

הרצאה מס' 8

נושאים לשיעור זה:

- צירוף חיצוני עם שמירת מידע וערכי NULL
- טיפול בערכי NULL
- השוואה בין 3 הפקודות לסינון נתונים
- ההבדל בין פקודת איחוד לפקודת צירוף
- פקודות DML נוספות
 - הכנסת ערכים – INSERT
 - עדכון ערכים – UPDATE
 - מחיקת רשומות – DELETE

פקודת צירוף עם שמירת מידע וערכי NULL

בשיעור הקודם עברנו משימוש במכפלה קרטזית לשימוש בפקודת צירוף (JOIN).

ע"י שימוש בפקודת ה- JOIN, שפת ה-SQL מאפשרת לנו להשתמש ב-2 פקודות חדשות לחיבור הטבלאות המצורפות, פקודות ON ו-USING שמטרתן לחבר רק את הרשומות "בעלי המשמעות" משתי הטבלאות.

כחלק מחיבור המידע בין 2 הטבלאות, יכול להיווצר מצב של איבוד מידע, לאור כך למדנו שקיימת פקודת צירוף עם שמירת מידע שעבורה יתקבלו ערכי NULL בתאים עבורם אין לנו מידע מלא זמין, אך עדיין המידע החלקי יופיע ולא "ילך לאיבוד" כמו שבמקרה של JOIN פשוט (שאינו שומר מידע).

[תזכורת:](#)

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

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

```
SELECT FirstName, LastName, CourseNumber
FROM Students LEFT OUTER JOIN Courses
USING (StudentID);
```

FirstName	LastName	CourseNumber
Avi	Cohen	289
Avi	Cohen	281
Dan	Israeli	281
Ofer	Bar	Null

טיפול בערכי NULL

א) ערכי Null

לאור העובדה שכאשר נשתמש בפקודת צירוף עם שמירת מידע נקבל ערכי Null בטבלאות התוצאה, נצטרך ללמוד על איך מתייחסים בשאלתא לערכי Null.
על מנת לבדוק אם תא מסוים מכיל ערכי Null נוכל להשתמש באופרטור יעודי לכך ובפקודת ה WHERE לשאול האם שדה מסוים IS NULL, מטרת אופרטור זה היא להחליף את השימוש באופרטור השווה.

לדוגמא: עבור הטבלה שקיבלנו בדוגמא הקודמת:

NewCourses		
FirstName	LastName	CourseNumber
Avi	Cohen	289
Avi	Cohen	281
Dan	Israeli	281
Ofer	Bar	Null

על מנת לשאול על הערך המופיע בשדה CourseNumber בשורה של Ofer נכתוב:

```
SELECT *
FROM NewCourses
WHERE CourseNumber is NULL
```

FirstName	LastName	CourseNumber
Ofer	Bar	Null

* כמו תמיד, נוכל לשאול גם על ההיפוך של תנאי זה ע"י הוספת not: **IS NOT NULL**.
* דוגמא לשימוש שגוי: `FirstName = 'NULL'`.

ב) מניעת ערכי Null בשלב יצירת הטבלה

בתחילת הסמסטר למדנו איך מגדירים סכמה לטבלה חדשה, כעת, עם הידע החדש שצברנו נוכל להבטיח ששדה בטבלה חדשה **לא יוכל לקבל ערכי Null** – ז"א שדות שיוגדרו כשדות חובה החייבים לקבל ערכים משמעותיים (חשוב מאוד עבור נושאים כמו מפתחות שנלמד בעתיד).

```
CREATE TABLE Workers
```

```
(
```

```
    ID int NOT NULL,
```

```
    LastName text NOT NULL,
```

```
    FirstName text,
```

```
    Address text,
```

```
    City text
```

```
)
```

השוואה בין 3 הפקודות לסינון נתונים:

הגענו לשלב בו עברנו על כל פקודות הסינון המרכזיות בשפת ה SQL שמטרתן לסנן נתונים שונים בנקודות שונות בפקודה, כעת על מנת להבין בדיוק את ההבדלים ביניהם, נבהיר מספר נקודות.

