שנקר

בית הספר לתעשייה וניהול
בסיס נתונים 31-028-24

תרגול מס' 1

מטרת הקורס

מטרת קורס זה הינה להקנות לסטודנטים היכרות ראשונית בעבודה למול בסיסי נתונים.
בקורס נחקור את אלגברת היחסים עליה מושתת השפה, נבין את הצורך בבסיסי נתונים בחיי היומיום, נלמד את רכיבי השפה והכלים הקיימים בכדי לבנות שאילתות יעילות לשליפת נתונים. התרגולים ילוו במקרי בוחן (case studies) מהתעשייה ויאפשרו לסטודנטים להתנסות בעבודה פרקטית בנוסף לידע התאורטי המועבר בשיעורים.

מנהלות

התכנים יועלו לאתר MOODLE ובמקביל לאתר המרצה http://www.korneto.com/courses.html

אימייל לשאלות ובעיות: roeizer@shenkar.ac.il או דרך "צור קשר" באתר האינטרנט.

מופעי שיעורים

כל התרגולים זהים מבחינת התוכן אך חשוב **להתמיד באותה קבוצת תרגול** כדי לשמור על רצף התכנים, כמו כן יש להגיש את התרגילים לפי קבוצות הלימוד בהם אתם רשומים.

שנקר

בית הספר לתעשייה וניהול
בסיס נתונים 31-028-24

הגדרות בסיסיות:

1) שדה Field: אוסף תווים בעלי קשר לוגי, יחידת הנתונים הקטנה ביותר (שם, ת.ז.), כאשר נוכל להגביל גודל שדה לכמות תווים מסוימת (ת.ז. תוגבל ל 9 ספרות)

2) רשומה Record: אוסף של שדות בעלי קשר לוגי (רשומת סטודנט, פרטי חשבון בנק).

לדוגמא (פרטי סטודנט):

שם פרטי	שם משפחה	מספר ת"ז	טלפון	משכורת
↓	↓	↓	↓	↓
10 תוים	10 תוים	9 ספרות	10 ספרות	4 ספרות (☹)

3) קובץ File: אוסף של רשומות [הבעיה היא שכל שדה יכול להיות בגודל מסויים מה שיצור מבנה לא אחיד, ואם נרצה לשמור על מבנה אחיד נצטרך בעצם להשחיל רווחים].

בעיות שיכולות להיגרם לנו בעבודה עם מערכת קבצים ?

- כל תוכנית שתרצה לעבוד עם הקובץ בעצם תדרוש לדעת את **המבנה הפיזי של הקובץ**.
- **תמיכה בשינויים** במבנה הקובץ (כל שינוי יצטרך להיות מעודכן כדי לתמוך בכל הניגשים לקובץ).
- כדי **למצוא רשומה צריך היה לסרוק את כל הקובץ**, בתקווה שהרשומה הדרושה איננה האחרונה.
- בעיה של **אבטחת מידע** (לא נרצה להציג את כל השדות, כמו טלפון של המתרגל בבית).
- **כפילויות ואי אמינות** (אם יש קובץ אב ושני קבצי בן אזי שינוי נתון בקובץ בן-1 לא יגרור שינוי בקובץ בן-2).
- **תקינות הנתונים** (שינוי שם יתכן ויתבצע רק בחלק מהקבצים ולא בכולם).
- בעיה לחבר בין מספר קבצים **ולשתף מידע** ביניהם.
- **ריבוי משתמשים**.
- **בעיה למיין** לפי שדות פנימיים.

← זו בעצם השיטה הישנה לאחסון נתונים, כי אנו בעצם שומרים את המידע בצורה סדרתית, לדוגמא את הרשומה למעלה היינו רושמים בקובץ בצורה הבאה:

Roei######Zerahia###012345678077123456720000

🡸 הפתרון לכל בעיות אלו: שימוש בבסיס נתונים !
המבנה הפיזי של הנתונים הופך לשקוף עבור המשתמש ומאפשר שיתוף בין התוכניות השונות.

4) בסיס נתונים (Database) הוא בעצם אוסף של נתונים מסוגים שונים המקושרים ביניהם בקשר לוגי כלשהו.

5) מערכת ניהול בסיסי נתונים (DBMS), שזו בעצם מערכת (תוכנה/אפליקציה) לניהול בסיס נתונים המאפשרת יצירה, תחזוקה, שימוש וגישה לנתונים עם אפשרות לעבודה של ריבוי משתמשים במקביל.

דוגמא למסד נתונים הקיים כמעט בכל בית בישראל

לכל משתמש בייתי או מוסדי העובד עם מערכת הפעלה של חלונות וודאי יש את חבילת האופיס מותקנת גם כן, חבילת האופיס מכילה 5 תוכנות עיקריות אשר מיקרוסופט חשבה לנכון שכל משתמש זקוק להן:

- Word – מעבד תמלילים.
- Excel – גיליון אלקטרוני.
- Power Point – תוכנה לבניית מצגות.
- Outlook – תוכנה לשליחה וקבלת מיילים.
- Access – מסד נתונים (DBMS).

