בסיסי נתונים - קורס מתקדם

2024 – תשפ"ד (סמסטר קיץ)

מרצה: רואי זרחיה

קורס מתקדם

<u>מטרת הקורס:</u>

הכרת נושאים מתקדמים במערכות בסיסי נתונים, על ידי שילוב של נושאים תיאורטיים והתנסות מעשית.

<u>תאור תמציתי</u>

בקורס ילמדו נושאים מתקדמים מעולם בסיסי הנתונים וביניהם שאילתות מורכבות, פונקציות ופרוצדורות, תזמונים, טרנזקציות, נעילות וכד'.

קורס מתקדם

שיטת הוראה

הרצאה בשילוב מצגות, ותרגול פרקטי.

תרומת הקורס

הקורס יקנה ידע מקצועי נוסף החיוני למהנדסים בתפקידיהם בתעשייה ויאפשר העמקה בעולמות בסיסי הנתונים ומערכות המידע עם הקניית מונחים מקצועיים ודרכי חשיבה לפתרון בעיות.

נושאי הקורס

1) **חזרה מהירה** על כל מונחי בסיס כתזכורת וכמבוא לנושאים המתקדמים הבאים, תוך כדי העמקה נוספת בנושאים הבסיסיים שנלמדו.

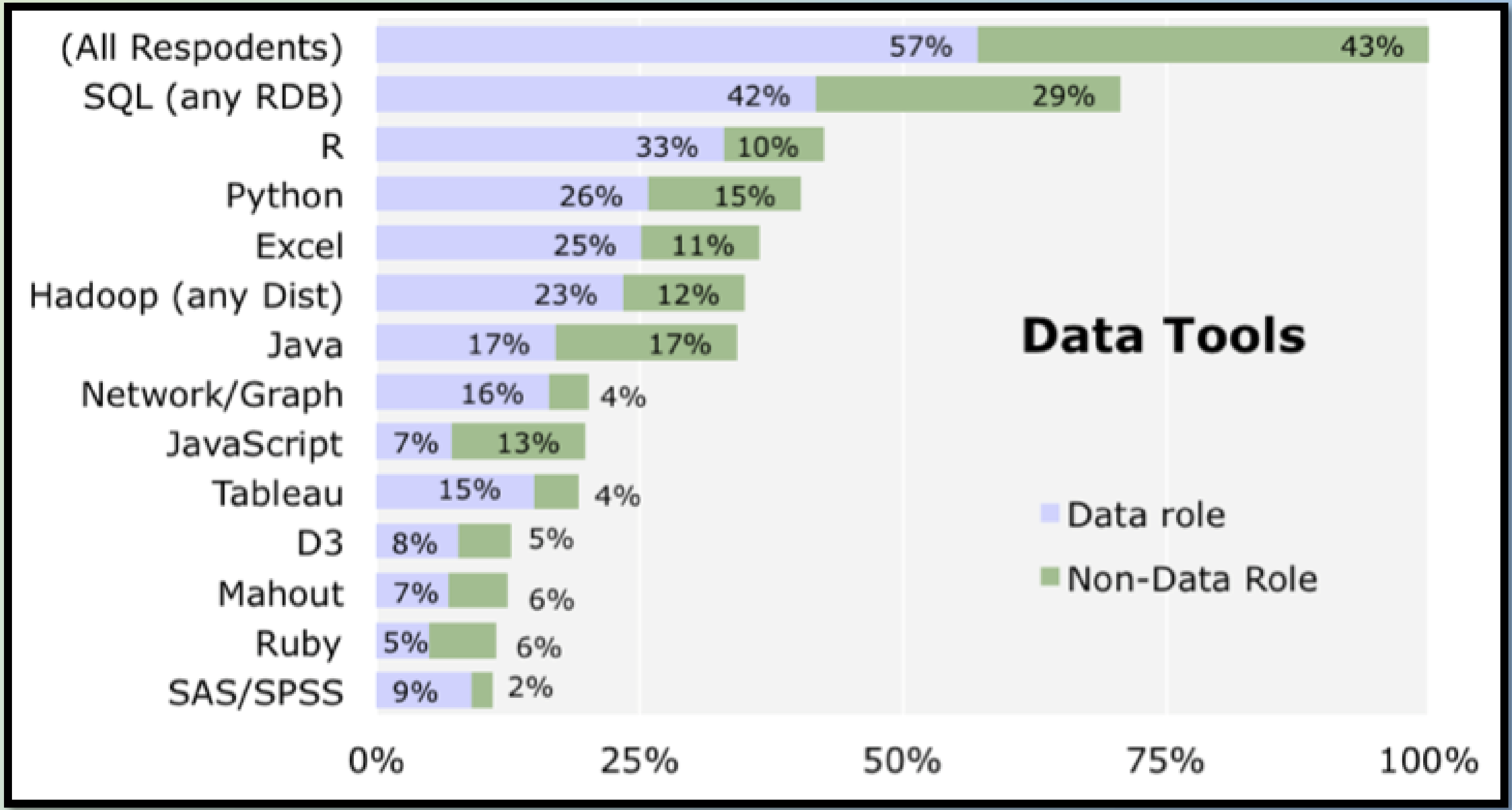
2) <u>**נושאים מתקדמים בקורס**</u>
- סקירה של מערכות בסיסי נתונים וסוגי המשתמשים השונים
- שאילתות מורכבות ויעילות הרצה (כולל השוואת זמני ריצה)
- פקודות ואופרטורים נוספים
- שילוב של פונקציות ופרוצדורות
- תזמונים ונעילות
- ניהול טרנזקציות ובקרת בו-זמניות

שיעור 1

מערכות בסיסי נתונים

מערכות בסיסי נתונים

כיום, כאשר חברה כלשהיא, בתחום כלשהוא מעוניינת לשמור, לאחסן, לנהל ולנתח נתונים, היא צריכה להחליט באיזה פלטפורמה להשתמש ונראה שלמרות שיש לא מעט חלופות בשוק, עדיין אחוז נכבד מאוד ישתמש בשפת SQL בכדי לבצע זאת :



מערכות בסיסי נתונים

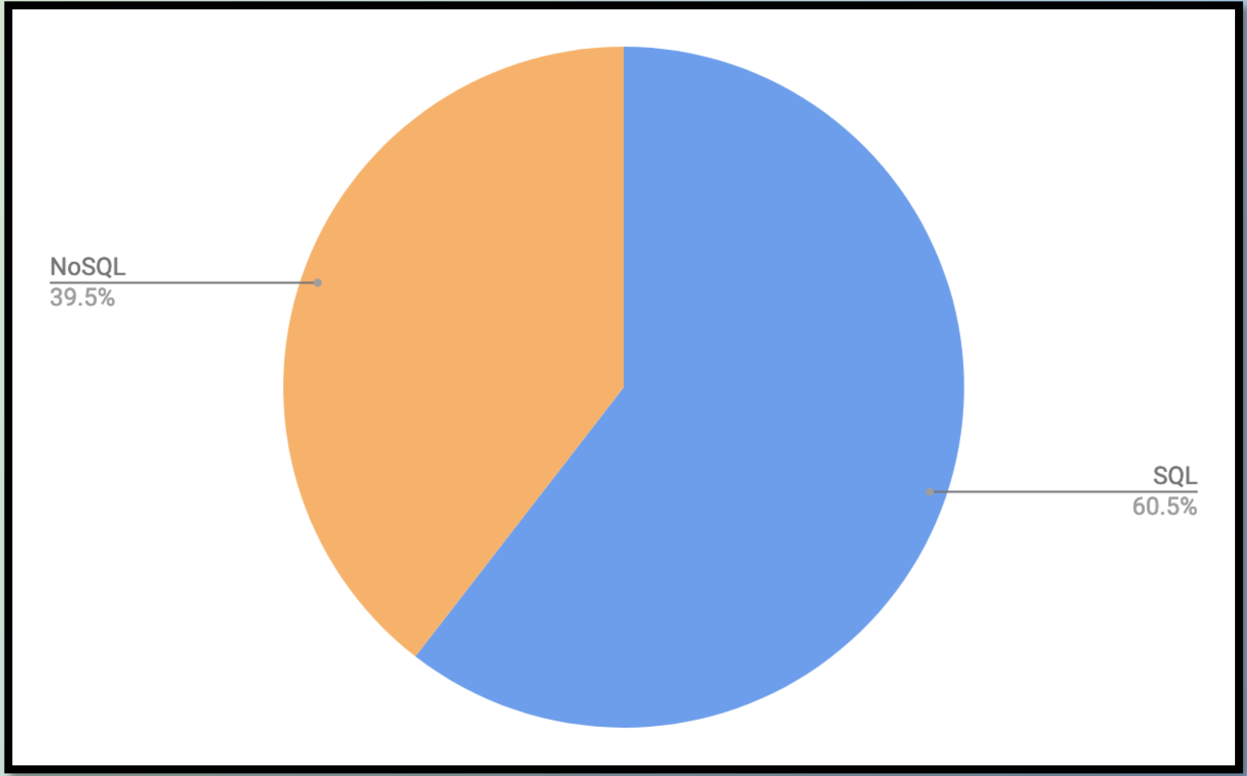
קיימות בשוק מספר מערכות לניהול בסיסי נתונים המשתמשות בשפת SQL:

SQL NO-SQL SQL NO-SQL

https://hackernoon.com/2019-database-trends-sql-vs-nosql-top-databases-single-vs-multiple-database-use-ly1cu3z18

מערכות בסיסי נתונים

הצורך בנתונים לא הולך להעלם בקרוב ובתחום זה עדיין לשפת SQL יש יתרון מובהק בשוק, עם מעל ל-60% פרישה בפתרונות לעבודה עם נתונים

