

INTRODUCTION TO SQL

DISPLAYING DATA FROM MULTIPLE TABLES (JOIN)

WEEK 8



By Kanokwan Atchariyachanvanich

Faculty of Information Technology

KMITL

Database System Concepts

2/2561

OUTLINE

Date	SQL
11~12-MAR-2019	13. Displaying Data from Multiple Tables (JOIN)
18~19-MAR-2019	14. Displaying Data from Multiple Tables (OUTER JOIN)
25~26-MAR-2019	QUIZ # 2: SELECT, JOIN
1~2-APR-2019	15. Group functions 14. Subqueries
8~9-APR-2019	NO CLASS
15~16-APR-2019	NO CLASS
22~23-APR-2019	QUIZ #3: GROUP FUNCTION, SUBQUERIES
29~30-APR-2019	16. Controlling transaction
6~7-MAY-2019	Project Presentation – Application
15 MAY-2019	FINAL examination

OBJECTIVE

- After completing this lesson, you should be able to do the following:
- Generate a Cartesian product of all rows from two or more tables
- Write SELECT statements to access data from more than one table using equijoins and nonequijoins
- Join a table to itself by using a self-join

RECALL: CARTESIAN PRODUCT

R

	P_CODE	P_DESCRIPTOR	PRICE
▶	123456	Flashlight	\$5.26
	123457	Lamp	\$25.15
	123458	Box Fan	\$10.99
	213345	9v battery	\$1.92
	254467	100V bulb	\$1.47
	311452	Powerdrill	\$34.99

S

	STORE	aisle	shelf
▶	23	vW	5
	24	K	9
	25	Z	6

*** Attributes of both relations have distinct names

3 rows

PRODUCT

yields

C

	P_CODE	P_DESCRIPTOR	PRICE	STORE	aisle	shelf
▶	123456	Flashlight	\$5.26	23	vW	5
	123456	Flashlight	\$5.26	24	K	9
	123456	Flashlight	\$5.26	25	Z	6
	123457	Lamp	\$25.15	23	vW	5
	123457	Lamp	\$25.15	24	K	9
	123457	Lamp	\$25.15	25	Z	6
	123458	Box Fan	\$10.99	23	vW	5
	123458	Box Fan	\$10.99	24	K	9
	123458	Box Fan	\$10.99	25	Z	6
	213345	9v battery	\$1.92	23	vW	5
	213345	9v battery	\$1.92	24	K	9
	213345	9v battery	\$1.92	25	Z	6
	311452	Powerdrill	\$34.99	23	vW	5
	311452	Powerdrill	\$34.99	24	K	9
	311452	Powerdrill	\$34.99	25	Z	6
	254467	100V bulb	\$1.47	23	vW	5
	254467	100V bulb	\$1.47	24	K	9
	254467	100V bulb	\$1.47	25	Z	6

SELECT *
FROM R, S ;

18 rows

6 rows

CARTESIAN PRODUCTS

```
SELECT *  
FROM   nametable1, nametable2;
```

- **A Cartesian product is formed when:**
 - **A join condition is omitted**
 - **A join condition is invalid**
 - **All rows in the first table are joined to all rows in the second table**
- **To avoid a Cartesian product, always include a valid join condition**

GENERATING A CARTESIAN PRODUCT

EMPLOYEES (107 rows)

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
100	Steven	King	SKING	515.123.4567	17-JUN-87	AD_PRES	24000			90
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89	AD_VP	17000		100	90
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	17000		100	90
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-90	IT_PROG	9000		102	60
104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-91	IT_PROG	6000		103	60
105	David	Austin	DAUSTIN	590.423.4569	25-JUN-97	IT_PROG	4800		103	60
106	Valli	Pataballa	VPATABAL	590.423.4560	05-FEB-98	IT_PROG	4800		103	60
107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-99	IT_PROG	4200		103	60
108	Nancy	Greenberg	NGREENBE	515.124.4569	17-AUG-94	FI_MGR	12000		101	100
109	Daniel	Faviet	DFAVIET	515.124.4169	16-AUG-94	FI_ACCOUNT	9000		108	100
110	John	Chen	JCHEN	515.124.4269	28-SEP-97	FI_ACCOUNT	8200		108	100

DEPARTMENTS (27 rows)

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700
40	Human Resources	203	2400
50	Shipping	121	1500
60	IT	103	1400
70	Public Relations	204	2700
80	Sales	145	2500
90	Executive	100	1700
100	Finance	108	1700
110	Accounting	205	1700
120	Treasury		1700
130	Corporate Tax		1700
140	Control And Credit		1700
150	Shareholder Services		1700
160	Benefits		1700
170	Manufacturing		1700
180	Construction		1700
190	Contracting		1700
200	Operations		1700

SELECT *
FROM employees, departments;

Cartesian Product:
107 x 27 = 2889 rows



EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
100	Steven	King	SKING	515.123.4567	17-JUN-87	AD_PRES	24000			90
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89	AD_VP	17000		100	90
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	17000		100	90
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-90	IT_PROG	9000		102	60
104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-91	IT_PROG	6000		103	60
105	David	Austin	DAUSTIN	590.423.4569	25-JUN-97	IT_PROG	4800		103	60
106	Valli	Pataballa	VPATABAL	590.423.4560	05-FEB-98	IT_PROG	4800		103	60
107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-99	IT_PROG	4200		103	60
108	Nancy	Greenberg	NGREENBE	515.124.4569	17-AUG-94	FI_MGR	12000		101	100
109	Daniel	Faviet	DFAVIET	515.124.4169	16-AUG-94	FI_ACCOUNT	9000		108	100
110	John	Chen	JCHEN	515.124.4269	28-SEP-97	FI_ACCOUNT	8200		108	100
111	Ismael	Sciarra	ISCIARRA	515.124.4369	30-SEP-97	FI_ACCOUNT	7700		108	100

2889 rows selected.

