الجمهورية العربية السورية
وزارة التعليم العالي
جامعة حمص
الكلية التطبيقية
قسم تقنيات حاسوب

قواعد معطيات 1

القسم العملي

المحاضرة السادسة

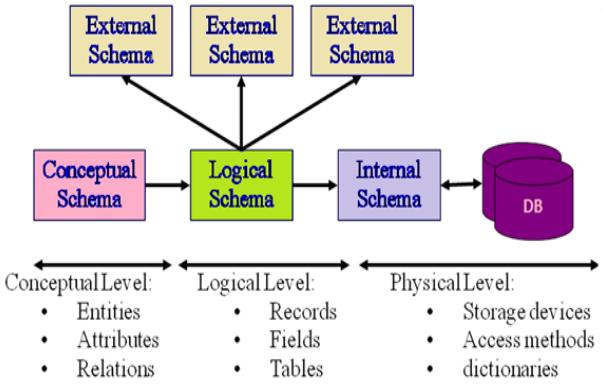
Structure Query Language

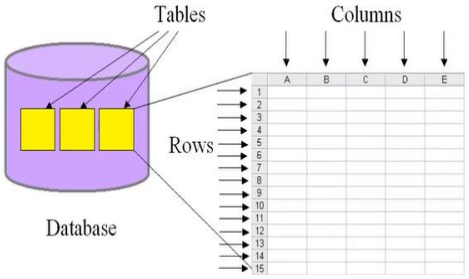
(SQL)

اعداد المدرسين:

م . زينب مراد

م . بتول الليوس





الجبر العلائقي

RELATIONAL ALGEBRA

الاختيار و الإسقاط (العرض) selection and projection

في الجبر العلائقي هنالك عمليتان أساسيتان الأولى لاختيار الصفوف وتسمى الاختيار، ويرمز لها بالرمز (σ) حيث يتم انتقاء بعض الصفوف التي ينطبق عليها شرط محدد واستثناء الصفوف التي لا ينطبق عليها الشرط.

تقوم باختيار مجموعة من الحدوديات التي تحقق شرطًا معينًا من علاقة، سنرمز إلى العملية كما يلي:

$$\sigma_{condition}(Relation)$$

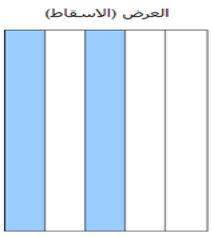
حيث condition هو شرط يجب أن تحققه الحدوديات المختارة .

والثانية لعرض الأعمدة وتسمى الإسقاط (العرض) و يرمز لها بالرمز (π)

وهي عملية وحيدة المعامل تسمح بانتقاء بعض الواصفات من العلاقة، نرمز إلى هذه العملية

بالشكل:

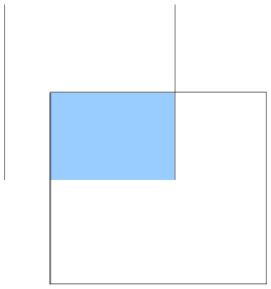
$$\Pi_{selected\ attributes}(Relation)$$



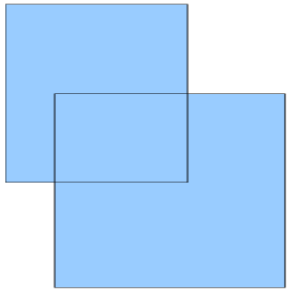
عمليات المجموعات set operations:

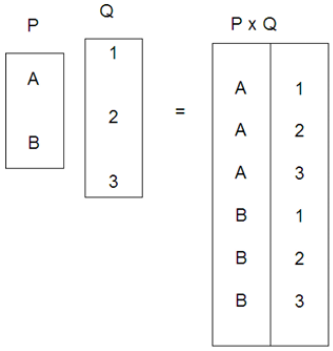
الاتحاد (∪) التقاطع (∩) الفرق (-) و الضرب الكارتيزي (×) هي عمليات معرفة على المجموعات و أيضا تعتبر جزء من الجبر العلائقي.