חשוב: יש לשים לב להבדלים בין WHERE ל HAVING ול ON (או ל USING) שכן שלושת הפקודות מטרתן היא לסנן שורות מהתוצאה המוחזרת, רק שכל אחד פועל בתנאים ובמקרים שונים.

```
SELECT X, count(Y)
FROM Students S JOIN Courses C
1 ON S.studentID = C.studentID
2 WHERE ( ) AND ( )
GROUP BY X
3 HAVING count(Y) > 1000
ORDER BY X
```

1 פקודת **USING / ON** מגדירה את התנאים לצירוף הטבלאות המופיעות בפקודת ה FROM.

2 פקודת ה **WHERE** מגדירה את התנאים לסינון שורות מהטבלאות הקיימות בפקודת ה FROM, יש לשים לב שפקודת WHERE מגיעה לאחר פקודת ON ולכן אם בוצעה פעולת צירוף בשלב ה FROM, פקודת ה WHERE תסנן נתונים כבר מתוך הטבלה המצורפת.

3 פקודת **HAVING** מגדירה את התנאים לסינון שורות מתוך הטבלה המקובצת שהתקבלה לאחר הפעלת פעולת GROUP BY.

ההבדל בין JOIN ל UNION:

לאור העובדה שכמות הפקודות הנלמדת בקורס הינה רבה, לפעמים בהתחלת הדרך יש בלבול בין שתי הפקודות הבינאריות האלו וזאת כיוון שבשתי הפעולות אנו משלבים מידע משתי טבלאות שונות.

על מנת לחדד את הנושא, נסביר את העיקרון בביצוע כל פעולה:

בפעולת JOIN אנו לוקחים 2 טבלאות ומבצעים מכפלה ביניהם על מנת לקבל טבלה אחת עם סכמה מורחבת המכילה את צירוף המידע לפי עמודות משותפות משתי הטבלאות (בדוגמא זו - צירוף טבלאות קורסים וסטודנטים במטרה לקבל רשימה אחת של כל הסטודנטים בשילוב כל הקורסים אליהם הם רשומים).

$$\square \times \square = \begin{array}{|c|c|} \hline \square & \square \\ \hline \end{array}$$

בפעולת UNION אנו מאחדים 2 טבלאות כאשר התשובה תהיה בעלת אותה סכמה עם אותה עמודות (ללא שורות כפולות) כאשר האיחוד מבוצע על בסיס סכמות תואמות ללא קשר לעמודות משותפות (איחוד רשימת הסטודנטים של בר אילן עם רשימת הסטודנטים של שנקר לרשימת סטודנטים מלאה), כמות השורות תקבע לפי התוכן של הטבלאות.

$$\begin{array}{|c|} \hline \square \\ \hline \end{array} \cup \begin{array}{|c|} \hline \square \\ \hline \end{array} = \begin{array}{|c|} \hline \square \\ \hline \square \\ \hline \end{array}$$

דוגמא להמחשה:

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

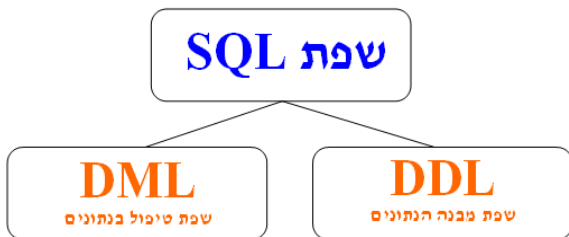
```
SELECT *
FROM R1 JOIN R2
```

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

פקודות DML נוספות:

ראינו ששפת SQL מכילה 2 מרכיבים:

- (א) פקודות DDL – העוסקות במבנה הטבלאות והסכמות.
- (ב) פקודות DML – העוסקות בטיפול בנתונים.



עד עכשיו עסקנו ברכיב ה DML ולמדנו את פקודת ה SELECT, אך כפי שלמדנו בתחילת הסמסטר, קיימות מספק פקודות נוספות שנרצה להשתמש בהן כאשר נעבוד עם נתוני טבלאות.

(א) הכנסת ערכים לתוך טבלה חדשה

למדנו בשיעורים הראשונים, שלאחר יצירת טבלה (פעולת DDL) נוכל להכניס לתוכה ערכים חדשים ע"י שימוש בפקודת Insert Into, קיימות 3 וריאציות להכנסת ערכים לסכמה ריקה:

(א) **וריאציה ראשונה** (שאותה למדנו) היא הגדרת שם הטבלה שאליה נרצה להכניס רשומות/נתונים:

Insert Into TableName
Values (Value1 , Value2 , Value3...)