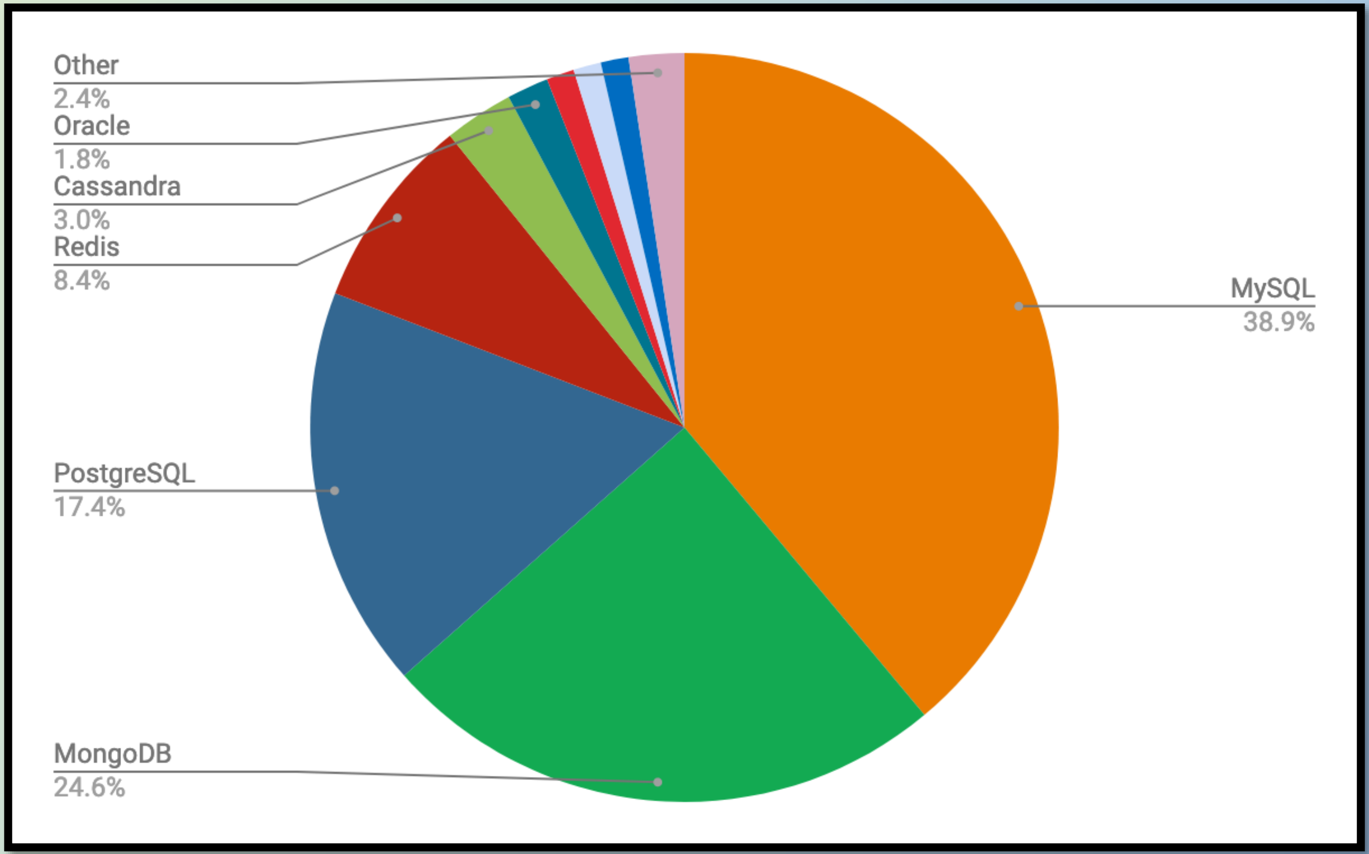


https://hackernoon.com/2019-database-trends-sql-vs-nosql-top-databases-single-vs-multiple-database-use-ly1cu3z18

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

מערכות בסיסי נתונים

תוכנת בסיסי הנתונים הפופולרית ביותר בשוק הינה עדיין MySQL



https://hackernoon.com/2019-database-trends-sql-vs-nosql-top-databases-single-vs-multiple-database-use-ly1cu3z18

בסיסי נתונים – רקע ומושגים בסיסיים

המטרות בטיפול בנתונים ע"י מערכת ייעודיות

- כל המידע מאוחסן ומנוהל במקום אחד (מניעת סרבול בעדכון)
- נגישות (Accessibility) מהירה לכל הנתונים (לבעלי הרשאות)
- זמינות הנתונים (Data Availability) ע"י כלים מתוחכמים לשליפה ועדכון
- מניעת כפילויות (Duplications) וחסכון במקום
- ניצול הקשרים בין הנתונים
- אמינות ושלמות הנתונים (Data Integrity)
- בטחון הגישה לנתונים (Data Security)
- אפשרות לעבודה במקביל (Parallel)
- אפשרות לעבודה באופן מבוזר (Distributed)
- קלות בתחזוקה השוטפת
- שיתוף נתונים (Data Sharing) בין יישומים שונים / יחידות ארגון שונות

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

משתמשי המערכת

- **<u>מתכנתים ואנשי IT</u>**
  - ממשקים לתוכנה חיצונית לשימוש במידע (הכתובה בשפת תכנות)
    - שפות: VB, Java, C++, C#, PHP , Python ועוד...
    - ממשקים (API: Application Program Interface)
- **<u>מנהל בסיס הנתונים (Database Administrator)</u>**
  - מערכת לניהול ומעקב המכילה נתונים ובנוסף מידע על המשתמשים (מי עשה מה ומתי)
- **<u>משתמשי קצה (End Users)</u>**
  - גישה למידע ולדו"חות לפי דרישה

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

רמות הפשטה של נתונים

ה-DBMS משמש כ-"**מתווך**" בין **המשתמש** במידע לבין **הנתונים בדיסק**.

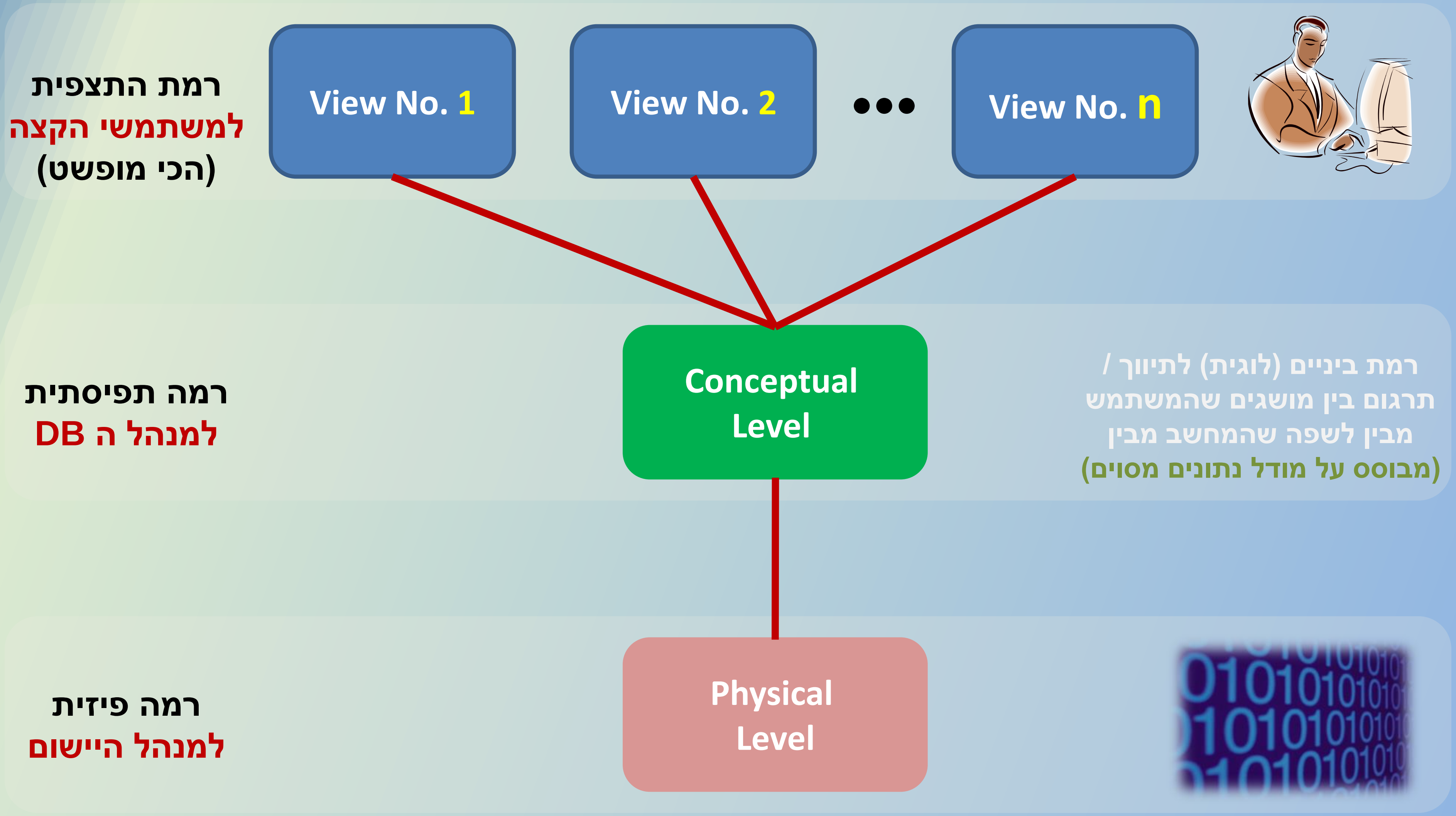
אחת המטרות העיקריות של ה DBMS היא לספק למשתמש תמונה מופשטת של הנתונים (מבט על) ולחסוך את הצורך לדעת פרטים שאינם נחוצים.

כדי להציג את ה DB בדרגת הפירוט הרצויה, מגדירים 3 רמות הפשטה:

- **רמה פיזית - Physical Level:** מתארת **איך הנתונים מאוחסנים** בפועל.
- **רמה תפיסתית - Conceptual Level:** **אילו נתונים מאוחסנים** ב DB ומהם הקשרים ביניהם - תיאור של אוסף מבנים פשוטים (לא צריך לעניין את המשתמש איך ממומש בפועל ברמה הפיזית)
- **רמת התצפית View Level:** תאור חלקי של בסיס הנתונים השלם (לא כל משתמש צריך לראות את כל המידע) המאפשר התאמת תצפית לצרכי המשתמש.

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

הקשרים בין שלושת רמות הפשטה



בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

מושגי בסיס: תבנית

תבנית (Scheme) הינה תיאור מבנה בבסיס הנתונים (לרוב נשאר קבוע).

✓**<u>תבנית פיסית</u>** (physical scheme) לתיאור ברמת המימוש:

```
Struct   BankAccount
{
        Int account_no;
        Int costumer_id;
        Int branch_no;
        Date birth_date;
        Char first_name[10];
        Char last_name[20];
        Char Address[30];
        Struct BankAccount *next;
}
```

✓**<u>תבנית תפיסתית</u>** (conceptual scheme) לתיאור ברמת על:

Account_no	Customer_id	Branch_no	Birth_date	First_name	Last_name	address

✓**<u>תת תבניות</u>** (sub-scheme) לתיאור תצפית ספציפית:

Account_no	Customer_id	Branch_no

Account_no	Customer_id	First_name	Last_name

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

מושגי בסיס: מופע

המידע המאוחסן ב DB משתנה לאורך הזמן, בעקבות הוספת/מחיקת נתונים (תחשבו לדוגמא על חשבון ה GMAIL שלכם במהלך יום נתון).

אוסף המידע המאוחסן ב DB ברגע מסוים נקרא **מופע רגעי (instance)** של בסיס הנתונים והוא מכיל נתונים לנקודת זמן ספציפית.

t=0

ID	NAME
2	Sigal

t=1

ID	NAME
1	Roei
2	Sigal

t=2

ID	NAME
1	Roei

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

דוגמא לרמות הפשטה

User A

User B

רמת התצפית
למשתמשי הקצה
(הכי מופשט)

ID F-NAME L-NAME

ID B-DATE SALARY

רמה תפיסתית
למנהל ה DB

ID F-NAME L-NAME B-DATE SALARY

רמת ביניים (לוגית) לתיווך /
תרגום בין מושגים שהמשתמש
מבין לשפה שהמחשב מבין
(מבוסס על מודל נתונים מסוים)

רמה פיזית
למנהל היישום

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

מה שמעניין אותנו... הרמה התפיסתית

רמת התצפית
למשתמשי הקצה
(הכי מופשט)

User A

ID F-NAME L-NAME

User B

ID B-DATE SALARY

רמה תפיסתית
למנהל ה DB

ID F-NAME L-NAME B-DATE SALARY

רמת ביניים (לוגית) לתיווך /
תרגום בין מושגים שהמשתמש
מבין לשפה שהמחשב מבין
(מבוסס על מודל נתונים מסוים)

רמה פיזית
למנהל היישום

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

מודלים לתיאור הרמה התפיסתית

מודל נתונים הוא כלי לתיאור הנתונים, הקשרים ביניהם, משמעות הנתונים וההגבלות החלות עליהם (תיאור נתונים ברמה תפיסתית וברמת תצפית (המודל מתווך בין המשתמש לרמה הפיזית תוך שמירה על אי תלות בנתונים).

