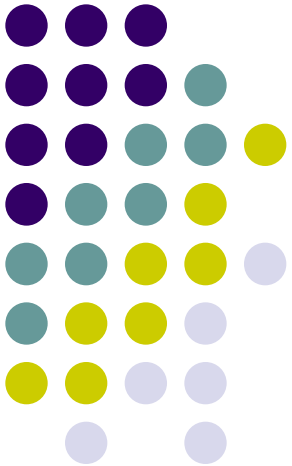
Applied Faculty
Hama University

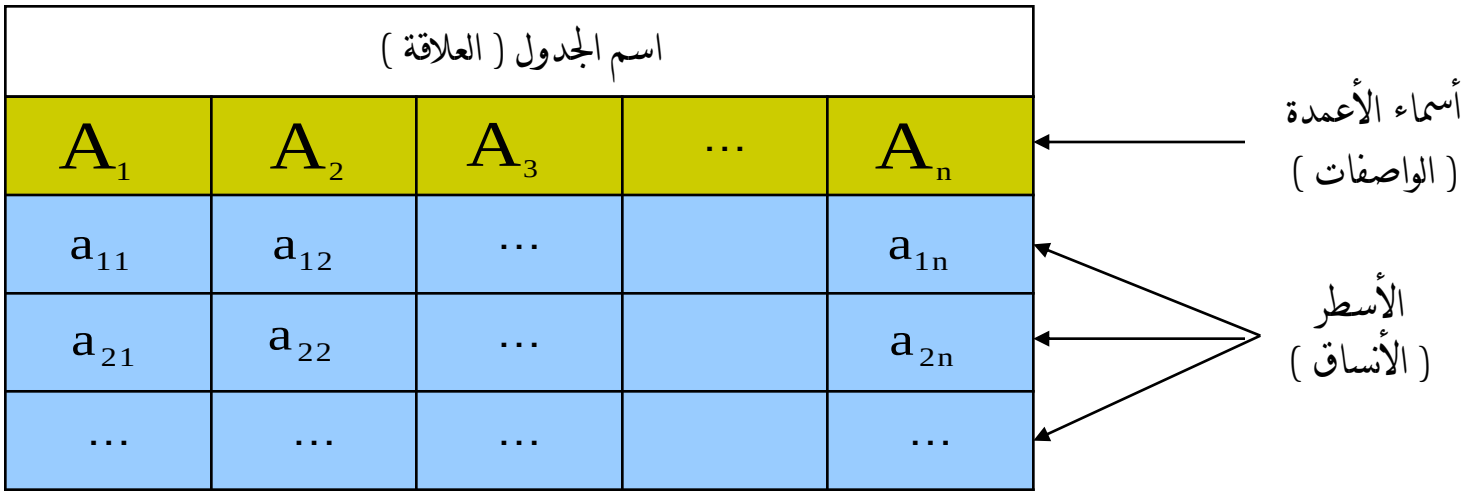
Database 2

مراجعة عامة

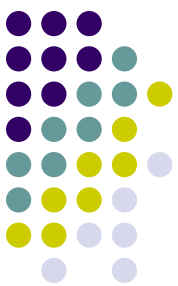


قواعد البيانات العلائقية

- العلاقة هي البنية المنطقية الأساسية في النموذج العلائقي
- تمثل العلاقة على شكل جدول
  - الأعمدة = واصفات العلاقة = حقول
  - الأسطر = أنساق العلاقة = سجلات



مثال

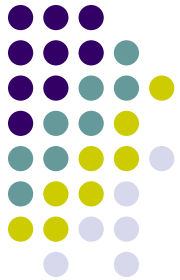


EMPLOYEE				
EMP_ID	**EMP_NAME**	**EMP_MGR**	**TITLE**	**EMP_DEPT**
1234	Green	***null***	President	40
4567	Gilmore	1234	Senior VP	40
1045	Rose	4567	Director	10
9876	Smith	1045	Accountant	10

المفتاح الرئيسي

- عمود أو مجموعة من أعمدة الجدول تستخدم كمعرف لتمييز أسطر الجدول بشكل وحيد .

- وحيد
  - لا يوجد سطران مختلفان لهما قيم متطابقة عند جميع واصفات المفتاح
- غير خالي (not null)
  - لا يمكن لأي واحدة من واصفات المفتاح أن تكون خالية null

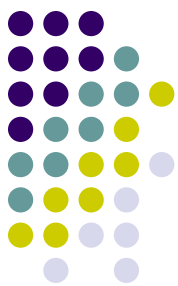


المفتاح الخارجي

- عمود أو مجموعة من أعمدة الجدول تستخدم للربط بين بيانات الجداول في قاعدة البيانات.