בפקודה זו הערכים יוכנסו לתוך הטבלה לפי סדר העמודות שהוגדרו בזמן יצירת הסכמה (בפקודת ה CREATE). ניתן לראות שבמקרה זה לא נוכל לדלג על עמודה מסוימת ולהימנע מלהכניס אליה ערכים. הפתרון היחיד שקיים יהיה הכנסת ערכי NULL לשדות אלו ובכך בעצם אנו עוברים להכנסת ערכים בשדה הבא ומשאירים את השדה הנוכחי "ריק".

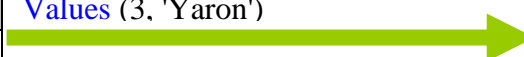
(ב) **וריאציה שנייה** כוללת את הגדרת שם הטבלה ושמות העמודות הספציפיים (לא חובה) שאליהם נרצה להכניס את הערכים, כאשר שאר השדות בשורה (שעמודותיהם לא הוגדרו) יישארו ריקים:

Insert Into TableName (Column1, column2, column3...)
Values (Value1 , Value2 , Value3...)

דוגמא: בהנחה ונרצה לקלוט עובד חדש לחברה, ז"א בטבלת עובדים (Employees) נרצה להוסיף רשומת עובד חדשה כאשר עדיין לא ידועה המשכורת אותה הוא יקבל, נוכל לכתוב את פקודת Insert הבאה ובשלב מאוחר יותר להשתמש בפקודת עדכון עבור שדה המשכורת:

INSERT INTO Employees (**ID, Name**)
Values (3, 'Yaron')

ID	Name	Salary
1	Roei	4000
2	Yaniv	3000



ID	Name	Salary
1	Roei	4000
2	Yaniv	3000
3	Yaron	Null

הערה: חשוב לשים לב שהערכים יוכנסו לתוך שמות העמודות שהוגדרו בפקודת ה Insert, כאשר במצב בו נרשמו רק חלק משמות העמודות הקיימות בטבלה, יישארו ערכי התאים של העמודות שלא צוינו – ריקים (ז"א יקבלו את הערך NULL).

חשוב לציין, שהכנסת ערכים לא חייבת להתבצע רק בשלב הראשוני של בניית הטבלה, אלא יכולה להתבצע בכל זמן נתון. מכאן ניתן להסיק שנוכל להוסיף נתונים לטבלה קיימת מתי שיהיה בכך צורך.

ג) **וריאציה שלישית** כוללת הוספת ערכים מורכבת, ז"א ביצוע שליפת נתונים מטבלה מסוימת (שאילתא רגילה) שאת תוצאותיה נכניס כנתונים לתוך הטבלה הנוכחית (יכול לשמש כמנגנון להעתקת נתונים מטבלה x לטבלה y):

דוגמא:

נניח שקיימת לנו טבלת עובדים בשם Employees (מהדוגמא הקודמת) וכעת אנו רוצים ליצור טבלה חדשה שתכיל מידע יותר מפורט לגבי משכורות העובדים - טבלת **משכורת לעובדים**, הבנויה בפורמט הבא:

```
CREATE TABLE EmpSalary  
(  
    Description Text,  
    Salary Int  
);
```

Employees		
ID	Name	Salary
1	Roei	4000
2	Yaniv	3000
3	Yaron	5000

בשלב זה נרצה להכניס לטבלה החדשה את המידע הבא:

- 1) הכנסת רשומה שתכיל את התיאור "**משכורת מינימאלית**" ואת הערך 3000.
- 2) הכנסת רשומה שתכיל את התיאור "**משכורת ממוצעת**" ואת הערך הממוצע מתוך טבלת עובדים (ערך זה יתקבל ע"י הפעלת שאילתה למציאת ערך AVG מטבלת עובדים).
- 3) הכנסת רשומה שתכיל את התיאור "**משכורת מקסימאלית**" ואת הערך המקסימאלי מתוך טבלת עובדים (ערך זה יתקבל ע"י הפעלת שאילתה למציאת ערך MAX מטבלת עובדים).

