בסיסי נתונים

SQL

*מבוסס על מדריך הלמידה של האוניברסיטה הפתוחה

תוכן

שאילתות

התאמת מחרוזות

מיון

עדכון בסיס נתונים – הוספה, מחיקה, עדכון, שינוי

בדיקת ערכים ריקים

משתני שורה

פעולות על קבוצות

כינון

כפילויות

צירוף טבעי

אילוצים

הרחבה

NO SQL

E.g. MongoDB, Cassandra, Redis, HBase, …

Db.SourceCollection.aggregate([$project:{<fields to show>}},
{$unwind:<array field>}, {$match:{<match criteria>}},
{$group:{_id:{<group by fields>}, count:{$sum:1}}}, …

Not **O**nly **SQL**

כי יש עוד דברים חוץ מ SQL

מבנה בסיסי של שאילתה

SELECT A1, A2, ... תכונות

FROM r1, r2, ... יחסים

WHERE ψ פרדיקט לוגי או נוסחא

תכונות לקבל בתוצאה

יחסים להשתמש כדי להגיע לתשובה

תנאים לוגיים על שורות ביחסים

$$\Pi_{A1,A2,\ldots An}(\sigma_{\psi}(r_1 \times r_2 \times \ldots \times r_m))$$

* במקום רשימת התכונות כדי לקבל את כולן.

דוגמא

Area (a_name, a_location, a_type, a_size)

Plant (p_name, p_type, p_diameter, p_maxheight)

Planting (p_name, a_name, date, amount)

רשימת שמות הצמחים בבסיס הנתונים:

SELECT p_name FROM plant

דוגמא תנאי

Area (a_name, a_location, a_type, a_size)

Plant (p_name, p_type, p_diameter, p_maxheight)

Planting (p_name, a_name, date, amount)

רשימת כל עצי הפרי:

SELECT p_name FROM plant WHERE p_type="עץ פרי"

שליפת מידע מכמה יחסים

Area (a_name, a_location, a_type, a_size)

Plant (p_name, p_type, p_diameter, p_maxheight)

Planting (p_name, a_name, date, amount)

המקומות (location) שנשתלו בהם עצי חרוב

להבחין בין הופעות תכונה בכמה יחסים רושמים את שם היחס נקודה ושם התכונה.

```
SELECT a_location
FROM area, planting
WHERE area.a_name = planting.a_name AND p_name = "חרוב מצוי"
```

מכפלה קרטזית X Cartesian product

המכפלה הקרטזית rxs של שני יחסים עם תבנית r1,r2,... וכן s1,s2, ... היא איחוד התבניות הכוללת את כל השירשורים האפשריים של n-יות מ S, וn-יות מ R.

מקביל ליכולת לפעול על מספר יחסים ב SQL.

מאפשרת לשלב מידע מכמה יחסים. r x s

R1	R2
ra	rb
rc	rd
re	rf

S1	S2	S3
sa	sb	Sc
sd	se	Sf

מכפלה קרטזית Cartesian product

המכפלה הקרטזית rxs של שני יחסים עם תבנית r1,r2,... וכן s1,s2, ... היא איחוד התבניות הכוללת את כל השירשורים האפשריים של n-יות מ S, ו n-יות מ R.

מאפשרת לשלב מידע מכמה יחסים. r x s

R1	R2	S1	S2	S3
ra	rb	sa	sb	Sc
rc	rd	sa	sb	Sc
re	rf	sa	sb	Sc
ra	rb	sd	se	Sf
rc	rd	sd	se	Sf
re	rf	sd	se	Sf

מכפלה קרטזית Cartesian product

דוגמא: מצאו את הרחובות בהם גרים לקוחות עם הפקדות של 500 ומעלה

customer

Customer-name	Street	Customer-city
Mor	Pinkas	Rishon
Tami	Alenby	Haifa
Avivi	Pinkas	Rishon
Owen	Alenby	Haifa

deposit

Branch-name	Account number	Customer-name	amount
Hamerkaz	101	Owen	500
Pinkas	215	Tami	700
Aviv	102	Avivi	400

מכפלה קרטזית Cartesian product

דוגמא: מצאו את הרחובות בהם גרים לקוחות עם הפקדות של 500 ומעלה

customer x Deposit תוצאת ביניים

Customer-name	Street	Customer-city	Branch-name	Account number	Customer-name	amount
Mor	Pinkas	Rishon	Hamerkaz	101	Owen	500
Tami	Alenby	Haifa	Hamerkaz	101	Owen	500
Avivi	Pinkas	Rishon	Hamerkaz	101	Owen	500
Owen	Alenby	Haifa	Hamerkaz	101	Owen	500
Mor	Pinkas	Rishon	Pinkas	215	Tami	700
Tami	Alenby	Haifa	Pinkas	215	Tami	700
Avivi	Pinkas	Rishon	Pinkas	215	Tami	700
Owen	Alenby	Haifa	Pinkas	215	Tami	700
Mor	Pinkas	Rishon	Aviv	102	Avivi	400

מכפלה קרטזית Cartesian product

דוגמא: מצאו את הרחובות בהם גרים לקוחות עם הפקדות של 500 ומעלה

$$\pi_{street}(\sigma_{customer.cust\_name = deposit.cust\_name \wedge amount >= 500}(customerX\ deposit))$$

Customer-name	Street	Customer-city	Branch-name	Account number	Customer-name	amount
Owen	Alenby	Haifa	Hamerkaz	101	Owen	500
Tami	Alenby	Haifa	Pinkas	215	Tami	700

Street
Alenby
Alenby

מתן שם לתכונה ביחס התוצאה

Area (a_name, a_location, a_type, a_size)

Plant (p_name, p_type, p_diameter, p_maxheight)

Planting (p_name, a_name, date, amount)

אפשר לקבוע לתכונה ביחס התוצאה שם חדש, שונה משם התכונה המקורית.

SELECT a_location **as haruv_location**
FROM area, planting
WHERE area.a_name = planting.a_name AND p_name = "חרוב מצוי"

מתן שם לתכונה ביחס התוצאה

Area (a_name, a_location, a_type, a_size)

Plant (p_name, p_type, p_diameter, p_maxheight)

Planting (p_name, a_name, date, amount)

אפשר לקבוע לתכונה ביחס התוצאה שם חדש, שונה משם התכונה המקורית, נשתמש ב

SELECT attribute **AS** new_attribute_name

SELECT attribute1 **AS** new_attribute_name1, attribute2 **AS** new_attribute_name2

SELECT a_location **AS haruv_location**
FROM area, planting
WHERE area.a_name = planting.a_name AND p_name = "חרוב מצוי"

ביטוי חשבוני ב SELECT

Area (a_name, a_location, a_type, a_size)

Plant (p_name, p_type, p_diameter, p_maxheight)

Planting (p_name, a_name, date, amount)

ניתן לרשום ביטוי חשבוני ב SELECT

דוגמא: מצא מהו השטח הדרוש לשתילת 5 עצי פרי לכל עץ פרי

$S=\pi r^2= \pi(d/2)^2 = \pi d^2/4$

```
SELECT  p_name, (5*3.14*p_diameter*p_diameter/4) as space_for_5_fruit_trees
FROM plant
WHERE p_type= "עץ פרי"
```

התאמת תבניות במחרוזות

Area (a_name, a_location, a_type, a_size)

Plant (p_name, p_type, p_diameter, p_maxheight)

Planting (p_name, a_name, date, amount)

משתמשים במילה like במקום ב =

% מציין תת-מחרוזת כלשהי (לעיתים *)

_ מציין תו כלשהו (לעיתים [*])

דוגמא: מצא את שמות כל העצים (עץ פרי או עץ נוי)

```
SELECT p_name AS tree_name
FROM plant
WHERE p_type like "%עץ"
```

מיון שורות ביחס התוצאה

Area (a_name, a_location, a_type, a_size)

Plant (p_name, p_type, p_diameter, p_maxheight)

Planting (p_name, a_name, date, amount)

ניתן למיין את הערכים ע"פ תכונה אחת (או יותר) בסוף השאילתה ע"י ORDER BY. אם לא נציין, המיון בסדר עולה asc, אם רוצים סדר יורד נוסיף desc אחרי שם התכונה.

