בסיסי נתונים - קורס מתקדם

2024 – תשפ"ד (סמסטר קיץ)

מרצה: רואי זרחיה

שיעור 2

האם יש שאלות משיעור קודם ?

?　　　　?　　　　?　　　　?

?　　　　?

?　　　　?

?　　　　?

?

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

פקודת ה SELECT – המשך...

תזכורת: המערכת הבנקאית

Bank Database

Branch

ID	Name	Assets	City

Customer

ID	Name	street	City

Deposit

Customer-ID	Branch-ID	Deposit-Number	balance

Borrow

Customer-ID	Branch-ID	Borrow-Number	balance

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

אופרטור LIMIT

קיים אופרטור נוסף שניתן להשתמש בו במהלך כתיבת השאילתות.
על מנת להבין לעומק את יכולות האופרטור הנ"ל, נבצע תרגיל שימחיש זאת.

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

<u>תרגיל</u>: עליכם לבצע צירוף בין טבלת Branch לעצמה ולהוסיף בתחתית השאילתא את הפקודה המופיעה למטה ומכילה את האופרטור LIMIT, הריצו את השאילתא ונסו להבין מה בדיוק עושה האופרטור...

```
SELECT  *
FROM    Branch B1 Join Branch B2
LIMIT   5,10
```


בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

הגבלת כמות הרשומות החוזרות

נניח שאנו מריצים שאילתת ענק והתוצאה שחוזרת ממנה היא מעל מליון שורות... מה ניתן לעשות בכדי לצפות רק בחלק מהשורות ולא בכל התוצאה ?

ובכן, לשם כך קיימת פקודת LIMIT שמקבלת פרמטר (מיספרי) אחד או שניים.

```
SELECT *
FROM Branch
LIMIT 5,10
```

כמה רשומות נרצה לשלוף

הרשומה הראשונה שנרצה לשלוף

במקרה שמצויין רק פרמטר אחד אזי פקודת LIMIT תחזיר את כמות השורות המבוקשת החל מהרשומה הראשונה

דוגמא זו מחזירה את רשומות מספר 6-15 מתוך התוצאה (הרשומה הראשונה הינה רשומה 0) ולכן ביקשנו להחזיר 10 רשומות החל מהרשומה ה-5 (שהיא בפועל רשומה 6 אם מתחילים לספור מ-0).

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

פקודות DML נוספות

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

עדכון בסיס הנתונים

שפות השאילתות הפורמאליות, כמו אלגברת יחסים לא כוללות כלים לעדכון בסיס הנתונים.

מאידך השפות המסחריות (SQL) מאפשרות ביצוע שוטף של עדכון בסיס הנתונים בכל זמן נתון, כמו גם הוספה ומחיקה של נתונים.

<u>פעולות DML לעדכון בסיס הנתונים:</u>

1) הוספת רשומות - INSERT
2) מחיקת רשומות - DELETE
3) עדכון רשומות - UPDATE

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

1) הוספת רשומות

ניתן להוסיף רשומות ליחס בשתי צורות:

1) הוספה של רשומה בצורה מפורשת
2) הוספת קבוצת רשומות מטבלה אחרת (מתוך תוצאת תת-שאילתא)

1) הוספה של רשומה בצורה מפורשת לטבלת הפקדות

```
INSERT INTO  deposit
VALUES       ('Darom', 55, 'Levi', 760)
```

Deposit			
branch-name	account-number	customer-name	balance
Hamerkaz	101	Even	500
Tsafon	215	Tamir	700
Aviv	102	Avivi	400
Darom	55	Levi	760

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

הוספת רשומות בשילוב תת-שאילתא

2) הוספת חשבון הפקדה שבו יופקדו 200 ₪ **<u>לכל לקוח</u>** שלקח הלוואה בסניף Aviv (מספר חשבון ההפקדה זהה למספר חשבון ההלוואה)

```
INSERT INTO  deposit
   SELECT    branch-name, loan-number, customer-name, 200
   FROM      borrow
   WHERE     branch-name = 'Aviv'
```

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

Deposit			
branch-name	account-number	customer-name	balance
Hamerkaz	101	Even	500
Tsafon	215	Tamir	700
Aviv	102	Avivi	400
Aviv	15	Avivi	200

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

הוספת רשומות - המשך

במקרה בו לא כל הנתונים ידועים מראש, ניתן להכניס ערכים רק לתכונות ספציפיות ע"י הגדרת שמות התכונות במפורש בפקודת ההוספה (כאשר שאר הערכים יקבלו באופן אוטומטי ערכי NULL).

```
INSERT INTO   deposit ( branch-name, customer-name )
VALUES        ('Aviv' , 'Avivi')
```

Deposit

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

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

2) מחיקת רשומות

דוגמא 29: מחקו את כל הרשומות מטבלת borrow

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

Borrow			
branch-name	loan-number	customer-name	amount

DELETE FROM borrow

פקודה זו תשמור על הסכמה borrow בבסיס הנתונים אך סכמה זו תישאר ריקה (למחיקת הסכמה עצמה נזדקק לפקודת DDL ייעודית לכך).

מחיקת רשומות ממוקדת

מחיקה מהיחס R של רשומות המקיימות את התנאי P

```
DELETE
FROM    R
WHERE   P
```

דוגמא 30: מחקו את כל חשבונות החסכון של Tamir

```
DELETE
FROM    borrow
WHERE   customer-name = 'Tamir'
```

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

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

בסיסי נתונים – קורס מתקדם – רואי זרחי ©

פקודת TRUNCATE

פעולת Truncate הינה פעולה המנקה את הטבלה מתוכנה ומחזירה אותה למצב ההתחלתי לפני הזנת נתונים כלשהם בתוכה. הפעולה משחררת את הזיכרון שהוקצה עבור הנתונים.

פקודה זו מהירה יותר מפקודת DELETE **ולא ניתן להפעילה על חלק מהרשומות אלא תמיד על כולן יחד** (פקודת TRUNCATE תאפס את הנומרטור האוטומטי, בניגוד ל DELETE שתמשיך לספור).

