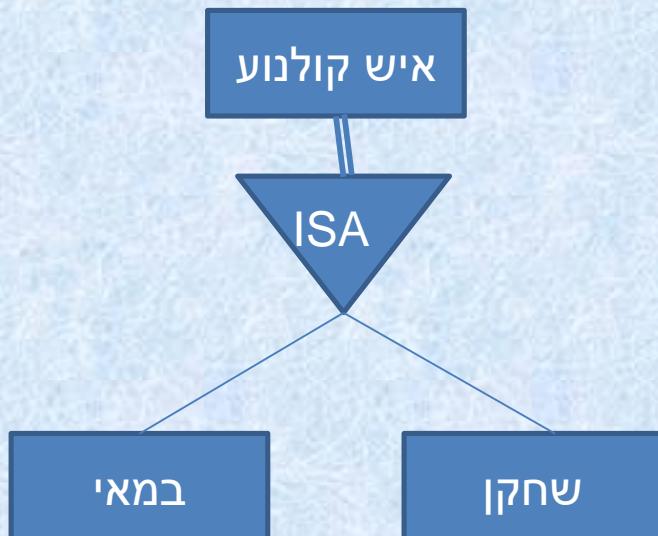
אילוצים של קשרי הכללה התמחות (המשר)

- **שילובים אפשריים בין שני סוגי אילוצים:**
- total + disjoint
- רופא חייב להיות מומחה או מתמחה
 - לא נשמר מידע על רופא שאינו אחד מהן"ל
- רופא מומחה לא יכול להיות מתמחה
ולהיפך



אילוצים של קשרי הכללה התמחות (המשך)

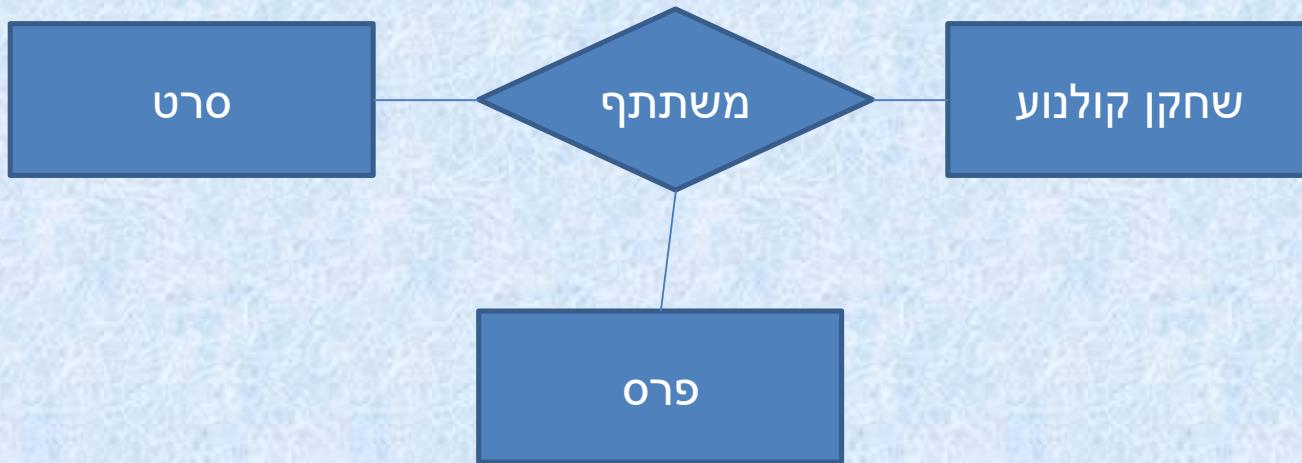
- total + overlapping
- אדם יכול להיות שכון או במאי
 - לא נשמר מידע על אדם שאינו שכון או במאי
- שכון יכול להיות גם במאי ולהיפר



באופן דומה ניתן להגיד גם קשרים מסווג disjunct + partial -
partial + overlapping

הקבוצה (Aggregation)

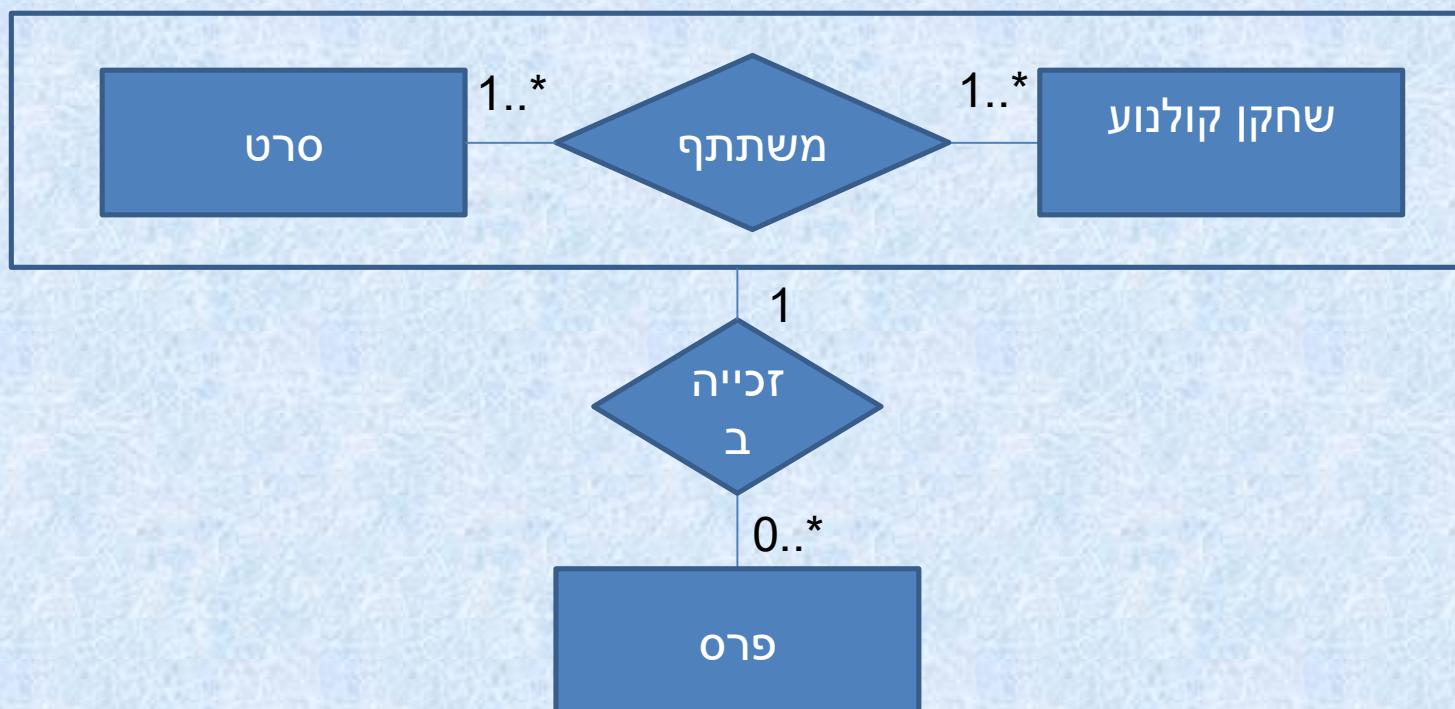
- שחקן קולנוע יכול לזכות בפרס על תפקידו בסרט
- נרצה לטעד את הפרסים בהם זכה
- פתרון אפשרי



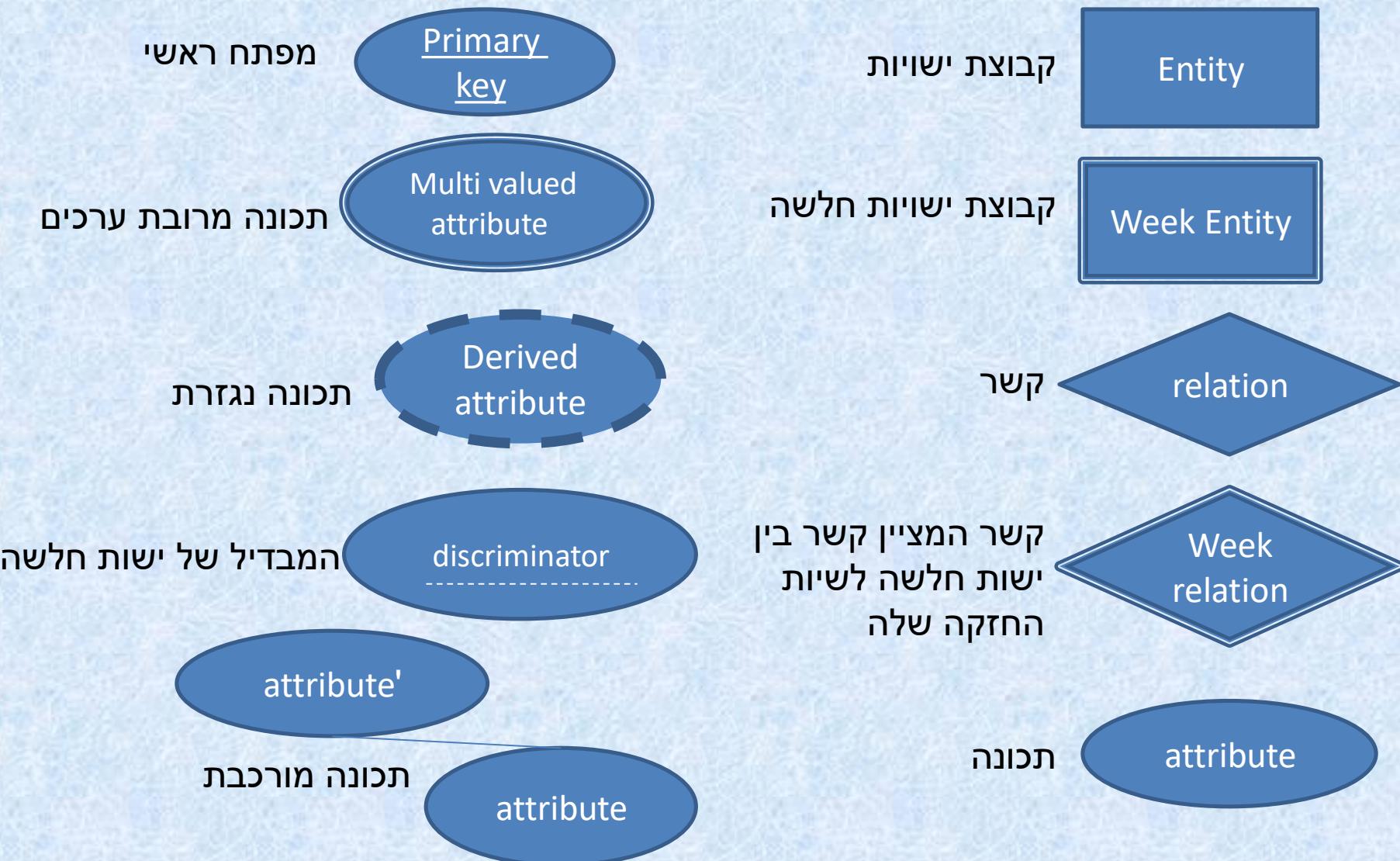
- מה הבעה בפתרון הנ"ל?
- -מה יקרה אם השחקן לא זכה בפרס??

הקבצה (Aggregation) (הmarsh)

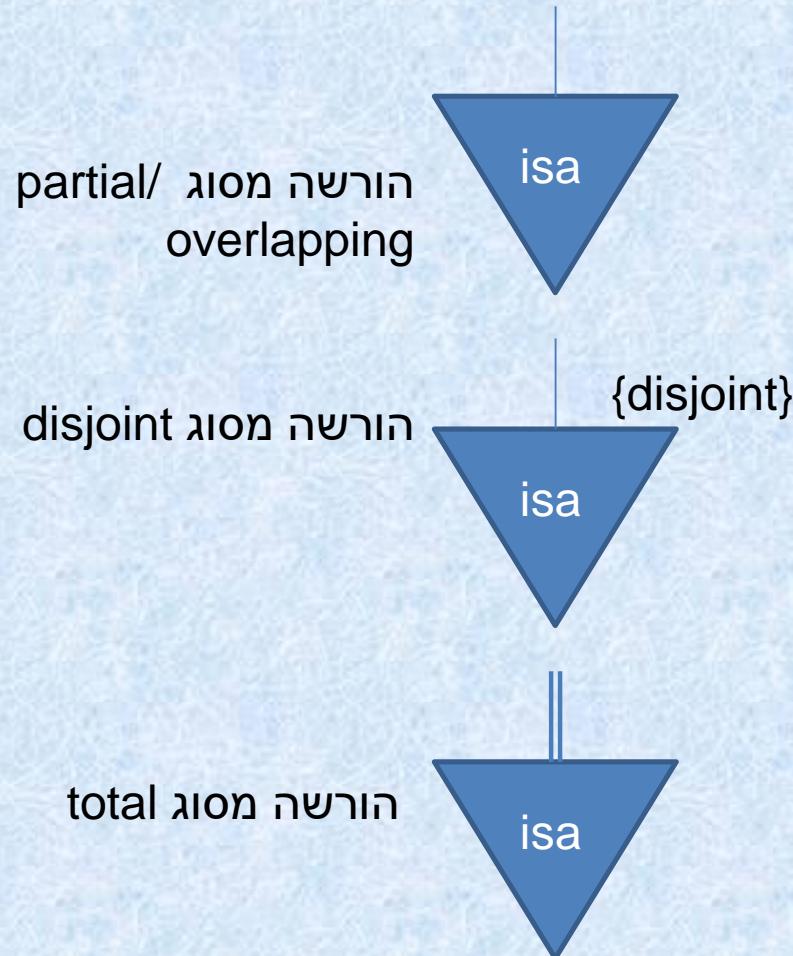
- נרצה לתעד בנפרד את עובdet השתתפות השחקן בסרט ואת זכיותו בפרס עבור השתתפותו בסרט (אם זכה)
 - נהפוך את "שחקן קולנוע משתתף בסרט" ל"ישות מקובצת" וניתור קשר בין הישות המקובצת הנ"ל לבין פרס



סיכום סימנים בתרשימים ER



סיכום סימנים בתרשימים ER (הmarsh)

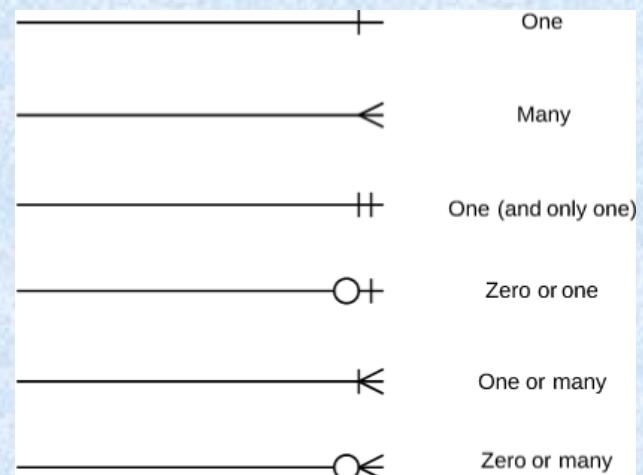
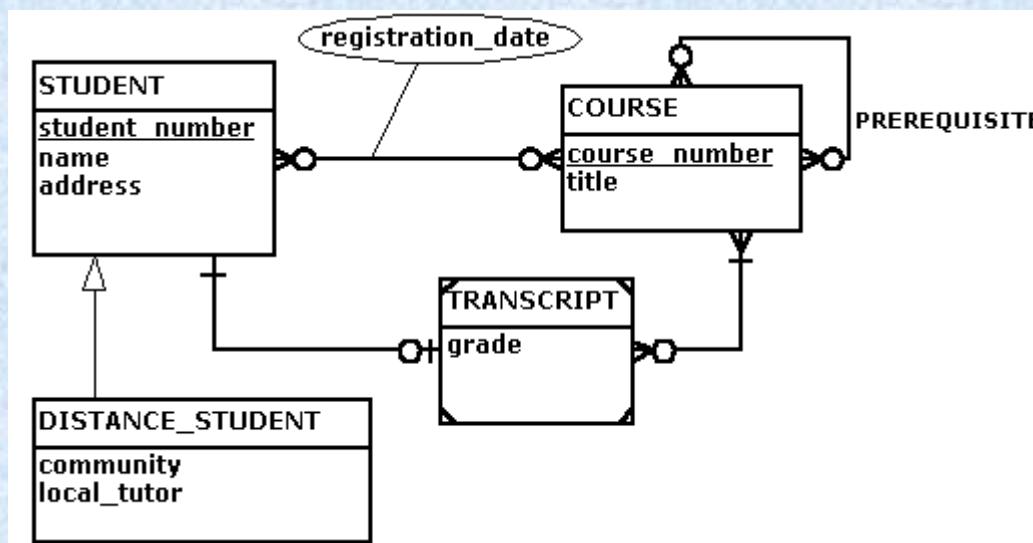


סימולים אלקטרוניים

- קיימות מערכות סימולים רבות לתיאור קשרי ישות וקשרים

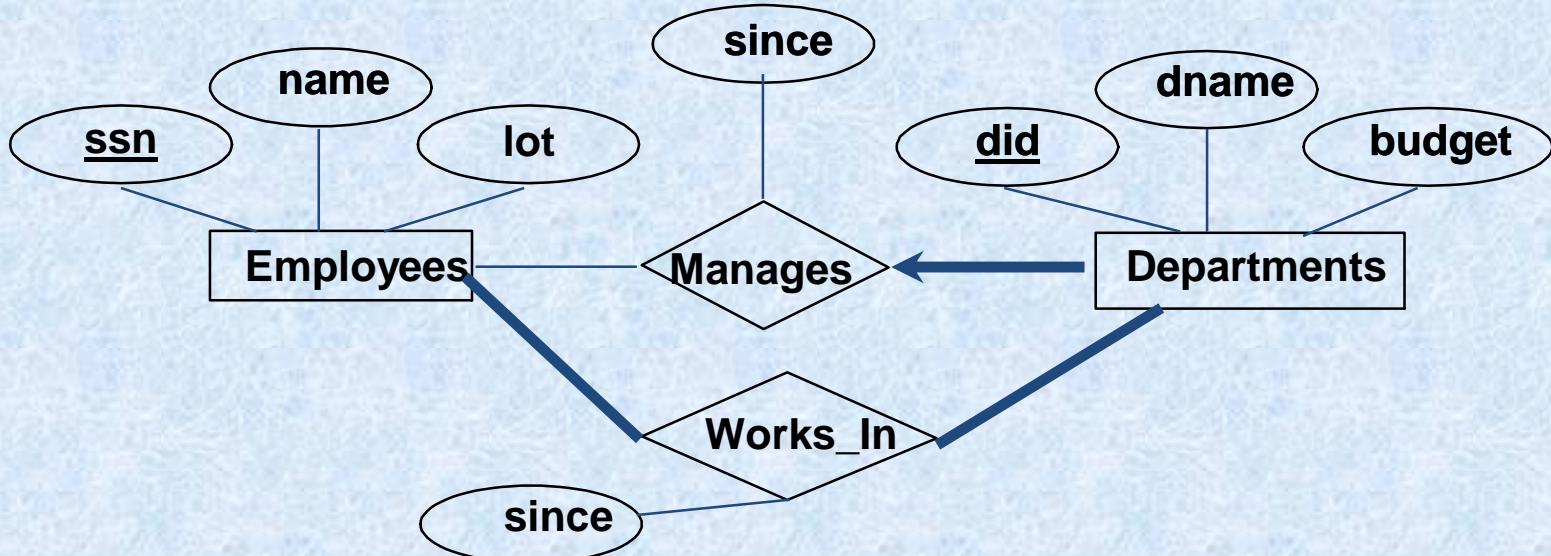
– אנחנו השתמשנו בשיטת סימול הנקראת Chen Notation

– שיטה נוספת נוספת הינה שיטת סימול הנקראת Crow's Foot Notation



סימולים אלטרנטיביים - המשך

- שיטת סימול נוספת



- כל מחלקה משתתפת בקשר **Manages** לפחות פעם אחת (קו עבה), ולכל היוטר לפחות פעם אחת (חצ'), כלומר בדיקת פעם אחת
- כל עובד משתתף בקשר **Manages** מספר kaliשו של פעמים
- כל עובד משתתף בקשר **Works_In** לפחות פעם אחת
- כל מחלקה משתתפת בקשר **Works_In** לפחות פעם אחת

חסרונות של תרשימים ER

- ב ER לא ניתן לייצג אילוצים מלבד אילוצי ריבוי והשתתפות
 - למשל לא ניתן לתאר את האילוצים הבאים
 - כל האנשים מעל גיל 65 הם פנסיונרים
 - שכר עובד לשעה לא יהיה גבוה משכר מנהל לשעה
- אין התייחסות לתכונות בהם חובה להכנס עריך לשדות בהם מותרים ערכי Null

בנייה של תרשימים ER

1. זיהוי קבוצות הישויות (אילו אובייקטים ישמרו בensed הנתונים) – בד"כ שמות עצם
2. זיהוי התכונות של כל קבוצה ישויות (אילו תכונות מאפייניות כל ישות ונרצה לשמר בensed הנתונים)
3. זיהוי מפתחות לכל קבוצה ישויות (איזו תכונה תזהה באופן חד ערכי ישות מתוך קבוצת ישויות)
4. זיהוי הקשרים והריבוי המתאים בין קבוצות הישויות (בד"כ פעלים)
5. זיהוי קשרי הכללה והתמחות
6. זיהוי ישויות חלשות והקבצות

תרגיל לדוגמא – סיפור מסגרת

חברת "love to somebody" מעוניינת להקים סטארט-אפ בתחום ההיירות. הרעיון הוא לפתח אפליקציה שתקרה "me match" שתאגד את פרטי המשתמשים השונים שרשומים לאפליקציות ההיירות הקיימות בשוק הישראלי ולבצע התאמה בין משתמשים ללא תלות באפליקציה בה המשתמש רשום. כאשר משתמש מhapus התאמה ונמצאה כזאת, האפליקציה תפנה את המשתמש המhapus לאפליקציית ההיירות המתאימה יחד עם שם המשתמש של המשתמש האחר שנמצא מתאים.

תרגיל לדוגמא – סיפור מסגרת

אפליקציית "me match" קיבל את המידע אודות המשתמשים מן אפליקציות ההיירויות הקיימות וכן מן המשתמשים השונים. משתמשים שעדיין לא רשומים לאפליקציות ההיירויות יכולים להירשם לאפליקציות ההיירויות השונות באמצעות "me match". המשתמשים יזינו את פרטייהם הכלליים מייל (יחידי, שיישמש אותם לזהוי בכלל אפליקציות ההיירויות השונות), שם פרטי, שם משפחה, תאריך לידה (על המשתמש להיות לפחות בן 18 בעת ההרשמה), מגדר (זכר/ נקבה/ אחר). שדות אלו יוגדר כחובבה. המשתמש יוכל להוסיף אם ירצה תיאור קצר אודוטיו (שיישמר כתקסט). על המשתמשים יהיה לבחור מtower רשימה את עיר מגוריהם בה הם גרים.

תרגיל לדוגמא – סיפור מסגרת

בנוסף המשתמשים יצינו את סוגי הקשרים אותם הם מוחפשים. דוגמאות לסוגי קשריהם: קשר לטווח ארוך, יחסים פתוחים, קשר סודי וכו'. כמו כן, על המשתמשים להגיד את המין של בן/בת הזוג איתו/הם מעוניינים עבור כל סוג קשר בו בחרו: נקבה/זכר/ שני המינים. המשתמשים יצינו באיזה אפליקציות קיימות הם נמצאים (עבור כל אפליקציה ישמר שם ייחודי, תיאור ותאריך הקמה). המערכת תמליץ על אפליקציות חדשות עבור המשתמש עפ"י מאפייני הקשר המבוקש אך המשתמש יכול לסמך באיזה אפליקציה הוא מעוניין להופיע. עבור כל אפליקציה שסומנה ע"י המשתמש שהוא נמצא / רוצה להרשם ישמר תאריך ההרשמה עבור האפליקציה.

תרגיל לדוגמא – סיפור מסגרת

עבור משתמש באפליקציה יש לשמור את הינו של המשתמש באפליקציה המסויימת. יש לציין כי באפליקציה מסויימת הינו המשתמש יכול להופיע רק פעם אחת, כלומר באפליקציה מסויימת לא יוכל כי יהיו שני משתמשים בעלי אותוmeno.

המערכת מייבאת עבור כל משתמש קיימים את תמונותיו מן האפליקציות השונות אליו הוא רשום. כל תמונה תקבל מספר סידורי ייחודי מן המערכת וישמר הנתיב בו נשמרת (אין צורך לשמור מailו אפליקציות יובאו). המשתמש יוכל להוסיף תמונות נוספות. במידה ולא תמונות שהועלו ע"י המשתמש ישירות ל-”me match” ולא יבואו מהאפליקציות, יוכל המשתמש להוסיף לתאריך צילום התמונה ולציין תיאור קצר אודות התמונה (עד 255 תווים).

תרגיל לדוגמא – סיפור מסגרת

לאחר שהמשתמש מסמן את האפליקציה בה הוא רוצה להופיע ”match me“ תציג לו שאלות מוקונות שדורשות פרטים נוספים עברו כל אפליקציה. למשל, עבור כל אפליקציה תוגדר סדרה של שאלות ייחודיות. השאלות ימוספרו בסדר עוקב (החל מהמספר 1) עבור כל אפליקציה. מלבד המספר הסידורי של השאלה עבור האפליקציה, יש לשמור את תוכן השאלה. עבור כל שאלה מוגדרות 4 תשובות אפשריות בלבד והמשתמש צריך לסמן את התשובה הרלוונטית עבורה (מתוך הארבע) עבור כל שאלה. בנוסף לתשובה שסימן (1-4), יש לשמור את תאריך מתן התשובה (תאריך מתן התשובה יהיה התאריך בו השאלה נענתה במערכת כערך ברירת מחדל). ישן שאלות שהן בגדר חובה ואחרות בגדר רשות. שימו לב כי על מנת למנוע כרטיסי משתמשים עם מידע חסר, עבור שאלות עליהן החלטת המשתמש לא לענות, אין צורך לשמור תשובה ריקה.

תרגיל לדוגמא – סיפור מסגרת

המשתמש יוכל בהמשך להחליט לענות/לעתק את תשובתו, וaz התשובה החדשה תשמור במקום התשובה הישנה (בנוסף לתאריך המענה החדש).

משתמש יוכל לבצע חיפוש להתאמה בין כל המשתמשים שמודרים ב- "me". ההתאמה נעשית על ידי אלגוריתם פנימי במערכת הלקח בחשבון עיר מגורים, גיל, תשבות דומות לשאלות, סוג הקשר המבוקש, מגדר מבוקש וכו'. כאשר נמצאת ההתאמה בין משתמשים שונים על ידי האלגוריתם, המערכת תציג למשתמש המחבר את רשימת המשתמשים המתאים שנמצאו בחיפוש שתכלול את שם המשתמש שנמצא מתאים ("המתאים") והאפליקציות בהן הוא רשום. ההתאמות האפשרות יוצגו למשתמש בסדר יורד לפי דירוג ההתאמה. המשתמש יוכל לסמן ולשמור מຕור תוצאות החיפוש מותאים פוטנציאליים באפליקציות ספציפיות שאיתם רוצה ליצור קשר.

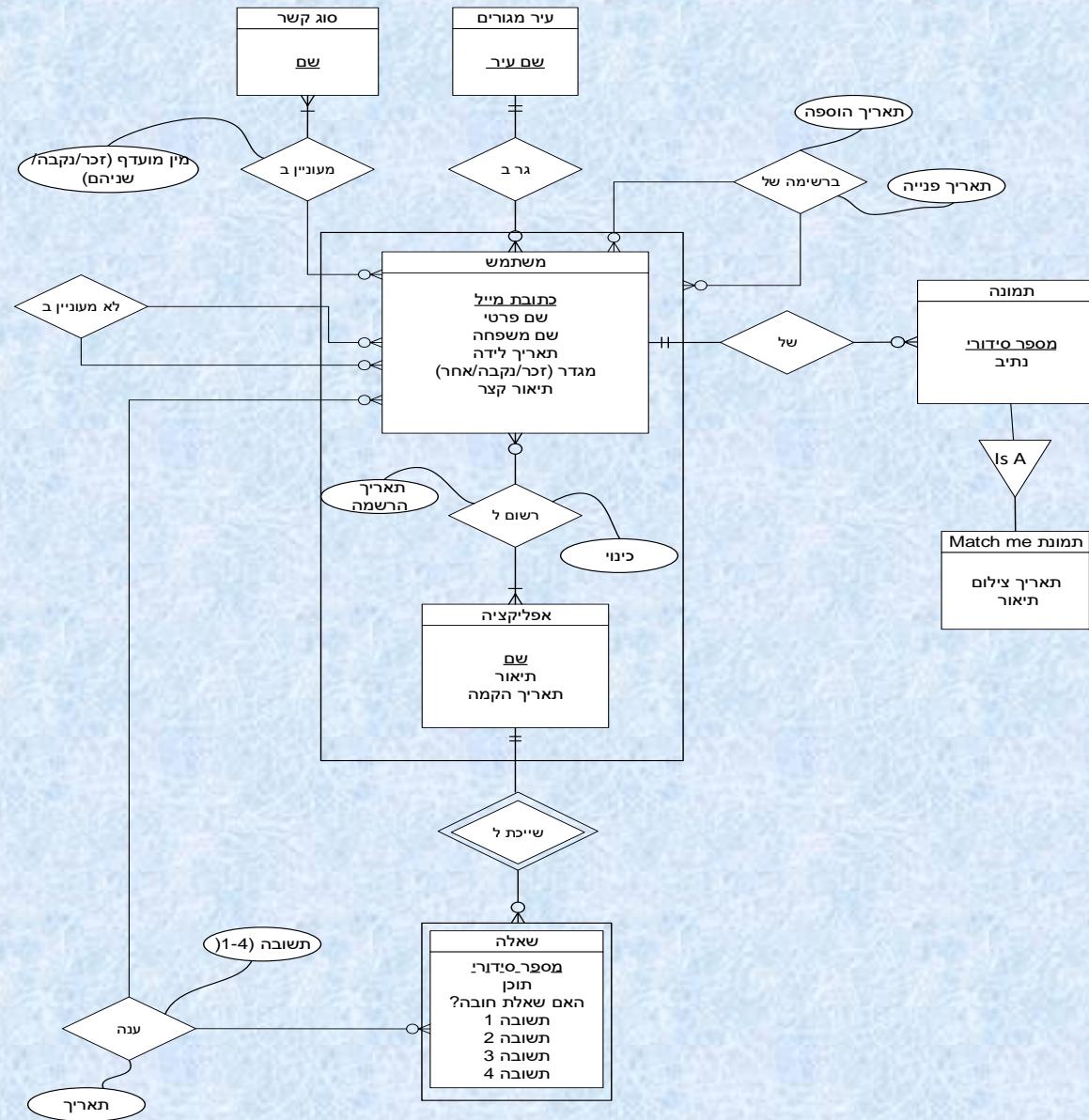
תרגיל לדוגמא – סיפור מסגרת

המשתמש יוכל לצפות בראשימת התאמות בכל זמן שיחסוץ ולחוק מותאים מן הרשימה. במידה ומבצע חיפושים נוספים, יוצגו לו רק מותאים שאינם מופיעים כבר בראשימה שלו. המחפש יוכל להוסיף אל הרשימה מותאים חדשים שיצאו בחיפוש החדש. במקרה, עברו כל מותאם שנבחר לשומר לרשימה ישמר תאריך ההוספה לרשימה. בעת צפייה בראשימה, המשתמש יוכל לחוץ על לינק של המותאם יחד עם האפליקציה איתה הוא רוצה ליצור קשר עם המותאם קשר ו-“match me”. תעביר את המשתמש לאפליקציה הרלוונטיית לשם יצירת קשר עם “המותאם”.

תרגיל לדוגמא – סיפור מסגרת

כאשר משתמש לוחץ על לינק לאפליקציה עבור יצירת קשר עם מותאם, האפליקציה תשמור את תאריך הפניה לאפליקציה עם המשתמש (המותאם) אליו הצד המחבר מעוניין ליצור קשר. מלבד הוספה לרשיימה, יוכל המשתמש לסמן כאשר מוצגות לו תוצאות החיפוש כי אינו מעוניין שמותאם אחר יוזג לו שוב בתוצאות החיפוש. במקרה זה המותאם לא יופיע בתוצאות החיפוש. על מנת למנוע מצבים שבו משתמש 1 שאינו מעוניין בשימוש 2 יופיע ברשיימה של משתמש 2, משתמש 1 אינו יופיע בתוצאות החיפוש של משתמש 2.

תרגיל לדוגמא - ERD



המודל הטבלאי – Relational Data Model

- המודל הטבלאי (רלציוני) הפך לפופולרי בשנות ה- 80 הוא פותח בסוף שנות ה- 60, ע"י Codd, שփש פתרונות לביעות המודלים הקיימים. מכיוון שהירה מתמטיKay, טבעי היה שהמודל שבנה היה מבוסס על קונספטיות מתמטיות
- עיקרו של המודל הוא רעיון הטבלה (גם נקראת יחס) שבה הנתונים מאוחסנים
- כל טבלה מורכבת מרשותן (שורות אופקיות), ושדות (עמודות אנכיות)
- כל טבלה מזוהה ע"י שם ייחודי, ושם זה משמש את בסיס הנתונים כדי לאתר את הטבלה. מנגנון ניהול בסיס הנתונים מאתר לפי שם זה את הטבלה "מאחורי הקלעים"

המודל הטבלאי (המשך)

- יתרון נוסף של המודל הרלציוני בכך שהוא מספק כלים יעילים לניהול בסיס הנתונים, זאת מושם שטבלאות יכולות להכיל לא רק את הנתונים המאוחסנים בפועל, אלא גם מידע על הטבלאות ושמות השדות שבבסיס הנתונים, מתן הרשות גישה לטבלאות, כללי הטיפול בנתונים וכו'.
- כל דבר במודל הרלציוני יכול להיות מאוחסן בטבלאות, משתמש יכול לשאול שאלות בנוגע לשמות הטבלאות, הרשות גישה אליהן, או נתון כל שהוא. תוצאות שאלות אלו יוצגו בפניו بصورة טבלה

המודל הטבלאי (המשר)

- המודל הטבלאי מאפשר תיאור נתונים ע"י אוסף של טבלאות. המודל הטבלאי מכיל
 - טבלאות
 - שדות, מפתח ראשי
- איך ייצגו הנתונים בטבלאות?

המודל הטבלאי (המשר)

- מושתת על מבנה נתונים ייחיד: טבלה
 - כל עמודה מצינית תכונה
 - כל שורה מצינית מופע: ישות (רלציה)

טבלה 1

| טבלה 1 | תכונה 1 | תכונה 2 | תכונה 3 | תכונה 4 |
|--------|---------|---------|---------|---------|
| מופע 1 | | | | |
| מופע 2 | | | | |
| מופע 3 | | | | |
| מופע 4 | | | | |
| . | | | | |
| . | | | | |
| . | | | | |



המודל הטבלאי (המשך)

- כל טבלה מסמלת קבוצת ישוות או קשר בין ישוות
 - לדוגמה: טבלת סטודנטים מסמלת את קבוצת הישויות סטודנטים

| תאריך לידה | שם משפחה | שם פרטי | ת"ז | תמונה |
|------------|----------|---------|------------|---------|
| 01.01.1999 | ישראל | ישראל | 1212121212 | |
| 01.08.2000 | דין | דני | 1313131313 | ← מופיע |
| 01.05.2000 | ירקוני | יפה | 1414141414 | |

- לדוגמה: טבלת ציונים מסמלת את קבוצת הקשיים סטודנט משותף-ב קורס

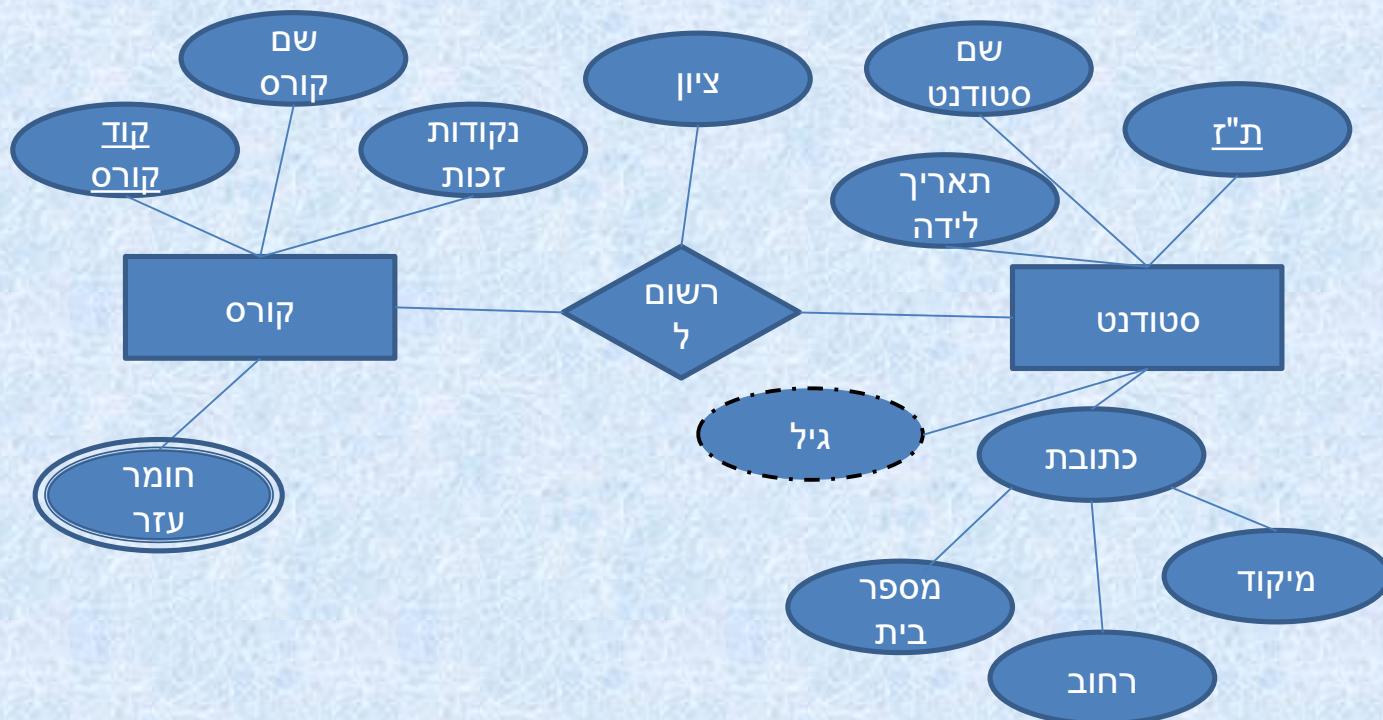
| ציון | מועד קורס | ת"ז סטודנט |
|------|-----------|------------|
| 90 | 2004.13 | 12121212 |
| 95 | 2004.16 | 12121212 |
| 89 | 2004.13 | 13131313 |

המודל הטבלאי (המשך)

- ניתן לבצע המרת אוטומטית של מודל ER למודל טבלאי
- המודל הטבלאי שמתקיים הינו "טוב"
 - מונע כפילות נתונים
 - מתקיים בצורה נורמלית שלישית – 3NF (פרטים בהמשך...)

תרגום טרשיים ER - טבלאות

- תרגום קבוצות ישוות חזקיות לutableאות
 - כל קבוצה ישות חזקה מתורגם לutableה
 - תכונות הישות הם שדות בutableה
 - המפתח הראשי של הישות הוא המפתח הראשי של הutableה
- אילוutableאות נקבל מתרגום קבוצת היחסות החזקיות בטרשיים הבא?



תרגומים תרשיים ER - טבלאות

- **טבלת קורס**
 - קוד קורס יהווה את מפתח הטבלה (לא ניתן יהיה להזין שני קורסים בעלי אותו קוד)

| קוד קורס | שם קורס | נקודות זכות | חומר עזר? |
|----------|---------|-------------|-----------|
| | | | |
| | | | |

- מה לגבי חומר עזר?
 - חומר עזר היא תוכנה מרובת ערכאים ולכן לקורס יכולים להיות 0 או יותר חומרי עזר (הפניות למקורות)
 - איך נשמר את המידע?
 - האם להקצות כמה שדות?
 - אם כן, כמה שדות?
 - פתרון אחר?

תרגום תרשימים ER – תכונה מרובת ערכיים

- **תכונה מרובת ערכיים M של ישות E תייצג ע"י טבלה נפרדת**
 - המפתח הראשי של הטבלה הנפרדת הוא המפתח הראשי של הטבלה המקורי + התכונה M
 - כל ערך של התכונה המרובה מופיע לשורה נפרדת בטבלה החדשה

| חומר עזר | קוד קורס |
|----------|----------|
| | |
| | |

| נקודות זכות | שם קורס | קוד קורס |
|-------------|---------|----------|
| | | |
| | | |

- באופן זה נוכל להוסיף עבור קורס הרבה חומרי עזר, ונמנע משדות NULL מיוחדים

תרגום תרשימים ER – תוכונה מורכבת

- **טבלת סטודנט**

– ת"ז סטודנט תהווה את מפתח הטבלה

| ת"ז | שם סטודנט? | תאריך לידה? | כתובת? | גיל? |
|-----|------------|-------------|--------|------|
| | | | | |
| | | | | |

- מה לגבי כתובות? ושם סטודנט?
 - כתובות היא תוכנה מורכבת, וכך גם שם סטודנט
 - איך נשמר את השדה?
 - תלוי במדיניות הארגון
- תוכנות מורכבות משוטחות ע"י הגדרה תוכנה נפרדת לכל רכיב של התוכנה המורכבת וכן שדות הטבלה יהיו: ת"ז, שם פרטי, שם משפחה,תאריך לידה(?), רחוב, מספר בית, עיר, מיקוד, גיל(?)