Intersept operation

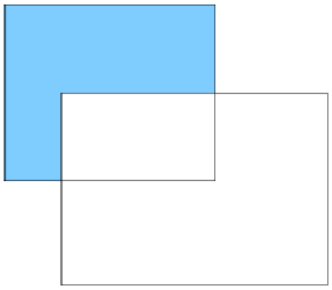


Union operation





Cartesian Product



Difference / Except operation

- لتكن لدينا قاعدة المعطيات لمكتبة مؤلفة من جدول المستخدمين وجدول الكتب وجدول الاستعارة

User (uid : integer, uname : string, rating : integer, age : real)

Book (bid : integer, btitle: string, type: string)

Borrow (uid: integer, bid: integer, date: Date)

$S_1 \cap S_2$

uid	uname	Rating	Age
31	عبد الستار	8	55.5
58	عبد الصمد	10	35.0

$S_1 \cup S_2$

uid	uname	Rating	Age
22	عبد الرحمن	7	45.0
28	عبد السلام	9	25.0
31	عبد الستار	8	55.5
44	عبد المنعم	5	25.0
58	عبد الصمد	10	35.0

تجسيد S1 للعلاقة users

uid	Uname	Rating	Age
22	عبد الرحمن	7	45.0
31	عبد الستار	8	55.5
58	عبد الصمد	10	35.0

تجسيد S2 للعلاقة users

uid	Uname	Rating	Age
28	عبد السلام	9	25.0
31	عبد الستار	8	55.5
44	عبد المنعم	5	25.0
58	عبد الصمد	10	35.0

$S_1 - S_2$

uid	uname	Rating	Age
22	عبد الرحمن	7	45.0

مثال شامل:سوف نعرض فيما يلي العمليات الأساسية والإضافية في الجبر العلاقاتي موضحين طريقة عملها بأمثلة على قاعدة المعطيات المتعلقة بالمصارف والمعرفة بالمخطط العلاقاتي التالي:

Loan-Schema (loan-number, amount, branch-name)
Customer-Schema(customer-name, customer-street, customer-city)
Borrower-Schema(customer-name, loan-number)

Employee-Schema(employee-name, phone-number)
Loan-Officer-Schema(banker-name, customer-name, loan-number)
Depositor-Schema (customer-name, account-number)
Account-Schema(branch-name,account-number, balance)
Branch-Schema(branch-name, branch-city, assets)

customer-name	*loan-number*
Adams	L-16
Curry	L-93
Hayes	L-15
Jackson	L-14
Jones	L-17
Smith	L-11
Smith	L-23
Williams	L-17

loan-number	*branch-name*	*amount*
L-11	Round Hill	900
L-14	Downtown	1500
L-15	Perryridge	1500
L-16	Perryridge	1300
L-17	Downtown	1000
L-23	Redwood	2000
L-93	Mianus	500

branch-name	*branch-city*	*assets*
Brighton	Brooklyn	7100000
Downtown	Brooklyn	9000000
Mianus	Horseneck	400000
North Town	Rye	3700000
Perryridge	Horseneck	1700000
Pownal	Bennington	300000
Redwood	Palo Alto	2100000
Round Hill	Horseneck	8000000

عملية الاختيارSelect:

لنأخذ مخطط العلاقة التالية:

Loan-Schema (branch-name, loan-number, amount)

لاختيار مجموعة الحدوديات القروض للفرع "branch-name = "perryridge نكتب العبارة التالية:

$$\sigma_{\text{branch-name = " perryridge"}}(\text{Loan})$$

يمكن أن يحوي الشرط المطبق في عملية الاختيار عمليات مقارنة= :،<>، >، <، >=، <= ومعاملات منطقية and, or, not ويمكن أن تطبق هذه المعاملات بين قيم الواصفات المكونة للعلاقة.

