

הקמנו את בסיס הנתונים של 5 הטבלאות לנהל סטודנטים שמשאילים ספרים מספריות והכנסנו לתוכן נתונים.

התחלנו בדיון על שאילתות פשוטות על טבלה בודדת עם פונקציות אגרגציה כמו: count(*) ו-sum(column) והמשכנו לדון בשאילתות קצת יותר מורכבות.

ראינו שאילתה שכוללת שילוב של שתי טבלאות. ע"מ למצוא את מספרי הזיהוי של הספרים שמצד אחד הושאלו על ידי מישהו, כלומר מופיעים בטבלה: Borrow, ומצד שני עדיין קיימים עותקים שלהם זמינים באחת או יותר מהספריות.

```
select ISBN
from Available av, Borrow bo
where a.ISBN = b.ISBN
and a.noOfCopies > 0
```

זה בעצם אומר תן לי את הספרים (מספר זיהוי של) שמופיעים בשתי הטבלאות ועבור כל צירוף כזה בדוק אם מספר העותקים שנותר עבור הספר בטבלה Available הוא גדול מאפס (כלומר יש עותק פנוי).

זוהי פעולה שניקראת Join, של שתי טבלאות על השדה ISBN שבמקרה זה הוא חלק מהמפתח בשתי הטבלאות (אבל לא חייב להיות).

ראינו כאן עוד אספקט שבו כתבנו אחרי שמות הטבלאות av עבור available ו-bo עבור borrow, זה ניקרא Alias (או שם נירדף), כדי לחסוך בכתיבת שם הטבלה על מנת לזהות שדות במדויק. יש לציין שאם השדה מופיע רק בטבלה אחת (כמו במקרה של noOfCopies) אפשר להשמיט את שם או כינוי הטבלה ורק לכתוב:

```
and noOfCopies > 0
```

בפעולת Join אנחנו מקבלים מכפלה קרטזית של שורות, כלומר כל השורות התואמות בין הטבלאות יופיעו בתוצאה. לדוגמה בטבלה: Available הספר בעל ISBN של: 6666666666 זמין גם בספריה Main וגם בספריה Spy (עותק אחד בכ"א). אם הוא היה מופיע בטבלה: Borrow אפילו רק מספריה אחת מהן, בתוצאה של השאילתה הנ"ל הוא היה מופיע פעמיים.

ראינו גם כיצד להתגבר על הבעיה הזו בעזרת מילת המפתח: distinct שעזרה להסיר את הכפילויות.

התחביר של שאילתת ה-Join הנ"ל אינו היחיד האפשרי, כפי שניראה מנוע ה-SQL ישכתב אותו בצורה קצת שונה.

פעולת ה-Join היא פעולה מאד בסיסית ב-SQL ולמעשה מדגימה את העוצמה שיש בבסיס נתונים יחסי להרכיב צירוף נתונים רצוי על ידי שילוב של מספר טבלאות (אין הגבלה מעשית למספר הטבלאות ב-join)

ע"מ להציג את שם הספר במקום את מספר הזיהוי שלו, צירפנו עוד טבלה לפעולת ה-join, זוהי הטבלה book, כיוון ששם הספר נימצא רק שם. שוב עשינו שימוש במפתח הזר של ISBN שהוא המפתח הראשי ב-book, כדי לבצע את פעולת ה-join של 3 הטבלאות המדוברות. זה נעשה כלהלן:

```
select title
from available av, borrow bo, book bk
where av.isbn= bo.isbn
and av.isbn= bk.isbn
```

שימו לב שאין חשיבות אם בשורה האחרונה היינו משווים את bo.isbn ל-bk.isbn במקום להשתמש ב-av.isbn כמו כלל החיבור: $a+b+c = b+a+c$

שאלתה נוספת הייתה למצוא את מספרי הזהות של הסטודנטים ששאלו ספר מהסופר: Bernard Cushing או מאת הסופר Sherman Chow.

לשם כך השתמשנו בפעולה הלוגית של OR:

```
select st.id
from student st, borrow bo, book bk
where bo.id = st.id
and ( bk.author = 'Bernard Cushing' or bk.author = 'Sherman Chow')
and bo.isbn = bk.isbn;
```

נסו להשמיט את השורה האחרונה ובידקו מה התקבל. אם לא מגבילים את השליפה עם השוויון בין השדות בכל הטבלאות, לעיתים נקבל יותר שורות מהרצוי (כפי שצויין קודם - מכפלה קרטזית)

הסוגריים סביב תנאי ה- or הכרחיים, כי ל and יש עדיפות גבוהה יותר.

נראה כעת את המבנה שבד"כ ישנו בסיפורות וגם בהרבה תוכנות, של פעולת ה- join

```
select st.id
from student st
      join borrow bo
      on st.id = bo.id
      join book bk
      on bo.isbn = bk.isbn
where ( bk.author = 'Bernard Cushing' or bk.author = 'Sherman Chow')
```

פעולת ה join היא קריטית לידיעת העבודה עם SQL - זה בעצם כל הרעיון מאחורי בסיסי נתונים יחסיים, שהרי כבר הזכרנו שבין הטבלאות השונות אין קשרים פיזיים, כך הקשר הוא לוגי בעזרת מפתחות ראשיים ומפתחות זרים.