# Advanced JOIN

**ל.ע. זה הוא המשך של Basic JOIN, ודאו קודם שאתם זוכרים ובטוחים בחומר הנלמד בל.ע. הקודם.**

אז עד עכשיו למדנו שכאשר אנחנו מצרפים בין 2 טבלאות, הראשונה היא השמאלית והשנייה היא ימנית, ועל סמך כך נוכל להשתמש בסוגי JOIN שונים. עכשיו אנחנו מגיעים לסוגיה קצת יותר מורכבת:

**שליפה מרובת טבלאות**

SELECT emp.firstname || ' ' || emp.lastname as "employee",

dpt.deptname,

proj.projname

FROM liat.t\_empl emp

INNER JOIN liat.t\_depart dpt ON

dpt.deptno = emp.deptno

INNER JOIN liat.t\_proj proj ON

proj.deptno = dpt.deptno;

אז אם יש לנו 2 פסוקיות - JOIN, איך נדע מה הטבלה הימנית ומה השמאלית?

כפי שאנחנו כבר יודעים, במידה והיה רק JOIN אחד, הטבלה שנמצאת ב – JOIN היא הטבלה הימנית והטבלה שנמצאת ב – FROM היא הטבלה השמאלית. ברגע שמתבצע ה – JOIN הראשון נוצרת לנו טבלה חדשה. אותה טבלה הופכת להיות הטבלה השמאלית החדשה שלנו, והטבלה שנמצאת ב – JOIN הבא הופכת להיות הטבלה הימנית. כך, נוצר מצב שכל המידע הרלוונטי נמצא כל הזמן ולא נעלם(תלוי בסוג ה – JOIN).

כאשר נכתוב שליפה מרובת טבלאות, נכתוב אותה בסדר הגיוני. כלומר נעשה את הקישורים בסדר הדרגתי – נחבר את הטבלה הראשונה לשנייה, השנייה לשלישית בהצהרות ה ON וכולי על פני לכתוב טבלה אחת ב FROM ולאחר מכן לקשר טבלאות שונות לאותה טבלה ראשונה.

**למה זה קיים?**

על בסיס הבדלה זו משתמשים בכל מיני סוגי JOINים שונים בהתאם למקרה ולמטרת השליפה. בואו נתמקד רגע במקרים השונים ובשימוש בפסוקיות השונות.

**תרגיל(על ה – DB של ליאת):**

הצג את כל המחלקות ואת שמות העובדים שלהם.

**פתרון:**

**עם LEFT JOIN:**

SELECT dpt.deptname,

emp.firstname || ' ' || emp.lastname AS emp\_name

FROM liat.t\_depart dpt

LEFT JOIN liat.t\_empl emp ON

emp.deptno = dpt.deptno;

הסר את המלבן הלבן בשביל לחשוף את התשובה

**עם RIGHT JOIN:**

SELECT dpt.deptname,

emp.firstname || ' ' || emp.lastname AS emp\_name

FROM liat.t\_empl emp

RIGHT JOIN liat.t\_depart dpt ON

emp.deptno = dpt.deptno;

הסר את המלבן הלבן בשביל לחשוף את התשובה

כפי שניתן לראות ניתן לפתור את השאלה בעזרת שני ה –JOINים.

אז מה ההבדל ביניהם?

**LEFT JOIN VS RIGHT JOIN**

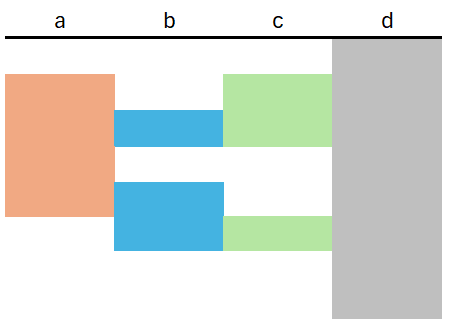
כפי שניתן לראות שתי הפסוקיות מאוד דומות, ולפי ההיגיון ניתן להשתמש באחת בצורה הפוכה מהשנייה כדי לקבל את אותה התוצאה.