מהם המודלים התפיסתיים (לוגיים) הקיימים ?

1) מודלים לוגיים המבוססים על **אובייקטים** - הצגת המציאות כאוסף של אובייקטים בסיסיים. המודל הנפוץ ביותר: **מודל ישויות קשרים**

2) מודלים לוגיים המבוססים על **רשומות** - משמשים להגדרת המבנה הלוגי של בסיס הנתונים:

1) **מודל היררכי/מדרגי** – עץ של רשומות נתונים וקווים מקשרים
2) **מודל רשתי** – גרף של נתונים המיוצגים ע"י רשומות והקשרים ע"י קווים מקשרים
3) **מודל היחסים/טבלאי** – נתונים ויחסים מוצגים ע"י אוסף של טבלאות

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

אי-תלות בנתונים (Data Independence)

הגדרנו שלוש רמות הפשטה שבאמצעותן ניתן לתאר את ה DB.

היכולת לבצע שינויים בהגדרה של תבנית ברמה מסוימת מבלי לפגוע בתבנית ברמה גבוהה יותר נקראת "**אי תלות בנתונים**". קיימים שני סוגים לאי תלות זו:

- **אי-תלות פיזית בנתונים** – אי תלות של התבנית התפיסתית בשינוים המבוצעים בתבנית הפיזית.
  דוגמא: נוכל לשנות את המבנה הפיזי (לדוגמא: לשיפור ביצועים) ללא צורך בשינוי האפליקציה.

- **אי-תלות לוגית בנתונים** – אי תלות של תבניות התצפית בשינוים המבוצעים בתבנית התפיסתית.
  דוגמא: נוכל לשנות את המבנה הלוגי (התבנית התפיסתית) של בסיס הנתונים (דוגמא: הוספת סוג חשבון חדש) ללא צורך בשינוי האפליקציה.

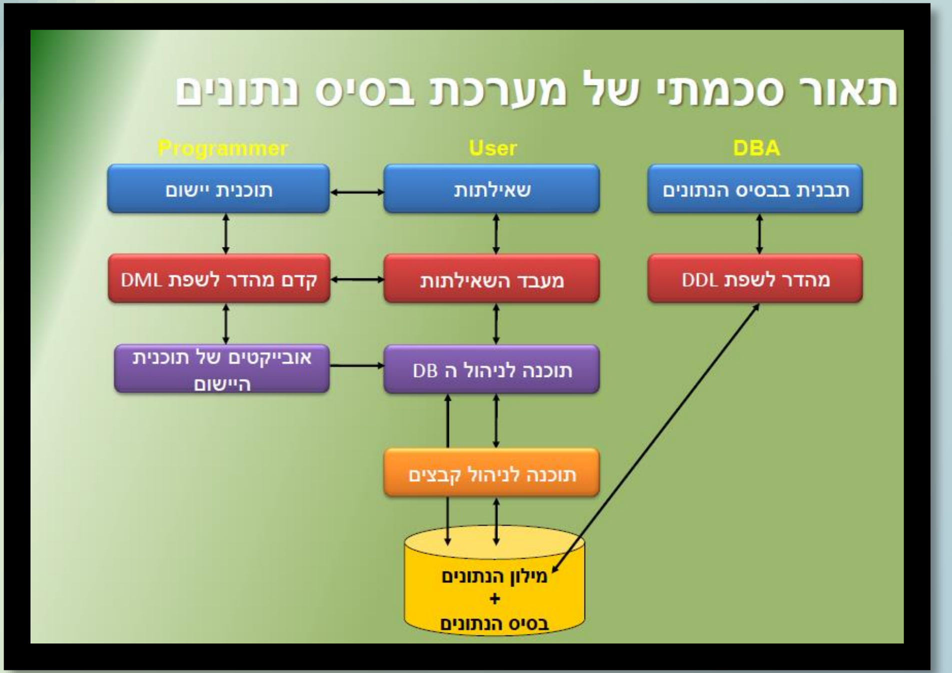
בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

מרכיבי מערכת בסיסי נתונים

- **תוכנה לניהול קבצים File Manager**: אחראית לאחסון (הקצאת זיכרון ומקום בדיסק) ולניהול מבני הנתונים המשמשים לייצוג המידע (מצויה לרוב במ"ה)
- **מנהל בסיס הנתונים Database Manager**: ניהול הממשק בין המידע בבסיס הנתונים לבין האפליקציה והשאילתות למערכת ("המתווך")
- **מעבד שאילתות Query Processor**: תרגום שאילתות לרמת מנהל בסיס הנתונים (ניתן לבצע אופטימיזציה של שאילתות לשיפור זמני השליפה
- **קדם מהדר DML (DML Pre-Compiler)**: המרת הוראות DML הקיימות בשפה המארחת לפרוצדורות בתוכנית היישום (בתאום עם מעבד השאילתות)
- **מהדר DDL (DDL Compiler)**: המרת הוראות DDL לקבוצת טבלאות (metadata) המאוחסנות במילון הנתונים

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

גישה לנתונים במערכת בסיסי נתונים



תוכניות היישום המבקשות לגשת אל הנתונים חייבות להשתמש בשפה לטיפול בנתונים:

SQL - שפה לאחזור ושינוי תוכן הנתונים ע"י שימוש בשאילתות לאחזור רשומות מטבלאות, עדכון ומחיקת רשומות קיימות והוספת רשומות חדשות.

Embedded SQL - שיבוץ פקודות SQL סטטיות או דינמיות (שאילתות) בתוך שפת תכנות מארחת הדורשת שירותי קדם-מהדר.

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

גישה לנתונים מתוכנית יישום

הקדם מהדר יודע לתרגם אוסף פקודות בשפה המארחת לפקודות SQL.

<u>לדוגמא</u>: Oracle תומך בעבודה עם קדם מהדר בשפות תכנות מסוימות ומאפשר לשבץ פקודות SQL בתוך שפת התכנות המארחת:

```
    EXEC SQL BEGIN DECLARE SECTION;
    varchar  userid (20);
    varchar  password (20);
    varchar  student_number (5);
    varchar  student_name (20);
    EXEC SQL END DECLARE SECTION;

main()
{
    printf ("\n Enter Student Number: ");
    scanf ("%s", student_number);

    EXEC SQL SELECT NAME INTO :student_name
             FROM    STUDENTS
             WHERE   STUDENT_ID = :student_number;
}
```


בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

גישה לנתונים משרת Web

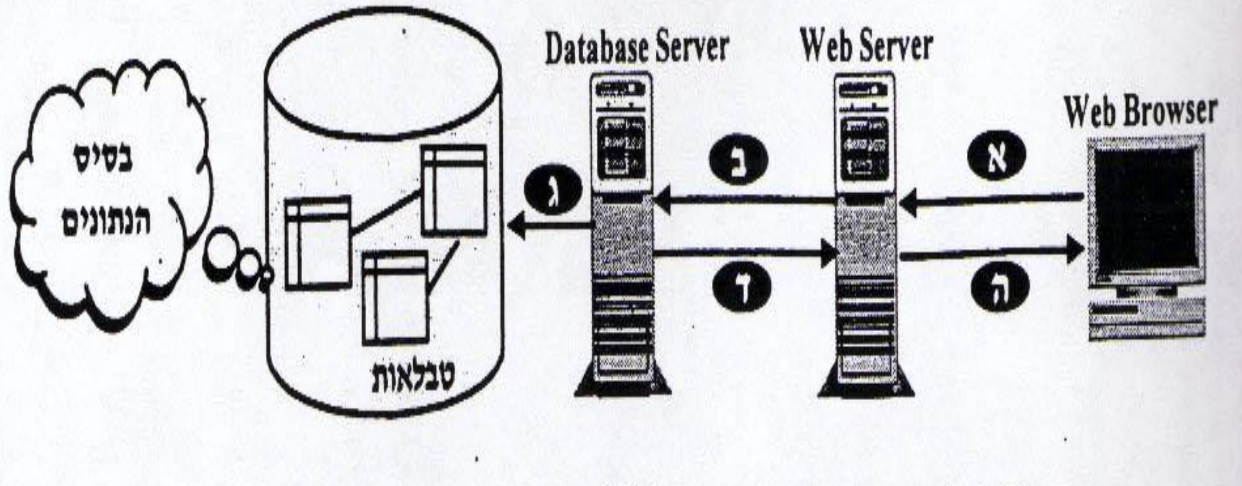
הנתונים מגיעים מגולש שהכניס מידע בדף אינטרנט כלשהו. המידע מועבר לשרת האינטרנט שמתקשר עם בסיס הנתונים על פי השלבים הבאים:

- דפדפן (Browser) מציג טופס כדף במבנה HTML
- הזנת נתוני קלט ע"י הגולש
- קלט מגיע לשרת WEB באמצעות פרוטוקול HTTP
- הפעלת יישום הבונה שאילתת SQL
- היישום נשלח לשרת בסיס הנתונים
- ביצוע פקודות (שאילתות) בשפת SQL
- החזרת תוצאות השאילתא (טבלה) לשרת WEB
- בניית דף HTML עם הפלט להצגה לגולש
- הצגת הדף ברשת האינטרנט בפרוטוקול HTTP על גבי הדפדפן

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

גישה לנתונים משרת Web

הנתונים מגיעים מגולש שהכניס מידע בדף אינטרנט כלשהו. המידע מועבר לשרת האינטרנט שמתקשר עם בסיס הנתונים:



גישה לבסיס נתונים משרת Web.

יתרונות טכנולוגיית בסיסי הנתונים

- **גמישות לשינויים** (Flexibility to Changes) - קלות בהתאמת בסיס הנתונים ותוכניות היישום לשינויים בארגון.
- **אבטחת נתונים** (Data Security) - הגדרה ברורה למי מותר לעשות מה ועל איזה נתונים.
- **תמיכה בעיבוד תנועות** (Transaction Processing) - כשל במהלך ביצוע תנועה או אי השלמתה גורם לשחזור המצב הקודם.
- **תמיכה בעדכון בו-זמני** (Concurrent Update) - אפשרות לבצע עדכון בו-זמני של נתונים ע"י מספר משתמשים (שירותי נעילה ושחרור).

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

חסרונות טכנולוגיית בסיסי הנתונים

- **מורכבות** - עבור דרישות גבוהות ופעולות מורכבות, התוכנה הופכת להיות מורכבת יותר ודורשת מיומנות גבוה בהפעלתה ואחזקתה.
- **עלות** - מאות דולרים לתחנת עבודה פרטית, עשרות אלפי דולרים לשרתים מחלקתיים ועד מאות אלפי דולרים למערכות מחשב גדולות.
- **משאבי חומרה** - יותר זיכרון, CPU, ושטחי עבודה בדיסק הקשיח.
- **רגישות לתקלות** - תקלה יכולה לגרום להשבתה של חלק גדול מפעילות הארגון.
- **מורכבות בשחזור** - בגלל הקשרים בין הנתונים, תיקון ברשומה אחת גורר תיקונים רבים נוספים בטבלאות הקשורות ועלול לקחת זמן ומשאבים רבים.