הסיבה למהירותה הגבוהה היא העובדה שפקודה זו מוחקת את הטבלה הקיימת ויוצרת טבלה חדשה (עם אותו שם) במקומה, בניגוד לפקודת DELETE שמוחקת שורה-שורה.

מבנה הפקודה:

Truncate table **TableName**

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

3) עדכון רשומות

מטרתה של פעולת העדכון הינה לבצע שינוי ערכים בשדות שבהם כבר הוזנו ערכים בעבר.

ניתן לבצע עדכון בשתי דרכים:

<u>א) עדכון של כל ערכי התכונה ביחס</u>

דוגמא 31: הוסיפו 5% **<u>לכל</u>** יתרות חשבונות החיסכון

```
UPDATE    deposit
SET       balance = balance*1.05
```


בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

עדכון רשומות - המשך

<u>ב) עדכון של ערכים ספציפיים (העונים לתנאי מסוים)</u>

דוגמא 32: הוסיפו 6% ליתרות חשבונות החיסכון עם יתרה מעל 1000 ₪ והוסיפו 7% ליתרות חשבונות החיסכון עם יתרה מתחת ל 1000 ₪

עדכון טבלאות שונות או עדכון לפי תנאי סינון שונים מחייבים שימוש בפקודות UPDATE נפרדות.

```
UPDATE    deposit
SET       balance = balance*1.06
WHERE     balance > 1000
```

```
UPDATE    deposit
SET       balance = balance*1.07
WHERE     balance ≤ 1000
```


בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

עדכון רשומות ע"י שימוש בתתי שאילתות

מה תחזיר השאילתא הבאה ?

```
UPDATE      Employees
SET         ( EmpNo, Salary, Department ) =
    ( SELECT EmpNo, Salary, Department
      FROM    Emp1
      WHERE  EmpNo = 7499 )
WHERE EmpNo = 7369;
```


בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

עדכון רשומות ע"י שימוש בתתי שאילתות

פתרון ל "מה תחזיר השאילתא הבאה ?"

```
UPDATE      Employees
SET         ( EmpNo, Salary, Department ) =

  ( SELECT EmpNo, Salary, Department
    FROM     Emp1
    WHERE  EmpNo = 7499 )

WHERE EmpNo = 7369;
```

תת השאילתא חייבת להחזיר
רק שורה אחת בכדי לבצע את
פקודת ה Update (במקרה זה
תוחזר הרשומה של עובד 7499)

השורה היחידה שתעודכן בטבלת עובדים תהיה של עובד מס'
7369 והיא תעודכן בערכים של עובד 7499 מטבלת Emp1

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

פקודות התנייה

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

פקודת תנאי - Case

פקודת ה CASE הינה כלי להצגת תוצאות שונות בכפוף לערכים או תנאים המבוססים על בדיקה מוקדמת של הערך המוחזר.

ב SQL קיימות 2 גירסאות של CASE:

Simple Case Expression

פקודה המאפשרת לבצע בדיקת ערך של משתנה בודד (שאלת תנאי פשוטה) והצגת ערך אחר במקומו.

Search Case Expression

פקודה שבודקת מספר משתנים ומגיבה לפי הצורך (שאלת תנאי מורכבת).

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

מבנה הפקודה – Simple Case

```
SELECT…
     CASE column_to_be_checked
        WHEN value THEN value
      [
      ELSE else_result_expression
      ]
     END
FROM …
```

*** חשוב: משפט ה Else הוא אינו חובה ובמקרה שלא יופיע אזי כל הערכים שלא מפורטים בשאילתא יקבלו אוטומטית ערכי NULL.

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

דוגמא – Simple Case

הדוגמה הבאה משתמשת בביטוי CASE בכדי לתאר 3 מקרים של ציוני בחינות עבור [1] סטודנט שלא ניגש [2] סטודנט שניגש והוציא ציון מושלם ועבור [3] כל שאר המקרים:

```
SELECT Grade, CASE Grade
    WHEN 0 THEN 'Did not attend the test'
    WHEN 100 THEN 'Perfect score'
    ELSE 'Normal grade'
    END 'grade'
FROM Students
```

השם הסוגר את פקודת ה END אינו חובה, הוא משמש כתגית לסגירת ה CASE

חשוב: פקודת ה Simple Case מבצעת רק את פעולת השיוויון למול הערך הנבדק. על מנת לשלב סימני השוואה אחרים (גדול, קטן, שונה) נצטרך להשתמש בפקודת Search Case.

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

מבנה הפקודה – Search Case

```
SELECT…
    CASE
    WHEN condition_or_function THEN value
    ]
    ELSE value
    [
    END
FROM…
```

<u>הערה</u>: ניתן לראות שבניגוד ל Simple Case כאן לא כותבים את שם העמודה הנבדקת מיד לאחר המילה CASE שכן אנו בעצם שואלים מספר שאלות עם תנאים שונים.

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

דוגמא – Search Case

```
SELECT Assets,
    CASE
        WHEN Assets IS NULL
                THEN 'No Assets Available'
        WHEN Round(Assets) > 1000000
                THEN 'There Are Many Assets'
        WHEN Assets < 0
                THEN 'You Owe Money To The Bank'
        ELSE 'There Are A Few Assets'
    END AS 'Assets'
FROM Branch
```

הערה: ניתן לראות שמוגדרים כאן תנאים מכל סוג שהוא ובעקבותיהם הערכים שנרצה להציג למשתמש.

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

פקודת תנאי - IF

כמו בפקודות בשפות תכנות, ניתן להשתמש בפקודת התניה (IF) במהלך כתיבת שאילתות כאשר יש צורך להגדיר את השאלה שאותה נרצה לבחון ואת המקרים והתגובות של כל תוצאה אפשרית.

שאלת ה IF תבדוק קיום של תנאי שעבורו תבוצע פקודה או סט של פקודות.

בנוסף בחלק ממערכות בסיסי הנתונים קיימת אפשרות להוספת ELSEIF שמשמעותו הגדרת התרחיש במקרה בו תנאי ה IF אינו מתקיים.