TYPES OF JOINS

- Joins that are compliant with the SQL:1999 standard include the following:
 1. Cross joins (ผลลัพธ์เหมือนกับ Cartesian product)
 2. Equijoin (Old-style join)
 3. Natural joins
 4. Join with ON clause
 5. Join with USING clause
 6. Self-Joins Using the ON Clause
 7. Full (or two-sided) outer joins

1. CREATING CROSS JOINS

- The CROSS JOIN clause produces the cross-product of two tables.
- This is also called a Cartesian product between the two tables.

```
SELECT *  
FROM      employees  
CROSS JOIN departments ;
```

Cartesian product

```
SELECT *  
FROM      employees, departments ;
```


GENERATING A CROSS JOIN

EMPLOYEES (107 rows)

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
100	Steven	King	SKING	515.123.4567	17-JUN-87	AD_PRES	24000			90
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89	AD_VP	17000		100	90
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	17000		100	90
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-90	IT_PROG	9000		102	60
104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-91	IT_PROG	6000		103	60
105	David	Austin	DAUSTIN	590.423.4569	25-JUN-97	IT_PROG	4800		103	60
106	Valli	Pataballa	VPATABAL	590.423.4560	05-FEB-98	IT_PROG	4800		103	60
107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-99	IT_PROG	4200		103	60
108	Nancy	Greenberg	NGREENBE	515.124.4569	17-AUG-94	FI_MGR	12000		101	100
109	Daniel	Faviet	DFAVIET	515.124.4169	16-AUG-94	FI_ACCOUNT	9000		108	100
110	John	Chen	JCHEN	515.124.4269	28-SEP-97	FI_ACCOUNT	8200		108	100

**SELECT *
FROM employees
CROSS JOIN departments;**

DEPARTMENTS (27 rows)

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700
40	Human Resources	203	2400
50	Shipping	121	1500
60	IT	103	1400
70	Public Relations	204	2700
80	Sales	145	2500
90	Executive	100	1700
100	Finance	108	1700
110	Accounting	205	1700
120	Treasury		1700
130	Corporate Tax		1700
140	Control And Credit		1700
150	Shareholder Services		1700
160	Benefits		1700
170	Manufacturing		1700
180	Construction		1700
190	Contracting		1700
200	Operations		1700

**Cartesian Product:
107 x 27 = 2889 rows**



EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
100	Steven	King	SKING	515.123.4567	17-JUN-87	AD_PRES	24000			90
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89	AD_VP	17000		100	90
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	17000		100	90
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-90	IT_PROG	9000		102	60
104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-91	IT_PROG	6000		103	60
105	David	Austin	DAUSTIN	590.423.4569	25-JUN-97	IT_PROG	4800		103	60
106	Valli	Pataballa	VPATABAL	590.423.4560	05-FEB-98	IT_PROG	4800		103	60
107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-99	IT_PROG	4200		103	60
108	Nancy	Greenberg	NGREENBE	515.124.4569	17-AUG-94	FI_MGR	12000		101	100
109	Daniel	Faviet	DFAVIET	515.124.4169	16-AUG-94	FI_ACCOUNT	9000		108	100
110	John	Chen	JCHEN	515.124.4269	28-SEP-97	FI_ACCOUNT	8200		108	100
111	Ismail	Sciarra	ISCIARRA	515.124.4369	30-SEP-97	FI_ACCOUNT	7700		108	100

2889 rows selected.

OBTAINING DATA FROM MULTIPLE TABLES

EMPLOYEES

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
101	Kochhar	90
...		
202	Fay	20
205	Higgins	110
206	Gietz	110

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
10	Administration	1700
20	Marketing	1800
50	Shipping	1500
60	IT	1400
80	Sales	2500
90	Executive	1700
110	Accounting	1700
190	Contracting	1700




EMPLOYEE_ID	DEPARTMENT_ID	DEPARTMENT_NAME
200	10	Administration
201	20	Marketing
202	20	Marketing
...		
102	90	Executive
205	110	Accounting
206	110	Accounting

2. EQUIJOIN (OLD-STYLE JOIN)

- used to combine rows from two or more tables, based on a related column between them.
- Cartesian product with condition **WHERE**

```
SELECT table1.column, table2.column
FROM   nametable1, nametable2
WHERE  nametable1.column_name = nametable2.column_name ;
```



Cartesian product

EXAMPLE: EQUIJOIN

Query: Find all orders with their product names and price.

ORDER

ORDER_NO	CUSTOMER_NO	P_CODE
1	C001	111110
2	C002	222220

PRODUCT

P_CODE	P_NAME	PRICE
222220	คอมพิวเตอร์	30000
111110	สมุด	120
333330	ปากกา	500

```
SELECT *  
  
FROM ORDER, PRODUCT  
  
WHERE  
  
ORDER.P_CODE =  
PRODUCT.P_CODE ;
```

EXAMPLE : EQUIJOIN

ORDER

ORDER_NO	CUSTOMER_NO	ORDER.P_CODE
1	C001	111110
2	C002	222220

PRODUCT

PRODUCT.P_CODE	P_NAME	PRICE
222220	คอมพิวเตอร์	30000
111110	สมุด	120
333330	ปากกา	500

Automatically
rename

Step 1: Cartesian Product

ORDER_NO	CUSTOMER_NO	ORDER.P_CODE	PRODUCT.P_CODE	P_NAME	PRICE
1	C001	111110	222220	คอมพิวเตอร์	30000
1	C001	111110	111110	สมุด	120
1	C001	111110	333330	ปากกา	500
2	C002	222220	222220	คอมพิวเตอร์	30000
2	C002	222220	111110	สมุด	120
2	C002	222220	333330	ปากกา	500

EXAMPLE : EQUIJOIN

Step 2 : Select Join attribute **P_CODE** values are equal

ORDER_NO	CUSTOMER_NO	ORDER.P_CODE	PRODUCT.P_CODE	P_NAME	PRICE
1	C001	111110	222220	คอมพิวเตอร์	30000
1	C001	111110	111110	สมุด	120
1	C001	111110	333330	ปากกา	500
2	C002	222220	222220	คอมพิวเตอร์	30000
2	C002	222220	111110	สมุด	120
2	C002	222220	333330	ปากกา	500



ORDER_NO	CUSTOMER_NO	ORDER.P_CODE	PRODUCT.P_CODE	P_NAME	PRICE
1	C001	111110	111110	สมุด	120
2	C002	222220	222220	คอมพิวเตอร์	30000

EXAMPLE : EQUIJOIN

Step 3 : use a Projection to eliminate the duplicate attributes

ORDER_NO	CUSTOMER_NO	ORDER.P_CODE	PRODUCT.P_CODE	P_NAME	PRICE
1	C001	111110	111110	สมุด	120
2	C002	222220	222220	คอมพิวเตอร์	30000



```
SELECT *  
FROM  ORDER, PRODUCT  
WHERE ORDER.P_CODE = PRODUCT.P_CODE ;
```