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

תכנון והקמת בסיס נתונים חדש

תהליך הקמת בסיס נתונים חדש

על מנת לנתח ולעצב מודל נתונים חדש, יש לדאוג לשלבים הבאים:

1. הבנת הדרישות (Requirements) – הצגת סיפור המעשה (בלשון פשוטה / מובנת ללקוחות)
2. יצירת המודל התפיסתי של מנתח המערכת המובן לבני אדם (ושניתן להציג אותו ללקוחות) ללא תלות בסכמה הלוגית (מערכת מסוימת) ובסכמה הפיזית (מבנה הטבלה) ע"י תכנון הסכמה הלוגית (עיצוב)
3. תרגום המודל הלוגי לתכנון הסכמה הפיזית (הגדרת העמודות וההגבלות של הטבלה)

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

הקמת בסיס נתונים חדש - דוגמא

(1) הבנת הדרישות (Requirements) – הצגת סיפור המעשה (בלשון פשוטה / מובנת ללקוחות):

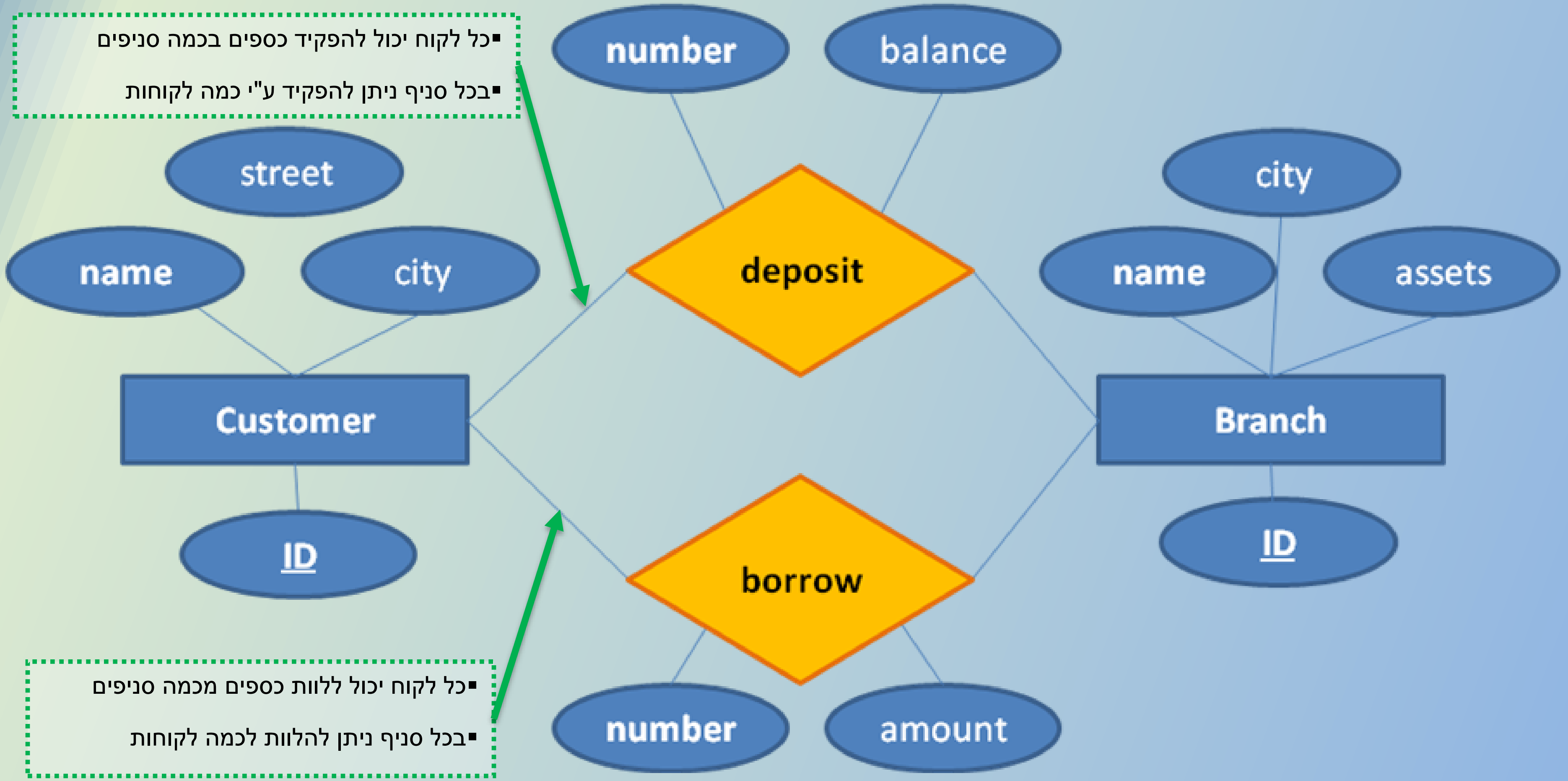
אחד הבנקים בישראל מעוניין להעביר את **הטיפול במערכת הבנקאית** לתוך מערכת מידע חדשה שתכיל בסיס נתונים שיטפל בכל הנתונים של הפקדות והלוואות מהסניפים השונים.

ידוע שמבחינת הפקדות: כל לקוח יכול להפקיד כספים בכמה סניפים ובכל סניף ניתן לבצע הפקדות ע"י כמה לקוחות, ומבחינת הלוואות: כל לקוח יכול ללוות כספים מכמה סניפים ובכל סניף ניתן להלוות לכמה לקוחות.

(2) יצירת המודל התפיסתי של מנתח המערכת המובן לבני אדם (ושניתן להציג אותו ללקוחות) ללא תלות בסכמה הלוגית (מערכת מסוימת) ובסכמה הפיזית (מבנה הטבלה) ע"י תכנון הסכמה הלוגית (עיצוב):

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

דוגמא לדיאגרמת ER – מערכת בנקאית



תרגום דיאגראת ER – המרה לטבלאות

(3) תרגום המודל הלוגי לתכנון הסכמה הפיזית (הגדרת העמודות וההגבלות של הטבלה):

תרשים ה ER מציג 2 **טיפוסי ישויות** שיומרו לשתי טבלאות בבסיסי הנתונים ודרגתם תקבע לפי מספר התכונות:

- **Branch-scheme** (Branch-ID, branch-name, assets, branch-city)
- **Customer-scheme** (customer-ID, customer-name, street, customer-city)

תרשים ה ER מציג גם שני יחסים עם מידת ריבוי **רבים-לרבים** שיומרו גם כן לשתי טבלאות בבסיס הנתונים:

- **Deposit-scheme** (Costumer-ID, Branch-ID, Deposit-number, balance)
- **Borrow-scheme** (Costumer-ID, Branch-ID, Borrow-number, amount)

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

תרגום דיאגראת ER – המרה לטבלאות

(3) תרגום המודל הלוגי לתכנון הסכמה הפיזית:

<u>חוקי התרגום מ ERD למודל הטבלאי:</u>

- לכל קבוצת יישות יוצרים טבלה (שם הטבלה זהה לשם קבוצת היישות)
- תכונות היישות הופכות להיות שדות (עמודות) בטבלה (שמות זהים)
- תכונות המפתח של קבוצות היישות יהפכו להיות שדות המפתח בטבלה
- אם ליישות תכונה מורכבת ("כתובת") אזי התכונות המרכיבות אותה ("רחוב", "עיר", "מספר") יהפכו להיות שדות בטבלה (התכונה המורכבת עצמה לא תיכלל בטבלה)
- קשרים 1:1 - הוספת מפתח זר
- קשרים 1:N - הוספת מפתח זר
- קשרים N:M – קשר זה ממופה לטבלה חדשה שבה המפתח מורכב מאיחוד המפתחות של טבלאות הבסיס (ולטבלה זו יתווספו התכונות השייכות לקשר עצמו, אם כאלה קיימות)

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

המערכת הבנקאית [טבלאות במסד הנתונים]

Bank Database

Branch			
<u>ID</u>	Name	Assets	City

Customer			
<u>ID</u>	Name	street	City

Deposit			
<u>Customer-ID</u>	<u>Branch-ID</u>	<u>Deposit-Number</u>	balance

Borrow			
<u>Customer-ID</u>	<u>Branch-ID</u>	<u>Borrow-Number</u>	balance

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

שפות שאילתות

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

שפות שאילתות מסחריות

השפות הפורמאליות (אלגברת יחסים ותחשיבי היחסים) מהוות כלי ביטוי מתומצת וקצר לייצוג שאילתות אך למערכות בסיסי נתונים מסחריות דרושה שפת שאילתות ידידותית יותר.

אלגברת יחסים היא שפה פורמלית המשמשת לצורך כתיבת שאילתות על מסדי נתונים טבלאיים ומבוססת על לוגיקה מסדר ראשון ועל תורת הקבוצות. אלגברת היחסים דומה למתמטיקה העוסקת בערכים מספריים, ומאפשרת לבצע עליהם פעולות חשבון, רק שהערכים בהם עוסקת אלגברת היחסים הם טבלאות.

קיימות 3 שפות שאילתות מסחריות עיקריות:

1) QBE המבוססת על תחשיב היחסים לפי תחומים.
2) QUEL המבוססת על תחשיב היחסים לפי n-יות.
3) SQL המבוססת על **שילוב של אלגברת יחסים ותחשיבי היחסים**.

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

SQL

השפות המסחריות הקיימות כיום מציעות אפשרויות רבות מעבר לביטוי שאילתות וכוללות גם:

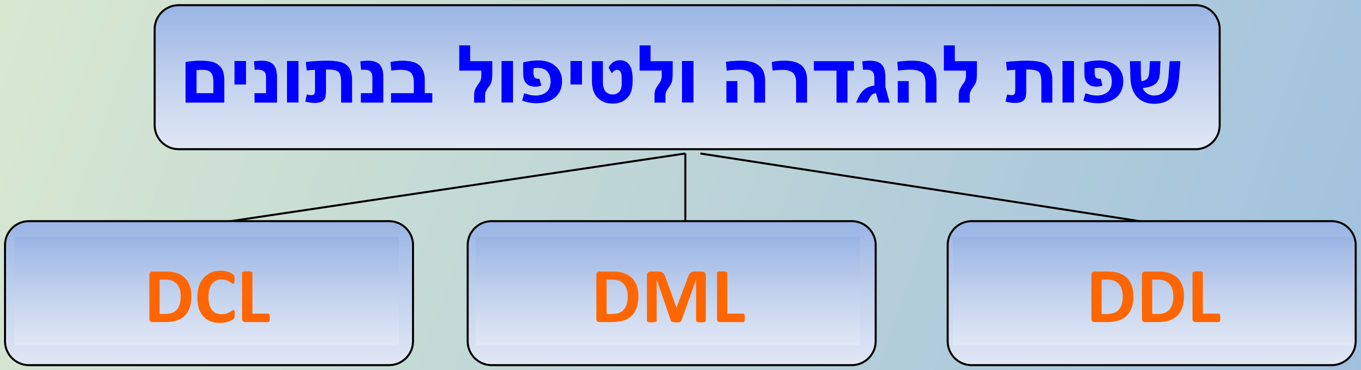
- הגדרת מבני נתונים
- עדכון ומחיקת נתונים ומבנים
- הגדרת אילוצי בטיחות

SQL הינה שפה השאילתות הדומיננטית כיום לטיפול בבסיסי נתונים יחסיים ומהווה מימוש פרוצדוראלי המרחיב את המסגרת הפורמאלית של אלגברת היחסים, שפת SQL היא השפה להגדרה ולטיפול בנתונים המאפשרת לבצע את כל הפעולות על הטבלאות שהוגדרו לאורך כל התהליך.

השפה מאפשרת **לטפל בכל שרשרת הערך של הטבלה**, משלב בניית הטבלה, הכנסת ערכים לטבלה, עדכון הערכים ו/או מחיקתם וכמובן הנושא החשוב ביותר <u>שליפת הנתונים </u>מהטבלה בחתכים שונים.

