הרצאה מספר 13

פקודות DDL (המשך):

חזרה על פקודות שיעור קודם:

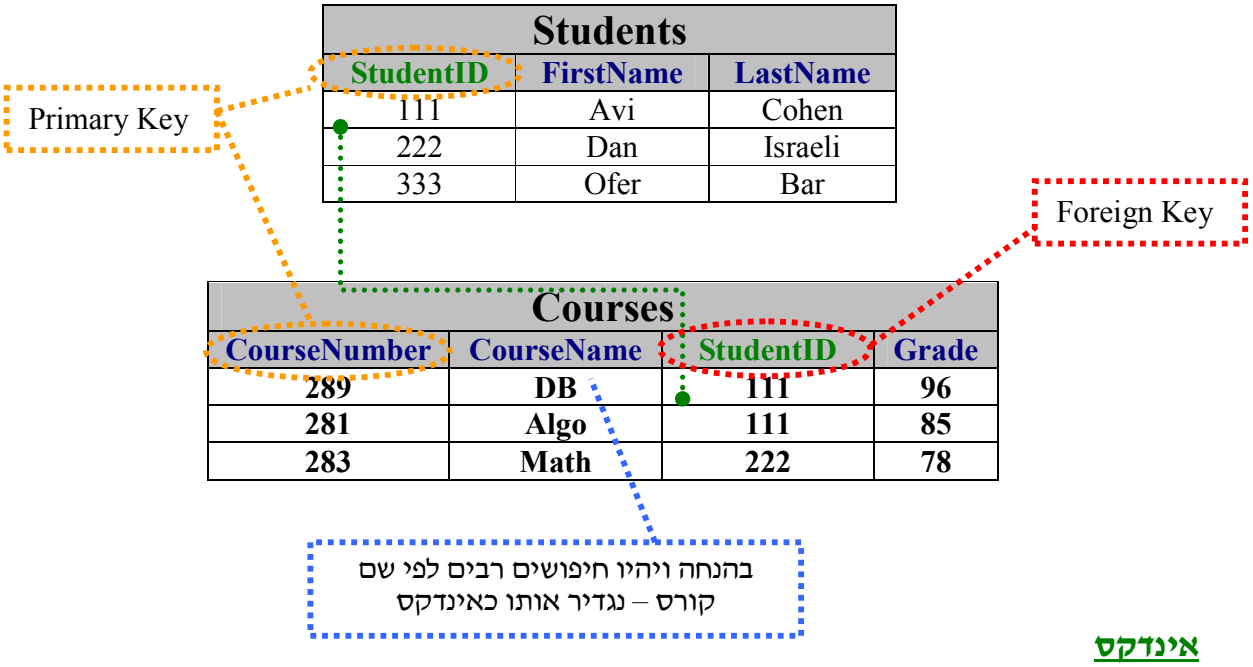
1) **מפתח ראשי** (Primary Key) – שדה או אוסף שדות המזהים באופן מינימאלי וחח"ע רשומה בטבלה.
2) **מפתח זר** (חיצוני – Foreign Key) – שדה או אוסף שדות המצביעים על מפתח ראשי בטבלה אחרת.
3) **אינדקס** – רכיב (שקוף למשתמש) המאפשר להשיג מהירות אופטימאלית בנגישות למידע הקיים.
4) **הגבלות/אילוצים** (Constraints)
5) **עדכון סכמה קיימת** (Alter)

הפקודות להיום:

6) **מחיקת סכמה קיימת** (Drop)
7) **שילוב תתי שאילתות בפקודות** DDL
8) **תצפיות** (Views)

רענון (סעיפים 1-5)

מפתח ראשי ומפתח זר



Students

StudentID	FirstName	LastName
111	Avi	Cohen
222	Dan	Israeli
333	Ofer	Bar

Courses

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

אינדקס

אינדקס – רכיב המשמש להשגת מהירות אופטימאלית בנגישות למידע. האינדקס מהווה טבלה מידע המאפשרת גישה ישירה לרשומות מבלי שיהיה צורך לסרוק את כל הרשומות בטבלה - מהלך מהיר משמעותית. האינדקסים יכולים להוריד משמעותית את כמות המידע שתטען לזיכרון במטרה להחזיר את התשובה לשאילתא שהורצה

TRADEOFF בין עדכון/מחיקה/הכנסת נתונים לביצוע חיפושים עבור עמודה שהוגדר עליה אינדקס.