ORDER_NO	CUSTOMER_NO	P_CODE	P_NAME	PRICE
1	C001	111110	สมุด	120
2	C002	222220	คอมพิวเตอร์	30000

2. EQUIJOIN

EMPLOYEE_ID	DEPARTMENT_ID
100	90
101	90
102	90
103	60
104	60
105	60
106	60
107	60
108	100
109	100
110	100


Foreign Key

DEPARTMENT_ID	DEPARTMENT_NAME
10	Administration
20	Marketing
30	Purchasing
40	Human Resources
50	Shipping
60	IT
70	Public Relations
80	Sales
90	Executive
100	Finance
110	Accounting


Primary Key

DEPARTMENTS
department_id
department_name
manager_id
location_id

EMPLOYEES

employee_id
first_name
last_name
email
phone_number
hire_date
job_id
salary
commission_pct
manager_id
department_id

2. RETRIEVING RECORDS WITH EQUIJOIN

จงเขียน SQL Query แสดงรหัสพนักงาน นามสกุล รหัสแผนก และรหัสสถานที่ของแผนกนั้นๆ

```
SELECT    employees.employee_id, employees.last_name,  
          employees.department_id, departments.department_id,  
          departments.location_id  
  
FROM      employees, departments  
  
WHERE     employees.department_id = departments.department_id ;
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID	LOCATION_ID
200	Whalen	10	10	1700
201	Hartstein	20	20	1800
202	Fay	20	20	1800
114	Raphaely	30	30	1700
115	Khoo	30	30	1700
116	Baida	30	30	1700
....				
113	Popp	100	100	1700
205	Higgins	110	110	1700
206	Gietz	110	110	1700

106 rows selected.

ADDITIONAL SEARCH CONDITIONS USING THE AND OPERATOR

To display employee Matos' employee_id, department number and department name.

```
SELECT    employees.employee_id, employees.department_id,  
          departments.department_name  
FROM      employees, departments  
WHERE     employees.department_id = departments.department_id  
AND       last_name = 'Matos';
```

EMPLOYEE_ID	DEPARTMENT_ID	DEPARTMENT_NAME
143	50	Shipping



USING TABLE ALIASES

- Use table aliases to simplify queries.
- Use table aliases to improve performance.

Table Aliases

```
SELECT e.employee_id, e.last_name,  
       d.location_id, d.department_id  
FROM   employees e, departments d  
WHERE  e.department_id = d.department_id;
```

EMPLOYEE_ID	LAST_NAME	LOCATION_ID	DEPARTMENT_ID
200	Whalen	1700	10
201	Hartstein	1800	20
202	Fay	1800	20
114	Raphaely	1700	30
115	Khoo	1700	30
116	Baida	1700	30

....

106 rows selected.

GUIDELINES FOR **TABLE ALIASES**

- Table aliases can be up to 30 characters in length, but shorter aliases are better than longer ones.
- If a table alias is used for a particular table name in the FROM clause, then that table alias must be substituted for the table name throughout the SELECT statement.
- Table aliases should be meaningful.
- The table alias is valid for only the current SELECT statement.
- Help keep SQL coder smaller therefore using less memory

JOINING MORE THAN TWO TABLES

EMPLOYEES

LAST_NAME	DEPARTMENT_ID
Whalen	10
Hartstein	20
Fay	20
Raphaely	30
Khoo	30
Baida	30

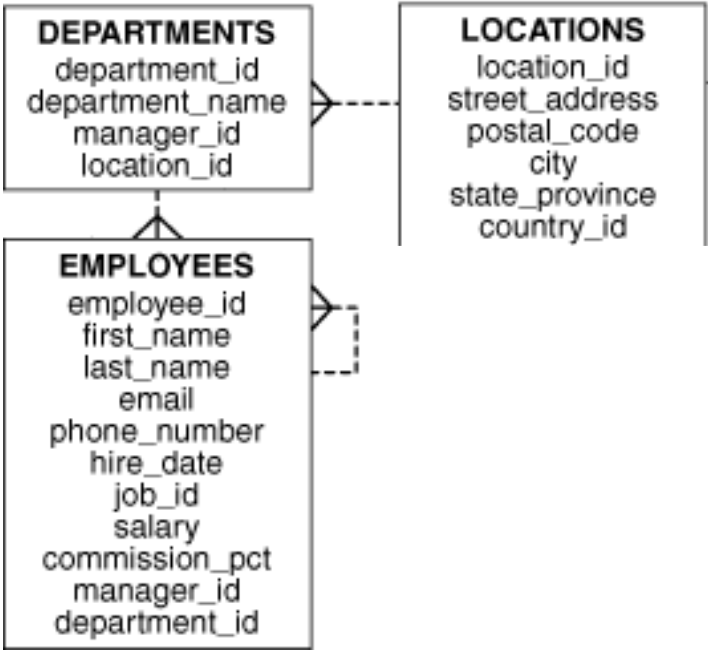
NOTE: To join n tables, you need a minimum of n-1 join conditions

DEPARTMENTS

DEPARTMENT_ID	LOCATION_ID
10	1700
20	1800
20	1800
30	1700
30	1700
30	1700

LOCATIONS

LOCATION_ID	CITY	STATE_PROVINCE
1000	Roma	
1100	Venice	
1200	Tokyo	Tokyo Prefecture
1300	Hiroshima	
1400	Southlake	Texas
1500	South San Francisco	California
1600	South Brunswick	New Jersey
1700	Seattle	Washington
1800	Toronto	Ontario



EXAMPLE: **EQUIJOIN** MORE THAN TWO TABLES

