الاستعلام باستخدام لغة SQL

د. روان قرعوني
محتوى مجاني غير مخصص للبيع التجاري

18/05/2023

RB Informatics;

قواعد معطيات 1

تحدثنا في المحاضرة السابقة عن كيفية إنشاء قاعدة معطيات وعن الاستعلام باستخدام لغة SQL والتي سنكمل الحديث عنها في هذه المحاضرة..♥

1. Select

تحدثنا في المحاضرة السابقة عن select وسنعزز فهمنا لكيفية استعمالها بعدة أمثلة

ليكن لدينا الجدولين الآتيين:

Employee

ID	Name	salary	CID
1	Ahmad	1000	1
2	Khaled	2000	1
3	Hasan	3000	2
4	Hasan	4000	2
...	...	...	...

Company

ID	Name	salary
1	SYR	1000
2	MTN	2000
...	...	...
...	...	...

تذكرة:

- لماذا قمنا بوضع CID؟؟
  - ✓ لأن العلاقة بين الـ tow tables هي علاقة one-to-many لأن الشركة فيها أكثر من موظف والموظف يعمل في شركة واحدة
- ال FK يوضع من جهة الـ many

سنستعرض عدة استعلامات ونقوم بالتعبير عنها بلغة الـ SQL:

- **إحضار كل الموظفين:**

```
Select*
From Employee
```

- **إحضار أسماء الموظفين ورواتبهم:**

```
Select name, salary
From Employee
```

- **إحضار الأسماء غير المكررة:**

نستخدم كلمة distinct

```
Select distinct name
From Employee
```

- **إحضار كل موظف اسمه واسم شركته:**

```
Select Employee.name, Company.name
From Employee, Company
```

تدل على الجداء الديكارتي

ما هو شكل الجدول الجديد؟؟

ID	Name	Salary	CID	ID	Name	Address
1	Ahmad	1000	1	1	SYR	Almazeh
1	Ahmad	1000	1	2	MTN	Aboromane
2	Khaled	2000	1	1	SYR	Almazeh
2	Khaled	2000	1	2	MTN	Aboromane
3	Hasan	3000	2	1	SYR	Almazeh
3	Hasan	3000	2	2	MTN	Aboromane

نتيجة الاستعلام السابق

Name	Name
Ahmad	SYR
Ahmad	MTN
Khaled	SYR
Khaled	MTN
Hasan	SYR
Hasan	MTN

إن تمثيل الاستعلام السابق خاطئ لأنه سيجلب لنا الجدول السابق وهو خاطئ لأننا نريد معرفة أين يعمل كل موظف

وكما نعلم أن العلاقة السابقة هي علاقة one to many حيث الموظف يعمل في شركة واحدة فقط

أما الجدول السابق يدل على أن الموظف الواحد يعمل في أكثر من شركة

ولهذا نحتاج إلى وجود شرط في الاستعلام (تعليمة where)

- **فيصبح الاستعلام كالآتي:**

```
Select Employee.name, Company.name
Form Employee, Company
Where Employee.CID=Company.ID
```

نتيجة الاستعلام تصبح بالشكل

Name	Name
Ahmad	SYR
Khaled	SYR
Hasan	MTN

أو يمكن التعبير عنه كالآتي (alias):

```
Select e.name, c.name
Form Employee [ ] e, company [ ] c
Where e.CID=c.ID
```

OR

```
Select e.name, c.name
Form Employee [as] e, company [as] c
Where e.CID=c.ID
```

2. إعادة التسمية:

تجري إعادة التسمية للعلاقات والواصفات باستخدام فقرة as: old name as new name

ماذا نعني بـ alias؟

إعادة التسمية ويمكن إعادة التسمية إما عن طريق كلمة as أو عن طريق فراغ كما في المثال السابق

قاعدة معطيات حول المصرف

- Branch-schema=(branch-name, branch-city, assets)
- Customer=(customer-name, customer-street, city)
- Loan=(branch-name, loan-number, amount)
- Borrower=(customer-name, loan-number)
- Account=(branch-name, account-number, balance)
- Depositor=(customer-name, account-number)

مثال على إعادة التسمية

لإيجاد أسماء و أرقام قروض جميع الزبائن الذين حصلوا على قرض من فرع perryridge مع التعويض عن اسم العمود loan-number باسم loan-id نكتب:

```
Select distinct customer-name, borrower.Loan-number as loan id
From borrower, loan
Where borrower.loan-number= loan.loan-number and branch-name="perryridge"
```

3. متحولات الحدودية

عبارة as مفيدة في تعريف متحولات من نمط الحدودية وكما في لغة القضايا فإن المتحول الحدودي في SQL يرتبط بعلاقة, لنبين ذلك في المثال التالي:

لإيجاد جميع الزبائن الذين حصلوا على قرض من المصرف (أسماء الزبائن وأرقام قروضهم)

```
Select distinct customer-name, T.loan-number
From borrower as T, loan as S
Where T.loan-number = S.loan-number
```