דוגמא: ערכו רשימה של שמות העצים, ממוינים לפי גובהם בסדר יורד.

```
SELECT p_name FROM plant
WHERE p_type like "%עץ"
ORDER BY  p_maxheight desc
```

מיון שורות ביחס התוצאה 2

Area (a_name, a_location, a_type, a_size)

Plant (p_name, p_type, p_diameter, p_maxheight)

Planting (p_name, a_name, date, amount)

דוגמא: ערוך רשימה של שמות האיזורים (ממוינים לפי סדר אלפביתי עולה) ובכל אחת את הצמחים השתולים בה ממוינים בסדר יורד לפי גובהם.

```
SELECT a_name, plant.p_name
FROM planting, plant
WHERE planting.p_name = plant.p_name
ORDER BY a_name asc, p_maxheight desc
```

משתני שורה

Area (a_name, a_location, a_type, a_size)

Plant (p_name, p_type, p_diameter, p_maxheight)

Planting (p_name, a_name, date, amount)

ניתן להגדיר משתנה שורה ולהתייחס אליו ב WHERE למען פישוט השאילה. לכל יחס המופיע ב FROM. שימוש עיקרי, הגדרת יותר ממשתנה אחד לאותו יחס. יש להקפיד על ההצמדה למשתנים.

דוגמא: מצא איזורים עם צמחים משני סוגים שונים לפחות

```
SELECT p1.a_name
FROM planting as p1, planting as p2
WHERE p1.a_name = p2.a_name AND p1.p_name != p2.p_name
```

IS NULL

Area (a_name, a_location, a_type, a_size)

Plant (p_name, p_type, p_diameter, p_maxheight)

Planting (p_name, a_name, date, amount)

ערכי NULL ההם שונים מ 0, הם שומר מקום לערכים לא רלוונטים. לא ניתן לשתמש בהשוואות = > <, נשתמש ב IS NULL או ב IS NOT NULL.

דוגמא: מצא צמחים ללא גובה מקסימלי

```
SELECT p_name
FROM plants
WHERE p_maxheight IS NULL
```

כפילויות

Area (a_name, a_location, a_type, a_size)

Plant (p_name, p_type, p_diameter, p_maxheight)

Planting (p_name, a_name, date, amount)

בטבלה יתכנו שורות כפולות. לא תמיד צריך למחוק. אבל מחיקה מתבצעת ע"י הוספת DISTINCT בפסוק SELECT.

דוגמא: כל איזורי השתילה המופיעים ביחס שתילה

```
SELECT DISTINCT a_name
FROM planting
```

פעולות על קבוצות

כשיחסי התוצאה של שתי שאילתות תואמים (תבנית זהה) אפשר להפעיל פעולות איחוד, חיתוך, הפרש ולקבל שאילתה מורכבת יותר.

חיתוך INTERSECT - JOIN ב MySQL

איחוד UNION

הפרש EXCEPT /MINUS

דוגמא: מצא את שמות כל העצים (בדרך של איחוד)

```
(SELECT p_name FROM plant WHERE p_type="עץ פרי")
UNION
(SELECT p_name FROM plant WHERE p_type="עץ נוי")
```

JOIN – דרכים זהות

FROM table1,table2

WHERE table1.att1 = table2.att1

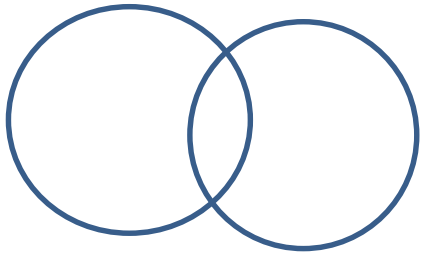
FROM table1 JOIN table2 ON table1.att1 = table2.att1

פרדיקט – כלומר ההשוואה בן שדות – לא חייבים להיות זהים

אפשר גם שניים.

FROM table1 JOIN table2 USING (att1)

JOIN



הפעלת INNER JOIN היא על תנאי או שדה כלשהו.

FROM table1 JOIN table2 ON predicate

FROM Table 1 JOIN table2 USING (attribute)

MySQL

Inner Join / Join – שורות עם התאמה בשתי הטבלאות.

Left Outer Join – כל השורות מהטבלה השמאלית + כל השורות מהטבלה הימנית שיש להן התאמה בטבלה השמאלית

Right Outer Join – כל השורות בטבלה הימנית + כל השורות מהטבלה השמאלית שיש להן התאמה בטבלה הימנית

Full (Outer) Join – כל השורות שיש להן התאמה מאוחדות וכן שורות שאין בהם התאמה אך נמצאות באחת הטבלאות

***ציון התאמה בעמודות מסוימות נעשה ע"י ON**

ההבדל בין איחוד UNION וחיתוך JOIN

פעולת האיחוד מוסיפה שורות משתי טבלאות כאשר התכונות שלהן זהות.

פעולת החיתוך JOIN מבצעת מכפלה קרטזית של שתי טבלאות

כאשר ניתן להוסיף פרדיקט או עמודות בהן צריכה להיות התאמה (ע"י ()ON או ()USING)

INNER JOIN/JOIN – יביא רק את השורות בהן היתה התאמה

LEFT OUTER JOIN – יאחד שורות שהיתה בהן התאמה ואת כל השורות מהטבלה השמאלית

RIGHT OUTER JOIN – יאחד שורות שהיתה בהן התאמה ואת כל השורות מהטבלה הימנית.

JOIN FULL – יאחד שורות שהיתה בהן התאמה, וגם יביא את כל שאר השורות.

חיתוך JOIN

customer

Cname	Street	Account No
Mor	Pinkas	101
Tami	Alenby	215
Avivi	Pinkas	102
Owen	Alenby	111

deposit

Branch-name	CusName	amount
Hamerkaz	Owen	500
Pinkas	Tami	700
Aviv	Avivi	400
Hadera	Scot	555

Cname	Street	AccountNo	amount	Branch-name	CusName

חיתוך JOIN

customer

Cname	Street	Account No
Mor	Pinkas	101
Tami	Alenby	215
Avivi	Pinkas	102
Owen	Alenby	111

deposit

Branch-name	CusName	amount
Hamerkaz	Owen	500
Pinkas	Tami	700
Aviv	Avivi	400
Hadera	Scot	555

Cname	Street	AccountNo	amount	Branch-name	CusName
		101			
		215			
		102			
		111			

חיתוך JOIN

customer

Cname	Street	Account No
Mor	Pinkas	101
Tami	Alenby	215
Avivi	Pinkas	102
Owen	Alenby	111

deposit

Branch-name	CusName	amount
Hamerkaz	Owen	500
Pinkas	Tami	700
Aviv	Avivi	400
Hadera	Scot	555

Cname	Street	AccountNo	amount	Branch-name	CusName
Mor	Pinkas	101			
Tami	Alenby	215			
Avivi	Pinkas	102			
Owen	Alenby	111			

חיתוך JOIN

customer

Cname	Street	Account No
Mor	Pinkas	101
Tami	Alenby	215
Avivi	Pinkas	102
Owen	Alenby	111

deposit

Branch-name	CusName	amount
Hamerkaz	Owen	500
Pinkas	Tami	700
Aviv	Avivi	400
Hadera	Scot	555