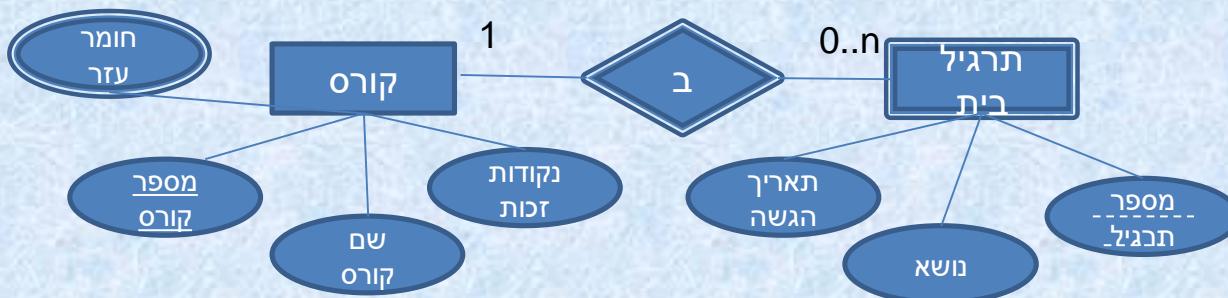
תרגום תרשימים ER – תוכונה מחושבת

- מה לגבי תאריך?
 - תאריך אף הוא שדה מורכב (יום, חודש, שנה) אבל קיימים טיפולים נתוניים Date שלא מחייב פירוק, ומאפשר באמצעות פונקציות ייעודיות גישה למרכיבי התאריך
- מה לגבי גיל?
 - גיל היא תוכונה נגזרת
 - האם יש טעם לשמר תוכנה נגזרת?
- תוכנות נגזרות בד"כ לא נשמרות בבסיס הנתוניים ומחושבות כאשר נדרש להציגן

תרגום טרשים ER – ישות חלשה

• תרגום קבוצות ישות חלשה לטבלאות

- כל קבוצת ישות חלשה מתורגם לatable, כאשר תוכנות קבוצת היסוד הם שדות בטבלה ומפתח הראשי של קבוצת היסוד חזקה מצטרף לשדות הטבלה
- המפתח הראשי של קבוצת היסוד חזקה + המבדיל של קבוצת היסוד הchlsha הוא המפתח הראשי של הטבלה

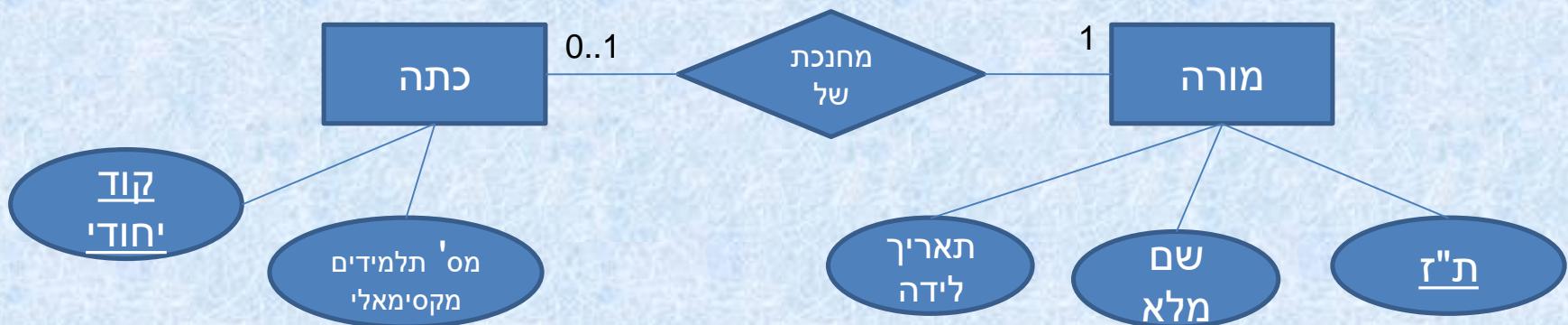


טבלת תרגיל בית

| תאריך הגשה | נושא | מספר תרגיל | מספר קורס |
|------------|----------|------------|-----------|
| 04/05/2016 | מודל ERD | 1 | 1 |
| 14/05/2016 | שאליות | 2 | 1 |
| 03/02/2016 | נגזרות | 1 | 2 |

תרגום תרשימים ER - קשרים

- קשר אחד – לאחד (1:1)



| שם תלמידי מוקם' | קוד ייחודי | קוד ייחודי |
|-----------------|------------|------------|
| 25 | 1 | A |
| 25 | 2 | A |
| 30 | 1 | B |

| ת"ז | שם מלא | תאריך לידיה |
|--------|------------|-------------|
| 121212 | ציפי שביט | 16.04.1980 |
| 131313 | לימור לבנת | 01.05.1979 |
| 141414 | וונדר וומן | 01.12.1975 |
| 151515 | מישל אובמה | 13.10.1962 |
| 161616 | ג'ים קארי | 29.11.1958 |

תרגום תרשימים ER - קשרים

אפשרות אחת

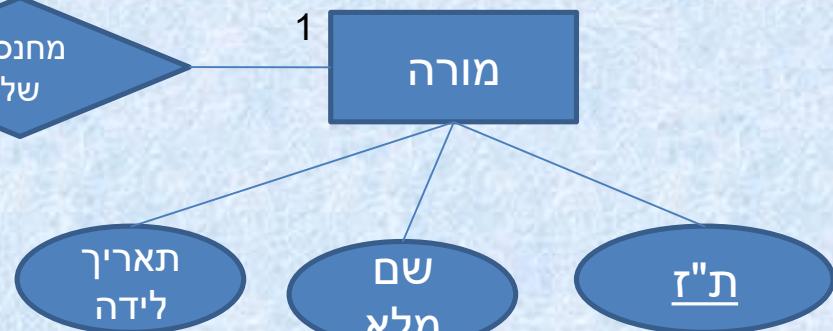


| мо' תלמידי马克' | קוד ייחודי |
|---------------|------------|
| 25 | 1א |
| 25 | 2א |
| 30 | ב1 |

| קוד_ctha | ת"ז_מורה | מישל אובמה |
|----------|----------|------------|
| 1ב | 121212 | ג'ט קארו |
| 1א | 131313 | |
| 2א | 151515 | |

- קשר אחד – לאחד (1:1)

- טבלה נפרדת עבור הקשר



| tarir | name | Tz |
|------------|------------|--------|
| 16.04.1980 | צippy שביט | 121212 |
| 01.05.1979 | ليمור לבנת | 131313 |
| 01.12.1975 | וונדר וומן | 141414 |

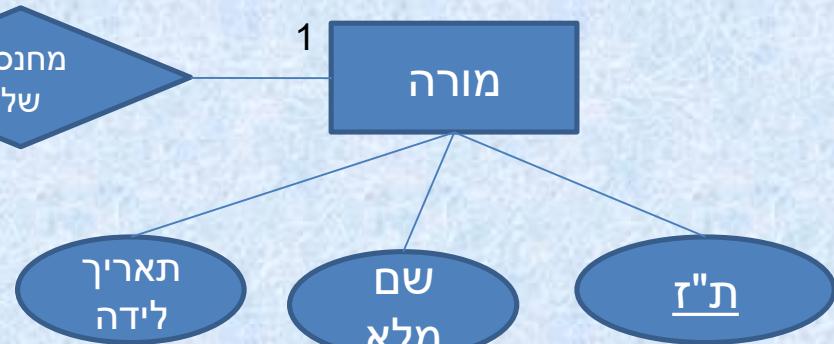
| tarir | name | Tz |
|--------|------------|--------|
| 121212 | מישל אובמה | 151515 |
| 131313 | | |
| 151515 | | |

תרגום תרשימים ER - קשרים

אפשרות שנייה



- קשר אחד – לאחד (1:1)



| קס' תלמידי מקו' | קוד_ученика |
|-----------------|-------------|
| 25 | א1 |
| 25 | א2 |
| 30 | ב1 |

| ת"ז | שם_млаа | tarir_lida | קוד_ктха | קוד_ученика |
|--------|------------|------------|----------|-------------|
| 121212 | ציפי شبיט | 16.04.1980 | б1 | |
| 131313 | לימור לבנת | 01.05.1979 | а1 | |
| 141414 | וונדר וומן | 01.12.1975 | | |
| 151515 | מישל אובמה | 13.10.1962 | 2а | |
| 161616 | ג'ים קארי | 29.11.1958 | | |

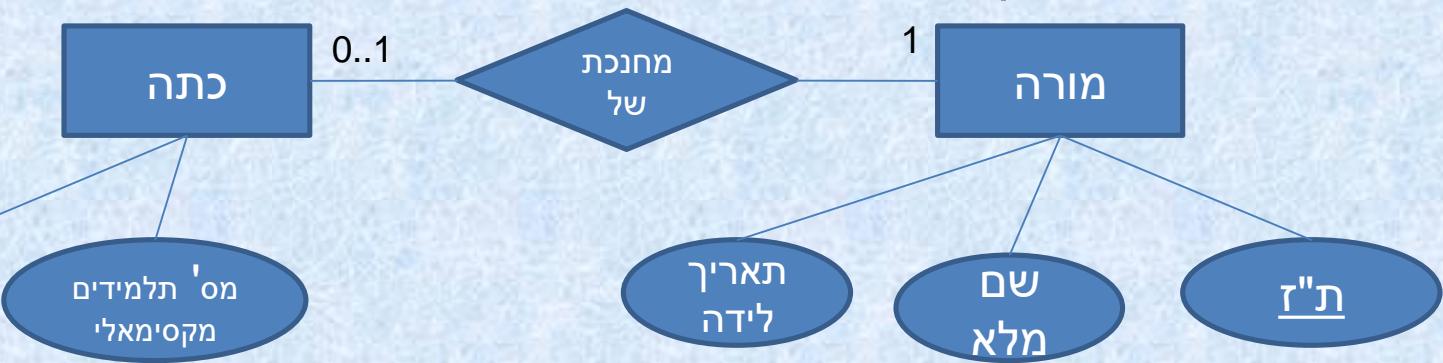
- מה לגבי מורים שאינם משמשים כמחנכים?

תרגום תרשימים ER - קשרים

אפשרות שלישית

- קשר אחד – לאחד (1:1)
- פתרון אופטימאלי

– חוסך הגדרת טבלה נוספת, וממעט בערכי Null



| ת"ז מחנן | מו' תלמידי | מו' תלמידי מקס' | קוד ייחודי |
|----------|------------|-----------------|------------|
| 121212 | 25 | 1 | א |
| 131313 | 25 | 2 | ב |
| 151515 | 30 | | כ |

| ת"ז | שם מלא | תאריך לידה | ת"ז מחנן |
|--------|-------------|------------|----------|
| 121212 | ציפי שביט | 16.04.1980 | 121212 |
| 131313 | לימור לבנת | 01.05.1979 | 131313 |
| 141414 | וונדר ומן | 01.12.1975 | |
| 151515 | מיישל אובמה | 13.10.1962 | 151515 |
| 161616 | ג'ים קארי | 29.11.1958 | |

תרגומים בראשים ER - קשרים

- סיכום קשר אחד – לאחד (1:1)

- אחד הצדדים "נבחר" והפתח עובר לטבלה של קבוצת היחסיות השנייה
- מהוות "מצבע" לטבלה השנייה (נקרא מפתח זר)
- חסכו מקום (אין צורך בטבלה נוספת)
- כאשר הקשר הוא 1:1 לא שלם (טוטאלי) יש להתחשב בשיקולי עלות- תועלת
- אם יותר מ 50% הוו ערכי allN נshall ליצור טבלה נוספת

תרגום תרשימים ER - קשרים

אפשרות אחת

- קשר אחד – לרבים (m:n)
 - טבלה נפרדת



| מספר תלמידי מקס' | קוד ייחודי | ת.ז תלמיד | קוד כיתה | תאריך לידה | שם מלא | ת.ז |
|------------------|------------|-----------|----------|------------|--------------|--------|
| 25 | 1א | | | 16.04.1980 | איציק שפיציק | 232323 |
| 25 | 2א | | | 01.05.1979 | רבקה גמליאל | 343434 |
| 30 | 1ב | | | 01.12.1975 | יוסף הלוּ | 454545 |
| | | 232323 | 1א | | | |
| | | 343434 | 2א | | | |
| | | 454545 | 1א | | | |

תרגום טרשיים ER - קשרים

אפשרות שנייה

- קשר אחד – לרבים (m:n)

- הוספה מפתח "זר"

- מפתח היחיד יעבור לצד של הרבים
- מפתח הרבים יעבור לצד של היחיד

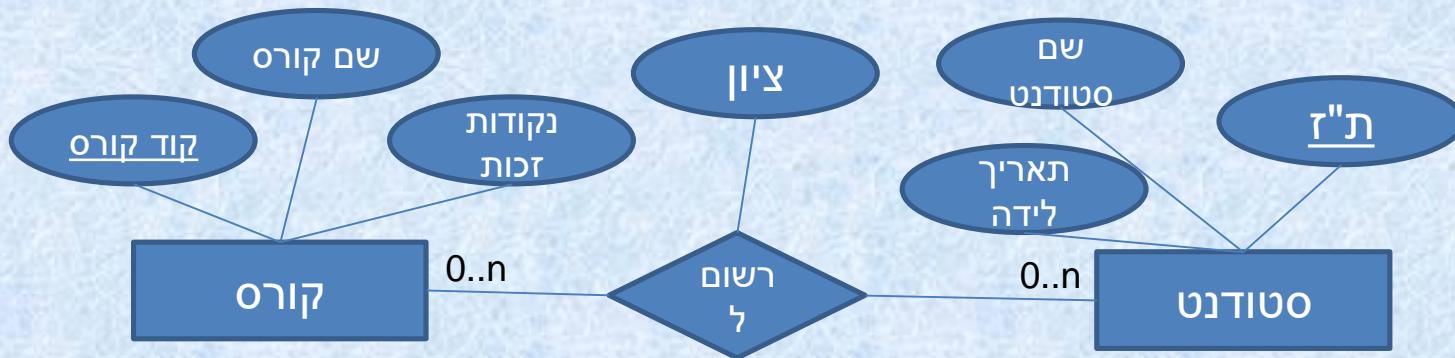


- השדה של קוד כיתה מתפרק למפתח זר בטבלת התלמיד, כלומר הוא מפתח רשומה בטבלתavit כיתות

| ת"ז | שם מלא | תאריך לידה | קוד כיתה | קוד תלמידי ייחודי | מספר תלמידי מקסימלי | קוד ייחודי |
|--------|--------------|------------|----------|-------------------|---------------------|------------|
| 232323 | איציק שפיציק | 16.04.1980 | 1 | 1א | 25 | |
| 343434 | רבקה גמליאל | 01.05.1979 | 2 | 2א | 25 | |
| 454545 | יוסף הלוּי | 01.12.1975 | 1 | ב1 | 30 | |

תרגום טרשיים ER - קשרים

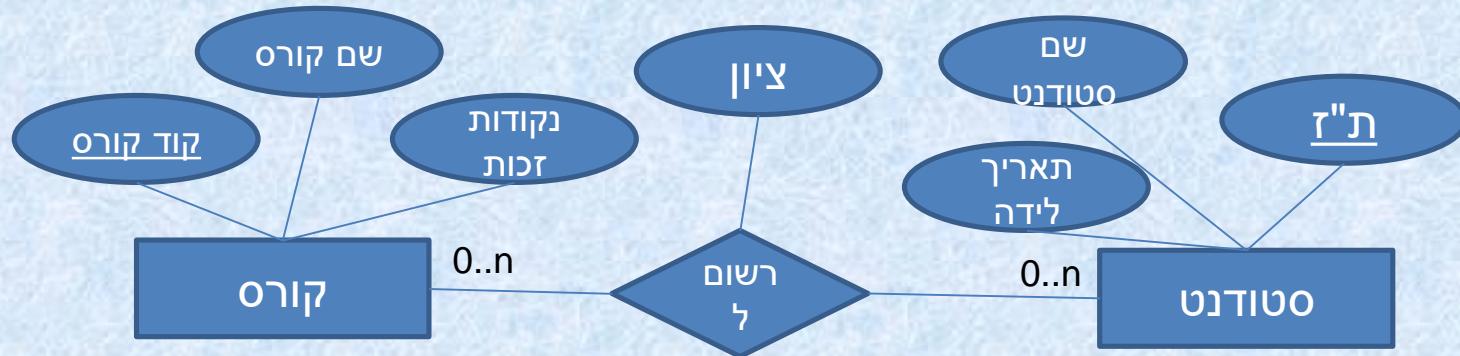
• קשר רבים – רבים (m:n)



- מתורגם כטבלה נפרדת הכוללת:

- מפתחות ראשיים של קבוצת הישויות המעורבות
- המפתח הראשי יורכב מключи ראשיים אלו, וכל אחד מהם בנפרד מהו他自己 מפתח "זר" לטבלה המקורית שלו
- תכונות נוספות של הקשר (בקשרים של 1:1 או 1:m אם יש תוכנה לחבר היא עוברת לטבלה ביחיד עם המפתח)

תרגום תרשימים ER - קשרים



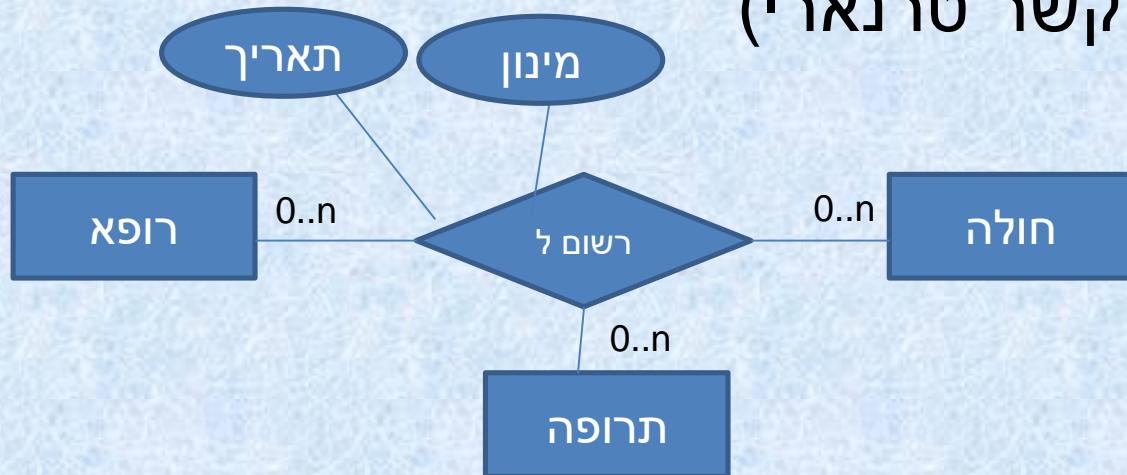
| נקודות צכות | שם קורס | קוד קורס |
|-------------|-------------|----------|
| 4 | מערכות מידע | 2004.13 |
| 3 | הנדסת תכונה | 2004.16 |

| טל | תאריך לידה | שם סטודנט | שם ציבור |
|----------|------------|--------------|----------|
| 12121212 | 16.04.1979 | ישראל ישראלי | |
| 13131313 | 29.01.1983 | דני דין | |

| טל | קוד קורס | טל סטודנט |
|----|----------|-----------|
| 90 | 2004.13 | 12121212 |
| 95 | 2004.16 | 12121212 |
| 89 | 2004.13 | 13131313 |

תרגום תרשימים ER - קשרים

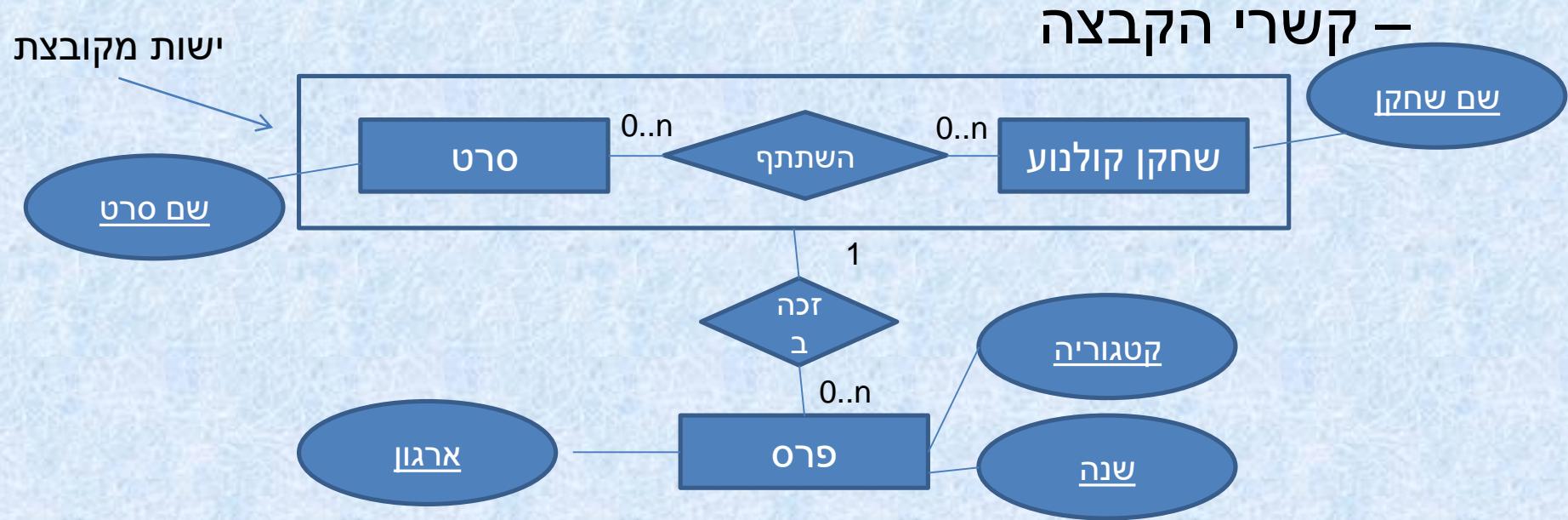
- קשר מרובה (למשל קשר טרנארי)



- בדומה לחבר מרובה גם כאן נדרשת טבלת קשר שהמפתח שלה הוא צירוף 3 המفاتחות של היחסות המעורבות בקשר

| תאריך | מיןון | קוד תרופה | מספר רישויון רפואי | ת"ז חולה |
|------------|-------|-----------|--------------------|----------|
| 01.01.2016 | 90 | AAA | 99999 | 12121212 |
| 01.02.2016 | 95 | AAA | 88888 | 12121212 |
| 31.07.2016 | 89 | BBB | 99999 | 13131313 |

תרגום טרשיים ER – יישיות מקובצות



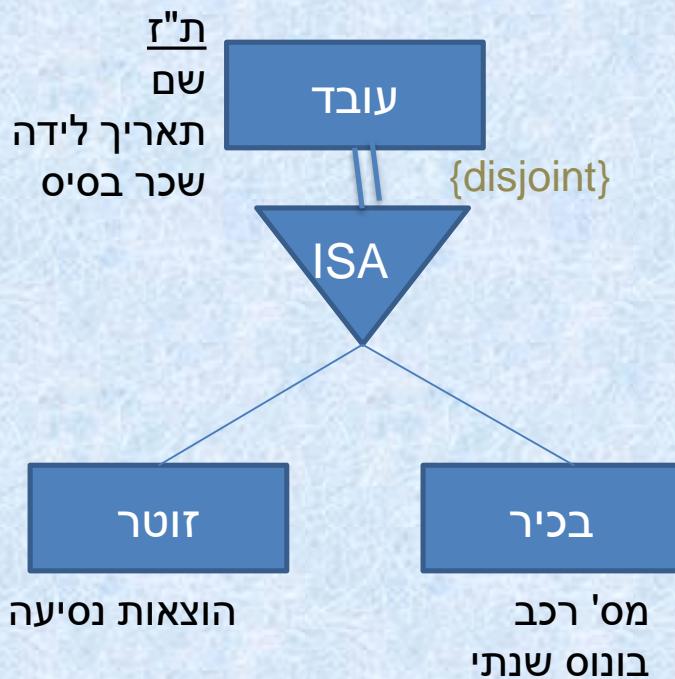
– עבור קשר הקבוצה יש ליצור טבלה נפרדת הכוללת

- מפתחות ראשיים של קבוצת היישיות המעורבות בהקבוצה
- המפתח הראשי יורכב ממפתחות ראשיים אלו
- תכונות נוספות של קשר ההקבוצה (אם קיימות)

תרגומים תרשיים ER - קשרים

- מתרגומים קשר הקבוצה בדוגמה נקלט:
 - טבלת שחקן קולנוע (בעל מפתח ראשי: שם שחקן)
 - טבלת סרט(בעל מפתח ראשי: שם סרט)
 - טבלת פרס (בעל מפתח ראשי: argon, קטגוריה, שנה)
 - טבלת השתף (בעל מפתח ראשי: שם שחקן, שם סרט)
- נתיחה לקובוצת ישות המקובצת כקובוצת ישות לכל דבר
 - מפתח "ישות המקובצת" הנו שילוב של מפתחות הטבלה הנוצרת מקשר הקבוצה
- תרגום של קשרי הקבוצה עם קבוצת ישות אחרות יהיה עפ"י החוקים שנלמדו עד כה (יחס אחד לרבים)
 - טבלת פרס (פתח ראשי: argon, קטגוריה, שנה, שם שחקן, שם סרט)

תרגום טרשים ER - קשרי הכללה והתמחות



- יש להתחשב בשיקולי עלות – תועלת

– כללי אכבע מקובלים:

- עבור קשרים מסווג disjoint + total

אם אין לשוט האב קשרים נוספים:

- צור טבלה לקבוצת הישיות בرمמה הנמוכה בלבד. טבלאות אלו יכילו את התכונות הנורשות מהרמה הגבוהה וכן את התכונות הייחודיות לקבוצת הישיות בرمמה הנמוכה

- מפתח כל טבלה יהיה המפתח של קבוצת הישיות בرمמה הגבוהה

אם יש לשוט האב קשרים נוספים:

- צור טבלה לקבוצת הישיות בرمמה הגבוהה – טבלה זו תכיל את כל התכונות של קבוצת הישיות בرمמה הגבוהה

- צור טבלה לכל קבוצת ישיות בرمמה הנמוכה – טבלה זו תכלול את התכונות המאפיינות באופן ייחודי רק קבוצת ישיות זו ואת המפתח הראשי של קבוצת הישיות בرمמה הגבוהה

תרגום תרשימים ER - קשרי הכללה והתמחות

- עובד זוטר

| ת"ז | שם | תאריך לידה | שכר בסיס | הוצאות נסיעה | מו' רכב |
|----------|--------------|------------|----------|--------------|---------|
| 12121212 | ישראל ישראלי | 16.04.1979 | 10,000 ₪ | 500 ₪ | |
| 13131313 | דני דין | 29.01.1983 | 12,000 ₪ | 60 ₪ | |

- עובד בכיר

| ת"ז | שם | תאריך לידה | שכר בסיס | בונוס שנתי | מו' רכב |
|----------|-------------|------------|----------|------------|----------|
| 14141414 | ברוך ג'מילי | 26.06.1969 | 50,000 ₪ | 100,000 ₪ | 12-34-56 |
| 15151515 | אלן אליהו | 28.05.1951 | 70,000 ₪ | 120,000 ₪ | 34-56-78 |

תרגום תרשימים ER - קשרי הכללה והתמחות

• כללי אצבע מקובלים:

– עבור קשרים מסוג total + overlapping

- צור טבלה לקבוצת הישויות ברמה הגבוהה –

טבלה זו תכיל את כל התכונות של קבוצת הישויות ברמה הגבוהה

- צור טבלה לכל קבוצת ישות ברמה הנמוכה –

טבלה זו תכלול את התכונות המאפיינות באופן ייחודי רק קבוצת ישות זו ואת המפתח הראשי של קבוצת הישויות ברמה הגבוהה

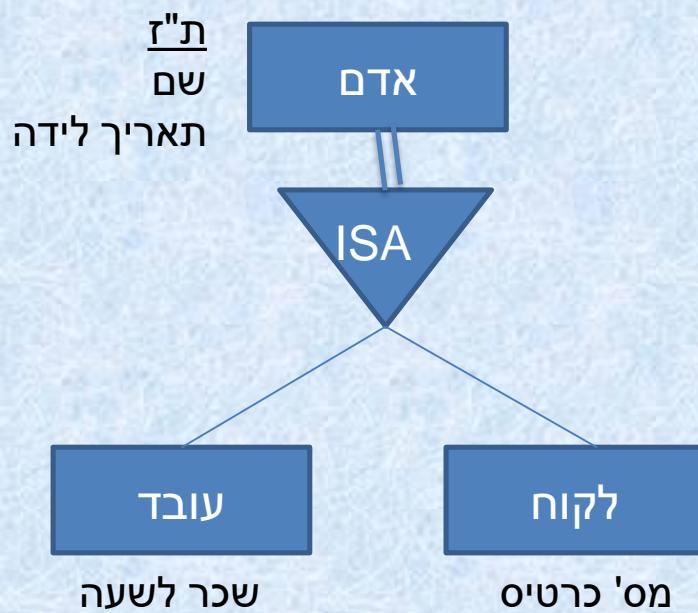
- יש להתחשב בנתון, כמה רשומות משותפות

יכולות להיות. אם מספר הרשומות קטן עדיף

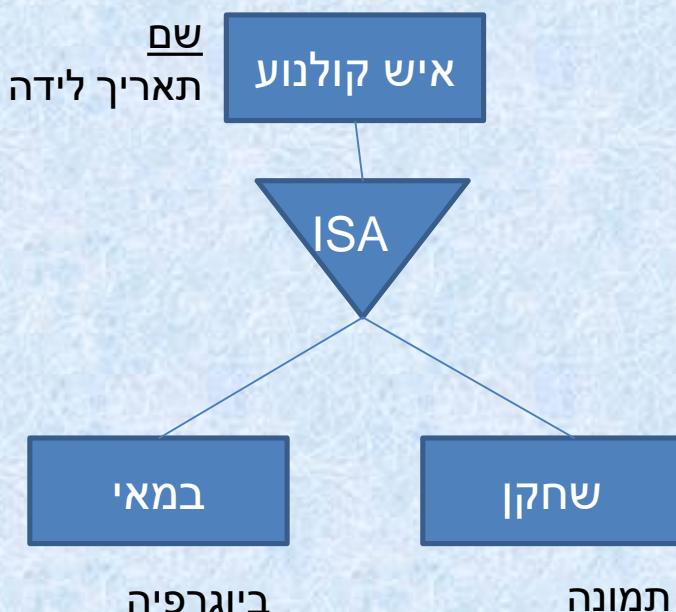
לשמר נתונים כפולים בשתי הטבלאות ולא ליצור את טבלת האב

- חסרון: כדי לשלוフ מידע צריך לגשת למספר

טבלאות ולחדר נתונים



תרגום טרשים ER - קשרי הכללה והתמחות



- עבור קשרים מסוג **partial**
 - **בכל מקרה** יש ליצור טבלה לקבוצת הישיות בرمמה הגבוהה עם התכונות המוגדרות
- יש לשמר מידע על הרמה הגבוהה כי לא כל הישיות בהכרח שייכות גם לرمמות נמוכות. לרוב נוסף גם שדה type לסוג הישות
- **partial + overlapping** –
 - הטבלאות בرمמה הנמוכה יכללו רק את התכונות היחידות ואת המפתח
- **partial + disjoint** –
 - כנ"ל, או שהטבלאות בرمמה הנמוכה יכללו את כל תכונות
- חסרון: כדי לשלווף מידע צרייר לגשת למספר טבלאות ולאחד נתונים

תרגום תרשימים ER - קשרי הכללה והתמחות

| סוג | תאריך לידה | שם |
|-----------|------------|---------------|
| במאי | 19.04.1949 | מנחם גולן |
| שחקן | 29.06.1939 | גילה אלמגור |
| צלם | 26.08.1979 | אליהו גוטמן |
| שחקן/במאי | 12.11.1937 | קלינט איסטווד |

• **שחקן קולנוע**

| תמונה | שם |
|--------------------|---------------|
| Gila_almagor.jpg | גילה אלמגור |
| Clint_eastwood.bmp | קלינט איסטווד |

• **שחקן**

| ביוגרפיה | שם |
|---|---------------|
| אחד ממכירי ומראשמי תעשיית הקולנוע הישראלי'... | מנחם גולן |
| איסטווד ידוע בעיקר בזכות תפקידיו כאיש קשה ... | קלינט איסטווד |

• **במאי**

תרגום תרשימים ER - קשרי הכללה והתמחות

- עוד פתרון אפשרי למיפוי היררכיה:
 - טבלה אחת עבור כל היררכיה שתכיל את כל השדות של כל קבוצות הישיות, ושדה נוסף המגדיר "סוג"
 - יתרונות:
 - מיפוי פשוט
 - טבלה אחת - אין צורך לצרף נתונים ממספר טבלאות
- חסרון
 - כל רומה כוללת שדות רבים מרובים (השדות שלא לבנתיות לסוג הישות)
 - כפיהן חוקים עסקיים הופכת להיות מורכבת
 - נניח שדה שכר עבור עובד זוטר הוא עד 10,000 ועבור בכיר לא מוגבל,
איך נגדיר את האילוץ על שדה השכר?
- על אף פשטותו לרוב לא מומלץ להשתמש במיפוי זה

נרטול טבלאות

- **כיצד נודא שההתוצאה שקיבלנו "טובה"?**
 - פשטות וmobונות של הטבלאות
 - אוסף השדות המרכיב טבלה צריך לתאר ישות אחת/ קשר אחד
 - אין היגיון שנתיוני סטודנטים, קורסים ומרצים ימצאו בטבלה אחת
 - מניעת כפילויות וביעיות עדכון
 - נתוניים הנשמרים ביוטר ממקום אחד גורמים לבזבוז שטח אחסון
 - כפילויות גורמות לביעיות עדכון ועקבויות (כתובות הלקוח בטבלת ההזדמנויות שונה מזו שבטבלת הלקוח. מהי הכתובת הנכונה?)
 - מניעת ערכיים ריקיים בשדות
 - גורמים לבזבוז בשטח האחסון
 - עלולים להקשות על הבנת הטבלה
 - מקרים על חיפושים והגדרת אינדקסיים

צורות נרמול

- הגדרה: תלות פונקציונאלית בין שדה A לשדה B ($A \leftarrow B$)
משמעותו, שערך של שדה A קובע חד ערכית את ערכו של
השדה B
 - מספר ת"ז \leftarrow שם הסטודנט
- אם ידוע מספר ת"ז ניתן לברר מהו שם הסטודנט. האם גם ההיפך נכון?
 - מספר ת"ז, קודקורס \leftarrow ציון
- האם ניתן לברר ציון ללא ידוע של שני השדות הללו?

נrmol מדרגה ראשונה

- **כל נrmol מסדר ראשון (1NF) :** בסיס נתונים הוא ב 1NF אם תחומי כל השדות של כל הטבלאות הם אוטומטיים
 - תחום הנו תחום אוטומי אם כל איבר בו שמותיים אינם אליו כל יחידה אחת, אין לו ניתן לפירוק לתתי יחידות
 - שדות מורכבים ושדות מרובי ערכים הן אינם אוטומטיות
- **בכדי להשיג נrmol 1NF**
 - שדה המורכב ממספר שדות יפוצל לשדות המכילים כל אחד שדה בודד (שיטת התוכנה המורכבת)
 - עבור שדה המורכב מערכיים חוזרים (תמונה מרובעת ערכים) ניצור טבלה חדשה שתכלול את מפתח הטבלה המקורית + תוכנה לשדה החזיר

נរמול מדרגה ראשונה

- דוגמא לטבלה שאינה מנורמלת בסדר ראשון:
 - מספר סטודנט, שם סטודנט (פרט + משפחה), מספרי טלפון
- מה הבעה?
 - שם סטודנט אינו אוטומי – ניתן לפירוק
 - מספרי טלפון הם רב-מושפעים
- נרמול הטבלה יגרום לפיצול לשתי טבלאות:
 - טבלה סטודנטים: מספר סטודנט, שם פרטי, שם משפחה
 - טלפת מספרי טלפון של סטודנט: מספר סטודנט, מספר טלפון

נrmol מדרגה שנייה

- **כל נrmol מסדר שני (2NF)** : בסיס נתונים הוא ב 2NF אם כל הテーブלאות ב-1NF וכל שדה בטבלה שאינו ב מפתח תלוי בכל שדות המפתח
 - אם לא כל השדות בטבלה תלויים בכל המפתח, הדבר מהוות בעיה בעדכון רשומות
 - אם יש במפתח רק שדה ייחיד אזי הטבלה מנורמלת מדרגה שנייה
- **בכדי להציג נrmol 2NF**
 - עבור שדה שאינו תלוי בכל שדות המפתח ניתן טבלה חדשה שתכלול את השדות מהמפתח המקורי בהן תלוי השדה + תוכנה לשדה החזר

נរמול מדרגה שנייה

- דוגמא לטבלה שאינה מנורמלת בסדר שני:
 - מספר סטודנט, מספרקורס, שם קורס, ציון
- מה הבעה?
 - שם קורס איננו תלוי בכל המפתח, אלא רק במספר קורס
- נרמול הטבלה יגרום לפיצול לשתי טבלאות:
 - טבלת סטודנטים בקורס: מספר סטודנט, מספרקורס, ציון
 - טבלת קורסים: מספרקורס, שם קורס

נrmול מדרגה שלישית

- **כלל נrmול מסדר שלישי (3NF)** : בסיס נתונים הוא ב F 3NF אם כל הテーブאות הן ב-2NF וכל שדה בטבלה שאינו ב מפתח תליי רק ב מפתח.
 - אם לא כל השדות בטבלה תלויים בכל המפתח, הדבר מהוות בעיה בעדכן רשותות
- **בכדי להציג נrmול 3NF**
 - עבור שדה שאינו תלוי רק בשדות המפתח ניתן טבלה חדשה שתכלול את השדות בהן תלוי השדה שאינו ב מפתח + תוכנה לשדה התלוי

נរמול מדרגה שלישית

- דוגמא לטבלה שאינה מנורמלת בסדר שלישי:
 - מספר עובד, שם עובד, משרה, שכר
- מה הבעה?
 - לעובדים שונים באותה משרה יש שכר זהה – لكن השכר תלוי במשרה, כלומר בשדה שאינו במפתח
- נרמול הטבלה יגרום לפיצול לשתי טבלאות:
 - טבלת עובדים: מספר עובד , שם עובד , משרה
 - טבלת משרות: משרה , שכר

כללי נרמול ו-ERD

- המירה תרשימים ER למודל טבלאי עפ"י הכללים שנלמדו
קודם לכן מבטיח מודל מנורמל מדרגה 3!
- קיימות רמות נרמול נוספות, אך נרמול מדרגה שלישית
נחשב לרמה מספקת

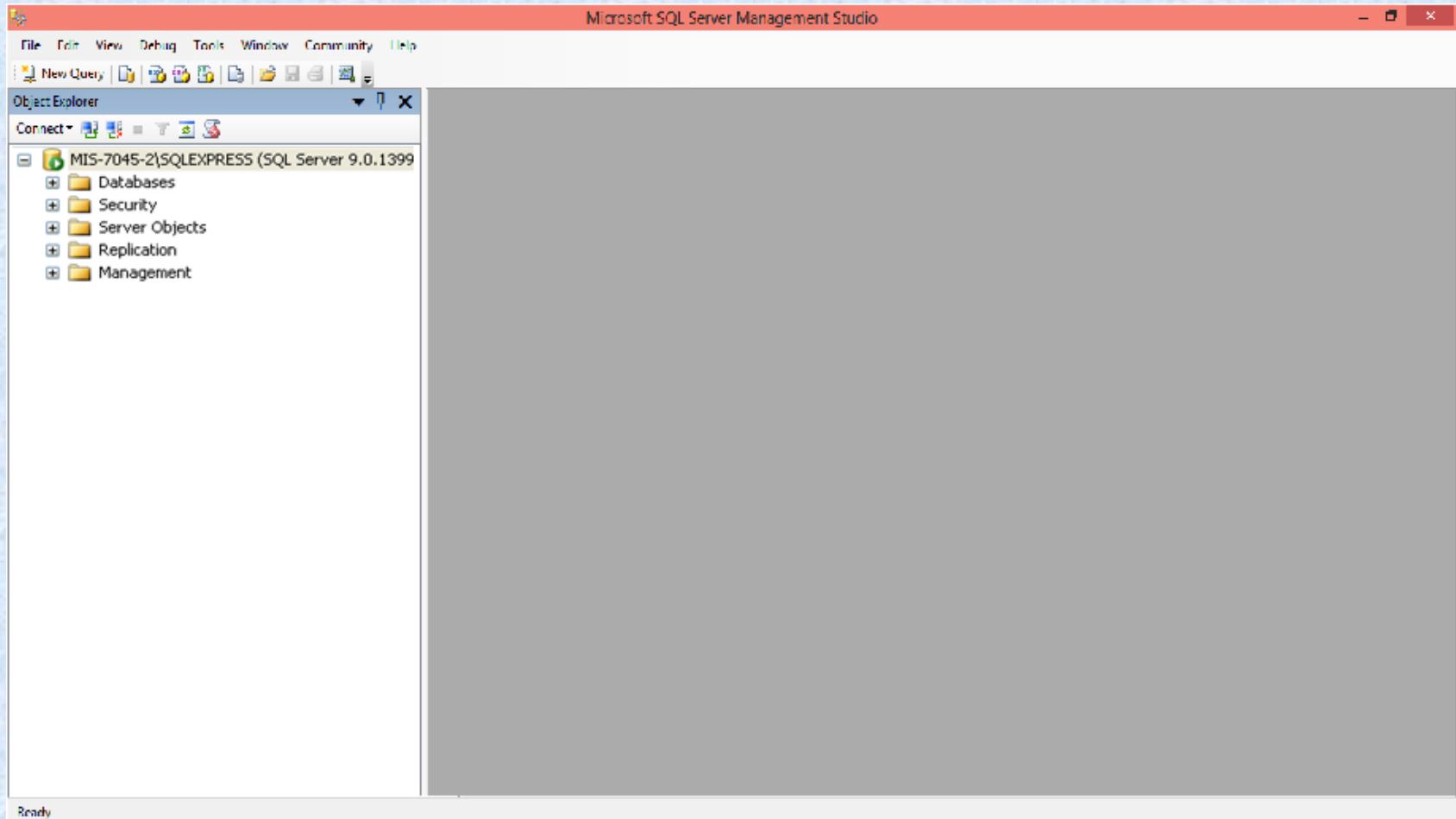
טבלאות מנורמלות 3NF עברו ה-ERD של :matchMe

| שם טבלה | שדות בטבלה |
|---|--|
| עיר מגורים tblCity | <u>שם עיר</u> <u>cityName</u> |
| סוג קשר tblRelationType | <u>שם קשר</u> <u>relationName</u> |
| משתמש tblUser | <u>כתובת מייל</u> , שם פרטי, שם משפחה, תאריך לידה, מגדר (זכר/נקבה/אחר), תיאור, שם עיר (מ.ז. לעיר) <u>mail</u> , <u>firstName</u> , <u>lastName</u> , <u>dateOfBirth</u> , <u>gender(F, M, O)</u> , <u>description</u> , <u>cityName</u> (<u>fk_tblCity</u>) |
| מעוניין ב (משתמש- סוג קשר) tblInterestedIn | <u>כתובת מייל</u> (מ.ז. משתמש), <u>שם קשר</u> (מ.ז. קשר), <u>מין</u> (זכר/ נקבה/ שניהם) <u>email(fk_tblUser)</u> , <u>relationName (fk_tblRelationType)</u> , <u>gender(F, M, B)</u> |
| לא מעוניין ב tblNotInterestedIn | <u>כתובת מייל</u> (מ.ז. משתמש), <u>כתובת מייל דחוי</u> (מ.ז. משתמש) <u>Email(fk_tblUser)</u> , <u>emailDeferred (fk_tblUser)</u> * <u>כתובת המail</u> בעמודה הראשונה תציג את הדוחה והכתובת המail בעמודה השנייה את הנדחה |
| תמונה tblPicture | <u>מספר סידורי</u> , <u>נתיב</u> <u>serialNum</u> , <u>path</u> , <u>email(fk_tblUser)</u> |
| תמונה me tblMatchMePicture | <u>מספר סידורי</u> (מ.ז. <u>תמונה</u>), <u>תאריך צילום</u> , <u>תיאור</u> <u>serialNum(fk_tblPicture)</u> , <u>photoDate</u> , <u>description</u> |

טבלאות מנורמלות 3NF עברו ה-ERD

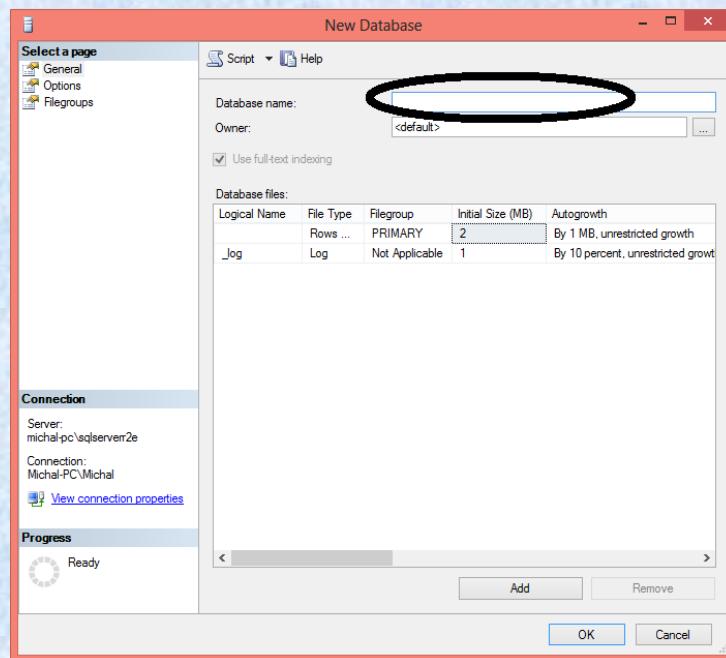
| שם טבלה | שדות בטבלה |
|-----------------|--|
| tblApplication | שם אפליקציה, תיאור, תאריך הקמה appName, description, establishingDate |
| tblRegisteredTo | כתובת מייל (מ.ז. משתמש), שם אפליקציה (מ.ז. אפליקציה), תאריך הרשמה, כינוי Email(fk_tblUser), appName(fk_tblApplication), registrationDate, nickName |
| tblQuestion | שם אפליקציה (מ.ז. אפליקציה), מספר שאלה, תוכן, האם חובה? תשובה 1, תשובה 2, תשובה 3, תשובה 4 appName (fk_tblApplication), questionNo, content, IsMandatory?, answer1, answer2, answer3, answer4 |
| tblAnsweredOn | כתובת מייל (מ.ז. משתמש), שם אפליקציה, מספר שאלה (יחד מ.ז. לשאלה), תשובה (1-4), תאריך מענה email(fk_tblUser), appName, questionNo (fk_tblQuestion), answer (1-4), answerDate |
| tblListOf | כתובת מייל (מ.ז. משתמש), כתובת מייל של המותאם, שם אפליקציה (יחד מ.ז. רשום ל), תאריך הוספה, תאריך פניה email(fk_tblUser), suitableEmail, appName (fk_tblRegisteredTo), insertionDate, applyingDate |

SQL Server 2012/2014/2017



יצירת DB חדש

יצירת Database חדש תעשה ע"י לחיצה ימנית בעבר על תיקיית Database ובחירה באפשרות '...new Database', לאחר בחירת אפשרות זו יופיעחלון הבא בו יש להגדיר שם ל-DB החדש וללחוץ על OK לאישור:



יבוא DB קיים (שאינו קיים ב-SQL הנוכחי)

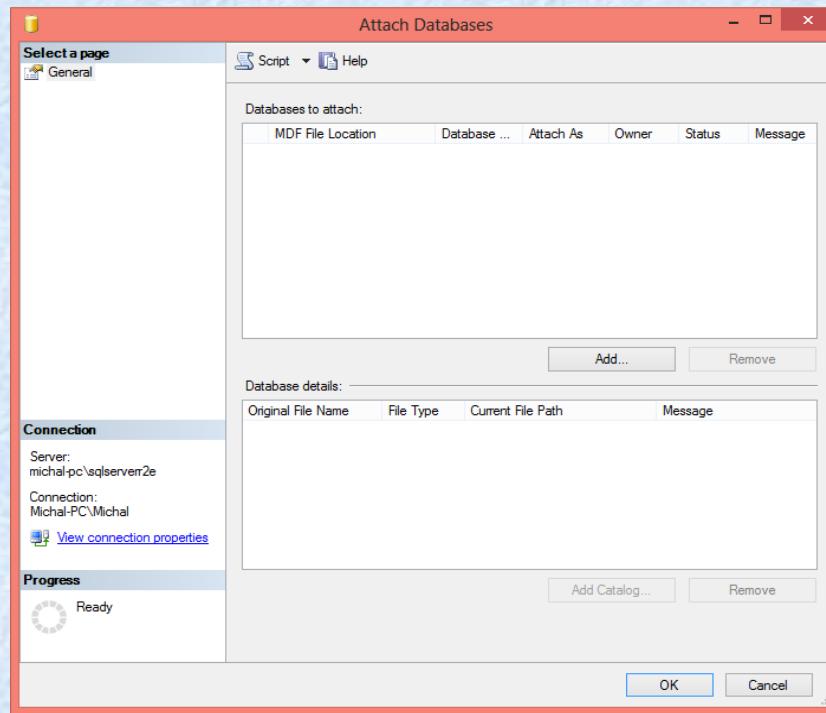
פתחת Database קים (שנוצר במחשב אחר, בגרסת SQL Server אחרת, וכו') תעשה ע"י לחיצה ימנית בעבר Database על תיקייה ובחירה באפשרות 'attach'.