```
SELECT    e. last_name, d. department_name, l.city
FROM      employees e , departments d, locations l
WHERE      e.department_id = d.department_id
AND        d.location_id = l.location_id;
```

LAST_NAME	DEPARTMENT_NAME	CITY
King	Executive	Seattle
Kochhar	Executive	Seattle
De Haan	Executive	Seattle
Hunold	IT	Southlake
Ernst	IT	Southlake
Austin	IT	Southlake
Pataballa	IT	Southlake
Lorentz	IT	Southlake
Greenberg	Finance	Seattle
....		
Baer	Public Relations	Munich
Higgins	Accounting	Seattle
Gietz	Accounting	Seattle

106 rows selected.

EXERCISE # 1- EQUI-JOIN (OLD-JOIN)

จงเขียน SQL Query ที่แสดงชื่อ นามสกุล รหัสแผนก (department_id) และ ชื่อแผนก (department_name) ของพนักงานที่ทำงานในแผนก Shipping



INNER VERSUS OUTER JOINS

- In SQL:1999, the join of two tables returning only matched rows is called an **inner join**. Inner Join consists of :
 - Natural join
 - Join with On clause
 - Join with Using clause
 - Equijoin (Old-style join)
- A join between two tables that returns the results of an inner join as well as the results of unmatched rows is **outer join**.
 - Left outer join
 - Right outer join
 - Full outer join

3. CREATING NATURAL JOINS

- The NATURAL JOIN clause is based on all columns in the two tables that have the same name.
- Natural join will perform the following tasks:
 - ตรวจสอบ common attribute โดยมองหา attribute ที่ชื่อเหมือนกันและมีชนิดข้อมูลเดียวกัน (ปกติคือ foreign key)
 - เลือกแถวจากทั้งสองตารางที่มีค่าเหมือนกันในทุกคอลัมน์ที่เป็น common attribute
 - เชื่อมตารางโดยเลือกเฉพาะแถวที่มีค่าเหมือนกันของ attribute ที่ชื่อเดียวกัน (common attribute)
 - เงื่อนไขการเชื่อมตารางสำหรับ Natural join คือการทำ equijoin ของทุกคอลัมน์ด้วยชื่อคอลัมน์ที่เหมือนกัน
 - If there are no common attributes, return the relational product of the two tables.
 - If the columns having the same names have different data types, an error is returned.

3. CREATING NATURAL JOINS

- The NATURAL JOIN clause is based on all columns in the two tables that have the same name.
- NATURAL JOIN is structured in such a way that, **columns with the same name of associate tables will appear once only**.
- เงื่อนไขการเชื่อมตารางสำหรับ Natural join คือ**การทำ equijoin ของทุกคอลัมน์ด้วยชื่อคอลัมน์ที่เหมือนกัน**

```
SELECT          * [[column_name]]
FROM            nametable1
NATURAL JOIN  nametable2 ;
```

EXAMPLE : NATURAL JOIN

Query: Find all customers with their agent code and agent phone.

```
SELECT *  
FROM CUSTOMER  
NATURAL JOIN AGENT ;
```

Table name: CUSTOMER

	CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE
►	1132445	Walker	32145	231
	1217782	Adares	32145	125
	1312243	Rakowski	34129	167
	1321242	Rodriguez	37134	125
	1542311	Smithson	37134	421
	1657399	Vanloo	32145	231

Table name: AGENT

	AGENT_CODE	AGENT_PHONE
►	125	6152439887
	167	6153426778
	231	6152431124
	333	9041234445

EXAMPLE : NATURAL JOIN

Table name: CUSTOMER

	CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE
▶	1132445	vWalker	32145	231
	1217782	Adares	32145	125
	1312243	Rakowski	34129	167
	1321242	Rodriguez	37134	125
	1542311	Smithson	37134	421
	1657399	Vanloo	32145	231

Table name: AGENT

	AGENT_CODE	AGENT_PHONE
▶	125	6152439887
	167	6153426778
	231	6152431124
	333	9041234445

Step 1: Cartesian Product

	CUS_CODE	CUS_LNAME	CUS_ZIP	CUSTOMER.AGENT_CODE	AGENT.AGENT_CODE	AGENT_PHONE
▶	1132445	vWalker	32145	231	125	6152439887
	1132445	vWalker	32145	231	167	6153426778
	1132445	vWalker	32145	231	231	6152431124
	1132445	vWalker	32145	231	333	9041234445
	1217782	Adares	32145	125	125	6152439887
	1217782	Adares	32145	125	167	6153426778
	1217782	Adares	32145	125	231	6152431124
	1217782	Adares	32145	125	333	9041234445
	1312243	Rakowski	34129	167	125	6152439887
	1312243	Rakowski	34129	167	167	6153426778
	1312243	Rakowski	34129	167	231	6152431124
	1312243	Rakowski	34129	167	333	9041234445
	1321242	Rodriguez	37134	125	125	6152439887
	1321242	Rodriguez	37134	125	167	6153426778
	1321242	Rodriguez	37134	125	231	6152431124
	1321242	Rodriguez	37134	125	333	9041234445
	1542311	Smithson	37134	421	125	6152439887
	1542311	Smithson	37134	421	167	6153426778
	1542311	Smithson	37134	421	231	6152431124
	1542311	Smithson	37134	421	333	9041234445
	1657399	Vanloo	32145	231	125	6152439887
	1657399	Vanloo	32145	231	167	6153426778
	1657399	Vanloo	32145	231	231	6152431124
	1657399	Vanloo	32145	231	333	9041234445

EXAMPLE : NATURAL JOIN

Step 2 : Select Join attribute **AGENT_CODE** values are equal

	CUS_CODE	CUS_LNAME	CUS_ZIP	CUSTOMER.AGENT_CODE	AGENT.AGENT_CODE	AGENT_PHONE
▶	1132445	vWalker	32145	231	125	6152439887
	1132445	vWalker	32145	231	167	6153426778
	1132445	vWalker	32145	231	231	6152431124
	1132445	vWalker	32145	231	333	9041234445
	1217782	Adares	32145	125	125	6152439887
	1217782	Adares	32145	125	167	6153426778
	1217782	Adares	32145	125	231	6152431124
	1217782	Adares	32145	125	333	9041234445
	1312243	Rakowski	34129	167	125	6152439887
	1312243	Rakowski	34129	167	167	6153426778
	1312243	Rakowski	34129	167	231	6152431124
	1312243	Rakowski	34129	167	333	9041234445
	1321242	Rodriguez	37134	125	125	6152439887
	1321242	Rodriguez	37134	125	167	6153426778
	1321242	Rodriguez	37134	125	231	6152431124
	1321242	Rodriguez	37134	125	333	9041234445
	1542311	Smithson	37134	421	125	6152439887
	1542311	Smithson	37134	421	167	6153426778
	1542311	Smithson	37134	421	231	6152431124
	1542311	Smithson	37134	421	333	9041234445
	1657399	Vanloo	32145	231	125	6152439887
	1657399	Vanloo	32145	231	167	6153426778
	1657399	Vanloo	32145	231	231	6152431124
	1657399	Vanloo	32145	231	333	9041234445

	CUS_CODE	CUS_LNAME	CUS_ZIP	CUSTOMER.AGENT_CODE	AGENT.AGENT_CODE	AGENT_PHONE
▶	1217782	Adares	32145	125	125	6152439887
	1321242	Rodriguez	37134	125	125	6152439887
	1312243	Rakowski	34129	167	167	6153426778
	1132445	vWalker	32145	231	231	6152431124
	1657399	Vanloo	32145	231	231	6152431124

Example : Natural Join

Step 3 : use a Projection to eliminate the duplicate attributes

	CUS_CODE	CUS_LNAME	CUS_ZIP	CUSTOMER.AGENT_CODE	AGENT.AGENT_CODE	AGENT_PHONE
►	1217782	Adares	32145	125	125	6152439887
	1321242	Rodriguez	37134	125	125	6152439887
	1312243	Rakowski	34129	167	167	6153426778
	1132445	Walker	32145	231	231	6152431124
	1657399	Vanloo	32145	231	231	6152431124

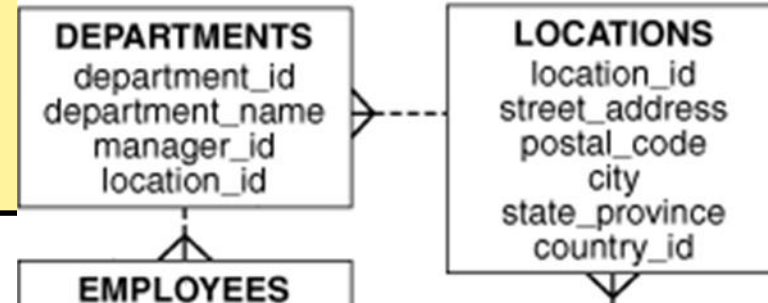


	CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE	AGENT_PHONE
►	1217782	Adares	32145	125	6152439887
	1321242	Rodriguez	37134	125	6152439887
	1312243	Rakowski	34129	167	6153426778
	1132445	Walker	32145	231	6152431124
	1657399	Vanloo	32145	231	6152431124



3. RETRIEVING RECORDS WITH NATURAL JOINS (2 TABLES)