Cname	Street	AccountNo	amount	Branch-name	CusName
Mor	Pinkas	101			
Tami	Alenby	215			
Avivi	Pinkas	102			
Owen	Alenby	111			

חיתוך JOIN

customer

Cname	Street	Account No
Mor	Pinkas	101
Tami	Alenby	215
Avivi	Pinkas	102
Owen	Alenby	111

deposit

Branch-name	CusName	amount
Hamerkaz	Owen	500
Pinkas	Tami	700
Aviv	Avivi	400
Hadera	Scot	555

Cname	Street	AccountNo	amount	Branch-name	CusName
Mor	Pinkas	101			
Tami	Alenby	215	700	Pinkas	Tami
Avivi	Pinkas	102			
Owen	Alenby	111			

חיתוך JOIN

customer

Cname	Street	Account No
Mor	Pinkas	101
Tami	Alenby	215
Avivi	Pinkas	102
Owen	Alenby	111

deposit

Branch-name	CusName	amount
Hamerkaz	Owen	500
Pinkas	Tami	700
Aviv	Avivi	400
Hadera	Scot	555

Cname	Street	AccountNo	amount	Branch-name	CusName
Mor	Pinkas	101			
Tami	Alenby	215	700	Pinkas	Tami
Avivi	Pinkas	102	400	Aviv	Avivi
Owen	Alenby	111	500	Hamerkaz	Owen
			555	Hadera	Scot

חיתוך INNER JOIN

customer

Cname	Street	Account No
Mor	Pinkas	101
Tami	Alenby	215
Avivi	Pinkas	102
Owen	Alenby	111

deposit

Branch-name	CusName	amount
Hamerkaz	Owen	500
Pinkas	Tami	700
Aviv	Avivi	400
Hadera	Scot	555

Cname	Street	AccountNo	amount	Branch-name	CusName
Mor	Pinkas	101			
Tami	Alenby	215	700	Pinkas	Tami
Avivi	Pinkas	102	400	Aviv	Avivi
Owen	Alenby	111	500	Hamerkaz	Owen
			555	Hadera	Scot

חיתוך OUTER JOIN

customer

Cname	Street	Account No
Mor	Pinkas	101
Tami	Alenby	215
Avivi	Pinkas	102
Owen	Alenby	111

deposit

Branch-name	CusName	amount
Hamerkaz	Owen	500
Pinkas	Tami	700
Aviv	Avivi	400
Hadera	Scot	555

Cname	Street	AccountNo	amount	Branch-name	CusName
Mor	Pinkas	101			
Tami	Alenby	215	700	Pinkas	Tami
Avivi	Pinkas	102	400	Aviv	Avivi
Owen	Alenby	111	500	Hamerkaz	Owen
			555	Hadera	Scot

חיתוך LEFT OUTER JOIN

customer

Cname	Street	Account No
Mor	Pinkas	101
Tami	Alenby	215
Avivi	Pinkas	102
Owen	Alenby	111

deposit

Branch-name	CusName	amount
Hamerkaz	Owen	500
Pinkas	Tami	700
Aviv	Avivi	400
Hadera	Scot	555

Cname	Street	AccountNo	amount	Branch-name	CusName
Mor	Pinkas	101			
Tami	Alenby	215	700	Pinkas	Tami
Avivi	Pinkas	102	400	Aviv	Avivi
Owen	Alenby	111	500	Hamerkaz	Owen
			555	Hadera	Scot

חיתוך RIGHT OUTER JOIN

customer

Cname	Street	Account No
Mor	Pinkas	101
Tami	Alenby	215
Avivi	Pinkas	102
Owen	Alenby	111

deposit

Branch-name	CusName	amount
Hamerkaz	Owen	500
Pinkas	Tami	700
Aviv	Avivi	400
Hadera	Scot	555

Cname	Street	AccountNo	amount	Branch-name	CusName
Mor	Pinkas	101			
Tami	Alenby	215	700	Pinkas	Tami
Avivi	Pinkas	102	400	Aviv	Avivi
Owen	Alenby	111	500	Hamerkaz	Owen
			555	Hadera	Scot

חיתוך INNER JOIN

customer

Cname	Street	Account No
Mor	Pinkas	101
Tami	Alenby	215
Avivi	Pinkas	102
Owen	Alenby	111

deposit

Branch-name	CusName	amount
Hamerkaz	Owen	500
Pinkas	Tami	700
Aviv	Avivi	400
Hadera	Scot	555

AccountNo	amount
215	700
102	400
111	500

INNER JOIN – יביא רק את השורות בהן היתה התאמה

```
SELECT accountNo, amount
FROM customer JOIN deposit
ON customer.Cname = deposit.CusName
```

חיתוך LEFT OUTER JOIN

customer

Cname	Street	Account No
Mor	Pinkas	101
Tami	Alenby	215
Avivi	Pinkas	102
Owen	Alenby	111

deposit

Branch-name	CusName	amount
Hamerkaz	Owen	500
Pinkas	Tami	700
Aviv	Avivi	400
Hadera	Scot	555

AccountNo	amount
215	700
102	400
111	500
101	

LEFT OUTER JOIN – יאחד שורות שהיתה בהן התאמה
ואת כל השורות מהטבלה השמאלית

```
SELECT accountNo, amount
FROM customer LEFT OUTER JOIN deposit
ON customer.Cname = deposit.CusName
```

חיתוך RIGHT OUTER JOIN

customer

Cname	Street	Account No
Mor	Pinkas	101
Tami	Alenby	215
Avivi	Pinkas	102
Owen	Alenby	111

deposit

Branch-name	CusName	amount
Hamerkaz	Owen	500
Pinkas	Tami	700
Aviv	Avivi	400
Hadera	Scot	555

AccountNo	amount
215	700
102	400
111	500
	555

RIGHT OUTER JOIN – יאחד שורות שהיתה בהן התאמה ואת כל השורות מהטבלה הימנית.

```
SELECT accountNo, amount
FROM customer RIGHT OUTER JOIN deposit
ON customer.Cname = deposit.CusName
```

חיתוך FULL JOIN

customer

Cname	Street	Account No
Mor	Pinkas	101
Tami	Alenby	215
Avivi	Pinkas	102
Owen	Alenby	111

deposit

Branch-name	CusName	amount
Hamerkaz	Owen	500
Pinkas	Tami	700
Aviv	Avivi	400
Hadera	Scot	555

AccountNo	amount
215	700
102	400
111	500
	555
101	

JOIN FULL – יאחד שורות שהיתה בהן התאמה, וגם יביא את כל שאר השורות.

```
SELECT accountNo, amount
FROM customer FULL JOIN deposit
ON customer.Cname = deposit.CusName
```

שאילתות משלימות

Area (a_name, a_location, a_type, a_size)

Plant (p_name, p_type, p_diameter, p_maxheight)

Planting (p_name, a_name, date, amount)

אפשר תמיד גם לנסח שאילתה משלימה. ביחסי קבוצות.

דוגמא: מצא את שמות הצמחים שאינם עצים

```
(SELECT p_name FROM plant)
MINUS
(SELECT p_name FROM plant
WHERE p_type="עץ נוי" OR p_type  ="עץ פרי")
```

פונקציות הקבצה

פונקציות המבצעות חישובים עבור קבוצה שלמה של ערכים

Count – מניה של מספר הערכים בקבוצה

Min – המינימום בקבוצה

Max – המקסימום בקבוצה

Sum – סכום האיברים בקבוצה

Avg – ממוצע האיברים בקבוצה

שאלה: באילו נדרשים ערכים מספריים?