- قيم المفتاح الخارجي عند سطر ما :
  - إما null (سجل غير مرتبط)
  - أو تأخذ قيمها من قيم مفتاح رئيسي عند سطر معين من جدول آخر في قاعدة البيانات (سجل مرتبط)

مثال



EMPLOYEE

EMP_ID	EMP_NAME	EMP_MGR	TITLE	EMP_DEPT
1234	Green	*null*	President	40
4567	Gilmore	1234	Senior VP	40
1045	Rose	4567	Director	10
9876	Smith	1045	Accountant	10

DEPARTMENT

ID	NAME	LOCATION
10	Accounting	New York
40	Sales	Miami

الجبر العلائقي

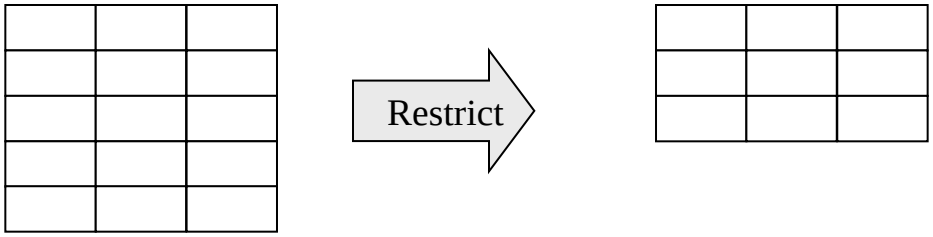
- العمليات العلائقية :
  - مجموعة العمليات التي تسمح بمعالجة الجداول في قاعدة البيانات
  - كل عملية تأخذ كدخل علاقة أو أكثر ، و تنتج كخرج علاقة واحدة

- عمليات رئيسية
  - القسر و الإسقاط و الربط
- عمليات إضافية
  - الجداء الديكارتي و الاجتماع و التقاطع و الفرق و القسمة

القسر (Restrict)

- عملية أحادية: تأخذ علاقة واحدة كدخل و تعطي علاقة واحدة كخرج
- يمكن أن تعتبر تشريح أفقي للجدول
  - العلاقة الخرج تتألف من مجموعة جزئية من أسطر العلاقة الدخل التي تحقق شرطاً معيناً

$$\sigma_{<condition>}(R)$$



القسر – مثال

Modules			
ModuleName	**Level**	**CourseCode**	**StaffNo**
Relational Database Systems	1	CSD	244
Relational Database Design	1	CSD	244
Deductive Databases	3	CSD	445
Object-Oriented Databases	3	CSD	445
Distributed Databases	2	CSD	247

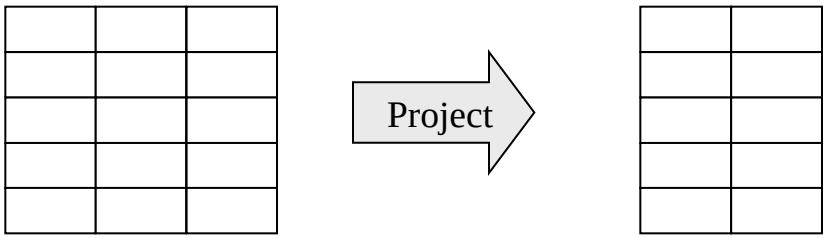
$$R1 = \sigma_{level=1}(Modules)$$

R1			
ModuleName	**Level**	**CourseCode**	**StaffNo**
Relational Database Systems	1	CSD	244
Relational Database Design	1	CSD	244

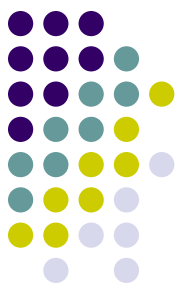
الإسقاط (Project)