6) מחיקת סכמה של טבלה קיימת ע"י פקודת Drop

פקודה זו מאפשרת לבצע **מחיקה של טבלה שלמה** בפקודה אחת קצרה ומהירה.

יש לקחת בחשבון שבעת הפעלת פקודת מחיקה לטבלה שלמה, ילך ה DBMS ויבדוק האם קיימות הגבלות/אילוצים של עמודות מסוימות בסכמת הטבלה.

```
DROP TABLE Workers
```

הערה: ניתן לבצע מחיקה של עמודות בסכמה ו/או את הסכמה עצמה ע"י שימוש באופציות הקיימות בחלון Edit Table.

סיכום סיווג הפקודות:

DML	DDL	פקודה
Insert	Create	הוספה
Update	Alter	עדכון
Delete	Drop	מחיקה

7) דוגמא: שילוב תתי-שאילתות בפקודת DDL

למדנו שניתן לרשום פקודות היכולות ליצור/לשנות/למחוק/לעדכן טבלאות. נוכל על אותו משקל לנצל ידע זה בשימוש של תתי-שאילתות שלמדנו בשיעורים קודמים.

אם אנו מעוניינים להעתיק תוכן (או חלק מהתוכן) של טבלה מסוימת לתוך טבלה חדשה, נוכל לבצע את סט הפעולות הבא:

מבנה הפקודה של יצירת טבלה:

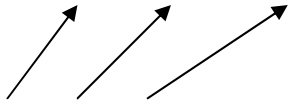
```
CREATE TABLE Students
(
        SId INT,
        SName TEXT,
        SSalary INT
);
```

מבנה הפקודה של תת שאילתא:

```
SELECT      Id, Name, Salary
From        Stu
Where       Salary > (Select AVG(Salary) From Stu )
```

שילוב 2 הפקודות: בניית טבלה חדשה בעזרת תת-שאילתא:

```
CREATE TABLE Students (SId, SName, SSalary)
AS
(
        SELECT      Id, Name, Salary
        From        Stu
        Where       Salary > (Select AVG(Salary) From Stu )
);
```



העמודות בטבלה החדשה יקבלו את ה type של העמודות הנשלפות מהטבלה הפנימית ותוכן השאילתא שהורצה יכנס ישירות לתוך הטבלה החדשה

8) תצפיות - Views

תצפית הינה מבנה מסוג טבלה וירטואלית המכיל שאילתא להרצה (היא אינה מהווה טבלה עצמאית).
ניתן לראות בתצפית כדרך להגדרת טבלה שניגשים אליה לעתים תכופות, למרות שאינה קיימת פיסית.

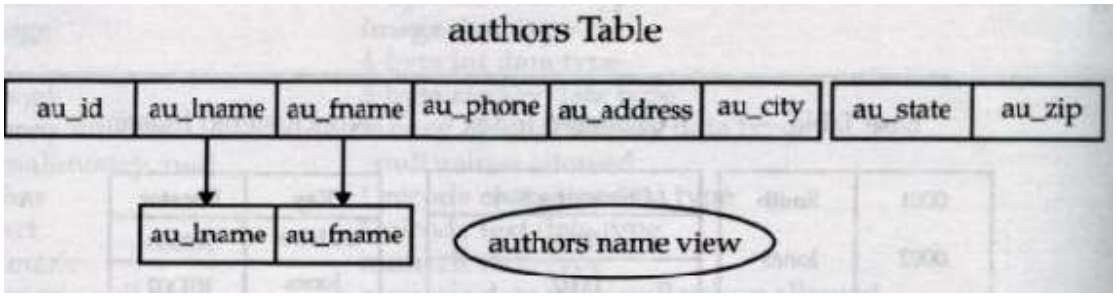
למה אנו צריכים תצפית ?

בוא נניח שכתבנו שאילתא בת 20 שורות (הכוללת צירוף של מספר טבלאות) ואנו צריכים **להריץ אותה בכל בוקר** שאנו מגיעים לעבודה בכדי לקבל דו"ח מסוים. על מנת למנוע מצב שבכל בוקר נצטרך לרשום את אותם 20 שורות ולהריץ שוב ושוב (כי הרי אנו יכולים לשמור סכמות או נתונים בתוך טבלה קיימת) קיימת **אפשרות לשמור את השאילתא** עצמה (לא את התוצאה שלה ולא את הסכמה אלא את פקודת ה SQL עצמה).

