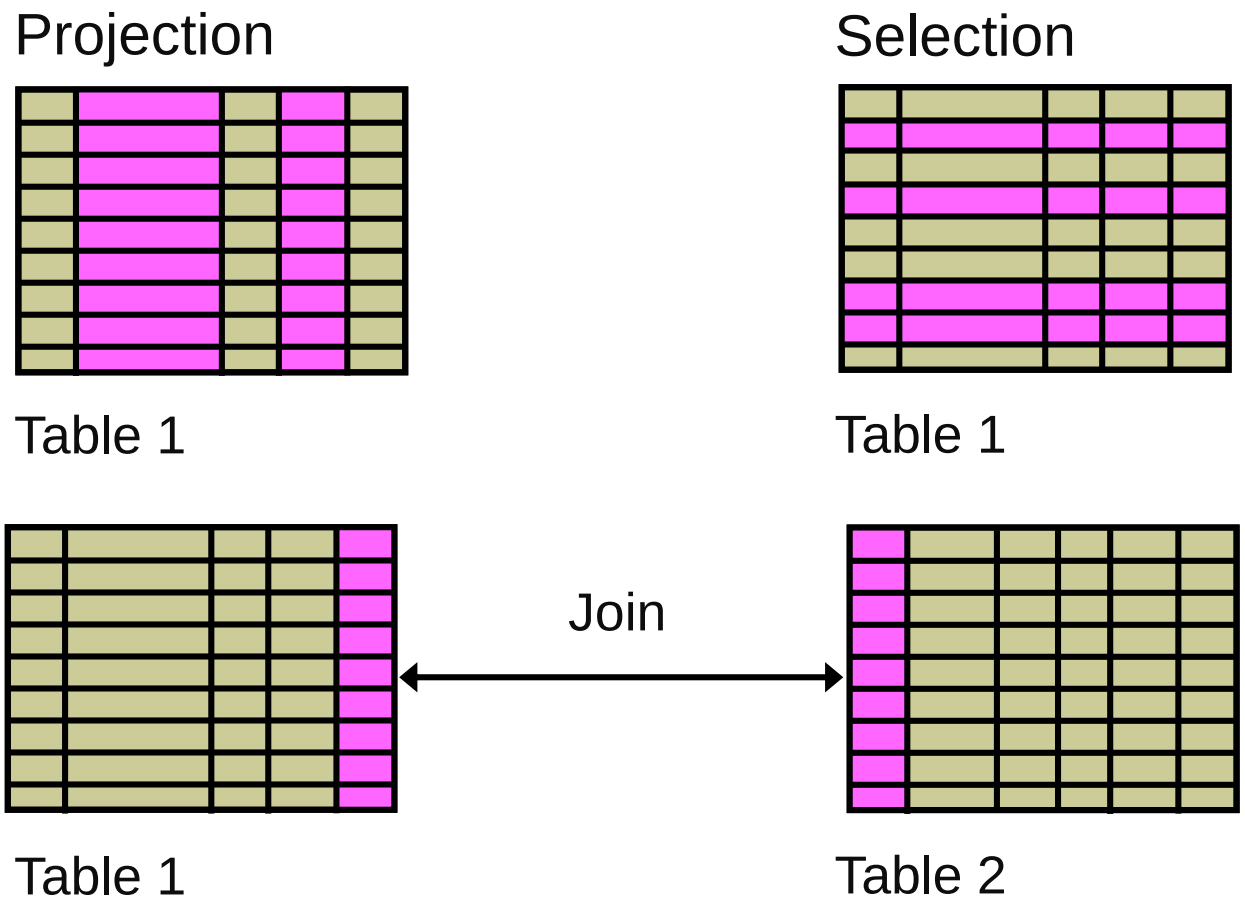
www.itsci.mju.ac.th/sayan

LEC 03: การเรียกดูข้อมูล (SQL)