- عملية أحادية: تأخذ علاقة واحدة كدخل و تعطي علاقة واحدة كخرج
- يمكن أن تعتبر تشريح عمودي للجدول
  - العلاقة الخرج تتألف من مجموعة جزئية من أعمدة العلاقة الدخل

$$\pi_{<attribute\ \ list>}(R)$$



الإسقاط – مثال



Modules			
ModuleName	**Level**	**CourseCode**	**StaffNo**
Relational Database Systems	1	CSD	244
Relational Database Design	1	CSD	244
Deductive Databases	3	CSD	445
Object-Oriented Databases	3	CSD	445
Distributed Databases	2	CSD	247

$$R1 = \pi_{\text{ModuleName}}(\text{Modules})$$

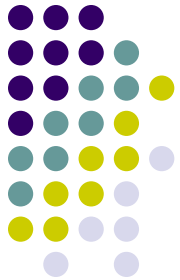
R1
ModuleName
Relational Database Systems
Relational Database Design
Deductive Databases
Object-Oriented Databases
Distributed Databases

الربط (Join)

- عملية ثنائية : تأخذ علاقتين كدخل و تعطي علاقة واحدة كخرج
- تجميع لجدولين معاً و لكن نأخذ فقط السجلات التي تتوافق فيها أعمدة الربط من الجدولين:
  - المفتاح الرئيسي من أحد الجدولين و المفتاح الخارجي من الجدول الآخر

$$R_1 \bowtie_{<join\ condition>} R_2$$

الربط – مثال



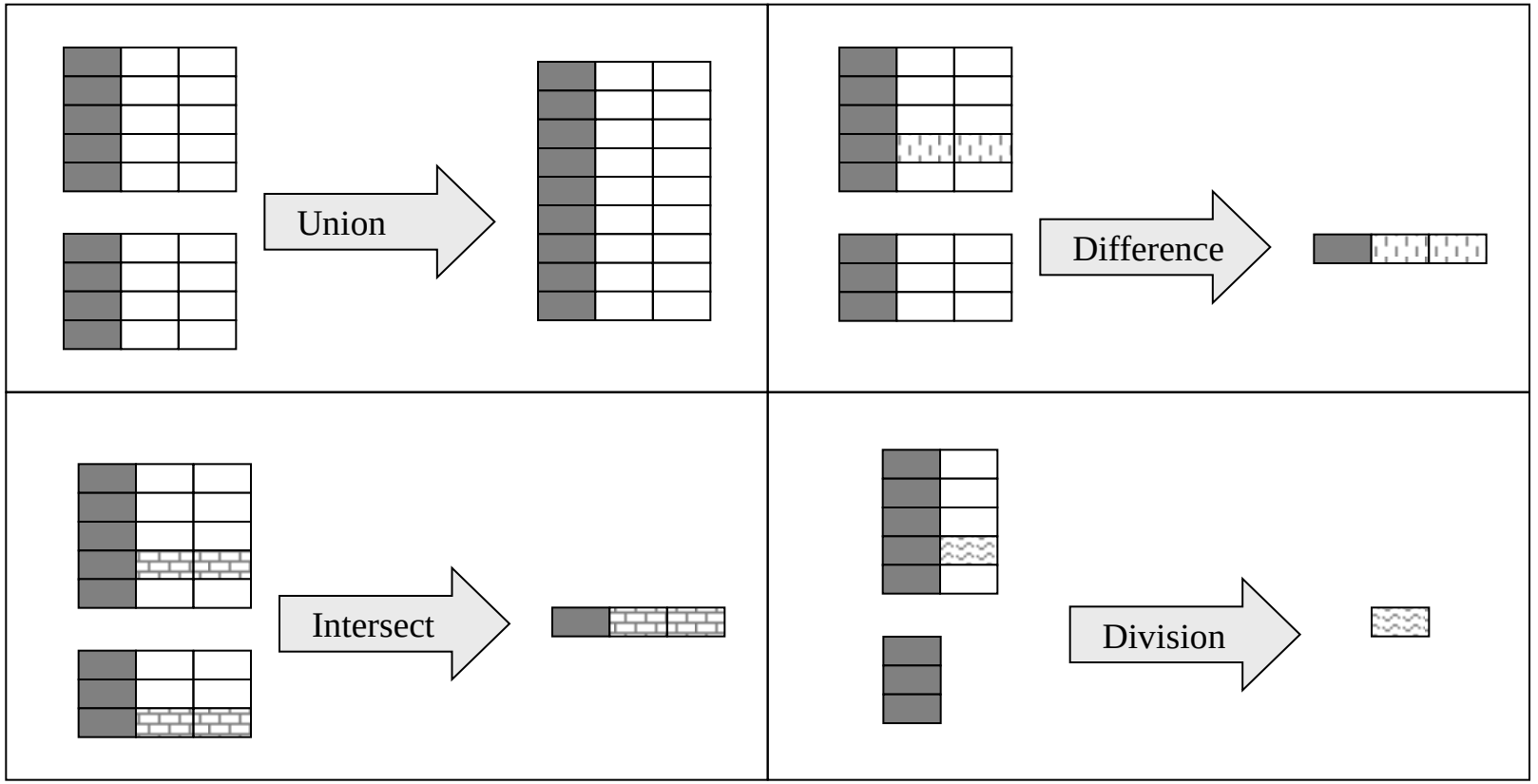
Modules			
ModuleName	**Level**	**CourseCode**	**StaffNo**
Relational Database Systems	1	CSD	244
Relational Database Design	1	CSD	244
Deductive Databases	3	CSD	445
Object-Oriented Databases	3	CSD	445
Distributed Databases	2	CSD	247