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

SQL – תתי שפות להגדרה ולטיפול בנתונים



שפה להגדרת נתונים (Data Definition Language)

- שפה להגדרת סכמות (מבנים) ואילוצים בבסיס הנתונים
- המהדר יוצר טבלאות הנשמרות **<u>בקובץ מיוחד</u>** המכיל Metadata (נתונים אודות הנתונים) - קובץ זה נקרא מילון נתונים (data dictionary)

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

שפות להגדרה ולטיפול בנתונים (2)

שפה לטיפול בנתונים (Data Manipulation Language)

- שפה המאפשרת שליפת מידע (אחזור), הוספה, עדכון ומחיקת מידע
- ברמה הפיזית יוגדר האלגוריתם לגישה יעילה לנתונים
- ברמות ההפשטה הגבוהות ינתן דגש על פשטות השימוש ויעילות העבודה
- **קיימים שני סוגים של שפות DML:**
  - שפות פרוצדוראליות בהן יוגדר "איך" לקבל את הנתונים
  - שפות לא פרוצדוראליות שיגדירו "מה" רוצים לקבל
- שאילתא (Query) – הינה בקשה לשליפת מידע מבסיס הנתונים

שפה לניהול הרשאות במסד הנתונים (Data Control Language)

- אוסף פקודות המאפשרות לנהל הרשאות במסד הנתונים (פקודות יעודיות כגון: GRANT , DENY , REVOKE)

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

SQL ואלגברת יחסים

מבנה פקודת SQL:

SELECT $A_1,A_2,\ldots A_n$
FROM $r_1,r_2,\ldots r_m$
WHERE P

- רכיב **SELECT** בוחר תכונות (מקביל לפעולת ההטלה באלגברת היחסים)
- רכיב **FROM** מפרט את (המכפלה הקרטזית של) היחסים המשתתפים בשאילתא
- רכיב **WHERE** מכיל נוסחת סינון נתונים (מקביל להפעלת אופרטור הבחירה באלגברת היחסים)

$$\Pi_{A_1,A_2,\ldots A_n}(\sigma_P(r_1 \times r_2 \times \ldots \times r_m))$$

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

SQL – מבנה פקודת SELECT בסיסית

```
SELECT    ID, Name
FROM      Students
WHERE     Students.id = 3344
```

- **פקודת ה SELECT** - בחירת העמודות ת.ז. ושם מתוך הטבלה (מופרדים בפסיקים ביניהם).
- **פקודת ה FROM** - אוסף הטבלאות המשתתפות בשליפה הנוכחית, במקרה זה נשלוף נתונים מטבלה בודדת - טבלת סטודנטים.
- **פקודת ה WHERE** – שליפת הסטודנט בעל ת.ז. 3344 - ביצוע פעולה של בחירת (סינון) השורות המקיימות את התנאים המוגדרים בה (פקודה זו **הינה אופציונאלית** בביצוע השליפה. ללא פקודה זו תתבצע שליפה של כל השורות בטבלה).

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

טבלאות המערכת הבנקאית

Customer			
ID	customer-name	street	customer-city
1	Morag	Pinkas	Rishon
2	Tamir	Allenby	Haifa
3	Avivi	Pinkas	Rishon
4	Even	Allenby	Haifa

Branch			
ID	branch-name	assets	branch-city
10	Hamerkaz	9,000,000	Tiberias
20	Pinkas	2,100,000	Eilat
30	Aviv	1,700,000	Jaffa
40	Tsafon	400,000	Jaffa

Borrow			
Customer-ID	Branch-id	Borrow-number	balance
1	30	101	1000
1	40	102	2000
2	10	103	1500
3	20	104	500

Deposit			
Customer-ID	Branch-id	Deposit-number	balance
1	30	201	500
3	20	202	700
4	40	203	400
4	30	204	650

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

SQL – דוגמאות בסיסיות

דוגמא 1: מצאו את שמות כל הסניפים ביחס deposit

```
SELECT    BranchName
FROM      deposit
```

דוגמא 2: מצאו את פרטי כל הסניפים ביחס deposit

```
SELECT    *
FROM      deposit
```

דוגמא 3: מצאו את הרחוב ושם העיר של לקוח בשם Morag

```
SELECT    street, CustomerCity
FROM      Customer
WHERE     CustomerName = 'Morag'
```


בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

כפילויות ערכים בתוצאה

דוגמא 4: מצאו את <u>שמות הרחובות</u> של כל לקוחות הבנק

Customer		
customer-name	**street**	**customer-city**
Morag	Pinkas	Rishon
Tamir	Allenby	Haifa
Avivi	Pinkas	Rishon
Even	Allenby	Haifa

```
SELECT   street
FROM     Customer
```

→

New Table
street
Pinkas
Allenby
Pinkas
Allenby

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

כפילויות ערכים בתוצאה

בשאילתות SQL לא מתבצע ביטול אוטומטי של n-יות כפולות. על מנת להציג תוצאה ללא כפילויות נשתמש בפקודה **distinct**

דוגמא 5: מצאו את שמות הרחובות של כל לקוחות הבנק (ללא חזרות)

Customer		
customer-name	**street**	**customer-city**
Morag	Pinkas	Rishon
Tamir	Allenby	Haifa
Avivi	Pinkas	Rishon
Even	Allenby	Haifa

```
SELECT    Distinct street
FROM      Customer
```

New Table
street
Pinkas
Allenby

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

תנאי לסינון ח-יות מיחס

דוגמא 6: מצאו את <u>שמות</u> הלקוחות שיש להם חשבון חסכון בסניף Aviv

Deposit			
branch-name	account-number	customer-name	balance
Hamerkaz	101	Even	500
Tsafon	215	Tamir	700
Aviv	102	Avivi	400

```
SELECT   CustomerName
FROM     Deposit
WHERE    BranchName = "Aviv"
```

→

New Table
customer-name
Avivi

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

שילוב אופרטורים

בשאילתות SQL נשתמש ב and, or, not לייצג את הקשרים הלוגים v, ^, ¬

דוגמא 7: מצאו את שמות הסניפים בעיר יפו המנהלים נכסים בשווי של מעל 500,000 ₪

Branch		
branch-name	assets	branch-city
Hamerkaz	9,000,000	Tiberias
Pinkas	2,100,000	Eilat
Aviv	1,700,000	Jaffa
Tsafon	400,000	Jaffa

```
SELECT    BranchName
FROM      Branch
WHERE     (BranchCity = “Jaffa”) AND (assets > 500,000)
```

branch-name
Aviv

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

שילוב קבועים וחישובים בשליפה

ניתן להפעיל את האופרטורים +,-,*,/ על ערכי השליפה ו/או על קבועים

ניתן להוסיף קבועי טקסט כעמודה וירטואלית

דוגמא 8: הציגו את <u>סכומי הנכסים</u> המוחזקים ע"י הסניפים ביפו בתוספת מע"מ (+17%) ועם עמודה מקדימה עם הכיתוב "סכום כולל מעמ"

Branch		
branch-name	assets	branch-city
Hamerkaz	9,000,000	Tiberias
Pinkas	2,100,000	Eilat
Aviv	1,700,000	Jaffa
Tsafon	400,000	Jaffa

assets including vat	assets
assets including vat	1,989,000
assets including vat	468,000

```
SELECT    'assets including vat' , assets*(1.17)
FROM      Branch
WHERE     BranchCity = “Jaffa”
```


בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

הגדרת טווח סינון

BETWEEN - תת-פקודה של WHERE המסננת נתונים ע"י הגדרת טווח

דוגמא 9: הציגו את שמות הלקוחות שנטלו הלוואות שמספרן בטווח 14-18

Borrow			
branch-name	Borrow-number	customer-name	amount
Hamerkaz	**17**	**Morag**	**1000**
Pinkas	23	Tamir	2000
Aviv	**15**	**Avivi**	**1500**
Tsafon	93	Even	500

customer-name
Morag
Avivi

```
SELECT    CostumerName
FROM      Borrow
WHERE     BorrowNumber BETWEEN 14 AND 18
```


בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

סינון מחרוזות והתאמת תבניות

מתבצע בעזרת אופרטור ההשוואה **LIKE** ע"י שימוש בתווים מיוחדים:

'_' - מתאים לתו אחד בדיוק

'%' - מתאים לאפס תווים או יותר (גודל בלתי מוגבל)

'\\' – תו המאפשר להשתמש בתווים המיוחדים כתווים רגילים

על מנת לבצע סינון של ח-יות העונות לתנאי המכיל תבנית השוואה מסוג טקסט, נוסיף בפקודת ה WHERE את תנאי הסינון בשילוב אופרטור LIKE והתבנית המבוקשת

WHERE ______________ **LIKE** '____________'

שם עמודה   תבנית סינון

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

סינון מחרוזות והתאמת תבניות

דוגמא 10: הציגו את <u>שמות הסניפים</u> הממוקמים בערים שמתחילות באות T או מסתיימת באות a.

Branch		
branch-name	assets	branch-city
Hamerkaz	9,000,000	Tiberias
Pinkas	2,100,000	Eilat
Aviv	1,700,000	Jaffa
Tsafon	400,000	Jaffa

branch-city
Tiberias
Jaffa
Jaffa

```
SELECT    BranchName
FROM      Branch
WHERE     (BranchCity LIKE 'T%') OR (BranchCity LIKE '%a')
```

שינוי התנאי ל-"וגם":

```
WHERE     (BranchCity LIKE 'T%a')
```


בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

סינון מחרוזות והתאמת תבניות

Branch		
branch-name	assets	branch-city
Hamerkaz	9,000,000	Tiberias
Pinkas	2,100,000	Eilat
Aviv	1,700,000	Jaffa
Tsafon	400,000	Jaffa

1) הציגו את <u>שמות הסניפים</u> הבנויים מ-5 תווים בדיוק

```
SELECT    BranchName
FROM      Branch
WHERE     BranchName LIKE '_ _ _ _ _')
```

2) הציגו את <u>שמות הסניפים</u> המכילים '_' בשם העיר

```
SELECT    BranchName
FROM      Branch
WHERE     BranchCity LIKE '% \_ %')
```


בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

שליפה ממספר יחסים

בפקודת ה FROM ניתן לרשום מספר יחסים כשהם מופרדים בפסיק (הפסיק מסמל את פעולת המכפלה הקרטזית בין הטבלאות – מכפלה של M:N).

דוגמא 11: מצאו את שמות הלקוחות שלוו כסף ואת עיר מגוריהם

```
SELECT    customer.CustomerName, CustomerCity
FROM      borrow, customer
WHERE     borrow.CustomerName = customer.CustomerName
```

על מנת להקל על תהליך העבודה נוכל להשתמש בכינוי לשמות הטבלאות:

```
SELECT    C.CustomerName, CustomerCity
FROM      borrow B, customer C
WHERE     B.CustomerName = C.CustomerName
```


בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

שליפה ממספר יחסים

Borrow			
branch-name	**loan-number**	**customer-name**	**amount**
Hamerkaz	17	Morag	1000
Pinkas	23	Tamir	2000
Aviv	15	Avivi	1500
Tsafon	93	Even	500

Customer		
customer-name	**street**	**customer-city**
Morag	Pinkas	Rishon
Tamir	Allenby	Haifa
Avivi	Pinkas	Rishon
Even	Allenby	Haifa

Borrow x Costumer	
customer-name	**customer-city**
Morag	Rishon
Tamir	Haifa
Avivi	Rishon
Even	Haifa

$\Pi_{customer.customer\text{-}name,\ customer\text{-}city}$ (borrow X customer)

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

שליפה ממספר יחסים - JOIN

מטרת פקודת ה JOIN היא לבצע מכפלה בין שתי טבלאות (בעלות לפחות עמודה משותפת אחת) כשבטבלת התוצאה יתקבלו רק רשומות "משמעותיות".