פונקציות הקבצה

Area (a_name, a_location, a_type, a_size)

Plant (p_name, p_type, p_diameter, p_maxheight)

Planting (p_name, a_name, date, amount)

דוגמא: מצא את הגובה הרב ביותר אליו עשוי צמח להגיע

```
SELECT max (p_maxheight)
FROM plant
```

פונקציות הקבצה

Area (a_name, a_location, a_type, a_size)

Plant (p_name, p_type, p_diameter, p_maxheight)

Planting (p_name, a_name, date, amount)

דוגמא: מספר החלקות השונות מתוך היחס planting

```
SELECT count (distinct a_name)
FROM planting
```

פונקציות הקבצה

Area (a_name, a_location, a_type, a_size)

Plant (p_name, p_type, p_diameter, p_maxheight)

Planting (p_name, a_name, date, amount)

חישוב יכול להיות גם על חלק מהשורות.

דוגמא: מספר החלקות ששטחן לפחות 2000 מטר

```
SELECT count (a_name)
FROM area
WHERE a_size >=2000
```

קיבוץ שורות ביחס התוצאה GROUP BY

Area (a_name, a_location, a_type, a_size)

Plant (p_name, p_type, p_diameter, p_maxheight)

Planting (p_name, a_name, date, amount)

השימוש הנוח הוא לחלק טבלה לתת קבוצות של שורות ע"פ קריטריון, ולחשב פונקציה על כל קבוצה בנפרד.

GROUP BY לפי שם תכונה או כמה תכונות. כל השורות שיש להן אותו ערך בתכונה או תכונות יקובצו

דוגמא: כמה צמחים מכל סוג יש

```
SELECT p_type, count(p_name) as sum_plants FROM plants
GROUP BY p_type
```

קיבוץ שורות ביחס התוצאה GROUP BY

Area (a_name, a_location, a_type, a_size)

Plant (p_name, p_type, p_diameter, p_maxheight)

Planting (p_name, a_name, date, amount)

ניתן לכלול גם פסוק WHRE בנוסף להקבצה, שיגביל את מספר השורות בתוצאה.

דוגמא: כמה צמחים **שגובהם מעל** מטר יש בכל אחד מסוגי הצמחים?

```
SELECT p_type, count(p_name) as num_high_plants
FROM plants
WHERE p_maxheight > 1
GROUP BY p_type
```

קיבוץ שורות ביחס התוצאה GROUP BY

Area (a_name, a_location, a_type, a_size)

Plant (p_name, p_type, p_diameter, p_maxheight)

Planting (p_name, a_name, date, amount)

אם ב FROM יש כמה יחסים, ההקבצה תחלק את המכפלה הקרטזית שלהם לתת-קבוצות.

דוגמא: מצא כמה שתילים של צמחים מסוג "שיח" נשתלו בכל חלקה?

```
SELECT a_name, sum(amount) FROM planting, plant
WHERE planting.p_name = plant.p_name AND p_type = "שיח"
GROUP BY a_name
```

קיבוץ שורות ביחס התוצאה GROUP BY

Area (a_name, a_location, a_type, a_size)

Plant (p_name, p_type, p_diameter, p_maxheight)

Planting (p_name, a_name, date, amount)

אם ב FROM יש כמה יחסים, ההקבצה תחלק את המכפלה הקרטזית שלהם לתת-קבוצות.

דוגמא: מצא כמה שתילים של צמחים מסוג "שיח" נשתלו בכל חלקה?

```
SELECT a_name, sum(amount) FROM planting JOIN plant USING(p_name)
WHERE p_type = "שיח"
GROUP BY a_name
```

תנאים ביחס התוצאה

אפשר לנסח תנאי לוגי שיילקח בחשבון **אחרי** יצירת תת הקבוצות. הוא מנוסח בפסוק **HAVING** אחרי GROUP BY, בדומה ל WHERE. אך חייבת להיות לו משמעות "קבוצתית" – אפשרות לבדוק לגבי תת קבוצה.

לא ניתן לבצע הרכבה של הקבצה

דוגמא: מצא שמות חלקות שהפרש הגבהים בין הצמח הגבוה והנמוך ביותר שנשתלו בהם יכול לעלות על 2 מטרים.

```
SELECT a_name FROM planting JOIN plant USING(p_name)
GROUP BY a_name
HAVING (max(p_maxheight)-min(p_maxheight)) > 2.00
```

השוואה בין ערך לקבוצת ערכים

V in R – אמת אם הערך V כלול בין איברי הקבוצה R

V not in R – אמת אם V אינו ב R

V = ANY R – בודק אם V שווה בערכו לאיבר כלשהו ב R (דומה לקודם)

V >ANY R – האם V גדול מאיבר כלשהו ב R

V > ALL R – בודק אם V גדול בערכו מכל איבר ב R

V !=ALL R – האם V שונה בערכו מכל איבר ב R (שקול ל NOT IN)

SOME, ANY ו ALL יכולים להופיע עם כל צירוף =< ,=> < ,> ,=! =, וניתן להוסיף NOT

השוואה בין ערך לקבוצה

Area (a_name, a_location, a_type, a_size)

Plant (p_name, p_type, p_diameter, p_maxheight)

Planting (p_name, a_name, date, amount)

דוגמא: מצא את שמות העצים.

ניתן לכתוב שאילתות במספר דרכים.

```
SELECT p_name
FROM plant
WHERE p_type IN ("עץ פרי", "עץ נוי")
```

קינון

Area (a_name, a_location, a_type, a_size)

Plant (p_name, p_type, p_diameter, p_maxheight)

Planting (p_name, a_name, date, amount)

קינון שאילתות, קבוצת ערכים יכולה להתקבל משאילתה פנימית.

דוגמא: מצא את שמות העצים

```
SELECT p_name
FROM plant
WHERE p_type IN (SELECT  p_type FROM plant
                 WHERE p_type LIKE "%עץ")
```

קינון – דוגמא סטודנטים

Student (SID, Name, dept, year)

Courses (CID, Name, Credit, dept)

Studies (SID, CID)

שאלה: מיהם הסטודנטים שלומדים במדעי המחשב וגם לומדים קורס כלשהוא של עיצוב

```
SELECT SID FROM Student JOIN Studies Using (SID)
WHERE dept = "מדעי המחשב" AND
CID IN (SELECT CID FROM COURSES WHERE dept = "עיצוב")
```

קינון – דוגמא סטודנטים

Student (SID, Name, dept, year)

Courses (CID, Name, Credit, dept)

Studies (SID, CID)

שאלה: מיהם הסטודנטים שלומדים במדעי המחשב וגם לומדים קורס כלשהוא של עיצוב

SELECT SID FROM Student JOIN Studies Using (SID)
WHERE dept = "מדעי המחשב" AND
קורס שלומד ההסטודנט שבודקים IN (קורסים של עיצוב)

קינון

Area (a_name, a_location, a_type, a_size)

Plant (p_name, p_type, p_diameter, p_maxheight)

Planting (p_name, a_name, date, amount)

דוגמא: מצא את הצמח בעל רדיוס השתילה המקסימלי

```
SELECT p_name
FROM plant
WHERE p_diameter >= ALL (SELECT  p_diameter FROM plant)
```

תשובות עם שגיאה

```
SELECT p_name, max(p_diameter) FROM plant

SELECT p_name, FROM plant WHERE p_diameter = max(p_diameter)
```

קינון

Area (a_name, a_location, a_type, a_size)

Plant (p_name, p_type, p_diameter, p_maxheight)

Planting (p_name, a_name, date, amount)

דוגמא: מצא את הצמח בעל רדיוס השתילה המקסימלי

ובדרך נוספת:

```
SELECT p_name
FROM plant
WHERE p_diameter IN (SELECT  max(p_diameter) FROM plant)
```

```
SELECT p_name
FROM plant
WHERE p_diameter = ANY(SELECT  max(p_diameter) FROM plant)
```

EXISTS – לפעמים חייבים

Area (a_name, a_location, a_type, a_size)

Plant (p_name, p_type, p_diameter, p_maxheight)

Planting (p_name, a_name, date, amount)

האם יחס שאילתה פנימית קיים או שהוא יחס ריק EXISTS

דוגמא: מצא צמח השתול רק בחלקה אחת.

```
SELECT p1.p_name FROM planting AS p1
WHERE NOT EXISTS (SELECT * FROM planting AS p2
        WHERE p1.p_name=p2.p_name
        and p1.a_name != p2.a_name)
```

שמות צמחים שלא קיימות שורות עם שם צמח זהה ושם חלקה שונה. כלומר הצמח לא מופיע ביותר מחלקה אחת.