```
SELECT department_id, department_name, location_id, city
FROM      departments
NATURAL JOIN locations ;
```



DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID	CITY
60	IT	1400	Southlake
50	Shipping	1500	South San Francisco
10	Administration	1700	Seattle
30	Purchasing	1700	Seattle
90	Executive	1700	Seattle
100	Finance	1700	Seattle
110	Accounting	1700	Seattle
120	Treasury	1700	Seattle
130	Corporate Tax	1700	Seattle
140	Control And Credit	1700	Seattle
150	Shareholder Services	1700	Seattle
160	Benefits	1700	Seattle
170	Manufacturing	1700	Seattle
180	Construction	1700	Seattle
190	Contracting	1700	Seattle
200	Operations	1700	Seattle
210	IT Support	1700	Seattle
220	NOC	1700	Seattle
230	IT Helpdesk	1700	Seattle
240	Government Sales	1700	Seattle
250	Retail Sales	1700	Seattle
260	Recruiting	1700	Seattle
270	Payroll	1700	Seattle
20	Marketing	1800	Toronto
40	Human Resources	2400	London
80	Sales	2500	Oxford
70	Public Relations	2700	Munich

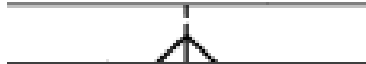
27 rows selected.

Note: number of rows equal to number of rows from the common attribute (location_id) between DEPARTMENTS and LOCATIONS.

3. NATURAL JOINS (2 TABLES) WITH 2 COMMON ATTRIBUTES

DEPARTMENTS

department_id
department_name
manager_id
location_id



EMPLOYEES

employee_id
first_name
last_name
email
phone_number
hire_date
job_id
salary
commission_pct
manager_id
department_id

SELECT employee_id, last_name, department_id,
department_id, location_id

FROM

employees

NATURAL JOIN

departments ;

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID	LOCATION_ID
101	Kochhar	90	90	1700
102	De Haan	90	90	1700
104	Ernst	60	60	1400
105	Austin	60	60	1400
106	Pataballa	60	60	1400
107	Lorentz	60	60	1400
109	Faviet	100	100	1700
110	Chen	100	100	1700
111	Sciarra	100	100	1700
112	Uman	100	100	1700
113	Popp	100	100	1700
115	Khoo	30	30	1700
116	Baida	30	30	1700
117	Tobias	30	30	1700
118	Himuro	30	30	1700
119	Colmenares	30	30	1700
129	Blissot	50	50	1500
130	Atkinson	50	50	1500
131	Marlow	50	50	1500
132	Olson	50	50	1500
150	Tucker	80	80	2500
151	Bernstein	80	80	2500
152	Hall	80	80	2500
153	Olsen	80	80	2500
154	Cambrault	80	80	2500
155	Tuvault	80	80	2500
184	Sarchand	50	50	1500
185	Bull	50	50	1500
186	Dellinger	50	50	1500
187	Cabrio	50	50	1500
202	Fay	20	20	1800
206	Gietz	110	110	1700

32 rows selected.

Note: number of rows equal to number of rows from EMPLOYEES and DEPARTMENTS that have equal values in all matched columns (department_id, manager_id).

JOINING MORE THAN TWO TABLES

EMPLOYEES

LAST_NAME	DEPARTMENT_ID
Whalen	10
Hartstein	20
Fay	20
Raphaely	30
Khoo	30
Baida	30

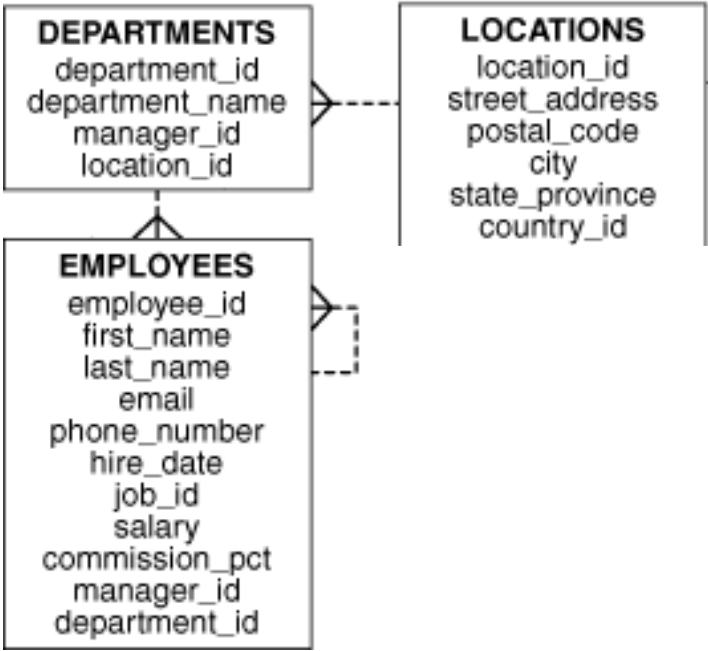
NOTE: To join n tables, you need a minimum of n-1 join conditions

DEPARTMENTS

DEPARTMENT_ID	LOCATION_ID
10	1700
20	1800
20	1800
30	1700
30	1700
30	1700

LOCATIONS

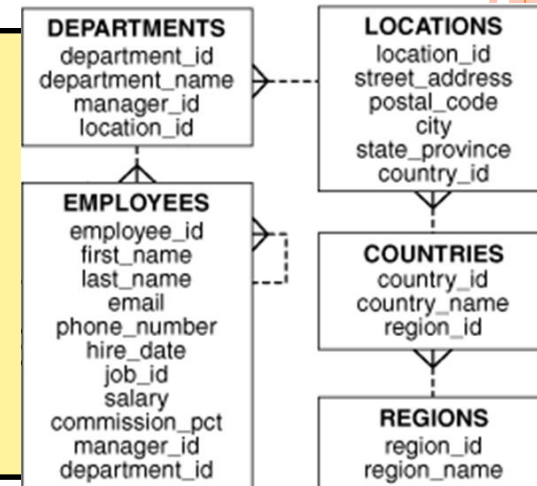
LOCATION_ID	CITY	STATE_PROVINCE
1000	Roma	
1100	Venice	
1200	Tokyo	Tokyo Prefecture
1300	Hiroshima	
1400	Southlake	Texas
1500	South San Francisco	California
1600	South Brunswick	New Jersey
1700	Seattle	Washington
1800	Toronto	Ontario



3. RETRIEVING RECORDS WITH NATURAL JOINS (3 TABLES)