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

פקודת תנאי – IF

דוגמא של פקודת IF להשוואה בין שתי תוצאות שאילתות ב **MYSQL**:

SELECT IF ((Value) = (Value) ***, "Equal", "Doesn't Equal")***

<u>דוגמא:</u>

SELECT IF (
(SELECT TestGrade from S) = (SELECT ExeGrade from S)
, "Same Grades", "Different Grades")

פקודת ה IF תריץ בעצם שתי תתי-שאילתות וישווה את התוצאות החוזרות במטרה לבדוק שוויון בערכים (השוואת ציון מבחן לציון תרגיל) ויחזיר את התוצאה שנגדיר במקרה של שוויון ובמקרה של אי-שוויון.

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

פקודות DDL

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

פקודות DDL

פקודות DDL מאפיינות את היחסים בבסיס הנתונים ע"י מספר פקודות:

- **יצירת בסיס נתונים** / בחירת בסיס נתונים קיים מתוך רשימה
- **יצירת סכמה** חדשה בבסיס הנתונים (תבנית, תחומי ערכים ואילוצים)
- **עדכון ומחיקת סכמה** קיימת מבסיס הנתונים
- **טבלאות זמניות**
- **הגדרת אינדקס** עבור טבלה
- **הגדרת תצפית** בבסיס הנתונים

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

פקודות DDL

בשלב הראשוני, בעת הכניסה למערכת יתכן ועדיין אין לנו שום בסיס נתונים לעבוד עליו (ב MYSQL זה לא נפוץ אך בשאר המערכות - כן):

כעת, בכדי ליצור בסיס נתונים חדש, נוכל להשתמש בפקודה:

CREATE DATABASE New_Databade_Name;

מחיקת בסיס הנתונים על כל טבלאותיו תתבצע ע"י הפקודה:

DROP DATABASE database_name;

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

פקודות DDL

על מנת לראות את כל בסיסי הנתונים הקיימים נוכל לרשום את פקודת:

```
SHOW DATABASES;
```

כעת, אם נרצה לבחור בסיס נתונים ספציפי מרשימת בסיסי הנתונים הקיימים, נצטרך לרשום מהו בסיס הנתונים שבו נרצה להשתמש החל מרגע זה:

```
USE Existing_Database_Name;
```


בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

פקודות DDL

על מנת לראות את כל הטבלאות
בבסיס הנתונים נוכל לרשום את פקודת:

SHOW TABLES;

Tables_in_bank
borrow
branch
customer
deposit

על מנת לקבל תאור של רכיבי טבלה (שדות, מפתחות וכד'), נרשום:

DESCRIBE TableName;

שינוי שם של טבלה קיימת:

ALTER TABLE Old_Borrow RENAME New_Borrow;

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

פקודות DDL

פקודות DDL מאפיינות את היחסים בבסיס הנתונים ע"י מספר פקודות:

- **יצירת בסיס נתונים** / בחירת בסיס נתונים קיים מתוך רשימה
- **יצירת סכמה** חדשה בבסיס הנתונים (תבנית, תחומי ערכים ואילוצים)
- **עדכון ומחיקת סכמה** קיימת מבסיס הנתונים
- **טבלאות זמניות**
- **הגדרת אינדקס** עבור טבלה
- **הגדרת תצפית** בבסיס הנתונים

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

יצירת ומחיקת סכמות

על מנת להוסיף סכמה חדשה לבסיס הנתונים יש להגדיר את שם הסכמה ולאחר מכן את התכונות, טיפוסיהם וההגבלות על תכונות אלו (אם קיימות).

דוגמא 33: צרו את הסכמה של Branch

```
CREATE TABLE   Branch
(
    branch-name    text,
    assets         int,
    branch-city    text,
    Primary key (branch name)
)
```

Branch		
branch-name	assets	branch-city

<u>מחיקת יחס מבסיס הנתונים</u>

דוגמא 34: מחקו את טבלת Branch

```
DROP TABLE  Branch
```

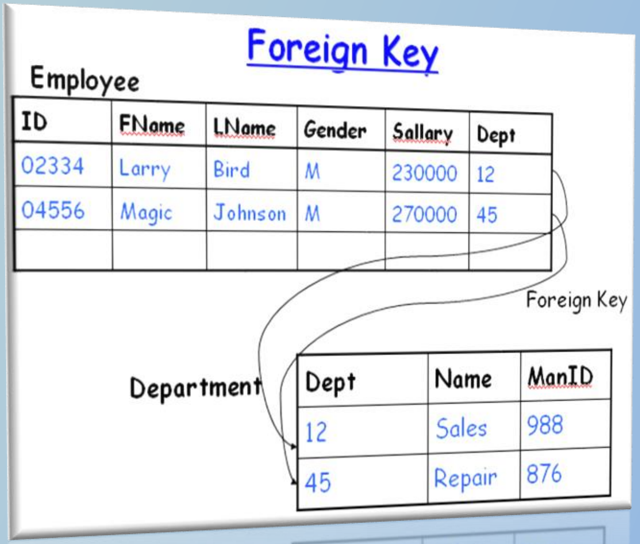

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

אילוצים ביצירת טבלה חדשה

כאשר נוסיף אילוץ ביצירת טבלה חדשה על אחת העמודות אנו בעצם מאפשרים למסד הנתונים לבצע אכיפה של שלמות המידע המוכנס.