EXISTS – לפעמים חייבים

Area (a_name, a_location, a_type, a_size)

Plant (p_name, p_type, p_diameter, p_maxheight)

Planting (p_name, a_name, date, amount)

לפעמים לא חייבים – אפשר לעשות גם בספירה

דוגמא: מצא צמח השתול רק בחלקה אחת.

```
SELECT p_name
FROM Planting
GROUP BY p_name
HAVING count(a_name)=1
```

השוואה בין איבר לקבוצת איברים

Area (a_name, a_location, a_type, a_size)

Plant (p_name, p_type, p_diameter, p_maxheight)

Planting (p_name, a_name, date, amount)

=, !=, <, <=, >, >= ואחריהם **ALL, SOME, ANY (SOME ו ANY זהים)**

דוגמא: מצא את רשימת הצמחים שמקסימום הגובה שלהם גדול ממקסימום הגובה של שיח כלשהו

```
SELECT p_name
FROM plant
WHERE p_maxheight > SOME (SELECT  p_maxheight FROM plant
          WHERE p_type LIKE "%שיח")
```

השוואה בין קבוצות

Area (a_name, a_location, a_type, a_size)

Plant (p_name, p_type, p_diameter, p_maxheight)

Planting (p_name, a_name, date, amount)

CONTAINS ו **NOT CONTAINS** בודקים האם קבוצה של ערכים מכילה (או לא) קבוצה אחרת של ערכים

דוגמא: מצאו שמות הצמחים ששתולים בכל האיזורים שמיקומם בחולון

```
SELECT DISTINCT P.p_name FROM planting As P
WHERE (SELECT A.a_name FROM planting As A
        WHERE P.p_name=A.p_name )
        CONTAINS
        (SELECT a_name FROM area
        WHERE a_location = "Holon")
```

השוואה בין קבוצות

דוגמא: מצאו שמות הצמחים ששתולים בכל איזורים שמיקומם בחולון

p_name	a_name
p1	a1
p2	a1
p2	a4
p1	a2

a_name	a_location
a1	חולון
a2	חולון
a3	רחוב הרצל
a4	רחוב הרצל
a5	גולומב

```
SELECT DISTINCT p_name FROM planting P
WHERE (SELECT A.a_name FROM planting A
        WHERE P.p_name=A.p_name )
        CONTAINS
        (SELECT a_name FROM area
        WHERE a_location = "Holon")
```

השוואה בין קבוצות

דוגמא: מצאו שמות הצמחים ששתולים בכל האיזורים שמיקומם בחולון

p_name	a_name
p1	a1
p2	a1
p2	a4
p1	a2

÷

a_name
a1
a2

=

P_name
p1

```
SELECT DISTINCT p_name FROM planting P
WHERE (SELECT A.a_name FROM planting A
        WHERE P.p_name=A.p_name )
        CONTAINS
        (SELECT a_name FROM area
        WHERE a_location = "Holon")
```

חילוק עם CONTAINS

תוצאת חילוק = מחלק ÷ מחולק

במחולק יש קשר בין התוצאה לבין המחולק. למשל "שתול ב", "לומד את" וכדומה

המחלק הוא קבוצה של ערכים

תוצאת החילוק היא נתון שמופיע במחולק עם **כל אחד ואחד** מהערכים במחלק.

```
SELECT DISTINCT p_name FROM planting P
WHERE (SELECT A.a_name FROM planting A
        WHERE P.p_name=A.p_name )
        CONTAINS
        (SELECT a_name FROM area
        WHERE a_location = "Holon")
```

חילוק עם CONTAINS

תוצאת חילוק = **מחלק** ÷ **מחולק**

במחולק יש קשר בין התוצאה לבין המחולק. למשל "שתול ב", "לומד את" וכדומה

המחלק הוא קבוצה של ערכים

תוצאת החילוק היא נתון שמופיע במחולק עם **כל אחד ואחד** מהערכים במחלק.

ב SQL – שורה זו היא מעין מצביעה לבדיקת כל אחד מהערכים מול קבוצת הערכים במחלק.

תוצאת חילוק

```
SELECT DISTINCT P.p_name FROM planting P
WHERE (SELECT A.a_name FROM planting A
        WHERE P.p_name=A.p_name )
        CONTAINS
        (SELECT a_name FROM area
        WHERE a_location = "Holon")
```

חילוק עם CONTAINS

תוצאת חילוק = **מחלק** ÷ **מחולק**

במחולק יש קשר בין התוצאה לבין המחולק. למשל "שתול ב", "לומד את" וכדומה

המחלק הוא קבוצה של ערכים

תוצאת החילוק היא נתון שמופיע במחולק עם **כל אחד ואחד** מהערכים במחלק.

```
SELECT DISTINCT P.p_name FROM planting P
WHERE (SELECT A.a_name FROM planting A
       WHERE P.p_name=A.p_name )
       CONTAINS
       (SELECT a_name FROM area
       WHERE a_location = "Holon")
```

מחולק

חילוק עם CONTAINS

תוצאת חילוק = **מחלק** ÷ **מחולק**

במחולק יש קשר בין התוצאה לבין המחולק. למשל "שתול ב", "לומד את" וכדומה

המחלק הוא קבוצה של ערכים

תוצאת החילוק היא נתון שמופיע במחולק עם **כל אחד ואחד** מהערכים במחלק.

ב SQL – שורה זו היא מעין מצביעה לבדיקת כל אחד מהערכים מול קבוצת הערכים במחלק.

```
SELECT DISTINCT P.p_name FROM planting P
WHERE (SELECT A.a_name FROM planting A
        WHERE P.p_name=A.p_name )
        CONTAINS
        (SELECT a_name FROM area
        WHERE a_location = "Holon")
```

מחלק

– EXISTS

Area (a_name, a_location, a_type, a_size)

Plant (p_name, p_type, p_diameter, p_maxheight)

Planting (p_name, a_name, date, amount)

דוגמא: מצא איזורים ששתולים בהם כל הצמחים השתולים במיקום רחוב הרצל. שימוש כפול בשלילה

p_name	a_name
p1	a1
p2	a1
p2	a4
p1	a2

P_name	a_name	a_location
p1	a1	רחוב הרצל
p2	a1	רחוב הרצל
p2	a4	חולון
p4	a1	רחוב הרצל
p5	a5	גולומב

– EXISTS

Area (a_name, a_location, a_type, a_size)

Plant (p_name, p_type, p_diameter, p_maxheight)

Planting (p_name, a_name, date, amount)

