מסדי נתונים – 2 SQL - 8 תרגול SQL - 8 - 9 פונקציות הקבצה

עמיעד רוזנברג

Windows error messages:

- Press any key to continue or any other key to exit.
- Press any key to... no, no, no, no, NOT THAT KEY!
- Keyboard not found. Press any key to continue.

פונקציות הקבצה

- פונקציות הקבצה הינן פונקציות המקבלות קבוצת ערכים כקלט ומחזירות ערך יחיד.
 - .AVG : מציאת ערך ממוצע
 - MIN : מציאת ערך מינימלי
 - MAX : מציאת ערך מקסימלי
 - SUM : מציאת סכום ערכים
 - מציאת מספר האיברים בקבוצה: COUNT

פונקציות הקבצה

- תמיד נפעיל פעולה מסויימת על עמודה מסויימת.
 - : דוגמא

: Perryridge הצג את היתרה הממוצעת בסניף

SELECT AVG (balance)
 FROM account
 WHERE branch-name='Perryridge';

פונקציות הקבצה

: הערות

- הפונקציות NAX ,MIN, יכולות לעבוד גם על מחרוזות. במקרה כזה – התוצאה תחושב עייפ סדר לקסיקוגרפי.
 - הפונקציות SUM, אל מחרוזות יחזירו o.
 - : COUNT •
- . (עם חזרות) a אופר את מספר הערכים השונים מNULL סופר את מספר הערכים השונים סופר את מספר הערכים השונים מ
 - NULL סופר את מספר הערכים השונים, שאינם COUNT(DISTINCT a) בעמודה a.
 - COUNT(*) סופר את מספר השורות בטבלה (כולל שורות שמכילות NULL).
 - אינו חוקי! COUNT(DISTINCT *) אינו חוקי
 - . אינו חוקי. SUM(AVG(balance)) אינו חוקי. •

- לעתים נרצה להפעיל את פונקציות ההקבצה לא על קבוצה אחת
 של ערכים, אלא על אוסף קבוצות של ערכים, במקרה כזה נשתמש
 ב GROUP BY.
 - GROUP BY מחלק את הטבלה לתתי-קבוצות לפי התכונות שמצוינות בפסוקית זו. שורות עם אותו ערך בעמודות שמופיעות בפסוק GROUP BY מקובצות לכדי תת-קבוצה אחת.
 בתוצאה שנקבל לכל תת-קבוצה תופיע שורת סיכום אחת.
 - במילים אחרות...
 אמרנו שפונקציית הקבצה מקבלת קבוצת ערכים ומחזירה ערך
 יחיד. אבל אם נרצה לקבל ערך עבור תתי-קבוצות של הקבוצה
 הגדולה נשתמש ב-GROUP BY.

- ניזכר בדוגמא שראינו : הצג את היתרה הממוצעת בסניף Perryridge :
- SELECT AVG (balance)
 FROM account
 WHERE branch-name='Perryridge';
 - : נניח שנרצה להציג את היתרה הממוצעת הכללית
- SELECT AVG (balance)
 FROM account

- נרצה להציג את היתרה הממוצעת בכל סניף...
- יש לנו יכולת לחשב את המידע עבור הקבוצה הגדולה הכוללת את כל החשבונות. אבל אנחנו רוצים להפריד את הקבוצה הזו לתתי-קבוצות ע"פ שם הסניף.

כאמור, את ההפרדה לתתי קבוצות ניתן לעשות בעזרת פסוקית .GROUP BY

SELECT branch-name, AVG (balance)
 FROM account
 GROUP BY branch-name;

דוגמא נוספת:הצג את מספר המפקידים בכל אחד מהסניפים השונים:

SELECT branch-name, COUNT(DISTINCT customer-name)
 FROM depositor, account
 WHERE depositor.account-number=account.account-number
 GROUP BY branch-name;

הערות - GROUP BY

- לפי הסטנדרט של SQL, במשפט SQL לפי הסטנדרט של SQL, במשפט GROUP BY שהופיעו בפסוק שהופיעו בפסוק שהום שהם יכולים להיות בעלי ערכים שדות נוספים להצגה (משום שהם יכולים להיות בעלי ערכים שונים עבור רשומות שונות באותה תת-קבוצה).
 - התשובה המוחזרת ממויינת כאילו משפט GROUP BY היה ORDER BY. ברירת המחדל היא מיון בסדר עולה. כדי למיין בסדר יורד ניתן להוסיף DESC בסוף המשפט.