ז"א שחברת מיקרוסופט חשבה שתוכנה לניהול בסיסי נתונים היא תוכנה מספיק חשובה כדי לצרפה לחבילת ה office.

הדרישות מבסיס הנתונים:

1) אחסון נתונים ושליפה נוחה
2) אבטחת מידע (רק בעל הרשאה יכול לגשת לנתונים).
3) ריבוי משתמשים בו זמנית.
4) פעולות אוטומטיות (גיבוי, שיחזור, העתקה, כיווץ וכד').

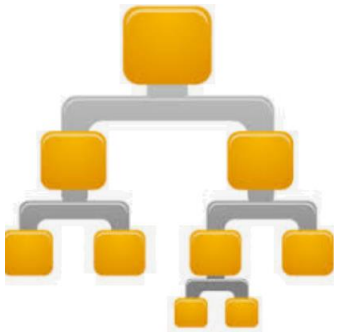
מודלים של בסיסי נתונים:

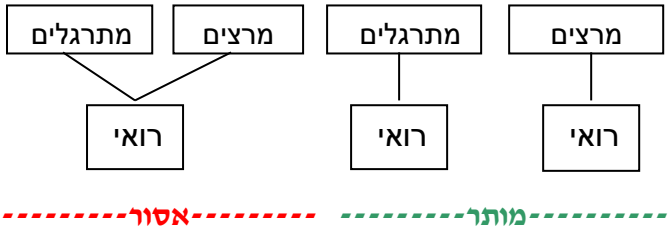
1) מודל היררכי
2) מודל רישתי
3) מודל יחסי/טבלאי

🡸 נעבור על המודלים הנ"ל ונראה את היתרונות והחסרונות של כל אחד מהם, כאשר ניתן לשים לב שהסיבה להתפתחות מודל מתקדם בעל יתרונות מסוימים הינה בעקבות חסרונות שהועלו מהמודל שקדם לו.

1) המודל ההיררכי:

זהו מודל שבו הנתונים מסודרים במבנה של עץ, כאשר קודקודי העץ מציינים את הרשומות והחצים את הקשרים בין הרשומות.





על פי מודל זה כאשר נוצר מצב שיש לנו מרצה שהוא גם סטודנט אזי נצטרך כפילות (בזבוז).

כללים:

a. לכל (רשומה) בן יש אבא אחד

b. לכל אבא יש מספר כל שהוא (0..n) של בנים (מבנה של עץ רגיל)
[שנקר ← מתרגלים, מרצים, סטודנטים]

c. קשר אחד לרבים 1:N בין האב לבן.

d. אסור שיהיו קשרים בין אחים.

e. השאילתא היא בעצם מסלול עד לרשומה מסוימת בעץ.

בעיות במודל זה:

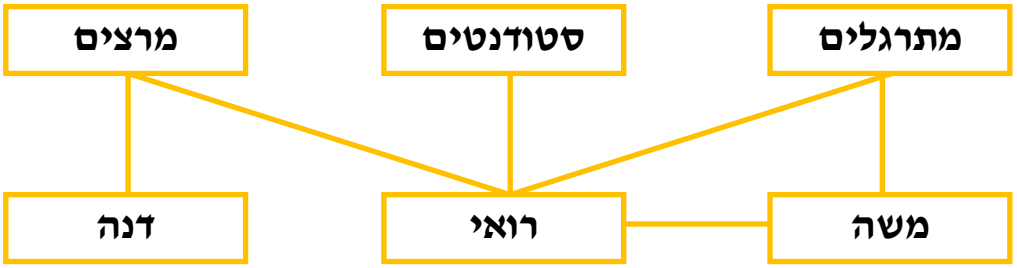
כפילויות ! כמו כן מודל זה לא יודע לתאר קשרים מסוג רבים לרבים, ואי אפשר להוסיף רשומה לטבלת בן עד שזו לא הוכלה בטבלת האב –יש בעיה להוסיף סטודנט שעדיין לא נרשם לאף קורס.