يُلاحظ في المثال السابق أن المتحول الحدودي T يرمز إلى حدودية لا على التعيين من العلاقة borrower وفي بعض الحالات نحتاج إلى إعطاء تسميات مختلفة لمتحولات حدودية كما يوضح ذلك في المثال التالي:

لنفترض أننا نحتاج إلى معرفة جميع الفروع التي توجد في نفس المدينة الذي يوجد فيها الفرع perryridge:

- لحل الاستعلام السابق نستخدم متحولين الأول S والثاني T يشير كلاهما إلى العلاقة branch-schema ونستخدم التعليمة:

```
Select T.branch-name
From branch T, Branch S
Where S.city=T.city and S.branch
Name=T.branch-name="perryridge";
```

	Branch-name	Branch-city	Assets
$S \longrightarrow$	perryridge		
$T \longrightarrow$	y		

4. ترتيب الحدوديات الناتجة:

يمكن للمستثمر أن يتحكم في ترتيب الحدوديات في العلاقة الناتجة وذلك باستخدام عبارة order by

فمثلا لاستخراج قائمة مرتبة ترتيبا أبجديا بأسماء الزبائن والذين حصلوا على قرض من الفرع "perryridge" نكتب:

```
Select distinct customer-name
From borrower, loan
Where borrower.loan number= loan.loan-number and brunch-name="perryridge"
order by customer-name;
```

- إن القائمة الناتجة ستكون مرتبة ترتيبا ويمكن تحديد طريقة الترتيب تصاعديا *ascending* أو تنازليا descending كما يمكن أن يطلب الترتيب على عدة واصفات

مثال: لنفترض أننا نريد قائمة العروض مرتبة ترتيبا تنازليا حسب مبلغ القرض وفي حال وجود عدة قروض لها نفس المبلغ نقوم بترتيبها تصاعديا حسب رقم القرض.

نكتب:

```
Select *
From loan
Order by amount desc, loan-number asc
```

- إن كلفة إجراء ترتيب على عدد كبير من الحدوديات هي كلفة عالية ولذلك يجري إجراء الترتيب فقط في حال الضرورة.

فكر معنا

ماهي نتيجة الاستعلامين الآتيين:

- select 'x', select 'x' From Employee
- select 'x' From Employee

X
X
X

بحيث يقوم بطباعة المحرف x بعدد مرات الأسطر (يضيف Column عدد أسطره بعدد أسطر ال table وعناصره هي المحرف x) لأننا قمنا بذكر (اسم الجدول From)

- **ماذا لو كانت** Select 'x' ,name From Employee

x	Ahmad
x	Khaled
x	Hasan

أيضا يضيف column جديدة بعدد أسطر الجدول بجانب عمود ال name وعدد عناصره هي عدد أسطر ال table وعناصره هي المحرف x

- Select 'x'

يطبع المحرف x لمرة واحدة وفي خانة واحدة لأنه لم يتم تحديد أي table

هنا تم وضع كل كلمة ضمن ' ' في عمود

بفرض لدينا:

```
Select 'Hello', 'name', 'your salary', 'salary'
From Employee
```

تكون النتيجة إضافة 4 أعمدة لجدول ال Employee عدد عناصرها بعدد أسطر ال table

لكن ماذا لو أردنا ظهور ما سبق كـ مسج ؟؟

- **عندها يترتب علينا إيجاد طريقة لدمج الأعمدة وهنا ظهر لدينا ما يسمى بـ وصل السلاسل** concatenating **باستخدام** (||)

بعد القيام بـ concatenating **تصبح النتيجة كالتالي:**

Hello name yoursalary salary
Hello Ahmad yoursalary 1000

بحيث نعبّر عنها بالاستفسار كالتالي (Query):

(with concatenating)

```
Select 'hello' || 'name' || 'yoursalary' || 'salary'
From Employee
```

نفس الاستعلام السابق ولكن بدلاً من وضع فواصل نضع concatenating

(without concatenation)

```
Select 'hello', 'name', 'yoursalary' , 'salary'
From Employee
```

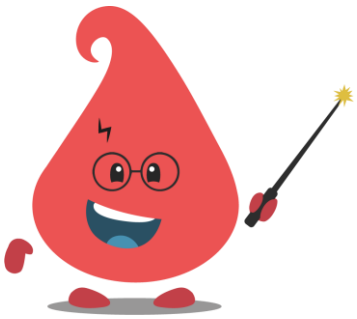
إن الراتب salary **السابق هو راتب السنوي**

ماذا لو أردنا معرفة الراتب الشهري؟؟

عندها نقسّم salary **على عدد الأشهر فيصبح الاستعلام بالشكل التالي:**

```
Select name, Salary/12 as monthly
Salary
From Employee
```

وتظهر نتيجة الاستعلام بالشكل:



name	Monthly salary

5. تعليمة where

تحدثنا في المحاضرة السابقة عنها وسنقوم بمراجعة أهم الأفكار معاً. يمكن وضع أكثر من شرط في where

مثال:

الاستعلام عن أسماء الموظفين مع شركاتهم التي تبلغ قيمة رواتبهم أكثر من 2000 والـ id الخاص بهم هو 2 أو 3 و أرقام شركاتهم بين 1 و 12

```
Select e.name, c.name
From Employee e, company c
Where e.CID= c.ID
And salary >2000
And e.Id in(2,3)
And e.CID between 1 and 12
```

يمكننا أيضا التعبير عن between كالتالي:
Between (1,12)

- داخل الـ where يمكننا القيام بـ (and, or, not, in, between)

6. Like

- **العمليات على سلسلة المحارف**

أكثر العمليات استخداما هي التشابه الجزئي like ونصف هنا هذا التشابه باستخدام حرفين:

- % : للدلالة على أي سلسلة أحرف جزئية
- Underscore: للدلالة على أي حرف

- **أمثلة:**

- إحضار الأسماء التي فيها حرف d: where e.name like '%d%'
- أو التي في نهايتها d: where e.name like '%d'
- التي يسبقها أربع أحرف قبل d: where e.name like '_ _ _ _d'
- لو فرضا كان الاسم فيه ٪ من أصله: where e.name like 'Ahmad / % S' ((نضع قبل % حرف {Esc}))
- للبحث عن الفروع التي تحوي سلسلة الأحرف idge في أي موقع

من أي فرع نستخدم التعليمة: select branch-name

```
From branch
Where branch-name like '% idge% ';
```

وتستخدم '_ _ _' للدلالة على أي سلسلة أحرف مؤلفة من ثلاثة أحرف بالضبط

- ولوضع الحرفين '%' و '_' ضمن السلسلة المراد مقارنتها لا لاستخدام وظيفتها يجب أن يسبقا بحرف ESC كما ذكرنا سابقاً, فنكتب:

Customer-street like 'ab ESC % cd %'

أي البحث عن اسم شارع سكن الزبون والذي يبدأ بسلسلة المحارف 'ab % cd'

7. بعض التوابع

- إحضار الأسماء التي طول اسمها 5 نستخدم التابع length

$$\Rightarrow length(e.name) = 5$$

- تحويل أحرف السلسلة لأحرف كبيرة:

$$\Rightarrow Upper$$

- تحويل أحرف السلسلة لأحرف صغيرة:

$$\Rightarrow Lower$$

8. معالجة القيم غير المعلومة

تسمح لغة SQL باستخدام القيم غير المعلومة null للدلالة على عدم توافر معلومات في واصف.

ملاحظة بديهية:

Is not null عكس is null

مثال: لإيجاد أرقام القروض التي لا نعرف قيمتها نكتب:

```
Select loan-number
From loan
Where amount is null;
```

مثال آخر:

```
Select *
From Employee
Where salary is null;
```

ترد كل الموظفين الذين ليس لديهم رواتب

9. العمليات على المجموعات (تعليمات الـ set operation

- تسمح SQL بالعمليات (except, union, intersect) المقابلة للاجتماع و التقاطع والفرق في الجبر العلاقاتي
- يجب أن تكون العلاقات التي تطبق عليها هذه العمليات متجانسة (لها نفس الواصفات).

نفس عدد الأعمدة ونفس النوع وحصراً نفس الترتيب

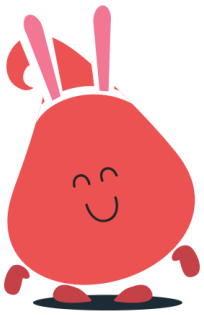
الاجتماع union

لإيجاد مجموعة كل الزبائن المتعاملين مع المصرف سواء مقترضين أو مودعين نكتب:

```
(select customer-name
From Depositor)
Union
(select customer-name
From borrower)
```

تحذف عملية الاجتماع الحدوديات المكررة آليّاً مثل عملية الاختيار , وفي حال الرغبة في الاحتفاظ بالتكرار نكتب union all

فتحوي النتيجة حدوديات مكررة بعدد ظهورها في كل من العلاقتين.



التقاطع intersect

لإيجاد الزبائن الذين حصلوا على قرض ولديهم حساب في المصرف نكتب:

```
(select distinct customer-name
From depositor)
Intersect
(select distinct customer-name
From borrower)
```

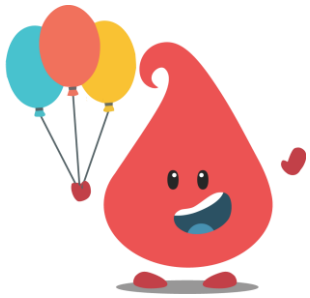
وتستخدم العبارة intersect all لإظهار الحدوديات المكررة وعندها تحتوي النتيجة الزبائن بعدد القروض والإيداعات المشتركة.

الفرق Except

لإظهار جميع الزبائن الذين لديهم حساب في المصرف ولم يستفيدوا من قرض منه نكتب:

```
(select distinct customer-name
From depositor)
Except
(select customer-name
From borrower)
```

يمكن استخدام العبارة except all لإظهار الحدوديات المكررة وبذلك نستطيع الحصول على أسماء الأشخاص الذين لديهم عدد من الحسابات أكبر من عدد القروض التي اقترضوها.



THE END