SAYAN UNANKARD

1/2558

CAPABILITIES OF SQL SELECT STATEMENTS



BASIC SELECT STATEMENT

```
SELECT *|{[DISTINCT] column|expression [alias],...}
FROM    table;
```

- SELECT เลือกคอลัมน์ที่ใช้แสดง
- FROM ระบุตารางที่ประกอบด้วยคอลัมน์นั้น ๆ

SELECTING ALL COLUMNS

```
SELECT *
FROM   departments;
```

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700
8	190	Contracting	(null)	1700

SELECTING SPECIFIC COLUMNS

```
SELECT department_id, location_id
FROM   departments;
```

	DEPARTMENT_ID	LOCATION_ID
1	10	1700
2	20	1800
3	50	1500
4	60	1400
5	80	2500
6	90	1700
7	110	1700
8	190	1700

WRITING SQL STATEMENTS

SQL statements ไม่เป็น case-sensitive

SQL statements สามารถตัดบรรทัดได้มากกว่า 1 บรรทัด

Keywords ไม่สามารถย่อรูป หรือ แยกข้ามบรรทัดได้

ประโยคย่อยจะถูกแยกบรรทัดใหม่

ใช้การย่อหน้าจะทำให้อ่านได้ง่ายขึ้น

ใน SQL Developer คำสั่ง SQL สามารถแยกแต่ละคำสั่งโดยใช้ semicolon (;)

ใน SQL*Plus ต้องใส่เครื่องหมายsemicolon (;) ปิดท้ายแต่ละคำสั่ง

ARITHMETIC EXPRESSIONS

การประมวลผลคำสั่งกับข้อมูลประเภทตัวเลข และ วันที่ โดยใช้เครื่องหมายทางคณิตศาสตร์

Operator	Description
+	Add
-	Subtract
*	Multiply
/	Divide

USING ARITHMETIC OPERATORS

```
SELECT last_name, salary, salary + 300
FROM   employees;
```

	LAST_NAME	SALARY	SALARY+300
1	King	24000	24300
2	Kochhar	17000	17300
3	De Haan	17000	17300
4	Hunold	9000	9300
5	Ernst	6000	6300
6	Lorentz	4200	4500
7	Mourgos	5800	6100
8	Rajs	3500	3800
9	Davies	3100	3400
10	Matos	2600	2900

OPERATOR PRECEDENCE

```
SELECT last_name, salary, 12*salary+100
FROM   employees;
```

1

	LAST_NAME	SALARY	12*SALARY+100
1	King	24000	288100
2	Kochhar	17000	204100
3	De Haan	17000	204100

```
SELECT last_name, salary, 12*(salary+100)
FROM   employees;
```

2

	LAST_NAME	SALARY	12*(SALARY+100)
1	King	24000	289200
2	Kochhar	17000	205200
3	De Haan	17000	205200

DEFINING A NULL VALUE

ค่าว่าง (Null) เป็นค่าที่ไม่พร้อมจะกรอก ไม่กำหนดค่า หรือ ไม่ทราบค่า

ค่าว่างไม่เหมือนกับค่าศูนย์ หรือ ช่องว่าง

```
SELECT last_name, job_id, salary, commission_pct
FROM   employees;
```

	LAST_NAME	JOB_ID	SALARY	COMMISSION_PCT
1	King	AD_PRES	24000	(null)
2	Kochhar	AD_VP	17000	(null)

12	Zlotkey	SA_MAN	10500	0.2
13	Abel	SA_REP	11000	0.3
14	Taylor	SA_REP	8600	0.2

19	Higgins	AC_MGR	12000	(null)
20	Gietz	AC_ACCOUNT	8300	(null)

NULL VALUES IN ARITHMETIC EXPRESSIONS

การคำนวณทางคณิตศาสตร์กับค่าว่าง จะทำให้ผลการคำนวณเป็นค่าว่างไปด้วย

```
SELECT last_name, 12*salary*commission_pct
FROM   employees;
```