קיימים 3 סוגי קשרים בין ישויות:

- יחיד ליחיד –יחס חד-חד ערכי (חח"ע) בין 2 ישויות (דוגמא: יחס בין ת.ז. לאזרח במדינת ישראל)
- יחיד לרבים – ישות אחת מחוברת לאוסף קשרים בישות שנייה (דוגמא: יחס בין כיתה לסטודנטים)
- רבים לרבים – יחס של אוסף קשרים דו צדדי (דוגמא: יחס בין כיתות לקורסים)

2) המודל הרשתי:

רשת של קודקודים וחצים ללא הגבלה, ז"א קשרים רב-רב-ערכיים M:N.
המודל הרשתי דומה מאד למודל ההיררכי (תת קבוצה שלו) כאשר במקום שימוש בעץ שבו לכל קודקוד רק הורה אחד, המודל הרשתי מציע קבוצה המאוגדת במבנה דמוי עץ, עם שינוי אחד: לבן מותרים כמה אבות ← זאת אומרת העץ מוחלף בגרף.
שינוי זה מאפשר למודל הרשתי לתמוך בקשרים מסוג רבים-לרבים.



בעיות במודל זה:

לאור העובדה שניתן לבצע קישור בין כולם לכולם, נוצר מצב של **מורכבות גבוהה ואי גמישות לשינויים** (מחיקת רשומה אחת תגרור המון פעולות), ולאור כך התחזוקה הינה קשה ומסובכת.

3) המודל היחסי / טבלאי (רלציוני מלשון relation):

בסיס הנתונים בנוי מטבלאות, כאשר הקשר בין הטבלאות יתבצע ע"י עמודה/ות משותפות.

Bank Clients

id	name	account
22455	Moshe Levi	233
23345	Yair Cohen	322
12234	Yoav Gal	227
11113	Michal Tam	998
23349	Yonit Ron	239

Bank Accounts

account	credit	type
233	2500	regular
322	300	regular
227	1000	regular
998	4000	silver
239	9560	gold

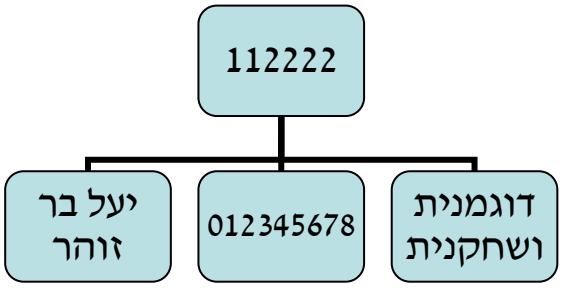
עקרונות:

- אפשר לתאר קשרים רב ערכיים.
- מספר הרשומות הינו בלתי מוגבל.

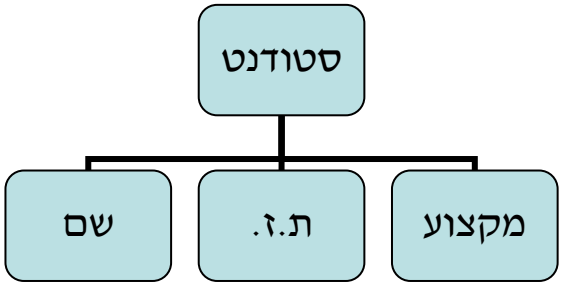
הגדרות:

- טבלה = יחס
- עמודה = תכונה
- **סכמה**: מבנה בסיס הנתונים, סדרה של תכונות המתארות את הפורמט / המבנה הלוגי של הנתונים.

דוגמא:



מופע – תאור מבנה הנתונים ברגע מסויים



מבנה נתונים / סכמה מסויימת

- **מפתח**: שדה או אוסף שדות המייצגים באופן חח"ע ומינימלי רשומה מסויימת (כמו ת.ז.).

הצגת רשומה בבסיס הנתונים וגישה לשדה בתוך הרשומה:

t1 = {Roei,Zerahia,012345678,01053081,1000)

t1[first_name] = 'Roei'

t1[Last_name] = 'Zerahia'

יתרונות המודל היחסי:

כל טבלה מורכבת משורות אופקיות ועמודות אנכיות.

כל טבלה מזוהה חד משמעית ע"י שם ייחודי, ושם זה משמש את בסיס הנתונים כדי לאתר את הטבלה. המשתמש על מנת להשתמש בטבלה צריך לדעת את שמה בלבד. השאלה כיצד המידע מאוחסן בפועל ברמה הפיסית לא מטרידה אותו, בזה מטפל ה DBMS.

חשוב להבין, זהו הבדל משמעותי לעומת המודל ההיררכי והרשתי, בהם המשתמש היה צריך לדעת כיצד הנתונים מסודרים בבסיס הנתונים, על מנת לעקוב אחרי המבנה ולהגיע לנתון אותו רוצים להוסיף, למחוק או לעדכן.

גישה לנתונים בשיטה זו הופכת את המודל היחסי להרבה יותר פשוט ויעיל מהקודמים לו, בגלל שהוא הרבה יותר קל להבנה.