**אז האם קיים הבדל? מתי נשתמש בכל JOIN?**

ברוב המקרים והשליפות בהם ניתקל, סביר להניח שנראה שימוש רחב יותר בפסוקיתLEFT JOIN ממספר סיבות:

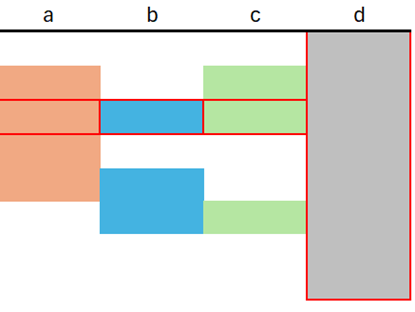
* שימושיות – ברוב המקרים מה שניתן לעשות עם RIGHT ניתן גם לעשות עם LEFT ולהפך
* אינטואיטיביות – משום שהכתיבה נעשית משמאל לימין, יותר אינטואיטיבי לקרוא כך גם את ה- JOIN.

למרות זאת, ישנן סיטואציות שבהן שימוש ב-RIGHT JOIN לא יניב לנו את אותה התוצאה כמו LEFT JOIN, ובגלל הדרך והסדר שבהם החיבורים בין הטבלאות נעשים נוכל להשיג את התוצאה הרצויה רק בעזרת ה- RIGHT JOIN.

**לדוגמה:** בשליפות בהן נרצה לחתוך מספר טבלאות ולאחר מכן לחבר אותן לטבלה ב- OUTER נשתמש ב RIGHT JOIN עבור הטבלה האחרונה.

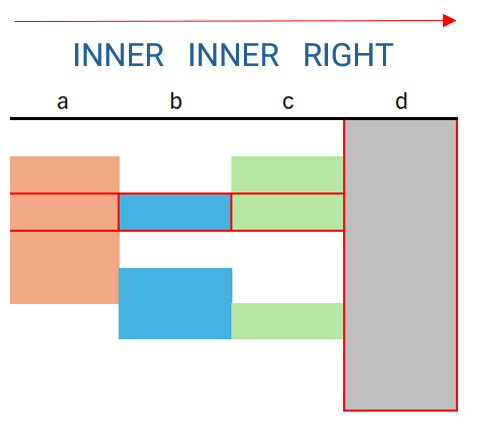
מה הכוונה?

המידע נראה כך:

(d - a מייצגים טבלאות)

ובמקרה זה נרצה לקחת את החיתוך בין

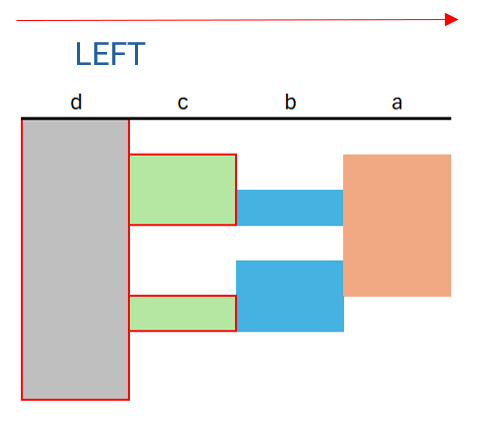
a, b, c ואת כל המידע מטבלה d (החלקים שמוקפים באדום).



כיצד נעשה זאת?

INNER JOIN בין שלוש הטבלאות (a, b, c) ו –

RIGHT JOIN כדי לשלוף את כל המידע מ- d.



האם זה היה עובד בדרך ההפוכה?

כלומר, האם היינו מקבלים את טבלת התוצאה שרצינו אם היינו עושים LEFT JOIN לטבלה d ואז INNER JOIN לשאר הטבלאות?

**התשובה היא לא.**

החיבור בין d ל-c באמצעות LEFT היה מניב לנו את כל טבלה d והמשותף לה מטבלה c, כפי שמודגש בתמונה,

תמונה שמכילה טקסט, צילום מסך, תרשים, קו

התיאור נוצר באופן אוטומטי

אך החיבור הנוסף לטבלת b באמצעות INNER היה מחזיר לנו רק את המידע המשותף לשלוש הטבלאות,

ו"יבטל" לנו את ערכי ה-NULL הנוספים

שקיבלנו ב- LEFT JOIN.