קובץ ה-DB שאנו מעוניינים לפתח צריך לשבת בתיקייה של ה-SQL הנמצאת לרוח בנתיב הבא:

C:\Program Files\Microsoft SQLServer\MSSQL14.SQLEXPRESS\MSSQL\DATA

יבוא DB קיים (שאינו קיים ב-SQL הנוכחי)

לאחר בחירת אפשרות זו יופיעחלון הבא ובו יש להגדיר מיקום קובץ ה- MDF (מtower...) וללחוץ על OK לאישור:



אפיון שפת SQL

שפת SQL הינה שפה הצהרתית המבוססת על אלגברה טבלאית, שעיקרה עבודה עם נתונים במודל הטבלאי הרלציוני. למרות שמה השפה איננה רק שפת שאלות, אלא למעשה שפת הנתונים המקיפה ביותר, המאפשרת קשת רחבה של עבודה מול בסיס נתונים. בשפה זו ניתן למצוא על כן את הפעולות הבאות:

- ייצירת טבלאות סטטיות
- ייצירת אינדקסים
- ייצירת טבלאות דינמיות מדומות – view
- הגדרת אילוצים עסקיים - Assertions
- ייצירת טריגרים

אפיון שפת SQL

SQL מאפשרת קריית נתונים בדרכים מגוונות, הכוללות פעולות בין טבלאות (כמו צירוף נתונים טבלאות ופעולות של איחוד וחיתוך בין נתונים טבלאות) וכליה במניפולציה על נתונים כמו הוספה רשומות, עדכון ומחיקתן, ואף אבטחת נתונים וניהול תנועות.

שפה DDL - data description language

שפה DDL הינה שפה להגדרת נתוני המשמשת לטיפול במבנה הנתונים (הוספת ומחיקת טבלאות, עמודות או אינדקסים).

אפשרויות לביצוע בשפה DDL:

- **משפט create** – יוצר פריט מבנה חדש – טבלה, טבלה מדומה, עמודה וכו'.
- **משפט alter** – משנה את תכונותיו של פריט קיים.
- **משפט drop** – מוחק פריט קיים.

DDL - יצרת טבלה חדשה

מבנה יצרת שאילתת create:

```
create table tablename  
( column1name datatype1,  
,...  
columnXname datatypeX);
```

מבנה יצרת שאילתת create תוך הגדרת אילוצים:

```
create table tablename  
( column1name datatype1 constraint1,  
,...  
columnXname datatypeX constraintX);
```

יצירת טבלאות ב-SQL

על מנת ליצור טבלאות ב-SQL علينا לבחור באפשרות `create query`. בחלק שנפתח ניתן לכתוב כל סוג שאלתה (יצירת טבלאות ועדרון, הכנסת נתונים, שאלות שליפה וכו').

לאחר כתיבת הפקודה יש להריצה דרך הסימון (סימון אדום מצד שמאל).

במידה וקייםת שגיאה סינטקטית בפקודה, היא לא תרוץ ובחלק התיכון של העמוד תופיע היקן הבעה.

במידה והפקודה רצה בצורה תקינה, נקבל בחלק התיכון של המסר הודעה מתאימה.

יצירת טבלאות ב-**SQL** – **סוגי טיפוסים נפוצים**

| שם טיפוס | תיאור |
|---------------|--|
| Bit | מאתון 0, 1 או null |
| Tinyint | מספר שלם בטווח של עד 256 (תופס 1 בתים) |
| Smallint | מספר שלם בטווח של -32,768 - 32,768 ועד 32,768 (תופס 2 בתים) |
| Int | מספר שלם בטווח 2,147,483,648 -2,147,483,648 ועד 2,147,483,648 (תופס 4 בתים) |
| Bigint | מספר שלם בטווח 9,223,372,036,854,775,808 - 9,223,372,036,854,775,808 (תופס 8 בתים) |
| Char(size) | תווים. יש לכתוב את כמות התווים המקסימלית שתוכנס. כל המיקומות ישמרו בזיכרון גם אם לא הוכנסו ערכים. |
| Varchar(size) | תווים. יש לכתוב את כמות התווים המקסימלית שתוכנס. ישמרו בזיכרון רק התווים שמשתמשים בהם בפועל. על מנת לאפשר כתיבה בעברית, נכתב varchar |
| Datetime | אחסון תאריך ושעה |
| Date | אחסון תאריך |
| Time | אחסון שעה |
| float | מספר ממשי (עם נקודה עשרונית). |

אילוצים ב-SQL

- **Primary key** – שדה המשמש **לזהות ייחודי** ומאלץ כי לא יהיו שתי רשומות עם אותו ערך בשדה זה.
- **Not null** – אוסר על השדה להחזיק ערך ריק.
- **Unique** – מניעת ערכים כפולים בשדה. ניתן להגדיר שדות אשר בכל אחת מהרשומות הערכים בה יהיו שונים (לא תלות במפתח הראשי).
- **Foreign key** – מפתח זר הוא שדה בטבלה המתיחס לשדה בטבלה אחרת. שדה זה משמש כהפנייה לשדה המוגדר **מפתח ראשי** בטבלה אחרת.
- **Check** – מאפשר לבצע בדיקות נתוניות המזננים לטבלה מסוימת (למשל עבור בדיקת מספרי טלפון, שמות וכו').
- **Default** – השמה של ערך דיפולטיבי אם לא הוכנס ערך אחר לשדה.

אלוצי primary key ב-SQL

הגדרת מפתח ראשי (כאשר המפתח הראשי הוא תכונה אחת בלבד) מתבצע על ידי הוספה של המילה primary אחרי נתינת שמו וסוגו.

```
Create table tbl1(  
    a1 int primary key)
```

כאשר המפתח הראשי מורכב מיותר מרשותה אחת, علينا להגדיר את המפתח הראשי בנפרד. דוגמא:

```
Create table tbl1(  
    a1 int,  
    a2 int,  
    PRIMARY KEY (a1,a2))
```

אלוצי primary key ב-SQL

אם אין לנו מפתח ראשי מובהק של הטבלה וANO
מעוניינים ליצור מספר סידורי רץ בהתאם למפתח
הראשי, צורת ההגדרה של התכונה תהיה:

serialNumber int identity(1,1) primary key

אילוצי foreign key ב-SQL

זהו שדה המהווה הפניה למפתח ראשי בטבלה אחרת. אילוץ זה מאפשר למעשה לבצע את הקשרים בין היחסיות השונות.

בפועל, לא נוכל להכנס ערך לשדה הזר בטבלה בה יש מפתח זר אם ערך זה לא קיים בטבלה המקורית.

```
Create table tbl2(  
    c1 int,  
    C2 int,  
    FOREIGN KEY (c1,c2) REFERENCES  
    tbl1(a1,a2))
```

השדות מהטבלה שלי
שמקבלים את הערכים
מהטבלה המקורית

הטבלה ממנה אנו לוקחים את
המפתחות הראשיים ושםות
המפתחות הראשיים בה

אילוצי foreign key ב-SQL

הቤיטוי **DELETE ON** מציין את אשר علينا לעשות בטבלה בה יש הצבעה כאשר מתבצעת操作 של הרחקה מהטבלה המקורית. שתי אפשרויות לביצוע:

1. ACTION ON – מציין כי הרחקה בטבלה המקושרת לא תבוצע (הערך הדיפולטיבי המתתקבל כאשר לא מתבצעת הגדעה) עד אשר לא נבצע操作 של הרחקה בטבלאות מהן נלקח המידע.
2. CASCADE – מציין כי יחד עם הרחקה של הערך מהטבלה הראשית, כל הרשומות בטבלה שקשורה אליה ימחקו גם כן (כלומר שדותם בהם המפתח הראשי מהטבלה המקורית זהה למפתח הזר).
3. NULL SET – מציין כי יחד עם הרחקה של הערך מהטבלה הראשית, כל הרשומות בטבלה שקשורה אליה יקבלו את הערך null.
4. SET DEFAULT – מציין כי בלבד עם הרחקה של הערך מהטבלה הראשית, כל הרשומות בטבלה שקשורה אליה יקבלו ערך default.

דוגמה ליצירת טבלה מקשרת ב-SQL

```
create table tblMakes
```

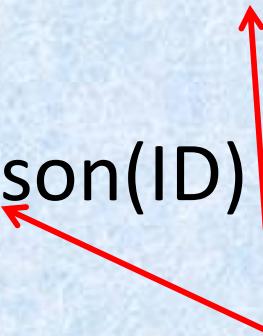
```
(OrderNo int references tblCarOrder(OrderNO)
```

```
on update cascade,
```

```
ID char(8) references tblPerson(ID) on update  
cascade,
```

```
primary key (OrderNo, ID));
```

המפתח הראשי מורכב
משני שדות של הטבלה



שדות אשר מהווים מפתח
ראשי בטבלה אחרת –
דוגמה לטבלה מקשרת

דוגמא ליצירת טבלה מקשרת של קשר טרינארי ב-SQL

קשר רבים-רבים-רבים:

```
create table tblMakes
```

```
(taskID int references tbltask(ID),  
projectID int references tblPoject(ID),  
workerID int references tblWorker(ID),  
startWorkeingDate datetime,  
endWorkingDate datetime,  
primary key (taskID, projectID, workerID));
```

שדות אשר מהווים
מפתח ראשי בטבלה
אחרת

ה מפתח הראשי מורכב משלושה שדות
שהינם מפתחות ראשיות בטבלה המקורית

דוגמא ליצירת טבלה מקשרת של קשר טרינארי ב-SQL

קשר ייחד-רבים-רבים:

create table tblMakes

(taskID int references tbltask(ID),

projectID int references tblPoject(ID),

workerID int references tblWorker(ID),

startWorkingDate datetime,

endWorkingDate datetime,

primary key (taskID, projectID));

שדות אשר מהווים
מפתח ראשי בטבלה
אחרת

המפתח הראשי מורכב משני שדות בלבד, כלומר רק הם המזהים
של הטבלה והשדה הנוסף (של העובד) הינו מפתח זר בלבד

דוגמא ליצירת קשר של אגרגציה ב-SQL

יש ליצור טבלה עבור מי שנמצא בתוך האגרגציה וזו לחבר את הטבלה הזאת שיצרנו עם הטבלה שמחוץ לאגרגציה.
יצירת טבלת משתתף ב...:

```
create table tblParticipateIn  
(movieID int references tblMovie(ID),  
playerID int references tblPlayer(ID),  
character varchar(20),  
primary key (movieID, playerID));
```

שדות אשר מהווים מפתח ראשי בטבלה אחרת – הטבלה המקשרת בתוך האגרגציה

```
create table tblRemembers  
(movieID int,  
playerID int,  
(movieID,playerID) references tblParticipateIn(movieID,playerID)  
watcherID int references tblWatcers(ID),  
primary key (movieID, playerID, watcherID));
```

קריאה לערכי מתווך טבלת הקשר שבתוך האגרגציה

המפתח הראשי מורכב מהשדות שבטבלת הקשר ומהטבלה החדשה

דוגמא ליצירת טבלה עבור ישות חלשה

```
create table tblCarJob  
(JobName varchar(30) references tblJob (Name)  
on delete cascade on update cascade,  
CarNoWithinJob int not null,  
LicenseType char not null,  
primary key (JobName, CarNoWithinJob));
```

עלינו ליבא את המפתח
הראשי של הישות
החזקת

המפתח הראשי מורכב
מהמפתח של הישות החזקה
והמזזה של הישות החלשה

דוגמא ליצירת ישות אב וישות ילד יורש מושג total disjoint

אופציה א':

```
create table tblAnimal  
(ID int primary key,  
Name varchar(10) not null);
```

```
Create table tblCat  
(ID int primary key references tblAnimal(ID),  
Type varchar(10));
```

```
Create table tblDog  
(ID int primary key references tblAnimal(ID),  
Volume double);
```

המפתח הראשי כולל
מפתח זר המתקבל
מטבלה אחת יחידה

דוגמא ליצירת ישות אב וישות ילד יורש מושג total disjoint

אפשרה ב:

```
Create table tblCat  
(ID int primary key,  
Name varchar(10) not null,  
Type varchar(10));
```

```
Create table tblDog  
(ID int primary key,  
Name varchar(10) not null,  
Volume double);
```

אילוצי check ב-SQL

אילוץ משמש לאכיפת שלמות הנתונים על ידי הגבלת ערכים מותרים בשדות.

לכל שדה ניתן להוסיף מספר אילוצי בדיקה.

האילוץ נכתב כביטוי בוליאני יוכל לכלול מספר ביטויים לוגיים המקשורים על ידי אופרטורים של AND, OR ו-NOT.

התנאי נבדק בעת הכנסת הנתונים ואם לא עומד בהם לא יכנס אותם.

AILOZI check ב-SQL

דוגמאות:

- בדיקה כי שדה מין מקבל ארך ורק גבר או אישה. נרצה לשמור שמות קוד מתאימים: M עבור גברים ו-F עבור נשים:

Gender char(1) CHECK ((Gender = 'F') OR (Gender = 'M'))

- בדיקה כי מספר הזמנה מכיל בדיק 9 ספרות:
orderNum char(9)
CHECK (orderNum like replicate('[0-9]',9))
- בדיקת כי הליקוח בן יותר מ-18:

birthdate date

check(datediff(year,birthdate,getdate())>=18

AILOZI check ב-SQL

דוגמאות:

- בדיקה כי שדה מין מקבל ארך ורק גבר או אישה. נרצה לשמור שמות קוד מתאימים: M עבור גברים ו-F עבור נשים:

Gender char(1) CHECK ((Gender = 'F') OR (Gender = 'M'))

אפשרה נוספת:

Gender char(1) CHECK (Gender in ('F','M'))

אילוצי check ב-SQL

דוגמאות:

- בדיקה כי מספר הזמנה מכיל בדיק 9 ספרות:

orderNum char(9)

CHECK (orderNum like '[0-9][0-9][0-9][0-9]
[0-9][0-9][0-9][0-9][0-9]')

אפשרה נוספת:

orderNum char(9)

CHECK (orderNum like replicate('[0-9]',9))

אפשרה נוספת:

orderNum char(9)

CHECK (orderNum like '[0-9]+')

AILOZI check ב-SQL

דוגמאות:

אופציה נוספת:

orderNum char(9)

CHECK (orderNum like '[0-9]{9}')

אופציה נוספת:

orderNum char(9)

CHECK (orderNum like '\d{9}')

- בדיקת כי הילקוּן בין יותר מ-18:

birthdate date

check(datediff(year,birthdate,getdate())>=18

יצירת מוד הנתונים ב-SQL ל-matchMe

```
create table city
(
    name nvarchar(20) primary key
)
```

```
create table relationType
(
    name nvarchar(20) primary key
)
```

יצירת מוד הנתונים ב-SQL ל-matchMe

```
create table users
(
    mail varchar(50) primary key check(mail like
        '%@%.%'),
    firstName nvarchar(20) not null,
    surName nvarchar(20) not null,
    birthDate date
        check(datediff(year,birthDate,getdate()) >= 18)
    not null,
    gender nchar(1) check(gender = N'ט' or gender =
        N'נ' or gender = N'x')
    description nvarchar(255),
    cityName nvarchar(20) references city(name)
)
```

יצירת מוד הנתונים ב-SQL ל-matchMe

```
create table userRelation  
(  
    userEmail varchar(50) references users(mail),  
    relationName nvarchar(20)  
        references relationtype(name),  
    gender nchar(1) check(gender in  
        (N'ז', N'נ', N'א')) not null,  
    primary key(userEmail, relationName)  
)
```

יצירת מוד הנתונים ב-SQL ל-matchMe

```
create table Picture
(
    serialNum int identity(3,10) primary key,
    path nvarchar(255) not null,
    userEmail varchar(50) references Users(mail) not null
)
```

```
create table matchMePic
(
    picNum int references picture(serialNum) primary key,
    picDate date not null,
    description nvarchar(255)
)
```

יצירת מודד הנתונים ב-SQL ל-matchMe

```
create table Application
(
    name nvarchar(20) primary key,
    description nvarchar(255) not null,
    creationDate date
    check(creationDate <= getdate())
    not null
)
```

יצירת מוד הנתונים ב-SQL ל-matchMe

```
create table RegisterdTo  
(  
    userMail varchar(50) references  
    Users(mail),  
    appName nvarchar(20) references  
    Application(name),  
    registrationDate date,  
    nickName nvarchar(20),  
    primary key(usermail,appName),  
    unique(appName,nickName)  
)
```

יצירת מוד הנתונים ב-SQL ל-matchMe

```
create table List
(
    userEmail1 varchar(50) references
    Users(mail),
    userEmail2 varchar(50),
    appName nvarchar(20),
    addDate date not null,
    talkDate date,
    foreign key (userEmail2,appName) references
    RegisterdTo(usermail,appName),
    primary key(userEmail1,userEmail2,appName),
    check(userEmail1 <> userEmail2)
)
```

יצירת מוד הנתונים ב-SQL ל-matchMe

```
create table DoesnLike
(
    userEmail1 varchar(50) references
    Users(mail),
    userEmail2 varchar(50) references
    Users(mail),
    primary key(userEmail1,userEmail2),
    check(userEmail1 != userEmail2)
)
```

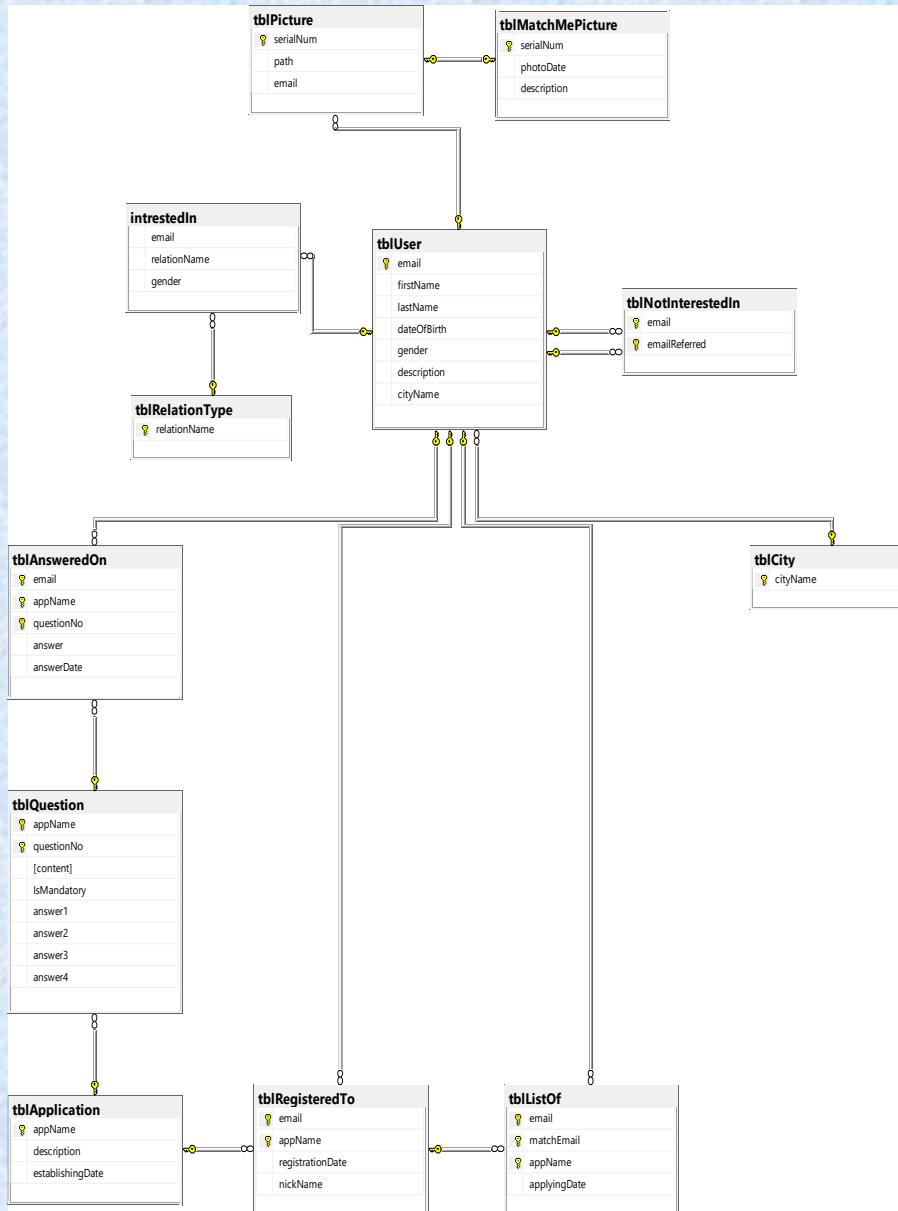
יצירת מוד הנתונים ב-SQL ל-matchMe

```
create table Question  
(  
    appName nvarchar(20) references Application(name) on delete cascade,  
    serialQuestion int,  
    content nvarchar(255) not null,  
    mandatory bit not null,  
    ans1 nvarchar(50) not null,  
    ans2 nvarchar(50) not null,  
    ans3 nvarchar(50) not null,  
    ans4 nvarchar(50) not null,  
    primary key(appName, serialQuestion)  
)
```

יצירת מוד הנתונים ב-SQL ל-matchMe

```
create table Answers
(
    userEmail varchar(50) references Users(mail),
    appName nvarchar(20),
    questionSerial int,
    ansDate date check(ansDate = getdate()) not null,
    ans char(1) check(ans in ('1','2','3','4')) not null,
    ans2 tinyint check(ans2 > 0 and ans2 <= 4) not null,
    ans3 tinyint check(ans3 between 1 and 4) not null,
    foreign key(appName,questionSerial) references Question(appName,serialQuestion),
    primary key(userEmail,appName,questionSerial)
)
```

קשרי הלקוח ב-SQL Server



DDL – עדכון טבלה קיימת

על מנת לשנות מבנה של טבלה כלשהי שיצרנו, علينا להשתמש במשפט `Alter table`.

פעולות שניתן לבצע בעזרת פקודה `Alter`:

1. הוספת שדה חדש לטבלה.
2. שינוי סוג המשתנה.
3. מחיקה של שדה כלשהו.
4. ייצור אילוצים על הטבלה.
5. הסרת אילוצים על הטבלה

DDL – עדכון טבלה קיימת

תחביר של הפקודה:

Alter table tableName

 add columnName dataType [constraint]

 drop Column columnName

 add Constraint constraintName ...

 drop constraint constraintName

 alter column columnName newDataType

או

DDL – עדכון טבלה קיימת

דוגמא ליצירת פקודה של הוספה שדה:

Alter table tblTrip

Add driver char(8) FOREIGN KEY
REFERANCES tblPerson(id)

דוגמא למחיקת שדה בטבלה:

Alter table tblPerson

Drop Column fullName

DDL – מחיקה טבלה קיימת

ניתן למחוק טבלה ממzd הנתונים (כל עוד היא לא שלחה מפתחות זרים לטבלה אחרת בה האילוץ הוא no) על ידי כתיבת הפקודה:

Drop table tableName

שפת DML – data manipulation language

DML הינה יכולה לעדכן רשומות בטבלאות בעזרת שלוש פקודות:

- Insert – הכנסת ערכים (רשומות) קבועים לטבלה (או תוצאה של שאלתה שנלמד בהמשך)
- Update – עדכון ערכים (COLUMN או COLUMN) ברשומות הקיימות בטבלה.
- Delete – מחיקה של רשומות המקיימות תנאי מסוימים.

DML – הזרת נתונים חדשים

תחביר המשפט:

```
INSERT INTO tableName [(columnList)] VALUES  
dataValues
```

אופציונלי
↑

tableName – שם הטבלה אליה נרצה להכניס נתונים.
columnList – רשימת השדות אליהם אנו מכניסים – פרמטר אופציונלי. אם לא נכנסו אותו, יכנס על פי הסדר בו נוצרה הטבלה.

dataValues – רשימת הערכים בהתאם ל-columnList אם הוכנו או בסדר בה הוצאה הטבלה.

DML – הזרנת נתונים חדשים

דוגמאות:

הוספת רשומה חדשה ל-tblCar :

Insert into tblCar VALUES(111,'2012/1/1',1)

הוספת רשומה חדשה ל-tblCar עם חלק מהערכים:

Insert into tblCar (carNo,kibbutzID) VALUES (333,4)

בשודות אשר לא הכנסנו בהם נתונים פועלת ה-
insert into תכנית את הערך null.

DML – הזרת נתונים חדשים

הערות לגבי הזרת נתונים:

- שימוש לב Ci אטם מכניסים את כל השדות שמודדרים להיות null not, לאחר מכן תקבלו הודעה שגיאה!
- יש צורך להכניס ערך ייחודי (שלא קיים כבר בטבלה) לשדה שמהו key primary, לאחר מכן תתקבל הודעה שגיאה.
- שדה key foreign חייב להכיל ערך אמיתי מהטבלה המקורית ממנו נלקח, לאחר מכן תתקבל הודעה שגיאה.

DML – הזרנת נתונים חדשים

ניתן להזין נתונים לטבלה גם דרך תוצאה של שאלתה, כך
שמה שנכתב בפסקית ה-select של השאלתה הופך להיות
הקלט לטבלה:

```
Insert into tblCar
Select val1, val2, ...
From tbl
```

DML – עדכון נתונים קיימים

תחבר המשפט:

UPDATE tableName

SET fieldName1=newValue1
[,fieldName2=newValue2..]

[WHERE condition]

אופציונלי

אופציונלי

DML – עדכון נתונים קיימים

הערות:

- פסוקית `where` מייצגת תנאי, אך שרק רשומות שעונות עליו ישתנו. אם לא נכנס את פסוקית `where`, יעדכן את כל הרשומות.
- חובה להכניס לפחות ערך אחד שרצים לשנות, אך ניתן להכניס כמה ערכים לשינוי.
- אחרי ביצוע העדכון, לא ניתן לחזור לערכים הקודמים, אך שיש צורך לוודא כי השינוי נכון לפני הרצתו.

DML – עדכון נתונים קיימים

דוגמה:

```
UPDATE tblCar  
SET insExpiration='2014/01/01',kibbutzID=2  
WHERE carNo=111
```

DML – מחיקת נתונים קיימים

משפט delete מאפשר לבצע את הפעולות הבאות:

- מחיקת רשומה בודדת על ידי קритריון המפתח.
- מחיקת קבוצת רשומות (העוננות על קритריון מסוים)
- מחיקת כל הרשומות
- אי מחיקה של רשומות

תחביר המשפט:

DELETE FROM tableName

[WHERE condition]

↑
אופציוניAli

DML – מהיקת נתוניים קיימים

הערות למחיקת ערכים:

- פקודת delete איננה מוחקת ערכים בשדות בודדים, אלא מוחקת את כל הרשומה. אם נרצה למחוק ערכים בשדות מסוימים, נבחר בעדכון נתוניים.
- פקודת ה-delete איננה מוחקת את הטבלה (גם אם מחקנו את כל תוכנה). אם נרצה למחוק את הטבלה כולה, נשתמש ב-drop table כפי שהוזכר ב-DDL.
- אם נרצה למחוק ערכים מטבלה שיש טבלה אחרת הנשענת עליה, נדרש קודם כל למחוק / לעדכן את הרשומות מטבלת הקשר ורק לאחר מכן לטפל בטבלה המקורית.

DML – מחיקת נתונים קיימים

דוגמאות:

מחיקת רשומה אחת במקסימום מהטבלה:

DELETE FROM tblCar

WHERE carNO=999

מחיקת כל הרשומות שתאריך הביטוח שלהם לפני

1/1/2012:

DELETE FROM tblCar

WHERE insExpiration<'2012/01/01'

שאלות Select

מטרת השאלות היא להציג, לשנות ולנתח נתונים בדרכים שונות.

מלבד שאלות מסוג DDL ו-DML שראינו עד כה, הכל החזק ביותר של SQL הינה שאלות הבחירה (שאלות select).

שאלת בחירה מקבלת נתונים מטבלאות, מתוצאות הרצת שאלתה אחרת, מ-view (טבלה מדומה) או מ-temp (טבלה זמנית), בהתאם לкрיטריונים שנקבעו בשאלתא, ומציגה את התוצאות לפי הסדר שנקבע.

פלט של שאלתא מוצג כטבלה.

ההבדל העיקרי בין טבלה לשאלתא הינו שהנתונים בטבלה נשמרים גם לאחר סגירתה בעוד הפלט של השאלתא איננו נשמר ועל כן יש צורך להריץ את השאלתא מחדש על מנת לקבל את התוצאה הרצויה.

מבנה של שאלת select

SELECT [distinct] column1[,column2..]

FROM table1[,table2]

[WHERE condotions]

[GROUP BY columnList]

[HAVING conditions]

[ORDER BY columnList [ASC/DESC]]

חלקி השאלתה

- רשימת השדות שיוצגו בתוצאה השאלתה Select
- רשימת הטבלאות בהן נמצאים הנתונים From
- Where – תנאים לוגיים הקובעים אילו רשומות ישלו מהטבלאות
- Group by – השדות לפייהם מבצעים הקבוצה של הרשומות המציגות
- Having – תנאי לוגי הקובע אילו קבוצות של רשומות יוצגו
- Order by – השדות לפייהם יבוצע המון של הרשומות המציגות

שאילה 1

צור שאלתה המחזיר את פרטי כל המשתמשים
פתרון:

```
select mail, firstName, surName,  
birthDate, gender, description, cityName  
from users
```

כasher מעוניינים בכל התכונות, ניתן לכתוב בפסוקית ה-
select את הסימן * במקום לפרט את השדות

```
select *  
from users
```

שאילה 2

כתב שאללה המחזירה עבור כל משתמש (כתובת מייל)
את שמות האפליקציות אליהן רשום

פתרון:

```
select userMail, appName  
from RegisterdTo
```

чисובים בשאלת SQL

ניתן לכלול חישובים ארכיטמטיים בשאלת SQL בפסקית ה-select או חלק מפסקית ה-.where.

ניתן להפעיל את האופרטורים הארכיטמטיים + ; - ; * ; / ; % על ערכים מספריים בשאלתה. כמו כן ניתן לבצע שרשור של מחזות על ידי האופרטור + ולהשתמש בפונקציות מוכנות נוספות.

שימוש בשמות נרדפים - Alias

ניתן לתת שם נרדף לטבלה, עמודה ושדה חשוב.

יתרונות במתן שמות נרדפים:

1. שמות נרדפים בדרך כלל מקצרים ולכך מצריכים פחות הקלה מהשמות המלאים.
 2. ניתן לתת שמות משמעותיים לעמודות בפלט של שאלתה (במקרה שמות השדות המנהלים לנו את מסד הנתונים).
 3. אם קיימים אותם שמות לשדות בטבלאות שונות, ניתן להבחין ביניהם בצורה קצרה יותר.
- להגדרת שם נרדף, השתמש במילה AS בין השם המקורי לשם החדש שאנו מעוניינים לתת (אך לא חובה לכתב מילה זו).

שאילה 3

כתבו שאלתה המחזיר את כל המשתמשים (מייל ושם פרטי ושם משפחה יופיעו בשדה אחד כ"שם מלא").

פתרון:

```
select mail, firstName + ' ' + surName as  
'fullName'  
from users
```

את המילה as לא חייבים לכתוב:

```
select mail, firstName + ' ' + surName 'full  
Name'  
from users
```

אם ה-alias הוא מילה אחת בלבד, ניתן להוסיף על הגרש סביבה שם החדש

```
select mail, firstName + ' ' + surName fullName  
from users
```

שימוש בפסקית ה-where בשאלתה

פסקית ה-where בשאלתה מאפשרת לשנן רשומות ולקבוע אילו רשומות יופיעו בפלט על פי כללי השאלה.

לאחר המילה where יש לרשום ביטוי לוגי תור שימוש באופרטורים not, or, and, במידת הצורך (יש יותר מתנאי אחד עבור or, and או כל מי שלא עונה על התנאי的帮助下 not).

השימוש בפסקית ה-where יעשה על נתונים "בסיסיים", כלומר על תכונות שונות בטבלאות שנמצאות בידינו עכשווי (טבלאות שנכללו ב-from).

שימוש בפוקית ה-where בשאלתה

אופרטורי השוואה שניתן להשתמש בהם:

| | |
|----------|--------------------------|
| = | Equal |
| > | Greater than |
| < | Less than |
| >= | Greater than or equal to |
| <= | Less than or equal to |
| <> or != | Not equal to |
| LIKE | String comparison test |

שימוש בפסקית ה-where בשאלתה

אופרטורים לוגיים שניית לשימוש בהם:

| Operator | Meaning |
|----------------|---|
| <u>ALL</u> | TRUE if all of a set of comparisons are TRUE. |
| <u>AND</u> | TRUE if both Boolean expressions are TRUE. |
| <u>ANY</u> | TRUE if any one of a set of comparisons are TRUE. |
| <u>BETWEEN</u> | TRUE if the operand is within a range. |
| <u>EXISTS</u> | TRUE if a subquery contains any rows. |
| <u>IN</u> | TRUE if the operand is equal to one of a list of expressions. |
| <u>LIKE</u> | TRUE if the operand matches a pattern. |
| <u>NOT</u> | Reverses the value of any other Boolean operator. |
| <u>OR</u> | TRUE if either Boolean expression is TRUE. |
| <u>SOME</u> | TRUE if some of a set of comparisons are TRUE. |

פונקציות על תאריכים ב-SQL

(DATEDIFF(interval, earlyDate, lateDate) – פונקציה המחשבת הפרש בין תאריכים. התוצאה מוחזרת שנים, חודשים, ימים, שעות או דקות.)
(GETDATE() – פונקציה המחזיר את התאריך הנוכחי (תאריך מחשב).)
(YEAR(date) – פונקציה המחזיר את השנה של התאריך (int).)
(MONTH(date) – פונקציה המחזיר את החודש של התאריך (int).)
(DAY(date) – פונקציה המחזיר את היום של התאריך (int).)
(HOUR(date) – פונקציה המחזיר את השעה של התאריך (int).)
(MINUTE(date) – פונקציה המחזיר את הדקה של התאריך (int).)
(DATEADD(interval, howMuchToAdd, dateAddTo) – הוספה של זמן (שנתיים, חודשים, ימים וכו') לתאריך קיימ.
(CONVERT(toType, theValue) – המרתו של משתנה מסוים למשתנה מסווג אחר. לדוגמה, ניתן להמיר משתנה מסווג varchar למשתנה מסווג date (או עומד בדרישות), להפר וכו').

שאילה 4

כתב שאלתה המחזיר את כל המשתמשים שנרשםו השנה האחרונה. על השאלתך להחזיר את המיל ואת האפליקציה אליה המשתמש נרשם השנה האחרונה

פתרונות:

```
select userMail, appName  
from RegisterdTo  
where  
datediff(day,registrationDate,getdate())  
<= 365
```

שימוש בפסקית ה-where בשאלתה

האופרטור LIKE משמש לאיתור מחרוזת על פי תבנית.
אופרטור זה רגיש לגודל האותיות.

- % מייצג רצף של תוים כלשהם שאנו מחפשים.
- _ מייצג חיפוש של תו בודד.

דוגמאות:

- 'Where str LIKE '%BACK%' – איתור הרשומות שבן השדה str מכיל את המחרוזת BACK.
- 'Where str LIKE 'BACK%' – איתור הרשומות שבן השדה str מתחילה במחרוזת BACK.

שימוש בפוסוקית ה-where בשאלתה

- ‘%BACK’ – איתור הרשומות שבן השדה str מостиים במחוזת BACK.
- ‘_a’ – איתור הרשומות שבן השדה str מתחילה בתו a ואחריו יש לפחות תו נוסף אחד.

שאילהה 5

כתבו שאלתה המחזיר את כל המשתמשים אשר שם
המייל שלהם מתחילה באות s. על השאלתה להחזיר את
המייל של המשתמש, שמו המלא וגילו

פתרונות:

```
select mail, firstName + ' ' + surName  
fullName,  
datediff(year,birthDate,getdate()) age  
from users  
where mail like 's%'
```

שאילהה 6

כתב שאלתה המחזיר את פרטי האפליקציות שנוצרו
בשנת 2014 וגם שם מכיל לפחות ארבע אותיות

פתרון:

```
select *
from Application
where YEAR(creationDate) = 2014 and name
like '_____%'
```

פתרון נוספת:

```
select *
from Application
where YEAR(creationDate) = 2014 and
LEN(name) = 4
```

שאילתה 7

כתוב שאלה המחזירה עבור כל אפליקציה את מספרי
ותוכן שאלות החובה שלה

פתרון:

```
select appName, serialQuestion, content  
from Question  
where mandatory = 1
```

שאילה 8

כתבו שאלה המחזיר עבור כל משתמש את כמהות הזמן ש עבר בין שהכניס משתמש לרישימה לבין שיצר אותו קשר. על השאלה להחזיר את שם המשתמש, שם המשתמש השני (המתאים), האפליקציה בה בחר להתחבר אליו והפרש בימים בין הרישום ליצירת הקשר

פתרון:

```
select userEmail1, userEmail2, appName,  
datediff(day, addDate, talkDate)  
howManyDays  
from List  
where talkDate is not null
```

שאילהה 9

כתב שאלתה המחזיר מיילים של משתמשים אשר נרשמו לאפליקציה כלשי בשנת 2014, 2015, 2016 או 2017

פתרון:

```
select userMail  
from RegisterdTo  
where YEAR(registrationDate) = 2014  
or YEAR(registrationDate) = 2015 or  
YEAR(registrationDate) = 2016 or  
YEAR(registrationDate) = 2017
```

שאילה 9 – פתרונות נוספים

```
select userMail  
from RegisterdTo  
where YEAR(registrationDate) in  
(2014,2015,2016,2017)
```

שאילהה 9 – פתרונות נוספים

```
select userMail  
from RegisterdTo  
where YEAR(registrationDate)  
between 2014 and 2017
```

```
select userMail  
from RegisterdTo  
where YEAR(registrationDate) >=  
2014 and YEAR(registrationDate) <=  
2017
```

שאילהה 9 – פרטיונות נוספים

```
select userMail  
from RegisterdTo  
where registrationDate >  
'2013/12/31' and registrationDate <  
'2018/01/01'
```

אופרטור UNION

האופרטור מחזיר תוצאות של שתי שאלות ללא כפילות נתונים. לרוב משתמשים באופרטור זה על מנת להחזיר תוצאות של שתי שאלות על פי תנאים. אם מעוניינים להשאיר כפילות נתונים, נשתמש ב- ALL UNION.

הערה: השדות חייבים להיות מאותו סוג (מספרים, מחרוזות וכו') אך יכולים להיות עם שמות שונים. אם רצים שיכילו את אותו שם בפלט, יש לבצע alias במשתנה הרצוי הראשון (ה-select שלפני ה-Union) אחרית ישר עם השם המקורי שלו.

צורת כתיבה:

```
SELECT field1,field2
```

```
FROM tbl1
```

```
UNION
```

```
SELECT field1,field2
```

```
FROM tbl1/2
```

שאילה 10

כתבו שאלתה המחזיר המיילים של משתמשים אשר מעוניינים בסוג קשר בשם יחסים פתוחים ומיילם של משתמשים אשר גרים בעיר תל אביב

פתרון:

```
select userEmail  
from userRelation  
where relationName = 'יחסים פתוחים'  
union  
select mail  
from users  
where cityName = 'תל אביב'
```

OPERATOR INTERSECTION

שימוש באופרטור ה-`intersect` יחזיר את התוצאות המשותפות של שתי השאלות. אם הרשימה הופעה במלואה בתוצאה השאלה הראשונה ובתוצאה השאלה השנייה, הערך יתקבל בתשובה הסופית של השאלה.

צורת כתיבה:

```
SELECT field1,field2
```

```
FROM tbl1
```

```
INTERSECT
```

```
SELECT field1,field2
```

```
FROM tbl1/2
```

שאילה 11

כתבו שאלתה המחזיר המיילים של משתמשים אשר נמצאים
ברשימה של אנשים אחרים וגם נרשם לאפליקציה love Me

פתרונות:

```
select userEmail2
from List
intersect
select userMail
from RegisteredTo
where appName = 'love me'
```

אופרטור EXCEPT

שימוש באופרטור except מחזיר בתוצאתו את כל הערכים שנכללו בשאלתה הראשונה ולא נכללו בשאלתה השנייה, כלומר הערכים המשותפים לשאלות לא יוחזו בתור תשובה.

צורת כתיבה:

```
SELECT field1,field2
```

```
FROM tbl1
```

```
EXCEPT
```

```
SELECT field1,field2
```

```
FROM tbl1/2
```

שאילה 12

צור שאילהה המחזיר את המיילים של כל המשתמשים אשר לא הוסיףו אף משתמש אחר לרשימה שלהם

פתרון:

```
select mail  
from users  
except  
select userEmail1  
from List
```

שאילהה 13

כתב שאילתת המחזירה מיילים של משתמשים אשר עוניים על התנאים הבאים:

- מעוניינים בסוג קשר לטוח ארוך
- הכניסו לפחות לבן אדם אחד לרשימה שלהם או שנמצאים ברשימה של משתמש כלשהו
- אין משתמש אשר הם לא אוהבים או שלא אוהב אותם

שאילה 13 - פתרון

```
select userEmail
from userRelation
where relationName = 'קשר לטווח אחר'
intersect
(select userEmail1
from List
union
select userEmail2
from List)
except
(select userEmail1
from Doesn'tLike
union
select userEmail2
from Doesn'tLike)
```

מכפלה קרטזית

תוצאת מכפלה קרטזית בין שתי טבלאות הינה טבלה חדשה
שבה מייצגות כל **הקומבינציות** בין שורות מהטבלה
הראשונה לבין שורות מהטבלה השנייה.