لاختيار القروض التي منحها الفرع "branch-name = "perryridge والتي لا تقل عن 1200 نكتب:

$$\sigma_{\text{branch-name = " perryridge"} \wedge \text{ amount >1200}}(\text{Loan})$$

لاختيار مجموعة الزبائن الذين يحملون نفس اسم المسؤول عن القرض الذي اقترضوه من علاقة loan-officerذات المخطط التالي:

Loan-officer = (customer-name, banker-name, loan-number)

نكتب:

$$\sigma_{\text{customer-name = banker-name}}(\text{Loan-Officer})$$

عملية الإسقاط Projection:

لنفترض أننا نريد الحصول على أرقام القروض ومبالغها دون أن نهتم بالأسماء الأفرع.

يكتب الاستعلام السابق بالشكل:

$$\Pi_{\text{loan–number ,amount}}(\text{Loan})$$

وبفرض أن علاقة القروض loan ممثلة بالجدول التالي:

loan-number	branch-name	Amount

تكون العلاقة الناتجة من عملية الإسقاط من الشكل:

loan-number	Amount

تركيب العمليات العلاقاتية

لإيجاد أسماء الزبائن الذين يعيشون في مدينة "Harrison".

$$\Pi_{\text{customer –name}}(\sigma_{\text{customer –city = " Harrison"}}(\text{Customer}))$$

ونقصد بها تركيب عمليتين :اختيار الحدوديات التي تحقق الشرط (مدينة= "Harrison") وإسقاط العلاقة الناتجة من العملية السابقة على العمود customer-name.

عملية الاجتماع Union Operation

للحصول على جميع الزبائن الذين يتعاملون مع المصرف (الذين لديهم حساب أو اقترضوا قرضًا أو للحصول على كلا الفريقين).

نحتاج إلى إيجاد مجموعة الزبائن الذين لديهم حساب في المصرف، وهي معلومات موجودة في علاقة depositor وإلى إيجاد مجموعة الزبائن الذين اقترضوا من المصرف، وهي معلومات موجودة في علاقةborrower ثمّ إلى إجراء عملية الاجتماع بين المجموعتين ومن ثم يكون التعبير الناتج هو:

$$\Pi_{\text{customer –name}}(\text{Depositor}) \cup \Pi_{\text{customer- name}}(\text{Borrower})$$

عملية الفرق Difference Operation:

لإيجاد مجموعة الزبائن الذين لديهم حساب مصرفي ولم يقترضوا من المصرف نكتب:

$$\Pi_{\text{customer –name}}(\text{Depositor}) - \Pi_{\text{customer –name}}(\text{Borrower})$$

عملية التقاطع:

تجرى هذه العملية بين علاقتين ويجب أن تحقق هاتان العلاقتان الشروط المذكورة في عملية الاجتماع .ونتيجة عملية التقاطع هي علاقة تحوي مجموعة الحدوديات الموجودة ضمن العلاقتين، التعبير الموافق لهذه العملي هو:

$$R \cap S = R - (R - S)$$

للحصول على أسماء الزبائن الذين لديهم حساب واقترضوا قرضًا من المصرف

$$\Pi_{\text{customer-name}}(\text{Depositor}) \cap \Pi_{\text{customer-name}}(\text{Borrower})$$

عملية الجداء الديكارتي Cartesian Product:

إذا أردنا الحصول على جميع الزبائن الذين اقترضوا من المصرف الفرع ""perryridge" للحصول على هذه المعلومات نحتاج إلى المعلومات الموجودة في كلتا العلاقتينLoan و Borrowerوتعطي عملية اختيار الحدوديات، المتعلقة بالفرع المطلوب من جداء العلاقتين معلومات عن الزبائن والقروض المأخوذة من الفرع المطلوب، ولكن ليست المعلومات المطلوبة،وللحصول على المعلومات المطلوبة نكتب:

$$\Pi_{customer-name}\left(\sigma_{borrower-loan-number=loan-number}\left(\sigma_{branch-name=''perryridge''}(Borrower \times Loan)\right)\right)$$

العمليات على المجموعات في SQL

تسمح لغة sql بعمليات الاجتماع UNION ,التقاطع INTERSECT و الفرق EXCEPT

ويجب أن تحوي العلاقات التي تطبق عليها هذه العمليات نفس السمات (الواصفات).