```
SELECT department_id, department_name, location_id,
       city, country_name
```

```
FROM      departments
NATURAL JOIN locations
NATURAL JOIN countries ;
```



DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID	CITY	COUNTRY_NAME
60	IT	1400	Southlake	United States of America
50	Shipping	1500	South San Francisco	United States of America
10	Administration	1700	Seattle	United States of America
30	Purchasing	1700	Seattle	United States of America
90	Executive	1700	Seattle	United States of America
100	Finance	1700	Seattle	United States of America
110	Accounting	1700	Seattle	United States of America
120	Treasury	1700	Seattle	United States of America
130	Corporate Tax	1700	Seattle	United States of America
140	Control And Credit	1700	Seattle	United States of America
150	Shareholder Services	1700	Seattle	United States of America
160	Benefits	1700	Seattle	United States of America
170	Manufacturing	1700	Seattle	United States of America
180	Construction	1700	Seattle	United States of America
190	Contracting	1700	Seattle	United States of America
200	Operations	1700	Seattle	United States of America
210	IT Support	1700	Seattle	United States of America
220	NOC	1700	Seattle	United States of America
230	IT Helpdesk	1700	Seattle	United States of America
240	Government Sales	1700	Seattle	United States of America
250	Retail Sales	1700	Seattle	United States of America
260	Recruiting	1700	Seattle	United States of America
270	Payroll	1700	Seattle	United States of America
20	Marketing	1800	Toronto	Canada
40	Human Resources	2400	London	United Kingdom
80	Sales	2500	Oxford	United Kingdom
70	Public Relations	2700	Munich	Germany

27 rows selected.

Note: number of rows equal to number of rows from the common attribute (location_id) between DEPARTMENTS and LOCATIONS.

3. NATURAL JOINS WITH A WHERE CLAUSE

```
SELECT department_id, department_name, location_id, city
FROM      departments
NATURAL JOIN locations
WHERE     department_id IN (20, 50) ;
```

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID	CITY
20	Marketing	1800	Toronto
50	Shipping	1500	South San Francisco



EXERCISE # 2

1. จงเขียน Query สำหรับแผนกบุคคลเพื่อแสดง location ID, street address, city, state or province, และ ชื่อประเทศ โดยใช้ ตาราง LOCATIONS และ COUNTRIES (ใช้ NATURAL JOIN)

LOCATIONS

location_id
street_address
postal_code
city
state_province
country_id



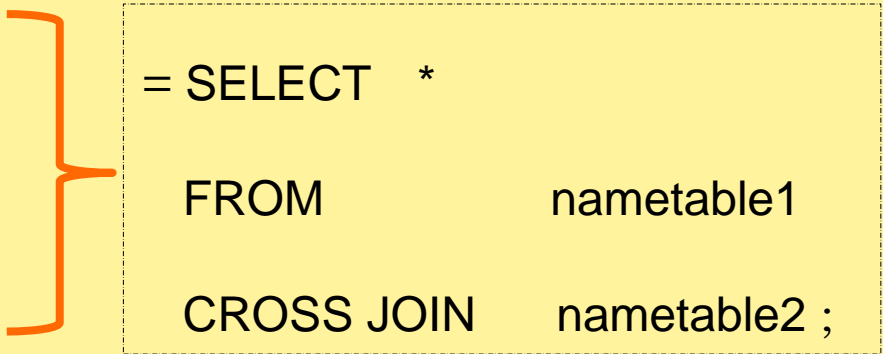
COUNTRIES

country_id
country_name
region_id

4. CREATING JOINS WITH THE ON CLAUSE

- ถ้า ใช้ join โดยไม่มี on คือการทำ catesian product หรือ cross join
- แต่ใช้ ON clause เพื่อระบุเงื่อนไขอื่นหรือระบุคอลัมน์ที่มีชื่อต่างกันเพื่อเชื่อมตาราง
- ON clause จำเป็นต้องมี ชื่อตารางหรือนามแฝงของตาราง สำหรับ common attribute