Lecturers		
StaffNo	**StaffName**	**Status**
244	Davies T	L
247	Patel S	SL
445	Evans R	PL

$$R1 = Modules \bowtie Lecturers$$

R1					
ModuleName	**Level**	**CourseCode**	**StaffNo**	**StaffName**	**Status**
Relational Database Systems	1	CSD	244	Davies T	L
Relational Database Design	1	CSD	244	Davies T	L
Deductive Databases	3	CSD	445	Evans R	PL
Object-Oriented Databases	3	CSD	445	Evans R	PL
Distributed Databases	2	CSD	247	Patel S	SL

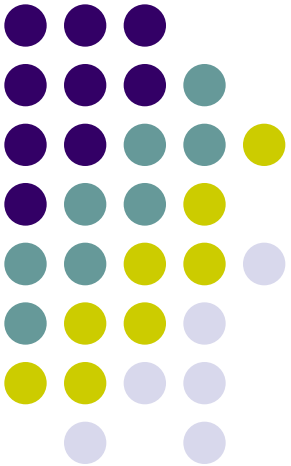
العمليات الإضافية



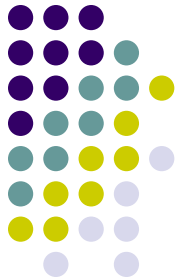
القسم الثاني

لغة الاستعلام البنيوية

SQL



إنشاء جدول

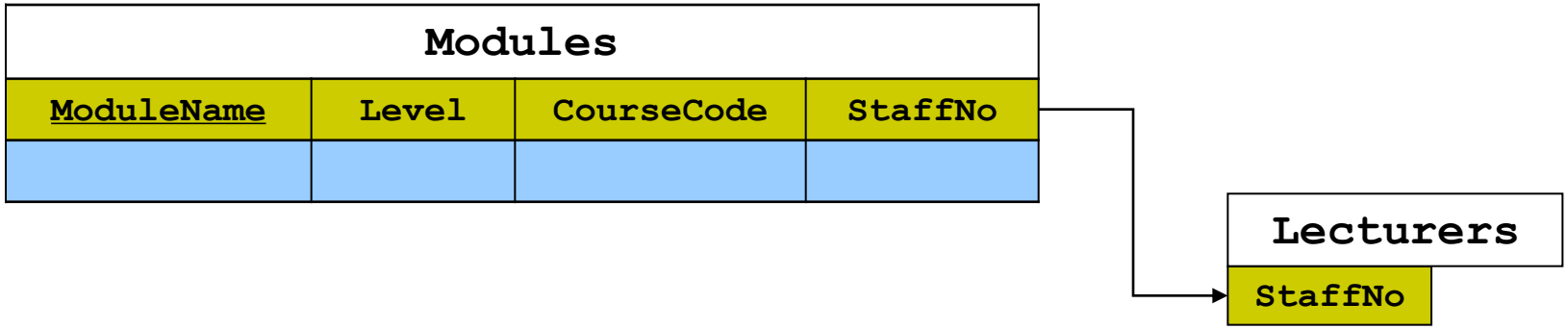


```
CREATE TABLE <tableName> (
   columnName1 type NOT NULL,
   columnName2 type,
   ...          ...,

   PRIMARY KEY (PK),
   FOREIGN KEY (FK) REFERENCES RefTable(RefColumn)
);
```