מצד שני, חשוב להבין שהוספת אילוץ תגרור האטה בהכנסת נתונים חדשים לטבלה (בדיקת NULL, מפתח ראשי/זר וכד').

```
CREATE TABLE Employees
(
    ID          int  NOT NULL,
    Fname  text(10),
    Lname  text(20),
    Gender char(1) DEFAULT 'F',
    Salary  float NOT NULL,
    Dept     int,
    PRIMARY KEY (ID),
    FOREIGN KEY (Dept) REFERENCES Department(Dept)
);
```



בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

אילוצים נוספים

ניתן להוסיף מספר אילוצים נוספים שיאפשרו לנו לקבוע את הערכים שיוכנסו לתוך העמודות בטבלה החדשה:

```
CREATE TABLE Employees
(
    ID      int  AUTO_INCREMENT,
    Fname   text(10),
    Lname   text(20),
    Gender  char(1) DEFAULT 'F',
    Salary  float NOT NULL,
    Rank ENUM('a', 'b', 'c') char set binary,
    Dept    int,
    PRIMARY KEY (ID),
    FOREIGN KEY (Dept) REFERENCES Department(Dept)
);
```

מספר אוטומטי

קביעת סט ערכים שהתא יכול לקבל (אם יוכנס ערך אחר תתקבל הודעת שגיאה) – זמין ב ORACLE

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

פקודות DDL

פקודות DDL מאפיינות את היחסים בבסיס הנתונים ע"י מספר פקודות:

- **יצירת בסיס נתונים** / בחירת בסיס נתונים קיים מתוך רשימה
- **יצירת סכמה** חדשה בבסיס הנתונים (תבנית, תחומי ערכים ואילוצים)
- **עדכון ומחיקת סכמה** קיימת מבסיס הנתונים
- **טבלאות זמניות**
- **הגדרת אינדקס** עבור טבלה
- **הגדרת תצפית** בבסיס הנתונים

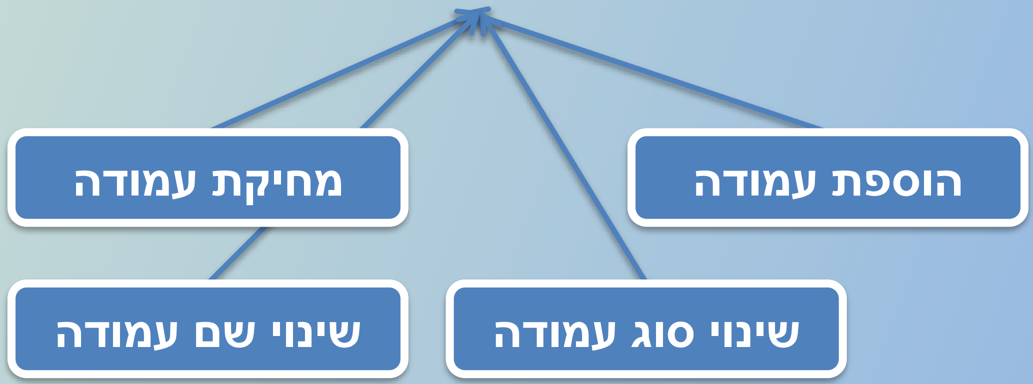
בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

עדכון סכמה

לעיתים בשלב בניית הטבלאות, איננו מודעים עדיין לכל האילוצים, ולכן יתכנו מקרים בהם נצטרך לבצע שינויים במבנה הטבלאות (הוספת עמודה חסרה או הורדת עמודה מיותרת).

הפתרון לסוגיות אלו הינו שימוש בפקודת ALTER TABLE כאשר פורמט הפקודה הינו:

ALTER TABLE Table_Name **Requested_Change**;



עדכון סכמה (ADD)

<u>1) עדכון טבלה קיימת ע"י הוספת עמודה חדשה:</u>

```
ALTER TABLE TableName
ADD ColumnName  DataType
```

דוגמא להוספת עמודת אימייל ותאריך יום הולדת לטבלת סטודנטים:

```
ALTER TABLE Students ADD
Email text,
BirthDate date
```


בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

עדכון סכמה (ADD)

קיימת אפשרות מתקדמת יותר שבה **ניתן לקבוע את מיקום העמודה החדשה** שתתווסף לטבלה.

ניתן לקבוע אותה להיות ראשונה או לחלופין להיות מייד לאחר עמודה מסויימת (ברירת המחדל היא הוספת העמודה כעמודה האחרונה).

<u>מבנה הפקודה המורחבת הינו:</u>

```
ALTER TABLE TableName
ADD ColumnName  DataType  [First / After Column]
```

העמודה החדשה תהיה העמודה הראשונה בטבלה

העמודה החדשה תופיע מיד לאחר עמודה X שתוגדר

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

עדכון סכמה (ADD)

דוגמא להוספת עמודת אימייל שתהיה העמודה הראשונה בטבלת סטודנטים:

```
ALTER TABLE Students ADD
Email text First
```

דוגמא להוספת עמודת תאריך לידה שתהיה העמודה שתופיע מיד לאחר עמודת שם הסטודנט:

```
ALTER TABLE Students ADD
BirthDate date After StudentName
```


בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

עדכון סכמה (DROP)

<u>2) עדכון טבלה קיימת ע"י מחיקת עמודה קיימת:</u>

```
ALTER TABLE TableName
DROP  ColumnName
```

דוגמא למחיקת עמודת אימייל מטבלת סטודנטים:

```
ALTER TABLE Students
DROP  Email
```


בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

עדכון סכמה (MODIFY)

<u>3) עדכון טבלה קיימת ע"י שינוי סוג העמודה הקיימת:</u>

בהנחה וקיימת לנו עמודת משכורת שהוגדרה בטעות להיות משתנה INT (ז"א מספר שלם ללא נקודה עשרונית) נוכל לעדכן את סוג העמודה.

<u>מבנה הפקודה:</u>

```
ALTER TABLE TableName
Modify ColumnName   NewType
```

עדכון של סוג ה TYPE של עמודת המשכורת (עמודה קיימת):

```
ALTER TABLE Students
Modify  Salary float
```

ב MS-SQL פקודת שינוי סוג העמודה תהיה Alter ולא Modify

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

עדכון סכמה (CHANGE)

4) עדכון טבלה קיימת ע"י שינוי שם העמודה הקיימת:

ניתן לשנות את שם העמודה של עמודה קיימת בטבלה (יש צורך להוסיף את סוג העמודה בעלת השם החדש גם אם הסוג זהה).

מבנה הפקודה:

```
ALTER TABLE TableName
Change OldColumnName  NewColumnName  NewType
```

עדכון שם העמודה מ Name להיות StudentName מסוג TEXT:

```
ALTER TABLE Students
Change Name   StudentName text
```


בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

עדכון סכמה (משולב)

ניתן לשלב בין כל 4 סוגי העדכון לסכמה קיימת, לפי הדרישות.