ניתן לבצע מכפלה זו על ידי שימוש בפסיק או על ידי כתיבה
של צמד המילים cross join **בפסוקית ה-from**.

צריך לשים לב שכאשר משתמשים באופציה זו, מקבלים
עבור כל רשותה מכל טבלה את כל הרשומות מהטבלה
השנייה, גם אם אין מתייחסות לאותן ערכים ואין קשרות
בפועל!

מכפלה קרטזית

דוגמא:

פלט השאילתת הבאה יציג את כל הצירופים האפשרים של field1 ו-field2 בין הטבלאות table1 ו-table2

```
SELECT [T1.]field1, [T2.] field3
```

```
FROM table1 [T1],table2 [T2]
```

או

```
SELECT [T1.] field1, [T2.] field3
```

```
FROM table1 [T1] CROSS JOIN table2 [T2]
```

מכפלה קרטזית

אם ניקח שתי טבלאות (אשר יש ביניהן קשר של ייחיד לרבים):

| Table 1 | | | Table 2 | | |
|-----------|-------------|--|-----------|-----------|---------------|
| T1.field1 | T1.field2 | | T2.field1 | T2.field2 | T2.field3(FK) |
| 1 | UK | | 1 | Anneka | 1 |
| 2 | North Korea | | 2 | Kim | 2 |
| 3 | Israel | | 3 | Yao | 2 |

כasher נבצע מכפלה קרטזית בין them (בין table1 ל-table2) נקבל:

מכפלה קרטזית

| field1 | field2 | field1 | field2 | field3 |
|--------|-------------|--------|--------|--------|
| 1 | UK | 1 | Anneka | 1 |
| 1 | UK | 2 | Kim | 2 |
| 1 | UK | 3 | Yao | 2 |
| 2 | North Korea | 1 | Anneka | 1 |
| 2 | North Korea | 2 | Kim | 2 |
| 2 | North Korea | 3 | Yao | 2 |
| 3 | Israel | 1 | Anneka | 1 |
| 3 | Israel | 2 | Kim | 2 |
| 3 | Israel | 3 | Yao | 2 |

אר ור크 הרשומות שבאדורם הן הרשומות שאנו חנו באמת מעוניינים לקבל, שכן רק הן מייצגות את הקשר האמיתי בין הטעלאות.

מכפלה קרטזית

על מנת להימנע מקבלת צירופים שאינם הגיוניים על פי הקשרים במדד הנחונים או אינם שימושיים, ישן שתי אופציות:

1. הנסעה של תנאי בפסוקית `the-where` כך שתיחסיר רק רשומות רלוונטיות.
2. שימוש במכפלה מסוג `join inner` שתיחסיר רק ערכים רלוונטיים (נראה בהמשך).

נרצה להשתמש במכפלה קרטזית כאשר אין לנו שום דרך להגיע מטבלה אחת לטבלה אחרת (למשל במקרה שיש צורך להשתמש ב"טבלה מרחפת" – טבלה שלא קשורה לשום טבלה אחרת)

שאילה 14

כתב שאלתה המחזירה עבור כל משתמש שמעוניין בקשר לטווח אורך את המail שלו, שמו המלא וגילו

פתרון:

```
select U.mail, U.firstName + ' ' +  
U.surName fullName,  
datediff(year,birthdate,getdate())  
age  
from users U, userRelation UR  
where U.mail = UR.userEmail and  
UR.relationName = 'קשר לטווח אורך'
```

צירופים בין טבלאות - Join

אחד הכלים החזקים ביותר ב-`SQL` הינו יכולת לאסוף נתונים מטבלאות שונות ולטפל בהם.

תוצאת פעולה `JOIN` בין שתי טבלאות הינה טבלה חדשה שבה מייצגות כל הקומבינציות בין שורות מהטבלה הראשונה לשורות מהטבלה השנייה, המקיימות את תנאי הקישור.

ניתן לשרسر כמה טבלאות באמצעות `JOIN`, וכך תוצאה פעולה ה-`JOIN` בין שתי הטבלאות הראשונות היא טבלה חדשה שנייה לחבר אותה בעזרת `JOIN` נוספת לטבלה שלישית וכו'.

צירופים בין טבלאות - Join

מספר תנאי החיבור (כמה פעמים שנבצע JOIN)
יהיה כמספר הטרבלאות שאנו מעוניינים לחבר פחות אחד (כי הטבלה האחורונה והראשונה לא "עטופה"
בשני צירופים).

סוגי צירופים נפוצים:

- **צירוף פנימי**
- **צירוף שוווני (צירוף טבעי – לא נתמך על ידי SQL Server אלא רק על ידי MySQL)**
- **צירוף חיצוני**

צירוף פנימי (INNER JOIN)

צירוף זה הוא הנפוץ ביותר לשימוש.

בצירוף זה אנו משווים בין שדות זרים בין שתי טבלאות. לרוב ההשוואה תבוצע בין מפתח ראשי של טבלה אחת למפתח זר (שיכול להיות גם ראשי) בטבלה אחרת.

דרך כתיבת:

```
SELECT field1,field2  
FROM tbl1 inner join tbl2 ON tbl1.primaryKey =  
tbl2.foreignKey/primaryKey
```

תנאי השווון (שחוובה לכתוב אותו לאחר התוכנית לא תরוץ) מורכב מהמילה `as` ואחריה תנאי ההשוואה. תמיד נכניס תנאי השווואה (ולא תנאי חוסר התאמה). ניתן לשרسر כמה תנאים (למשל במקרה שבו המפתח הראשי מורכב מכמה שדות) על ידי כתיבת של המילה `and` בין התנאים.

צירוף פנימי (INNER JOIN)

דוגמאות לשימוש:

- מפתח זר בטבלת רבים מול למפתח ראשי בטבלת היחיד.
- מפתח ראשי בטבלה מקורית מול למפתח ראשי-זר בטבלת קשר שלה.
- מפתח ראשי בטבלת האב בהורשה מול מפתח ראשי-זר בטבלת הבן בהורשה.
- מפתח ראשי בטבלת ישות חזקה מול מפתח ראשי(חלק)-זר בטבלת ישות הchlsha שלה.

שאילה 14 – תור שימוש ב-inner join

כתבו שאילה המחזירה עבור כל משתמש שמעוניין בקשר לטווח אורך את המail שלו, שמו המלא וגילו

פתרון:

```
select U.mail, U.firstName + ' ' +  
U.surName fullName,  
datediff(year,birthdate,getdate()) age  
from users U inner join userRelation  
UR on U.mail = UR.userEmail  
where UR.relationName = 'קשר לטווח אורך'
```

שאילה 15

כתב שאלתה עבור אפליקציות בעלות משתמשים חדשים את שמה ורשימת המשתמשים החדשניים באפליקציה. משתמש חדש הנו משתמש שנרשם לאפליקציה בחצי שנה האחרונות. עבור כל משתמש יש להציג את המיל שלו, שם המשתמש באפליקציה, שמו המלא, גילו, מגדרו, תאריך הרישום לאפליקציה ושם העיר בה מתגורר.

שאילה 15 - פתרון

```
select appName, RT.userMail, nickName,  
firstName + ' ' + surname fullName,  
gender, registrationDate,  
datediff(year,birthDate,getdate()) age,  
cityName  
from RegisterdTo RT inner join  
Users U on RT.userMail=U.mail  
where  
datediff(month,registrationDate,getdate())  
<= 6
```

צירוף חיצוני (OUTER JOIN)

בצירופים הקודמים שבירצנו, שורה בטבלה אחת שאיננה התאימה לשורה בטבלה השנייה על בסיס תנאי השוואה נופתה מהתוצאה הסופית.

ב-JOIN OUTER הניפוי נעשה באופן חלק (באחת מהטבלאות) או בכלל לא נעשה. עבור רשומות שאין להם ערכים מתאימים בטבלה השנייה, יוכנס הערך null במקום.

צירוף חיצוני (OUTER JOIN)

סוגי JOIN : OUTER

- JOIN JOIN – LEFT OUTER המשאיר בפלט שורות יתומות שבאות מהטבלה הראשונה (השמאלית מביניהם).
- JOIN JOIN – RIGHT OUTER המשאיר בפלא שורות יתומות שבאות מהטבלה השנייה (הימנית מביניהם).

שימוש ב-Null Is

SQL מאפשרת לבדוק אם ערך מסוים הוא ערך דמה (ערך null – ערך שלא קיבל השמה – הוא חסר ערך או לא מוגדר).

ב-SQL אין אפשרות להשתמש באופרטור = או <> על מנת להשוות שדה מסוים ל-null. השוואה מסוג זה תמיד תחזיר את הערך false, משום שכל ערך null מקבל למעשה מספר ייחודי ולכן אף פעם לא יהיה זהה לאחר.

ניתן להשתמש באופרטור null is ו-null not is על מנת לבדוק האם ערך כלשהו הוא null. הבדיקה נעשית בשורת ה-where ומסוננת בצורה זו.

שאילה 16

כתב שאלתה המחזיר מיילים של משתמשים אשר לא הוסיפו אף משתמש אחר לרשימה שלהם

פתרון:

```
select U.mail  
from users U left outer join List L  
on U.mail = L.userEmail1  
where L.userEmail2 is null
```

צירוף עצמי

צירוף עצמי מתאר מצב בו אנו מצרפים לטבלה שאנו מעוניינים להשתמש בה את אותה טבלה שוב.

תנאי ההשוואה שלנו לרוב לא יהיה המפתח הראשי שכן במקרה זה בהכרח ייצור את אותה רשומה פעמיים.

יש לשים לב כי מצב זה יכול להחזיר לנו את אותה רשומה פעמיים (פעם אחת התקבלה מהטבלה הראשונה שהעלאנו ופעם שנייה מהטבלה השנייה שהעלאנו ותנאי ההשוואה שלנו מתקיים) ואם אנו מעוניינים לסנן רשומות כפולות שכאהלה, נוסיף תנאי המעיף רשומות עם אותו מפתח ראשי.

בנוסף יש לשים לב כי כל צמד רשומות יתקבל פעמיים, שכן יופיע פעם ראשונה בטבלה הראשונה (וימצא מישהו מתאים בטבלה השנייה) ואז מי שהופיע בטבלה השנייה יופיע בטבלה הראשונה (וימצא את אותו ערך תואם שהתקבל בהשוואה הראשונה).

שאילהה 17

כתב שאלתה המציגת את פרטי המשתמשים שמעוניינים לפחות שני סוגי קשרים שונים עם מין מועד שונה בשני סוגי הקשרים האלו. עבור המשתמשים שעומדים בתנאי יש להציג את המיל, שם פרטי, שם משפחה, מגדר, ורשמיית סוג הקשרים השונים ומהן המועד בסוגי קשרים אלו (כל סוג קשר ומין בשורה נפרדת עם פרטי המשתמש). שימוש לב המשתמשים שעומדים בתנאי יופיע יותר מפעם אחת בתוצאות השאלתה.

שאילה 17 - פתרון

```
select U.mail, firstName, surName,  
U.gender , UR1.relationName, UR1.gender  
from Users U inner join userRelation UR1  
on U.mail=UR1.userEmail inner join  
userRelation UR2 on U.mail=UR2.userEmail  
where UR1.gender<> UR2.gender
```

שאילה 18

כתב שאלתה המחזיר את פרטי המשתמשים שמשתמש אחר מעוניין בהם. על השאלתה להחזיר את המail והשם המלא של המשתמש המעוניין והmail והשם המלא של המשתמש שהוא מתעניין בו

פתרון:

```
select U1.mail firstUserEmail, U1.firstName  
+ ' ' +U1.surName firstUserName, U2.mail  
secondUserEmail, U2.firstName + ' '  
+U2.surName secondUserName  
  
from List L inner join users U1 on  
L.userEmail1 = U1.mail inner join users U2  
on L.userEmail2 = U2.mail
```

פעולת קיבוץ – Group By

אפשרית ה-`by group` קובעת על פי אילו שדות יקבצו הרשומות לקבוצות. לכל קבוצה צואת תופיע רשומה אחת ייחידה בפלט השאלה.

אחרי שביצענו הקבוצה של הנתונים, ניתן להפעיל על הקבוצה פונקציות מקבוצה סגורה של פונקציות.

דרך כתיבה:

Select field1,func(field2)

From tblName

Group by field1

פעולת קיבוץ – Group By

פעולת קיבוץ היא פעולה המאחדת מספר שורות לשורה אחת. סוג הfonקציות שניתן לבצע על הרשומות המקבוצות:

עובד על
ערכים
מספריים



Sum – סיכום הערכים שקיבצנו

Average – ממוצע הערכים שקיבצנו

Max – הערך המקסימלי מהערכים שקיבצנו

Min – הערך המינימלי מהערכים שקיבצנו

Count – סכימת כמות הרכים שקיבצנו – מוחזירה את כמות הרכים הנבחרים בשאלתה. ניתן לבצע סכימה עבור כל הרכים (שימוש ב-*) או עבור שדה ספציפי.

הารגומנט שחוזר מfonקציות אלה תמיד יחיד.

סינון רשומות מקובצות - HAVING

תנאים על רשומות מקובצות יעשו בפסקית ה-
.having

פסקית זו נעשית אחרי קיבוץ הרשומות (פסקית ה-
by group) ועל כן כל פעולה על פונקציות האgregציה
ישנה בפסקית זו.

דרך כתיבה:

Select field1,func(field2)

From tblName

Group by field1

Having func(field2)>0

שאילה 19

כתב שאלתה המחזיר עבר כל משתמש (מייל ושם מלא) את כמה אפליקציות אליו רשום

פתרון:

```
select U.mail, U.firstName + ' ' +  
U.surName fullName, count(*)  
numOfApps  
from users U inner join RegisterdTo  
RT on U.mail = RT.userMail  
group by U.mail, U.firstName,  
U.surName
```

שאילה 20

כתב שאלתה המחזירה עבור כל משתמש (מייל ושם מלא) את תאריך ההוספה הראשון של משתמש לרשימה שלו (המשתמש המוחזר הוא זה שהוסיף לרשימה של עצמו)

פתרון:

```
select U.mail, U.firstName + ' ' +  
U.surName fullName, min(L.addDate)  
firstDate  
from users U inner join List L on  
U.mail = L.userEmail1  
group by U.mail, U.firstName,  
U.surName
```

שאילה 21

כתוב שאלתה המחייבת עברור כל משתמש (מייל ושם מלא) את תאריך ההוספה הראשון של משתמש לרשיימה שלו (המשתמש המוחזר הוא זה שהוסיף לרשיימה של עצמו). במקרה שבו המשתמש לא הוסיף אף אחד לרשיימה, יש להחזירו גם עם>User ריק בתאריך הפניה

פתרון:

```
select U.mail, U.firstName + ' ' +  
U.surName fullName, min(L.addDate)  
firstDate  
from users U left outer join List L on  
U.mail = L.userEmail1  
group by U.mail, U.firstName, U.surName
```

שאילה 22

כתב שאלתה המחזירה עבור כל מייל של משתמש את כמות הימים הממוצעת שעברה בין ההוספה לרשימה לפניה למשתמש (במקרה ופנה אליו)

פתרון:

```
select userEmail1,  
avg(datediff(day,addDate,talkDate))  
avgTimeInDays  
from List  
where talkDate is not null  
group by userEmail1
```

שאילה 23

כתוב שאילה המחזירה עבור כל מייל של משתמש את
כמה הימים הממוצעת שעברה בין ההוספה לרשימה לפניה
למשתמש (במקרה ופנה אליו) במקרה ובו כמה הימים
קטנה מ- 7

פתרון:

```
select userEmail1,  
avg(datediff(day,addDate,talkDate))  
avgTimeInDays  
from List  
where talkDate is not null  
group by userEmail1  
having  
avg(datediff(day,addDate,talkDate))<7
```

שאילה 24

כתב שאלתה המחזיר עבר כל משתמש (מייל ושם מלא) את כמות המשתמשים שהוא לא מעוניין שיופיעו לו בעתיד. במקרה שבו המשתמש לא נכנס מישו לרשימת המשתמשים הלא רצויים, יש להחזיר 0

פתרון:

```
select U.mail, U.firstName + ' ' +  
U.surName fullName, count(DL.userEmail2)  
numOfUnLikedUsers  
from users U left outer join DoesnLike  
DL on U.mail = DL.userEmail1  
group by U.mail, U.firstName, U.surName
```

שימוש ב-DISTINCT

כפי שראינו, את המילה `distinct` אפשר לכתוב ישר אחרי ה-`select` לפני רשימת הערכים שאנו מעוניינים שיופיעו בפלט. כתיבת ה-`distinct` מוריד ערכים כפולים ולמעשה זהה לכתיבה של `group by` על כל ערכי החזרה של השאלתה. ניתן להשתמש ב-`distinct` גם בתוך פונקציית הקיבוץ אז ערכים כפולים יספרו פעם אחת בלבד.

שאילה 25

כתב שאלתה המחזירה עבור כל מייל של משתמש שנרשם לאפליקציות את כמות התאריכים השונים בהם

נרשם

פתרון:

```
select userMail,  
count(distinct registrationDate)  
distinctDates  
from RegisterdTo  
group by userMail
```

שימוש בפוקית ה-order by

פוקית ה-order by בשאלתה מאפשרת את מילוי פלא השאלתה.

ברירת המחדל אם לא נכתב דבר הינו סדר עולה.
על מנת למיין בסדר יורד יש צורך לכתוב את המילה DESC.
ניתן לבצע מילון ראשוני, שניוני וכו', על פי סדר כתיבת
השדות משמאל לימין.

שאילה 26

כתב שאלתה המחזירה עבור כל אפליקציה (על כל פרטיה) את כמות הנרשמים אליה. את התוצאות יש לסדר בסדר יורד על פי כמות הנרשמים

פתרון:

```
select A.name, A.description,  
A.creationDate, count(*) numOfUsers  
from Application A inner join  
RegisterdTo RT on A.name = RT.appName  
group by A.name, A.description,  
A.creationDate  
order by numOfUsers desc
```

שאילה 27

כתבו שאלתה המחזירה עבור כל משתמש (מייל ושם מלא) שהעלה תמונות לאפליקציה את כמהות התמונות שהוא צילם בשנה הנוכחית. יש להציג רק משתמשים אשר העלו לפחות חמיש תמונות בשנה הנוכחית.

פתרון:

```
select U.mail, U.firstName + ' ' +  
U.surName fullName, count(*) numOfPics  
from users U inner join Picture P on  
U.mail = P.userEmail inner join matchMePic  
MMP on P.serialNum = MMP.picNum  
where year(MMP.picDate) = year(getdate())  
group by U.mail, U.firstName, U.surName  
having count(*) >= 5
```

שאילות מקווננות

ביצוע שאילות מסוימות מחייב כי הערכים מבוסיס הנתונים יוחזרו ראשית ורק אז נעשה בהם שימוש בתנאי של ההשוואה.

שאילות מסוג זה ניתן לנוכח על ידי שימוש בשאילות מקווננות, שמהוות שאילות select שלמות בתוך פסוקית ה-where או ה-having של השאילה אחרת.

השאילה אחרת נקראת השאילה חיצונית (outer query) והשאילה המקוונת בה נקראת השאילה פנימית (inner query).

הערה: ניתן להשתמש בשאילתה מקוונת גם בפסוקית ה-from וכן תהפוך להיות טבלת קלט לשאילתה.

סוגי שאלות מקוננות

שאלות מקוננות בפוסקית ה-where (או ה-having) יכולות להיות מכמה סוגים:

- במקורה ובו מהשאלה הפנימית חוזר ערך ייחיד, ניתן להשתמש ב-!=,<=,>,<=,>
- שאלתת all – האם האיבר בשאלה החיצונית מקיים משהו שנכון לגביו כל מי שבטליה שחרזה מהשאלה הפנימית? (לרוב האם גדול / גדול-שווה מכל מי שחר או קטן / קטן-שווה מכל מי שחר).
- שאלתת some – האם האיבר בשאלה החיצונית מקיים את התנאי ביחס לשאלה הפנימית, כלומר האם מישו שענה על התנאי? (לרוב האם אני גדול / קטן ממי שהוא בשאלה הפנימית).
- שאלתת who – האם האיבר בשאלה החיצונית נמצא בטלה של השאלה הפנימית?
- שאלתת exist – האם השאלה הפנימית החזירה ערך כלשהו ביחס לאיבר מהשאלה החיצונית?

שאלות בלתי תלויות

המטרה שלנו היא לבדוק תנאי מסוים, ועל כן נמצא בפסקית ה-*where* או ה-*having*.

שאלות מסוג some, all, who משווים בין איבר מסוים לקבוצת רשות שחוזרת מהשאילתת הפנימית (שמהוña למעשה טבלת פלט).

שאלות אלה על כן שאלות בלתי תלויות, כלומר השאילת הפנימית אינה תלויות בטבלאות מהשאילת החיצונית (יכולת רוץ באופן עצמאי), ועל כן מתקיימת קודם כל השאילת הפנימית ולאחריה השאילת החיצונית.

שאילהה 28

כתב שאליה המחזיר את המיילים והשמות המלאים של המשתמשים אשר גרים באותה עיר כמו רומי אבולעפה
דרך פתרון:

```
select U.mail, U.firstName + ' ' +
U.surName fullName
from users U
where U.cityName in
(select U2.cityName
from users U2
where firstName = 'רומי'N and
surName = 'אבולעפה'N)
```

שאילה 29

כתב שאלתה המחזיר את המיילים והשמות המלאים של המשתמשים המבוגרים ביותר ששמורים במערכת
דרך פתרון א:

```
select U.mail, U.firstName + ' ' +  
U.surName fullName  
from users U  
where U.birthDate <= all  
(select U2.birthDate  
from users U2)
```

שאילהה 29

כתב שאלתה המחזיר את המיילים והשמות המלאים של המשתמשים המבוגרים ביותר שהמוחדרים במערכת
דרך פתרון ב:

```
select U.mail, U.firstName + ' ' +  
U.surName fullName  
from users U  
where U.birthDate =  
(select min(U2.birthDate)  
from users U2)
```

שאילהה 29

כתב שאלתה המחזיר את המילים והשמות המלאים של המשתמשים המבוגרים ביותר שהמוחדרים במערכת
דרך פתרון ג (קצת פחות מדויקת):

```
select U.mail, U.firstName + ' ' + U.surName
fullName
from users U
where datediff(year,U.birthDate,getdate())
      >= all
(select datediff(year,U2.birthDate,getdate())
from users U2)
```

שאילהה 30

כתב שאלתה המחזיר את המיילים והשמות המלאים של כל המשתמשים שנרשמו לכל האפליקציות הקיימות במערכת

דרך פתרון:

```
select U.mail, U.firstName + ' ' +  
U.surName fullName  
from users U inner join RegisterdTo RT on  
U.mail = RT.userMail  
group by U.mail, U.firstName, U.surName  
having count(*) = (select count(*)  
from Application)
```

שאילה 31

כתב שאלתך המחזירה את שם האפליקציה ומספר השאלה
שהכי הרבה משתמשים ענו עליה
דרך פתרון (כמובן שיש עוד דרכים):

```
select A.appName, A.questionSerial
from Answers A
group by A.appName, A.questionSerial
having count(*) >= all
(select top 1 count(*)
from Answers A2
group by A2.appName, A2.questionSerial
order by count(*) desc)
```

שאלות תלויות

שאלות תלויות יקחו חלק גם כן בפוסוקית ה-*where* (או ה-*having*) על מנת לבדוק תנאי כלשהו.

שאלות מסוג *exists* בודקת קבוצת רשותות לגופה. בסוג זה של שאלות הטבלה הפנימית צריכה להיות **קשורה לשאלתה החיצונית**.

כלומר השאלה הפנימית תלויות בטבלאות מהשאלה החיצונית - טבלה שנבחרה בשאלתה החיצונית תיקח חלק פעיל בשאלתה הפנימית, ועל כן עברו כל ערך שמוחזר בשאלתה החיצונית מתבצעת השאלה הפנימית ומחזירה לה תשובה רלוונטיות (שכן לא ניתן מראש לדעת לאילו תשבות אלו מ暢ים).

שאילה 32

כתב שאלתה המחזיר עבור כל משתמש (מייל בלבד) את שם האפליקציה שנרשם אליה לראשונה. במקרה ונרשם לכמה אפליקציות באותו יום, יש להחזיר את כלן

דרך פתרון:

```
select RT.userMail, RT.appName  
from RegisterdTo RT  
where not exists  
(select *  
from RegisterdTo RT2  
where RT.userMail = RT2.userMail and  
RT2.registrationDate < RT.registrationDate)
```

שאילה 33

כתב שאלתה המחזירה מיילים ושמות של משתמשים כר שבכל אפליקציה בה הם רשומים הם ענו על לפחות שאלת רשות אחת (שאלה שאינה מוגדרת כשאלת חובה). במקרה שבו המשתמש לא רשום לאף אפליקציה, אין להחזיר אותו

שאילה 33 - פתרון

```
select U.mail, U.firstName + ' ' + U.surName
fullName
from Users U left outer join RegisterdTo RT on
U.mail = RT.userMail
where RT.userMail is not null
and not exists
(select RT1.userMail, RT1.appName
from RegisterdTo RT1
where RT1.userMail=U.mail
except
select A.userEmail, A.appName
from Answers A inner join Question Q on
A.appName=Q.appName
where mandatory=0 and A.userEmail=U.mail)
```

שאילה 33 – פתרון נוסף

```
select U.mail, U.firstName + ' ' + U.surName
fullName
from Users U left outer join
Answers A on A.userEmail=U.mail left outer join
Question Q on A.appName=Q.appName
where mandatory=0
group by U.mail, U.firstName, U.surName
having count(distinct A.appName) =
(select count(distinct Q1.appName)
from Users U1 inner join RegisterdTo RT1 on
RT1.userMail=U1.mail inner join Question Q1 on
RT1.appName=Q1.appName
where RT1.userMail=U.mail and mandatory=0)
```

שאילתה 34

כתב שאלתה המחזירה פרטי אפליקציות "עלות". אפליקציה עלה הינה אפליקציה שמספר הנרשמים אליה גדל ב-50% לפחות ב-30 הימים האחרונים מיום הרצת השאלתה (לעומת 30 הימים הקודמים להם). על השאלתה להחזיר את כלל פרטי האפליקציה

שאילה 34 - פתרון

```
select A.name, A.description, A.creationDate
from RegisterdTo RT inner join Application A
on RT.appName=A.name
where datediff(day, registrationDate,
GETDATE())<=30
group by A.name, A.description,
A.creationDate
having count(RT.userMail)/2 >=
(select count(RT1.userMail)
from RegisterdTo RT1
where RT1.appName=A.Name and
datediff(day,RT1.registrationDate,GETDATE())
between 30 and 60)
```

שאילות מבט - views

SQL Server איננו שומר את השאילות שאנו כותבים ועל כן יש צורך לשמר אותם בקובץ חיצוני על מנת להשתמש בהן בהמשך.

ישנה אפשרות להגדיר תוצאה שאילתת מסויימת כטבלה מדומה (view).

טבלה זו איננה שומרת ערכים בתוכה אלא כאשר משתמשים בה בפוסקית ה-from היא מרים את השאילתת שכותבה בה מאחריו הקליים, מחרירה את התשובה ומשתמשת בה השאילתת החדשה (התוצאות דינמיות בגין לערכים הקיימים בטבלאות בעת הרצת ה-view).

Views דומים לפונקציות בתכנות – נרצה להשתמש בהן אם יש חלק משאלתה ש חוזר פעמים רבים (בТО록 הרבה השאילות) או שהשאלתה מורכבת מדי ונרצה לפרק אותה לבעיות קטנות ולבסוף להרכיב פתרון שלם מהבעיות הקטנות שפתרנו.

שאילות מבט - views

שאילות מבט נשמרות כחלק מה-DB ועוד שאילתות "רגילות" אינן נשמרות, ועל כן ניתן להשתמש בהן בכל שאלה (ויהו טבלת קלט לשאילתה).

סיבות לשימרת תצפיות חלק מה-DB:

- **שימושי** למספר משתמשים שונים.
- **שימוש מרובה** בשאילתה.
- **התצפיות מסתירות** את הטבלאות. מונע גישה ישירה של משתמשים לטבלאות ומאפשר למשתמשים לטפל בנתונים דרך התצפיות המוגדרות בלבד.

חסרון השימוש בתצפיות הינו נשמרות במדד הנתונים על כן "ocabidot" על מasad הנתונים.

שאילות מבט - views

יצירת שאילתת מבט נעשית בצורה דומה ליצירת שאילתת.
יצירת שאילתת המבט כוללת בתוכה השאילתת רגילה, אך
מקבלת הגדירה של שאילתת מבט וכר ה-SQL יודע לשמר
אותה בתור "טבלה וירטואלית" (לכל DB יש תיקיה של
views) במקום להריצה ולתת פלט.

אופן כתיבה:

create view *viewName* as
SQL statement

צורת שימוש:

Select *
From tbl1 inner join *viewName* on ID=*viewID*

טבלאות זמניות - temp

טבלה זמנית היא טבלה אשר אנו יוצרים ב-script (עמוד) הנוכחי והוא קיים רק ב-script הנוכחי.

בניגוד ל-view, הוא לא מתעדכן בעת ריצה אלא בעת השמת הנתונים בו.

ניתן להגדיר טבלה שהיא זמנית (על ידי... create table) או שנייתן להכנס תוצאה של שאלתה לתוכה ללא טבלה זמנית שלא הייתה קיימת עד כה. צורת הזיהוי של טבלה זמנית הוא על ידי זה שלפני שם הטבלה כתוב #.

במסגרת אותו ה-script, ניתן להשתמש בטבלה זו כאשר היא טבלה שקיימת במאסד הנתונים.

טבלאות זמניות – temp – צורת כתיבה ליצירת טבלה כתוצאה של שאלתה

Select *

Into #myTemp2

From flights

שימוש לב Ci בשיטה זו לא ניתן להכניס ערכים נוספים
לטבלה מלבד הערכים שהוכנסו בעת ייצירת הטבלה.

טבלאות זמניות – temp – צורת כתיבה ליצירת טבלה

Create table #myTemp

(

ID int,

First nvarchar(20)

....

)

בשיטה זו על כן ניתן להכניס לכל אורך ה-script נתונים לטבלה זו ועל כן נרצה להשתמש בשיטה זו כאשר אנו מעוניינים לקבל נתונים מגוון שאילתות.

בהמשך, על ידי השאילתת insert into נכניס נתונים לטבלה זו (insert into #myTemp select....)

שימוש ב-case בשפת SQL

ניתן להשתמש במילה השמורה case המאפשרת לבדוק מספר תנאים ייחודיים תוך מניעת שימוש במשפטי תנאי חוזרים או מקוונים.

דרך שימוש:

```
SELECT CASE columnName/expression
        WHEN value1 THEN result1
        WHEN value2 THEN result2
        [ELSE elseResult]
      END
```

שאילתה 35

כתב שאלתה המחזירה עbor כל אפליקציה את מידת הפופולריות שלה על פי ההגדרה הבאה:

- אם רשומים אליה עד 1000 משתמשים, יש להציג שימוש פחות
- אם רשומים אליה בין 1000 ל-5000 משתמשים, יש להציג שימוש סביר
- אם רשומים אליה מעל ל-5000 משתמשים, יש להציג שימוש מסיבי.

על השאלתה להציג מלבד מידת הפופולריות את כל פרטי האפליקציה (ניתן להניח כי לכל אפליקציה רשום לפחות משתמש אחד)

שאילה 35 - פתרון

```
select A.name, A.description,  
A.creationDate,  
case when count(*)<5000 then 'שימוש פחות' N  
when count(*) between 5000 and 10000 then  
'שימוש סביר' N  
'מידת פופולריות' N end 'שימוש מסיבי' N  
from RegisteredTo RT inner join  
Application A on RT.appName = A.name  
group by A.name, A.description,  
A.creationDate
```

שאילה 36

כתב שאלתה המחזירה עבור המשתמשים הרשומים במערכת את מייל המשתמש, שם מלא, גיל ומגדל. כמו כן, יש להציג את רמת השימוש של המשתמש באפליקציות השונות כך שם יצר קשר עם משתמשים אחרים בחצי שנה האחרונות באמצעות 1-2 אפליקציות שונות יוצג "שימוש בקטנה", במידה יצר קשר ע"י 3-5 אפליקציות שונות בחצי שנה האחרונות יוצג "שימוש ביןוני", לאחר מכן יוצג "שימוש נרחב". במידה ולא יצר קשר עם אף משתמש בחצי שנה האחרונה יוצג "לא משתמש".

שאילה 36 - פתרון

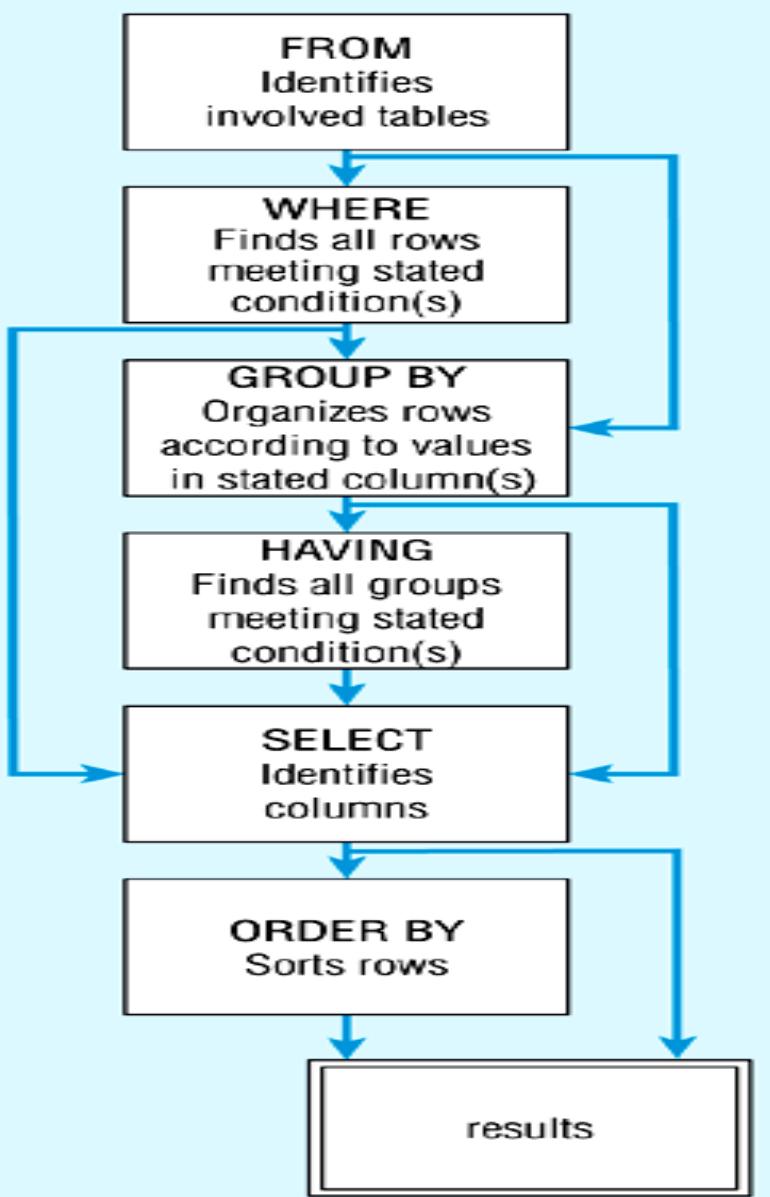
```
select U.mail, firstName + ' ' + surName fullName,  
gender, datediff(year, birthDate, getdate()) age,  
case when activityLevel is null then 'לא מחפש'  
else activityLevel end activityLevel  
from ( select U.mail, case when COUNT(distinct  
userEmail2) between 1 and 10 then N'מחפש בקטנה'  
when COUNT(distinct userEmail2) between 11 and 20 then  
N'מחפש ברכיניות'  
when COUNT(distinct userEmail2)>20 then N'מחפש בunnerות'  
end activityLevel  
from users U inner join List L on U.mail=L.userEmail1  
where datediff(month, L.addDate, getdate())<=6  
group by U.mail)x right join Users U on U.mail=x.mail
```

שאילה 36 – פתרון תוך שימוש ב-view

```
create view HowMuchLookingFor as
select U.mail, case when COUNT(distinct userEmail2) between 1
and 10 then ' מחפש בקטנה' N
when COUNT(distinct userEmail2) between 11 and 20 then
' מחפש ברצינות' N
when COUNT(distinct userEmail2)>20 then ' מחפש בנהרות' N end
activityLevel
from users U inner join List L on U.mail=L.userEmail1
where datediff(month, L.addDate, getdate())<=6
group by U.mail

select U.mail, firstName + ' ' + surName fullName, gender,
datediff(year, birthDate, getdate()) age,
case when activityLevel is null then ' לא מחפש' N else
activityLevel end as activityLevel
from HowMuchLookingFor HMLF right join Users U on
U.mail=HMLF.mail
```

SQL statement processing order



1. השלב הראשון הוא בחירת הטרבלאות אליהן יש לגשת (FROM)
2. אם ישנו תנאים לסינון ישנו מעבר לפסוקית ה WHERE
3. במידה ויש פונקציות הקבוצה, ישנו מעבר לפסוקית GROUP BY לשם ביצוע תנאי הקיבוץ
4. במידה יש תנאי על קבוצה ישנו מעבר לפסוקית ה having
5. לאחר מכן ניתן היה לנפות עבור התוצאה רק את העמודות המתבקשות בשורת ה SELECT
6. במידה יש דרישת למין ישנו מעבר לפסוקית ה ORDER BY שמיינית רק את התוצאה הסופית
7. לבסוף הצגת הפלט

Stored Procedures

- **Stored Procedures** - פ्रוצדורה הכתובה בשפה פנימית של מודulia הנתונים, המבוססת על SQL וכוללת פקודות בקרה – Stored procedure מכילה הוראות המבצעות פעולה על ה-DB, כולל קראות ל- stored procedures אחרות
- Stored procedure יכולה לקבל/להחזיר פרמטרים
- Stored procedure מחזירה status value המציין הצלחה או כישלון
- Stored procedure הנה אובייקט מקומפל הכלל לפחות משפט SQL אחד
- Stored procedure נשמרת כאובייקט קבוע ב- DB
- Stored procedure משמשת גם כדי להס提ר משפט SQL מסווג מהאפליקציה

Stored Procedures

- יתרונות השימוש ב- **:Stored Procedure**
 - מאפשר תכננות מודולרי (modular programming). ניתן לעדכן ללא תלות בקוד המקור של היישום (faster execution יוטר)
 - מאפשר ריצה מהירה-יעילה יותר (faster execution יוטר)
 - הפרוצדורה מנוטחת ועובד אופטימיזציה כאשר היא נוצרת וניתן להשתמש בגרסתה של הפרוצדורה הנמצאת בזיכרון לאחר הריצה הראשונה שלה. ללא stored procedure, בכל ריצה של הוראה ל- DB הנשלחת מהיישום, ההוראה מקומפלט, עוברת אופטימיזציה ומורצת ע"י SQL Server (איטי יותר)
 - מפחיתים עומס תנועה ברשת (פעולה הכוללת מאות שורות קוד הנשמרת כ- stored procedure מבוצעת כהוראה אחת הנשלחת לרשת במקום שליחת של מאות שורות קוד הנשלחות ברשת)

Stored Procedures

- הגדרת Stored Procedure כוללת 2 מרכיבים עיקריים:
 - שם הPROCEDURE והפרמטרים
 - גוף הPROCEDURE
- דוגמה: PROCEDURE שמקבלת מייל של משתמש ו嬗ה נתונים ומחזירה את פרטי האפליקציות (שם וטיואר) שלהם רשום

```
CREATE PROC AppToUserInfo @userEmail varchar(50), @Year int AS  
SELECT C.courseId, StyleName, difficultyRank  
FROM RegisteredTo RT INNER JOIN Application A  
    ON RT.appName = A.name  
WHERE userEmail = @userEmail AND Year= @Year
```

Stored Procedures

- לאחר שמיירת הסכמה של ה SP בבסיס הנתונים, ניתן להריץ את ה SP בכל פעם שרצוים עם פרמטרים אחרים.
- דרך ההרצה (בשפת SQL):

EXECUTE procedure_name value1, value2....

GO

EXECUTE AppToUserInfo 'romi@gmail.com', 2016

EXECUTE AppToUserInfo 'Sharon@gmail.com', 2014

Stored Procedures

- בירית מחדל לפורמטר שערך לא סופק
 - ניתן להחליט האם לא הוזן ערך לפורמטר, תנתן הודעה שגיאה
 - לחופין, אם לא ניתן ערך לפורמטר, תבצע פקודה SQL אחרת

```
CREATE PROC AppToUserInfo @userEmail varchar(50), @Year  
int = NULL AS  
IF @dancerNum IS NULL PRINT 'Give a required year'  
ELSE  
SELECT C.courseId, StyleName, difficultyRank  
FROM RegisteredTo RT INNER JOIN Application A  
    ON RT.appName = A.name  
WHERE userEmail = @userEmail AND Year= @Year
```

מבנה בקרה כמו
בשפת תכנות "רגילה"
IF, IF-ELSE

```
EXECUTE AppToUserInfo 'romi@gmail.com'
```

Stored Procedures

- פרמטר מסווג output (מקבל ערך בפרקודה)

```
CREATE PROC GetOldestUser @oldest INT OUTPUT
AS
BEGIN
    SELECT @oldest = (SELECT MAX(datediff(year,birthDate,getdate()))
                      FROM Users)
END
```

– אחרי הפעלת הפרקודה אפשר להשתמש בפרמטר @managerID

```
DECLARE @oldest int;
EXECUTE GetOldestUser @oldest output
PRINT @oldest
```

טריגרים

מטרת הטריגר הינה לאכוף כללים עוקיים בבסיס הנתוניים.

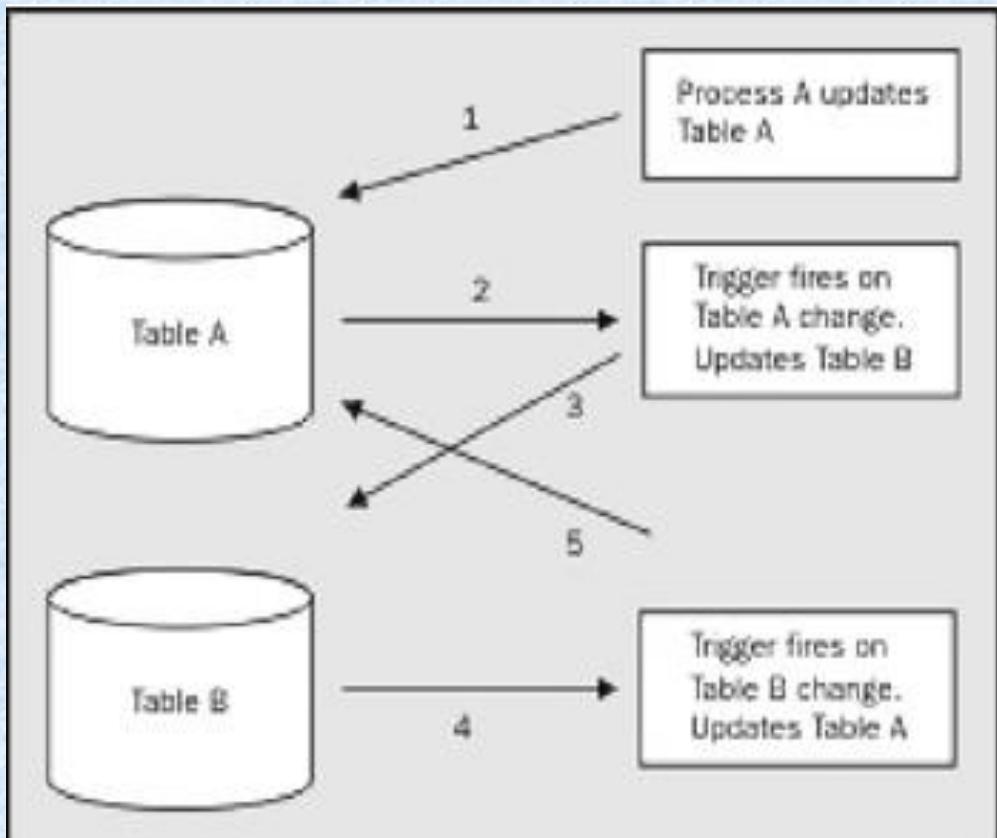
כללים עוקיים אלה לא יכולים לשמור / לבדוק ברמת-h-DB ועל כן יש צורך לבצע בדיקות אלה בזמן ניהול הנתוניים.