شروط دمج قيم الجداول

يجب مراعاة الشروط التالية حتى يسمح لك بدمج قيم الجداول في جدول واحد:

- عدد أعمدة الجداول التي نريد دمجها يجب أن يكون متطابق، فمثلاً يكون الجدولين الذين نريد دمجهما يتألفان من 3 أعمدة.
- أعمدة الجداول التي سيتم دمجها يجب أن تكون من نفس النوع، فمثلاً لا يمكنك دمج عمود نوعه `INT` مع عمود نوعه `DATE` .
- يجب أن تراعي ترتيب الأعمدة التي تريد دمجها مع بعضها، أي يجب أن تجلبها بنفس الترتيب.

UNION الاجتماع : تستخدم لدمج الاسطر بين عدة جداول بدون تكرار

سبق و تعلمنا أنه يمكن استخدام الامر select لجلب القيم من أي جدول ، سواء أردنا جلبها كلها أو تحديد القيم

ل التي نريد جلبها منها و في النهاية النتيجة النهائية سيتم إرجاعها لنا كجدول.

- في حال أردت تنفيذ أكثر من أمرselect في ذات الوقت ومن ثم وضع النتيجة في جدول واحد يمكنك استخدام

UNION ALL و UNION حتى تدمجهم مع بعض.

عند دمج قيم الجداول من الطبيعي أن يكون هناك أسطر قيمها مكررة خاصةً إن كانت الجداول تتضمن عدد كبير من الأسطر.

لهذا السبب قاعدة البيانات تتيح لك تحديد ما إن كنت تريد أن يحتوي الجدول الذي سينتج عند دمج قيم الجداول على قيم مكررة أم لا.

- إذا لم ترد وجود قيم مكررة، قم بدمج الجداول بواسطة العامل `UNION` .
- إذا كان لا يهمك ما إن كان يوجد قيم مكررة أم لا، قم بدمج الجداول بواسطة العامل `UNION ALL` .

دمج قيم الجداول مع عدم وضع قيم مكررة:

```
SELECT column_name(s) FROM table1          -- هنا نحدد أسماء الأعمدة التي نريد جلبها بالترتيب من الجدول الأول
UNION                                      -- العامل UNION سيقوم بدمج نتيجة الأمرين SELECT في جدول واحد مع عدم وضع قيم مكررة
SELECT column_name(s) FROM table2;         -- هنا نحدد أسماء الأعمدة التي نريد جلبها بالترتيب من الجدول الثاني
```

دمج قيم الجداول مع السماح بوجود قيم مكررة:

```
SELECT column_name(s) FROM table1          -- هنا نحدد أسماء الأعمدة التي نريد جلبها بالترتيب من الجدول الأول
UNION ALL                                  -- UNION ALL سيقوم بدمج نتيجة الأمرين SELECT في جدول واحد مع السماح بوجود قيم مكررة
SELECT column_name(s) FROM table2;         -- هنا نحدد أسماء الأعمدة التي نريد جلبها بالترتيب من الجدول الثاني
```

مثال: لو عندنا جدولين: موظفين قدامى وموظفين جدد، ونريد قائمة موحدة بالأسماء

```
SELECT Name FROM OldEmployees
UNION
SELECT Name FROM NewEmployees;
```

