مدخل إلى الـ SQL

د. روان قرعوني

11/05/2023

RB Informatics;

قواعد معطيات 1

مقدمة:

تحدثنا في المحاضرات السابقة عن تمثيل مخططات الـERD وعن الجبر العلاقاتي وسنقوم بمراجعة أهم الأفكار معاً لندخل في موضوع جديد وهو لغة SQL.

مراجعة:

ما هي قاعدة المعطيات Database؟!

سابقاً كانت المعلومات تخزن على شكل ورقي (ملفات ورقية) تتطلب تعديل , تحوي تكرارات , لا يوجد ترابط بين هذه المعطيات المتواجدة مثل (طالب اسمه كذا مرتبط بالمدرسة كذا برقم كذا بالسنة كذا ...) ومع مرور الزمن والتطور أصبحت هذه الملفات الورقية تكتب على شكل ملفات الكترونية Notepad (كملفات الWord والExcel) ولكنها لم تعالج مشاكل الملفات الورقية.

ملف الكتروني
((notepad))

Ahmad	.....	Syriatel	salary
Hasan	.....	Syriatel	.....
Khaled	.....	Syriatel	.....
Wael	.....	MTN	.....
Omar	.....	MTN	.....

إن الملف التالي يوضح معلومات عن موظفين في شركات مختلفة.

And it's fine to fake it till you make it
till you do, till it's true.

ما هي مشاكل هذا الملف؟!

البحث:

لو مثلاً أردنا معرفة مَن من الموظفين يعمل في شركة سيرياتيل Syriatel سنحتاج إلى المرور على سطر سطر ومن الممكن أن يكون لدينا مليون موظف أي أننا في هذه الحالة سنحتاج إلى مسح مليون موظف (سطر)، أو مَن من الموظفين يعمل في شركة Syriatel فرع المزة أيضاً سنحتاج إلى عملية المسح السابقة وهنا نلاحظ أن عملية البحث صعبة جداً لأنها تتطلب المرور على كل أسطر هذا الملف واحداً تلو الآخر.

ترابط المعطيات:

ماذا لو أردنا معرفة عدد الموظفين في شركة MTN أو متوسط الرواتب لجميع الموظفين في Syriatel أو متوسط أعمار الموظفين بفرع المزة في MTN أو ..... إن استخراج هذه المعلومات بتطلب عملية معالجة كبيرة وضخمة وذلك لعدم وجود ترابطات بين المعطيات.

التعديل:

لنفرض أن مالكي Syriatel قاموا بتغيير الاسم التجاري لها وأصبحت Syriatel أقرب إليك بدلاً من Syriatel أو قاموا بتغيير عنوانها.

في هذه الحالة يترتب علينا القيام بعملية المسح السابقة لإيجاد كل موظف يعمل في Syriatel ومن ثم تعديل تلك البيانات ولكن احتمال كبير جداً حدوث خطأ في طريقة كتابة الاسم مثلاً (1000 موظف يعمل في Syriatel، تم تعديل بيانات 900 موظف وكان الاسم التجاري Syriatel أقرب إليك و100 موظف Syriatel) في هذه الحالة أصبح لدينا بيانات خاطئة ولن يتم اكتشاف مثل هذه الأخطاء (نادراً ما يتم كشفها).

التكرار:

نلاحظ أن اسم Syriatel وعنوانها وتاريخ تأسيسها تم تخزينهم بجانب كل موظف يعمل فيها.

أي تم ذكر هذه المعلومات بعدد العاملين في الشركة وهذا يعني أن حجم التخزين على الهارد كبير جداً.

(بفرض أن كل حرف يمثل بـ بايت ولدينا مليون موظف أي حجم التخزين على الهارد هو مليون بايت وهذا ضخم).

ومن هنا ظهرت الحاجة إلى البحث عن طرق أخرى لتخزين البيانات وإلى ظهور ما يسمى بجداول المعطيات Databases.