בדיקת הכללים העוקיים מתבצע אחרי ביצוע פעולה כלשהי על רשומה ב-h-DB

טריגרים יכולים להיות משלושה סוגים:

- טריגר עדכון - update triggers
- טריגר הכנסה – insert triggers
- טריגר מחיקה – deleted triggers

טראגירים – איך זה עובד?



1. תהליך A עדכן את טבלה A
2. פעולה זו "ירתה" מטבלה A ה trigger שנורה מטבלה A עדכן את טבלה B
3. פעולה זו "ירתה" מטבלה B (נוסף) מטבלה B ה trigger שנורה מטבלה B עדכן את טבלה A

צורת כתיבה ומינוחים שימושיים

- **צורת כתיבה עבור הכנסה או עדכון:**

create TRIGGER triggerName ON tblName AFTER
UPDATE/INSERT AS
SQL statement / SQL check

- **צורת כתיבה עבור מחיקה:**

create TRIGGER triggerName ON tblName FOR
DELETE AS
SQL statement / SQL check

מינוחים שימושיים:

- Rollback – ביטול הפעולה שהתבצעה וחרזרה למצב קודם.
- Inserted – שומר את הרשומה האخונה שהוכנסה (שהביאה להתחלה הטריגר).

מינוחים משמעותיים

- Deleted – שומר את הרשומה האחרונה שנמחקה (שהביאה להתחלה הטריגר).
- If – הפעלת תנאים. רק למי שעומד בתנאי הערכים אכן ישונו ב景德 הנתונים ואם לא עומד בתנאים אז נבקש ביצוע rollback, כלומר שהערכים לא ישתנו.
- Else – הפעלת תנאי אשר לא התקיימ בפסקית ה-if.
- Declare – הצהרה על משתנים זמניים. דרך שימוש:

Declare @varName varType

cutet ניתן לבצע השמה למשנה ולהשתמש בו בכל מקום (תור שימוש ב-@ והשם שלו בכל מקום).

שאילתת trigger ראשונה

על מנת למנוע מרובוטים לענות על שאלות כדי לשבש את נתוני האפליקציה, יש לבדוק כי משתמש שעונה על שאלה אכן רשום לאפליקציה שאליה השאלה שייכת. לאחר מכן עוזר על השאלה (הכנסת התשובה) לא יתאפשר

שאילתת trigger ראשונה – הצהרה על משתנים והשימוש

```
create trigger checkIfRegister on  
answers after insert, update as
```

```
declare  
@email varchar(50),  
@appName nvarchar(20)
```

```
select @email=I.userEmail,  
@appName=I.appName  
from inserted I
```

שאילתת trigger ראשונה - בדיקת הכלל

העסק

if not exists

(select *

from RegisterdTo

where userMail=@email and

appName=@appName)

Rollback

שאילתת trigger שנייה

כאשר משתמש מסמן כי הוא אינו מעוניין במותאם מסוים, יש למחוק את המותאם מהרשימה של המשתמש ובמידה והוא מופיע בה

שאילתת trigger שלישית – הוצאה על משתנים והשנתם

```
reate trigger deleteUnwanted on  
doesntLike after insert as
```

```
declare  
@email varchar(50),  
@emailReferred varchar(50)
```

```
select @email=I.userEmail1,  
@emailReferred=I.userEmail2  
from inserted I
```

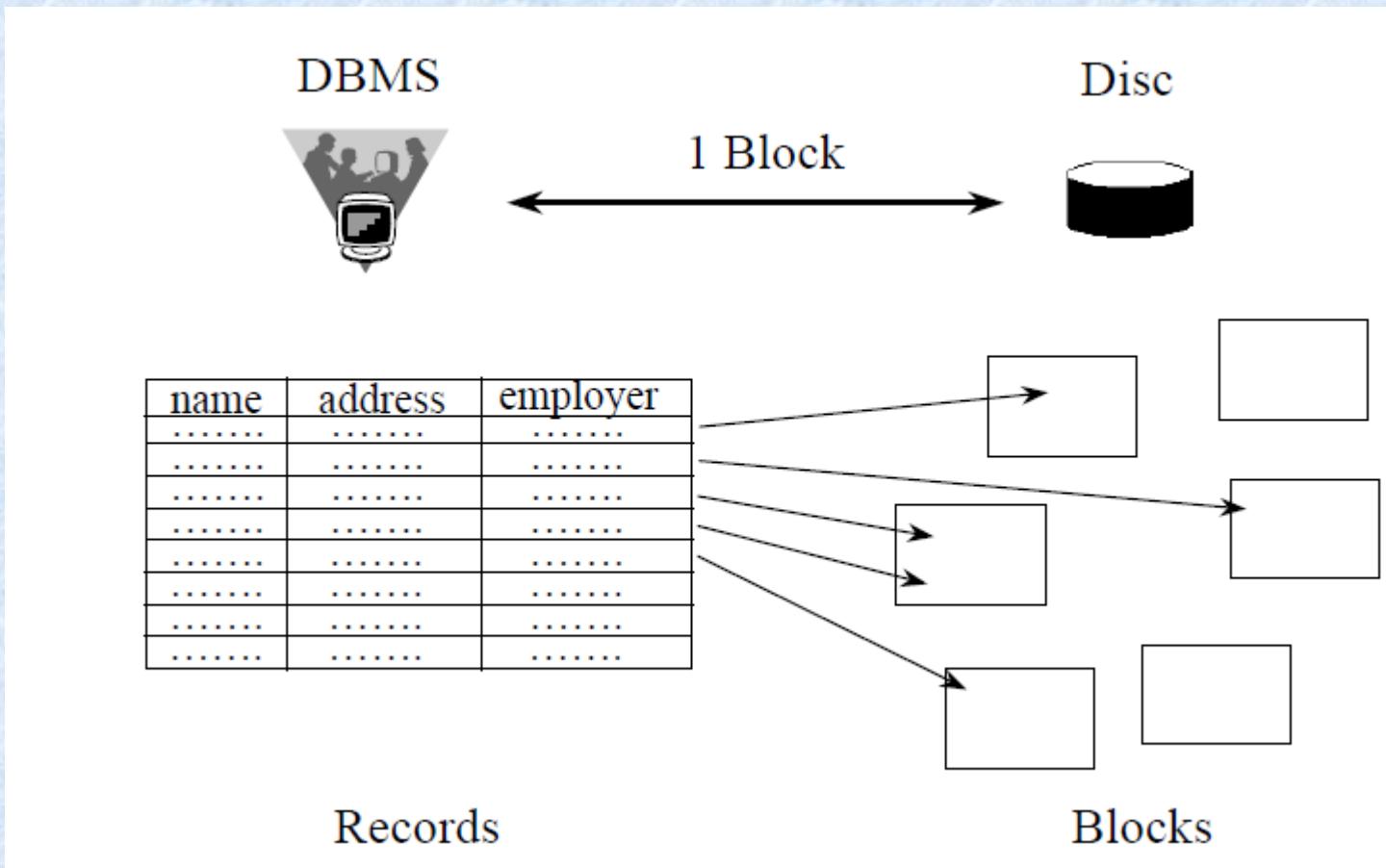
שאילתת trigger שלישית – בדיקת הכלל העסק

```
begin
delete from List
where userEmail1=@email and
userEmail2=@emailReferred
end
```

אינדקסים ו- QUERY PROCESSING

תכנון פיסי של מסדי נתונים: אינדקסים, חישוב עלות שאלות, אילוצים

אינדקסים, לשם מה?



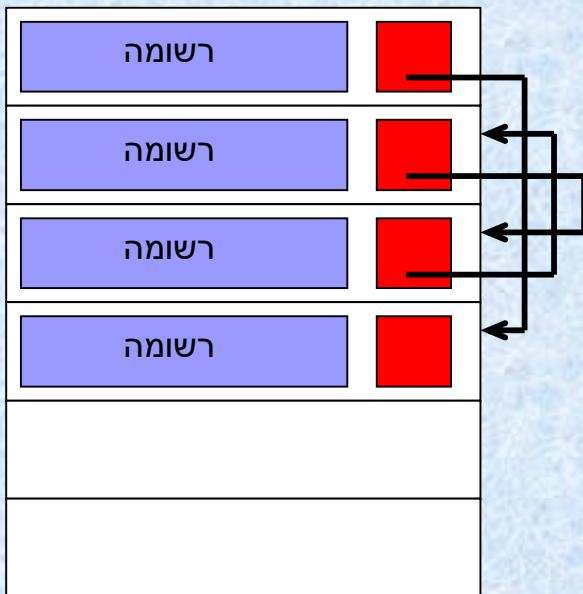
אינדקסים, לשם מה? (המשך)



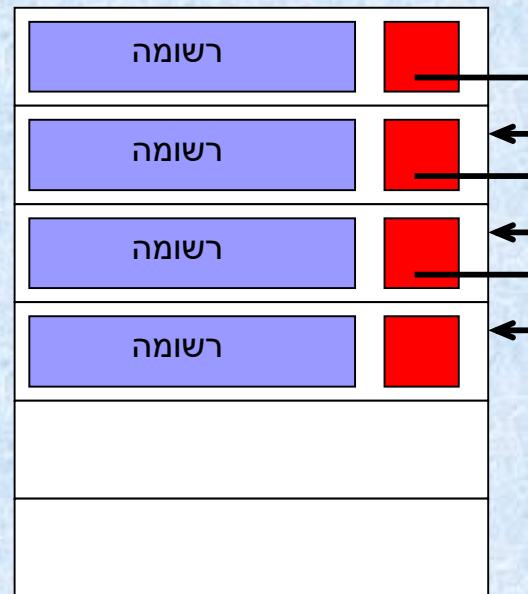
- הדיסק הינו המקום בו נשמרים הנתונים של מוד הנתונים.
 - הדיסק מחולק לבלוקים.
- הזיכרון הראשי הינו המקום בו ניתן לעבוד נתונים.
- בלוק הינו היחידה המינימאלית אותה ניתן להעביר מהדיסק אל הזיכרון או לכתוב מהזיכרון לדיסק.
- הבאת בלוק מהדיסק אל הזיכרון ו כתיבת בלוק מהזיכרון לדיסק הין פועלות יקרות מאד מבחינה זمان חישוב.
- אינדקס "יביא" רק את הבלוקים בהם נמצאים הרשומות הרלוונטיות.

אינדקסים, לשם מה? (המשך)

מבנה בLOC בDIOSK



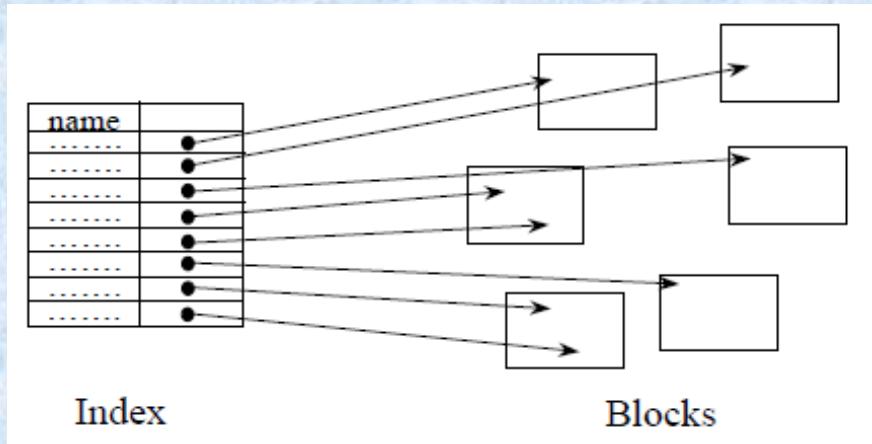
רשומות בLOC ממוגנות עפ"י
שדה כלשהו



רשומות בLOC

אינדקסים, לשם מה? (המשך)

- השימוש באינדקסים נועד כדי להאריך זמן גישה לנ נתונים
- קבצי אינדקס הינם קטנים יותר מקובץ הנתונים
- אינדקסים מאפשרים חיפוש מהיר של רשומות על פי ערכי שדה או שדות.
- שימוש באינדקסים משפר זמן ביצוע של שאלות במחיר של מקום על הדיסק ועלות בעת הכנסה, מחיקה, או עדכון של רשומות.



An index file

Search Key

Pointer

- רשומה אינדקס הינה מהצורה:
 - על איזה שדה/ שדות כדאי להגדיר אינדקס?
 - שאלות על ערך מסוים
 - שאלות על תחום ערכים
- זמן גישה לנוטונים – כמה זמן לוקח להחזיר תוצאות
שאיילתא? כמה תקיפות וڌיפות משתמשים בה?
- זמן הכנסת רשומה – כל כמה זמן מכניםים/ מעדכנים
רשומות בטבלה?
- זמן מחיקת רשומה – כל כמה זמן מוחקים רשומות
מהטבלה?
- מקום פיסי – כמה מקום פיסי תופסת הטבלה? כמה
מקום פיסי יתפס האינדקס?

מתי להגדיר אינדקסים?



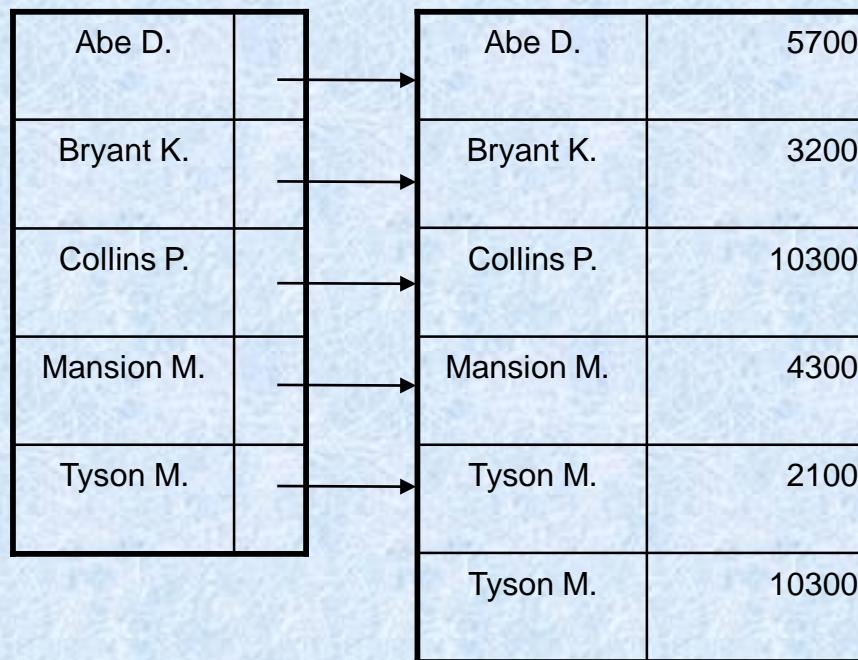
- כאשר כמות הנתונים בטבלאות רבה
- עברו מפתחות ראשיים של כל טבלה
- עברו שדות שיש חיפוש רב לפיהם (ב- WHERE clause)
- שדות עבורם יש שימוש ב- ORDER BY clause ו- GROUP BY clause
- כאשר יש לפחות 100 ערכים שונים (ולא כאשר יש פחות מ- 30 ערכים שונים)
- תוך התחשבות במספר האינדקסים המקיים לכל טבלה (המודגר בהתאם ל- DBMS)
- מיעוט ערכי null (אם בכלל) בשדות אינדקס
- אינדקסים עדיפים (יעילים) במקרים שאינם משתנים בתכיפות גבוהה

- **קובץ אינדקס סדרתי ממוקן לפי ה- Search Key.**
 - קובץ הנתונים ממוקן לפי שדה או שדות מסוימים (בד"כ לפי המפתח הראשי).
- **באינדקס ראשי ה- Search Key הינו השדה (השדות) על פיו (פיהם) ממוקן קובץ הנתונים.**
 - כל אינדקס אחר הינו **אינדקס שני**.
- **אינדקס יכול להיות**
 - **צפוף** - מכיל רשומת אינדקס עבור כל ערך קיים של ה- Search Key בקובץ הנתונים.
 - **دلיל** – אינדקס שאינו צפוף.
 - רק אינדקס ראשי יכול להיות דليل !!!
 - אינדקס ראשי יכול להיות צפוף או דליל, אינדקס שני יכול להיות רק צפוף

אינדקסים – מושגים (המשך)

- נסתכל על הטבלה הבאה:
 - ניתן לבנות אינדקס ראשי צפוף לפי השדה הראשון (על פי ממויין קובץ הנתונים)

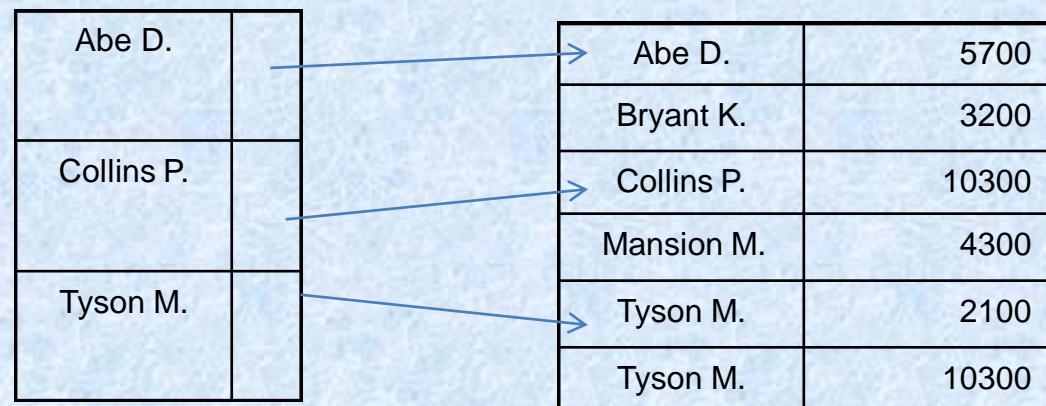
| | |
|------------|-------|
| Abe D. | 5700 |
| Bryant K. | 3200 |
| Collins P. | 10300 |
| Mansion M. | 4300 |
| Tyson M. | 2100 |
| Tyson M. | 10300 |



אינדקסים – מושגים (המשך)

- ניתן לבנות אינדקס ראשי דليل לפי השדה הראשוני:

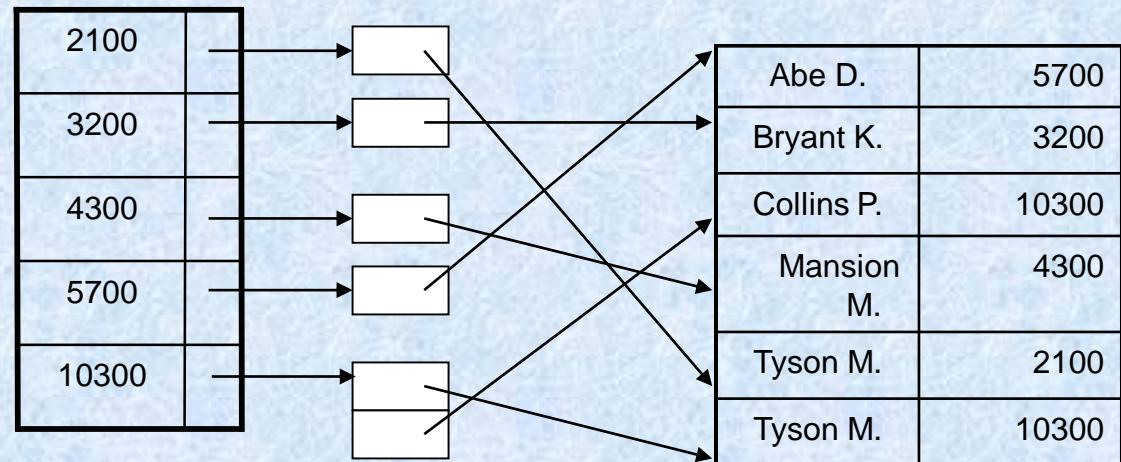
| | |
|------------|-------|
| Abe D. | 5700 |
| Bryant K. | 3200 |
| Collins P. | 10300 |
| Mansion M. | 4300 |
| Tyson M. | 2100 |
| Tyson M. | 10300 |



אינדקסים – מושגים (המשך)

- ניתן לבנות אינדקס משני צפוף לפי השדה השני:

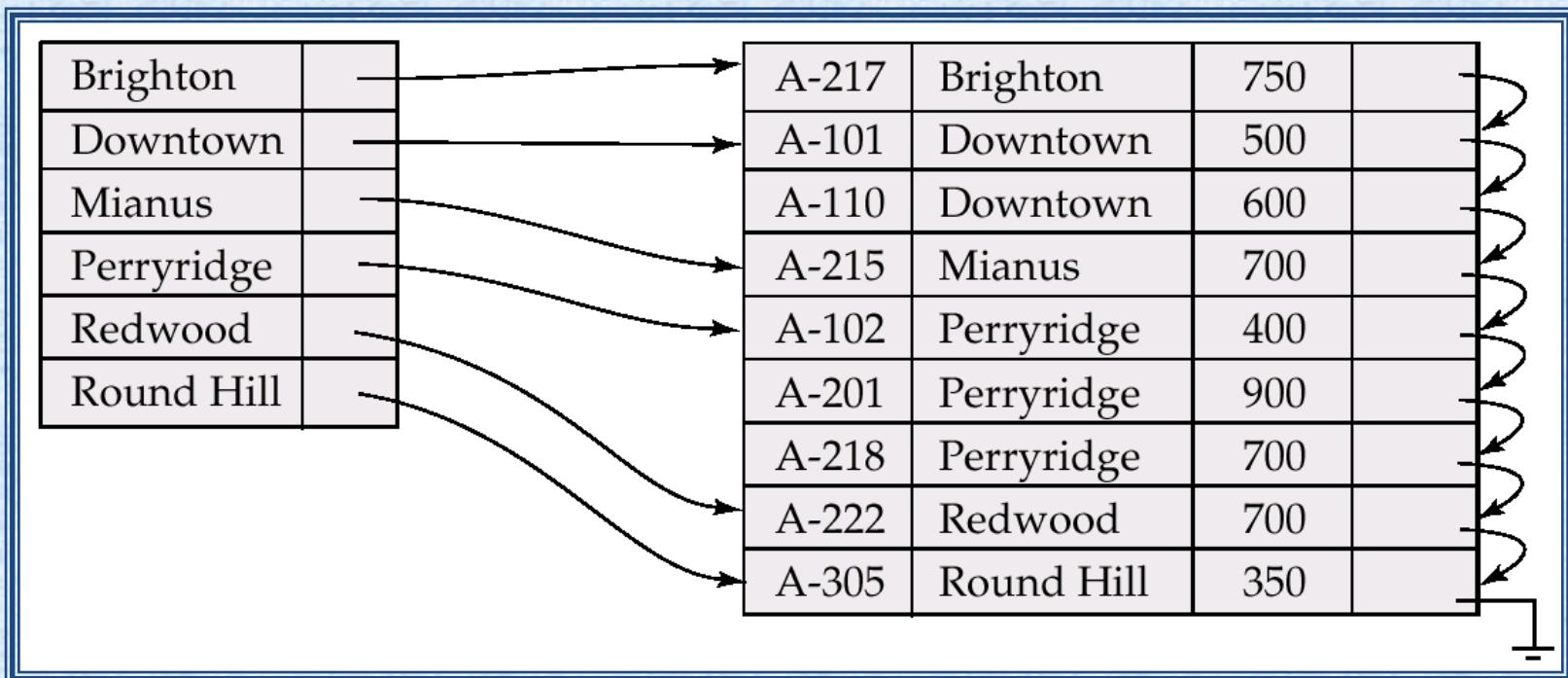
| | |
|------------|-------|
| Abe D. | 5700 |
| Bryant K. | 3200 |
| Collins P. | 10300 |
| Mansion M. | 4300 |
| Tyson M. | 2100 |
| Tyson M. | 10300 |



- קובץ הנתונים הוא יחיד ועל אותו קובץ ניתן להגדיר מספר קבועי אינדקס

אינדקס צפוף

- **אינדקס צפוף** מכיל רשומת אינדקס עבור כל ערך קיים של ה-
 בקובץ הנתונים Search Key



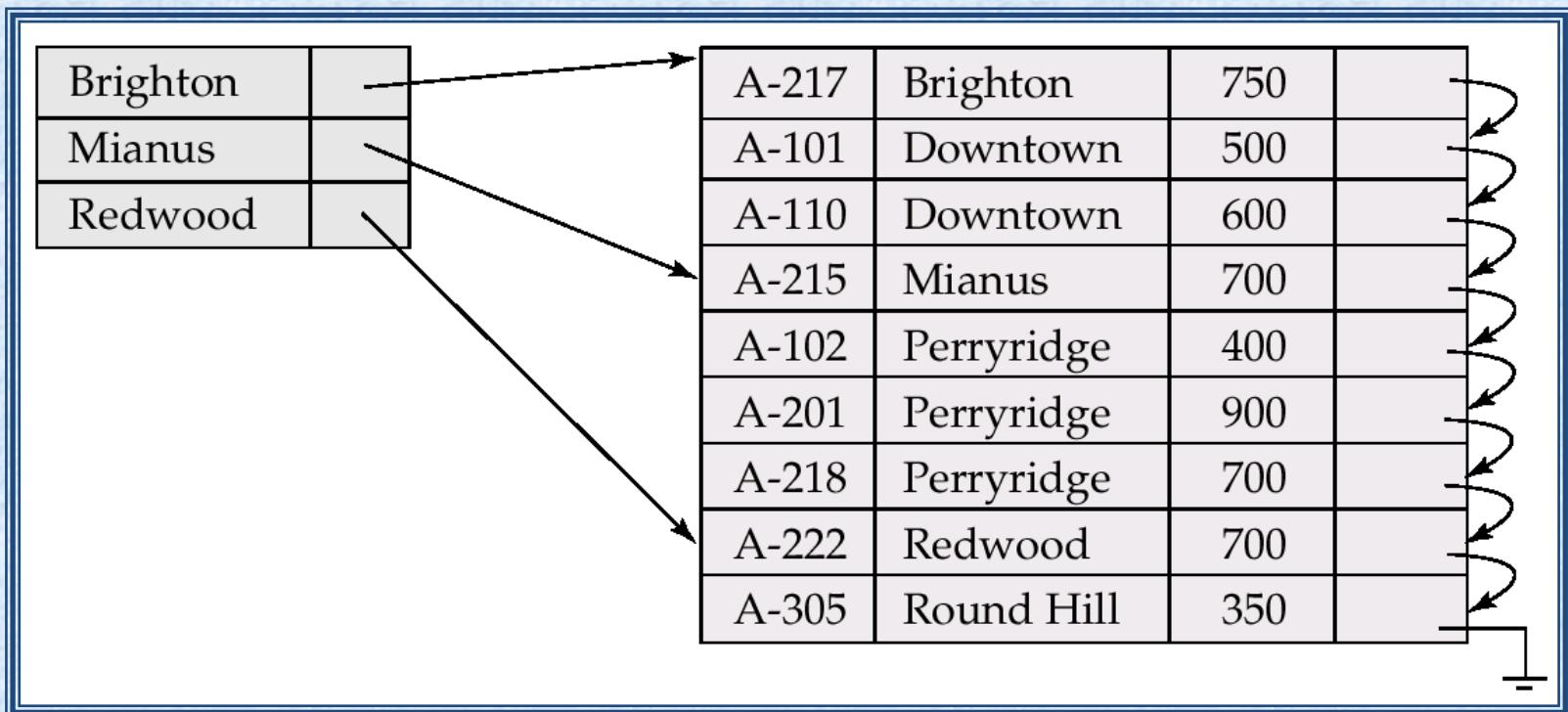
המפתח הראשי של הטבלה הוא מספר חשבון. האינדקס לא מוגדר על המפתח הראשי במקרה זה. למה?

אינדקס צפוף (המשר)

- **כיצד מתבצעת הכנסת רשומה?**
- מוחפשים את ה `search key` של הרשומה המוכנסת בקובץ הנתונים, אם לא נמצאה נכניס אותה גם לקובץ האינדקס ונדעכו את המצביע (הכתובת של הרשומה המוכנסת)
- **כיצד מתבצעת מחיקת רשומה?**
- אם הרשומה הנמחקת היא הרשומה היחידה עם ה `search key` המופיעים אז נמחק את האינדקס הספציפי מקובץ האינדקס
- אם לא, לא נבצע מחיקה של האינדקס אך בעת הצורך (אם האינדקס מצביע על הרשומה הנמחקת) נעדכו את הכתובת של האינדקס לרשומה הבאה)

אינדקס דלייל

- **אינדקס דלייל** מכיל רשימת אינדקס עבור ערכים נבחרים של ה-
 Search Key בקבוץ הנתונים.

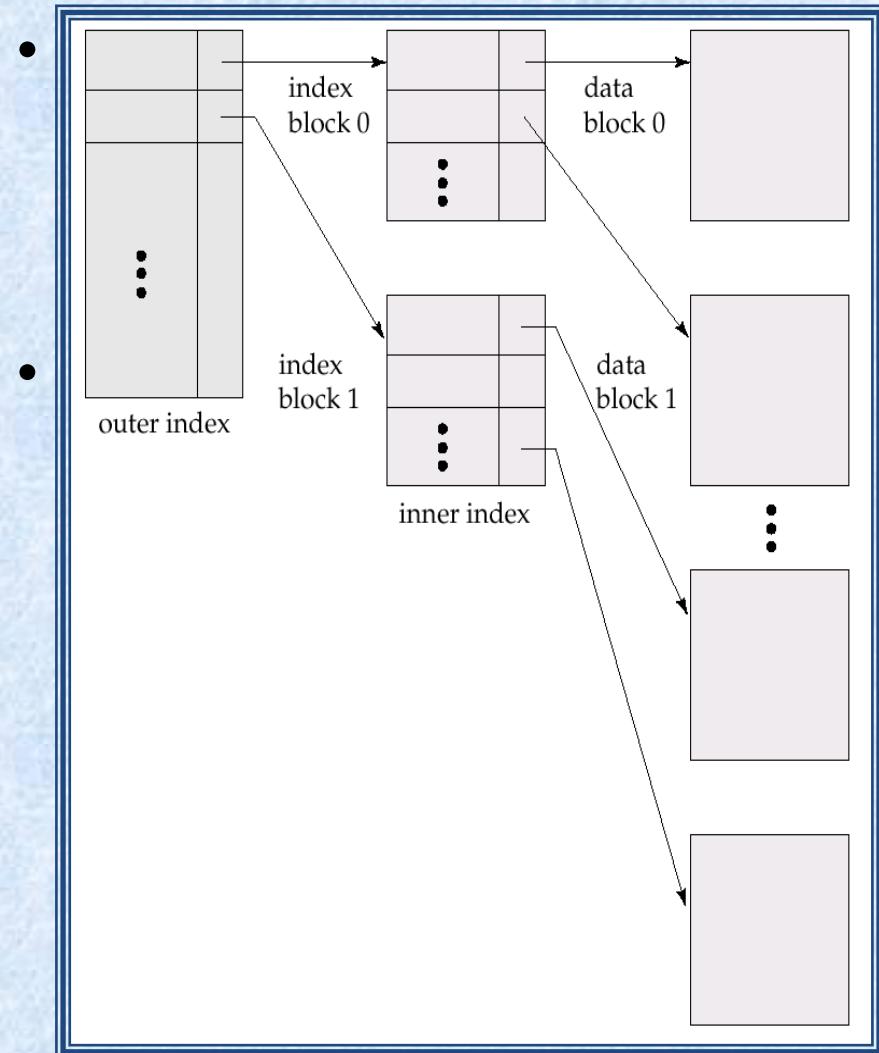


אינדקס דليل (המשר)

- **כיצד מתבצעת הכנסת רשומה?**
 - מכניסים את הרשימה למקום המתאים, אם אין צורך בבלוק נוסף בזיכרון, לא עושים כלום
 - אם כתוצאה מהכנסת הרשימה נדרש בלוק נוסף, יש לעדכן את קובץ האינדקס כך שיכיל אינדקס עם ה search key של הרשימה הראשונה בבלוק שנוצר.
- **כיצד מתבצעת מחיקת רשומה?**
 - אם הרשימה הנמחקת היא הרשימה עם ה search key המופיעים אז נמחק את האינדקס הספציפי מקובץ האינדקס וחתת תוחלף ע"י ה search key הבא (שנקבע עפ"י מבנה הנתונים בו שומר קובץ האינדקס)

אינדקס רב שכבותי

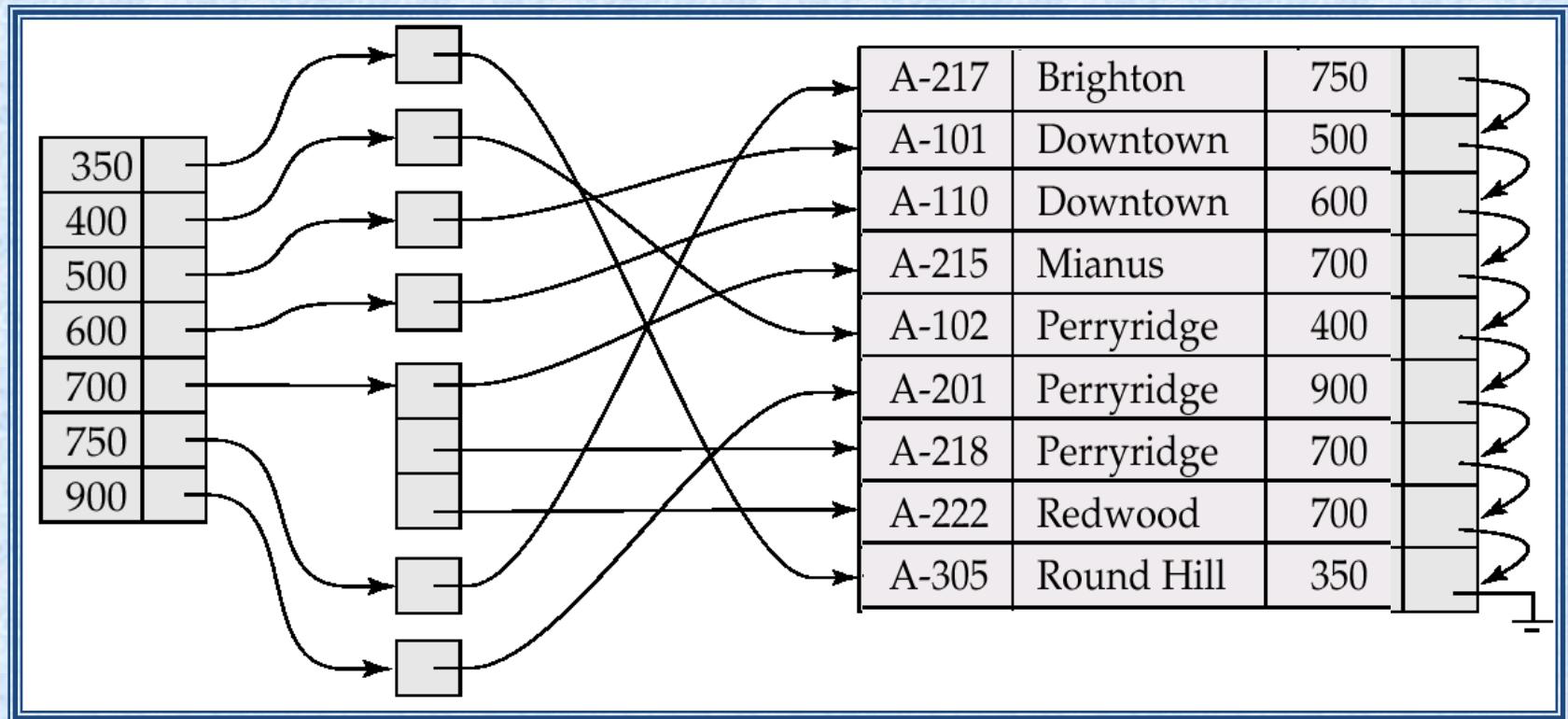
- אם רמה אחת של אינדקס לא נכנסת לזיכרון, נגיד:
 - אינדקס דليل ברמה החיצונית
 - אינדקס צפוף ברמה הפנימית
- האינדקסים בכל הרמות צריכים להיות מעודכנים בעת הכנסה ומחיקה של רשומה



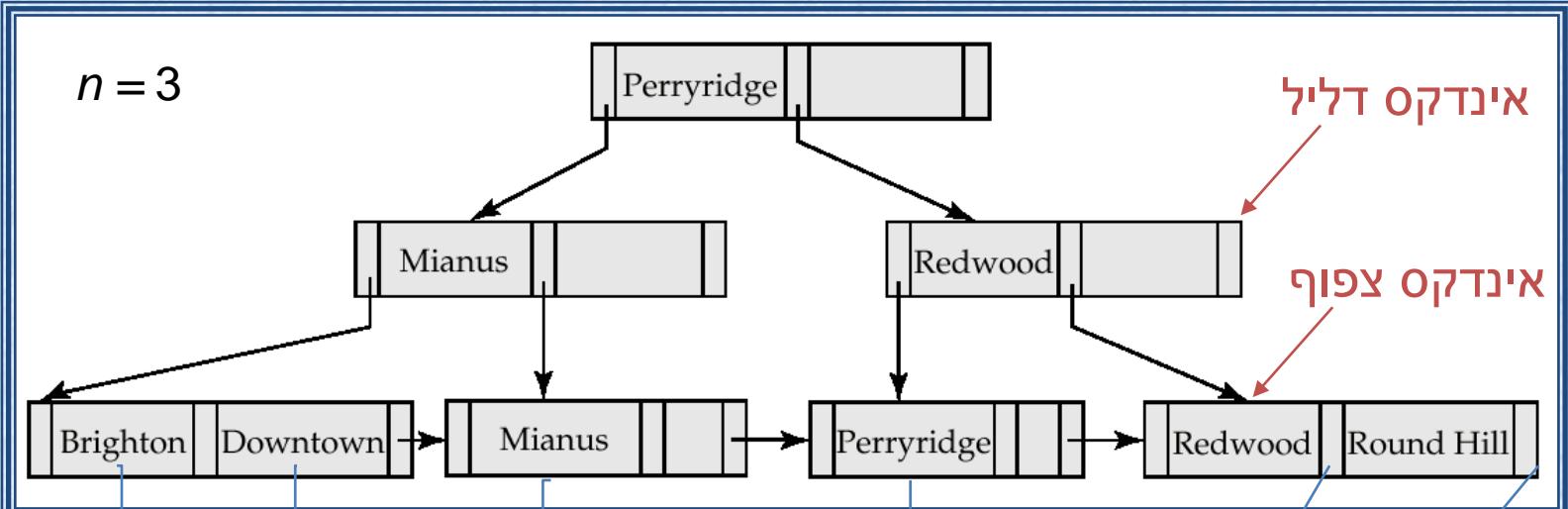
אינדקס שני, לשם מה?

- לעתים, אנו מעוניינים לאתר רשומות לפי שדה שאינו המפתח/ הטבלה אינה ממוינת עפ"י שדה זה
 - דוגמה 1: שליפת כל החשבונות בסניף Brooklyn מטבלת account שבה המפתח הראשי הוא soh_account
 - דוגמה 2:כנ"ל, כאשר מעוניינים לשולוף את כל החשבונות עם יתרה (או טווח יתרות) מסוים.
- לצורך זה, ניתן להגדיר אינדקס שני.
- אינדקס שני, הוא אינקדס שלא על פי (על פי מפתח החיפוש שלו) ממויין קובץ הנתונים

דוגמא לאינדקס משני (על balance)



- אינדקסים שניים חייבים להיות צפופים (מצבייע לכל רשומה בטבלה)
- מעבר על כל רשומות הקובץ תור שימוש באינדקס שני איננה יquila
(במהשך)

$n = 3$ 

עצי + הם קבצים שמאורגנים כך
שמהווים עצי חיפוש מאוזנים

| | | | |
|-------|------------|-----|--|
| A-217 | Brighton | 750 | |
| A-101 | Downtown | 500 | |
| A-110 | Downtown | 600 | |
| A-215 | Mianus | 700 | |
| A-102 | Perryridge | 400 | |
| A-201 | Perryridge | 900 | |
| A-218 | Perryridge | 700 | |
| A-222 | Redwood | 700 | |
| A-305 | Round Hill | 350 | |

- עץ שבו כל המסלולים מהעלים אל השורש הם באותו אורך.
- כל צומת שאינו השורש מכילה בין $\lceil \frac{n}{2} \rceil$ ל- $\lceil \frac{n}{2} \rceil$ מצביעים, כאשר n קבוע לעץ $+B$ נתון
 - **למשל:** עבור $n=7$, מספר הבנים הוא בתחום $[4,7]$ (**מעגלים מעלה**)
- השורש והעלים מכילים לכל היותר n מצביעים.
- לצומת בעלת n מצביעים יש $1 - n$ מפתחות חיפוש (search keys) והם **מוויינים בסדר עולה**.
- המצביעים בעליים מצביעים לרשותם (במקרה שהאינדקס הראשי) או לדליים (במקרה שהאינדקס משתני).

- בדרכו כלל נהוג לבחור את גודל הצומת בהתאם לגודל בלוק על הדיסק. (בלוק=צומת)
- בעץ + B בעל K מפתחות חיפוש יש לעבור על לכל היותר $\lceil \log_{n/2} \rceil$ צמתים עד למציאת הרשומה המבוקשת.
- אם, לדוגמה, $n=100$ ו- $k=1,000,000$ אז יש לעבור על לכל היותר 4 צמתים עד למציאת הרשומה המבוקשת.
- החיסרון בעעץ + B הוא בהזובź במקום (במקרה שבצומת יש פחות מ - n מצביעים) ובעיקר העובדה שהכנסה/מחיקה/עדכון של רשומות דורשת פעולות נוספות.

עצי + B (המשר)

- **מקרים מיוחדים**
 - לשורש שאינו עליה יש בין 2 ו-ח בניים
 - לשורש שהוא עליה (צומת יחיד בעז) יש בין 0 ל-1-n ערכאים
- האינדקסים בכל הرمאות צריכים להיות מעודכנים בעת הכנסה או מחיקה של רשומה (ולעתיתים גם בעת עדכון).

