

הרצאה מס' 10

נושאים לשיעור זה:

▪ פקודות DDL:

- מפתח ראשי (Primary Key)
- מפתח זר (חיצוני – Foreign Key)
- אינדקס
- הגבלות/אילווצים (Constraints)
- עדכון סכמה קיימת (Alter)

סיכום הפקודות עד לשיעור זה:

```
SELECT id, count(id) + (S-Q)
FROM Students AS S
LEFT OUTER JOIN Courses AS C JOIN (S-Q) AS SQ
ON S.studentID = C.studentID
WHERE (id=3) AND (name is not NULL)
AND X = (S-Q)
GROUP BY id
HAVING count(id) > 1 / count(id) = (S-Q)
ORDER BY id, name
```

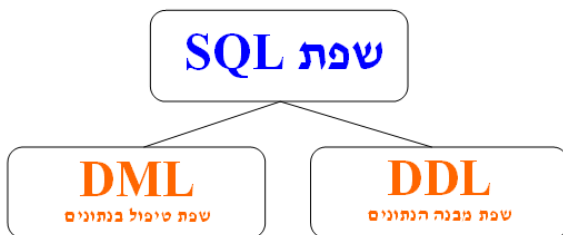
*** (S-Q) = מקומות בהם ניתן לשלב תתי שאילתות

פקודות DML:

בשיעורים האחרונים עסקנו בפקודות DML הנוגעות בנתון פקודות העוסקות בשליפה, הוספה, עדכון ומחיקה של נתונים/רשומות בטבלה.

עד כה, למדנו את הפקודות הבאות:

- Insert – הכנסת נתונים לטבלה
- Update – עדכון נתונים בטבלה קיימת
- Delete – מחיקת נתונים מטבלה קיימת
- Select – שליפת נתונים



השיעור נתחיל ללמוד את החלק השני בשפת ה SQL – פקודות DDL העוסקות במבנה.

פקודות DDL:

(1) מפתח ראשי (Primary Key)

מפתח ראשי – שדה (או אוסף שדות) המזהה באופן חד ערכי (ומינימאלי) רשומה מסוימת בטבלה.

דוגמא 1: מפתח ראשי הבנוי משדה בודד - מספר תעודת זהות בטבלת סטודנטים.

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

Primary Key

דוגמא 2: מפתח ראשי הבנוי מאוסף שדות – נבנה טבלה המציגה רשימת ספקים ורשימת מוצרים, כאשר

נתון:

- שמות הספקים יהיו: אסם, שטראוס ותנובה
- המוצרים המיוצרים על ספקים אלו הם: קוטג', קטשופ וגבינה לבנה


Vendor	Product	Price	Weight (gr)
Osem	Ketchup	10.00	1000
Strauss	Cottage	6.20	250
Tnuva	Cottage	6.30	250
Strauss	Cheese	5.80	200
Tnuva	Cheese	6.20	200

מי יהיה שדה המפתח בטבלה ?

נוכל לראות שלא ניתן להגדיר את הספק כמפתח כי יש ספקים החוזרים על עצמם, על אותו משקל נראה שלא ניתן להגדיר את המוצר כמפתח ראשי כי יש מוצרים בעלי אותו שם לספקים שונים (וכנ"ל לגבי מחיר ומשקל בהם ערכים חוזרים על עצמם).

← הפתרון הוא להגדיר מפתח מורכב (מפתח המורכב ממספר שדות) המכיל את השדות ספק+מוצר. שילוב של שדות אלו מבטיח בצורה חד ערכית שהשילוב של ספק+מוצר הינו ייחודי ומינימאלי עבור כל שורה בטבלה.