לכן, לא תמיד ניתן להפוך ביןRIGHT ו- LEFT, באמצעות שינוי סוג וסדר החיבורים בין הטבלאות, מאחר שלסדר החיבורים בין הJOINים השונים יש חשיבות שמשפיעה על המידע שמתקבל.

לכן יש לאפיין את סוגי ה- JOIN שעלינו לעשות לפני פתירת השאלה בהתאם לצורך שלנו, ולהבחין במקרי הקצה ובערכי הNULL שאנחנו מקבלים בכל חיבור שאנחנו מבצעים.

**– שימו ♥! –**

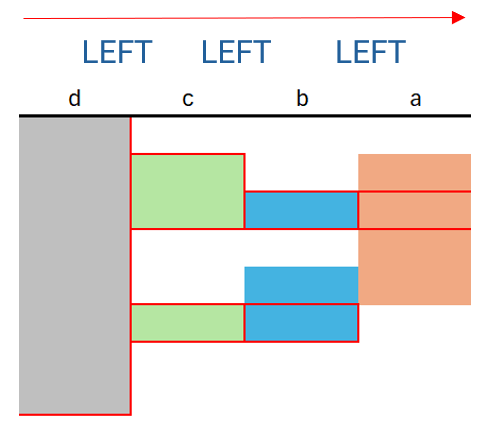
כאשר נבצע INNER JOIN אחרי JOIN מסוג LEFT/RIGHT, אנחנו נקבל רק את **המשותף** בין הטבלה השלישית לשתיים האחרות, ונאבד את כל ערכי הNULL שקיבלנו מהחיבור הראשון.

לכן ביצוע INNER JOIN אחרי JOIN מסוג LEFT/RIGHT יהיה **שווה ערך** לחיבור הטבלאות באמצעות שני INNER JOIN.

אז הבנו שבדרך ההפוכה, כאשר אנחנו משנים את LEFT ל-RIGHT או להפך, ובהתאם גם הופכים את הסדר, לא תמיד נקבל את אותה התוצאה.

ואם היינו מנסים לבצע את כל החיבורים באמצעות LEFT JOIN?

מה היינו מקבלים?



היינו מקבלים את כל טבלה d, (הטבלה השמאלית)

ואת מה שמשותף לה מהטבלה c (הימנית) + את המידע

שמשותף לשני הצירופים האלו עם טבלה b, ואת המידע

שמשותף לכל צירופים אלו עם טבלה a. בדיוק כפי שמודגש

בתמונה.

גם דוגמה זו אינה עונה על הצורך המקורי שלנו –

וכפי שניתן להבין, השורה התחתונה היא שלא ניתן לענות על הצורך ללא RIGHT JOIN.

**דוגמאות לחיבורי הJOINים השונים מהמפעל של ליאת**

INNER ו- RIGHT

השליפה עונה על השאלות: מי עובד בכל פרויקט ומאיזו מחלקה הוא? האם יש פרויקטים שאין בהם עובדים?

SELECT emp.firstname || ' ' || emp.lastname AS emp\_name,

dpt.deptname,

proj.projname

FROM liat.t\_empl emp

INNER JOIN liat.t\_depart dpt ON

dpt.deptno = emp.deptno

RIGHT JOIN liat.t\_proj proj ON

proj.deptno = dpt.deptno

ORDER BY deptname ASC,

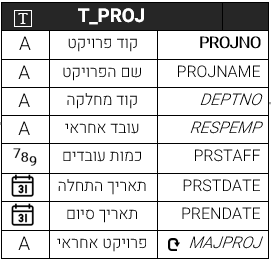
Projname ASC;

בדוגמה זו, נקבל מהחיבור הראשון את כל העובדים שיש להם מחלקות ורק מחלקות שיש בהם עובדים, ולהם נשייך את כל הפרויקטים. כך שערכי הNULL שנקבל הם פרויקטים שאין להם מחלקות או עובדים.