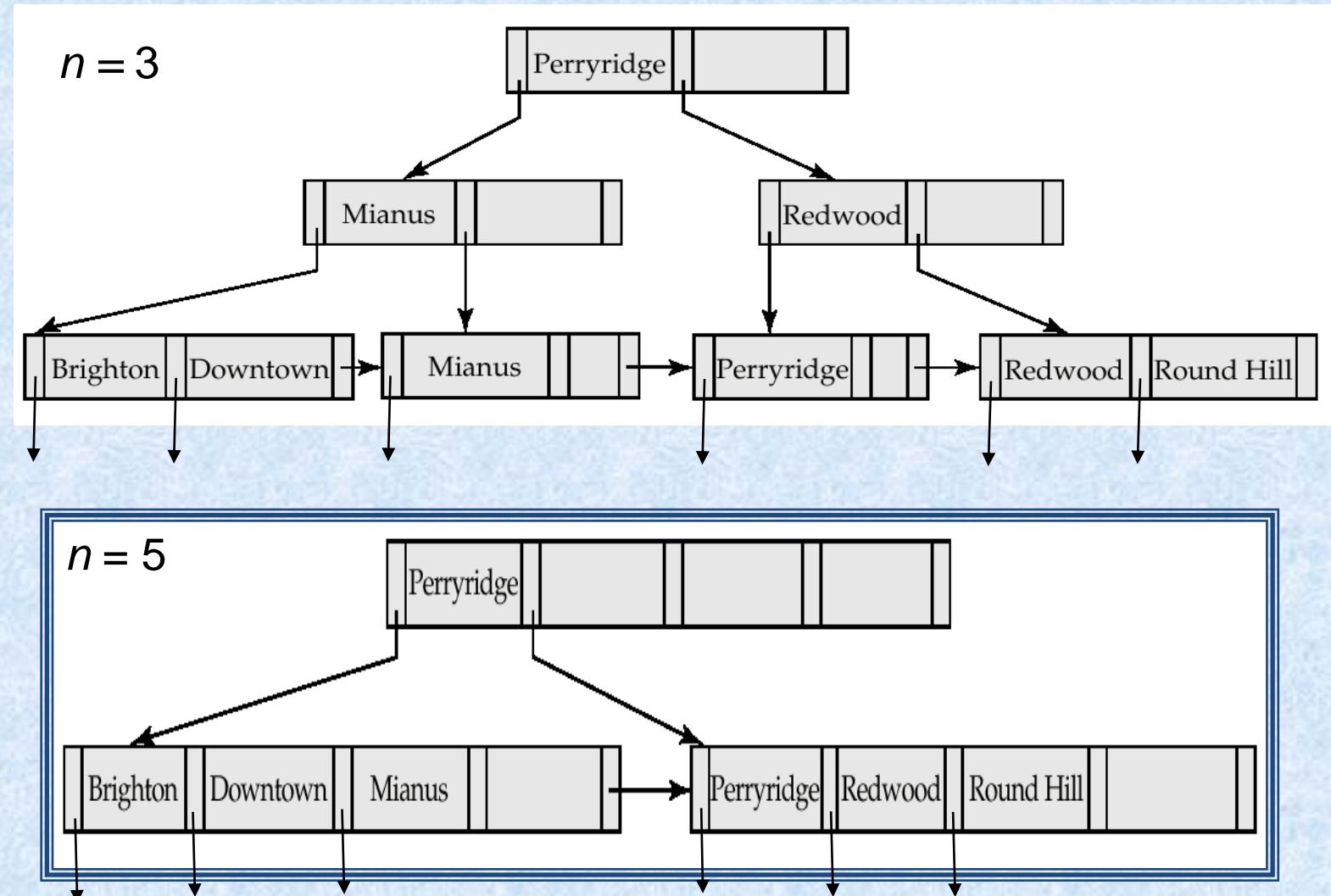
מבנה של צומת בעץ +

- **מבנה טיפוסי**

| | | | | | | |
|-------|-------|-------|-----|-----------|-----------|-------|
| P_1 | K_1 | P_2 | ... | P_{n-1} | K_{n-1} | P_n |
|-------|-------|-------|-----|-----------|-----------|-------|

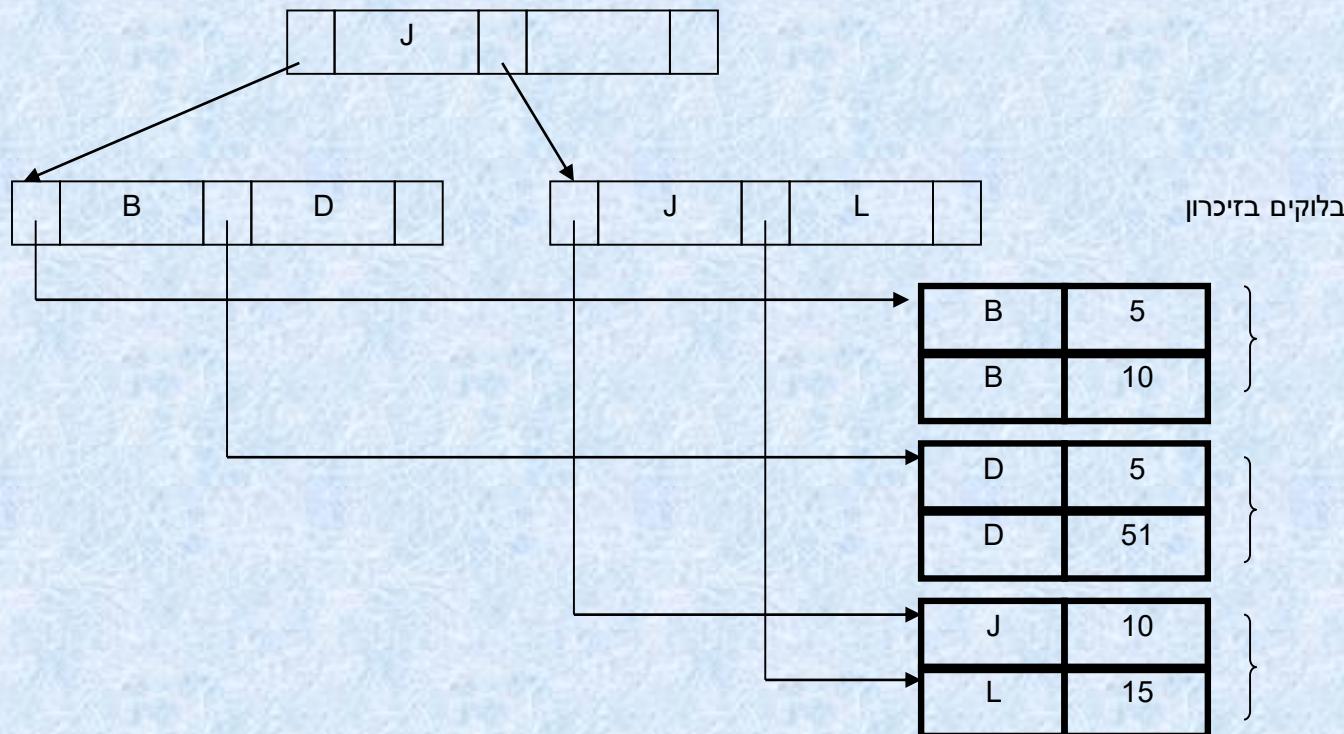
- אם ערכי מפתחות החיפוש (key) הערך על פי מוחשיים
- הם מצביעים לבנים (בצמתים שאינם עליים) או מצביעים לרשומות (או לדליים) (בצמתים שהנם עליים)
- סדר ערכי מפתחות החיפוש הנם בסדר עולה כך ש
$$K_1 < K_2 < K_3 < \dots < K_{n-1}$$
- **דלי (bucket)** הוא ייחdet שמירה המכילה מס' רשומות או מס' מצביעים לבлок
 - הגודל של דלי הוא בדרך כלל בלוק

דוגמאות לעצי B+



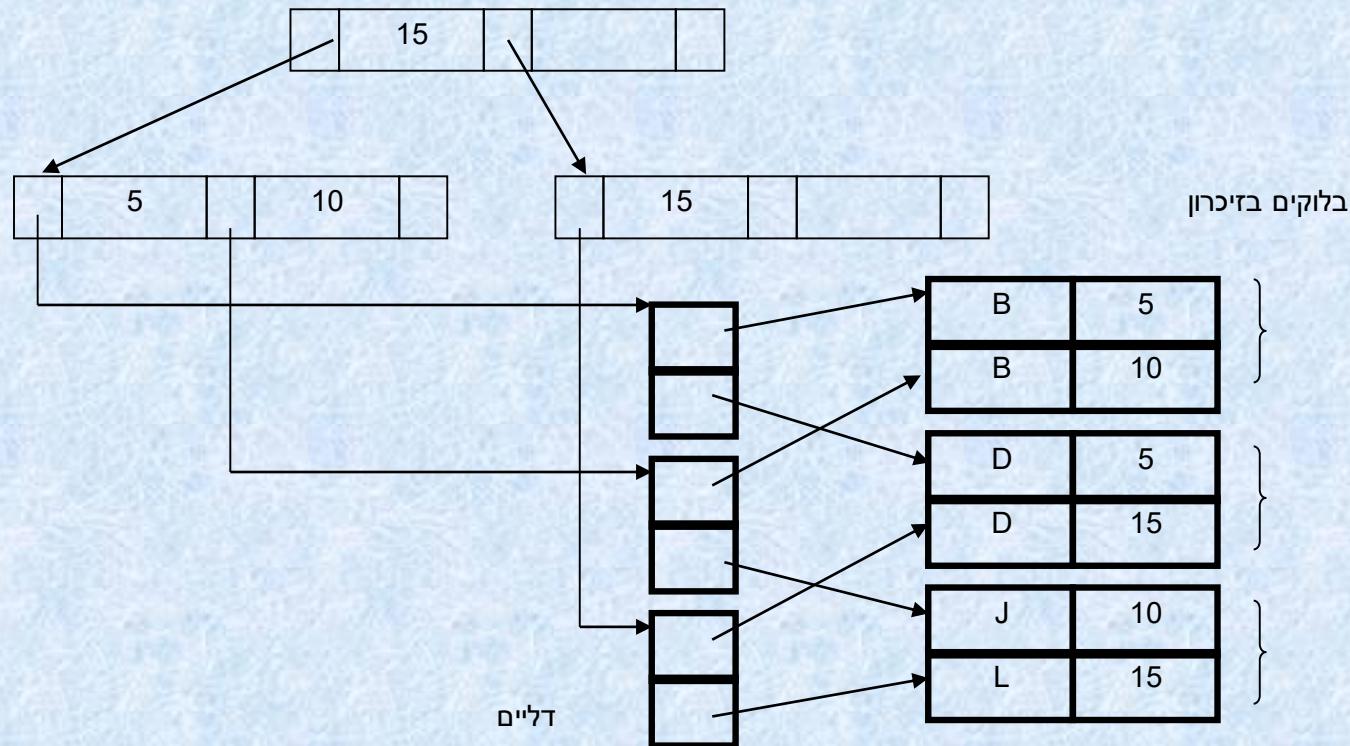
דוגמאות לעצי B+ (המשר)

- דוגמא לעץ B+ (אינדקס ראשי, קלומר קובץ הנתונים
ממוין עפ"י ה key (search key עם $n=3$):



דוגמאות לעצי + B (המשר)

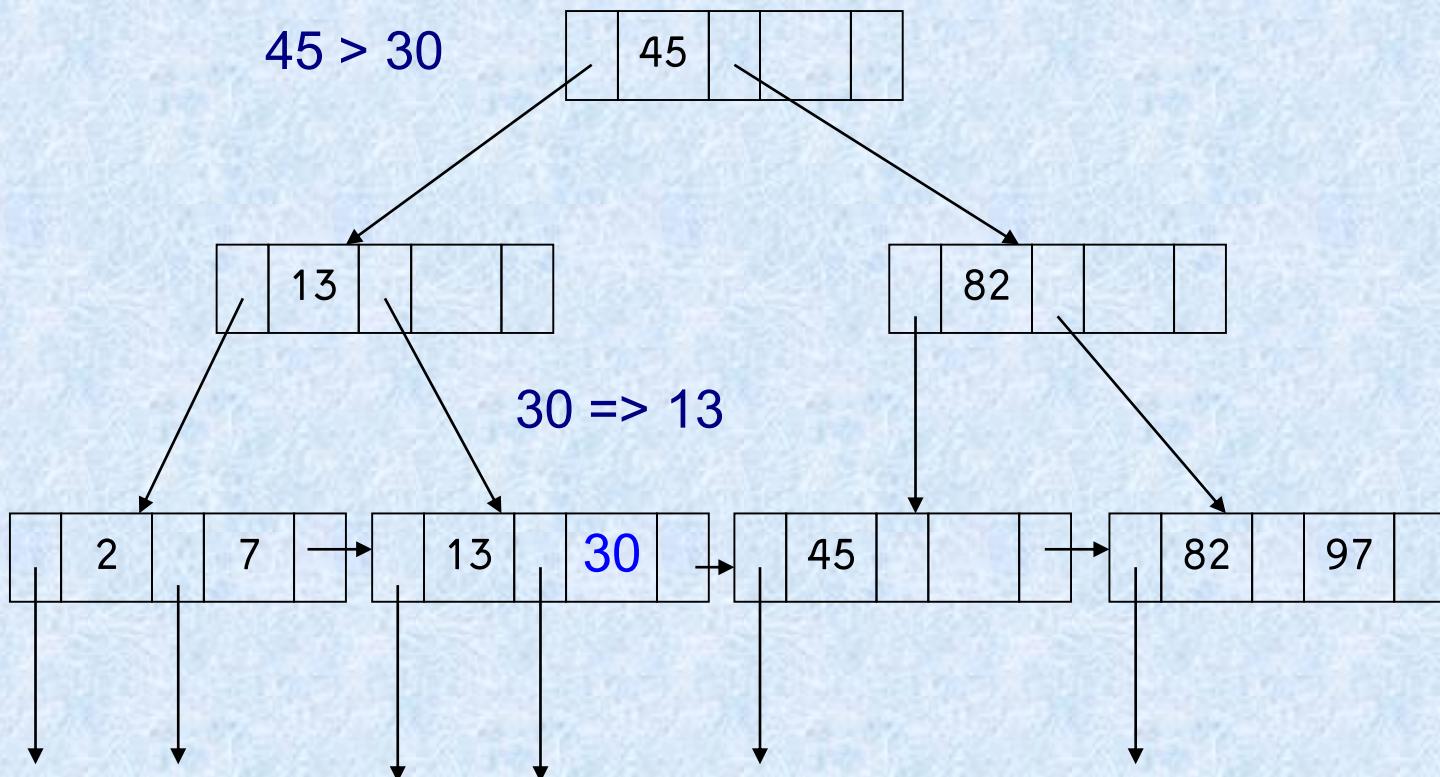
- דוגמא לעץ + B (אינדקס שני, קלומר קובץ הנתונים אינו ממוקן עפ"י ה key עם $n=3$):



- כדי למצוא את כל הרשומות שפתח החיפוש שליהם הוא k :
 - התחל בשורש
 - מצא בצומת את הערך בעל מפתח החיפוש הקטן ביותר שגדול מ- k
 - אם קיימן ערך כזה (K_i), עקוב אחר P_i
 - אחרת ($k \geq K_{m-1}$) עקוב אחר P_m
- כאשר הגיענו לעלה, אם עברו איזשהו i , $k = K_i$ עקוב אחר P_i לבלוק/לDLL (bucket) אחרת, לא קיימת רשומה עם הערך k בפתח החיפוש

חיפוש בעצי B (המשר)

- נחפש בעץ רשומה עם הערך 30:

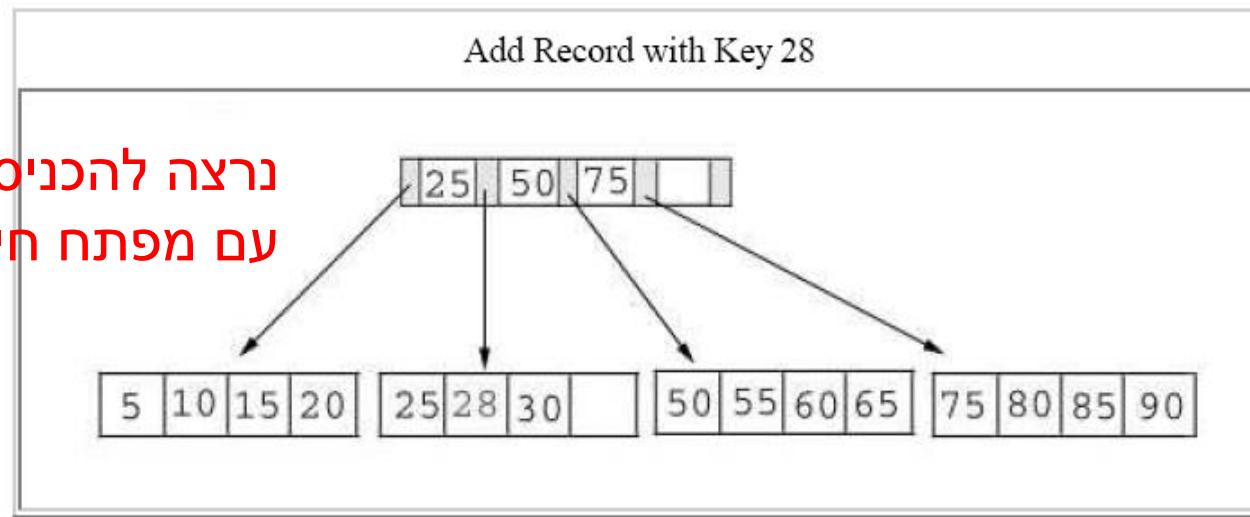
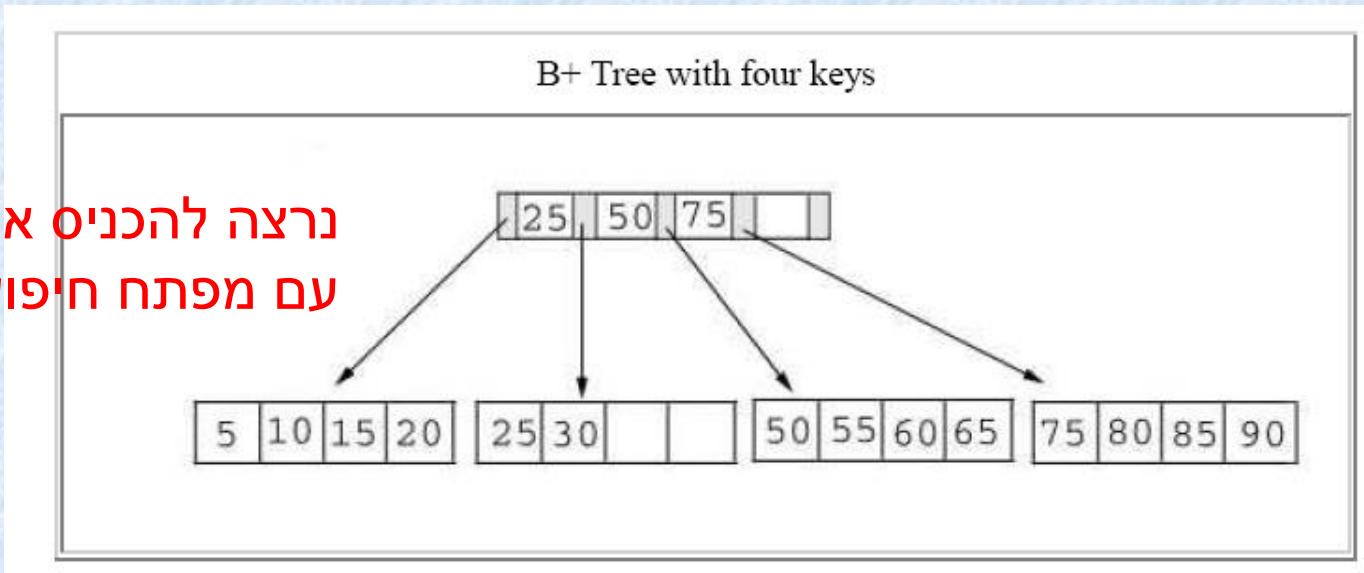


- מצא את העלה שבו מפתח החיפוש צריך להופיע (השווות עפ"י ערכו של מפתח החיפוש).
- אם מפתח החיפוש נמצא כבר בעלה, מכניסים את הרשומה לקובץ ובמידת הצורך (אינדקס משנה) מוסיפים לדלי המתאים מצביע עליה.
- אם מפתח החיפוש לא נמצא, מכניסים את הרשומה לקובץ ובמידת הצורך (אינדקס משנה) מוסיפים לדלי חדש עם מצביע לרשותה. אח"כ:

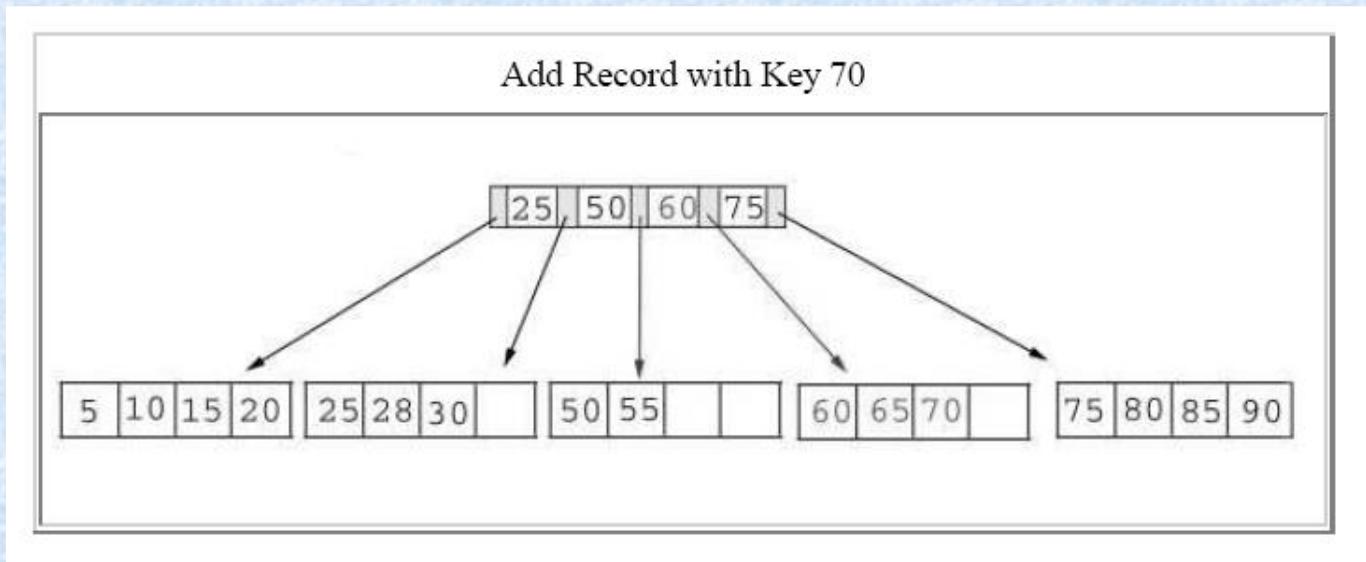
הוספת רשומה ועדכו עז + B

- אם קיים מקום לעלה למפתח חיפוש נוסף, מוסיפים לעלה את הזוג (search key, pointer)
- אחרת (אין מקום), מפיצלים את העלה (הכולל את הכניסה החדשה במקום המתאים) **באופן הבא:**
 - המפתח האמצעי עולה למעלה ומוכנס לאב במקום המתאים $\lceil \alpha/2 \rceil$ הערכים הראשונים נשאים נשאים בצומת המקורי והשאר בצומת חדש.
 - אם הצומת המפוצל אינו עלה, אז המפתח האמצעי (שעליה לאב) לא יופיע באחד מהצמתים שנוצרו בפיצול.
 - אם צומת האב מלא, נמשיך לפצלו (במקרה הגורע ביותר נפצל את השורש).

הוספת רשומה ועדכו עץ B+ (המשר)



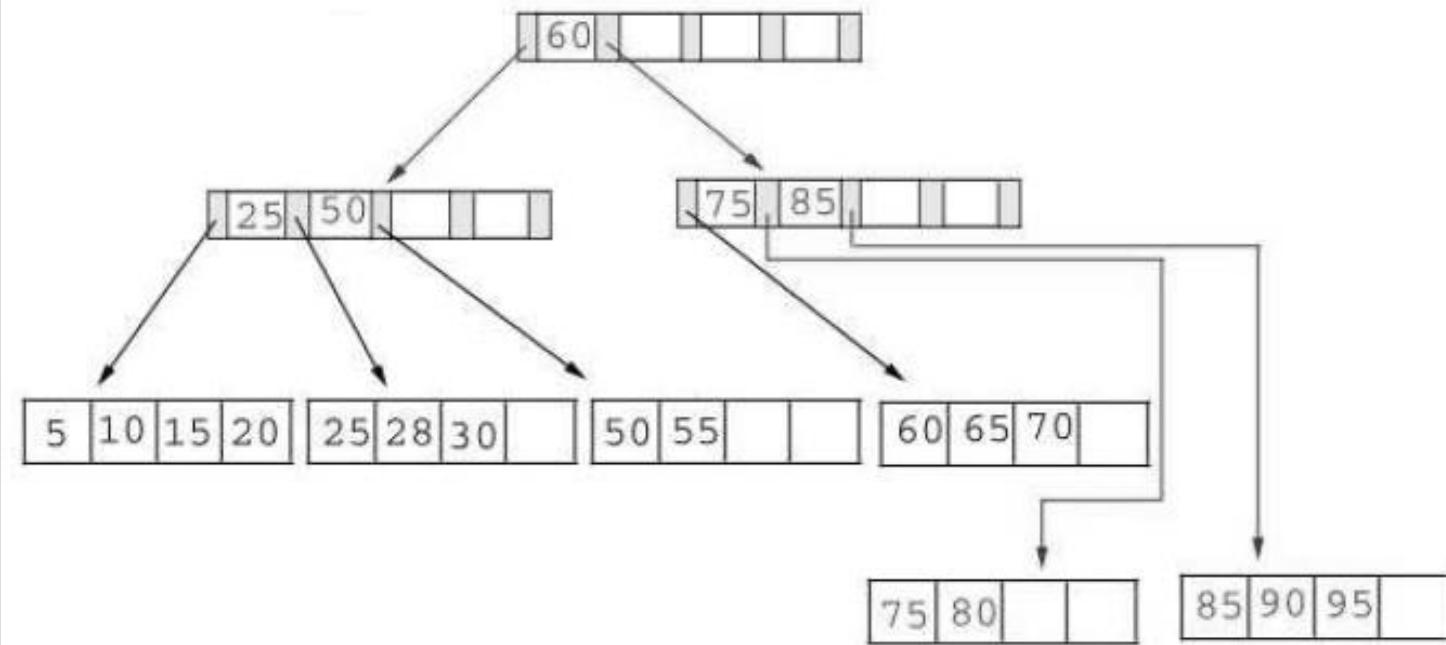
הוספת רשומה ועדכו עץ B+ (המשר)



נרצה להכניס את רשומה
עם מפתח חיפוש 95

הוספת רשומה ועדכון עץ B+ (המשר)

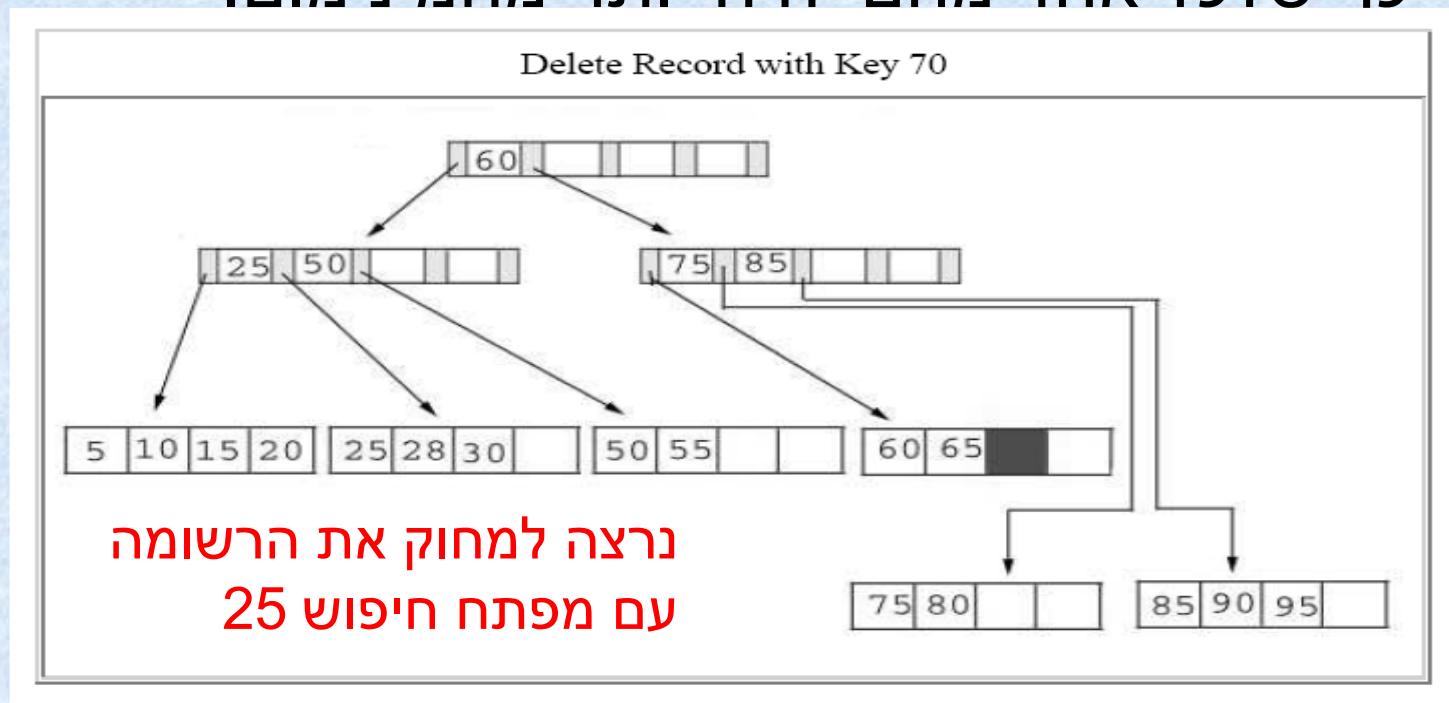
Add Record with Key 95



- מצא את הרשומה למחיקה מחק אותה מהקובץ
ואת המצביע עליו מהدلוי (אם אינדקס משני).
- מחק את הזוג (key, pointer) מהעליה אם אין דלי
או שהدلוי ריק.
- אם בעקבות הממחיקה לצומת יש פחות מדי ערכי
מפתח ($\lceil \alpha/2 \rceil$) וניתן לאחד אותו עם צומת אח
לצומת יחיד, אז:
 - הכנס את כל ערכי מפתח החיפוש לצומת היחיד
(השמאלי) ומחק את הצומת השני.
 - מחק את הזוג (key_i, p_{i-1}) מהאב באופן רקורסיבי,
casr k מצביע לצומת שנמחק.

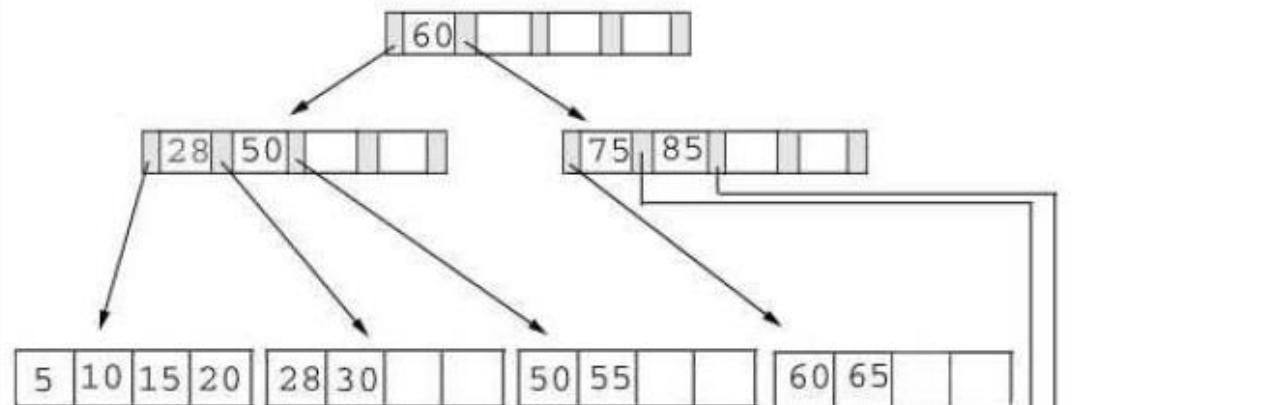
מחיקת רשומה ועדכו עץ B+ (המשר)

- לאחרת (לצומת יש פחות מדי ארכ לא ניתן לאחדו עם אח):
 - חלק מחדש את ערכי מפתחות החיפוש בשני הצמתים, כרך שלכל אחד מהם יהיה יותר מהמינימום.



מחיקת רשומה ועדכו עץ B+ (המשר)

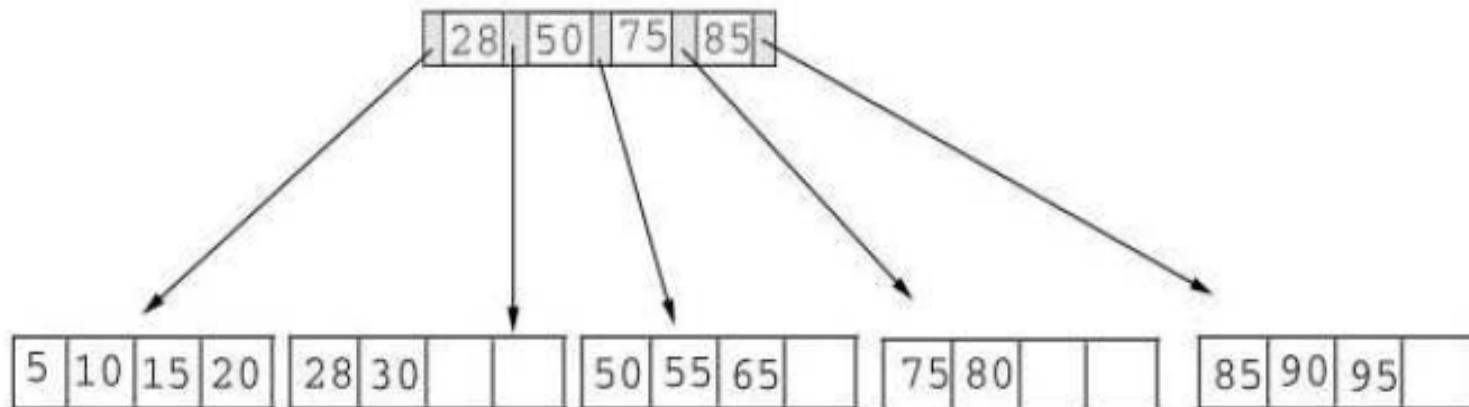
Delete Record with Key 25



נרצה למחוק את הרשומה
עם מפתח חיפוש 60

מחיקת רשומה ועדכון עץ B+ (המשר)

Delete Record with Key 60



עצי + B - תרגיל כתה



- נתונות הרשומות החלקיות הבאות של טבלת Teacher :tblTeacher

| teacherNum | surName | city | mainStyleId |
|------------|----------|-------------|-------------|
| 1 | דודיה | חיפה | 20 |
| 2 | דביר | ת"א | 30 |
| 3 | אמדורסקי | ירושלים | 25 |
| 4 | אמדורסקי | בת ים | 25 |
| 5 | דביר | פתח תקווה | 30 |
| 6 | לי | רשל"צ | 12 |
| 7 | תדמור | חיפה | 14 |
| 8 | בת שבע | באר שבע | 18 |
| 9 | פיק | קריית שמונה | 22 |
| 10 | גולן | ת"א | 33 |
| 11 | קשת | ת"א | 34 |
| 12 | פלמן | מטולה | 23 |
| 13 | סוויז' | אשקלון | 25 |
| 14 | כהלון | אשדוד | 15 |
| 15 | בומר | ת"א | 16 |
| 16 | ארבטובה | קריית ים | 25 |
| 17 | פלמן | אשדוד | 28 |

עצי + B - תרגיל כתה

1. הכניסו את הרשומות (בסדר המופיע בטבלה) לקובץ סדרתי המופיע לפני המפתח הראשי תוך יצירת אינדקס מסוג עץ B+ עם $3=3$ על השדה City. הציגו את הנתונים בעץ, במבנים הנלוויים (אם יש) ובקובץ הנתונים לאחר כל שלב. ה nicho כי כל דלי מכיל עד 3 מצביעים.
2. צינו את הנתונים בעץ, בקובץ ובמבנים הנלוויים (אם יש) לאחר כל אחת מהמתקומות הבאות (בסדר זה).

DELETE FROM tblTeacher

WHERE surName BETWEEN 'א' and 'ג'

DELETE FROM tblTeacher

WHERE mainStyleId<=15



הגדרת אינדקסים ב- SQL

- יצרת אינדקס:

```
create index <index-name> on <relation-name>  
          (<attribute-list>)
```

- דוגמא:

```
create index b-index on branch(branch-name)
```

- ליצירת אינדקס שאינו מאפשר כפילויות:

```
create unique index <index-name> on <relation-name>  
          (<attribute-list>)
```

- למחיקת אינדקס:

```
drop index <index-name>
```

הגדרת אינדקסים ב- SQL (המשך)

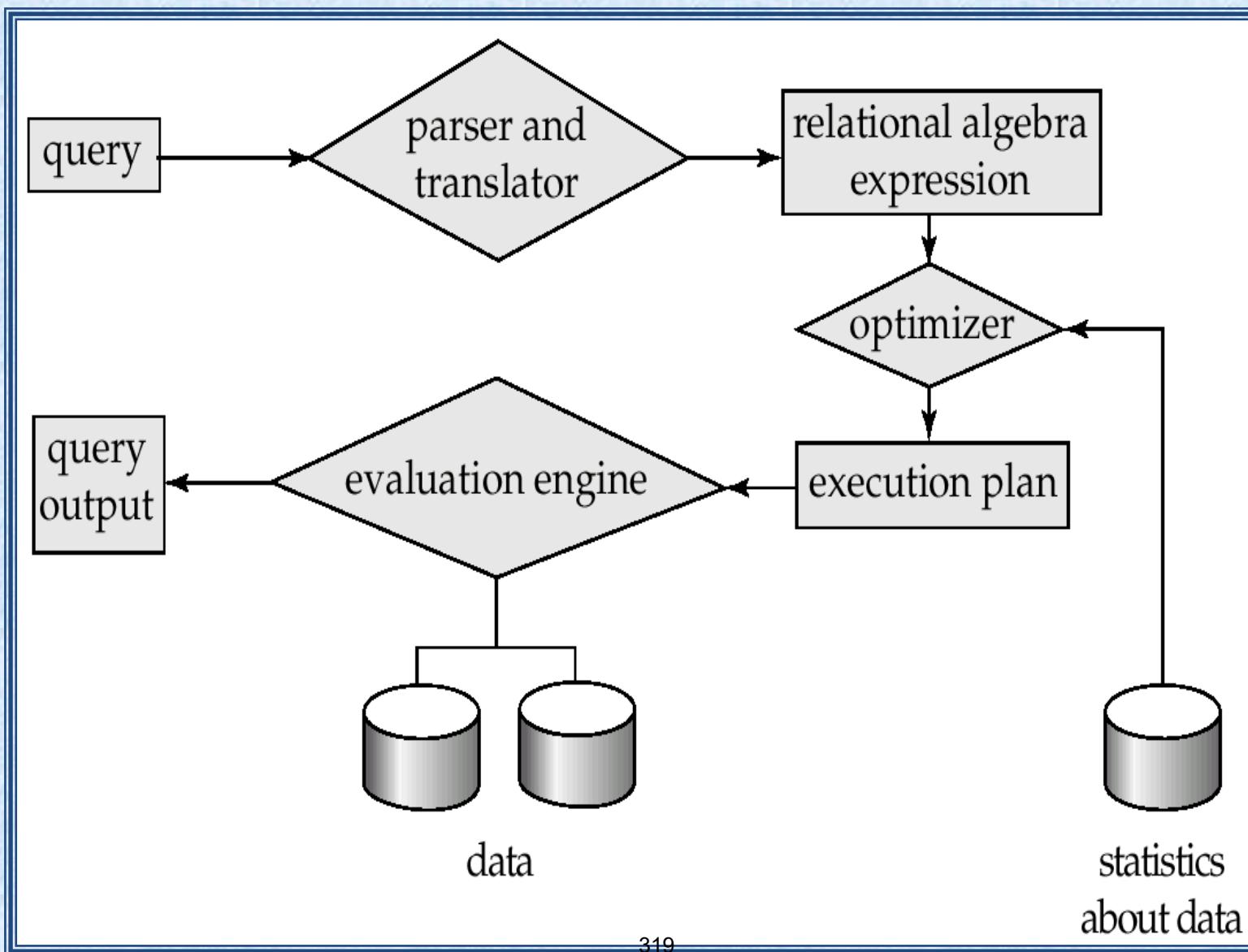
- ה RDBMS קובע את סוג האינדקס, אנו מגדירים רק את האינדקס עצמו. (הצורה על אינדקס עברו שדה בטבלה)
- במערכות RDBMS קיים אופטימיזר המריץ את השאלות בהתאם לנ吐ונים הקיימים ב DB ומחליט אם ומתי להשתמש באינדקס שהוגדר.



- 
 - הזיכרון ברכב המחשבים מורכב משני חלקים:
 - זיכרון ראשי (RAM) - מהיר
 - זיכרון משני (DISK) - גדול, איטי. יחסית לזיכרון הראשי.
 - כל החישובים מתבצעים בזיכרון הראשי.
 - כאשר כמות הנתונים גדולה מאוד, לא ניתן לקרוא את כל הנתונים בביטחון אחד לזיכרון הראשי ולהתחליל בחישוב ולקמן יש לחלק את החישוב לשלבים:

- ממבצעת קריאה של חלק מהנתונים לזיכרון הראשי
- על פי הנתונים שנקרו לזכרון הפנימי, מחושבת חלק מהתוצאה
- ממבצעת כתיבה לדיסק של התוצאה שחושמה
- ושוב חוזרים על ההליך
- הקריאה מהדיסק והכתיבה אל הדיסק הינם איטיים בהרבה מפעולות המבצעות בזיכרון הראשי. לכן, הגורם הנדרש לקריאה מהדיסק ולכתיבה על דיסק הינו הגורם העיקרי המשפיע על זמן חישוב שאילותות.

Query Processing



Query Processing (המשר)

- בשביל מה?
 - לדעת איל אינדקסים כדאי להגדיר (אם בכלל)
 - לדעת לבנות שאילתות יעילות
- סוגים אופטימיזציה בגישה למסדי נתונים:
 - אופטימיזציה מבוססת תחביר (Syntax-Based Optimization)
- המערכת מtabססת על הסדר שבו מופיעות היטבות ורשומים התנאים הלוגיים בפקודות ה- SQL
- יתרונות: פשטות ושליטה המשתמש
- חסרונות: דרישה מiomנות גבואה והכרה טובה של המבנה הפיסי של מסד הנתונים, השיטה סטטית.
- אופטימיזציה מבוססת עלות (Cost-Based Optimization)
- המערכת בונה אסטרטגיה "חכמה" על פי מצב מסד הנתונים וסטטיסטיות הנאספות באופן שוטף (קיים אינדקסים, גודל היטבה, וכו')

אופטימיזציה מבוססת עלות

- **לשם פשוטות:**
 - עלות שאליתה תקבע אך ורק לפי מספר הבלוקים שיקראו/יכתבו מן/אל הדיסק.
 - נתעלם מההבדל בין עלות קריית בלוק לעלות כתיבת בלוק
 - בכתיבה מתבצעת גם קריאה
 - כתיבה לא תמיד מתבצעת (לעתים רק מציגים את התוצאה)
 - לא נתחשב בפעולות המתבצעות בזיכרון (CPU)
 - כל החישובים הם הערכות – תור התחשבות במקרה הגרוע ביותר או במקרה הממוצע
 - לא נתחשב בעלות הכתיבה האחרונה של התוצאה חזרה אל הדיסק.
- עברו אופטימיזציה מבוססת עלות, דרישים נתוניים
קטלוגיים

מידע קטלוגי לצורך חישוב עלויות

מידע קטלוגי על טבלה r

- r_a - מספר הרשומות בטבלה r .
 - r_f - מספר הרשומות ב- r שבלוק (אחד) בזיכרון יכול להכיל.
 - r_b - מספר הבלוקים בזיכרון המכילים רשומות של r .
 - $v(A,r)$ - מספר הערכים השונים תחת עמודה A בטבלה r .
 - $sc(A,r)$ - מספר ממוצע של רשומות שתחרזיר שאלתה מן הצורה $(r_x = A)$.
- כלומר מספר הרשומות הממוצע המקיים שוויון כל
- תכונה A

מידע קטלוגי לצורך חישוב עלויות - דוגמא

| <i>customer-id</i> | <i>customer-name</i> | <i>customer-street</i> | <i>customer-city</i> |
|--------------------|----------------------|------------------------|----------------------|
| 019-28-3746 | Smith | North | Rye |
| 182-73-6091 | Turner | Putnam | Stamford |
| 192-83-7465 | Johnson | Alma | Palo Alto |
| 244-66-8800 | Curry | North | Rye |
| 321-12-3123 | Jones | Main | Harrison |
| 335-57-7991 | Adams | Spring | Pittsfield |
| 336-66-9999 | Lindsay | Park | Pittsfield |
| 677-89-9011 | Hayes | Main | Harrison |
| 963-96-3963 | Williams | Nassau | Princeton |

$$n_{customers} = 9$$

$$f_{customers} = 3$$

$$b_{customers} = 9/3=3$$

$$V(Customer-City, customers) = 6$$

$$SC(Customer-City, customers) = 9/6 = 1.5$$

מידע קטלוגי לצורך חישוב עלויות (המשר)
מידע קטלוגי על אינדקס ?