Databases:	عبارة عن جداول وعلاقات بين هذه الجداول كل جدول منها له خصائص attributes

It's not always easy.
But that's life. Be strong because there are better days ahead.✨

لنحول معاً الملف الإلكتروني السابق إلى Database:

لو أن الموظف تابع لشركة واحدة فالعلاقة مابين جدولي Employee-Company هي علاقة one-to-many

ولكن إذا انتقل موظف ما للعمل من شركة MTN إلى شركة SYRIATEL وأردنا حفظ ذلك هنا أصبح لدينا الموظف تابع لأكثر من شركة والشركة تحوي أكثر من موظف وهنا تحولت العلاقة من علاقة one-to-many إلى علاقة many-to-many

Employee

ID	name	address	age	CID
1	Ahmad	...	20	1
2	Hasan	...	25	1
3	Khaled	...	30	1
4	Wael	...	25	2
5	Omar	...	22	2
...	...	...	...	...

Company

ID	name	...
1	MTN	...
2	Syriatel	...
3	...	...
4	...	...
5	...	...
...	...	...

CID: company ID

نلاحظ أننا قمنا بتخزين كلاً من Syriatel وMTN لمرة واحدة فقط. وبدلاً من تخزين اسم الشركة التي يعمل فيها كل موظف قمنا بوضع الـID الخاص بالشركة ضمن حقل في جدول Employee أي عوضاً عن كتابة وتخزين

[1 Ahmad .... 20 MTN] قمنا بتخزين [1 Ahmad .... 20 1] أي قمنا باستبدال الMTN بـ1 (الـID الخاص بـMTN)

وذلك تجنبا للمشاكل التي قمنا بذكرها سابقاً وأن CID هذه تسمى Foreign key (FK).

كما نلاحظ أن الـID الخاص سواءً بالـEmployee أو بالـCompany هو تسلسلي و Unique أي ممنوع تكراره وهو مميز.

مثلاً: (1) يشير إلى MTN ولا يمكن أن يشير إلى MTN وSyriatel معاً و (2) يشير إلى Syriatel أي أن 1 محجوز وخاص لـMTN وهكذا وإن هذا الـID يسمى بـPrimary key (PK).

ملاحظة:

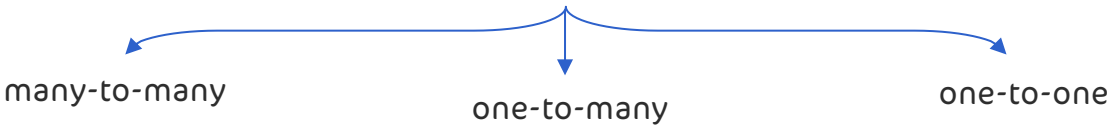
في جدول معطيات Students مثلاً.. من الممكن أن يكون هذا الـPrimary key هو رقم الطالب أو رقمه الامتحاني (بشرط أن يحقق شروط الـPK).

أصبح بإمكاننا الاستعلام عن الموظفين العاملين في شركة MTN مثلاً من خلال تحديد ID الـCompany باستعلام واحد كما أصبحت عمليات التعديل والبحث أسهل ولم يعد هناك تكرار.

Life is a journey to be experienced,
Not a problem to be solved.

أنواع العلاقات:

العلاقات بين الجداول تقسم إلى 3 أقسام:



1. one-to-one

مثال:

كل شخص لديه جواز سفر واحد وكل جواز سفر هو عائد لشخص واحد.

2. one-to-many

مثال:

(نضع الـ FK عند(من جهة) الـ many).

كل موظف تابع لشركة واحدة ولكن الشركة لديها أكثر من موظف.

نضع الـFK عند الـ Employee لأنه لو وضعناه عند الـ Company هنا سيصبح لدينا في جدول الـ Company لكل موظف Employee عمود وهذا مرفوض.

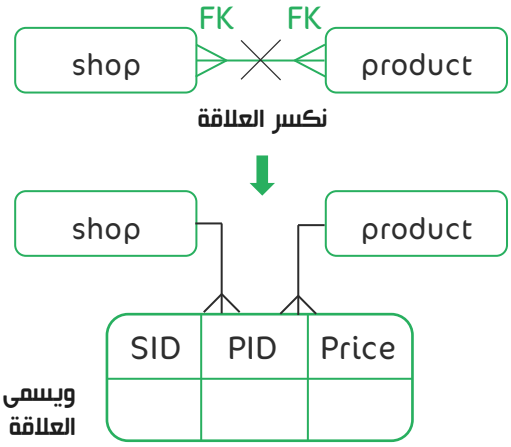
هذا يعني أنه يكفي وضع عمود واحد في جدول الـ Employee يشير إلى الـ Company.