שמירת השאילתא תאפשר לנו בכל זמן נתון להריץ את התצפית (כל ה 20 שורות) ולקבל תוצאה מיידית ועדכנית, ז"א שבכל זמן נתון בעצם נוכל **לקבל דגימה מעודכנת של מסד הנתונים** – תהיה לנו ביד תצפית שתוכל לומר לנו בכל נקודת זמן מהם העדכונים שבוצעו.

תצפית יכולה לשמש למספר פונקציות:

- ניתן להציג שאילתות לתצפית בדומה לטבלאות רגילה
- ניתן לעדכן דרך התצפית נתונים, כל עוד העדכון משפיע על טבלה יחידה
- לרוב נשתמש בתצפית במטרה לשלוף סט ממוקד של שורות או עמודות מתוך טבלה, איחוד עמודות מטבלאות שונות וכדי



מראה כיצד ניתן באמצעות תצפית לסנן רק את שמות המחברים מתוך בסיס הנתונים של חברת ההוצאה לאור

מסד נתונים לדוגמא

נגדיר את ה DB שעליו נבצע את הדוגמאות עבור עבודה למול תצפיות בשיעור זה:

Students			
StudentID	**Name**	**City**	**Department**
1	Avi Cohen	Ramat-Gan	Computers
2	Avi Levin	Ramat-Gan	Math
3	Ben Levi	Tel-Aviv	Math
4	Chen Levin	Tel-Aviv	Computers
5	Debi Dvir	Givatiim	Math
6	Zvi Kaner	Givatiim	Computers
7	Haim Itzhak	Jerusalem	Computers

מבנה פקודת VIEW:

את התצפית מגדירים בעזרת הפקודה create view המגדירה את שם התצפית (שם הטבלה הוירטואלית), רשימת עמודות, וכן שאילתה המגדירה את תוכן התצפית (אם אף אחת מהעמודות בתצפית אינה תוצאה של הפעלת פונקציות או אופרטורים אריתמטיים, אין צורך להגדיר את שמות התכונות בתצפית, ושמות התכונות יהיו זהים לשמות התכונות בטבלאות המקוריות המגדירות את התצפית).

CREATE VIEW College_Students **AS**
(

שאילתא כלשהיא

)

דוגמא לצורך ושימוש ב VIEW:

דוגמא: הצג את שמות הסטודנטים ואת עיר מגוריהם עבור סטודנטים שלומדים מחשבים:

```
SELECT      Name, City
FROM        Students S
WHERE       S.Department = "computers"
```

Name	City
Avi Cohen	Ramat-Gan
Chen Levin	Tel-Aviv
Zvi Kaner	Givatiim
Haim Itzhak	Jerusalem

בשלב א' ביצענו שליפה פשוטה, אך כעת בואו נניח שראש המחלקה מריץ **כל יום** את אותה השאילתא ונניח גם שלפעמים הוא צריך להוסיף מספר תנאים נוספים על שאילתא זו – לשם כך קיימת לנו אפשרות לשמור טבלה זו כתצפית (כטבלה מדומה).

בפועל **נשמור את פקודת ה SQL ולא את הנתונים שהתקבלו**, וע"י כך תשמר בזיכרון השאילתא עצמה - מה שיאפשר לנו להריץ אותה כל פעם ולא יהיה צורך להקליד אותה מחדש.

במקרה זה נשמור את השאילתא כטבלה מדומה תחת השם CompStudents בצורה הבאה:

```
CREATE VIEW CompStudents AS
(
        SELECT      Name, City
        FROM        Students S
        WHERE       S.Department = "computers"
)
```

כעת קיים לנו VIEW בשם CompStudents שע"י כל הרצה שלו נקבל את רשימת הסטודנטים למחשבים.