إنشاء جدول – مثال

```
CREATE TABLE Modules (
  moduleName VARCHAR2(15),
  level      SMALLINT,
  courseCode VARCHAR2(3),
  staffNo    INTEGER,
  PRIMARY KEY (moduleName),
  FOREIGN KEY (staffNo) REFERENCES Lecturers (staffNo)
);
```



حذف و تعديل جدول

حذف جدول

```
DROP TABLE Modules;
```

تغيير اسم عمود

```
ALTER TABLE Modules
RENAME COLUMN level_ TO level1;
```

إضافة عمود

```
ALTER TABLE Lecturers
ADD COLUMN roomNo SMALLINT;
```

تغيير نمط عمود

```
ALTER TABLE Lecturers
ALTER COLUMN staffName VARCHAR2(20);
```

حذف عمود

```
ALTER TABLE Lecturers
DROP COLUMN staffName;
```

إدراج و حذف أسطر

إدراج أسطر

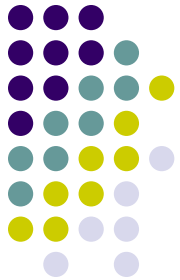
```
INSERT INTO Modules
VALUES ('Relational Database Systems', 1, 'CSD', 244);
```

```
INSERT INTO Modules (moduleName, level, courseCode, staffNo)
VALUES ('Relational Database Systems', 1, 'CSD', 244);
```

حذف أسطر

```
DELETE FROM Modules
WHERE moduleName = 244;
```

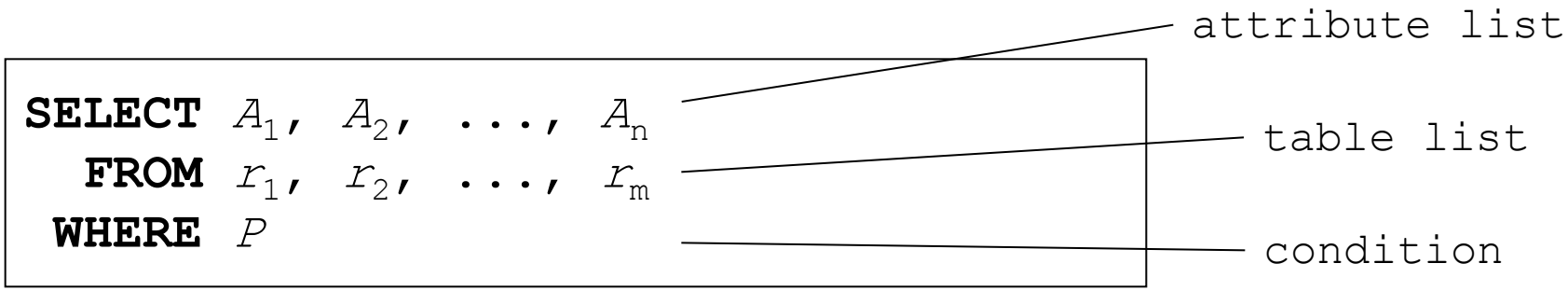
تعديل أسطر



```
UPDATE Modules
SET staffNo = 250
WHERE moduleName = 'Relational Database Systems';
```

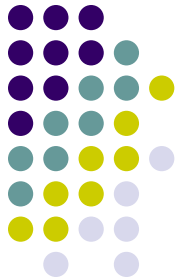
```
UPDATE account
SET balance = CASE
                WHEN balance <= 10000
                  THEN balance * 1.05
                  ELSE balance * 1.06
                END
```

الاستعلامات



$$\pi_{A_1,A_2,\ldots,A_n}(\sigma_P(r_1 \times r_2 \times \cdots \times r_m))$$

عبارة SELECT



```
SELECT branch_name
  FROM loan
```

$$\pi_{branch\_name}(loan)$$

اختيار عمود واحد

```
SELECT DISTINCT branch_name
  FROM loan
```

اختيار عمود واحد بقيم غير مكررة

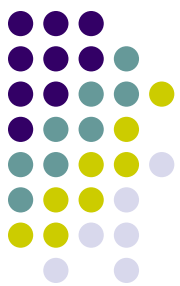
```
SELECT *
  FROM loan
```

اختيار جميع أعمدة جدول

```
SELECT loan_number, branch_name, amount * 100
FROM loan
```