Primary Key



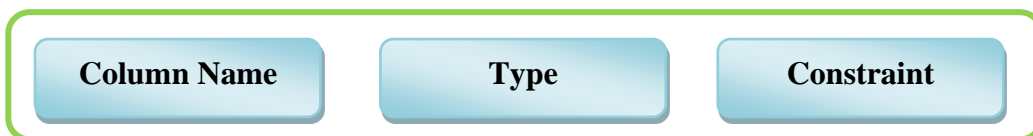
Vendor	Product	Price	Weight (gr)
Osem	Ketchup	10.00	1000
Strauss	Cottage	6.20	250
Tnuva	Cottage	6.30	250
Strauss	Cheese	5.80	200
Tnuva	Cheese	6.20	200

הערות

- **כפילות ערכים בשדה מפתח** (בהמשך לדוגמא הקודמת) - אם מפתח ראשי מורכב מיותר משדה אחד (כמו בדוגמא לעיל), אזי **ערכי אחד השדות יכולים להיות כפולים** בתוך עמודה נתונה (לדוגמא: בעמודת ספק נוכל לראות שתנובה מופיעה מספר פעמים) אבל השילוב של כל שדות המפתח הראשי חייבים להיות ייחודיים ברמת טבלה.

הוספת אילוף בשלב יצירת טבלה

כפי שלמדנו עד היום, שאנו באים ליצור טבלה חדשה עלינו להגדיר עבור כל טבלה את השדות (העמודות) שירכיבו את הסכמה של הטבלה לעתיד. במהלך בניית טבלה חדשה אנו יכולים בנוסף לשם העמודה העתידי ולסוג העמודה (type) להוסיף גם אילוף עבור כל עמודה המוגדרת בסכמה. מבנה שורה בפקודת Create יוכל להכיל את המבנה הבא:



לדוגמא:



הגדרת מפתח ראשי

הגדרת מפתח ראשי - אנו נגדיר את המפתח הראשי בטבלה בשלב של יצירת הטבלה ע"י הוספת הפקודה Primary Key. והשדות שמרכיבים את המפתח.

*** הערה, בעקרון בטבלה יוצג מספר הספק וקוד המוצר ולא השמות שלהם, אך בדוגמא זו על מנת להבהיר את הנקודה הוצגו הנתונים בשמותיהם ולא בקודים המסתמים אותם. ***

```
CREATE TABLE Example
(
  Vendor      TEXT NOT NULL,
  Product     TEXT NOT NULL,
  Price       FLOAT NOT NULL,
  Weight      FLOAT,
  PRIMARY KEY (Vendor, Product)
);
```

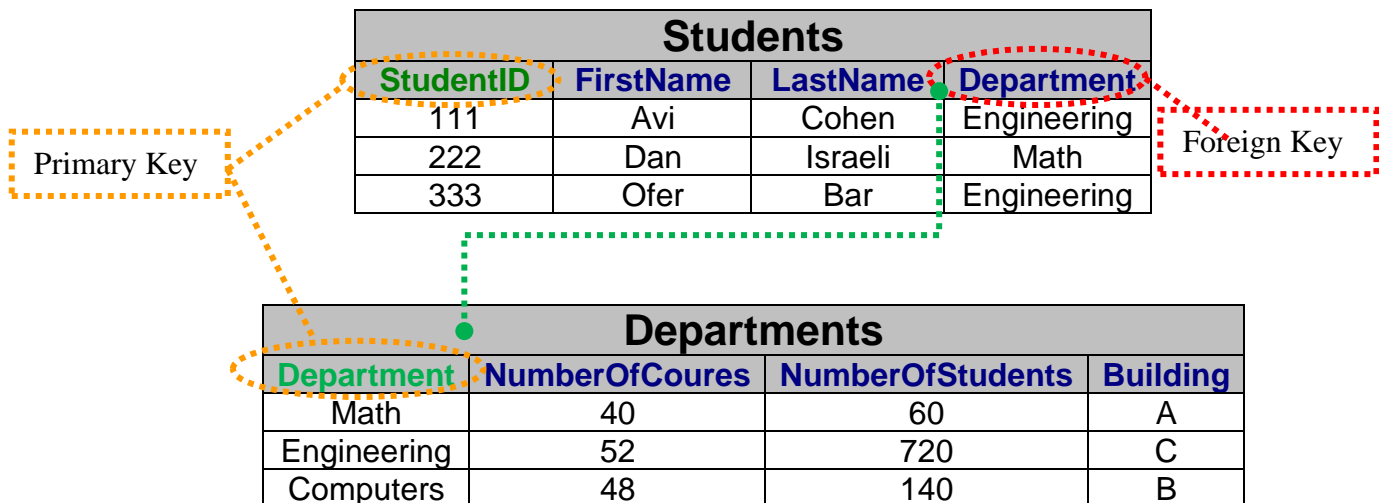
- **כמות מפתחות** - כל טבלה יכולה שיהיה לה **רק מפתח ראשי אחד** (שיכול להיות מורכב ממספר שדות) והוא אינו יכול להכיל ערכי NULL מכיוון ששדות המפתח חייבים להכיל ערכים ייחודיים.
- **מניעת חזרות** – ע"י הגדרת מפתח ראשי אנו בעצם מונעים אפשרות של הכנסת שורות כפולות לתוך אותה טבלה. ע"י הגדרת מפתח ראשי, ה DBMS הוא זה שמונע מצב של הכנסת שורה כפולה.