הערות - GROUP BY

- ניתן לציין מספר עמודות בפסוק GROUP BY (מופרדות ע"י פסיק). עבור כל צירוף של ערכי העמודות תיוצר תת-קבוצה.
 - לדוגמה: הצג את מספר הלקוחות שגרים בכל רחוב בכל עיר:
- SELECT customer-city, customer-street, COUNT(customer-name)
 FROM customer
 GROUP BY customer-city, customer-street;

HAVING

- .GROUP BY מאפשר לנו ליצור תנאי עבור הקבוצות הנוצרות עייי
 - : לדוגמה

הצג את שמות הסניפים והיתרה המקסימאלית שלהם, עבור סניפים בהם היתרה הממוצעת גבוהה מ – 1200 :

SELECT branch-name, MAX (balance)
 FROM account
 GROUP BY branch-name
 HAVING AVG(balance)>1200;

הערות - HAVING

- אם באותה שאילתא מופיע גם פסוק WHERE אם באותה שאילתא מופיע גם פסוק HAVING, סדר הפעלות הוא:
 - .WHERE − ראשית מתבצע הפרדיקט שמופיע ב
- השורות שמקיימות את התנאי מקובצות לקבוצות לפי העמודות ב – GROUP BY.
 - רק לאחר מכן נבחרות הקבוצות שמקיימות את התנאי שמופיע ב – HAVING.

הערות - HAVING

: לדוגמה

מציאת היתרה הממוצעת לכל לקוח שגר ב Harrison ויש לו לפחות 3 חשבונות:

SELECT depositor.customer-name, AVG(balance)
 FROM depositor, account, customer
 WHERE depositor.account-number=account.account-number
 AND depositor.customer-name=customer.customer-name
 AND customer-city='Harrison'
 GROUP BY depositor.customer-name
 HAVING COUNT(DISTINCT depositor.account-number)>=3;

פעולת AS

- בתירגול הקודם הכרנו את פעולת AS שמאפשרת לקרוא לטבלאות
 בשם שונה.
- ב-SQL, ניתן גם לקרוא לעמודות בשם שונה בעזרת אותה הפעולה בפסוק ה-SELECT.
- SELECT old_name AS new_name

- : לדוגמא
- הנהלת הבנק שוקלת לציפר את לקוחותיה עייי הורדת סכום של 100 ש מכל הלוואה שנלקחה בבנק. נרצה להציג בפני הנהלת הבנק את מספרי ההלוואות וסכום כל הלוואה לאחר הורדת הסכום הנייל:
- SELECT loan-number, amount-100 AS new_amount FROM loan;

פעולת AS

- : הערות
- כאשר משנים שם של עמודה, אפשר להשתמש בשם החדש בפסוקיות GROUP BY ,ORDER BY. אי אפשר להשתמש בכינוי החדש לשם עמודה בפסוק WHERE.
 - את המילה AS. גם כאן, ניתן להשמיט את המילה

INNER JOIN

- : יש לנו 2 דרכים שונות לבצע מכפלה קרטזית SQL בשפת
- רשימת שמות הטבלאת (עם פסיקים ביניהם) בפסוק FROM.
- רשימת שמות הטבלאות כאשר בין הטבלאות נרשום INNER) JOIN).
 - סינון הרשומות לאחר מכן מתבצע בצורות שונות
 - ניתן להוסיף את התנאי בפסוק ה-WHERE.
 - . (Theta Join ניתן להוסיף את התנאי לפעולת הצירוף עצמו (כמו
 - הוספת התנאי מחייבת אותנו להשתמש ב-JOIN.
 - נשתמש באופרטור ON עיימ לקבוע את התנאי.
 נשתמש באופרטור USING עיימ לקבוע את שמות העמודות.

INNER JOIN

: ON אופרטור

- בפסוק ON יופיע תנאי כלשהו (ניתן להשתמש ב-AND). תתבצע מכפלה קרטזית בין הטבלאות, ויוחזרו השורות המקיימות את התנאי.
- באמצעות אופרטור זה ניתן להפריד בין כתיבת התנאים שמציינים כיצד לבצע את הצירוף בין הטבלאות (ON) ובין כתיבת התנאים המציינים את השורות שנרצה בתוצאה המוחזרת (WHERE).
 - דוגמא: הצג את שמות המפקידים והיתרה בחשבונם:
- SELECT customer-name, balance
 FROM depositor JOIN account
 ON depositor.account-number=account.account-number;

INNER JOIN

- : USING אופרטור
- בפסוק זה ניתן לרשום רשימת שמות של עמודות המופיעות בשני היחסים. מתבצעת מכפלה קרטזית ונבחרות השורות בהן יש ערכים זהים בעמודות אלה.
 - דוגמא: הצג את שמות המפקידים והיתרה בחשבונם:
 - SELECT customer-name, balance FROM depositor JOIN account USING (account-number);

- באופן דומה לאלגברת היחסים, צירוף חיצוני מאפשר שמירת מידע. כזכור, ישנם 3 סוגים של צירוף חיצוני (עם שמירת מידע):
- שמירת מידע מהטבלה השמאלית.
 שורות מהטבלה השמאלית שלא מתאימות לאף שורה מהטבלה הימנית
 (לפי התנאי שמצוין ב ON/USING) מתווספות לתוצאה המוחזרת,
 כאשר בעמודות הנוספות יופיע ערך NULL.
 - .RIGHT JOIN שמירת מידע מהטבלה הימנית.
- MySQL שמירת מידע משתי הטבלאות. לא ממומש ב-FULL JOIN •