ID	Name	Emp_1	Emp_2	Emp_3	...
1	MTN	...	...	Null	...
2	Syriatel	...	Null	...	Null
...	...	...	...	...	...
...	...	...	...	...	...

الفرق ما بين علاقة الـ one-to-one وعلاقة الـ one-to-many هو أن الـ FK في علاقة الـ one-to-one لا يمكن تكراره (unique).

It's your story.
Feel free to hit them with a plot twist at any moment.

3. many-to-many



مثال:

المنتج يباع في أكثر من محل (متجر)
والمتجر يحوي أكثر من منتج.

ملاحظة:

الواصفات التي توضع في جدول كسر العلاقة هي واصفات خاصة بالعلاقة ما بين الكيانين.

بالعودة للمثال السابق:

Employee

ID	name	address	age
1	Ahmad	...	20
2	Hasan	...	25
3	Khaled	...	30
4	Wael	...	25
5	Omar	...	22
...	...	...	...

Company

ID	name	...	...
1	MTN	...	...
2	Syriatel	...	...
3	...	...	...
4	...	...	...
5	...	...	...
...	...	...	...

~~$Email_1$ | $Email_2$ | $Email_3$ | ...~~

...	...	...	...

EID	CID	Salary

Work

إن الواصفة salary لا تنتمي
إلى الـ company وحدها
ولا إلى الـEmployee لوحده

ID	Email	EID
1		
2		
3		

Email

الواصفات متعددة القيم:

مثلاً موظف له email من الممكن أن يكون له أكثر من email وعوضاً عن وضع أعمدة بجدول الـ Employee نحول هذه الواصفة إلى جدول.

SQL

تعد أكثر لغات الاستعلام التجارية انتشاراً، وهي تركيب من لغة الجبر العلاقاتي الذي تحدثنا عنه سابقاً والحساب العلاقاتي.

تجمع لغة SQL إمكانات إضافية إلى الاستعلام من قاعدة المعطيات فتسمح بتعريف بنية المعطيات وإضافة وتعديل المعطيات في قاعدة المعطيات وبتحديد أمانها.

وكان يطلق عليها اسم sequel ثم جرى تطوير هذه اللغة وتغير اسمها إلى SQL (Structured Query Language).

تعريف بلغة SQL:

تتكون لغة SQL من عدة أجزاء:

- **لغة تعريف المعطيات DDL (Data Definition Language):**

تقدم التعليمات اللازمة لتعريف وتعديل مخطط علاقة، حذف علاقة، بناء فهارس.

- **لغة التعامل مع المعطيات DML Language (Interactive Data Manipulation):**

وتعتمد طريقة لصياغة الاستعلامات المرتكزة على الجبر العلاقاتي وجبر القضايا، وتحوي تعليمات لإضافة، وحذف وتعديل حدوديات في قاعدة المعطيات.

- **لغة التعامل مع المعطيات المضمنة Embedded:**

وهي مصممة للتضمين في لغات البرمجة الاعتيادية مثل PL/I، باسكال، C، كوبول.

- **تعريف المنظار:**

تحوي لغة تعريف المعطيات في SQL تعليمة تسمح بتعريف منظار.

- **السماحيات:**

تحوي لغة تعريف المعطيات في SQL تعليمات لتحديد حقوق الوصول إلى العلاقات والمناظير.

- **التكامل:**

تحوي DDL في SQL تعليمات لتحديد شروط التكامل.

- **التحكم في المناقلات Transaction Control:**

في SQL يوجد تعليمات لتحديد بداية المناقلة ونهايتها، كما يوجد تطويرات تسمح بقفل المعطيات للتحكم في الوصول المتزامن.