רואי זרחיה - כל הזכויות שמורות ©

(2) מפתח זר / חיצוני (Foreign Key)

- מפתח זר – שדה (או אוסף שדות) המצביע על מפתח ראשי בטבלה אחרת.
- מפתח זר הינו שדה (עמודה) בטבלה אשר ערכיו האפשריים (Domain) נשאבים משדה מקביל בטבלה אחרת אשר בה הוא משמש כמפתח ראשי.
- ערכי המפתח הזר אינם בלעדיים (Not Unique) בטבלת הברן ויכולים לחזור על עצמם.

דוגמא: קיימת טבלת סטודנטים שבה כל סטודנט משייך למחלקה בה הוא לומד, כמו כן קיימת טבלת מחלקות המכילה נתונים על כל מחלקה בפקולטה למדעים מדוייקים. ניתן לראות שבכל טבלה יש מפתח ראשי ובמקביל בטבלת סטודנטים עמודת המחלקה מהווה מפתח זר לטבלת מחלקות (כיוון ששדה זה מהווה מפתח ראשי בטבלת מחלקות).

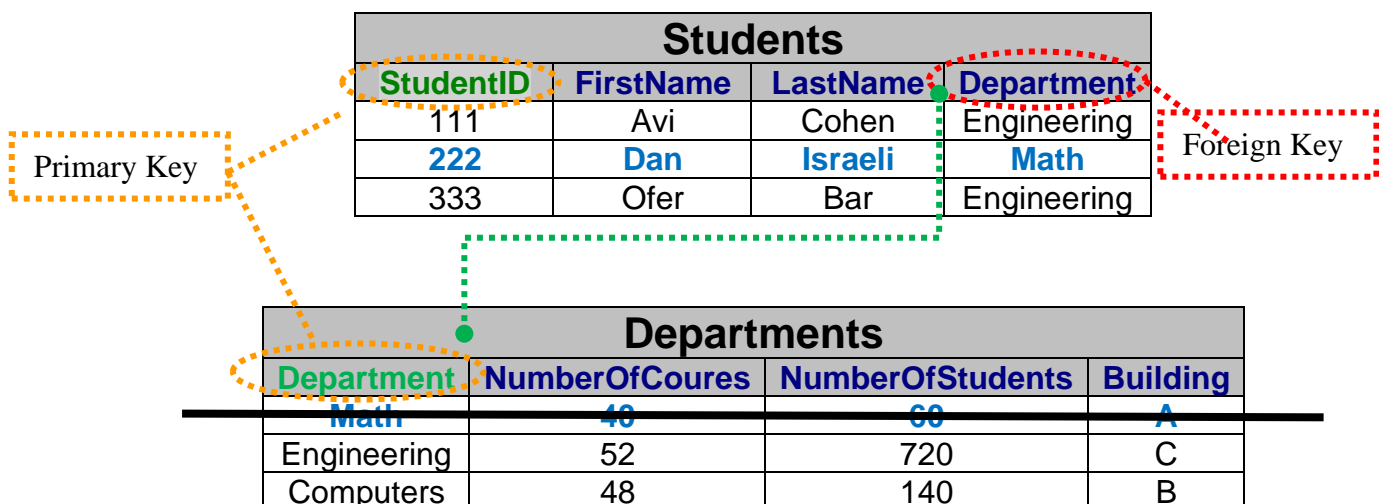


נוכל לראות שבהפניה של מפתח זר, נוצר קישור (לינק) בין 2 טבלאות (בירווק) כאשר בצד אחד מופיע שדה המהווה מפתח ראשי בטבלה אחת (מוקף בכתום) – שדה זה הופך להיות מפתח זר בטבלת השנייה (מוקף באדום).