<u>לדוגמא:</u>

```
ALTER TABLE  T2
MODIFY a TINYINT NOT NULL, CHANGE b c CHAR(20);
```

<u>פקודה זו תבצע את השינויים הבאים בטבלה T2:</u>

- תשנה את עמודה a להיות שדה חובה (not null) מסוג tinyint
- תשנה את שם העמודה b להיות c כמחרוזת בעלת 20 תווים.

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

פקודות DDL

פקודות DDL מאפיינות את היחסים בבסיס הנתונים ע"י מספר פקודות:

- **יצירת בסיס נתונים** / בחירת בסיס נתונים קיים מתוך רשימה
- **יצירת סכמה** חדשה בבסיס הנתונים (תבנית, תחומי ערכים ואילוצים)
- **עדכון ומחיקת סכמה** קיימת מבסיס הנתונים
- **טבלאות זמניות**
- **הגדרת אינדקס** עבור טבלה
- **הגדרת תצפית** בבסיס הנתונים

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

יצירת טבלה זמנית

ניתן להשתמש במילה Temporary (טבלה זמנית) בעת יצירת טבלה.

טבלה זמנית גלויה רק לריצה הנוכחית (current session), והיא תימחק באופן אוטומטי כאשר ה-session יסגר (הטבלה הזמנית קיימת אך חבויה מעיני המשתמש) ושימושית במקרים בהם נרצה לבצע חישובי ביניים או שמירת נתונים שלא יהיו רלוונטים בפעם הבאה.

<u>לדוגמא:</u>

```
CREATE Temporary TABLE temp1
(
  id int,
  name text
);
```

ב MS-SQL
משתמשים בתו #
לפני שם הטבלה
לסימון טבלה זמנית

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

יצירת טבלה זמנית

<u>דוגמא להמחשה:</u>

נוכל ליצור טבלה זמנית ולהכניס אליה נתונים של הטיסות שממריאות בשעה הקרובה לחו"ל (כל עוד לא סגרנו את ה-session נתונים אלו ישארו זמינים (במקרה שלנו – סגירת MYSQL תשמיד את הטבלה).

כעת, נוכל ללכת לטבלת נוסעים שרוצים לטוס ולבצע עבורם חיפוש של הטיסות הזמינות בשעה הקרובה ע"י JOIN או כל פעולה אחרת על גבי הטבלה הזמנית (הטבלה הזמנית מתפקדת כטבלה רגילה לכל דבר).

כאשר נסגור את התוכנה בסוף היום וניכנס שוב מחר, המידע שהיה בטבלה הזמנית לא יהיה זמין עבורנו יותר, אך מצד שני, אנו כבר לא זקוקים לו (את מי מעניין איזה טיסות היו זמינות בשעה מסוימת אתמול?)

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

פקודות DDL

פקודות DDL מאפיינות את היחסים בבסיס הנתונים ע"י מספר פקודות:

- **יצירת בסיס נתונים** / בחירת בסיס נתונים קיים מתוך רשימה
- **יצירת סכמה** חדשה בבסיס הנתונים (תבנית, תחומי ערכים ואילוצים)
- **עדכון ומחיקת סכמה** קיימת מבסיס הנתונים
- **טבלאות זמניות**
- **הגדרת אינדקס** עבור טבלה
- **הגדרת תצפית** בבסיס הנתונים

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

אינדקס

רכיב מסוג DDL (מבנה נתונים) המאפשר גישה מהירה למבנה נתונים אחר לפי שדה/ות מסוים/ים ומשמש **להשגת מהירות אופטימאלית** בנגישות למידע.

דוגמא: A הוא אינדקס של טבלה T לפי שדה X

- אינדקס לטבלת סטודנטים לפי מספר תעודת זהות
- אינדקס לטבלת קורסים לפי שם קורס
- אינדקס לטבלת סטודנטים לפי שם משפחה ושם פרטי

אינדקס לרוב ייוצג בצורת "עץ" או Hash Table ובכך יאפשר גישה מהירה יותר לנתונים, ז"א במקום N חיפושים יבצע חיפוש רק על $Log_2(N)$ ערכים (שיפור בסדר גודל). להלן המחשה של היתרון בשימוש באינדקס:

- *חיפוש של מיליון רשומות ללא אינדקס יקח בממוצע 500,000 פעולות*
- *חיפוש של מיליון רשומות כאשר קיים אינדקס יקח בממוצע 20 פעולות*

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

אינדקס

דוגמא להמחשה: נתונה השאילתא הבאה שמטרתה לשלוף רשימת ציונים בקורס 'מתמטיקה'

Courses			
CourseNumber	CourseName	StudentID	Grade
289	DB	111	96
281	Algo	111	85
283	Math	222	78

```
SELECT   Grade
FROM     Courses
WHERE    CourseName = 'Math'
```

שאלה על שדה שאינו מפתח תיקח זמן רב יותר

ניתן לראות שבטבלת הקורסים ישנם 3 קורסים (שמות הקורסים מסודרים בהתאם לסדר הכנסתם ליחס) ולכן על מנת למצוא את הרשומה העונה לתנאי במקרה זה, נצטרך לעבור על כל רשומות הטבלה.

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

שימוש באינדקס

בעיה: מה היה קורה אילו הטבלה הייתה מכילה 10,000 רשומות ? היינו צריכים לעבור סדרתית על כל הרשומות (לא יעיל ולוקח הרבה מאוד זמן)

פתרון: שימוש באינדקס

פקודת האינדקס הינה טבלת מידע (טבלת ביניים) המכילה קישורים היוצרים מיפוי של נתוני הטבלה ומיקומם בזיכרון.