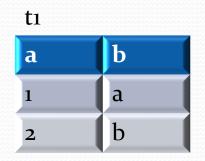
: דוגמא

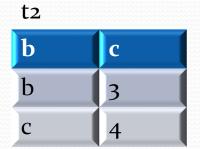
הצג את שמות כל הלקוחות ומספר הלוואה אם לקחו הלווה:

• SELECT customer-name, loan-number FROM customer LEFT JOIN borrower USING (customer-name);

- : הערות
- איבוד מידע יכול להתרחש רק אם יש תנאי על הצירוף, לכן אי אפשר USING להשתמש בצירוף חיצוני מבלי לרשום פסוק
 - שימו לב!

השלמת המידע מתבצעת עבור שורות שיינפלויי בתנאי ON או OV (כלומר התנאי מחושב ושורות שלא עמדו בתנאי מרופדות ב NULL כלומר התנאי מחושב ושורות שלא עמדו בתנאי מרופדות ב WHERE בהתאם לכיוון הצירוף). פסוק WHERE, לעומת זאת, מתבצע אחרי שמירת המידע ולכן אין שמירת מידע על שורות שנפלו בתנאי WHERE.



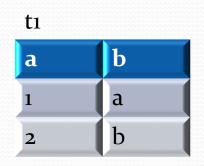


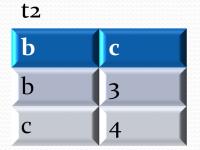
: הערות

: דוגמא

SELECT *
 FROM t1 LEFT JOIN t2
 ON (t1.b=t2.b);

a	tı.b	t2.b	c
1	a	NULL	NULL
2	b	b	3





: הערות

: דוגמא

: לעומת זאת •

SELECT *
 FROM t1 LEFT JOIN t2
 ON (TRUE)
 WHERE (t1.b=t2.b);

לאשורומ∈ן הותנטציעשול שמצוכ שלפסוק ה- לאשורומ∈ן הותנטציעשול של אלי אלי אלי אלי אלי אלי אלי במקרה זה – כל הרשומות עונות על התנאי.

a	tı.b	t2.b	c
1	a	b	3
a	tı.b	t2.b	c
2	b	b	3
2	b	С	4

- : הערות
- בגלל אופטימיזציות של MySQL, עדיף להשתמש בצירוף שמאלי (מאשר בימני).
 - :SQL שיפול בערכי איפול בערכי
 - על מנת לבדוק אם ערך מסוים הוא NULL באופרטור IS NULL.
 - על מנת לבדוק אם ערך מסוים אינו ערך NULL, נשתמש באופרטור IS NOT NULL.

NATURAL JOIN

- כמו באלגברת יחסים, גם ב-SQL פעולה זו מבצעת מכפלה קרטזית בין הטבלאות ובוחרת את הרשומות המכילות ערכים זהים בעמודות המופיעות בשני הטבלאות.
- מקביל לביצוע צירוף פנימי עם USING על רשימת כל העמודות המופיעות בשני הטבלאות.
 - :ניתן לבצע צירוף טבעי עם שמירת מידע ע"יי
 - NATURAL LEFT JOIN •
 - NATURAL RIGHT JOIN •

NATURAL JOIN

- : דוגמאות
- הצג את שמות הלקוחות שלקחו הלוואה, מספר הלוואה ושם עיר
 הלקוח:
 - SELECT customer-name, loan-number, customer-city FROM customer NATURAL JOIN borrower;
 - שמות כל המפקידים שלא לקחו הלוואה:
 - SELECT customer-name FROM depositor NATURAL LEFT JOIN borrower WHERE loan-number IS NULL;

JOIN

- : דוגמא חשובה
- הצגת סניפי הבנק הקיימים, כאשר עבור סניפים בהם קיים חשבון
 עם יתרה גבוהה מ-400 יש להציג את פרטי החשבון
 - SELECT branch.branch-name,account.*
 FROM branch LEFT JOIN account
 ON (branch.branch-name= account.branch-name
 AND balance>400);

אז מה היה לנו?

- ראינו פעולות הפועלות על קבוצת ערכים
 - .AVG •
 - .MIN •
 - .MAX •
 - .SUM •
 - .COUNT •
- ראינו יכולת לבצע את הפעולות על תתי-קבוצות בעזרת הפעולה BY GROUP.
 - .HAVING ראינו יכולת לסנן קבוצות עייי
 - שונות שמאפשרות איחוד מידע ממסי JOIN ראינו פעולות שונות שמאפשרות איחוד מידע ממסי טבלאות.