النتيجة : جميع الأسماء في الجدولين بدون تكرار

INTERSECT التقاطع : تستخدم لدمج الاسطر المشتركة فقط بين عدة جداول

عملية تقاطع النتائج ما بين الاستعلامات، أي إظهار النتائج المشتركة فقط ما بين الاستعلامات

التي يربط بينها المعامل INTERSECT

الصيغة العامة :

```
SELECT column_list FROM Table1          --هنا نحدد أسماء الأعمدة التي نريد جلبها بالترتيب من الجدول الأول
INTERSECT                               -- سيقوم بدمج نتيجة الأمرين SELECT في جدول واحد (الاسطر المشتركة بين الجداول)
SELECT column_list FROM Table2;         -- هنا نحدد أسماء الأعمدة التي نريد جلبها بالترتيب من الجدول الثاني
```

مثال: لو أردنا معرفة الموظفين الذين ظهروا في كلا الجدولين (قدامى وجدد)

```
SELECT Name FROM OldEmployees
INTERSECT
SELECT Name FROM NewEmployees;
```

النتيجة : ستظهر أسماء الموظفين المشتركة فقط بين الجدولين

EXCEPT الفرق : تستخدم لجلب البيانات الموجودة في سطر او عدة اسطر من جدول وغير موجودة في جدول آخر

الصيغة العامة :

```
SELECT Name FROM Table1
EXCEPT
SELECT Name FROM table2;
```

مثال: لو أردنا معرفة الموظفين الذين كانوا قدامى ولم يعودوا موجودين في الجدد

```
SELECT Name FROM OldEmployees
EXCEPT
SELECT Name FROM NewEmployees;
```

النتيجة : الأسماء الموجودة في الأول وغير موجودة في الثاني

JOIN

نستخدم عبارة الربط JOIN اذا اردنا الربط بين عدة جداول من قاعدة المعطيات أو دمج أسطر من جدولين أو أكثر .

آلية الربط :

إضافة كل سطر من الجدول الثاني الى نهاية كل سطر من الجدول الأول ولكن الأسطر التي سوف يتم عليها الدمج تحدد من خلال نوع الربط.

أنواع الربط :

INNER JOIN الربط الداخلي:

الصيغة العامة للربط الداخلي:

```
Select  < Select_list> From table1 inner join table2
On table1.primarykey = table2.foreignkey
```

الصيغة الأخرى للربط الداخلي:

```
Select <Select_list>  From table1 , table2
Where table1.primarykey =  table2.foreignkey
```

ملاحظات:

- يمكن استبدال Inner join ب Join .
- عند عدم ذكر نوع الربط فهو ربط داخلي .
- عند إعطاء اسم مستعار للجدول (خاصة عند التمييز بين الحقول المتشابهة (الاسم نفسه في جدولين مختلفين) يجب استخدام الاسم المستعار دائما" والا تصبح العبارة خاطئة لو استخدمنا مرة الاسم المستعار ومرة الاسم الحقيقي.

مثال : إظهار الموظفين مع أسماء أقسامهم.

```
SELECT Employees.Name, Departments.DeptName
FROM Employees
INNER JOIN Departments
ON Employees.DeptID = Departments.DeptID;
```

النتيجة: فقط الموظفين الذين لهم قسم مطابق

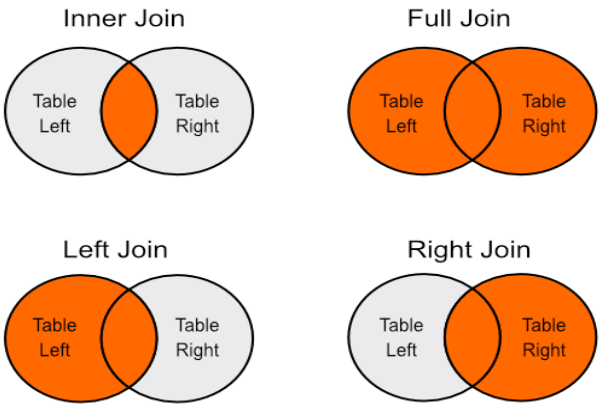
Outer join (Left , Right , Full) الربط الخارجي :