דוגמא 12: מצאו את שמות הלקוחות שלוו כסף ואת עיר מגוריהם

```
SELECT   C.CustomerName, CustomerCity
FROM     borrow B , customer C
WHERE    B.CustomerName = C.CustomerName
```

מעבר ממכפלה קרטזית לשימוש בצירוף טבלאות JOIN

```
SELECT   C.CustomerName, CustomerCity
FROM     borrow B  JOIN  customer C
ON       B.CustomerName = C.CustomerName
```


בסיסי נתונים – קורס מתקדם – רואי זרחי ©

ON מול USING

פקודת ON מגדירה את תנאי החיבור בין הטבלאות.

```
SELECT      C.CustomerName, CustomerCity
FROM        borrow B   JOIN   customer C
ON          B.CustomerName = C.CustomerName
```

פקודת USING "חוסכת" את הגדרת התנאי במקרה של שמות עמודות זהות.

```
SELECT      C.CustomerName, CustomerCity
FROM        borrow B   JOIN   customer C
USING       (CustomerName)
```


בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

צירוף עם שמירת מידע

כפי שראינו באלגברת היחסים, קיימות שלוש פקודות להגדרת צירוף עם שמירת מידע:

1) צירוף עם שמירת מידע שמאלה LEFT OUTER JOIN
2) צירוף עם שמירת מידע ימינה RIGHT OUTER JOIN
3) צירוף עם שמירת מידע דו צדדית FULL OUTER JOIN
(לא קיים בגירסא החינמית של MYSQL)

נתונות הטבלאות הבאות:

R1

X	Y
a	1
b	2

R2

Y	Z
2	F
3	M

צירוף עם שמירת מידע שמאלה

```
SELECT   *
FROM     R1   LEFT OUTER JOIN   R2
ON       R1.y = R2.y
```

R1

X	Y
a	1
b	2

R2

Y	Z
2	F
3	M

X	R1.y	R2.y	Z
a	1	NULL	NULL
b	2	2	f

צירוף עם שמירת מידע ימנה

SELECT *

FROM R1 **RIGHT OUTER JOIN** R2

ON R1.y = R2.y

R1

X	Y
a	1
b	2

R2

Y	Z
2	F
3	M

X	R1.y	R2.y	Z
b	2	2	f
NULL	NULL	3	m

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

צירוף עם שמירת מידע דו כיוונית

SELECT *

FROM R1 FULL OUTER JOIN R2

ON R1.y = R2.y

לא עובד ב
MYSQL החינמי !

X	R1.y	R2.y	Z
a	1	NULL	NULL
b	2	2	f

LOJ

UNION

X	R1.y	R2.y	Z
b	2	2	f
NULL	NULL	3	m

ROJ

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

פעולות איחוד, חיתוך וחיסור

פעולות union, intersect, minus שקולות ל U, ∩, — באלגברת היחסים.

דוגמא 13: מצאו את <u>שמות הלקוחות</u> שיש להם חשבון חיסכון **<u>או</u>** הלוואה בסניף Aviv (או שניהם גם יחד)

```
SELECT    CustomerName
FROM      Deposit
WHERE     BranchName = 'Aviv'

UNION

SELECT    CustomerName
FROM      Borrow
WHERE     BranchName = 'Aviv'
```

[Query] **UNION** [Query]
[Query] **MINUS** [Query]
[Query] **INTERSECT** [Query]

לא עובד ב MYSQL החינמי !

שאלה: מצאו את <u>שמות הלקוחות</u> שהם בעלי חשבון הפקדה בסניף Aviv **<u>וגם לוו</u>** כסף בסניף זה – החלפה ב INTERSECT

שאלה: מצאו את <u>שמות</u> בעלי חשבונות ההפקדה בסניף Aviv **<u>שלא לוו</u>** כסף בסניף זה – החלפה ב EXCEPT / MINUS

בסיסי נתונים – קורס מתקדם – רואי זרחי ©

כפילויות בפעולת האיחוד

R1		
X	Y	Z
1	A	3
2	B	4
3	C	5

R2		
X	Y	Z
2	B	4
3	C	5
4	D	6

```
SELECT    *
FROM      R1
UNION
SELECT    *
FROM      R2
```

R1 UNION R2		
X	Y	Z
1	A	3
2	B	4
3	C	5
4	D	6

```
SELECT    *
FROM      R1
UNION ALL
SELECT    *
FROM      R2
```

R1 UNION ALL R2		
X	Y	Z
1	A	3
2	B	4
2	B	4
3	C	5
3	C	5
4	D	6

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

מיון נתונים

ניתן למיין את תוצאות השאילתא ע"י [ASC/DESC] ..... **ORDER BY**
פקודת המיון תופיע כפקודה האחרונה בשליפה.
ASC - מיון עולה (ברירת מחדל) ו **DESC** - יורד.

דוגמא 14: מצאו את פרטי הלקוחות במיון לפי עיר מגוריו של הלקוח

Customer		
customer-name	**street**	**customer-city**
Morag	Pinkas	Rishon
Tamir	Allenby	Haifa
Avivi	Pinkas	Rishon
Even	Allenby	Haifa

```
SELECT      *
FROM        Costumer
ORDER BY    CustomerCity
```

→

customer-name	**street**	**customer-city**
Tamir	Allenby	**Haifa**
Even	Allenby	**Haifa**
Avivi	Pinkas	**Rishon**
Morag	Pinkas	**Rishon**

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

מיון נתונים

דוגמא 15: מיינו את הסניפים לפי ערים ובכל עיר לפי שם סניף במיון יורד

Branch		
branch-name	assets	branch-city
Hamerkaz	9,000,000	Tiberias
Pinkas	2,100,000	Eilat
Aviv	1,700,000	Jaffa
Tsafon	400,000	Jaffa

branch-name	assets	branch-city
Pinkas	2,100,000	Eilat
Tsafon	400,000	Jaffa
Aviv	1,700,000	Jaffa
Hamerkaz	9,000,000	Tiberias

```
SELECT      *
FROM        Branch
ORDER BY    BranchCity ASC, BranchName DESC
```


בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

פונקציות הקבצה

פונקציות המקבלות כקלט <u>אוסף של ערכים</u> ומחזירות <u>ערך יחיד</u>.

הפונקציה (F) תופיע בפקודת ה Select בפורמט הבא:

```
SELECT    F(column)
FROM      Table
```


בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

פונקציות הקבצה

<u>קיימות חמש פונקציות הקבצה</u>:

- ממוצע: **avg**
- מינימום: **min**
- מקסימום: **max**
- סכום: **sum**
- ספירה: **count**

```
SELECT    MAX(Amount)
FROM      Table
```

- הפעולה count(a) תבצע ספירה של כל הערכים הקיימים השונים מ-NULL.
- הפונקציות MIN,MAX פועלות גם על מחרוזות (לפי סדר לקסיקוגרפי).
- הפונקציה COUNT גם כן פועלת על מחרוזות.

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

פונקציות הקבצה - דוגמא

דוגמא 16: מצאו את <u>סכום הנכסים הממוצע</u> בסניפים ביפו

Branch		
branch-name	assets	branch-city
Hamerkaz	9,000,000	Tiberias
Pinkas	2,100,000	Eilat
Aviv	1,700,000	Jaffa
Tsafon	400,000	Jaffa

```
SELECT   AVG(assets)
FROM     Branch
WHERE    BranchCity = 'Jaffa'
```

→

AVG(assets)
1,050,000

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

פונקציות הקבצה - דוגמא

דוגמא 17: מצאו את <u>כמות הלקוחות</u> בעלי חשבון הפקדה עם יתרה מעל 450 ₪ וששם הסניף בו מופקד הכסף מכיל את האות a.

Deposit			
branch-name	account-number	customer-name	balance
Hamerkaz	101	Even	500
Tsafon	215	Tamir	700
Aviv	102	Avivi	400

count
2

```
SELECT   COUNT(costumer-name)
FROM     Deposit
WHERE    (balance > 450)  AND  (BranchName like '%a%')
```


בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

פונקציות הקבצה

<u>שאלה</u>: האם ניתן לבצע הרכבה של פונקציות: AVG(MIN(amount)) ?

פונקציות הקבצה מקבלות כקלט <u>אוסף של ערכים</u> ומחזירות <u>ערך יחיד</u>.

Borrow			
branch-name	loan-number	customer-name	amount
Hamerkaz	17	Morag	1000
Pinkas	23	Tamir	2000
Aviv	15	Avivi	1500
Tsafon	93	Even	500

```
SELECT MIN(amount)
FROM Borrow
```

→

amount
500

```
SELECT AVG(amount)
FROM Borrow
```

→

amount
1250

~~SELECT AVG(MIN(amount))~~
~~FROM Borrow~~

פונקצית המינימום מחזירה ערך בודד ולכן אין שום משמעות להפעלת פונקצית הממוצע על ערך בודד זה

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

פונקציות הקבצה – Group by

הפעלת פונקצית הקבצה על אוסף קבוצות של n-יות (ולא על קבוצה), תתבצע ע"י שימוש ב **GROUP BY**.

פעולת **GROUP BY** מחלקת את הטבלה לקבוצות לפי התכונות שמצוינות בפסוקית זו, כאשר שורות בעלות ערכים זהים מקובצות לאותה קבוצה.

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

פונקציות הקבצה – Group by

דוגמא 18: מצאו כמה לקוחות גרים בכל עיר

```
SELECT      CostumerCity, Count(*)
FROM        Costumer
GROUP BY    CostumerCity
```

Customer

customer-name	street	customer-city
Morag	Pinkas	Rishon
Tamir	Allenby	Haifa
Avivi	Pinkas	Rishon
Even	Allenby	Haifa

→

customer-city	Count
Rishon	2
Haifa	2

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

פונקציות הקבצה – Group by

דוגמא 18 המשך: מצאו <u>בכמה ערים שונות</u> מתגוררים הלקוחות

Customer		
customer-name	**street**	**customer-city**
Morag	Pinkas	Rishon
Tamir	Allenby	Haifa
Avivi	Pinkas	Rishon
Even	Allenby	Haifa

Count
2

SELECT Count(**Distinct** CostumerCity)
FROM Costumer

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

הקבצה כפולה

דוגמא 19: מצאו את <u>כמות הסטודנטים</u> שגרים בכל רחוב בכל עיר מתוך טבלת התלמידים במיון לפי שם העיר

Talmidim			
Name	**Street**	**City**	**Amount**
Avi	Begin	Ramat-Gan	1200
Avi	Begin	Ramat-Gan	3600
Ben	Hayarkon	Tel-Aviv	4000
Chen	Herzel	Tel-Aviv	2000
Debi	Hashalom	Givatiim	700
Zvi	Vaitzman	Givatiim	3500
Haim	Haela	Jerusalem	1000

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

הקבצה כפולה – המשך...