יתרון נוסף של המודל היחסי בכך שהוא מספק כלים יעילים מאד לניהול בסיס הנתונים. זאת משום שטבלאות יכולות להכיל לא רק את הנתונים המאוחסנים בפועל, אלא גם מידע לניהול בסיס הנתונים:

- מידע על הטבלאות עצמן
- שמות השדות שבבסיס הנתונים
- הרשאות גישה לטבלאות
- כללי הטיפול בנתונים ועוד

בעצם כל רכיב במודל היחסי יכול להיות מאוחסן בטבלאות.

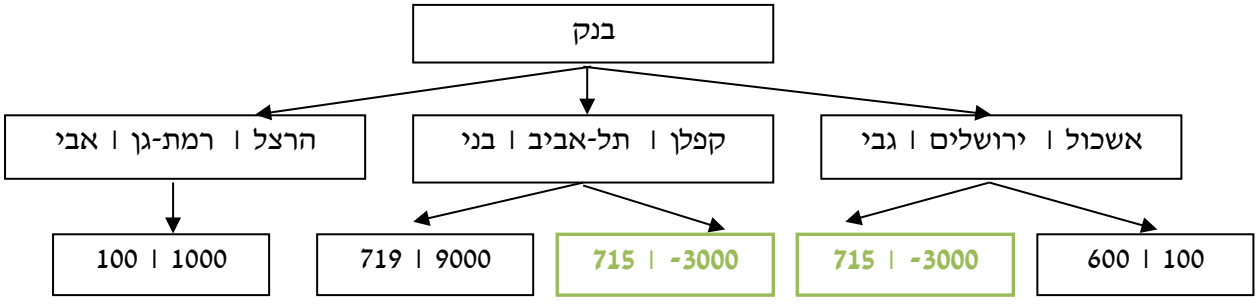
דוגמא מסכמת:

בנק שבו יש חשבונות של לקוחות כאשר עבור חשבון ידועים הנתונים הבאים:

- מספר החשבון
- היתרה בכל חשבון
- שם לקוח
- כתובת הלקוח (עיר ורחוב).

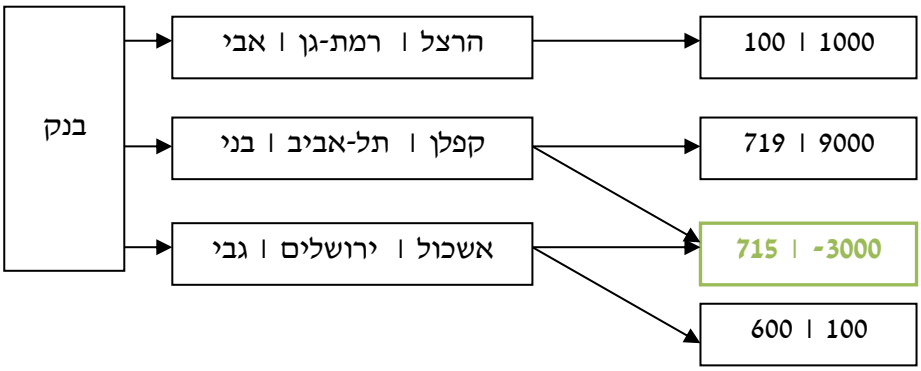
קיים יחס 1:n בין כל שתי רמות בעץ.
היה ניתן להחליף בין רמה 2 לרמה 3
ואז במקום הופעת החשבון פעמיים
הייתה לנו כפילות של פרטי הלקוח –
כך שבכל מצב יש כפילות של ערכים
במודל זה.

הדוגמא במודל ההיררכי:



נוכל לראות שיש 2 אנשים בעלי אותו חשבון (שותפים) וחשבון זה מוצג בעץ פעמיים (כפילות) ולכן כאשר נבוא להפקיד סכום של 1000 ₪ לחשבון המשותף – הפקיד בבנק יצטרך לעדכן את 2 התאים (המסומנים בירוק) להיות ביתרה של 2000- (בוצע עדכון כפול בגלל הכפילויות במודל).

הדוגמא במודל הרשתי:



נוכל לראות שהחשבון של השותפים מוצג בגרף רק פעם אחת (אין כפילות) ולכן כשנבוא להפקיד סכום של 1000 ₪ לחשבון המשותף – הפקיד יעדכן אותו רק פעם אחת. הבעיה שקיימת כאן היא מורכבות וסרבול וזו מתבטאת כאשר נרצה לבוא ולמחוק את הפרטים של בני (בהנחה והוא עזב את הארץ).

הדוגמא במודל הטבלאי:

DBMS

טבלת חשבונות	
מס' חשבון	יתרה
100	1000
719	9000
715	-3000
600	100

טבלת לקוחות			
מס' חשבון	רחוב	עיר	שם
100	הרצל	רמת גן	אבי
719	קפלן	תל אביב	בני
715	קפלן	תל אביב	בני
715	אשכול	ירושלים	גבי
600	אשכול	ירושלים	גבי