تظهر جميع سجلات الجدول الأول (يحدد الجدول الأول حسب جهة الربط) مع مقابلاتها من الجدول الثاني اعتمادا" ذات قيم على عمود أو أكثر أما سجلات الجدول الأول التي ليس لها مقابل من الجدول الثاني فتقابل بسجلات NULL

الصيغة العامة للربط الخارجي:

Select from table1 [full | left | right] outer join table2

On table1.**primarykey**=table2.**foreignkey**



مثال: اظهار جميع الموظفين حتى لو قسمهم غير موجود

SELECT Employees.Name, Departments.DeptName FROM Employees

LEFT JOIN **Departments** ON **Employees.DeptID** = **Departments.DeptID;**

النتيجة : تظهر أسماء كل الموظفين وأسماء اقسامهم واذا لم يوجد قسم يظهرNULL

مثال: اظهار جميع الأقسام حتى لو ليس لديهم موظفين

SELECT Employees.Name, Departments.DeptName FROM Employees

RIGHT JOIN **Departments** ON **Employees.DeptID** = **Departments.DeptID;**

النتيجة : تظهر أسماء كل الأقسام وأسماء الموظفين فيهم واذا لم يوجد اسم موظف يظهرNULL

مثال: اظهار جميع الموظفين وجميع الأقسام مع دمج المطابقات

SELECT Employees.Name, Departments.DeptName FROM Employees

FULL JOIN **Departments** ON **Employees.DeptID** = **Departments.DeptID;**

النتيجة :اسماء كل الموظفين وأسماء كل الأقسام سواء كان هناك تطابق أم لا

:CROSS JOIN

نحتاج له من أجل الحصول على الجداء الديكارتي لجدولين

الصيغة العامة: تأخذ احدى الشكلين:

Select from table1**,**table2

Select from table1 **cross join** table2

:SELF JOIN

نحتاج له من أجل نتائج من الصعب الحصول عليها بالاستعلامات البسيطة

ملاحظة: يجب إعطاء أسماء مستعارة للجدول من أجل التمييز بين الحقول

مثال: اظهار أسماء الموظفين الذين يعملون في المشروعين ١ و٢ معا"

```
select w1.essn from WORKS_ON w1 ,WORKS_ON w2
 where w1.essn =w2.essn and w1.pno=1 and w2.pno =2
```

مثال: ليكن لدينا الجدولين التاليين :

Employee (employee-name, street, city)
Ft-works (employee-name, branch-name, salary)

Employee

employee-name	street	city
A	S1	C1
B	S2	C1
C	S4	C2
D	S5	C3

Ft-Works

employee-name	branch-name	salary
A	Br1	500
B	Br2	700
D	Br1	1220
F	Br3	2000

: Inner join / natural join / join

Employee ⋈ Ft-works

employee-name	street	city	branch-name	salary
A	S1	C1	Br1	500
B	S2	C1	Br2	700
D	S5	C3	Br1	1220

Left outer join (1 :Outer join

employee ⟕ ft-works

employee-name	street	city	branch-name	salary
A	S1	C1	Br1	500
B	S2	C1	Br2	700
C	S4	C2	null	null
D	S5	C3	Br1	1220

Right outer join (2

Employee ⟖ Ft-works

employee-name	street	city	branch-name	salary
A	S1	C1	Br1	500
B	S2	C1	Br2	700
D	S5	C3	Br1	1220
F	null	null	Br3	2000

Full outer join (3

Employee ⟗ ft-works

employee-name	street	city	branch-name	salary
A	S1	C1	Br1	500
B	S2	C1	Br2	700
D	S5	C3	Br1	1220
F	null	null	Br3	2000
C	S4	C2	null	null