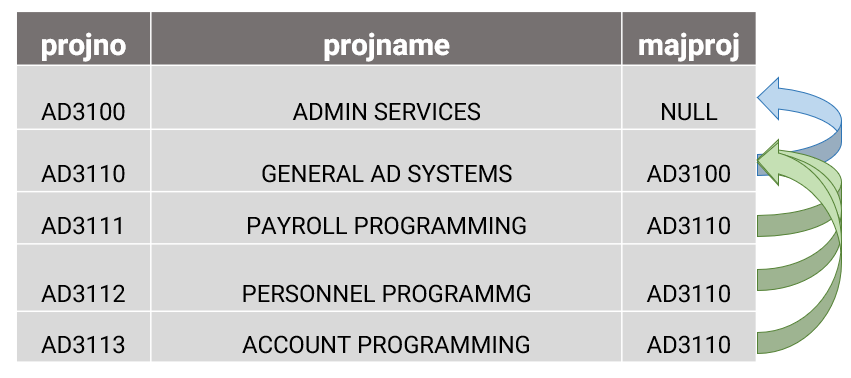
**SELF JOIN**

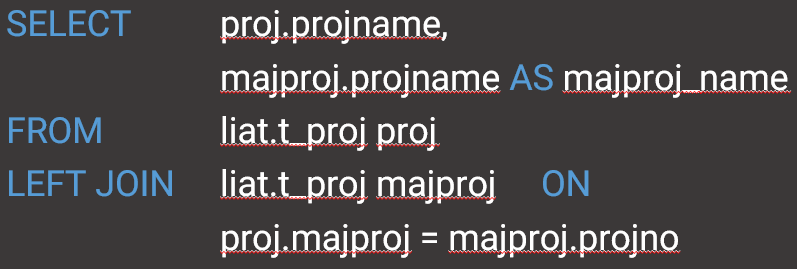
שליפת מידע משותף בין טבלה לעצמה על בסיס מפתח משני לאותה הטבלה.

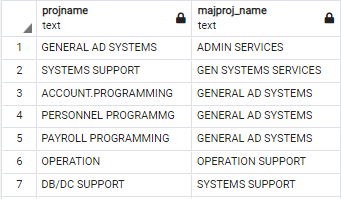
SELF JOIN היא למעשה לא פסוקית אלא פעולה אותה נבצע בעזרת פסוקיות ה - JOIN השונות על אותה הטבלה. כלומר לא נכתוב בשליפה עצמה SELF JOIN אלא נממש את הפעולה בעזרת הפסוקיות השונות (INNER, RIGHT, LEFT...)

נוכל לעשות SELF JOIN בטבלאות בהן יש עמודת מפתח משני שמצביעה על המפתח הראשי של אותה הטבלה.

בואו נסתכל לדוגמה על טבלת הפרויקטים בDB של ליאת.

אנחנו רואים שיש לנו עמודה בשם majproj – פרויקט אחראי. זו עמודת מפתח משני שמתקשרת למפתח הראשי של טבלת הפרויקטים – קוד פרויקט. אם נסתכל על הטבלה עצמה נוכל לראות שהעמודה הזו מפנה לרשומות שקיימות בטבלת הפרויקטים. מאחר שגם פרויקטים שאחראיים על פרויקטים אחרים הם פרויקטים בפני עצמם ולכן מקומם בטבלה זו.

אם נרצה עכשיו להציג עבור כל פרויקט את שם הפרויקט האחראי עליו, נשלוף את הטבלה שלנו פעמיים – פעם אחת נתייחס אליה כטבלת פרויקטים ופעם שנייה כטבלת פרויקטים אחראים ונבצע את הקישור בין מפתח ראשי למשני – בין עמודת majproj בטבלת הפרויקטים ל projno בטבלת הפרויקטים האחראיים. 

טבלת התוצאה שלנו תיראה כך:

נסתכל על דוגמה נוספת.

**דוגמא:** הצג את שמות המחלקות ואת שם מחלקת האב שלהן

SELECT dpt.deptname Dept,

father.deptname Father

FROM liat.t\_depart dpt

INNER JOIN liat.t\_depart father ON

dpt.admrdept = father.deptno;

