מעבדה מס' 8 ותחילת 9

תת-שאילתות מתואמות (Corrlated Subqueries)

לעתים אנו רוצים שתת השאלתה תחושב הרבה פעמים, פעם אחת לכל השמה של ערך לביטוי בתת השאילתה – ערך המגיע מחוץ לתת השאלתה. תת שאלתה כזו נקראת מתואמת.

נריץ את הסקריפט Operations.txt ונוסיף לטבלה Table2 את שתי השורות:

```
INSERT INTO Table2 VALUES (2,4,7);
INSERT INTO Table2 VALUES (5,3,17);
```

לכן נקבל בטבלה:

Table2		
Num1	Num2	Num3
1	3	5
1	4	3
1	10	12
2	4	7
5	3	17

כעת נריץ את השאילתה הבאה:

```
SELECT Num2 FROM Table2 Dup WHERE Num1< ANY (SELECT Num1 FROM Table2 WHERE
Num2=Dup.Num2);
```

מה קיבלנו? את כל הערכים של Num2 שמופיעים יותר מפעם אחת.

1) בהינתן:

```
SELECT *
FROM PART

    PARTNUM DESCRIPTION                 PRICE
=========== ==================== ===========

         54 PEDALS                       54.25
         42 SEATS                        24.50
         46 TIRES                        15.25
         23 MOUNTAIN BIKE               350.45
         76 ROAD BIKE                   530.00
         10 TANDEM                     1200.00


SELECT *
FROM ORDERS

  ORDEREDON NAME           PARTNUM    QUANTITY REMARKS
=========== ========== =========== =========== =======

15-MAY-1996 TRUE WHEEL          23           6 PAID
19-MAY-1996 TRUE WHEEL          76           3 PAID
 2-SEP-1996 TRUE WHEEL          10           1 PAID
30-JUN-1996 TRUE WHEEL          42           8 PAID
30-JUN-1996 BIKE SPEC           54          10 PAID
30-MAY-1996 BIKE SPEC           10           2 PAID
30-MAY-1996 BIKE SPEC           23           8 PAID
17-JAN-1996 BIKE SPEC           76          11 PAID
17-JAN-1996 LE SHOPPE           76           5 PAID
 1-JUN-1996 LE SHOPPE           10           3 PAID
 1-JUN-1996 AAA BIKE            10           1 PAID
 1-JUL-1996 AAA BIKE            76           4 PAID
 1-JUL-1996 AAA BIKE            46          14 PAID
11-JUL-1996 JACKS BIKE          76          14 PAID
```

מה נקבל בשאילתה הבאה?

```
SELECT * FROM ORDERS O
WHERE 'ROAD BIKE' =
(SELECT DESCRIPTION FROM PART P WHERE P.PARTNUM = O.PARTNUM);
```

אגב, זהה לשאילתה הפשוטה:

```
SELECT O.ORDEREDON, O.NAME, O.PARTNUM, O.QUANTITY, O.REMARKS
FROM ORDERS O, PART P
WHERE P.PARTNUM = O.PARTNUM AND P.DESCRIPTION = 'ROAD BIKE';
```

הפלט:

```
  ORDEREDON NAME           PARTNUM    QUANTITY REMARKS
=========== ========== =========== =========== =======

19-MAY-1996 TRUE WHEEL          76           3 PAID
 1-JUL-1996 AAA BIKE            76           4 PAID
17-JAN-1996 LE SHOPPE           76           5 PAID
17-JAN-1996 BIKE SPEC           76          11 PAID
11-JUL-1996 JACKS BIKE          76          14 PAID
```

קיבלנו למעשה, את הזמנות פריט 76 (ROAD BIKE).