- f_i - מספר הממוצע של קשותות יוצאות מצומת כלשהו באינדקס.
- T_H - מספר הרמות באינדקס :
– $\lceil \log_{f_i}(V(A,r)) \rceil = T_H$.
- B_B - מספר מקסימלי של מצביעים בDALI.
- B_L - מספר הבלוקים שנמצאים בrama התחתונה של אינדקס .
– מספר הבלוקים שתופסים העלים והDALI

עלות שאילותות חיפוש ללא שימוש באינדקסים

• אלגוריתם A1 – חיפוש לינארי

– השאלתה: $\sigma_{A=V}$

– עיקרון: סרוק את כל הקובץ (r) באופן סדרתי – עבור כל רשומה בדוק את התנאי הנדרש

– עלות:

- מס' הבלוקים ב- r : b_r

– מה קורה אם החיפוש הוא על מפתח קביל?

- העלות הממוצעת היא $2/b_r$

• (החיפוש נוצר ברשימה הראשונה בה הערך גדול מהערך המבוקש).

– יתרונות:

- ניתן להשתמש באלגוריתם ללא תלות בדאנאי ³²⁵ בסדר הרשומות בקובץ או בקיום אינדקסים



עלות שאילותות חיפוש ללא שימוש באינדקסים (המשר)

• אלגוריתם A2 – חיפוש בינהרי

- השאלתה: $\sigma_{A=V}$
- הנחות: מתאים כאשר החיפוש מתרחש על שדה של פו הקובץ ממון בדיסק.
- עיקרון: דגום את הבלוק האמצעי בקובץ. אם הרשומות לא נמצאות בבלוק זה מרים את החלק בו נמצאת הרשומה (לפני/אחרי הבלוק הנוכחי). חצה את החלק הרלוונטי והמשר הלאה...
- עלות:

$$\lceil \log_2(b_r) \rceil + \lceil SC(A, r)/f_r - 1 \rceil$$

עלות מציאת הבלוק הראשון
עם הרשומה המקיימת התנאי

מספר הבלוקים הנוספים המכילים
רשומות המקיימות את התנאי

עלות שאלות חיפוש ללא שימוש באינדקסים (המשר)

• אלגוריתם A2 – חיפוש ביןארי

– שאלות לדין:

- מהי העלות במקרה ש- A הינו מפתח קביל?

$$\lceil \log_2(b) \rceil$$

ברגע שנמצאה הרשומה סימנו, ישנה רק רשומה אחת שכזאת.

- מהי עלות האלגוריתם במקרה שהשאילה היא $\sigma_{A \geq r}$, כאשר A הינו השדה/שדות לפיו/לפיים ממון קובץ הנתונים?

$$\bullet \quad v-1 - \lceil \log_2(r/f_r) \rceil + \lceil \log_2(b) \rceil * sc(A, r/f_r)$$

- נctrar להעריך כמה רשומות יחזירו מהתנאי $(r)_{\sigma_{A \geq r}}$ ועפ"י זה לחשב כמה בלוקים נוספים נctrar לקרוא.

עלות שאילותות חיפוש עם שימוש באינדקסים

- השאלה: $\sigma_{A=V}$, כאשר על השדה A ישנו אינדקס
- **אלגוריתם A3 – A הינו אינדקס ראשי (קובץ הנתונים ממוקן לפיו) ופתח קביל**
 - מחזיר רשומה אחת (בלוק אחד) המקיימת את השוויון
 - עיקרון: מצא את הערך V באינדקס ועקבות אחר המצביע הרלוונטי לקבלת הרשימה המבוקשת
 - עלות: $1 + HT_i$
 - בעצם, עלות קריית העץ + בלוק נוסף לקריאת הרשימה

עלות שאילותות חיפוש עם שימוש באינדקסים

- **אלגוריתם A4 – A הינו אינדקס ראשי (קובץ הנתונים ממוקין לפיו), אך לא מפתח קביל**
 - מחזיר מס' רשומות – הרשומות נמצאות בבלוקים סמוכים
 - עיקרון: מצא את הערך v באינדקס ועקבו אחר המצביע הרלוונטי לקבלת הבלוקים המכילים את הרשומות הרלוונטיות
 - עלות: $\lceil \frac{r}{f(A)} \rceil + HT_i$
 - בעצם, עלות קריית העץ + מס' הבלוקים הנוספים לקריאת שאר הרשומות העוננות על התנאי



עלות שאילותות חיפוש עם שימוש באינדקסים

• אלגוריתם A5 – A הינו אינדקס משני

- יש לקרוא מס' דליים – כל מצביע בדלי יכול להצביע לבлок אחר.
- עיקרון: מצא את הערך π באינדקס ועקבו לדלי (دلויים) הרלוונטיים. עבור כל מצביע בדלי רלוונטי עקוב לקבלת רשומה מבודקת
- הערות: $\lceil r / SC(A, \lceil r / SC(A, Bu) \rceil + HT) \rceil$
- הערכה: יקר מאד! במקרה הגרוע, כל רשומה נמצאת בבלוק שונה. חיפוש לינארי עשוי להיות זול יותר.

$$HT, + SC(A, r)/Bu + SC(A, r)$$

איתור הדלי הראשון
בקובץ האינדקס

מספר בלוקים
המכילים את הדליים
עם המצביעים
לרשומות

מספר בלוקים
מקסימלי (במקרה
של רשומה בבלוק
אחר)

עלות שאילותות חיפוש עם שימוש באינדקסים

- השאילה: $(r)_{\forall A \leq V} \sigma_A \geq r$: כאשר A הוא שדה עליו מוגדר אינדקס:

• **אלגוריתם A6 – A הינו אינדקס ראשי**

- הרחבת אלגוריתם A4: יש לקרוא את כל הרשומות שעונות על התנאי ולא רק את $r_{SC(A)}$
- עלות: $\lceil t/f_r \rceil + HT_i$
 - t הינה הערכה של מס' הרשומות המקיימות את התנאי
 - אם אין מידע על t מניחים שחצי מהרשומות מקיימות את התנאי

• **אלגוריתם A7 – A הינו אינדקס שני**

- הרחבת אלגוריתם A5
- מתבצע מעבר על מס' עליים ומס' דליים – כחלקם היחסי של מספר הרשומות העונות על התנאי מסך כל הרשומות
 - עלות: $\lceil t/n_r \rceil + LB_i * t/HT_i$
 - הערה: יקר מאוד! בהרבה מקרים חיפוש לינארי עדיף

דוגמאות לחישובי עלות שאילתא

TicketItem

ID

AdultTickets

ChildTickets

Purchase

Show

- נתונה הטבלה הבאה:

- נתונים הנתוניים הבאים:

n (מספר הרשומות בטבלה) $TicketItem=200,000$ –

f (מספר הרשומות של הטבלה שנכנסות בבלוק אחד) $TicketItem=500$ –

$b_r = \lceil n_r / f_r \rceil$ מספר הבלוקים המכילים את הרשומות של הטבלה $TicketItem=4000$ –

$V(Purchase, TicketItem) = 1,000$ –

$V(Show, TicketItem) = 5,000$ –

v ID הננו המפתח הראשי של הטבלה $V(ID, TicketItem) = 200,000$ –

$SC(Purchase, TicketItem)=200$ –

$40 = SC(Show, TicketItem)$ –

$75 = Bu$ (מספר המצביעים בDALI) –

ידעו שגודל כל מפתח חיפוש הנשמר בצומת של אינדקס מסווג + Bu שווה לגודל
שתופס כל מצביע בצומת (גודל $= search key$ גודל Pointer)

דוגמאות לחישובי עלות שאילתא (המשר)

שאילתא 1 (Q1): שיליפת כל הרשומות ששדה purchase שלהם שווה X

Select *

From TicketItem

Where Purchase = X

שאילתא 2 (Q2): שיליפת כל הרשומות ששדה show שלהם שווה Y

Select *

From TicketItem

Where Show = Y

דוגמאות לחישובי עלות שאילתא (המשך)

- נניח כי לא מוגדרים אינדקסים במערכת
- נניח כי הטבלה ממויינת עפ"י השדה purchase
- חישוב עלות שאילתת Q1: מה ניתן לעשות?
 - מכיוון שלא מוגדרים אינדקסים נוכל לבצע חיפוש סדרתי
- הוצאות – קריית כל הבלוקים בטבלה = 4000
- מכיוון שהטבלה ממויינת עפ"י שדה החיפוש נוכל לבצע חיפוש בינהר
- הוצאות $\lceil \log_2(b_r) \rceil + \lceil \frac{SC(A, r)}{f_r} \rceil - 1$

$$\lceil \log_2(b_{\text{TicketItem}}) \rceil + \lceil \frac{SC(\text{Purchase}, \text{TicketItem})}{f_{\text{TicketItem}}} \rceil - 1 = \\ \lceil \log_2(4,000) \rceil + \lceil \frac{200}{50} \rceil - 1 = 12 + 4 - 1 = 15$$

- חישוב עלות שאילתת Q2: מה ניתן לעשות?
 - רק חיפוש סדרתי – קריית כל הבלוקים בטבלה = 4000

דוגמאות לחישובי עלות שאילתא (המשר)

- נניח כי הטבלה ממויינת עפ"י השדה show
- חישוב עלות שאילתת Q1: מה ניתן לעשות?
 - מכיוון שלא מוגדרים אינדקסים נוכל לבצע רק חיפוש סדרתי
- עלות - קריית כל הבלוקים בטבלה = 4000
- חישוב עלות שאילתת Q2: מה ניתן לעשות?
 - מכיוון שהטבלה ממויינת עפ"י שדה החיפוש נוכל לבצע חיפוש בינארי
- עלות $1 - \lceil \log_2(b_r) \rceil + \lceil SC(A, r)/f_r \rceil * \lceil \log_2(TicketItem) \rceil + \lceil SC(Show, TicketItem)/f TicketItem \rceil - 1 =$

$$\lceil \log_2(4,000) + \lceil 40/50 \rceil - 1 = 12 + 1 - 1 = 12$$

דוגמאות לחישובי עלות שאילתא (המשך)

- נניח כי מוגדר אינדקס ראשי + B לפי Purchase (הטבלה ממויינת עפ"י Show Purchase) ואינדקס שני + B לפי Show
- חישוב עלות שאילתא Q1: מה ניתן לעשות?
 - את השאלתה Q1 ניתן לחשב ע"י שימוש באלגוריתם A4 (השדה A4 הוא אינדקס ראשי אך לא על מפתח קביל):
$$EC(Q1) = EC(A4[Q1]) = HTi + \lceil sc(Purchase, TicketItem) / f_{misTicketItem} \rceil$$
 - נדרש לחשב את $HTi = logfi(V(A,r))$
 - כיצד-Calculates את f_i ? (כלומר את המספר הממוצע של קשותות היוצאות מכל צומת בעז?)

דוגמאות לחישובי עלות שאילתא (הmarsh)

- חישוב זי:
- כל צומת שאינה השורש מכילה בין $\lceil \frac{n}{2} \rceil$ מלבנים, כאשר n קבוע לעצמו + נתון (צריך למצוא את n)
- לצומת בעלת n מלבנים יש $1 - \binom{n}{0}$ מפתחות חיפוש ידוע מהנתונים שגודל כל מפתח חיפוש הנשמר בצדמת של אינדקס מסווג + שווה לגודל שתופס כל מלבן בצדמת.
- נתון $75 = Bu$ (מספר המלבנים הנכנסים בבלוק אחד)

$$\frac{1}{Bu} * n + \frac{1}{NoOfSearchKeyInBlock} * (n-1) = 1$$
- נחשב את $n = 1 : 1 = 75$

$$1/75 * n + 1/75 * (n-1) = 1$$

$$2 * n = 76$$

כל צומת בגודל בלוק

בכל צומת יהיו מקסימום 38 = n מלבנים
 מאחר שלא נתון מידע אחר על תפוזת העץ נניח $n = 38$
 $fi = 0.75 * 38 = 29$
 ממוצע

דוגמאות לחישובי עלות שאילתא (המשך)

- עלות השאילתא בעזרת שימוש באינדקס ראשי על השדה *purchase* היא:

$$\begin{aligned} HT_i + \lceil SC(Purchase, TicketItem) / f_{TicketItem} \rceil &= \\ \lceil \log_{f_i}(V(Purchase, TicketItem)) \rceil + \lceil SC(Purchase, TicketItem) / f_{TicketItem} \rceil &= \\ \lceil \log_{2^9} 1000 \rceil + \lceil 200/50 \rceil &= 6 \end{aligned}$$

דוגמאות לחישובי עלות שאילתא (המשר)

- נחשב עלות של שאילתת Q2 כאשר השדה show הנו אינדקס מימי עז
B+

$$EC(Q_2) = EC(A5 [Q_2]) =$$

$$HT_i + \lceil SC(Show, TicketItem)/Bu \rceil + SC(Show, TicketItem) =$$

$$\lceil \log_{29}(V(Show, TicketItem)) \rceil + \lceil 40/75 \rceil + 40 =$$

$$\lceil \log_{29}5000 \rceil + \lceil 40/75 \rceil + 40 = 3 + 1 + 40 = 44$$

דוגמאות לחישובי עלות שאילתא (המשך)

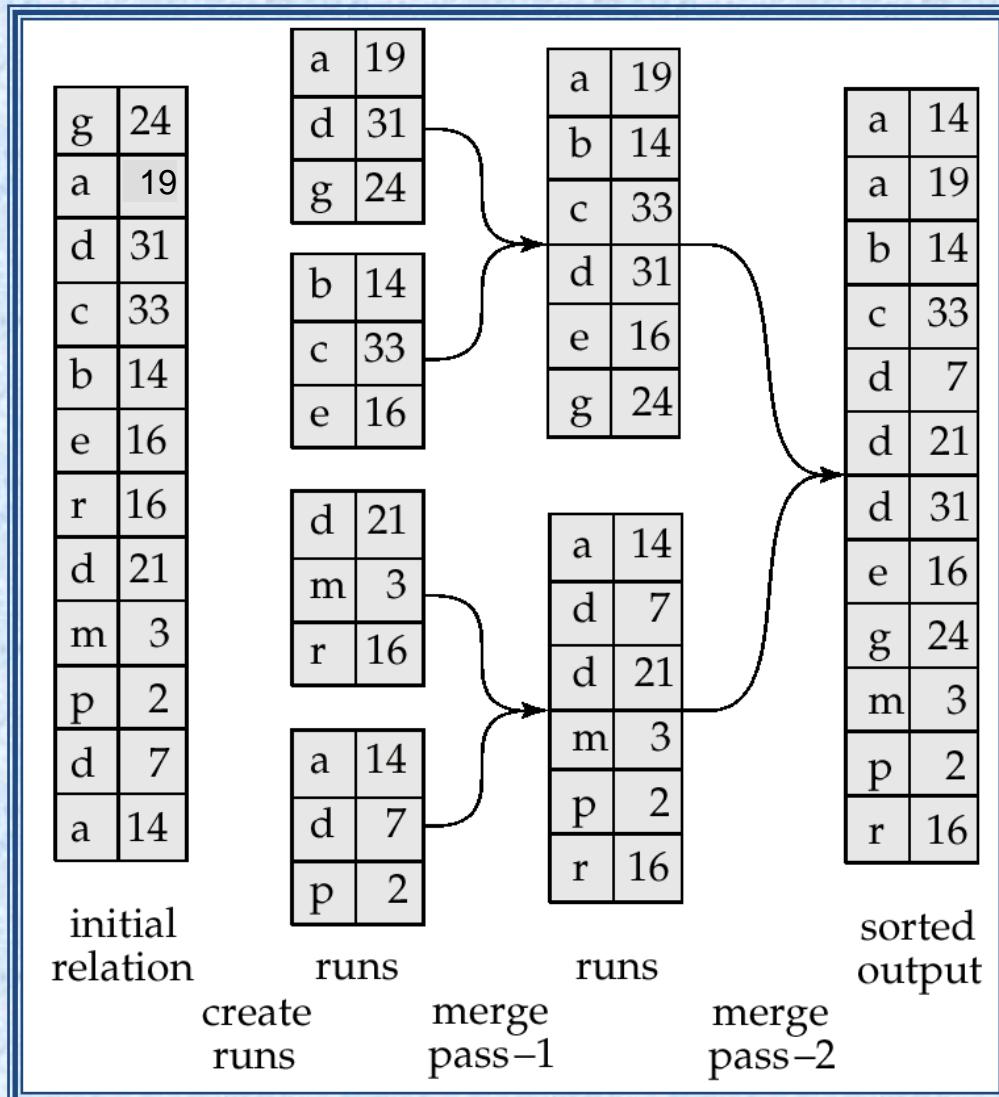
- האם כתוצאה שימוש באינדקסים הוזלה עלות השאלתה?

| האלגוריתם הנבחר | Q1 | Q2 |
|---|----------------------------|-----------------------------|
| לא אינדקסים, הרשומות בדיסק ממינות לפि השדה Purchase | חיפוש בינהר: עלות 15 | חיפוש לינהר: עלות 4000 |
| לא אינדקסים, הרשומות בדיסק ממינות לפি השדה show | חיפוש לינהר: עלות 4000 | חיפוש בינהר: עלות 12 |
| מוגדר אינדקס ראשי + B לפி Purchase | שימוש באלגוריתם A4: עלות 6 | לא כדאי |
| מוגדר אינדקס משני + B לפி Show | לא כדאי | שימוש באלגוריתם 5A: עלות 44 |

חישוב עלות שאילתא (המשר)

- עד עכšíו ראיינו שאילתות שבמציאות פועלות בחירה (select) בלבד
- מה קורה אם נדרש לבצע פעולה אgregציה? (group by)
- מה קורה אם נדרש לבצע מיון של הפלט (order by)
- עבור ביצוע פעולה אgregציה ועבור פעולה מיון הפלט נדרש לבצע מיון של הרשומות.
- יש לנקח בחשבון עלות מיון (בד"כ יקרה מאוד!)

מיון חיצוני Sort Merge



- מתאים לשאלתה מהצורה

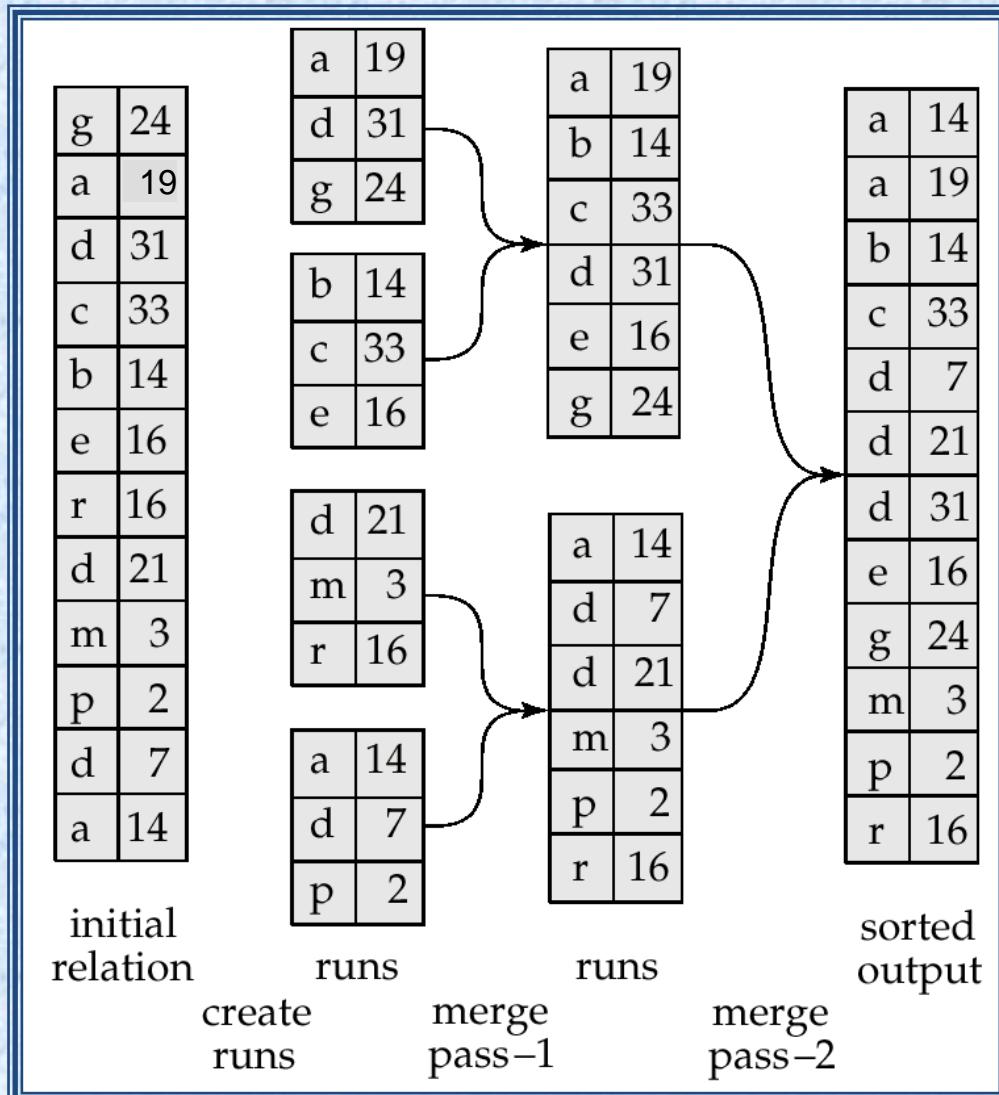
Select ...

From ...

Order by

- **העיקرون:**
 - במעבר הראשון ממיניים בכל פעם M בלוקים (גודל הזיכרון) מהטבלה וכותבים אותם (ממוניים) לדיסק
 - בכל מעבר, מציגים (merge) רשימות ממונות לרשימה ממונת אחת גדולה יותר

עלות מיון חיצוני



• הבעיות:

- מס' המעברים (אחרי ייצור ה- $\lceil \log_{M-1}(b_r/M) \rceil$ runs הראשוניים)
- מעבר נוסף כדי ליצור את ה- $\lceil \log_{M-1}(b_r/M) \rceil + 1$ runs הראשוניים
- עלות כל מעבר b_r (אחד לקריאה ואחד לכתיבה)
- כמו כן, במעבר האחרון אין צורך לבצע את הכתיבה
- לכן עלות הכוללת היא: $2 b_r (\lceil \log_{M-1}(b_r/M) \rceil + 1) - b_r$

חישובי עלות שאילתא (המשר)

- דיוון בנושא פועלות מין
 - כאשר פעלת המין מתבצעת עבור פעלת ארגזית
- נתחשב במספר הרשומות הנכנסות בבלוק לאחר פעלת הסינון ב `where`
- נתחשב בכל גודל הרשומות שיש בטבלה/ טבלאות
- כאשר המין מתבצע עבור פעלת מין `order by`
- נתחשב רק במספר הרשומות הסופי שהשאילתא תחזיר
- נתחשב בחלוקת החלקי של גודל הרשומה שוחרר מהתוצאות `select`

חישובי עלות שאילתא (המשר)

- עד עכšíו פעלנו על טבלה אחת בלבד.
- מה קורה כמשמעות 2 טבלאות? מה קורה כאשר משתפות ה טבלאות?
- נדרש לציג את הנתונים החזריים מפעולת ה join.
- פעולה המיזוג מחפש התאמות על צירופים מהטבלאות המשתפות
- כיצד?

עלות פעולות Block Nested-Loop Join - Join

- כאשר לא משתמשים באינדקסים לשם פעולה ה **Join**
- **העיקרון:**

```
for each block  $B_r$  of  $r$  do begin  
    for each block  $B_s$  of  $s$  do begin  
        for each tuple  $t_r$  in  $B_r$  do begin  
            for each tuple  $t_s$  in  $B_s$  do begin  
                Check if  $(t_r, t_s)$  satisfy the join condition  
                if they do, add  $t_r \bullet t_s$  to the result.
```

• ה - עלות?

- במקרה הביניים: שימוש ב- M בלוקים (כאשר M גודל הזיכרון בבלוקים) עבור הטבלה החיצונית, שימוש בבלוק אחד עבור הטבלה הפנימית ושימוש בבלוק האחרון לכתיבה ה- $output$:

$$b_r * b_s + b_r * (M-2) / M$$

- במקרה הנוכחי שבו קיימים בלוק אחד לכל טבלה + בלוק ל- $output$ ($M=3$):

$$b_r * b_s + b_r$$

- במקרה הטוב ביותר (התבלה החיצונית נכנסת לזיכרון, כלומר $b_r + b_s = b_r + 2$) ($M=3$):

³⁴⁶

עלות פעולות Join - Indexed Nested-Loop Join

- כאשר משתמשים באינדקסים לשם פעולה ה Join
 - **העיקרון:**
 - לכל רשומה r בטבלה החיצונית z , השתמש באינדקס כדי לחפש את הרשומות המתאימות ב- z (המקיימות את תנאי ה- Join)
- **עלות?**
 - $c * n_r + r b$, כאשר c הוא עלות חיפוש הרשומות המתאימות ב- z
- **הערות:**
 - מה יקרה אם קיימים גם אינדקס על z ?
 - נבדוק איזו טבלה כדאי לקרוא קודם
 - מה קורה אם ישנו תנאי סינון ב where על הטבלה המשתתפת ב Join?
 - נשלוף את הרשומות החזרות מהתנאי תוך שימוש באינדקס ורק עם השורות החזרות נבצע את ה Join (או ההיפר)

דוגמאות לחישובי עלות שאילתא (הmarsh)

- נתונות הטבלאות show ו-TicketItem

$$n_{Show} = 1,000$$

$$n_{TicketItem} = 20,000$$

$$b_{Show} = 50$$

$$b_{TicketItem} = 1,000$$

$$Bu = 100$$

$$SC(Show, TicketItem) = 20$$

$$HT_i = 4$$

$$LB_i = 2000$$

- כאשר ז אינדקס משנה המוגדר על `TicketItem.Show` על `Show` והנתונים ממוחנים לפי ID בשתי הטבלאות.

| TicketItem |
|--------------|
| ID |
| AdultTickets |
| ChildTickets |
| Purchase |
| Show |

| Show |
|-----------|
| ID |
| Film |
| StartTime |
| Theater |

דוגמאות לحسابי עלות שאלתא (המשר)

- נתונה השאלה הבאה (Q3)

```
SELECT S.*  
FROM Show AS S  INNER JOIN  
        TicketItem AS MIS  ON S.id = MIS.Show  
WHERE ChildTickets > 0
```

- נבחן את דרכי הפעולה השונות לביצוע שאלתא זו

דוגמאות לحسابי עלות שאלתא (המשר)

- נמצא את כל הרשומות ב- $Show \bowtie_{ID=Show} TicketItem$
 - נבצע קודם את המכפלה הקרטזית בין הרשיות $ChildTickets$
 - עבור כל רשומה שנמצאה מתאימה, "יבדק בזיכרון התנאי $0 > ChildTickets$ "
(אין עלות כי הרשומה כבר נמצאת בזיכרון)
- דרך א': נחשב את פועלות ה `join` תוך שימוש ב-block nested loop join $EC(Block_Nested_Loop_Join[Q1]) = b_{Show} + b_{TicketItem} * b_{Show} = 50 + 1,000 * 50 = 50,050$

דוגמאות לחישובי עלות שאלתא (המשר)

• דרך ב':

- ראשית נקרא את טבלת **TicketItem** ועבור כל רשומה שתחזיר מהתנאי $\sigma_{\text{Child} > 0}$ נבצע חיפוש ביןארי על השדה ID בטבלת **Show** (תנאי ההשוואה ב **join**)
 - ידעו שבסתי הטבלאות הנתונים ממוקמים לפי ID ושה-ID הוא גם מפתח ראשי
 - על התוצאה נפעיל את אלגוריתם ה- **join** סה"כ הוצאות תהיינה:

$$b_{\text{TicketItem}} + \#Records(\sigma_{\text{Child} > 0} \text{TicketItem}) * EC(A2[\text{Show}]) =$$

$$b_{\text{TicketItem}} + \#Records(\sigma_{\text{Child} > 0} \text{TicketItem}) * \log_2 b_{\text{show}} =$$

$$1000 + (n_{\text{TicketItem}}) * \log_2 50 = 1000 + 20,000 * \log_2 50 =$$

$$1000 + 20,000 * 6 = 121000$$

- אם היינו יודעים שכטוצאה מהתנאי יוחזרו במקסימום 5000 רשומות?

דוגמאות לحساب עלות שאילתא (המשך)

- דרך ג': שימוש באינדקס על השדה show בטבלה TicketItem – נבצע index nested loop
- לכל רשומה ב- show נשתמש באינדקס המשי�� המוגדר על TicketItem.Show כדי לבצע את ה- join inner, כלומר, לחפש את הרשימה המתאימה. (התנאי יבדק בזיכרון).
- סה"כ הוצאות תהיינה:

$$b_{\text{Show}} + n_{\text{Show}} * (\text{HT}_i + \lceil \text{SC}(\text{Show}, \text{misTicketItem}) / \text{Bu} \rceil + \text{SC}(\text{Show}, \text{misTicketItem})) = 50 + 1000 * (4+1+20) = 25,050$$

חישוב עלות שאילתת הכללת שאילותות מקונות

- נבחין בשני מקרים של שאילותות מקונות
 - השאילה הפנימית **בלתי-תלויה** בשאלתה החיצונית.
(ולכן רצה פעם אחת בלבד)
 - השאילה הפנימית **תלויה** בשאלתה החיצונית. (ולכן רצה עבור כל איטרציה של השאלתא החיצונית)

חישוב עלות שאילתת הכללת שאילותות מקונות

- **מקרה א': השאלתה הפנימית בלתי-תלויה בשאלתה החיצונית (Q4)**

```
select *
```

```
from Show
```

```
where Show.ID in (select Show
```

```
        from TicketItem as ticket
```

```
        where ticket.ChildTickets > 0)
```

- מהי דרך הפעולה?

חישוב עלות שאילתת הכללת שאילותות מקונות

- בהנחה כי הזיכרון "גדול ממספר", עלות השאלתה תהיה : (ש.חיצונית)EC + (ש.פנימית)EC.
- השאלתה הפנימית תחשב ראשונה והתוצאה תשמר בזיכרון, ולאחר מכן תחשב השאלתה החיצונית, כאשר כל רשותה תיבדק מול התוצאה של השאלתה הפנימית בזיכרון
- מה קורה אם השאלתה הפנימית אינה נכנסת בזיכרון?



חישוב עלות שאילתת הכללת שאילותות מקוונות

$$EC(Q4) = EC(Q4.\text{inner}) + EC(Q4.\text{outer}) =$$

$$EC(A1[\sigma_{\text{ChildTickets} > 0} \text{TicketItem}]) + EC(A1(\text{Show})) =$$

$$b_{\text{TicketItem}} + b_{\text{Show}} = 1,000 + 50 = 1,050$$

חישוב עלות שאילתת הכללת שאילותות מקונות

- **מקרה א': השאלתה הפנימית תלויה בשאלתה החיצונית (Q5)**

```
select *  
from Show  
where exists (select *  
               from TicketItem as ticket  
               where ticket.Show = Show.ID and  
                     ticket.ChildTickets > 0)
```

- מהי דרך הפעולה?

חישוב עלות שאלתה הכללית שאילתות מקונות

- עלות השאלתה תהיה: $(\text{ש.פנימית}(\text{EC}) * (\text{ש.חיצונית}(\text{Records}\#) + (\text{ש.חיצונית}(\text{EC}$)
– השאלתה הפנימית תחשב כל פעם מחדש עבור כל רשומה בשאלתה החיצונית.

$$b_{Show} = 1,000 * 1,000 + 50 * b_{misTicketItem}$$

- מכיוון שהערך המשווה בתוך השאלת הפנימית הוא שדה עליי מוגדר אינדקס, ניתן לבצע חיפוש של הערך המשווים בתוך הטבלה במקום חיפוש בכל הטבלה. במקרה זה העלות תהיה

$$b_{Show} + n_{Show} * (HT_i + \lceil SC(Show, misTicketItem)/Bu \rceil + SC(Show, misTicketItem)) = 50 + 1000 * (4+1+20) = 25,050$$

טרנზקציית ובקרת מקבילות

זמןים ברι סידור, פרוטוקלים לנעילות

מושג הטרנזקציה (תנוועה)

- טרנזקציה הינה ייחdet ביצוע אוטומית בסיסד הנתונים.
- מורכבת בדרך כלל ממספר פעולות בסיסיות.
- יכולה לשלוּף, לעדכן ולמחוק נתונים
- לרבות מבצעת בקשה משתמש קצה: הרשמה, העברה, משיכה
וכד' ...
- טרנזקציה תrhoץ על בסיס נתונים עקי
 - במהלך ריצת הטרנזקציה, בסיס נתונים עשוי להיות לא עקי
למשך זמן מסוים
- כאשר טרנזקציה מסתיימת בהצלחה (committed), בסיס נתונים חייב להיות עקי



מושג הטרנזקציית (הmarsh)

- מה קורה אם המערכת נופלת בזמן ביצוע הטרנזקציה?
 - הפסקת חשמל
 - קרייסט המחשב
 - תקלת בתכנה

- מה קורה בעת ביצוע מקביל של טרנזקציות?
 - יותר מטרנזקציה אחת מבצעת פעולות טבלה/ טבלאות



מושג הטרנזקצייה: מוטיבציה

- מה קורה אם המערכת נופלת בזמן ביצוע הטרנזקציה?

Insert into registration

```
(course_id, student_id, semester)  
('M-100', '210', 'SUM1999')
```

Update courses

```
set enroll=enroll+1  
where course_id='M-100' and semester='SUM1999'
```

- מה קורה בעת ביצוע מקביל של הטרנזקציות הבאות?

Get input (amount1)

```
output1=Read (balance)  
  
output1=output1+amount1  
  
Write (balance=output1)
```

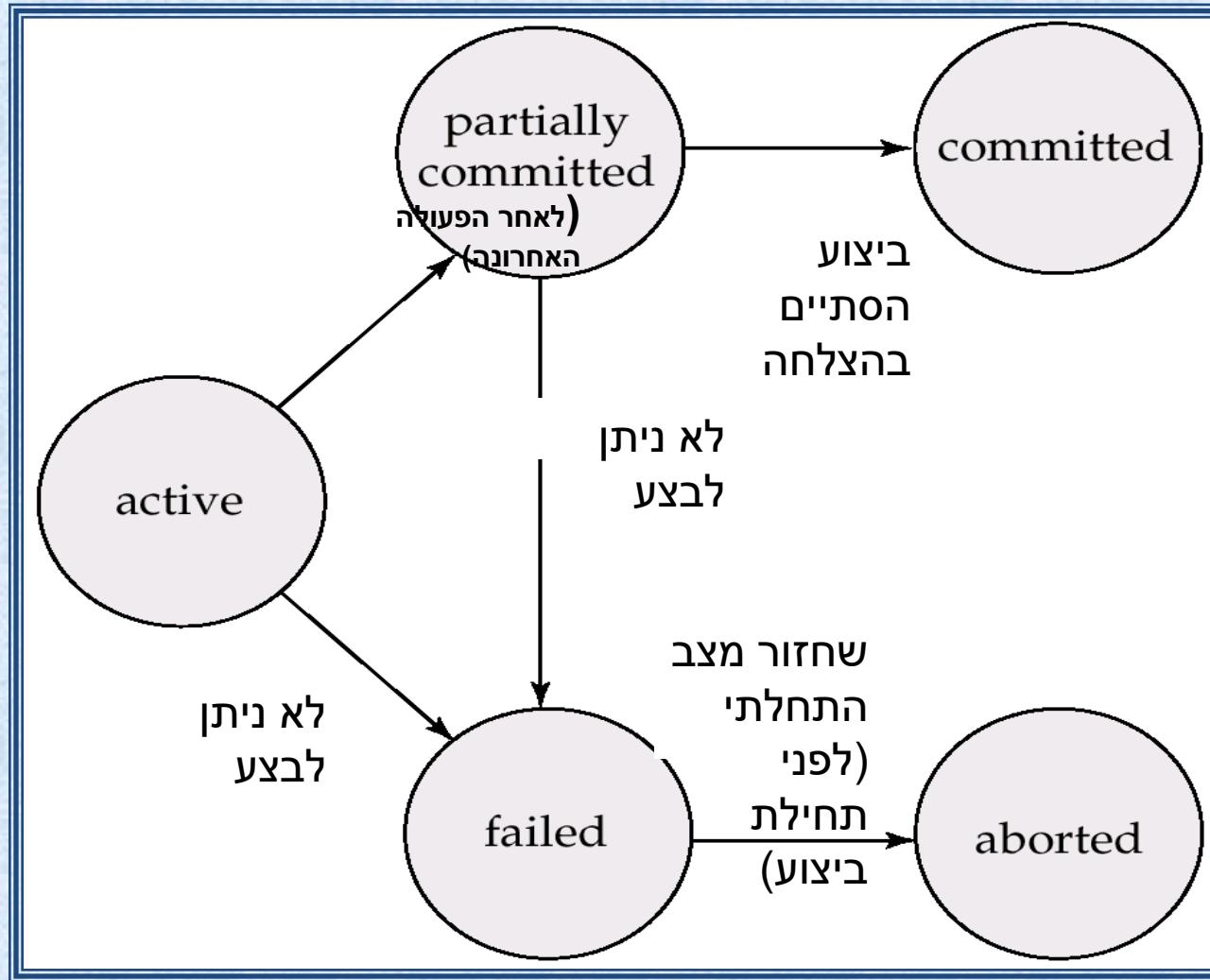
Get input (amount2)

```
output2=Read (balance)  
  
output2=output2+amount2  
  
Write (balance=output2)
```

תכונות בסיסיות של טרנסקציות (ACID)

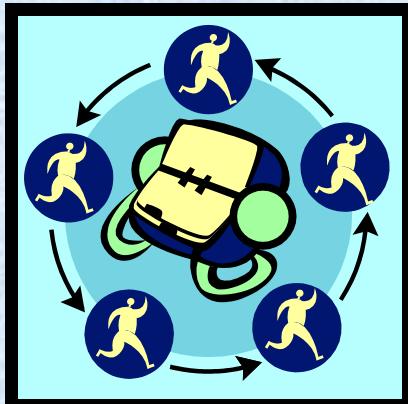
- **אטומיות (Atomicity)** – טרנסקציה מתבצעת בשלמותה או אינה מתבצעת כלל (כלומר, כל פעולות הטרנסקציה מתבצעות או שאף אחת מהן).
- **עקביות (Consistency)** – העקביות של מסד הנתונים נשמרת בכל טרנסקציה.
- **בידוד (Isolation)** – למروת שטרנסקציות יכולות להתבצע במקביל, כל טרנסקציה איננה מודעת ליתר הטרנסקציות – לכל שני טרנסקציות 6 ו-7 המתבצעות במקביל, עכברicol רואה את מסד הנתונים לפני 6 ו-7 התחיליה או אחרי סיוםה.
- **יציבות (Durability)** – אחרי שטרנסקציה סיימה להתבצע בהצלחה, השינויים שטרנסקציה זו בוצעה למסד הנתונים נשמרים (אפילו אם המערכת נופלת)

מצבים של טרנסקציות

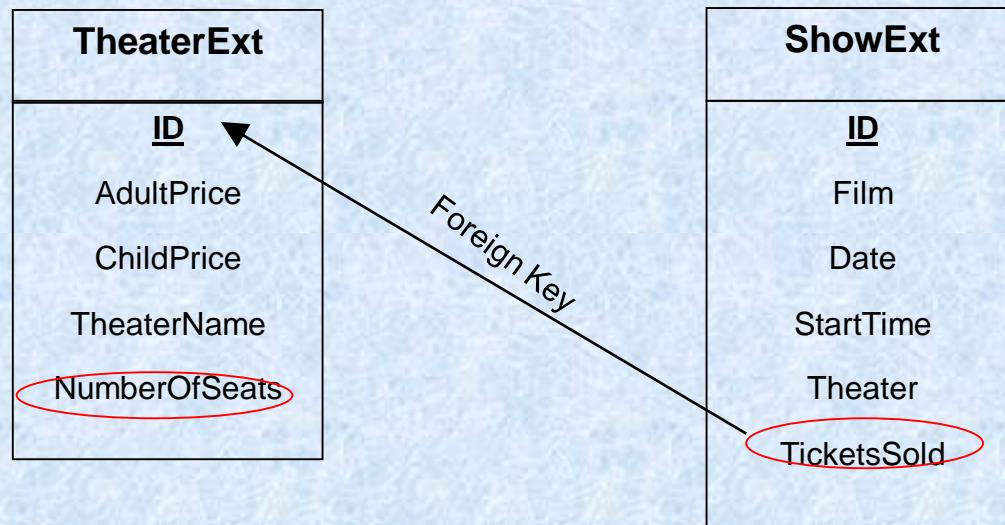


זמן (schedule

- זמן הינו רצף המגדיר את הסדר הcronologי שבו מתבצעות פעולות של טרנזקציות.
- מכיל את כל הפעולות הכלולות בטרנזקציות.
- סדר ביצוע הפעולות של כל טרנזקציה בזמן הוא סדרן בטרנזקציה.
- ברגע נתון יכולה להתבצע פעולה אחת בלבד
- זמן סדרתי – זמן שכל הטרנזקציות המוגדרות בו מתבצעות אחת אחרי השנייה באופן סדרתי
- זמן מקבילי – זמן שהטרנזקציות המוגדרות בו רצות במקביל (הפעולות בכל טרנזקציה)



- דוגמא: נתונות 2 טבלאות הבאות



- משרד כרטיסים מעוניין למכור ללקוח כרטיס לסרט מסויים המוקרן באחד מאולמות הקולנוע.
- מה משרד הכרטיסים צריך לבצע כדי למכור כרטיס ללקוח?
 - נתון שלסרט מסויים נמכרו 249 כרטיסים ובאולם בו מוקרן הסרט יש 250 מקומות ישיבה.



זמן (המשך)

- כדי למכור בהצלחה כרטיס ללקוח יש לבצע:
 - בדיקה שנשארו כרטיסים לסרט המבוקש (השוואה בין מספר הcartisim שנמכרו לבין מספר המקומות באולם בו מוקרן הסרט).
 - אם נשארו כרטיסים, יש לעדכן את מספר הcartisim שנמכרו (להגדיל המונה ב- 1).
- הטרנץקציה יכולה להיקتب כך:

A=Read (TheaterExt.NumberOfSeats)

B = Read (ShowExt.TicketsSold)

IF A > B

 B = B +1

 Write (ShowExt.TicketsSold = B)

ELSE

 Print "No Tickets Left"

זמן (המשך)

- דוגמא לזמן סדרתי:
 - נבדוק מה ערכם של השדות A ו- B לאחר ביצוע סדרתי של הטרנזקציות: (בלי הגבלת הכלליות, נניח כי A מtbodyעת לפני B)

| T1 | T2 | (B) |
|---|--|---|
| <p><i>Read A</i> <i>Read B</i> <i>IF A > B</i> <i>B=B+1</i> <i>Write (B)</i></p> | <p>Read A Read B IF A > B B=B+1 Write (B)</p> | <p>A = 250 B = 249 A = 250 B = 250 A = 250 B = 250 A = 250 B = 250</p> |

- בסיסים הרצת הטרנזקציות בזמן הסדרתי נマー כרטיס אחד בלבד.

זמן (המשך)

- דוגמא לזמן מקבילי (הנותן תוצאה "טובה" ביחס לזמן סדרתי):
 - נבדוק מה ערכם של השדות A ו- B לאחר ביצוע מקבילי של הטרנזקציות:

| | | |
|--|---|--|
| T_1 <i>Read A</i> <i>Read B</i> <i>IFA > B</i> <i>B=B+1</i> <i>Write (B)</i> | T_2 <i>Read A</i> <i>Read B</i> <i>IF A > B</i> <i>B=B+1</i> <i>Write (B)</i> | $A = 250$ $A = 250$ $B = 249$ $B = 250$ $B = 250$ $B = 250$ |
|--|---|--|

- בסיסם הרצת הטרנזקציות בזמן המקבילי נמכר כרטיס אחד בלבד.
- כדי לשמור על עקביות של טרנזקציות אלו, יש לדאוג שהקריאה של שדה B בטרנזקציה T_2 תבוצע לאחר הכתיחה של שדה זה שנעשתה ע"י T_1 .