**תשובה:** נשלוף פעמיים מטבלת המחלקות, אל אחת נתייחס כאל טבלת מחלקות אב ולשנייה כאל טבלת מחלקות.

**שאלה:** הצג את שמות כל המחלקות ואת שם מחלקת האב שלהן במידה ויש

SELECT dpt.deptname Dept,

father.deptname Father

FROM liat.t\_depart dpt

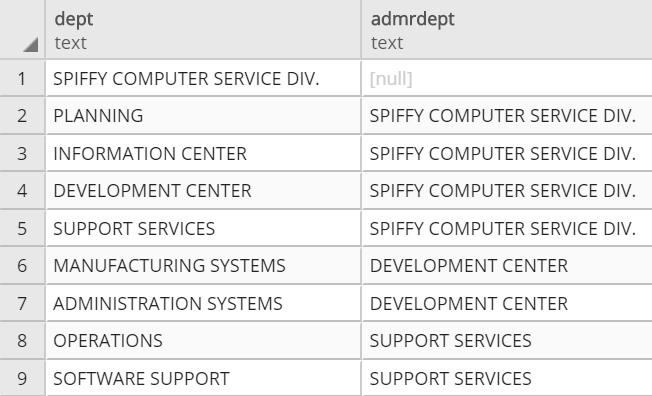
LEFT JOIN liat.t\_depart father ON

dpt.admrdept = father.deptno;

**תשובה:** הפעם נשתמש ב LEFT JOIN בכדי לקבל את כלל המחלקות גם אם אין להן מחלקת אב.

הסר את המלבן הלבן בשביל לחשוף את התשובה

טבלת התוצאה:

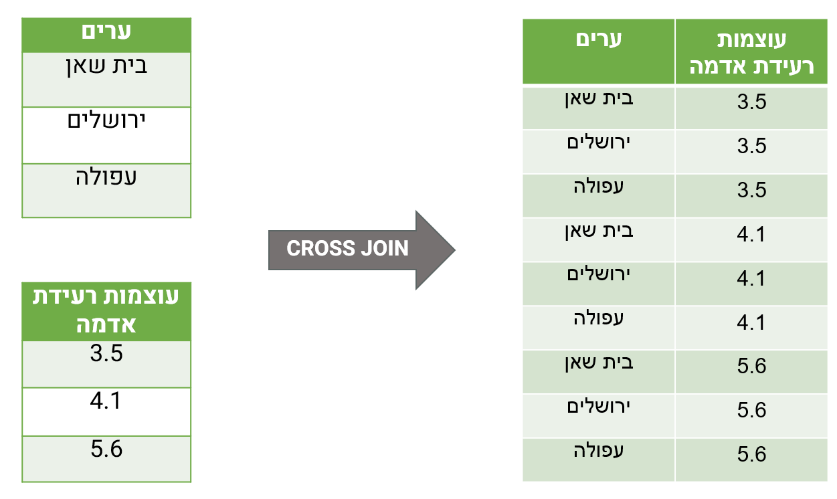


**CROSS JOIN**

CROSS JOIN שולף את כל הקומבינציות האפשריות בין שתי עמודות או יותר. (לפי בחירה שלנו – העמודות אותן נבחר).

בואו נסתכל על דוגמה אמיתית מהיחידות.

בפיקוד העורף מתעסקים במיגון של ערים שונות לקראת רעידות אדמה. כחלק מההתכוננות צריכים להבין כיצד תהיה ההתמודדות של כל עיר/ אזור בארץ עם עוצמות שונות של רעידות אדמה.

אנחנו רוצים לדעת כיצד בית שאן תתמודד עם רעידה בעוצמה 1,2,3, כיצד חיפה תתמודד עם רעידות אדמה בעוצמה 1,2,3 וכולי.

במצב זה נרצה להשתמש ב CROSS JOIN – על מנת לקבל את כל התרחישים האפשריים בכל עיר, ובהמשך לנתח כיצד כל עיר מותאמת לכל עוצמה.

כיצד נכתוב שליפה עם CROSS JOIN?

לדוגמה, נרצה לשלוף את כל הקומבינציות האפשריות של עובדים ושל מחלקות(שמות). ידוע שישנם 43 עובדים ו – 9 מחלקות, לכן נוכל לדעת כמה שורות יצאו מהשליפה – 9\*43 = 387

SELECT   emp.firstname || ‘ ‘ || emp.lastname AS emp\_name,

dpt.deptname

FROM      liat.t\_empl emp

CROSS JOIN liat.t\_depart dpt;

(יכולים להריץ את השאילתה ולראות שעבור כל עובד מוצגות כל המחלקות). שימו לב שלא נציב תנאי ON לאחר ציון הJOIN, מאחר שאנחנו מציגים את כל הקומבינציות האפשריות, ללא קשר להתאמה של הערכים בין המפתח הראשי למפתח המשני בטבלאות.