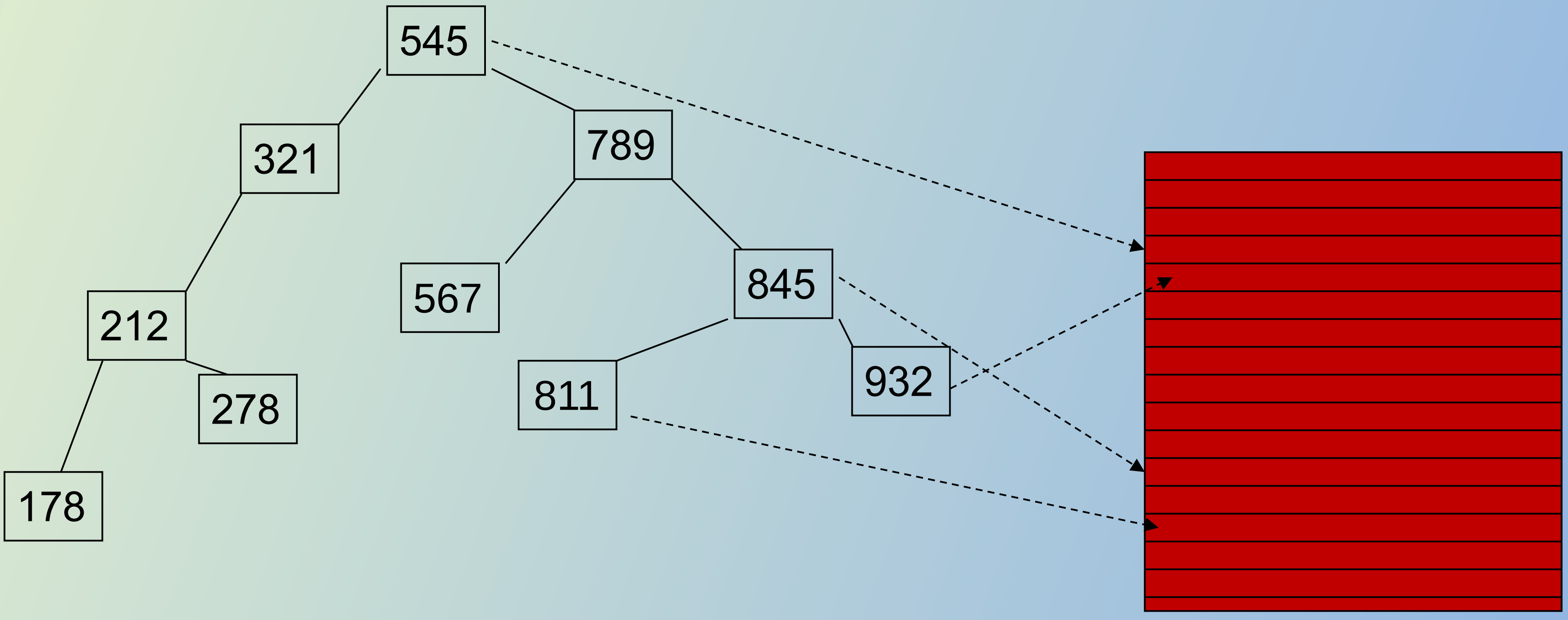
ע"י שימוש בטבלת מידע זו (שלרוב תעבור פעולת מיון) נוכל להגיע לרשומות רלוונטיות ע"י מספר חיפושים מצומצם ומבלי לסרוק את כל הרשומות בטבלה.

ללא אינדקסים, בכל פעם שמשתמש היה בוחר סט של שורות מתוך הטבלה, היה צורך לסרוק את כל הטבלה כדי להשלים את בקשת המשתמש. באופן ברור התוצאה הייתה ביצועים נמוכים, במיוחד כאשר מדובר בטבלאות גדולות.

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

אינדקס – תאור פיזי

אוסף הרשומות (באדום) מייצג טבלת נתונים רגילה. כעת בהנחה ונפעיל אינדקס על עמודה מסוימת אזי יהיו קיימים תאים בזיכרון שיצביעו על הרשומות בטבלה המקורית ויאפשרו חיפוש מהיר יותר בהנחה ושאילתת החיפוש תתבצע על העמודה שעליה הוגדר האינדקס.



בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

אינדקס

<u>בניית האינדקס</u> נעשית ע"י ה DBMS וגוזלת משאבים בעת פעולת היצירה עצמה

שם האינדקס החדש

CREATE INDEX index_name
ON table_name (col_1, col_2, …, col_N)

שם הטבלה שעליה נרצה להוסיף אינדקס חדש

שם העמודות בטבלה שיהוו את האינדקס

כעת, כשנשלוף מידע מטבלת קורסים ובתוך תנאי ה Where נרצה להתייחס לעמודת CourseName שאינה מהווה מפתח בטבלת קורסים, כמות הזמן שייקח לסרוק את עמודה זו יהיה נמוך משמעותית (בסדרי גודל) אם עמודת CourseName תוגדר כאינדקס.

יצירת ומחיקת אינדקס

בניית אינדקס עבור שם הקורס בטבלת קורסים:

```
CREATE INDEX CourseNameIndex
ON Courses (CourseName)
```

לאור העובדה שמטרת האינדקס הינה שיפור ביצועים אזי מומלץ לשים אינדקסים על עמודות שכמות החיפושים עליהם גבוהה.

אז למה לא להגדיר את כל העמודות כאינדקסים ?

TRADEOFF - חשוב להבין שביצוע פעולת Update לטבלה הכוללת אינדקסים תיקח זמן רב יותר מאשר ביצוע עדכון לטבלה ללא אינדקסים וזאת בשל העובדה שיש צורך לעדכן גם את האינדקסים עצמם בעת עדכון הטבלה.

מחיקת אינדקס:

```
DROP INDEX CourseNameIndex
```


בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

פקודות DDL

פקודות DDL מאפיינות את היחסים בבסיס הנתונים ע"י מספר פקודות:

- **יצירת בסיס נתונים** / בחירת בסיס נתונים קיים מתוך רשימה
- **יצירת סכמה** חדשה בבסיס הנתונים (תבנית, תחומי ערכים ואילוצים)
- **עדכון ומחיקת סכמה** קיימת מבסיס הנתונים
- **טבלאות זמניות**
- **הגדרת אינדקס** עבור טבלה
- **הגדרת תצפית** בבסיס הנתונים

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

תצפית

תצפית (VIEW) היא **טבלה וירטואלית** (טבלה מדומה), שניגשים אליה לעיתים קרובות, למרות שאיננה קיימת באופן פיסי בבסיס הנתונים.