פתרון: מצאו את כמות הסטודנטים שגרים בכל רחוב בכל עיר מתוך טבלת התלמידים במיון לפי שם העיר

```
SELECT    City , Street , COUNT(Name) AS Counter
FROM      Talmidim
GROUP BY  City , Street
ORDER BY  City
```

City	Street	Counter
Givatiim	Hashalom St.	1
Givatiim	Vaitzman St.	1
Jerusalem	Haela St.	1
Ramat-Gan	Begin St.	2
Tel-Aviv	Hayarkon St.	1
Tel-Aviv	Herzrl St.	1

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

סינון שורות לאחר פעולת הקבצה

שאלה: מצאו את <u>כמות הסטודנטים</u> שגרים בכל רחוב בכל עיר מתוך טבלת התלמידים במיון לפי שם העיר.

תוספת: יש להציג רק את הערים בהם כמות הסטודנטים גדולה מ-1.

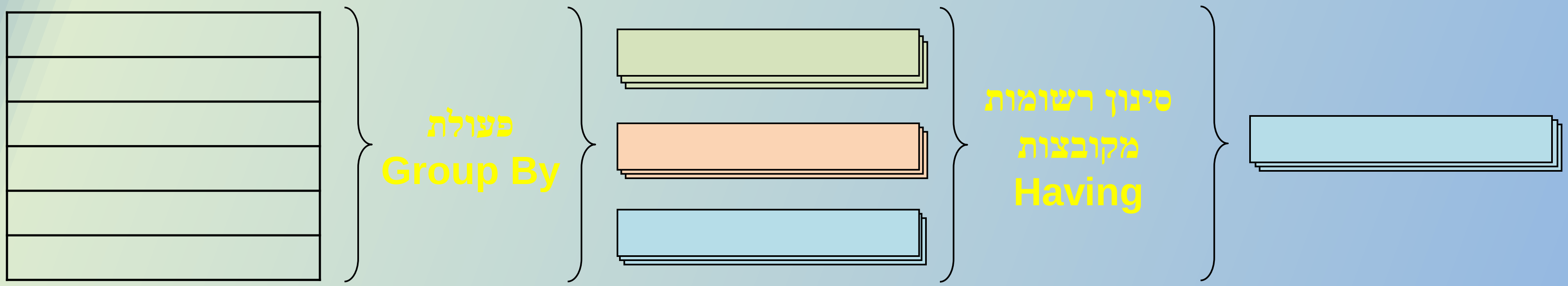
City	Street	Counter
Givatiim	Hashalom St.	1
Givatiim	Vaitzman St.	1
Jerusalem	Haela St.	1
Ramat-Gan	Begin St.	2
Tel-Aviv	Hayarkon St.	1
Tel-Aviv	Herzrl St.	1

City	Street	Counter
Ramat-Gan	Begin St.	2

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

פעולת Having

פקודת ה HAVING בוחרת **<u>קבוצות המקיימות</u>** תנאי על גבי פונקצית ההקבצה



בפועל מתבצעת הצבת תנאי סינון על כל קבוצה וזאת לאחר שבוצעה פעולת הקבצה שהשתמשה ב - Group by.

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

הפעלת תנאי על קבוצות

City	Street	Counter
Givatiim	Hashalom St.	1
Givatiim	Vaitzman St.	1
Jerusalem	Haela St.	1
Ramat-Gan	Begin St.	2
Tel-Aviv	Hayarkon St.	1
Tel-Aviv	Herzrl St.	1

```
SELECT      city, street, Count(*)  AS counter
FROM        Talmidim
GROUP BY    City, street
HAVING      Count(*) >1
```

City	Street	Counter
Ramat-Gan	Begin St.	2

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

שימוש בפונקצית הקבצה זהה

דוגמא 20: מצאו את שמות הערים וממוצע הנכסים שלהם רק עבור ערים בהם <u>הממוצע</u> גבוה מ 2,000,000 ₪

```
SELECT    BranchCity, AVG(assets)
FROM      Branch
GROUP BY  BranchCity
HAVING    AVG(assets) > 2,000,000
```

Branch		
branch-name	assets	branch-city
Hamerkaz	9,000,000	Tiberias
Pinkas	2,100,000	Eilat
Aviv	1,700,000	Jaffa
Tsafon	400,000	Jaffa

Group by

branch-city	assets
Tiberias	9,000,000
Eilat	2,100,000
Jaffa	1,050,000

Having

branch-city	assets
Tiberias	9,000,000
Eilat	2,100,000

שימוש בפונקצית הקבצה שונה

דוגמא 21: מציאת היתרה הממוצעת לכל סטודנט שגר ברמת-גן שיש לו מעל שני חשבונות

Talmidim			
Name	**Street**	**City**	**Amount**
Avi	Begin	Ramat-Gan	1200
Avi	Begin	Ramat-Gan	3600
Ben	Hayarkon	Tel-Aviv	4000
Chen	Herzel	Tel-Aviv	2000
Debi	Hashalom	Givatiim	700
Zvi	Vaitzman	Givatiim	3500
Haim	Haela	Jerusalem	1000

```
SELECT      Name, Avg(Amount) AS AvgAmount
FROM        Talmidim
WHERE       City = 'Ramat-Gan'
GROUP BY    Name
HAVING      Count(*) > 2
```

→

Name	AvgAmount

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

COUNT בשילוב עם DISTINCT

האם יש הבדל ? ספירה ללא חזרות או ביטול חזרות לאחר ספירה...

Students			
Name	**Street**	**City**	**Amount**
Avi	Begin	Ramat-Gan	1200
Avi	Begin	Ramat-Gan	7600
Ben	Hayarkon	Tel-Aviv	4000
Chen	Herzrl	Tel-Aviv	2000
Debi	Hashalom	Givatiim	700

```
SELECT DISTINCT count(Name)
FROM Students;
```

Count
5

```
SELECT count(DISTINCT  Name)
FROM Students;
```

Count
4

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

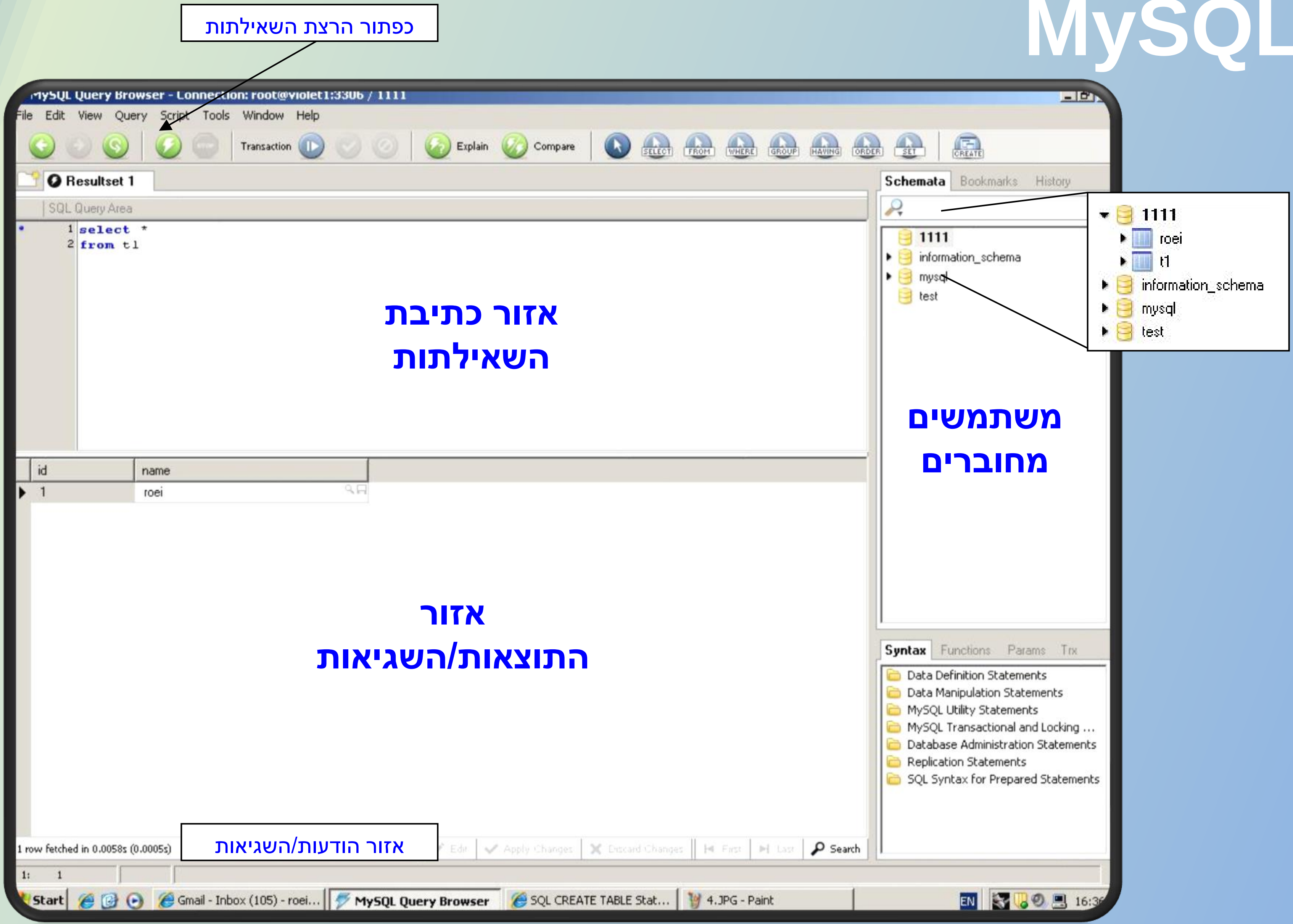
SQL – סיכום ביניים - מבנה פקודה

SELECT	Name, Count(*)
FROM	Students **AS** S
LEFT JOIN	Courses **AS** C
ON	S.courseID = C. courseID
WHERE	S.id **BETWEEN** 3344 **AND** 3355
GROUP BY	Name
HAVING	Count(*) >2
ORDER BY	Name

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

תזכורת: עבודה עם MySQL

MySQL



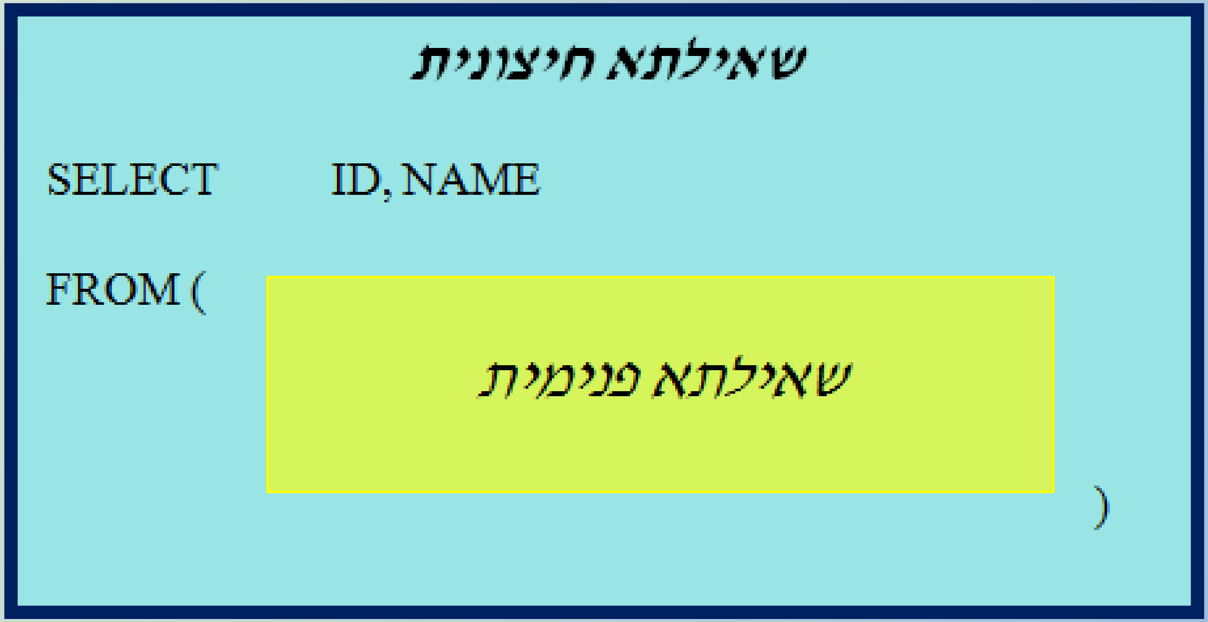
בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

תתי שאילתות ואופרטורים נוספים

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

תתי שאילתות Sub-Queries

תת שאילתא הינה שאילתא רגילה הנמצאת בתוך שאילתא אחרת, כך שתוצאת השאילתא הפנימית (טבלה) משמשת לחישוב השאילתא החיצונית.