2) נסתכל על השאילתה הבאה שמחזירה את סכום המכירות של פריטים שנמכרו ביותר ממכירה אחת.

```
SELECT O.PARTNUM, SUM(O.QUANTITY*P.PRICE)
FROM ORDERS O, PART P WHERE P.PARTNUM = O.PARTNUM GROUP BY O.PARTNUM
HAVING SUM(O.QUANTITY*P.PRICE) >
(SELECT AVG(O1.QUANTITY*P1.PRICE)
FROM PART P1, ORDERS O1 WHERE P1.PARTNUM = O1.PARTNUM
                                AND P1.PARTNUM = O.PARTNUM);
```

```
    PARTNUM         SUM
=========== ===========

         10     8400.00
         23     4906.30
         76    19610.00
```

VIEWS (מבטים)

דנו בעבר במבטים. נענה על השאלה האם ניתן להוסיף שורות למבט. התשובה היא כן, בתנאי ש:

- השימוש הוא ב-SELECT ולא ב-SELECT DISTINCT
- R המתושאל אינו מופיע ב-WHERE כ-SubQuery.
- ב-SELECT של ה-VIEW יש מספיק שדות, כך שלכל שורה שנוסיף ל-VIEW נוכל לשים לשאר השדות בטבלה המקורית ערך NULL או ערך ברירת מחדל שהוגדר.

```
CREATE VIEW ThirdView AS
        SELECT Num1, Num2 FROM Table1 WHERE Num2>3;
```

ונוסיף שורה:

```
INSERT INTO ThirdView VALUES (1,7);
```

שימו לב שנוספה שורה לטבלת Table1.

אנלוגית, השורה:

```
DELETE FROM ThirdView WHERE Num2=10;
```

תמחק שורה בטבלה.

תוכלו בעצמכם לנסות לגבי UPDATE.

Joins

התמיכה המפורשת ב-Joins כוללת מעין גירסה של Outer Join. אם נכתוב:

```
SELECT * FROM Table1, Table2 WHERE Table1.Num2 = Table2.Num2;
```

הרי לא נראה בפלט שום זכר לשורות ב-Table1 שאינן מסכימות במה שהתבקש עם שום שורה ב-Table2.
אנלוגית, לא נראה בפלט שום זכר לשורות ב-Table2 שאינן מסכימות במה שהתבקש עם שום שורה ב-Table1.
לעתים נרצה כן לראות גם את השורות שאינן מסכימות. לצורך כך נכתוב:

```
SELECT * FROM Table1, Table2 WHERE Table1.Num2(+) = Table2.Num2;
SELECT * FROM Table1, Table2 WHERE Table1.Num2 = Table2.Num2(+);
```

משמעות השורה הראשונה: בנוסף לפלט הרגיל שאנו מצפים מהשאילתה, הצג גם את השורות של Table2 שאינן מסכימות במה שהתבקש עם שום שורה ב-Table1, כאשר ערכי השדות של Table1 מרופדים ב-NULLS.
משמעות השורה הראשונה: בנוסף לפלט הרגיל שאנו מצפים מהשאילתה, הצג גם את השורות של Table1 שאינן מסכימות במה שהתבקש עם שום שורה Table2, כאשר ערכי השדות של Table2 מרופדים ב-NULLS.

הערה: לא ניתן להשתמש ב(+) משני הצדדים יחד.

אילוצים (Constrains)

Primary Key & Keys

דנו בעבר במפתח ראשי (Primary Key), והדרכים להגדרתו. ניתן גם להשתמש במלה UNIQUE ולשלבה **בארבע הגישות שראינו להגדרת מפתח ראשי**. המשמעות של UNIQUE זהה לזו של Primary Key. אך ניתן להגדיר כמה Keys (UNIQUE) ורק Primary Key אחד לטבלה. נראה דוגמה:

```
Create Table Empl (SSN INTEGER NOT NULL, Emp# INTEGER NOT NULL, Name VARCHAR(30),
                   PRIMARY KEY (SSN), UNIQUE (Emp#));
```

Foreign Keys

דנו בעבר במפתח זר (Foreign Key). נציג את ארבע דרכים להגדרתו (בדומה למפתח ראשי) (בהנחה ש-Table1 מוגדרת כנדרש):

```
CREATE TABLE Table2 (j INTEGER REFERENCES Table1(i));

CREATE TABLE Table2 (j INTEGER,
                     FOREIGN KEY (j) REFERENCES Table1 (i));

CREATE TABLE Table2 (j INTEGER,
                     CONSTRAINT cc FOREIGN KEY (j) REFERENCES Table1 (i));

CREATE TABLE Table2 (j INTEGER);
ALTER TABLE Table2 ADD CONSRAINT cc FOREIGN KEY (j) REFERENCES Table1(i);
```