1) **הוספת ערכים פשוטה**, ע"י הכנסת ערכים קבועים הידועים מראש:

```
INSERT INTO EmpSalary  
VALUES( 'Minimal salary', 3000 );
```

(2) הוספת ערכים מורכבת, נשתמש בשאילתא על טבלה קיימת בשביל לאכלס טבלה חדשה:

```
INSERT INTO EmpSalary  
VALUES( 'Average salary', ( SELECT AVG(Salary)  
FROM Employees ) );
```

```
INSERT INTO EmpSalary  
VALUES( 'Maximal salary', ( SELECT MAX(Salary)  
FROM Employees ) );
```

ב) עדכון ערכים ע"י פקודת Update

נוכל בכל זמן נתון לעדכן ערכים (פקודת DML) בתוך טבלה קיימת ע"י שימוש בפקודת SQL פשוטה הנקראת UPDATE, קיימות 3 שיטות לעדכון נתונים בטבלה נתונה:

(1) עדכון ערך-ערך

Update TableName

Set column1 = Value1 , column2 = Value2 , column3 = Value3 ...

במקרה זה נכניס את הערך Value1 לתוך כל ערכי השדות הקיימים בעמודה column1 וכד'.

(2) עדכון אוסף ערכים (רשומה) בבת אחת (לא נתמך בתוכנת MySQL):

Update TableName

Set (column1, column2, column3, ...) = (Value1, Value2, Value3, ...)

גם במקרה זה נכניס את הערך Value1 לתוך כל ערכי השדות הקיימים בעמודה column1 וכד'.

נוכל לראות ש-2 פקודת ה UPDATE הנ"ל מבצעות עדכון ערכים עבור כל הערכים בעמודה, אך ברוב המקרים נרצה להפעיל את פקודת ה Update בצורה ממוקדת, ז"א - רק עבור רשומה ספציפית, וזאת ניתן לבצע ע"י הוספת פקודת Where:

(3) עדכון ערך בודד ע"י שימוש בהתניה

Update TableName

Set column1 = Value1

Where column2 = Value2

במקרה זה נכניס את הערך Value1 לתוך שדה ספציפי שעונה להתניה שהוגדרה ב WHERE.

הערות:

(1) חשוב לשים לב שסוג ה type המוכנס יתאים ל type של העמודה שהוגדר בשלב יצירת הטבלה.
(2) גם בפקודת Update ניתן להכניס ערכים מתוך פקודת SELECT (כמו שראינו בפקודת Insert):

דוגמא: קיימת טבלת עובדים ישנה בשם Emp1. לאחרונה המפעל עבר להשתמש בטבלת עובדים חדשה בשם Employees ומילא את נתוני העובדים בטבלה זו (ידוע שלשתי הטבלאות אותה סכמה).

במהלך הכנסת הנתונים לטבלה החדשה נפלה טעות בנתוני עובד 7369, לאור כך הוחלט על עדכון פרטיו של העובד לפי פרטיו בטבלת העובדים הישנה, בה מס' העובד היה 7499.

Employees		
EmpNo	Salary	Department
1471	4800	Sales
2822	7600	Marketing
7369	1200	Finance

UPDATE Employees
SET (EmpNo, Salary, Department) =

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

חייב להחזיר רק
שורה אחת בכדי
לבצע את פקודת
Update ה

WHERE EmpNo = 7369;

הוספת התניה

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

ג) מחיקת רשומות ע"י פקודת Delete

לאור העובדה שראינו שנוכל בכל זמן נתון לעדכן ערכים בתוך טבלה קיימת, על אותו משקל נוכל גם למחוק שורה מתוך טבלה קיימת.

מבנה פקודת המחיקה הינו:

Delete From TableName
Where EmployeeID = 7698

פקודה זו תמחק את השורה של העובד שמספרו 7698

הערות:

- 1) אם בפקודה לא הייתה מופיעה פקודת ה WHERE – כל הרשומות בטבלה היו נמחקות.
- 2) אם ברצוננו למחוק ערך של שדה מתוך טבלה, נוכל להשתמש בפקודת Update ולהכניס ערך Null לתוך התא הספציפי שאת תוכנו נרצה למחוק.

יש אפשרות למחוק את כל השורות בטבלה (את תוכן הטבלה) מבלי למחוק את הטבלה (הסכמה) עצמה, מחיקה מסוג זה תשמור על מבנה הטבלה, העמודות, המפתחות והאינדקסים.

מחיקת כל תוכן הטבלה יתבצע ע"י הפקודה:

Delete From TableName

חשוב: פקודה זו לא ניתנת לביטול (אין אפשרות ל Undo) ולכן יש לנהוג בה בזהירות.