תתי שאילתות Sub-Queries

דוגמא 22: נניח שקיימות בבנק 2 טבלאות סניפים: טבלת סניפים ראשיים (B1) וטבלת סניפים זמניים (B2) ומנכ"ל הבנק ביקש לקבל את מספר הסניפים הכולל.

B1		
name1	Assets1	city1
Hamerkaz	9,000,000	Tiberias
Pinkas	2,100,000	Eilat
Aviv	1,700,000	Jaffa
Tsafon	400,000	Jaffa

B2		
name2	Assets2	city2
Lev	5,500,000	Tel-Aviv
Darom	3,600,000	Haifa
Migdal	750,000	Arad

```
SELECT    Count (name1) + ( SELECT Count (name2) FROM B2 )
 FROM     B1
```

```
SELECT    Count (name1) + ( 3 )
 FROM     B1
```


בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

סוגי תתי שאילתות

אנו נעסוק בשני הסוגים העיקריים של תתי שאילתות שניתן לשלב במהלך כתיבת שאילתות.

א) תתי שאילתות המחזירות ערך יחיד

ב) תתי שאילתות המחזירות אוסף ערכים

הערה: ניתן לשלב תתי שאילתות בפקודות WHERE ו HAVING (השימוש הנפוץ ביותר) אך ניתן לשלבם גם ב:

- **בפקודת SELECT - החזרת ערך בודד**
- **בפקודות FROM - החזרת טבלה**

א) תתי שאילתות המחזירות ערך יחיד

דוגמא 23: רשימת שמות התלמידים שהיתרה בחשבונותיהם גבוהה מהיתרה הממוצעת בכל חשבונות התלמידים

Talmidim			
Name	**Street**	**City**	**Amount**
Avi	Begin	Ramat-Gan	1200
Avi	Begin	Ramat-Gan	3600
Ben	Hayarkon	Tel-Aviv	4000
Chen	Herzel	Tel-Aviv	2000
Debi	Hashalom	Givatiim	700
Zvi	Vaitzman	Givatiim	3500
Haim	Haela	Jerusalem	1000

```
SELECT   Name
FROM     Talmidim
WHERE    Amount > (  SELECT  AVG(amount)
                     FROM    Talmidim  )
```

2286

Name
Avi
Ben
Zvi

הערה:
בתת השאילתא נפוץ השימוש בפונקצית הקבצה (המחזירה תמיד ערך בודד).

בסיסי נתונים – קורס מתקדם – רואי זרחי ©

תתי שאילתות המחזירות ערך יחיד

דוגמא 24 א': מצאו את שמות הסניפים שנמצאים בעיר של סניף Aviv.

הנחה: אין שתי ערים בהן קיים סניף בעל אותו שם

```
SELECT     BranchName
FROM       Branch
WHERE      BranchCity = ( SELECT    BranchCity
                          FROM      Branch
                          WHERE     BranchName = 'Aviv' )
```

branch-city
Jaffa

branch-name
Aviv
Tsafon

הערה:

אם תת השאילתא לא הייתה מחזירה ערך בודד, אלא אוסף ערכים הייתה מתקבלת הודעת שגיאה (לא היה ניתן לבצע את פעולת ההשוואה)

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

תתי שאילתות המחזירות ערך יחיד

דוגמא 24 ב': מצאו את שמות הסניפים שנמצאים בעיר של סניף Aviv.

איך היינו פותרים שאלה זו ללא שימוש בתת-שאילתא ?

```
SELECT    B1.BranchName
FROM      Branch B1, Branch B2
WHERE     (B1.BranchCity = B2.BranchCity) AND
          (B2.BranchName = 'Aviv')
```


בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

תתי שאילתות המחזירות ערך יחיד

שאלה: במקרה זה, מה באמת יותר יעיל ואיך ניתן לבדוק זאת ?

- האם מכפלה קרטזית ?
- האם צירוף טבלאות ?
- האם תת-שאילתא ?

תרגיל
כיתה 1

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

פתרון - תרגיל

ע"י הרצת 3 השאילתות בעצם בדקנו את מהירות הריצה וכמות הפעולות שמבוצעות בפועל מאחורי הקלעים.

תוצאת ההרצה של שאילתא ב-SQL מופיעה כך:

A rows fetched in X seconds (Y seconds)

כמות הרשומות שמופיעות בתוצאת השאילתא

הזמן (כמות שניות) שלקח לשרת של MYSQL לשלוח את טבלת התוצאות למשתמש (**להציג** על המסך)

הזמן (כמות שניות) שלקח לשרת של MYSQL **להריץ** את השאילתא עצמה (ולכן תמיד X > Y)

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

פתרון - תרגיל

מכפלה קרטזית

```
SELECT   B1.BranchName
FROM     Branch B1, Branch B2
WHERE    (B1.BranchCity = B2.BranchCity) AND
         (B2.BranchName = "Aviv")
```

צירוף

```
SELECT   B1.BranchName
FROM     Branch B1 join Branch B2
ON       (B1.BranchCity = B2.BranchCity)
WHERE    B2.BranchName = "Aviv"
```

תת-שאילתא

```
SELECT   BranchName
FROM     Branch
WHERE    BranchCity = (  SELECT   BranchCity
                         FROM     Branch
                         WHERE    BranchName = "Aviv" )
```


בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

פתרון - תרגיל

פעולה ב SQL	תוצאה	זמן ריצה	כמה פעולות בוצעו
מכפלה קרטזית	2 רשומות	0.0008s (**0.0004s**)	50*50 = 2500
צירוף	2 רשומות	0.0006s (**0.0003s**)	196, שכן מכפיל רק את המקרים בהם הערכים שווים
תת שאילתא	2 רשומות	0.0016s (**0.0002s**)	50+50 = 100

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

ב) תתי שאילתות המחזירות אוסף ערכים

תתי שאילתות מסוג זה מכילות אופרטורים המאפשרים לבצע השוואה על אוסף ערכים (ולא על ערך בודד) ולשם כך נלמד 4 אופרטורים חדשים*:

SOME() – יופיע לאחר אופרטור השוואה ויענה לשאלה האם קיים לפחות ערך אחד מתוך רשימת איברי הקבוצה שנמצאים בסוגריים.

ALL() – יופיע לאחר אופרטור השוואה, ויבדוק האם מתקיים התנאי לכל איברי הקבוצה הנמצאים בסוגריים.

IN() – בודק השתייכות לקבוצת ערכים (גם למול רשימת ערכים*).

EXISTS() – בודק האם קיימות שורות בתוצאת תת השאילתא העונות לתנאי שהוגדר.

*בעיקרון השימוש באופרטורים אלו מתבצע רק בעבודה מול תתי-שאילתות.

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

אופרטור SOME

דוגמא 25: מצאו את רשימת הסניפים שערך נכסיהם שווה לערך הנכסים של סניף כלשהו ב Jaffa

```
SELECT   BranchName
FROM     Branch
WHERE    assets = SOME ( SELECT assets
                         FROM   Branch
                         WHERE  BranchCity = 'Jaffa' )
```

assets
Aviv
Tsafon

assets
1,700,000
400,000

** במקרה זה הסניפים שערכי נכסיהם הם אותם הנכסים עליהם שאלנו.

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

אופרטור ALL

דוגמא 26: מצא את שם הלקוח שגובה ההלוואה שלקח גדול שווה <u>להלוואות</u> שנלקחו ע"י לקוחות אחרים

```
SELECT    CostumerName
FROM      Borrow
WHERE     amount ≥ ALL ( SELECT  amount
                         FROM    Borrow )
```

customer-name
Tamir

amount
1000
2000
1500
500

** שקול לפעולת מקסימום.

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

אופרטור IN

דוגמא 27: עבור הלקוחות בעלי חשבון חיסכון בסניף Aviv מצאו את אלו שהם גם בעלי חשבון הלוואה בסניף זה

```
SELECT   CostumerName
FROM     Borrow
WHERE    BranchName = 'Aviv' AND
         CostumerName IN (SELECT CostumerName
                          FROM   Deposit
                          WHERE  BranchName = 'Aviv' )
```

customer-name
Avivi

customer-name
Avivi

הערות:

1) ניתן להשתמש ב **NOT IN** בכדי להגדיר תנאי של אי-שייכות.

2) ניתן להשתמש ב IN גם לבדיקה למול רשימת ערכים.

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

טווח ייחוס של טבלאות בתתי שאילתות

בתת-שאילתא ניתן להשתמש:

1) בטבלאות שהוגדרו בתת השאילתא עצמה.
2) בטבלאות שהוגדרו בכל שאילתה שמכילה אותה.

```
SELECT    R.ID
FROM      R
WHERE     R.name = (SELECT S.name FROM S WHERE…)
```

```
SELECT    R.ID
FROM      R
WHERE     R.name = (SELECT R.name FROM R WHERE…)
```

```
SELECT    R.ID
FROM      R
WHERE     R.name = (SELECT S.name FROM S WHERE R = …)
```

אם טבלה מוגדרת גם בתת השאילתה וגם בשאילתא העוטפת אותה, הייחוס יהיה להגדרה המקומית בתת השאילתא.

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

אופרטור EXISTS

דוגמא 28: מצאו את שמות הלקוחות שיש (קיים) להם חשבון חיסכון וגם חשבון הלוואה בסניף Aviv

```
SELECT    CostumerName CN
FROM      Customer C

WHERE     EXISTS (  SELECT CostumerName CN
                    FROM   Deposit D
                    WHERE  D.CN = C.CN AND BranchName = 'Aviv' )

      AND EXISTS (  SELECT CostumerName CN
                    FROM   Borrow B
                    WHERE  B.CN = C.CN AND BranchName = 'Aviv' )
```

מתבצעות שתי בדיקות קיום - האם קיים לקוח (מטבלת לקוחות) שיש לו חשבון חיסכון בסניף אביב וגם **שאותו לקוח** בעל חשבון הפקדה בסניף אביב

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

תתי שאילתות במשפטי FROM

צורת השימוש:

חובה לבצע את פעולת הכינוי

```
SELECT ....
FROM    (Sub-Query) AS NewName;
```

נחזור לדוגמא 24: מהם שמות הסניפים שנמצאים בעיר של סניף Aviv.

```
SELECT      BranchName
FROM        Branch B, ( SELECT BranchCity
                        FROM    Branch
                        WHERE   BranchName = 'Aviv' ) AS X
WHERE       B.BranchCity = X.BranchCity
```

** חלק מגרסאות MYSQL אינן תומכות בכל סוגי תתי השאילתות.

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

סיום שיעור 1