```
SELECT  *  
FROM    nametable1  
JOIN    nametable2  
ON      (nametable1.column_name = nametable2.column_name) ;
```



= SELECT *
FROM nametable1
CROSS JOIN nametable2 ;

4. RETRIEVING RECORDS WITH **THE ON CLAUSE**

```
SELECT    e. employee_id, e. last_name, e.department_id,  
          d.department_id, d. location_id  
  
FROM      employees e  
  
JOIN      departments d  
  
ON        (e.department_id = d.department_id) ;
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID	LOCATION_ID
200	Whalen	10	10	1700
201	Hartstein	20	20	1800
202	Fay	20	20	1800
114	Raphaely	30	30	1700
115	Khoo	30	30	1700
116	Baida	30	30	1700
119	Colmenares	30	30	1700
118	Himuro	30	30	1700
117	Tobias	30	30	1700
203	Mavris	40	40	2400

4. RETRIEVING RECORDS WITH THE ON CLAUSE

```
SELECT    e.employee_id, e.last_name, e.department_id,
          d.manager_id
FROM      employees e
JOIN      departments d
ON        (e.employee_id < d.manager_id) ;
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	MANAGER_ID
100	King	90	103
100	King	90	108
100	King	90	114
100	King	90	121
100	King	90	145
100	King	90	200
100	King	90	201
100	King	90	203
100	King	90	204
100	King	90	205
101	Kochhar	90	103
101	Kochhar	90	108
101	Kochhar	90	114
101	Kochhar	90	121
101	Kochhar	90	145
101	Kochhar	90	200
101	Kochhar	90	201
101	Kochhar	90	203

APPLYING **ADDITIONAL CONDITIONS TO A JOIN**

- displays only employees who have a manager ID of 149.

```
SELECT      e.employee_id, e.last_name, e.department_id,  
            d.department_id, d.location_id  
  
FROM        employees e  
  
JOIN        departments d  
  
ON          (e.department_id = d.department_id)  
  
AND        e.manager_id = 149 ;
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID	LOCATION_ID
174	Abel	80	80	2500
175	Hutton	80	80	2500
176	Taylor	80	80	2500
179	Johnson	80	80	2500
177	Livingston	80	80	2500

Remark: You can use a WHERE clause instead of AND clause

CREATING THREE-WAY JOINS WITH THE ON CLAUSE

```
SELECT      employee_id, city, department_name
FROM        employees e
JOIN        departments d
ON          (e.department_id = d.department_id)
JOIN        locations l
ON          (d.location_id = l.location_id) ;
```

EMPLOYEE_ID		CITY	DEPARTMENT_NAME
100	Seattle		Executive
101	Seattle		Executive
102	Seattle		Executive
103	Southlake		IT
104	Southlake		IT
105	Southlake		IT
106	Southlake		IT
107	Southlake		IT
108	Seattle		Finance
109	Seattle		Finance
110	Seattle		Finance

EXERCISE # 3 —JOIN ON CLAUSE

จงเขียน Query เพื่อแสดง location ID, street address, city, state province, และ ชื่อประเทศ

โดยที่location นั้นตั้งอยู่ที่ประเทศมีลำดับมาก่อน India และ เรียงลำดับตามชื่อประเทศจาก A-Z

LOCATIONS

location_id
street_address
postal_code
city
state_province
country_id



COUNTRIES

country_id
country_name
region_id

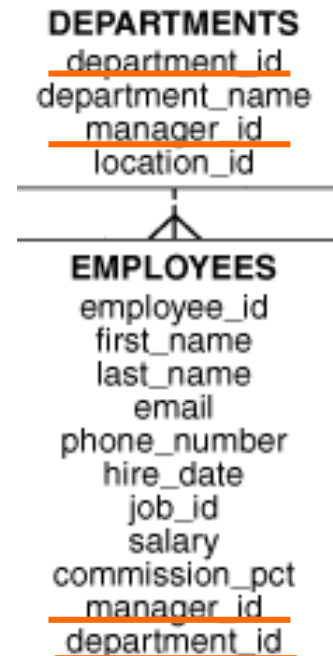
5. CREATING JOINS WITH THE USING CLAUSE

- สำหรับ Natural Join, เชื่อมตารางโดยเลือกเฉพาะแถวที่มีค่าเหมือนกันของ attribute ที่ชื่อเดียวกัน (common attributes)
- ถ้าการเชื่อมตาราง มี common attributes มากกว่า 1 ตัว สามารถใช้ join โดยเพิ่ม **USING clause** เพื่อระบุคอลัมน์ที่ต้องการเชื่อมกันเท่านั้น
- สรุป ใช้ **USING clause** เพื่อจับคู่เพียง 1 คอลัมน์เมื่อมี common attribute มากกว่า 1 ห้ามใช้ นามแฝงของตาราง

```
SELECT *  
FROM   nametable1  
JOIN   nametable2  
USING  (/common attribute) ;
```

5. RETRIEVING RECORDS WITH THE USING CLAUSE

```
SELECT employee_id, last_name,
       location_id, department_id
FROM   employees
JOIN   departments
USING  (department_id) ;
```



EMPLOYEE_ID	LAST_NAME	LOCATION_ID	DEPARTMENT_ID
200	Whalen	1700	10
201	Hartstein	1800	20
202	Fay	1800	20
114	Raphaely	1700	30
115	Khoo	1700	30
116	Baida	1700	30
119	Colmenares	1700	30
118	Himuro	1700	30
117	Tobias	1700	30
203	Mavris	2400	40
120	Weiss	1500	50
121	Fripp	1500	50
123	Vollman	1500	50
132	Olson	1500	50
131	Marlow	1500	50