إنشاء قاعدة معطيات database (لغة تعريف المعطيات DDL)

باستخدام برنامج SQL-Server

بعد أن قمنا بإنشاء مخططات الـ ERD وفهمها أصبح بإمكاننا إنشاء قاعدة معطيات

⟶ New Database ⟶ Create Table

أعمدة + علاقات

وهذا يعني اسم العمود ونوعه

```
Create table Employee
( ID int, name varchar (50), Address varchar (50), age int,
 CID int, Primary Key (ID), Foreign Key(CID) references
company (ID), birthdate date,  )

Create table Company
( ID int, Name varchar(20), Primary Key (ID),  )
```

تُعرّف شروط التكامل على المخطط العلاقاتي والمتضمنة:

- عدم احتوائه على قيم غير معلومة not null
- تعريف المفتاح الرئيسي Primary Key
- تعريف قضية check (ρ) حيث ρ قضية.

إن تعريف المفتاح الرئيسي على واصفة يجعل اختبار عدم احتوائها على قيم غير معلومة آلياً.

تعريف المجالات بلغة SQL

يمكن تعريف أنماط مختلفة للمجالات بلغة SQL منها:

- **char (n):** لتعريف سلسلة أحرف ذات طول ثابت حيث n طول ثابت يحدده المستثمر.
- **varchar(n):** لتعريف سلسلة أحرف ذات طول متغير حيث n الطول الأعظمي الذي حدده المستثمر.
- **Int :** عدد صحيح.
- **number (p, d):** i رقم ممثل بالنقطة الثابتة.
- **real, double precision:** رقم ممثل بالفاصلة العائمة مع دقة مضاعفة.
- **Date:** لتاريخ يحوي 4 خانات للسنة.
- **Time:** الوقت في اليوم، ساعة، دقيقة، ثانية.

ملاحظات:

- يُسمح باستخدام القيم غير المعلومة في جميع أنماط المجالات، والتصريح يكون واصفة لا تقبل قيماً غير معلومة (يُحرّم استخدام القيم غير المعلومة لهذه الواصفة).
- يمكن للمستثمر أن يُعرف مجالات خاصة به باستخدام لغةSQL كما في المثال التالي:

```
create domain person-name char (20) not null
```

الواصفة المشتقة: هي واصفة يمكن استنتاجها من معلومات متوافرة. مثال:لدينا واصفة Birthdate منها يمكننا استنتاج العمر age وجعلها واصفة.

Delete And Insert

تسمح تعليمة drop table بحذف جميع المعلومات المتعلقة بالعلاقة من قاعدة المعطيات، وتسمح تعليمة alter table بإضافة واصفات إلى مخطط علاقة موجودة، وبحيث تأخذ هذه الواصفة قيمة غير معلومة في جميع الحدوديات الموجودة سابقاً في العلاقة.

```
Drop table Employee;
```

يجب حذف جدول الـ Employee أولاً ومن ثم جدول الـ company وليس العكس أي يجب حذف العلاقة التي تربط ما بين الكيانين أولاً ثم نحذف الكيانين المرتبطين.

```
Insert into table
insert into employee
values ( I, "Ahmad", "Al mazzah" 20,1)
```

تستخدم لإدخال سطر جديد على Entity (الجدول)

لنناقش معاً

عندما نريد القيام بعملية insert الموظف جديد في الشركة،

هل من المعقول أن نتذكر ID آخر موظف تم توظيفه حتى نتمكن من تحديد ID الموظف الجديد والقيام بـ عملية insert؟؟؟ بالتأكيد لا.

ولهذا وُجِد ما يسمّى بـ Identity بدلاً من الـ ID

بالعودة إلى create table Employee

(..... Identity (1,1) .....)

رقم البداية ← مقدار الزيادة

وتصبح عملية الـ insert كالتالي:

```
insert into employee
values ("Ahmad", “Al mazzah”, 20,1)
```

أي استغنينا عن الـ ID لوجود الـ Identity التلقائية.

إدخال أعمدة محددة:

في هذه الحالة كل الأعمدة التي لم يتم تحديدها و إسناد قيم لها يتم وضع null فيها.

```
insert into employee (Name, age, CID)
values ("Ahmad", 20, 1);
```

حذف سطر محدد:

```
Delete from employee
where ID=5;
```

ملاحظة هامة :

عدم كتابة where condition أي كتابة Delete from employee; تعمل عملاً مشابهاً لـ drop table employee.

الفرق أن drop تحذف الجدول مع البيانات كاملاً أمّا delete تحذف البيانات (الأسطر) أي لا يزال بإمكاننا القيام بعملية insert.