הערות

מטרה נוספת של מפתח זר היא **שמירה על שלמות הנתונים** [Data integrity], הכוונה בכך היא שניתן להגדיר קשר זה כאילוץ (Constraint) בהגדרת הטבלה, כך שמערכת ניהול בסיס הנתונים (DBMS) תמנע מצב שיוכנסו ערכים בשדה המפתח הזר שאינם עונים לאילוץ, כלומר אינם נמצאים בטבלת האב. כתוצאה מהגדרת אילוץ כזה, לא נוכל למחוק רשומה מטבלת האב כל עוד ישנן רשומות בטבלת הבן המכילות את מפתח הרשומה כמפתח זר. בשורה התחתונה: על ידי שימוש במפתח זר מגדירים, ולמעשה יוצרים, את הקשרים בין הטבלאות השונות בבסיס הנתונים.

לדוגמא: נניח שהיינו **רוצים למחוק את מחלקת מתמטיקה מתוך טבלת מחלקות**, מחיקה מסוג זה תיצור בעיה לאור העובדה שקיים סטודנט (222) ששייך למחלקה זו (בטבלת סטודנטים) ולכן בשלב המחיקה מטבלת מחלקות, המפתח הזר יגרור את הופעתה של הודעת שגיאה המונעת את מחיקת שורה זו מטבלת מלקות אלא אם קודם לכן נמחק את הסטודנט השייך למחלקה זו (שזו כמובן בקשה הגיונית לאור העובדה שאנו מתכוונים למחוק את המחלקה מהמאגר והגיוני שנוכל למחוק אותה רק אם אין בה סטודנטים).



- בטבלת סטודנטים, שם המחלקה הינו מפתח זר לטבלת מחלקות (כיוון ששדה זה מהווה מפתח ראשי בטבלת מחלקות).
- נוכל לראות שבהפניה של מפתח זר, נוצר קישור (לינק) בין 2 טבלאות (בירוק) כאשר בצד אחד מופיע שדה המהווה מפתח ראשי בטבלה אחת (מוקף בכתום) – שדה זה הופך להיות מפתח זר בטבלת השנייה (מוקף באדום).

3) אינדקס

- **אינדקס** – רכיב מסוג DDL המשמש להשגת מהירות אופטימאלית בנגישות למידע.

דוגמא: נתונה השאילתא הבאה שמטרתה לשלוף את המחיר של זוגות הנעליים הנמצאים בטבלת הקניות המצורפת, כאשר ידוע ששדה המפתח בטבלה הוא **BarCode**.

```
SELECT Price
FROM Shopping
WHERE product = 'Shoes'
```

Shopping			
BarCode	StoreName	Product	Price
289	ZARA	Shirt	120
281	Ralph Lauren	Shoes	350
476	Hugo Boss	Jeans	750

ניתן לראות שבטבלת **Shopping** קיימות חנויות בגדים שונות ומוצרים שונים (המסודרים בסדר אקראי לחלוטין) ולכן על מנת למצוא את הרשומות העונה על תנאי ה **Where** נצטרך לעבור על כל רשומות הטבלה.

בעיה:

אך מה היה קורה אילו הטבלה הייתה מכילה 10,000 רשומות – היינו צריכים לעבור סדרתית על כל הרשומות שכן יתכן שבחנות נוספת נרכשו נעליים (יתכנו חזרות בעמודה שאינה מפתח ראשי) ולכן במצב זה התהליך לא כל כך יעיל ועלול לקחת הרבה מאוד זמן שכן פקודת ה **Where** תהיה חייבת לעבור על כל הרשומות בטבלה בכדי לאתר את הרשומות שעונות על התנאי.