استخدام عملية حسابية

عبارة WHERE

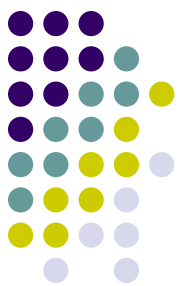


```
SELECT loan_number
  FROM loan
 WHERE branch_name = 'Perryridge'
   AND amount > 1200
```

$$\pi_{loan\_number}(\sigma_{branch\_name='Perryridge' \wedge amount>1200}(loan))$$

```
SELECT loan-number
  FROM loan
 WHERE amount BETWEEN 90000 AND 100000
```

عبارة FROM



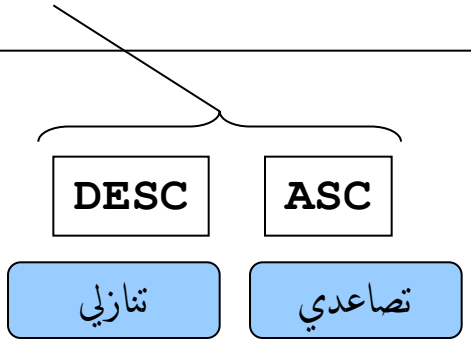
```
SELECT *
  FROM borrower, loan
```

جداء ديكارتي

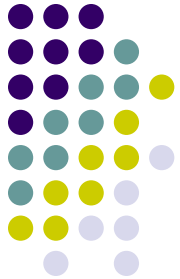
```
SELECT customer_name, borrower.loan_number, amount
  FROM borrower, loan
 WHERE borrower.loan_number = loan.loan_number
   AND branch_name = 'Perryridge'
```

ترتيب الأسطر

```
  SELECT DISTINCT customer_name
    FROM borrower, loan
   WHERE borrower.loan_number = loan.loan_number
         AND branch_name = 'Perryridge'
ORDER BY customer-name
```



عمليات المجموعات



الاجتماع

```
(SELECT customer_name FROM depositor)
 UNION
(SELECT customer_name FROM borrower)
```

التقاطع

```
(SELECT customer_name FROM depositor)
 INTERSECT
(SELECT customer_name FROM borrower)
```

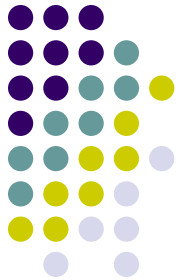
الاستثناء

```
(SELECT customer_name FROM depositor)
 EXCEPT
(SELECT customer_name FROM borrower)
```

توابع التجميع

AVG	القيمة المتوسطة
MIN	القيمة الصغرى
MAX	القيمة العظمى
SUM	مجموع القيم
COUNT	عدد القيم

توابع التجميع – أمثلة



Find the average account balance at the Perryridge branch.

```
SELECT AVG (balance)
FROM account
WHERE branch_name = 'Perryridge'
```

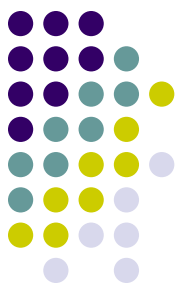
Find the number of tuples in the customer relation.

```
SELECT COUNT(*)
FROM customer
```

Find the number of depositors in the bank.

```
SELECT COUNT(DISTINCT customer_name)
FROM depositor
```

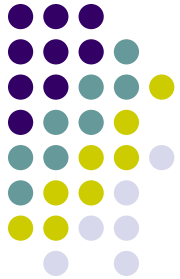
GROUP BY – توابع التجميع



Find the number of depositors for each branch

```
  SELECT branch_name, COUNT(DISTINCT customer_name)
    FROM depositor, account
   WHERE depositor.account_number = account.account_number
GROUP BY branch_name
```

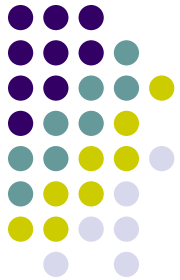
HAVING – توابع التجميع



Find the names of all branches where the average account balance is more than $1,200.

```
  SELECT branch_name, AVG (balance)
    FROM account
GROUP BY branch_name
  HAVING AVG (balance) > 1200
```

الاستعلامات المتداخلة



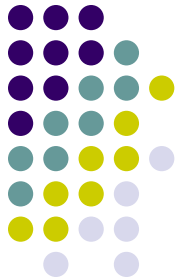
Find all customers who have both an account and a loan at the bank

```
SELECT DISTINCT customer_name
FROM borrower
WHERE customer_name IN
      (SELECT customer_name
       FROM depositor )
```

Find all customers who have a loan at the bank but do not have an account at the bank

```
SELECT DISTINCT customer_name
FROM borrower
WHERE customer_name NOT IN
      (SELECT customer_name
       FROM depositor )
```