דוגמא: מצא איזורים ששתולים בהם כל הצמחים השתולים במיקום רחוב הרצל. שימוש כפול בשלילה

p_name	a_name
p1	a1
p2	a1
p2	a4
p1	a2

÷

P_name	a_name	a_location
p1	a1	רחוב הרצל
p2	a1	רחוב הרצל
p4	a1	רחוב הרצל

– EXISTS

Area (a_name, a_location, a_type, a_size)

Plant (p_name, p_type, p_diameter, p_maxheight)

Planting (p_name, a_name, date, amount)

דוגמא: מצא איזורים ששתולים בהם כל הצמחים השתולים במיקום רחוב הרצל. שימוש כפול בשלילה

p_name	a_name
p1	a1
p2	a1
p2	a4
p1	a2

÷

P_name
p1
p2
p4

EXISTS – לפעמים חייבים

Area (a_name, a_location, a_type, a_size)

Plant (p_name, p_type, p_diameter, p_maxheight)

Planting (p_name, a_name, date, amount)

דוגמא: מצא איזורים ששתולים בהם כל הצמחים השתולים במיקום רחוב הרצל. שימוש כפול בשלילה

p_name	a_name
p1	a1
p2	a1
p2	a4
p1	a2

÷

P_name
p1
p2
p4

=

a_name

EXISTS – לפעמים חייבים

Area (a_name, a_location, a_type, a_size)

Plant (p_name, p_type, p_diameter, p_maxheight)

Planting (p_name, a_name, date, amount)

דוגמא: מצא איזורים ששתולים בהם כל הצמחים השתולים במיקום רחוב הרצל. שימוש כפול בשלילה

p_name	a_name
p1	a1
p2	a1
p2	a4
p1	a2
p4	a1

÷

P_name
p1
p2
p4

=

a_name
a1

EXISTS – לפעמים חייבים(אין CONTAINS ב MYSQL)

Area (a_name, a_location, a_type, a_size)

Plant (p_name, p_type, p_diameter, p_maxheight)

Planting (p_name, a_name, date, amount)

האם יחס שאילתה פנימית קיים או שהוא יחס ריק EXISTS

דוגמא: מצא איזורים ששתולים בהם כל הצמחים השתולים במיקום רחוב הרצל. שימוש כפול בשלילה

```
SELECT DISTINCT a_name FROM planting AS P
WHERE NOT EXISTS (SELECT p_name FROM planting JOIN Area
USING(a_name)   WHERE a_location= "רחוב הרצל" AND p_name NOT IN
            (SELECT p_name FROM planting AS P1
            WHERE P.a_name = P1.a_name))
```

*** NOT EXISTS מאפשר שאילתות שבאלגברה כוללות פעולת חילוק.**

EXISTS – לפעמים חייבים

הנוסחא אומרת – **לא** קיים צמח בשדרות ברחוב הרצל, כך שהצמח **לא** נמצא בחלקה המופיעה בתוצאה. שזה $\neg\exists X(\neg\vartheta(X))$

זה שקול לוגית ל $\forall X(\vartheta(X))$ - כל צמח ברחוב הרצל נמצא בחלקה המופיעה בתוצאה.

לבטא "כל" משתמשים לעיתים קרובות בשלילה כפולה.

תוצאת חילוק = **מחלק** ÷ **מחולק**

```
SELECT DISTINCT P.a_name FROM planting AS P
WHERE NOT EXISTS (SELECT p_name FROM planting JOIN area
USING (a_name) WHERE a_location = "רחוב הרצל" AND p_name
NOT IN (SELECT p_name FROM planting AS P1
        WHERE P.a_name = P1.a_name))
```

*** NOT EXISTS מאפשר שאילתות שבאלגברה כוללות פעולת חילוק.**

פתרון המשתמש ב Contains

מצא איזורים ששתולים בהם **כל** הצמחים השתולים במיקום רחוב הרצל.

להשוואה

```
SELECT DISTINCT P.p_name FROM planting P //  שורה תורנית המתייחסת לחלקה הנבחנת
WHERE (SELECT A.a_name FROM planting A
        WHERE P.p_name=A.p_name ) //כל הצמחים בחלקה הנבחרת
        CONTAINS
        (SELECT a_name FROM area
        WHERE a_location = "Holon")
```

חילוק דוגמא נוספת

Student (SID, Name, dept, year)

Courses (CID, Name, Credit, dept)

Studies (SID, CID)

שאלה: מיהם כל הסטודנטים שלומדים במדעי המחשב וגם לומדים קורס בעיצוב?

האם השאלה היא חילוק?

לא

חילוק דוגמא נוספת

Student (SID, Name, dept, year)

Courses (CID, Name, Credit, dept)

Studies (SID, CID)

שאלה: מיהם כל הסטודנטים שלומדים במדעי המחשב ולומדים את כל הקורסים בעיצוב?

האם השאלה היא חילוק?

כן

חילוק דוגמא נוספת

Student (SID, Name, dept, year)

Courses (CID, Name, Credit, dept)

Studies (SID, CID)

שאלה: מיהם כל הסטודנטים שלומדים במדעי המחשב ולומדים את כל הקורסים בעיצוב?

אם כן – אז מה המחלק במילים?

כל הקורסים בעיצוב

חילוק דוגמא נוספת

Student (SID, Name, dept, year)

Courses (CID, Name, Credit, dept)

Studies (SID, CID)

שאלה: מיהם כל הסטודנטים שלומדים במדעי המחשב ולומדים את כל הקורסים בעיצוב?

אם כן – אז מה המחלק במילים? נכתוב שאילתה:

SELECT CID FROM Course WHERE dept = "עיצוב"

חילוק דוגמא נוספת

Student (SID, Name, dept, year)

Courses (CID, Name, Credit, dept)

Studies (SID, CID)

שאלה: מיהם כל הסטודנטים שלומדים במדעי המחשב ולומדים את כל הקורסים בעיצוב?

אם כן – ננסח את השאלה בשלילה כפולה:

מיהם כל הסטודנטים ממדמ"ח שלא לומדים קורס שלא בעיצוב – לא נכון

יש שלילה כפולה, אבל לא שולל את המשפט.

חילוק דוגמא נוספת

Student (SID, Name, dept, year)

Courses (CID, Name, Credit, dept)

Studies (SID, CID)

שאלה: מיהם כל הסטודנטים שלומדים במדעי המחשב ולומדים את כל הקורסים בעיצוב?

אם כן – ננסח את השאלה בשלילה כפולה:

מיהם כל הסטודנטים ממדמ"ח שלא לומדים קורס בעיצוב – לא נכון

חילוק דוגמא נוספת

Student (SID, Name, dept, year)

Courses (CID, Name, Credit, dept)

Studies (SID, CID)

שאלה: מיהם כל הסטודנטים שלומדים במדעי המחשב ולומדים את כל הקורסים בעיצוב?

אם כן – ננסח את השאלה בשלילה כפולה:

מיהם כל הסטודנטים ממדמ"ח **כך שלא קיים**

נוסיף "כך שלא קיים" כדי לשלול נכון ונוסיף גם "שלא" במשפט.

חילוק דוגמא נוספת

Student (SID, Name, dept, year)

Courses (CID, Name, Credit, dept)

Studies (SID, CID)

שאלה: מיהם כל הסטודנטים שלומדים במדעי המחשב ולומדים את כל הקורסים בעיצוב?

אם כן – ננסח את השאלה בשלילה כפולה:

נוסיף "כך שלא קיים" כדי לשלול נכון ונוסיף גם "שלא" במשפט.

מיהם כל הסטודנטים ממדמ"ח **כך שלא קיים** קורס מעיצוב שהם **לא** לומדים אותו.

חילוק דוגמא נוספת

Student (SID, Name, dept, year)

Courses (CID, Name, Credit, dept)

Studies (SID, CID)

שאלה: מיהם כל הסטודנטים שלומדים במדעי המחשב ולומדים את כל הקורסים בעיצוב?