נרצה להציג גם את כל הקומבינציות האפשריות ביחד עם הפרויקטים.

כמה שורות ישלפו?

**תשובה(שאילתה):**

SELECT   emp.firstname || ‘ ‘ || emp.lastname AS emp\_name,

dpt.deptname,

proj.projname

FROM      liat.t\_empl emp

CROSS JOIN liat.t\_depart dpt

CROSS JOIN liat.t\_proj proj;

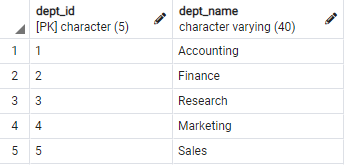
7740 = 9\*43\*20

הסר את המלבן הלבן בשביל לחשוף את התשובה

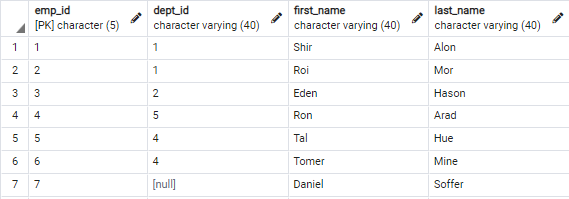
סוג JOIN זה פחות שימושי ופחות נתעמק בו במהלך הקורס, אך חשוב להבין מה הוא עושה וכיצד להשתמש בו.

**ההבדלים בין סוגי ה-JOINים**

כדי להבין יותר את ההבדלים בין סוגי הJOINים השונים, נראה דוגמא קצרה.

יש לנו שתי טבלאות. הראשונה, מראה מידע על מחלקות:

יש לנו חמש מחלקות, על כל אחת יש קוד מחלקה ואת שמה.

הטבלה השנייה מראה מידע על עובדים.

יש לנו שישה עובדים, על כל אחד יש קוד עובד, קוד מחלקה שבה הוא נמצא, ואת שמו המלא.

שימו לב שאין אף עובד במחלקה 3 – מחלקת החיפוש ושלעובד מספר 7, דניאל סופר, לא שייך למחלקה.

בואו נתחיל לראות מה קורה כשאנחנו מחברים בין שתי הטבלאות.

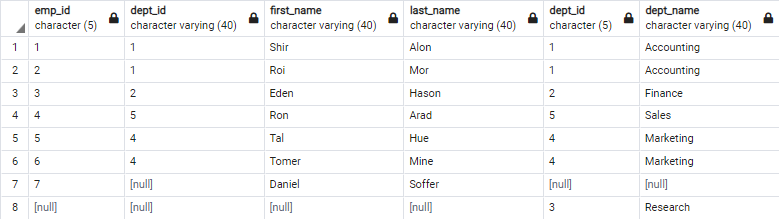
**FULL JOIN**

SELECT \*

FROM employees emp

FULL JOIN departments dpt ON

emp.dept\_id = dpt.dept\_id;

אנחנו שולפים הכל מטבלת העובדים וטבלת המחלקות, ומחברים אותן באמצעות FULL JOIN. נחבר על המפתח הראשי – dept\_id.

אנחנו רואים שלמרות שאין שום עובד במחלקה 3, היא עדיין נשלפה כי השתמשנו ב-FULL JOIN – הוא שולף את כל המידע משתי הטבלאות, בלי קשר לאם הוא משותף או לא. כך גם העובד דניאל סופר נשלף, על אף שהוא לא משויך לשום מחלקה. בנוסף, אנחנו רואים שעמודת dept\_id נשלפה פעמיים, כי למרות שהיא אותה עמודה היא נמצאת בשתי הטבלאות, והDB לא מאחד ביניהן.