שימושים:

- כאשר רוצים **להסתיר שורות/עמודות** ניצור תצפית ללא המידע ה"סודי" (לדוגמא: נרצה למנוע ממנהלת החשבונות לראות את המשכורות של העובדים הבכירים בחברה).
- כאשר עולה צורך, **מטעמי נוחות**, ליצור אוסף יחסים לשליפה שאינו חלק מהתבנית התפיסתית (לדוגמא: על פרטי לקוח ופרטי חשבונותיו, ניצור תצפית שהיא הצירוף הטבעי של deposit ו customer).
- כאשר נרצה להשתמש ליצור "**הרשאות**" לטבלה אחת ברמות גישה שונות.

הערה: העובדה שתצפיות לא בהכרח ממומשות באופן פיזי תטיל מגבלות על אפשרויות העדכון של התצפיות (אך לא על האפשרויות לבצע שאילתות עליהן).

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

בניית תצפיות

<u>שאלה</u>: איזה יחסים מוגדרים כתצפיות ואיזה לא ?

כל יחס הנוצר לנוחיות המשתמש ואינו חלק מהתבנית התפיסתית של בסיס הנתונים מכונה תצפית או יחס מדומה.

התצפית מוגדרת בבסיס הנתונים כטבלה וירטואלית, כאשר בפועל נשמרת **<u>השאילתא עצמה</u>** בבסיס הנתונים ולא טבלת המידע (התצפית אינה מכילה נתונים בפועל).

<u>שאלה</u>: באיזה שלב מחושבת התצפית ?

התצפית מחושבת כל פעם מחדש (לפי הצורך) ומשום כך מתייחסים אליה כאל יחס מדומה, אך זה בעצם היתרון שלה.

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

יצירת ומחיקת תצפיות

פקודת Create View מאפשרת להגדיר תצפיות לפי המבנה הבא:

CREATE VIEW V
AS
(
Query
)

ב DB **נשמרת השאילתא עצמה** !

פקודת Drop View מוחקת את התצפית מבסיס הנתונים

DROP VIEW V

__הערה__: הסוגריים שעוטפים את השאילתא הם לנוחות בלבד ולא חובה לרשום אותם.

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

בניית תצפיות

דוגמא 35: צרו תצפית של הלקוחות הגרים בחיפה

```
CREATE VIEW Haifa-Customers
AS
(
    SELECT *
    FROM   Customer
    WHERE customer-city='Haifa'
)
```

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

```
SELECT *
FROM   Haifa-Customers
```

טבלה מדומה

Customer		
customer-name	street	customer-city
Tamir	Allenby	Haifa
Even	Allenby	Haifa

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

יצירת ומחיקת תצפיות

דוגמא 36: צרו תצפית המציגה את שמות כל הלקוחות ושמות הסניפים בהם ללקוחות אלו קיים חשבון מסוג הלוואה או חסכון.

```
CREATE VIEW Customers-Accounts
AS
        ((
            SELECT branch-name, customer-name
            FROM    Deposit
        )
        UNION
        (
            SELECT branch-name, customer-name
            FROM    Borrow
        ))
```

דוגמא 37: מחיקת מבנה התצפית:

```
DROP VIEW  Customers-Accounts
```


בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

יצירת תצפיות עם תכונות חדשות

דוגמא 38: צרו תצפית של הסניפים והיתרה הכוללת בהם

```
CREATE VIEW Branch-Total (branch-name, total-balance)
AS
(
        SELECT      branch-name, sum(balance)
        FROM        deposit
        GROUP BY    branch-name
)
```

דוגמא 39: הציגו את היתרה הכוללת בסניף Aviv

```
SELECT    *
FROM      Branch-Total
WHERE     branch-name = 'Aviv'
```


בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

תצפית – חריגת הוספה/עדכון

תצפית מהווה טבלה וירטואלית אך משמשת כמעט כטבלה רגילה לפעולות השליפה והעדכון. לאור העובדה שתצפית מציגה לרוב רק חלק מטבלה נתונה, אזי ביצוע עדכון בתצפית נתונה עלול לגרום לחריגת עדכון.

דוגמא 40: צרו תצפית המציגה את שם הסניף, מספר הלוואה ושם הלווה.

```
CREATE VIEW Loan-info
AS
(
        SELECT      branch-name, loan-number, customer-name
        FROM        Borrow
        ORDER BY    branch-name
)
```

כעת הוסיפו את ה-ח-יה הבאה:

```
INSERT INTO Loan-info
VALUES          ('Aviv', 90, 'Levy')
```

שאלה: מה יהיה ערך סכום ההלוואה בשורה המתאימה בטבלת Borrow?

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

ערכי NULL

פתרון 1 – לדחות את הוספת השורה ולהוציא הודע שגיאה.
פתרון 2 – להוסיף את השורה בתוספת **ערכי דמה לתכונות ללא ערכים**.

```
INSERT INTO Loan-info
VALUES        ('Aviv', 90, 'Levy', NULL)
```

הערות:

1) בביצוע פעולת הקבצה (דוגמת sum) הערך הנ"ל לא יכנס לחישוב.
2) ניתן לבצע שאילתא לאיתור ערכי NULL בתכונה מסויימת.

דוגמא 41: מצא את שמות הלקוחות שסכום ההלוואה שלהם אינו ידוע.

```
SELECT    customer-name
FROM      Borrow
WHERE     amount IS NULL
```


בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

השוני בין טבלה רגילה למדומה

- טבלה מדומה **אינה מכילה ערכים** בתוכה (לכן נקראת טבלה מדומה).
- התצפית **שומרת את השאילתא** עצמה, בעוד טבלה שומרת בתוכה את נתוני הרשומות.
- ניתן ליצור "מידור" זאת אומרת **הגדרת רמות הרשאות שונות** בין אנשים שונים על אותה טבלה ע"י בניית תצפיות שונות לכל אחד מהם, אנו בעצם יוצרים רמה גבוהה של בקרה ואבטחת מידע ובכך מונעים גישה לנתונים ולטבלאות עצמן ומאפשרים גישה רק לשכבת התצפית.