	LAST_NAME	12*SALARY*COMMISSION_PCT
1	King	(null)
2	Kochhar	(null)

12	Zlotkey	25200
13	Abel	39600
14	Taylor	20640

19	Higgins	(null)
20	Gietz	(null)

DEFINING A COLUMN ALIAS

การตั้งนามแฝงของคอลัมน์

- เป็นการเปลี่ยนหัวชื่อคอลัมน์
- เหมาะสำหรับการคำนวณ
- ตั้งชื่อต่อท้ายชื่อคอลัมน์ที่ต้องการหรือ คั่นด้วยคำสั่ง AS ระหว่างชื่อคอลัมน์ กับ นามแฝง
- หากชื่อคอลัมน์มีช่องว่างหรืออักษรพิเศษ หรือ ตัวพิมพ์เล็ก/ใหญ่ ให้อยู่ในเครื่องหมาย (double quotation marks) (“ ”)

USING COLUMN ALIASES

```
SELECT last_name AS name, commission_pct comm
FROM   employees;
```

	NAME	COMM
1	King	(null)
2	Kochhar	(null)
3	De Haan	(null)

...

```
SELECT last_name "Name", salary*12 "Annual Salary"
FROM   employees;
```

	Name	Annual Salary
1	King	288000
2	Kochhar	204000
3	De Haan	204000

...

CONCATENATION OPERATOR

คำสั่งในการเชื่อมข้อความ

- เป็นการเชื่อมคอลัมน์หรือข้อความกับคอลัมน์อื่น ๆ
- ใช้เครื่องหมาย vertical bars (||)
- ผลลัพธ์ที่ได้เป็นชนิดข้อความ

```
SELECT    last_name||job_id AS "Employees"
FROM      employees;
```

	Employees
1	AbelSA_REP
2	DaviesST_CLERK
3	De HaanAD_VP
4	ErnstIT_PROG
5	FayMK_REP

LITERAL CHARACTER STRINGS

literal คือข้อมูลที่เป็นตัวอักษร ตัวเลข และ ข้อมูลวันที่ ที่อยู่ในคำสั่ง SELECT

วันที่ และ ตัวอักษรต้องอยู่ในเครื่องหมาย single quotation marks (‘ ’)

USING LITERAL CHARACTER STRINGS

```
SELECT last_name ||' is a '||job_id
       AS "Employee Details"
FROM   employees;
```

	Employee Details
1	Abel is a SA_REP
2	Davies is a ST_CLERK
3	De Haan is a AD_VP
4	Ernst is a IT_PROG
5	Fay is a MK_REP

18	Vargas is a ST_CLERK
19	Whalen is a AD_ASST
20	Zlotkey is a SA_MAN

ALTERNATIVE QUOTE (Q) OPERATOR

หากในข้อมูลมีเครื่องหมาย quotation mark จำเป็นต้องใช้สัญลักษณ์พิเศษโดยใช้ตัวอักษร q’[.....]’

```
SELECT department_name || ' Department' ||
       q'['s Manager Id: ]'
       || manager_id
       AS "Department and Manager"
FROM departments;
```

	Department and Manager
1	Administration Department's Manager Id:200
2	Marketing Department's Manager Id:201
3	Shipping Department's Manager Id:124
4	IT Department's Manager Id:103
5	Sales Department's Manager Id:149
6	Executive Department's Manager Id:100
7	Accounting Department's Manager Id:205
8	Contracting Department's Manager Id:

DUPLICATE ROWS

คำสั่งในการเลือกข้อมูลอาจจะได้ข้อมูลที่ซ้ำกันออกมาหลายๆ แถว ดังนั้นหากไม่ต้องการข้อมูลที่ซ้ำกันหลายแถว ให้ใช้คำสั่ง Distinct

```
SELECT department_id
FROM   employees;
```

1

	DEPARTMENT_ID
1	90
2	90
3	90
4	60
5	60

```
SELECT DISTINCT department_id
FROM   employees;
```

2

	DEPARTMENT_ID
1	(null)
2	90
3	20
4	110

โจทย์

select * from l2order_detail ;

ORDERID	PRODUCTID	QUANTITY
10254	1	300
10254	2	300
10254	3	200
10255	4	240
10255	5	200
10256	3	240

select distinct OrderId, productId from l2order_detail;

ผลลัพธ์ ??