SELECT SID FROM Student WHERE student.dept = "מדמ"ח" AND
NOT EXISTS ((המחלק) קורס מעיצוב AND
CID NOT IN ((המחולק) קורסים שלומד הסטודנט מההתחלה))

חילוק דוגמא נוספת

Student (SID, Name, dept, year)

Courses (CID, Name, Credit, dept)

Studies (SID, CID)

שאלה: מיהם כל הסטודנטים שלומדים במדעי המחשב ולומדים את כל הקורסים בעיצוב?

```
SELECT SID FROM Student WHERE student.dept = "מדמ"ח" AND
NOT EXISTS (SELECT CID FROM COURSE WHERE Dept = "עיצוב" AND
CID NOT IN ( SELECT CID FROM Studies WHERE Student.SID = Studies.SID))
```

בדיקת כפילויות בשאילתה

Area (a_name, a_location, a_type, a_size)

Plant (p_name, p_type, p_diameter, p_maxheight)

Planting (p_name, a_name, date, amount)

WHERE UNIQUE (sub-query)

דוגמא: מצא צמח השתול רק בחלקה אחת

```
SELECT P1.p_name FROM planting AS P1
        WHERE  UNIQUE (SELECT a_name FROM planting As P2
        WHERE P1.p_name = P2.p_name)
```

תת שאילתות בפסוק FROM

Area (a_name, a_location, a_type, a_size)

Plant (p_name, p_type, p_diameter, p_maxheight)

Planting (p_name, a_name, date, amount)

דוגמא: מצא שמות איזורים שיש בהם לפחות 100 שתילים

```
SELECT a_name
FROM (SELECT a_name, sum(amount) AS num_plants
        FROM planting GROUP BY a_name)
        AS plants_per_area(a_name, num_plants)
WHERE num_plants >= 100
```

שרשור עמודות

Area (a_name, a_location, a_type, a_size)

Plant (p_name, p_type, p_diameter, p_maxheight)

Planting (p_name, a_name, date, amount)

SELECT **CONCAT(p_name, ` `, a_name)** AS p_area…

דוגמא: צור טבלה בה שם הצמח וסוגו הם חלק מאותה עמודה

SELECT CONCAT(p_name, ` `, p_type) AS FullName FROM Plant

דרכים לממש צירוף טבעי

$$\Pi_{\text{customer.customer-name, customer-city}} (\text{borrow} \bowtie \text{customer})$$

SELECT DISTINCT customer.customer-name, customer-city

FROM borrow, customer

WHERE borrow.customer-name = customer.customer-name

SELECT DISTINCT customer.customer-name, customer-city

FROM borrow **JOIN** customer

ON (borrow.customer-name = customer.customer-name)

SELECT DISTINCT customer.customer-name, customer-city

FROM borrow **JOIN** customer

USING (customer-name)

הגדרת משתנים SCOPE

אם משתנה - חיה מוגדר גם לוקלית בתת-שאילתה וגם גלובלית, תופסת ההגדרה הלוקלית.

שתי הדוגמאות יעבדו באופן זהה.

```
SELECT R.name, ….
FROM r1 R
WHERE R.id =
(SELECT P.id
FROM r2 P
WHERE….. )
```

```
SELECT R.name, ….
FROM r1 R
WHERE R.id =
(SELECT R.id
FROM r2 R
WHERE….. )
```

אילוצים ב SQL

SQL כוללת גם שפת הגדרת הנתונים DDL=Data Definition Language

שפה זו מאפשרת להגדיר:

תבניות יחסים

אילוצי תחום – תחום ערכים לכל תכונה, אוסף אינדקסים ליחס,

הרשאות ליחס, אילוצי שלמות ועוד

הוספת רשומות

הוספת רשומה/רשומות לטבלה בודדת בכל פעם.

הנתון המוכנס חייב להתאים לעמודה ולתנאים שלה (למשל לא ניתן לעדכן מחרוזת בעמודה של מספרים).

```
INSERT INTO table_name
(colum1, column2, …) VALUES (value1, value2, …)
```

עדכון ערכים ברשומות

עדכון רשומה/רשומות בטבלה בודדת בכל פעם.

הנתון המוכנס חייב להתאים לעמודה ולתנאים שלה (למשל לא ניתן לעדכן מחרוזת בעמודה של מספרים).

אפשר לשלב משפטי התניה.

```
UPDATE table_name
SET column1=value1, column2=value2,…
WHERE column_name = some_value
```

מחיקה

פעולת מחיקה מתבצעת על רשומה שלמה, או על אוסף של רשומות (שורות).

לפני מחיקות – מומלץ לגבות את בסיס הנתונים.

```
DELETE FROM table_name
WHERE column_name = some_value
```

מחיקת יחס /טבלה שלמה ע"י

```
DROP TABLE table_name
```

אילוצי תחום

ההנחה היא שכל תחום כולל גם ערך ריק NULL, אך ניתן להגביל זאת.

למשל לא נרצה לאפשר ערך ריק בתכונה המשמשת כמפתח.

מבנה כללי להגדרת תבנית:

CREATE TABLE table_name

(A1 D1, A2 D2, …, An, Dn

Constraint1, constraint 2, …constraintK)

A – תכונות / עמודות

D – טווח ערכים

Constraints – אילוצים על התכונות

אילוצי תחום

דוגמא:

```
CREATE TABLE course
(course_no char(5) not null,
course_name char(40),
points integer,
primary key (course_no),
unique key(course_name),
check (points >=3 and points <=6))
```

אילוצי תחום

דוגמא:

```
CREATE TABLE course
(course_no CHAR(5) NOT NULL // מגדיר שאסור ערך ריק
AUTO_INCREMENT, // מגדיר שהערכים ינתנו אוטומטית בסדר עולה
course_name CHAR(40),
points INTEGER DEFAULT 3, // מגדיר ערך ברירת מחדל
PRIMARY KEY (course_no),  // מגדיר מפתח ראשי
UNIQUE KEY(course_name),   //מגדיר תכונה/תכונות כמפתח קביל אחר
check (points >=3 and points <=6))  // אילוץ המגדיר תנאי לוגי נוסף
```

שינוי סכמה בטבלה קיימת

הוספת עמודה, או מחיקה, מטבלה קיימת.

```
ALTER TABLE table_name
ADD  column_name datatype
```

Or

```
ALTER TABLE table_name
DROP COLUMN column_name
```

שינוי סכמה בטבלה קיימת

שינוי סוג הנתונים

ALTER TABLE table_name

ALTER COLUMN column_name datatype

אילוצי זיקה Referencial Integrity Constraint

אילוץ בין שני יחסים.

קובע כי ערך בתכונה מסויימת (או תכונות) ביחס אחד, חייב להופיע בתכונה מסויימת ביחס אחר.

למשל בקשר בין ישויות, או בישות חלשה התלויה במפתח של ישות חזקה.

מפתח זר Foreign Key – תכונה (או קבוצת תכונות) שהן מפתח בתבנית יחסים אחרת.