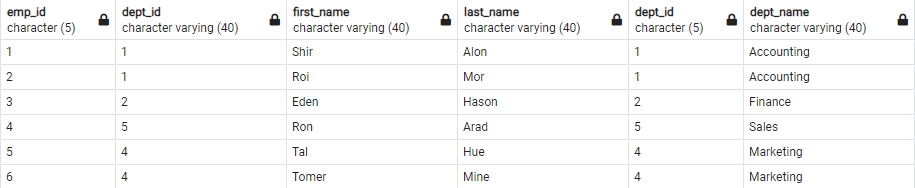
**INNER JOIN**

SELECT \*

FROM employees emp

INNER JOIN departments dpt ON

emp.dept\_id = dpt.dept\_id;

עכשיו נשלוף את שתי הטבלאות ונחבר בינהן באמצעות INNER JOIN.

פה אנחנו יכולים לראות שלא נשלף מידע על מחלקה 3 או על עובד מספר 7, כי INNER JOIN מראה רק את המידע המשותף לשתי הטבלאות – אז בגלל שאין עובד במחלקה 3 ושלעובד מספר 7 אין מחלקה, המידע עליהם לא נשלף.

**RIGHT JOIN**

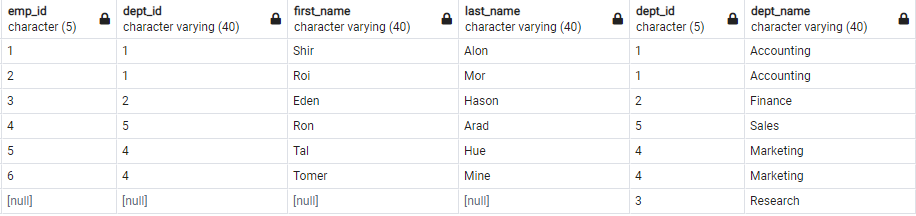
SELECT \*

FROM employees emp

RIGHT JOIN departments dpt ON

emp.dept\_id = dpt.dept\_id;

עכשיו נשלוף את שתי הטבלאות ונחבר בינהן באמצעות RIGHT JOIN.

הטבלה השמאלית היא טבלת העובדים, והטבלה הימנית היא טבלת המחלקות.

בגלל שRIGHT JOIN מראה את המידע המשותף לשתי הטבלאות, ואת כל המידע מהטבלה הימנית, אנחנו נראה את מחלקה 3. מחלקה 3 היא מידע שיש רק בטבלה הימנית. לעומת זאת, אנחנו לא נראה את עובד מספר 7, מאחר שאין לו מחלקה, הוא קיים רק בטבלה השמאלית.

**LEFT JOIN**

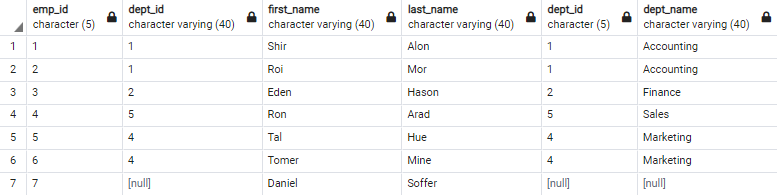
SELECT \*

FROM employees emp

LEFT JOIN departments dpt ON

emp.dept\_id = dpt.dept\_id;

עכשיו נשלוף את שתי הטבלאות ונחבר ביניהן באמצעות LEFT JOIN.



בגלל שLEFT JOIN מראה את המידע המשותף לשתי הטבלאות, ואת כל המידע מהטבלה השמאלית, אנחנו נראה את עובד מספר 7 אבל לא נראה את מחלקה 3.

**סדר פעולת הפסוקיות**

**נקודות נוספות**

* שימו לב, שה – JOIN condition חייב להשוות בין שני data types שווים. הכוונה היא שניתן להשוות בין עמודות שונות כל עוד משנים את ה – types של ה – data(לדוגמא: בעזרת CAST).
* כל ה – JOINים חוץ מ INNER נחשבים OUTER. אין צורך לכתוב את המילה OUTER.
* קיימים עוד סוגי JOIN ביניהם NATURAL JOIN (מוזמנים לקרוא באינטרנט).

**לסיכום:**

פסוקיות JOIN הן חלק בלתי נפרד מהשפה ומתהליך העבודה של אנליסטים, בעזרתן נוכל לבצע חיתוכים, להשתמש בכלל המידע בDB ולבצע שאילתות מורכבות שיעזרו לנו להכיר את המידע טוב יותר, לענות על שאלות, ולעבד את המידע להצגה.