← פתרון: שימוש באינדקס

פתרון

- האינדקס מהווה טבלת מידע המאפשרת גישה ישירה לרשומות (אם ניתן אז על-פי מפתח), מבלי שיהיה צורך לסרוק את כל הרשומות בטבלה, וזאת באמצעות התאמה בין המפתח של הרשומה לבין כתובתה באמצעי האחסון, בפועל האינדקס מכיל קישורים לטבלאות שונות, כאשר הקישורים יוצרים מיפוי של המידע היושב בזיכרון.
- האינדקס ממין את הערכים - באופן זה, גישה לרשומה ספציפית דרך האינדקס של אותה רשומה דורשת רק חיפוש ברשימה ממוינת (האינדקס) – מהלך מהיר משמעותית, ולכך קיימים אלגוריתמים יעילים מאוד. זאת, בניגוד לחיפוש הרשומה עצמה באמצעי האחסון, מהלך שעשוי להיות ארוך ומאוד לא יעיל כאשר כמות הרשומות המאוחסנות הינה גדולה.
- האינדקסים יכולים להוריד משמעותית את כמות המידע שתטען לזיכרון במטרה להחזיר את התשובה לשאלתא שהורצה (ללא אינדקסים בכל פעם שמתמש 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). בפועל האינדקסים מונעים את הצורך לבצע פעולות רבות הגוזלות זמן של סריקת טבלאות (וזאת על ידי הצגת טבלאות במספר צורות חלופיות ויעילות ברמה ארגונית. אינדקסים בנויים משימוש בסט עמודות אשר באופן מיוחד מזהים סט מידע).

(א) יצירת אינדקס:

- בניית האינדקס נעשית ע"י ה DBMS וגוזלת משאבים בעת פעולת היצירה עצמה.
- יצירת אינדקס בטבלה קיימת, תאפשר לאתר שורות מידע במהירות וביעילות רבה יותר.

```
CREATE INDEX index_name
ON table_name ( col1, col2, ..., coln )
```

שם האינדקס החדש

שם הטבלה שעליה נרצה להוסיף אינדקס חדש

שם העמודות בטבלה שיהיו את האינדקס

רואי זרחיה - כל הזכויות שמורות ©

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

דוגמא: נוכל ליצור אינדקס לטבלת Students עבור עמודת LastName וזאת בכדי שנוכל לשלוף סטודנטים לפי שמות משפחה בצורה מהירה ויעילה יותר:

```
CREATE INDEX StudentIndex  
ON Students (LastName)
```

הערה: עבור מקרה שבו שמות הסטודנטים יושבים בשתי עמודות שונות (שם פרטי ושם משפחה) היינו יכולים ליצור את האינדקס כך:

```
CREATE INDEX StudentIndex2  
ON Students (LastName, FirstName)
```

הערות

- ניתן ליצור אינדקסים על עמודה אחת או מספר עמודות כאשר כל אינדקס מקבל שם ייחודי משלו.
- 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 (כנ"ל לגבי פעולת Insert או פעולת Delete).
- המשתמש שירצה לבצע שליפה מבסיס הנתונים אינו רואה את האינדקס וזאת כי האינדקס נמצא מאחורי הקלעים ומטרתו היא שיפור ביצועי השליפות.

(ב) מחיקת אינדקס:

```
DROP INDEX index_name
```

4 הגבלות/אילוצים (Constraints)

אילוצים כופים אמינות ודואגים לשלמות המידע בטבלאות מסד הנתונים. אילוצים בדרך כלל מתווספים בשלב ה Create Table וניתן למקד אותם ברמת העמודה או הטבלה.