106 row selected

5. RETRIEVING RECORDS WITH THE USING CLAUSE

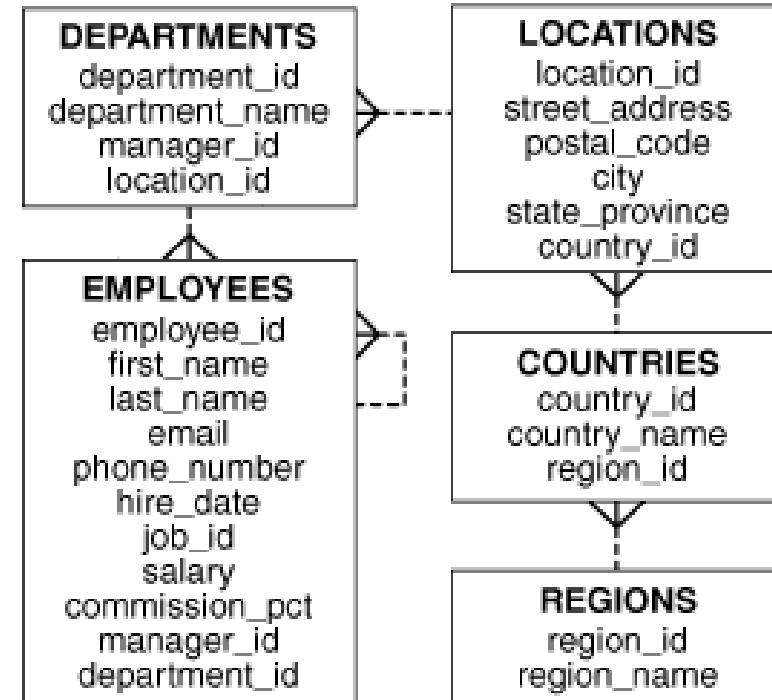
```
SELECT department_id, department_name, location_id, city
FROM   locations
JOIN    departments
USING   (location_id) ;
```

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID	CITY
60	IT	1400	Southlake
50	Shipping	1500	South San Francisco
10	Administration	1700	Seattle
30	Purchasing	1700	Seattle
90	Executive	1700	Seattle
100	Finance	1700	Seattle
110	Accounting	1700	Seattle
120	Treasury	1700	Seattle
130	Corporate Tax	1700	Seattle
140	Control And Credit	1700	Seattle
150	Shareholder Services	1700	Seattle
160	Benefits	1700	Seattle
170	Manufacturing	1700	Seattle
180	Construction	1700	Seattle
190	Contracting	1700	Seattle
200	Operations	1700	Seattle
210	IT Support	1700	Seattle
220	NOC	1700	Seattle
230	IT Helpdesk	1700	Seattle
240	Government Sales	1700	Seattle
250	Retail Sales	1700	Seattle
260	Recruiting	1700	Seattle
270	Payroll	1700	Seattle
20	Marketing	1800	Toronto
40	Human Resources	2400	London
80	Sales	2500	Oxford
70	Public Relations	2700	Munich

27 row selected

EXERCISE # 4 JOIN- USING CLAUSE

จงเขียน SQL Query ที่แสดงชื่อ นามสกุล รหัสงาน
รหัสแผนก และชื่อแผนกของพนักงานทุกคนที่
ทำงานในเมืองโตรอนโต (Toronto) (ใช้ USING
clause)



6. SELF-JOINS USING THE ON CLAUSE

- ต้องการทราบชื่อผู้จัดการของพนักงานแต่ละคน

EMPLOYEES
employee_id
first_name
last_name
email
phone_number
hire_date
job_id
salary
commission_pct
manager_id
department_id

EMPLOYEES

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	MANAGER_ID
100	Steven	King	
101	Neena	Kochhar	100
102	Lex	De Haan	100
103	Alexander	Hunold	102
104	Bruce	Ernst	103
105	David	Austin	103
106	Valli	Pataballa	103
107	Diana	Lorentz	103
108	Nancy	Greenberg	101
109	Daniel	Faviet	108
110	John	Chen	108

EMPLOYEES (MANAGER)

EMPLOYEE_ID	FIRST_NAME	LAST_NAME
100	Steven	King
101	Neena	Kochhar
102	Lex	De Haan
103	Alexander	Hunold
104	Bruce	Ernst
105	David	Austin
106	Valli	Pataballa
107	Diana	Lorentz
108	Nancy	Greenberg
109	Daniel	Faviet
110	John	Chen

```
SELECT  attribute1, attribute2
FROM    employees e
JOIN    employees m
ON      (e.manager_id = m.employee_id) ;
```

MANAGER_ID in the **EMPLOYEES** table is equal to **EMPLOYEES_ID** in the **MANAGER** table.

6. SELF-JOINS USING THE ON CLAUSE

- เพื่อหาชื่อผู้จัดการของพนักงานแต่ละคน จำเป็นต้องทำการเชื่อมตาราง EMPLOYEE กับตัวมันเอง เรียกว่า SELF JOIN

```
SELECT  e.last_name emp, m.last_name mgr
FROM    employees e
JOIN    employees m
ON      (e.manager_id = m.employee_id) ;
```

MANAGER_ID in the **EMPLOYEES** table is equal to **EMPLOYEES_ID** in the **MANAGER** table.

EMP	MGR
Hartstein	King
Zlotkey	King
Cambrault	King
Errazuriz	King
Partners	King
Russell	King
Mourgos	King
Vollman	King
Kaufling	King
Fripp	King
Weiss	King
Raphaely	King
De Haan	King
Kochhar	King
Higgins	Kochhar



EMPLOYEES

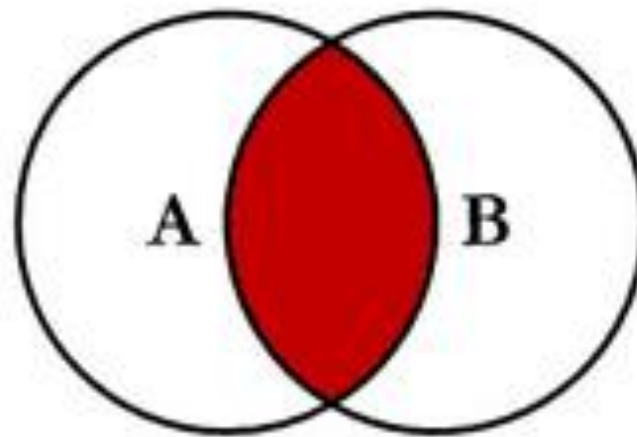
employee_id
first_name
last_name
email
phone_number
hire_date
job_id
salary
commission_pct
manager_id
department_id



EXERCISE # 5 SELF-JOIN

จงแสดงชื่อ นามสกุล วันที่เริ่มทำงานของพนักงานที่เริ่มทำงานก่อน
ผู้จัดการของตนเอง พร้อมกับ ชื่อ นามสกุล วันที่เริ่มทำงานของ
ผู้จัดการของพนักงานคนนั้นๆ (ตั้งชื่อคอลัมน์ Mgr First Name, Mgr
Last Name, Mgr Hired)

SQL JOINS



```
SELECT <select_list>  
FROM TableA A  
INNER JOIN TableB B  
ON A.Key = B.Key
```