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

יצירת תצפית למול יצירת טבלה

Grades			
StudentID	CourseID	TestGrade	ExeGrade
1	281	82	85
2	281	67	84
3	281	80	80
4	281	80	90

```
CREATE TABLE Talmidim1( ID, AvgGrade) AS
(
    SELECT StudentID, (TGrade+EGrade)/2
    FROM Grades
    WHERE StudentID in (3,4)
)
```

הפעלנו פונקציה בשאילתא ולכן נצטרך להגדיר את שם העמודה שתיוצר בתצפית

```
CREATE VIEW Talmidim2( ID, AvgGrade) AS
(
    SELECT StudentID, (TGrade+EGrade)/2
    FROM Grades
    WHERE StudentID in (3,4)
)
```

SELECT * FROM Talmidim1

ID	AvgGrade
3	75
4	85

SELECT * FROM Talmidim2

ID	AvgGrade
3	75
4	80

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

עבודה עם תצפית

- התצפית תפסיק לפעול אם בוצעה **<u>מחיקה של הטבלה</u>** עלייה היא בנויה.

- התצפית תפסיק לפעול אם **<u>שונו שמות העמודות בטבלה</u>** עלייה היא בנויה. ז"א שהתצפית מושפעת משינויי סכמה אך אינה מושפעת משינויי DATA.

- <u>ניתן להוסיף/לעדכן/למחוק שורות מתוך תצפית קיימת בתנאים הבאים:</u>
  - בתנאי שהתצפית לא מכילה פונקצית הקבצה (וכנגזרת מכך גם לא Group By או Having)
  - בתנאי שהתצפית בנויה על **<u>טבלה אחת בלבד</u>** (ז"א תצפית ללא פעולות צירוף, איחוד, חיתוך וכד')
  - בתנאי שעמודות המפתח הראשי כלולות בה.
  - בתנאי שאין בה פקודת DISTINCT או הופעת עמודה בודדת מספר פעמים ב SELECT.

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

עבודה עם תצפית

- ניתן לבצע **<u>הרכבה של תצפיות</u>** (יצירת תצפית השולפת מתצפית אחרת).
- ניתן להשתמש בתצפית **<u>במהלך פקודת JOIN</u>** כאילו הייתה טבלה רגילה.
- הרעיון העומד מאחורי תצפית טובה הוא **<u>שניתן לשלוף מידע רלוונטי באופן מחזורי</u>** מבלי להתייחס למורכבות בסיס הנתונים הקיים ומבלי צורך לחזור ולכתוב את השאילתא בכל פעם שנרצה לשלוף.
- תמיד **<u>ניתן להוסיף עוד תנאים לתצפית קיימת</u>** בכדי לקבל תשובה ספציפית יותר וע"י כך בעצם נוכל להתאים תצפיות למנהלים שונים וללקוחות שונים. ע"י שימוש בהרכבת תצפיות כך גם נוכל ליצור מידור פנים אירגוני ויכולות אבטחה טובים יותר.
- **<u>חשוב לזכור</u>**: שינוי ערכים דרך VIEW יגרור שינויים של טבלאות הבסיס

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

אופרטור בדיקה לתצפיות

בעת יצירת תצפית ניתן להוסיף אופרטור בדיקה מיד לאחר כתיבת השאילתא.

אופרטור **With Check Option** לא יאפשר לבצע פעולות DML אשר מפרות את תנאי הסינון (Where) של השאילתא שהוגדרה בתצפית

דוגמא: יצירת תצפית של סטודנטים מצטיינים (מעל ציון 90)

```
CREATE VIEW TopStudents
AS
(
        SELECT *
        FROM Students
        WHERE Grade > 90
)
WITH CHECK OPTION
```

במקרה זה בהנחה ופקודת Insert תרצה להוסיף שורה חדשה בטבלת סטודנטים (דרך התצפית) אזי האופרטור יבדוק שאכן הסטודנט החדש הוא בעל ציון של מעל ל-90, בהנחה ולא - תתקבל הודעת שגיאה של אי עמידה בתנאי

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

אופרטורים לתצפיות

<u>במקרה זה לא ניתן יהיה לבצע את הפעולות הבאות:</u>

- להכניס סטודנט עם ציון מתחת ל-90
- לעדכן ציון של סטודנט קיים להיות מתחת ל-90

כל שאר העדכונים (שם סטודנט, כתובת) יתאפשרו ללא הפרעה

מכיוון שהתנאי המופיע ב WHERE אינו משפיע על שדה כתובת, אזי גם אם בתצפית היה מוגדר **With Check Option** היינו יכולים לבצע את השינוי:

```
UPDATE    TopStudents
SET       StudentCity = 'Tel-Aviv'
WHERE     StudentID =123
```

ניתן לבצע את העדכון שכן לא סותר את תנאי ה WHERE

אך לא ניתן היה לבצע את השינוי הבא:

```
UPDATE    TopStudents
SET       Grade = 83
WHERE     StudentID =123
```

עדכון הציון ל 83 סותר את תנאי ה WHERE שמאפשר רק ציון מעל 90

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

עדכון של תצפית

עדכון של תצפית בעצם מעדכן את השאילתא הנשמרת בתוך המבנה.

ניתן לבצע עדכון של תצפית ע"י פקודת DDL מתאימה – ALTER:

```
ALTER VIEW ViewName
AS
(
    Query
)
```

מחיקת תצפית תתבצע ע"י:

```
DROP VIEW ViewName
```


בסיסי נתונים – קורס מתקדם – רואי זרחיה ©

סיכום ביניים – פקודות DML ו DDL

תרגיל
כיתה 2

בסיסי נתונים – קורס מתקדם – רואי זרחיה ©