סוגי האילוצים העיקריים:

1. **בעל ערך שאינו Null** : משתמשים באילוץ זה כדי לוודא שעמודה לא מכילה ערכי Null .
 2. **מפתח ראשי (primary key)** : מבטיח מצב שבו לכל טבלה קיים מפתח אחד ויחיד שאינו NULL. (הגדרת מפתח ראשי יוצרת אוטומטית אינדקס ייחודי לטבלה).
 3. **מפתח זר (foreign key)** : מפתח זר קושר עמודה אחת או יותר בטבלה עם המפתח הראשי שהוגדר בטבלה אחרת. אילוץ מפתח זר מבטיחים שלמות נתונים בין 2 טבלאות.
לדוגמא: נשתמש במפתח זר כדי לוודא שמספר מחלקה אליה רשום סטודנט בטבלת סטודנטים מייצגת באמת מחלקה הקיימת בטבלת מחלקות.
 4. **אילוץ בדיקה (check)** : מחייב אמינות תחום הערכים (domain integrity) על ידי קביעת תחומי ערכים אשר יכולים להיות מוכנסים לעמודה (עמודה יכולה להיות בעלת אילוץ בדיקה אחד או יותר).
לדוגמא: ניתן להשתמש באילוץ בדיקה כדי לוודא שעמודה מקבלת ערכים בטווח מסוים.
- ### ערך ברירת מחדל (default)** : מאפשר לקבוע ברירת מחדל לשדה במקרים בהם לא יוכנס ערך לשדה זה.
לדוגמא: ניתן להכניס את הערך "0" לשדה ניסיון כאשר לא ידוע אם למועמד יש ניסיון. במקרה בו ידווח מועמד שיש לנו ניסיון אזי מספר שנות הניסיון שלו ידרסו את הערך "0".

דוגמא: יצירת טבלה חדשה עם מפתחות ואילוצים (Constraints):

```
CREATE TABLE Workers
(
  ID          INT    PRIMARY KEY,
  Name       TEXT(20) NOT NULL,
  StartDate  DATE,
  Address    TEXT(30),
  Phone      TEXT(12) NOT NULL,
  Height     FLOAT,
  Weight     FLOAT  CHECK(Weight Between 50 and 150),
  Experience INT    DEFAULT 0,
  Department_ID INT REFERENCES Departments (Dep_ID) ON DELETE CASCADE,
  FOREIGN KEY (Department_ID)
);
```

הנחה: טבלת מחלקות קיימת כבר ב DB

Departments			
Dep_ID			

בדיקת ערכים – יסכים לקבל רק אנשים שמשקלם בטווח שבין 50 ל 150

ערך 0 כברירת מחדל

- כל עובד (מטבלת workers) חייב להיות משויך למחלקה בטבלת מחלקות (Departments) ולכן קוד מחלקה הוא **מפתח זר** בתוך טבלת Workers.
- כיוון שכל עובד מקושר לטבלת מחלקות, אזי **ברגע שיחליטו על סגירת מחלקה מסוימת** (מחיקת המחלקה ממסד הנתונים) תופעל פעולת המחיקה "ON DELETE CASCADE" שתגרור את מחיקת העובדים השייכים למחלקה זו בצורה אוטומטית.
- **הערה:** הקשר בין 2 המחלקות הינו: שדה Dep_ID שהוא מפתח ראשי בטבלת מחלקות מחובר לשדה Department_ID בטבלת עובדים שבה הוא מהווה מפתח זר.

5) עדכון סכמה של טבלה קיימת ע"י פקודת Alter

ראינו איך יוצרים טבלה חדשה ב SQL ע"י פקודת Create Table ובתרגול האחרון ראינו איך ניתן לבצע שינויים במבנה הטבלה ע"י חלון Edit Table, כעת נלמד איך ניתן לבצע הוספת מבנה בטבלה (תוספת לסכמה הקיימת) ע"י פקודה:

הוספת עמודות/הגבלות לסכמה הקיימת תבוצע ע"י פקודת ALTER.

- ניתן להוסיף עמודות חדשות בסוף הסכמה הקיימת.
- ניתן להוסיף הגבלות ע"ג עמודות קיימות.

ALTER TABLE Employees

ADD

```
(  
  Bonus INT CHECK( Bonus <> 666 ),  
  BirthDate DATE  
);
```

הוספת עמודת בונוס לטבלת עובדים עם הגבלה על סכומה (לסוף הסכמה)

הוספת עמודת תאריך לידה לטבלת עובדים (לסוף הסכמה)