נשים לב כי הדרך הראשונה היא הפשוטה ביותר אך אינה מאפשרת מפתח של יותר משדה אחד. הדרך השנייה כן מאפשרת זאת. הדרך השלישית מאפשרת מפתח של יותר משדה אחד ונותנת שם לאילוץ המפתח הזר , על מנת שנוכל לבטלו בהמשך. הדרך הרביעית מאפשרת הגדרת המפתח הזר לאחר הגדרת הטבלה, וכמובן מאפשרת מפתח של יותר משדה אחד.

נשים לב, כי ניתן להגדיר Foreign Key רק בעזרת שדה(שדות) שהוא(שהם) Primary key, אך לא בעזרת שדה (שדות) שהוא (הם) Unique.

<u>התנהגות המפתח הזר במקרה של שינויים</u>

עלינו לציין כיצד יתנהג השדה שמקבל ערכים ממפתח ראשי בטבלה אחרת, במקרה של שינויים במפתח הראשי. למשל אם נחזור לטבלת הסטודנטים, הקורסים והסטודנטים הרשומים לקורס נוכל לציין:

1) מה אם אנו **מוחקים** קורס/סטודנט (מטבלת Courses/Students) הרשום בטבלת Student_in_Course. ברירת המחדל היא **דחיית** המחיקה. אך האפשרויות האחרות הן:

א) מחיקת כל הרשומות הרלוונטיות ב-Student_in_Course. לשם כך נכתוב:
ON DELETE CASCADE.

ב) השמת ערך NULL לשדה הרלוונטי ברשומות הרלוונטיות ב-Student_in_Course . לשם כך נכתוב:
ON DELETE SET NULL.

2) מה אם אנו **מעדכנים** קורס/סטודנט (מטבלת Courses/Students) הרשום בטבלת Student_in_Course. ברירת המחדל היא **דחיית** העדכון. אך האפשרויות האחרות הן:

א) עדכון בהתאמה לשדה הרלוונטי ברשומות הרלוונטיות ב-Student_in_Course . לשם כך נכתוב:
ON UPDATE CASCADE

ב) השמת ערך NULL לשדה הרלוונטי ברשומות הרלוונטיות ב-Student_in_Course . לשם כך נכתוב:
ON UPDATE SET NULL.

מצורפת דוגמה, **תוכלו להציץ בה בזמנכם החופשי!!!**

טבלת Students		
sname	snumber	Department
Moshe	1	SE
David	2	SE
Avraham	3	SE

טבלת Courses		
cname	Cnumber	Department
Databases	1	SE
Graphics	2	SE
Networks	3	SE

טבלת Student_in_Course	
ssnumber	Ccnumber
1	1
1	2
2	2
2	3
3	1
3	3

אם נרצה למחוק את משה מטבלת הסטודנטים, הרי ש:

אם בחרנו באפשרות 1א נקבל לאחר המחיקה:

טבלת Students		
sname	snumber	Department
Moshe	1	SE
David	2	SE
Avraham	3	SE

טבלת Courses		
cname	Cnumber	Department
Databases	1	SE
Graphics	2	SE
Networks	3	SE

טבלת Student_in_Course	
ssnumber	Ccnumber
2	2
2	3
3	1
3	3

אם בחרנו באפשרות 1ב נקבל לאחר המחיקה:

טבלת Students		
sname	snumber	Department
Moshe	1	SE
David	2	SE
Avraham	3	SE

טבלת Courses		
cname	cnumber	Department
Databases	1	SE
Graphics	2	SE
Networks	3	SE

טבלת Student_in_Course	
ssnumber	Ccnumber
NULL	1
NULL	2
2	2
2	3
3	1
3	3

אם נרצה לעדכן את מספרו של משה ל-4 הרי ש:

אם בחרנו באפשרות 2א נקבל לאחר העדכון:

טבלת Students				טבלת Courses				טבלת Student_in_Course	
sname	snumber	Department		Cname	cnumber	Department		ssnumber	Ccnumber
Moshe	4	SE		Databases	1	SE		4	1
David	2	SE		Graphics	2	SE		4	2
Avraham	3	SE		Networks	3	SE		2	2
								2	3
								3	1
								3	3

אם בחרנו באפשרות 2ב נקבל לאחר העדכון:

טבלת Students				טבלת Courses				טבלת Student_in_Course	
sname	snumber	Department		Cname	cnumber	Department		ssnumber	Ccnumber
Moshe	4	SE		Databases	1	SE		NULL	1
David	2	SE		Graphics	2	SE		NULL	2
Avraham	3	SE		Networks	3	SE		2	2
								2	3
								3	1
								3	3

הערה חשובה: האפשרויות ON UPDATE CASCADE, ON UPDATE SET NULL לא נתמכות ב-Oracle.

דוגמה לשימוש באפשרויות 1א ו-1ב. נגדיר שוב את טבלת Student_in_Course:

```
CREATE TABLE Student_in_Course (ssnumber NUMBER,
                                ccnumber NUMBER,
                                PRIMARY KEY (ssnumber, ccnumber),
                                FOREIGN KEY (ssnumber) REFERENCES Students (snumber) ON
                                DELETE CASCADE,
                                FOREIGN KEY (ccnumber) REFERENCES Courses (cnumber) ON
                                DELETE SET NULL);
```

אילוצים על ערכי שדות

ראינו אילוצי Keys ו-Foreign Keys. ראינו גם אילוץ NOT NULL. נראה כעת אילוצים בעזרת CHECK. בפרקטיקה, אילוץ CHECK קובע אנמרציה של ערכים או איזשהו אי שוויון אריתמטי. אך ככלל, ניתן לתת כל ביטוי, מסובך כמה שיהיה, שיכול להופיע לאחר WHERE בשאילתה. התנאי יכול להתייחס אפילו ל-Relation אחר המופיע ב-FROM של SubQuery.
התנאי נבדק בכל פעם ששורה מקבלת ערך חדש ל-attribute שעליו יש CHECK. אם ישנה הפרה, המודיפיקציה נדחית.
נראה דוגמאות:

```
CREATE TABLE Table3 (Num1 INTEGER,
                     Num2 INTEGER CHECK (Num2<70),
                     Ch VARCHAR(1) CHECK (Ch IN ('A', 'B')),
                     PRIMARY KEY (Num1, Num2),
                     FOREIGN KEY (Num1, Num2) REFERENCES Table1(Num1, Num2));
```

נראה דוגמה לתנאי שמתייחס ל-Relation אחר המופיע ב-FROM של SubQuery. בהגדרה הבסיסית של Student_In_Course, הגדרנו:

```
CREATE TABLE Student_in_Course (ssnumber NUMBER,
                                ccnumber NUMBER,
```

```
PRIMARY KEY (ssnumber, ccnumber),
FOREIGN KEY (ssnumber) REFERENCES Students (snumber),
FOREIGN KEY (ccnumber) REFERENCES Courses (cnumber));
```

ננסה **"לחקות"** את אילוץ ה-Foreign Key בעזרת CHECK:

```
CREATE TABLE Student_in_Course (ssnumber NUMBER,
ccnumber NUMBER CHECK (ccnumber IN (SELECT cnumber
FROM Courses)),
PRIMARY KEY (ssnumber, ccnumber),
FOREIGN KEY (ssnumber) REFERENCES Students (snumber));
```

האילוץ על ccnumber:

- ידאג לכך שכשנוסיף שורה בטבלה זו, אם הערך של ccnumber אינו מספר קורס בטבלת הקורסים, ההוספה תידחה.
- ידאג לכך שכשנעדכן ערך של ccnumber באחיד השורות, אם הערך אינו מספר קורס בטבלת הקורסים, העדכון יידחה.
- **אך**, מה יקרה כשנמחק שורה בטבלת Courses? מדוע?

לכן החיקוי לא הצליח.