על מנת לקבל את הסטודנטים הלומדים מחשבים ניתן מעכשיו להריץ את השאילתא הבאה:

```
SELECT      *
FROM        CompStudents
```

שלב ב': נניח שכעת ראש המחלקה רוצה לסנן **<u>מתוך רשימה זו</u>** רק את הסטודנטים שגרים בתל אביב (ז"א את אלו שלומדים מחשבים וגם גרים בתל אביב), ניתן לבצע פעולה זאת ע"י בניית השאילתא הבאה:

```
SELECT      *
FROM        CompStudents
WHERE       City = "Tel-Aviv";
```

ובעקבות זאת (למרות שרשמנו * ב SELECT הראשי) תתקבל התוצאה הבאה:

Name	City
Chen Levin	Tel-Aviv

← בפועל, יוצגו 2 העמודות מה VIEW המתאימות לתנאי שהתווסף לשליפה.

<u>מאחורי הקלעים בעצם מתבצעת השאילתא הבאה:</u>

```
SELECT      Name, City
FROM        Students
WHERE       Department = "computers"
AND
City = "Tel-Aviv";
```

```
SELECT      *
FROM        CompStudents

WHERE       City = "Tel-Aviv";
```

← מכאן נוכל להבין את השלבים של בניית טבלה מדומה:

1) הגדרת טבלה מדומה חדשה (הגדרת השאילתא ויצירתה ב DB).
2) הפעלת שאילתא רגילה הפונה כחלק מהשליפה לטבלה המדומה.
3) ה DBMS יוצר את השאילתא המשולבת (כמו שראינו בדוגמא האחרונה).
4) מופעלת השאילתא המשולבת ומוחזרת התוצאה.

קווי דמיון בין יצירת טבלה מדומה (חדשה) ליצירת טבלה רגילה (חדשה):

Grades			
StudentID	**CourseID**	**TestGrade**	**ExeGrade**
1	281	82	85
2	281	67	84
3	281	70	80
4	281	80	90

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

```
CREATE VIEW Talmidim2( 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

Select * From **Talmidim1**

ID	AvgGrade
3	75
4	85

Select * From **Talmidim2**

ID	AvgGrade
3	75
4	85

השוני בין טבלה מדומה לטבלה רגילה הוא בכך:

1) טבלה מדומה אינה מכילה ערכים בתוכה (לכן נקראת טבלה מדומה).
2) התצפית שומרת את השאילתא עצמה, בעוד טבלה שומרת בתוכה את נתוני הרשומות.
3) ניתן ליצור "מידור" בין אנשים שונים על אותה טבלה ע"י בניית תצפיות שונות לכל אחד מהם, אנו בעצם יוצרים רמה גבוהה של בקרה ואבטחת מידע: מונע גישה לנתונים ולטבלאות עצמן ומאפשר גישה רק לתצפית.

הרכבה של תצפיות

ניתן להגדיר **הרכבה של תצפיות**, ז״א ניתן להגדיר תצפית הבנויה על בסיס תצפית אחרת:

```
CREATE VIEW BestTalmidim AS
(
    Select *
    From Talmidim
)
```

נוכל להשתמש בתצפית בביצוע JOIN כאילו היא הייתה טבלה רגילה

```
SELECT    *
FROM      Talmidim T , Courses C
WHERE     T.ID = C.ID
```

חוקים שיש לזכור בעבודה עם תצפיות:

1. התצפית תפסיק לפעול אם בוצעה **מחיקה של הטבלה** עלייה היא בנויה.
2. התצפית תפסיק לפעול אם **שונו שמות העמודות בטבלה** עלייה היא בנויה. ז״א שהתצפית מושפעת משינויי סכמה אך אינה מושפעת משינויי DATA.
3. ניתן לעדכן/למחוק שורות מתוך תצפית קיימת בתנאי שהתצפית לא מכילה פונקציית הקבצה ושהיא בנויה על **טבלה אחת בלבד** (ז״א תצפית ללא פעולות צירוף, איחוד, חיתוך וכד׳) ובתנאי שעמודות המפתח הראשי כלולות בה.
4. הרעיון העומד מאחורי תצפית טובה הוא **שניתן לשלוף מידע רלוונטי באופן מחזורי** מבלי להתייחס למורכבות בסיס הנתונים הקיים ומבלי צורך לחזור ולכתוב את השאילתא בכל פעם שנרצה לשלוף.
5. תמיד **ניתן להוסיף עוד תנאים לתצפית** קיימת בכדי לקבל תשובה ספציפית יותר וע״י כך בעצם נוכל להתאים תצפיות למנהלים שונים וללקוחות שונים ע״י שימוש בהרכבת תצפיות וכך גם נוכל ליצור מידור ואבטחה טובים יותר.