الاستعلامات المتداخلة

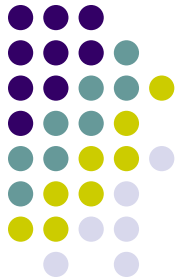


Find all customers who have both an account and a loan at the Perryridge branch

```
SELECT DISTINCT customer_name
  FROM borrower, loan
 WHERE borrower.loan_number = loan.loan_number
   AND branch-name = 'Perryridge'
   AND (branch-name, customer_name) IN
        (SELECT branch_name, customer_name
         FROM depositor, account
         WHERE depositor.account_number =
               account.account_number )
```

```
CREATE VIEW V AS <query expression>
```

المناظير



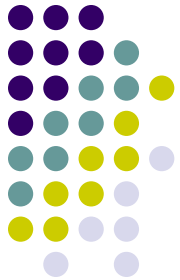
A view consisting of branches and their customers

```
CREATE VIEW all_customer AS
   (SELECT branch_name, customer_name
    FROM depositor, account
    WHERE depositor.account_number = account.account_number)
    UNION
   (SELECT branch_name, customer_name
    FROM borrower, loan
    WHERE borrower.loan_number = loan.loan_number)
```

Find all customers of the Perryridge branch

```
SELECT customer-name
  FROM all_customer
 WHERE branch-name = 'Perryridge'
```

ربط الجداول



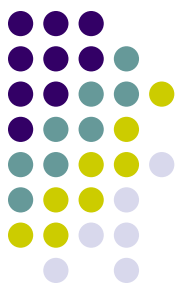
Join Types
inner join left outer join right outer join full outer join

Join Conditions
natural on <predicate> using (A_1, A_2, ..., A_n)

Loan		
loan_number	**branch_name**	**amount**
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

Borrower	
customer_name	**loan_number**
Jones	L-170
Smith	L-230
Hayes	L-155

INNER JOIN – ربط الجداول



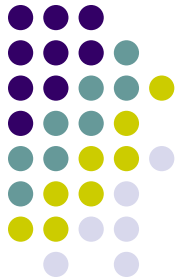
```
SELECT *
  FROM loan INNER JOIN borrower
    ON loan.loan_number = borrower.loan_number
```

loan_number	branch_name	amount	customer-name	loan-number
L-170	Downtown	3000	Jones	L-170
L-230	Redwood	4000	Smith	L-230

```
SELECT *
  FROM loan NATURAL INNER JOIN borrower
```

loan_number	branch_name	amount	customer-name
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith

LEFT OUTER JOIN – ربط الجداول



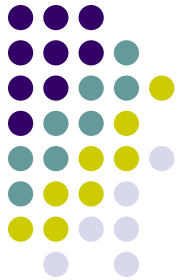
```
SELECT *
  FROM loan LEFT OUTER JOIN borrower
    ON loan.loan_number = borrower.loan_number
```

loan_number	branch_name	amount	customer-name	loan-number
L-170	Downtown	3000	Jones	L-170
L-230	Redwood	4000	Smith	L-230
L-260	Perryridge	1700	*null*	*null*

```
SELECT *
  FROM loan NATURAL LEFT OUTER JOIN borrower
```

loan_number	branch_name	amount	customer-name
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-260	Perryridge	1700	*null*

RIGHT OUTER JOIN – ربط الجداول



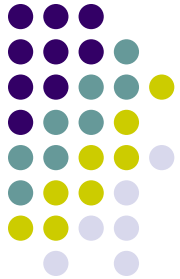
```
SELECT *
  FROM loan RIGHT OUTER JOIN borrower
    ON loan.loan_number = borrower.loan_number
```

loan_number	branch_name	amount	customer-name	loan-number
L-170	Downtown	3000	Jones	L-170
L-230	Redwood	4000	Smith	L-230
null	*null*	*null*	Hayes	L-155

```
SELECT *
  FROM loan NATURAL RIGHT OUTER JOIN borrower
```

loan_number	branch_name	amount	customer-name
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-155	*null*	*null*	Hayes

FULL OUTER JOIN – ربط الجداول



```
SELECT *
  FROM loan FULL OUTER JOIN borrower
 USING (loan_number)
```

loan_number	branch_name	amount	customer-name
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-260	Perryridge	1700	*null*
L-155	*null*	*null*	Hayes