זמן (המשך)

- דוגמא לזמן מקבילי (שלא נותן תוצאה "טובה" ביחס לזמן סדרתי):
 - נבדוק מה ערכם של השדות A ו- B לאחר ביצוע מקבילי של הטרנזקציות:

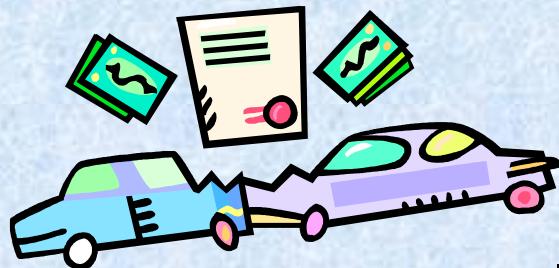
| T1 | T2 | |
|--|--|---------------------|
| <i>Read A</i> <i>Read B</i> | | $A = 250 \ B = 249$ |
| <i>IF A > B</i> <i>B=B+1</i> <i>Write (B)</i> | <i>Read A</i> <i>Read B</i> | $A = 250 \ B = 249$ |
| | <i>IF A > B</i> <i>B=B+1</i> <i>Write (B)</i> | $A = 250 \ B = 250$ |
| | | $A = 250 \ B = 251$ |

- בסיום הרצת הטרנזקציות בזמן זה נמכרו 2 כרטיסים – מצב לא תקין.

- הנחה, כל טרנזקציה משמרת קונסיסטנטיות לפיך בצעד סדרתי של כל התנועות משמר אף הוא קונסיסטנטיות.
- זמן סדרתי תמיד יחזיר תוצאות "טובות"
- יש לדאוג שזמנים מקבילים יחזירו תוצאות "טובות" ביחס לזמן סדרתי כלשהו
- זמן נקרא בר-סדר אם הוא שקל לביצוע סדרתי.
- סוגים של זמינים ברי סידור:
 - בר סידור קונפליקט
 - בר סידור מבט

פועלות מתנגשות

- לצורך בדיקת ברות-סידור, רק פועלות קריאה וכתיבה נחשבות (רק דרך יכולות טרניזקציית לדעת אחת על רעוגה).
- פועלות מתנגשות



- שתי פועלות א ו- ל מתנגשות אם:
 - א ו- ל מתייחסות לאותו פריט מידע Q.
 - פועלות א ו- ל שייכות לטרניזקציות שונות.
 - פחות אחת מהן היא פועלה כתיבה.
- פועלות מתנגשות חייבות להתבצע בסדר זמני מסוים.

פעולות מתנגשות (המשך)

- **פעולות** I_i, I_j של טרנסקציות T_i, T_j **מתנגשות** אם קיימם פריט מידע Q שפעולות I_i, I_j ניגשות אליו ולפחות אחת מעולות אלו היא של כתיבה ל- Q
 - 1. $I_i = \text{read}(Q), I_j = \text{read}(Q) \Rightarrow I_i$ and I_j don't conflict
 - 2. $I_i = \text{read}(Q), I_j = \text{write}(Q) \Rightarrow$ They conflict
 - 3. $I_i = \text{write}(Q), I_j = \text{read}(Q) \Rightarrow$ They conflict
 - 4. $I_i = \text{write}(Q), I_j = \text{write}(Q) \Rightarrow$ They conflict
- לצורך בדיקת זמנים ביחס בזמן סדרתי כלשהו, נתעלם מעולות שאינן **read** או **write**

זמן בר סידור קונפליקט (conflict serialization schedule)

- שני זמנים נקראים **שקלים קונפליקט** אם ניתן לקבל את האחד מהآخر ע"י החלפת סדרן של פעולות עוקבות לא-מתנגשות בלבד.
- זמן נקרא **בר סידור קונפליקט** אם הוא שקול **קונפליקט** לזמן סדרתי.
- כדי לבדוק אם זמן הוא **בר סידורKonflikt**, ניתן לבנות גרפ' קדימיות:
 - גרפ' מכון
 - צמתי הגרף הינם הטרנזקציות
 - קיימת קשת $T \rightarrow T'$ אם קיימות פעולות מתנгשות בין הטרנזקציות T וטרנזקציה T' פונה לפriet המידע קודם.
- אם בגרף המכון אין מעגל מכון, משמע שהזמן הינו זמן **בר סידור Konflikt**.

זמן בר סידור קונפליקט (המשר)

- האם הזמן הבא הינו בר סידור קונפליקט?

| | | |
|--|---|--|
| <u>T1</u> <i>Read A</i> <i>Read B</i> <i>IFA > B</i> <i>B=B+1</i> <i>Write (B)</i> | <u>T2</u> Read A Read B IF A > B B=B+1 Write (B) | $A = 250$ $A = 250$ $B = 249$ $B = 250$ $B = 250$ $B = 250$ |
|--|---|--|

- אין מעגל מכ้อน, הזמן בר סידור קונפליקט וشكול קונפליקט בזמן הסדרתי $T1 \rightarrow T2$



זמן בר סידור קונפליקט (המשך)

- האם הזמן הבא הנז בבר סידור קונפליקט?

| T1 | T2 |
|--|--|
| <i>Read A</i> <i>Read B</i> | Read A Read B |
| <i>IF A > B</i> <i>B=B+1</i> <i>Write (B)</i> | <i>IF A > B</i> <i>B=B+1</i> <i>Write (B)</i> |

- ישנו מעגל מכoil, הזמן אינו בר סידור קונפליקט



זמן בר סידור קונפליקט (המשר)

- נתונות 3 הטרנזקציות הבאות:

T_1 מכירת כרטיס

Read A ()

Read B ()

IF A > B

B = B + 1

Write B

ELSE

Print "No Tickets Left"

T_2 החזרת כרטיס

Read B (ShowExt.TicketsSold)

B = B - 1

Write B

T_3 הדפסת מספר מקומות פנויים
שנותרו

Read A (TheaterExt.NumberOfSeats)

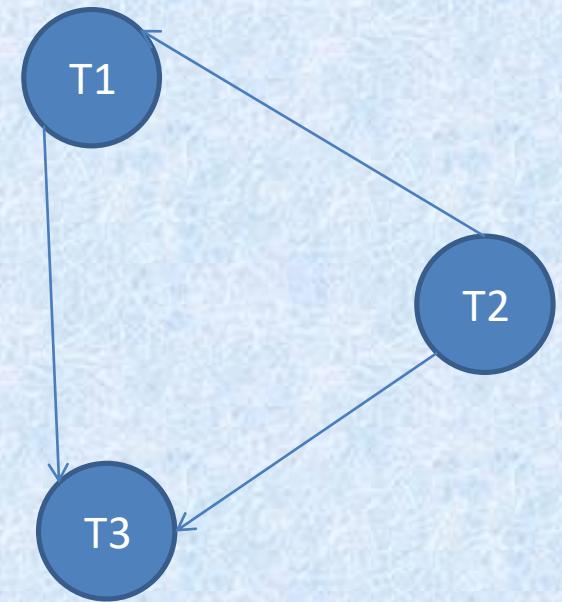
Read B (ShowExt.TicketsSold)

Print A - B

זמן בר סידור קונפליקט (המשך)

| <u>T1</u> | <u>T2</u> | <u>T3</u> |
|-------------------------------|-------------------------------|--------------------|
| <i>Read A</i> | | |
| | Read B | |
| | $B = B - 1$ | <i>Read A</i> |
| | Write B | |
| <i>Read B</i> | | |
| <i>IFA > B</i> | | |
| $B = B + 1$ | | |
| Write B | | |
| | | <i>Read B</i> |
| | | <i>Print A – B</i> |

- האם הזמן הבא הינו בר סידור קונפליקט?

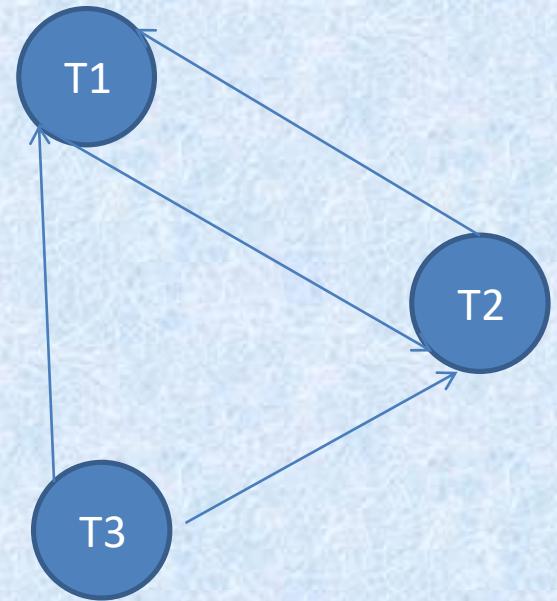


אין מעגל מכ้อน, הזמן בר סידור קונפליקט וشكול בזמן הסדרתי $T3 \rightarrow T1 \rightarrow T2$

זמן בר סידור קונפליקט (המשך)

- האם הזמן הבא הינו בר סידור קונפליקט?

| T1 | T2 | T3 |
|-------------|-------------|-------------|
| | Read B | |
| Read A | | |
| Read B | | |
| | | Read A |
| | | Read B |
| | | Print A – B |
| IF $A > B$ | | |
| $B = B + 1$ | | |
| Write B | | |
| | $B = B - 1$ | |
| | Write B | |



קיים מעגל מכון, הזמן אינו בר סידור קונפליקט

זמן בר סידור מבט : (view serializability)

- זמןים S ו- S' של אותם טרנסזקציות הם **שקלים מבט** (view equivalent) אם מתקיימים שלושת התנאים הבאים:
 - לכל פריט מידע Q , אם T_i קוראת ערך הinitial של Q ב- S , אז אותה טרנסזקציה תקרה ערך הinitial של Q ב- S'
 - לכל פריט מידע Q , אם T_i קוראת ב- S ערך שטרנסזקציה T_j יקרה, אז T_j תקרה גם ב- S' את הערך שטרנסזקציה T_j יקרה
 - לכל פריט מידע Q , הטרנסזקציה (אם קיימת) שביצעה את הכתיבה האخונה ל- Q ב- S תבצע את הכתיבה האחונה ל- Q גם ב- S'
- זמן S הוא **בר סידור מבט** (view serializable) אם הוא שקול מבט לאיזשהו זמן סדרתי
 - כל זמן בר סידור קונפליקט הוא גם בר סידור מבט
 - לא כל בר סידור מבט הוא בר סידור קונפליקט



זמן בר סידור מבט (המשר)

• האם הזמן הבא הינו בר סידור מבט?

• הן T1 והן T2 קוראות ערך התחלתי ל- A, אך אין בעיה עם זה שכן אף אחת מהטרנסזקציות לא כותבת ל- A.

• T1 קוראת ערך התחלתי ל- B ו- T2 קוראת ל- B את הערך ש- T1 כתבה. זהו המצב גם בזמן הסדרתי $T1 \rightarrow T2$.

• T2 כותבת אחורונה ל- B, כמו גם בזמן הסדרתי $T2 \rightarrow T1$.

| T1 | T2 |
|--------------------|--------------------|
| <i>Read A</i> | <i>Read A</i> |
| <i>Read B</i> | |
| <i>IF A > B</i> | |
| <i>B=B+1</i> | |
| <i>Write (B)</i> | |
| | <i>Read B</i> |
| | <i>IF A > B</i> |
| | <i>B=B+1</i> |
| | <i>Write (B)</i> |

זמן בר סידור מבט וSKUל מבט בזמן הסדרתי $T2 \rightarrow T1$ אין מעגל מכoon, הזמן בר סידור קונפליקט ולכן גם בר סידור מבט

זמן בר סידור מבט (המשר)

- האם הזמן הבא הינו בר סידור מבט?

| | |
|---|--|
| <p>T1 <i>Read A</i> <i>Read B</i></p> <p><i>IF A > B</i> <i>B=B+1</i> <i>Write (B)</i></p> | <p>T2</p> <p><i>Read A</i> <i>Read B</i></p> <p><i>IF A > B</i> <i>B=B+1</i> <i>Write (B)</i></p> |
|---|--|

זמן לא שקוֹל מבט לזמן הסדרתי
 $T1 \rightarrow T2 \rightarrow S2$ טרנסזקציה T
קוראת ערך התחלתי ל- B ובזמן
הסדרתי – לא.

זמן לא שקוֹל מבט לזמן הסדרתי
 $T2 \rightarrow T1 \rightarrow S2$ הכתיבה
האחרונה ל- B היא של T2 ובזמן
הסדרתי – לא.

זמן אינו בר סידור מבט

זמן בר אישוש (Recoverability)

- **זמן בר אישוש (recoverable schedule)** – אם טרנסקציה T קוראת ערך שנכתב ע"י טרנסקציה Z , אז על טרנסקציה Z להסתיים בהצלחה (commit) לפני הסיום המוצלח של טרנסקציה T .
- מדוע? טרנסקציה T קוראת ערך שנכתב ע"י טרנסקציה Z וזה commit של Z יבוצע לפני ה $commit$ של Z , יכול להיווצר מצב שם Z לא תסתיים בהצלחה, הערך שכתבה ימחק(לא ישמר) ← הערך ש T קראה יהיה שגוי.
- נקרא בר אישוש, כי במצב של אי הצלחת טרנסקציה, שומר על בסיס נתונים עקיبي.

זמן בר אישור (המשר)

- האם הזמן הבא הינו בר אישוש?

| | |
|--|---|
| <p><u>T1</u> Read A Read B</p> <p><i>IF A > B B=B+1 Write (B) commit</i></p> | <p><u>T2</u></p> <p>Read A Read B</p> <p><i>IF A > B B=B+1 Write (B) commit</i></p> |
|--|---|

- הזמן בר אישור
- T1 לא קוראת ערך ש T2 כתבה
- T2 לא קוראת ערך ש T1 כתבה
- במקרה זה אין משמעות לסדר ה commit

זמן בר אישור (המשר)

- האם הזמן הבא הינו בר אישוש?

| | |
|---|--|
| <p><u>T1</u></p> <p>Read A</p> <p>Read B</p> <p>IF A > B</p> <p>B = B +1</p> <p>Write B</p> <p>Read C</p> <p>Write C</p> <p>Commit</p> | <p><u>T2</u></p> <p>Read A</p> <p>Read B</p> <p>IF A > B</p> <p>B = B +1</p> <p>Write B</p> <p>Commit</p> |
|---|--|

- הזמן אינו בר אישור
- T2 קוראת ערך B ש T1 כתבה
(ולכן נרצה שה commit של T1
יתבצע לפני ה commit של T2)
- ה commit של T2 מתבצע לפני ה
commit של T1

- **זמן cascadeless** - כשלון של טרנזקציה אחת גורם לטרנזקציה
אחרת או יותר לבצע rollback
- הטרנזקציות שיכשלו הן אלו הקוראות ערך שנכתב ע"י
הטרנזקציה בה קرتה התקלה.
- התקלה אחת עלולה לגרום לאבדן כמות רבה של עבודה
(יקר מאד מבחינת זמן ביצוע).
- בזמן Cascadeless, אם טרנזקציה זו קוראת ערך שנכתב ע"י
טרנזקציה אחרת, אז על טרנזקציה אחרת לבצע commit לפני הקריאה
בטרנזקציה אחרת. כלומר ניתן לקרוא רק ערכים שהם committed.
- התנאי הוא מספיק כדי להימנע מ-Cascading Rollbacks
- **כל הזמן** Cascadeless הינו בר אישוש. (מחמיר יותר)

זמן (הmarsh) cascadeless

| <u>T1</u> Read A $A = A + 10$ Write A | <u>T2</u> <i>Read A</i> <i>$A = A + 20$</i> <i>Write A</i> | <u>T3</u> <i>Read A</i> <i>$A = A + 30$</i> <i>Write A</i> |
|--|--|--|
| Commit | <i>Commit</i> | <i>Commit</i> |

- האם הזמן בר אישוש?
- הזמן בר אישוש: כל טרנסקציה הקוראת פריט אחריו שנכתב ע"י טרנסקציה אחרת מבצעת את ה-commit לאחר ה commit של הטרנסקציה ממנה קראה את הפריט.
- האם הזמן ?cascadeless
זמן אינו cascadeless
הקריאה ב T2 של ערך A מתבצעת לפני ביצוע ה commit ב T1
הקריאה ב T3 של ערך A מתבצעת לפני ביצוע ה commit ב T2

במקרה זה הזמן חייב להיות סדרתי
(T1 → T2 → T3) כדי להיות .cascadeless

טרנזקציות - תרגיל כתה



- לפניכם 4 טרנזאקטזיות המגייעות למערכת:

| <i>T1</i> | <i>T2</i> | <i>T3</i> | <i>T4</i> |
|---------------|---------------|---------------|---------------|
| $W(X=3)$ | $R(X)$ | $R(Z)$ | $W(Z=3)$ |
| $R(X)$ | $W(X=5^*X)$ | $R(F)$ | $R(F)$ |
| $R(F)$ | $W(Z=X)$ | $W(Z=F^*Z)$ | $R(X)$ |
| $W(F=X^*F)$ | <i>commit</i> | $R(Y)$ | $W(Y=X-F)$ |
| $R(Y)$ | | <i>commit</i> | <i>commit</i> |
| <i>commit</i> | | | |

- עבור כל סעיף, האם קיימים זמן מקבילי (לא סדרתי!!) של 4 הטרנזאקטזיות הנ"ל המקיים את כל התנאים? אם יש זמן כזה, יש להציגו ולהסביר כיצד הוא מקיים את התנאים. אם אין זמן כזה, יש להסביר מדוע לא קיימים.

טרנסקציותות - תרגיל כתה

1. **זמן מקבילי** שהינו בר סידור קונפליקט, בר אישוש
אר לא cascadeless.
2. **זמן מקבילי** שהינו בר סידור קונפליקט ו-
cascadeless
3. **זמן** שהינו בר סידור מבט, אינו בר סידור קונפליקט
ואיננו בר אישוש



פרוטוקולים לברית מקבילים

- המטרה: להגדיר פרוטוקולים המבטיחים "זמן טובים".
 - ברι סידור קונפליקט או מבט cascadeless או אישוש
- יתרונות בהרצת טרנסקציות במקביל:
 - אפשר ניצול טוב יותר של זמן O/I ע"י ביצוע חישובים O/במקביל.
 - קצר زمنית תגובה (טרנסקציה קצרה אינה מחייבת טרנסקציה ארוכה שהתחילה לפני).
- נעילה
 - נעילה הינה הגבלת גישה סלקטיבית לנוטרי מסד הנתונים.
 - רק חלק מסד הנתונים מושפע.
 - רק חלק מהגישות מושפעות.

פרוטוקולים לבקורת מקבילות

- ניתן לבצע בקרת מקבילות ע"י נעילות של פריטי מידע.
- פריט מידע יכול להינעל באחד משתי הנסיבות:
 - משותף (S) (אפשר קריאה בלבד ונitinן לכמה טרנסקציות במקביל).
 - בלעדי (X) (אפשר גם כתיבה ורק טרנסקציה אחת יכולה להחזיק במנעל מסוג זה על אותו פריט ובפרט שאין לטרנסקציה אחרת מנעל מסוג S על אותו פריט מידע).
- טרנסקציה ניגשת לפרט מידע ע"י רכישת מנעל לאותו פריט מידע.
- אם הבקשה נדחתה הטרנסקציה נכנסת למצב של wait (במצב זה הטרן' ממתינה לקבלת אישור לרכישת המנעל והיא מנועה מהתקדם כל עוד הבקשה נדחתה).
- לעיתים הגבלת גישה למידע ע"י מנעל מגבילה את הזמן לסדרתיים בלבד.

| | Locked-S | Locked-X |
|--------|----------|----------|
| Lock-S | true | false |
| Lock-X | false | false |

פרוטוקולים לברחת מקבילות

- **רמות הנעילה (Locking Granularity)** – תחומי מסד הנתונים שכפפים להוראות הנעילה
 - שדה של רשומה
 - רשומה
 - דף פיסי (בלוק)
 - טבלה
 - מסד הנתונים
- לא כל שימוש בנעלות יכול להבטיח זמנים " טובים "

פרוטוקולים לברור מקבילות

- ניתן להגדיר בעזרת מניעולים פרוטוקולים המבטיחים זמנים ברישידור.
- הקשיים:
 - שימוש במניעולים לא מבטיח זמן בר סידור. זמנים סדרתיים מבטיחים "תשובות נכונות" Caucus לא הייתה מקבילות (כלומר בידוד Isolation).
 - יש להימנע ממצב של **deadlock**, מצב בו טרנזקציה / מחזיקה מניעול על A בעוד מחרכה למניעול על B, ובאותה העת טרנזקציה / מחזיקה מניעול על B ומחרכה למניעול על A.
 - יש להימנע ממצב של **starvation**, מצב בו טרנזקציה מחרכה למניעול לנצח" (זמן ארוך מדי). לדוגמה, טרנזקציה יכולה לחכות למניעול מסווג X, בעוד שסדרה אחרת של טרנזקציות מקבלת מניעול S על אותו פריט מידע.

Two-Phase Locking Protocol (2PL)

- עפ"י פרוטוקול 2PL ישנו שתי פאזהות בחיה טרנזקציה
 - פאזה הגדילה
 - הטרנזקציה יכולה לרכוש מנעולים
 - הטרנזקציה מנועה משחרר מנעולים
 - פאזה התכווצות
 - הטרנזקציה יכולה לשחרר מנעולים
 - הטרנזקציה מנועה מרכוש מנעולים חדשים
- המשמעות ביחס לטרנזקציה אחת, מיציאת נקודה שעד אליה רוכשים מנעולים בלבד, ומנקודת זו ועד סוף הטרנזקציה משחררים מנעולים בלבד.



Two-Phase Locking Protocol (2PL) (המשך)

- נתונות 3 הטרנסקציות הבאות:

T_1 מכירת כרטיס

Read A ()

Read B ()

IF A > B

B = B + 1

Write B

ELSE

Print "No Tickets Left"

T_2 החזרת כרטיס

Read B (ShowExt.TicketsSold)

$B = B - 1$

Write B

הדפסת מספר מקומות פנויים
שנותרו

Read A (TheaterExt.NumberOfSeats)

Read B (ShowExt.TicketsSold)

Print A - B

Two-Phase Locking Protocol (2PL)

- זמן מקבילי 2PL עברו הטרנץקציות הנ"ל

Lock-S(A)

Read A

Lock-X(B)

Read B

$B = B - 1$

Write B

Unlock(B)

Lock-X(B)

Read B

Unlock(A)

IF A > B

$B = B + 1$

Write B

Unlock(B)

Lock-S(A)

Read A

Lock-S(B)

Unlock(A)

Read B

Unlock(B)

Print A - B

- מהם מאפייני הזמן

– בר סידור מבט?

– בר סידור קונפליקט?

Two-Phase Locking Protocol (2PL)

- חסרוןות 2PL:
 - יתכנו Deadlocks (טרנזקציה ב- deadlock מנועה מלהתקדם):
 - טרנזקציה / מחזיקה מנעול על A ועודיה מחכה למנעול על B,
ובאותה העת טרנזקציה / מחזיקה מנעול על B ומחכה למנעול
על A.
 - יתכן Starvation:
 - טרנזקציה יכולה לחייב למנעול מסווג X, בעוד שסדרה
אחרת של טרנזקציות מקבלת מנעולי X או סעל אותו פריט
מידע
 - יתכנו Cascading Rollbacks:
 - ניתן למשם Strict 2PL – טרנזקציה מחזיקה את כל המנעולים מסווג Exclusive עד לביצוע commit/abort
 - ניתן למשם Rigorous 2PL – טרנזקציה מחזיקה את כל המנעולים (מכל סוג) עד לביצוע commit/abort (מחמיר!)

פרוטוקולים למניעת deadlocks

- Wait-die schema
 - טרנסקציות ישנות מוחכות עד שטרנסקציות חדשות יותר ישחררו מנעולים.
 - טרנסקציות חדשות לא מוחכות לטרנסקציות ישנות יותר, הן מבצעות rollback.
 - מונע starvation.
 - כל טרנסקציה ישנה יותר היא נוטה לחכotta יותר. והדרת פני זkan!!
- Wound-wait schema
 - טרנסקציה יכולה "למות" במספר פעמים.
 - טרנסקציות ישנות "פוצעות" (מחייבות rollback) טרנסקציות חדשות יותר במקום לחכotta.
 - טרנסקציות חדשות עשוות לחכotta לטרנסקציות ישנות יותר.
 - מונע starvation.
 - פחות rollbacks מאשר ב프וטוקול wait-die.

אלגוריתם לזייה deadlocks

- בניית **wait-graph** בזמן המבוצע בצורה דינמית, ככלומר תוך כדי שהזמן מ.toolbox.
- **wait-graph** הינו גרף מכoon בו:
 - הצמתים מייצגים טרנסזקציות בביוץ.
 - קיימת קשת $T_j \rightarrow T_i$ אם ורק אם טרנסזקציה T_j מתחילה למנעול על פריט מידע כלשהו שנעול ע"י טרנסזקציה T_i .
 - המערכת נמצאת במצב של deadlock אם ורק אם יש בו- **wait-graph** מעגל מכoon.
- כאשר **deadlock** מתגלה:
 - טרנסזקציה אחת מتوزן המעגל נבחרת ומבצעת **rollback** (בהתאם לפרוטוקול)
 - הבחירה ממבצעת בהתאם לעלות **rollback** מינימלי.
 - ניתן לבצע **rollback** חלקי
 - יתכן מצב של הרעבה במידה ואוותה טרנסזקציה נבחרת כל פעם לבצע **rollback**.

• נתון סט הפעולות (הטרנזקציות) הבא:

| T1 | T2 | T3 | T4 |
|----------------|---------------|----------------------|---------------|
| $R(F)$ | $W(Y=5)$ | $R(F)$ | $R(Z)$ |
| $R(Z)$ | $R(F)$ | $W(F=F^*3)$ | $R(F)$ |
| $R(X)$ | $R(Z)$ | <i>if</i> ($F>80$) | $R(X)$ |
| $W(X=F^*Z^*X)$ | $W(X=F^*Z)$ | $W(X=F)$ | $W(Z=F+X)$ |
| <i>commit</i> | <i>commit</i> | $R(Y)$ | $W(X=Z-F)$ |
| | | <i>commit</i> | <i>commit</i> |

1. הראה ביצוע מקבילי (לא סדרתי!) של הטרנזקציות כר' הזמן המתקובל יעבד תחת פרוטוקול 2PL ללא deadlocks. צין תחת איזו גרסה של 2PL הוא עובד (ריגילה, strict, או rigorous). מהם מאפייני הזמן (בר סידור קונפליקט, בר סידור מבט, בר אישוש, ?(cascadeless

| T1 | T2 | T3 | T4 |
|---|--|--|---|
| <i>Lock-S(F) R(F)</i> | <i>Lock-X(Y) W(Y=5) Lock-S(F) R(F)</i> | <i>Lock-S(F) R(F) WaitForLockX(F)</i> | <i>Lock-S(Z) R(Z) Lock-S(F) R(F)</i> |
| <i>Lock-S(Z) R(Z)</i> | <i>Lock-S(Z) R(Z)</i> | <i>WaitForLockX(X)</i> | |
| <i>Lock-X(X) R(X) W(X=F'Z*X) Unlock(F) Unlock(Z) Unlock(X) commit</i> | | | <i>Lock-X(X) W(X=F'Z) Unlock(Y) Unlock(F) Unlock(Z) Unlock(X) commit</i> |
| | | <i>Upgrade(F) W(F=F*3) Lock-X(X) If (F>80) W(X=F) Lock-S(Y) R(Y) Unlock(F) Unlock(X) Unlock(Y) commit</i> | <i>Lock-X(X) R(X) Upgrade(Z) W(Z=F+X) W(X=Z-F) Unlock(Z) Unlock(F) Unlock(X) commit</i> |

בנייה של Wait-graph עבור הזמן הנוכחי:
אין מעגל מכ้อน, אין deadlocks

מאפייני הזמן:
זמן הנהו בר סידור
كونפליקט - להלן גרפ
קדימות: אין מעגל מכ้อน,
זמן בר סידור קוןפליקט
לזמן סדרתי:

$T_1 \rightarrow T_2 \rightarrow T_4 \rightarrow T_3$
זמן הנהו cascadeless גם

בר אישוש (כל זמן cascadeless)
T3 קוראת ערך Y ש T2 כתבה, ה commit של
T2 מתרבע לפני הקראיה של ערך Y ב T3.
T4 קוראת ערך X ש T2 כתבה, ה commit של
T2 מתרבע לפני הקראיה של ערך X ב T4.

– דוגמא (המשך) – 2PL

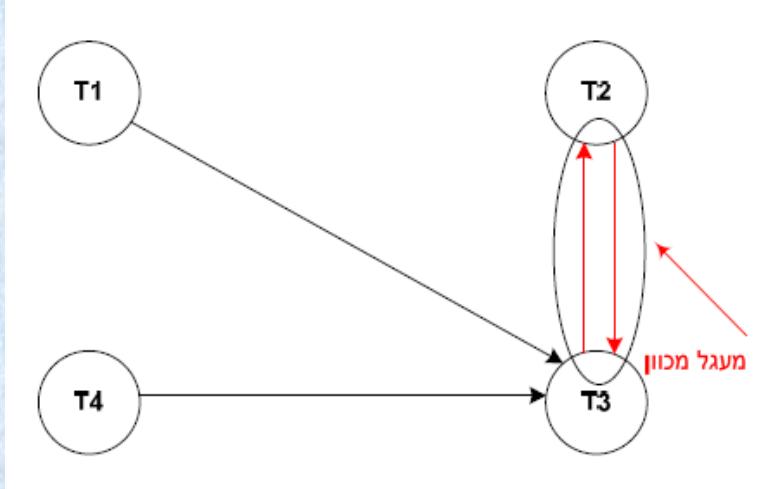
2. הראה תרחיש של deadlock בעת ביצוע מקבילי של הטרנזקציות תחת פרוטוקול 2PL. פטור את ה- deadlock ע"י שימוש באחת הסכומות (wait-die/wound-wait) וציין מהם מאפייני הזמן.

| T1 | T2 | T3 | T4 |
|-------------------------|-----------------------------|---------------------------|---------------------------|
| <i>WaitForLock-S(F)</i> | <i>Lock-X(Y) W(Y=5)</i> | <i>Lock-X(F) R(F)</i> | <i>Lock-S(Z) R(Z)</i> |

A red oval highlights the sequence of operations for T2, T3, and T4:

- T2:** *WaitForLock-S(F)*
- T3:** *Lock-X(X)
if (F>80)
W(X=F)*
waitForLock-S(Y)
- T4:** *WaitForLock-S(F)*

- בניית wait graph עבור הזמן:



- נבחר לפתר את ה deadlock באמצעות סכמה : Wound-die
 - טרנסקציות ישנות מוחכות עד שטרנסקציות חדשות יחררו מנעולים
 - טרנסקציות חדשות לא מוחכות לטרנסקציות ישנות יותר, הן מבצעות rollback
 - כאשר T3 הגיע למצב של Deadlock, מכיוון שהנה חדשה לעומת T2, היא לא תחכה לה ותבצע Rollback.

– דוגמא (המשר) – 2PL

- תאר מצב של starvation תחת פרוטוקול 2PL של הטרנזאקטיות הנ"ל.
 - הנחה: כאשר הטרנזאקטיות מבצעות commit הן מתייחסות לרוץ שוב.

| T1 | T2 | T3 | T4 |
|---|--|----|--|
| <i>Lock-s(F) R(F)</i> | <i>Lock-X(Y) W(Y=5)</i> | | <i>Lock-S(Z) R(Z)</i> |
| <i>Lock-S(Z) R(Z)</i> | <i>Lock-S(F) R(F) waitForLockX(F)</i> | | <i>Lock-S(F) R(F)</i> |
| <i>Lock-X(X) R(X) W(X=F*Y*Z) Unlock(F) Unlock(Z) Unlock(X) commit</i> | <i>Lock-S(F) R(F) Lock-S(Z) R(Z) Lock-X(X) W(X=F*Z) Unlock(Y) Unlock(Z) Unlock(X) commit</i> | | <i>Lock-X(X) R(X)</i> |
| <i>Lock-s(F) R(F)</i> | <i>Lock-X(Y) W(Y=5)</i> | | <i>Upgrade(Z) W(Z=F+X) W(X=Z-F) Unlock(Z) Unlock(F) Unlock(X) commit Lock-S(Z) R(Z) Lock-S(F) R(F)</i> |
| <i>Lock-S(Z) R(Z)</i> | | | |

הסביר לקיום התנאים:
 T3 מנסה לרכוש מנעול X על F,
 (צריכה לכתוב וכרגע רכשה רק
 מנעול מסוג S), T2 T4 , T1 ,
 מתחילה מחדש כל פעם
 שמסתיימות, מכיוון שיש בהם
 חפיפה על האחזקה של מנעול מסוג
 S על F , T3 אינה יכולה לבצע
 Upgrade למנעול X על F ולכן
 מנסה לנצל - נוצר מצב של הרעה.



כתרת-Two phase locking

- לפניכם 4 טרנזאקטיות המגיעות למערכת:

| <i>T1</i> | <i>T2</i> | <i>T3</i> | <i>T4</i> |
|-----------------|-----------------|-----------------|-----------------|
| <i>W(X=3)</i> | <i>R(X)</i> | <i>R(Z)</i> | <i>W(Z=3)</i> |
| <i>R(X)</i> | <i>W(X=5*X)</i> | <i>R(F)</i> | <i>R(F)</i> |
| <i>R(F)</i> | <i>W(Z=X)</i> | <i>W(Z=F*Z)</i> | <i>R(X)</i> |
| <i>W(F=X*F)</i> | <i>commit</i> | <i>R(Y)</i> | <i>W(Y=X-F)</i> |
| <i>R(Y)</i> | | <i>commit</i> | <i>commit</i> |
| <i>commit</i> | | | |

- עבור כל סעיף, האם קיימן זמן מקבילי (**לא סדרתי!!**) של 4 הטרנזאקטיות הנ"ל המקיים את כל התנאים? אם יש זמן כזה, יש להציגו ולהסביר כיצד הוא מקיים את התנאים. אם אין זמן כזה, יש להסביר מדוע לא קיים.

Two phase locking - תרגיל כתה

1. זמן מקבילי המתקיים תחת פרוטוקול 2PL ללא deadlocks. ציין מה מאפייני הזמן: בר סידור קונפליקט, בר סידור מבט, בר אישוש, cascadeless.
2. זמן מקבילי המתקיים תחת פרוטוקול 2PL עם deadlock אחד. הציעו פתרון ל- deadlock less寥無死鎖的. לפי אחת השיטות שנלמדו (wait wound או die wound) וציינו במפורש באיזו שיטה בחרתם.
3. זמן מקבילי היוצר מצב של starvation (ניתן להריץ את הטרנסקציות מס' פעמים לצורך מטרה זו).

פרוטוקול Timestamp ordering

- כל טרנסקציה T_i מקבלת תווית זמן ($TS(T_i)$) עם כניסה למערכת.
- $T_i \Rightarrow TS(T_j) < TS(T_i) \Leftrightarrow T_i$ נכנסה לפני T_j .
- בסיום המערכת מתחזקת עברו כל פריט מידע Q שני נתוניים:
 - $(Q)-Timestamp-W$ – תווית הזמן הגדולה ביותר של טרנסקציה שכתבה ל- Q .
 - $(Q)-Timestamp-R$ – תווית הזמן הגדולה ביותר של טרנסקציה שקרה מ- Q .
- הפרוטוקול מבטיח שפעולות מתנגשות יבוצעו לפי סדר ה- timestamp של הטרנסקציות המתאימות
- הפרוטוקול מבטיח שלא יהיו deadlocks

פרוטוקול Timestamp ordering (המשר)

- טרנסקציה T_i מבקשת לבצע פעולה קרייה על פריט Q :
 - אם $(TS(Q) < W-Timestamp(Q))$ אזי: הקריאה לא מתאפשרת ו- T_i מבצעת rollback (בגלל ש- T_i צריכה לקרוא Q שנכתב ע"י טרנסקציה מאוחרת יותר (התחילת אחרי T_i)).
 - אחרת - הקריאה מתאפשרת.
 - $\{ (T_i, TS(Q), R-Timestamp(Q) = \text{MAX} \{ R-Timestamp(Q), TS(T_i) \}) \}$.
- טרנסקציה T_i מבקשת לבצע פעולה כתיבה על פריט Q :
 - $(R-Timestamp(Q) < TS(T_i))$ אזי: הכתיבה לא מתאפשרת ו- T_i מבצעת rollback (בגלל שערכו של Q ש- T_i כתבה היה נחוץ קודם והמערכת הינה שערך זה לא יעודכן).
 - $(W-Timestamp(Q) < TS(T_i))$ אזי: הכתיבה לא מתאפשרת ו- T_i מבצעת rollback (בגלל ש- T_i מנסה לכתוב ערך מישן ל- Q).
 - אחרת - הקריאה מתאפשרת.
 - $TS(T_i) = TS(Q) = W-Timestamp(Q)$.

| T1 | T2 | T3 | T4 | |
|--|---|----------------|--|---|
| $R(F)$ $R(Z)$ $R(X)$ | $W(Y=5)$ $R(F)$ $R(Z)$ $W(X=F^*Z)$ | $R(F)$ | $R(Z)$ $R(F)$ $R(X)$ $W(Z=F+X)$ | $TS(T1)=1$ $TS(T2)=2$ $TS(T4)=3$ $TS(T3)=4$ $TS(T1) < W-TS(X), R-TS(X)$ <i>Rollback Restart (T1)</i> |
| $R(F)$ | $W(F=F^*3)$ | | | $W-TS(F)=4$ $R-TS(F)=5$ $W-TS(X)=3$ $W-TS(Z)=3$ |
| $R(Z)$ | $IF(F>80)$ $W(X=F)$ | | $W(X=Z-F)$ | $W-TS(X)=4$ |
| $R(X)$ $W(X=F^*Z^*X)$ | <i>commit</i> | $R(Y)$ | <i>commit</i> | $R-TS(Z)=5$ $R-TS(Y)=4$ $R-TS(X)=5$ $W-TS(X)=5$ |
| | | | | <i>commit</i> |

Thomas's Write Rule

- עידון קל לפרוטוקול Timestamp ordering.
- אם טרנסקציה T_i מבקשת לכתוב לפריט Q , ומתקיים $TS(Q) < TS(T_i)$:
 - לא אפשר כתיבה של ערך מיושן.
 - אין שום סיבה לבצע rollback ל- T_i .
- פרוטוקול Timestamp ordering תחת עידון זה מאפשר זמנים ברוי סידור מבט שאינם ברוי סידור קונפליקט.
- דוגמא לזמן מקבילי המתקבל תחת פרוטוקול timestamp ordering, בו יש הפעלה (לפחות אחת) של Thomas's Write rule ושאינו מבצע כל Rollbacks

**הסבר לקיום התנאים
(ביצוע של תנאים : (write rule**

- T2 מבקשת לכתוב ערך X: ולכן לא מתאפשרת כתיבה של ערך מיושן, 2 לא כותבת ערך X ואינה מבצעת rollback.

- T4 מבקשת לכתוב ערך X: ולכן לא מתאפשרת כתיבה של ערך מיושן, 4 לא כותבת ערך X ואינה מבצעת rollback.

- T1 מבקשת לכתוב ערך X: ולכן לא מתאפשרת כתיבה של ערך מיושן, 1 לא כותבת ערך X ואינה מבצעת rollback.

| T1 | T2 | T3 | T4 | |
|-------------------------|--------------------|---|-----------------------------------|--|
| $R(F)$ | $W(Y=5)$ $R(F)$ | | | $TS(T1)=1$ $R-TS(F)=1$ $TS(T2)=2$ $W-TS(Y)=2$ $R-TS(F)=2$ $R-TS(Z)=1$ $R-TS(Z)=2$ |
| $R(Z)$ | $R(Z)$ | | $R(Z)$ $R(F)$ | $TS(T4)=3$ $R-TS(Z)=3$ $R-TS(F)=3$ $TS(T3)=4$ $R-TS(F)=4$ $W-TS(F)=4$ $R-TS(X)=1$ $R-TS(X)=3$ $W-TS(X)=4$ |
| $R(X)$ | | $R(F)$ $W(F=F^*3)$ $IF(F>80)$ $W(X=F)$ | $R(X)$ | $W-TS(Z)=3$ $TS(T2) < W-TS(X)$ <i>Thomas's Write Rule</i> $TS(T4) < W-TS(X)$ <i>Thomas's Write Rule</i> $R-TS(Y)=4$ $TS(T1) < W-TS(X)$ <i>Thomas's Write Rule</i> |
| $\cancel{W(X=F^*Z)}$ | | | $W(Z=F+X)$ $\cancel{W(X=Z-F)}$ | |
| $\cancel{W(X=F^*Z^*X)}$ | <i>commit</i> | <i>commit</i> | <i>commit</i> | |
| <i>commit</i> | | | | |



Timestamp ordering - תרגיל כתה

- לפניכם 4 טרנזאקטיות המגיעות למערכת:

| T_1 | T_2 | T_3 | T_4 |
|---------------|---------------|---------------|---------------|
| $W(X=3)$ | $R(X)$ | $R(Z)$ | $W(Z=3)$ |
| $R(X)$ | $W(X=5^*X)$ | $R(F)$ | $R(F)$ |
| $R(F)$ | $W(Z=X)$ | $W(Z=F^*Z)$ | $R(X)$ |
| $W(F=X^*F)$ | <i>commit</i> | $R(Y)$ | $W(Y=X-F)$ |
| $R(Y)$ | | <i>commit</i> | <i>commit</i> |
| <i>commit</i> | | | |

- עבור כל סעיף, האם קיימים זמן מקבילי (לא סדרתי!!) של 4 הטרנזאקטיות הנ"ל המקיימים את כל התנאים? אם יש זמן כזה, יש להציגו ולהסביר כיצד הוא מקיים את התנאים. אם אין זמן כזה, יש להסביר מדוע לא קיים.

Timestamp ordering - תרגיל כתה

1. זמן מקבילי המתקבל תחת פרוטוקול timestamp ordering ושהינו בר אישוש.
2. זמן מקבילי המתקבל תחת פרוטוקול timestamp ordering ובו יש פעולה (פחות אחת) של Rollbacks אר אין כלל Thomas's Write Rule.



טיפול בTRANSACTION בשפה SQL

- לא כל מערכות מסדי הנתונים המסחריות מטפלות באופן זהה בTRANSACTION
- **שיטה א':**
 - כל שאלתה הינה טרנסקציה
 - כדי להרכיב טרנסקציה המורכבת מיותר משאלתה אחת, יש להשתמש במילה שמורה המגדירה את תחילת הטרנסקציה (start/begin transaction)
 - הטרנסקציה תסתיים ב- commit/rollback
- **שיטה ב':**
 - יש לציין באופן מפורש commit/rollback עבור כל טרנסקציה

TRANSACTION ב- SQL

- טנזקציה מכילה מספר פקודות SQL לקריאה ועדכון של מסד נתונים. הטרנזקציה מתארת מספר פעולות הנשלחות מהלך, לביצוע כיחידה אחת.
- טנזקציה ה כוללת מספר פקודות SQL אינה מבצעת שינוי קבוע ב- DB עד לביצוע ה- COMMIT TRANSACTION.
- טנזקציה מתחילה במלחים: BEGIN TRAN. הזוג begin...commit גורם לפקודות שביניהם להתבצע כיחידה אחת במטרה לשמר על מאפייני ACID של ה- DB (Atomicity, Consistency, Isolation, and Durability).
- טנזקציה מבטלת את כל השינויים (הזמןניים) ברגע שמתבצע ROLLBACK TRANSACTION

טרנסקציה ב- SQL (המשך)

BEGIN TRAN

INSERT authors VALUES (etc.)

GO

SELECT * FROM authors

GO

UPDATE publishers SET pub_id = (etc.)

GO

COMMIT TRAN

GO

טרנסקציה ב- SQL (המשך)

- תסריט הכלל 2 טרנסקציות

BEGIN TRAN

INSERT authors VALUES (etc.)

SELECT * FROM authors

COMMIT TRAN

BEGIN TRAN

UPDATE publishers SET pub_id = (etc.)

INSERT publishers VALUES (etc.)

COMMIT TRAN

GO



מערכת ה RDBMS
קולעת את הטרנסקציות
"ודואגת" לסדר נכון

- ב- SQL Server רק כאשר מתרחשת תקלת קרייטית (fatal error) הגורמת לנפילת המערכת, מבצעים rollback לטרנזקציה. אם המערכת יכולה מפסיקה לתקן SQL Server צריך להיות מאותחל מחדש ותהליך ההטאוששות האוטומטי יבצע rollback לכל הטרנזקציות שלא הושלמו. לכל תקלת אחרת, רק המשפט שגורם לתקלה הנוכחית יבוטל (Aborted) ושאר המשפטים בטרנזקציה ימשיכו להתבצע אף יבצעו commit.