אילוצי זיקה Referencial Integrity Constraint

לדוגמא,

```
CREATE TABLE course_in_semester
(course_no CHAR(5) REFERENCES course,
semester CHAR(5),
teacher CHAR(40),
PRIMARY KEY (course_no, semester))
```

אילוצי זיקה Referencial Integrity Constraint

דרך נוספות, שהכרחית כשבמפתח הזר יש יותר מתכונה אחת:

```
CREATE TABLE study
(student_id CHAR(10),
course_no CHAR(5),
semester CHAR(5),
grade INTEGER,
PRIMARY KEY (student_id, course_no, semester),
FOREIGN KEY (student_id) REFERENCES student,
FOREIGN KEY (course_no, semester) REFERENCES course_in_semester)
```

אילוצי זיקה Referencial Integrity Constraint

אפשר גם להוסיף את שם היחס:

```
CREATE TABLE study
(student_id CHAR(10),
course_no CHAR(5),
semester CHAR(5),
grade INTEGER,
PRIMARY KEY (student_id, course_no, semester),
FOREIGN KEY (student_id) REFERENCES student(student_id),
FOREIGN KEY (course_no, semester) REFERENCES course_in_semester)
```

טיפול בהפרת אילוצי זיקה

ההנחה היא שהמערכת מונעת הפרת אילוצים, כאשר ניתנת פקודה למחיקה, הוספה, עדכון, המערכת יכולה להתריע ולפסול את הפעולה. באילוצי זיקה יש כמה אפשרויות שיתאימו לזיקה מסויימת.

פסילת הפעולה המבוקשת – זו המדיניות הרגילה.

ביצוע פעולה בשרשרת על נתונים קשורים - עדכון או ביטול תכונות שהן מפתח זר ביחס שבו הן המפתח. לדוגמא – שינוי מספר קורס ביחס קורס, זאת הפרה של תלות הזיקה ביחס אחר. נהיה מוכנים לעדכון אם הוא יעודכן בכל המופעים ביחסים האחרים. מתאים למשל בתלות קיום כשמוחקים course.

ניתן להגדיר מה לעשות בהמשך להגדרת הזיקה:

מפל של פעולות עדכון או מחיקה // ON UPDATE CASCADE

ON DELETE CASCADE / NO ACTION (RESTRICT)/ SET NULL

טיפול בהפרת אילוצי זיקה

לדוגמא:

השמת ערך ריק מתאימה בטיפוס קשרים רבים-לאחד שמיוצג כמו טיפוס הישויות שהוא ה"רבים".

למשל תבנית של מרצה והתואר שלו, שהמרצה נוסף לקורס בסמסטר. ניתן ליצג את המרצה כתכונה בקורס בהינתן שלכל קורס יש מרצה אחד. Teacher היא מפתח זר ב course_in_semster, אם מוחקים מרצה, מתאים להפוך את ערכו ל NULL, ולא למחוק את הקורסים שלימד או לא לאפשר מחיקה.

ON DELETE SET NULL

ASSERTIONS

למשל: כל סטודנט ילמד בקורס אחד לפחות.

לא קיים ערך של student_id ב student שאינו מופיע ב student_id ב study.

```
CREATE ASSERTION  participation_constraint
        CHECK (NOT EXIST (SELECT * FROM student
        WHERE student_id NOT IN
                (SELECT student_id FROM study)))
```

אילוצי זיקה Referencial Integrity Constraint

אילוץ בין שני יחסים.

תלות הכלה מאפשרת לבטא אילוצי השתתפות.

הכלה בין העמודה A ביחס r לעמודה A ביחס s, באלגברת היחסים:

$$\Pi_A(r) \subseteq \Pi_A(s)$$

למשל: כל סטודנט ילמד בקורס אחד לפחות.

כלומר כל ערך בתכונת המפתח של היחס student חייב להופיע ביחס study.

זוהי תלות הכלה בין בין student_id ב student לבין student_id ב study.

אפשר לממש ע"י ASSERTION – אילוצים כללים שבסיס הנתונים צריך לקיים. יש ל assertion שם ותנאי לוגי בפסוק CHECK שמבוטא כמו בשאילתות.

TRIGGERS

ראינו תגובות אפשריות לאירועי הפרת אילוץ מרשימה נתונה מראש.

ניתן גם להגדיר אירוע ותגובה משלנו.

אירוע – עדכון בתכונה, מחיקת שורה, וכו'נ

תגובה – הדפסת הודעה, מחיקת שורה, עדכון ערך אחר, וכו'

התחביר לא אחיד ולא נרחיב בין מערכות שונות, יש לחפש לכל שפה.

TRIGGERS

דוגמא:

הגדר trigger המונע הוספת סטודנט לקורס אלגברה בסמסטר א 2016

אם מספר הסטודנטים הרשומים הגיע למעל 70, ומחזיר הודעה מתאימה.

```
CREATE TRIGGER maxStudentTrigger
        instead of INSERT ON study
                when(70>= (SELECT count(student_id)
                FROM study
                WHERE course_name="algebra" AND
```

תצפית View

תצפית היא טבלה וירטואלית, שלא קיימת בבסיס הנתונים פיסית, אבל ניגשים אליה לעיתים תכופות.

למשל – תצפית שהיא צירוף שתי טבלאות. או טבלה עם עמודות מוסתרות. ניתן לבצע שאילתות על תצפית, אך לא לעדכן ערכים.

CREATE VIEW view_name AS (query)

Query היא שאילתה חוקית.

DROP VIEW view_name מבטלת את התצפית.

אפשר לעדכן תצפית ע"י insert into אך תהיה חריגה בעדכון, כי הטבלה המקורית לא תתעדכן.

ה View נוצר מחדש בכל פעם שניגשים אליו עם שאילתה.

INDEXES

אינדקס משמש לגישה מהירה ושאילתות מהירות על שדות מסויימים.

```
CREATE INDEX index_name
ON table_name (column1, column2, …)
```

UNIQUE אם רוצים שלא יתאפשרו אינדקסים זהים.

```
CREATE UNIQUE INDEX index_name
ON table_name (column1, column2, …)
```

שימו לב הסינטקס שונה בין DB.

עוד תרגילים

נתונות תבניות היחסים הבאות:

S = (D, E, F) R = (B, C, D) T = (B, C, D, E, F)

עבור היחסים r(R) t(T) s(S):

<u>סעיף א:</u> נתונה שאילתה ב- SQL :

```
select E
from s, r
where  r.D=s.D  and  s.D not in (select D from t)
```

(1) תאר מילולית (נוסח קצר ככל האפשר) את המשמעות של השאילתה.

<u>סעיף ב:</u> נסח שאילתה ב- SQL לקבלת כל זוגות ערכי התכונות C,B ביחס r(R) אשר מופיעים יחד עם כל אחד מערכי התכונה D המופיעים ביחס s(S) .

<u>סעיף ג:</u> נסח שאילתה ב- SQL למציאת הערך המינימלי של התכונה F ביחס s(S) .

עוד תרגילים

נתונות תבניות היחסים הבאות:

S = (D, E, F) R = (B, C, D) T = (B, C, D, E, F)

עבור היחסים r(R) t(T) s(S):

<u>סעיף א:</u> נתונה שאילתה ב- SQL :

```
select E
from s, r
where  r.D=s.D  and  s.D not in (select D from t)
```

(1) תאר מילולית (נוסח קצר ככל האפשר) את המשמעות של השאילתה.

בחר את כל ה E שמופיעים עם D שקיים ב R, אבל ה D לא קיים ב T.

עוד תרגילים

סעיף ב:

```
SELECT C,B
FROM s,r
WHERE r.D in (SELECT D FROM S)
```

זה לא פתרון, כי לא עונה על **כל אחד מערכי D**

עוד תרגילים

סעיף ב:

```
SELECT C,B
FROM r As r1
WHERE (SELECT r2.D FROM r As r2 WHERE r1.C=r2.C AND r1.B=r2.B)
CONTAINS (SELECT D FROM s)
```

עוד תרגילים

סעיף ב:

```
SELECT C,B
FROM r As r1
WHERE NOT EXIST (SELECT D FROM s
WHERE D NOT IN (SELECT D FROM r as r2 WHERE r1.C=r2.C AND
r1.B=r2.B))
```