تعديل جدول بعد بنائه :

```
Alter table Employee Add email varchar (50)
```

مثلاً:

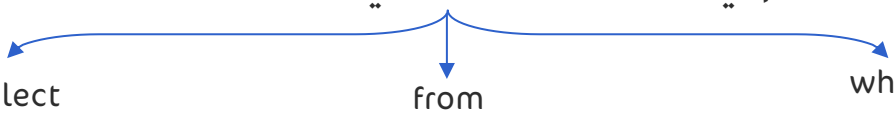
نملأ جدولاً يحوي 1000 موظف وقمنا باستخدام تعليمة Alter table.... يصبح لدينا عمود Email لـ1000 موظف بإسناد قيمته null.

ولتعبئتهم يمكن استيراد ملف Excel كـ table في SQL Server.

لغة الاستعلام:

البنية الأساسية للاستعلام في لغة SQL.

تتألف البنية الأساسية للاستعلام في لغة SQL من ثلاثة أجزاء هي:



يُعبّر الجزء select عن عملية الإسقاط في الجبر العلاقاتي، ويُستخدم الجزء from لتحديد العلاقات المستخدمة في عملية الاختيار، ويُكافئ الجداء الديكارتي في الجبر العلاقاتي أما الجزء where فيتضمن قضية منطقية يجب أن تحققها الواصفات الموجودة في العلاقة التي جرى تحديدها في جزء from.

ويصبح الشكل العام للاستعلام في لغة SQL كما يلي:

select $A_1, A_2, ..., A_n$
from $r_1, r_2, ..., r_m$
where p

حيث A_i: هي واصفات

r_j: علاقات

p: قضية

وهي تكافئ في الجبر العلاقاتي العملية التالية:

$$\prod_{A_1-A_m} (\sigma_p(r_1 * r_2 * ....* r_m))$$

قاعدة معطيات حول المصارف:

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)

- select

تستخدم لتحديد قائمة الواصفات المطلوب إظهارها في نتيجة الاستعلام.

مثال:

لإيجاد أسماء جميع الأفرع في علاقة القروض نستخدم التعليمة:

```
select branch-name
from loan;
```

والنتيجة هي علاقة تحوي واصفاً واحداً (branch-name).

- تسمح لغة SQL بتكرار الحدوديات في العلاقة أو في نتيجة الاستعلام على عكس الجبر العلاقاتي.
- ولحذف التكرار نضيف كلمة distinct بعد select

```
select distinct branch-name
from loan;
```

- وتستخدم كلمة all لمنع حذف التكرار في الحدوديات.

مثال:

```
select all branch-name
from loan;
```

- يستخدم الرمز * للدلالة على اختيار جميع الواصفات من علاقة.

مثال:

```
select *
from loan;
```

- ويمكن أن تحوي select تعابير حسابية تستخدم العمليات (+,-,*, /).

مثال:

```
select branch-name, loan-number, amount *100
from loan;
```

where

تستخدم لتحديد الشروط التي يجب أن تحققها الحدوديات المختارة فمثلاً لإيجاد أرقام القروض في الفرع "perryridge" والتي تزيد عن 1200 نكتب:

```
select loan-number
from loan
where branch-name ="perryridge" and amount >1200;
```

وتستخدم المعاملات المنطقية not, and, or في where وعمليات المقارنة <>, =, <=, >=,>, <, between not between

مثال:

```
where amount between 90000 and 100000;
```

From:

تعرّف الجداء الديكارتي بين العلاقات المراد اختيار الحدوديات منها، ولمّا كانت عملية الدمج تعرّف كجداء ديكارتي وعملية اختيار وإسقاط، فإنه يمكننا التعبير بلغة SQL عن عملية الدمج التالية:

$$\prod_{customer-name,loan-number} (borrower \bowtie loan)$$

بالشكل:

```
select distinct customer-name, borrower. Loan-number
From borrower, loan
where borrower. Loan-number= loan. Loan-number;
```

ويلاحظ في المثال السابق أننا استخدمنا relation name. attribute name في select بسبب وجود نفس اسم الواصف في أكثر من علاقة وذلك لتجنب الالتباس.

The End.

A winner is a dreamer who never gives up.