เฉลย

select * from l2order_detail ;

ORDERID	PRODUCTID	QUANTITY
10254	1	300
10254	2	300
10254	3	200
10255	4	240
10255	5	200
10256	3	240

select distinct OrderId, productId from l2order_detail;

ผลลัพธ์ ??

ORDERID	PRODUCTID
10254	1
10254	2
10254	3
10255	4
10255	5
10256	3

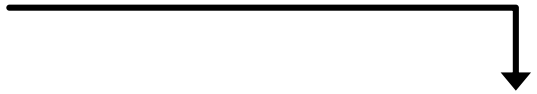
LIMITING ROWS USING A SELECTION

EMPLOYEES

	EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	100	King	AD_PRES	90
2	101	Kochhar	AD_VP	90
3	102	De Haan	AD_VP	90
4	103	Hunold	IT_PROG	60
5	104	Ernst	IT_PROG	60
6	107	Lorentz	IT_PROG	60

...

"retrieve all employees in department 90"



	EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	100	King	AD_PRES	90
2	101	Kochhar	AD_VP	90
3	102	De Haan	AD_VP	90

LIMITING THE ROWS THAT ARE SELECTED

การกำหนดจำนวนแถวที่ต้องการให้คืนค่า โดยใช้คำสั่ง WHERE clause

```
SELECT *|{[DISTINCT] column/expression [alias],...}
FROM   table
[WHERE condition(s)];
```

USING THE WHERE CLAUSE

```
SELECT employee_id, last_name, job_id, department_id
FROM   employees
WHERE  department_id = 90 ;
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	100	King	AD_PRES	90
2	101	Kochhar	AD_VP	90
3	102	De Haan	AD_VP	90

CHARACTER STRINGS AND DATES

วันที่ และ ตัวอักษรต้องอยู่ในเครื่องหมาย single quotation marks (‘ ’)

ตัวอักษรเป็น case-sensitive และ วันที่ต้องถูกรูปแบบ format-sensitive

รูปแบบของวันที่ปกติคือ DD-MON-RR

```
SELECT last_name, job_id, department_id
FROM   employees
WHERE  last_name = 'Whalen' ;
```

```
SELECT last_name
FROM   employees
WHERE  hire_date = '17-FEB-96';
```

COMPARISON OPERATORS

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to
BETWEEN ...AND...	Between two values (inclusive)
IN(set)	Match any of a list of values
LIKE	Match a character pattern
IS NULL	Is a null value

USING COMPARISON OPERATORS

```
SELECT last_name, salary
FROM   employees
WHERE  salary <= 3000 ;
```

	LAST_NAME	SALARY
1	Matos	2600
2	Vargas	2500

RANGE CONDITIONS USING THE BETWEEN OPERATOR

การใช้คำสั่ง BETWEEN สำหรับแสดงข้อมูลที่อยู่ในช่วงที่ต้องการ

```
SELECT last_name, salary
FROM   employees
WHERE  salary BETWEEN 2500 AND 3500 ;
```

Lower limit Upper limit

	LAST_NAME	SALARY
1	Rajs	3500
2	Davies	3100
3	Matos	2600
4	Vargas	2500

MEMBERSHIP CONDITION USING THE IN OPERATOR

การใช้คำสั่ง IN เพื่อกำหนดเงื่อนไขในการเปรียบเทียบในคำสั่ง WHERE

```
SELECT employee_id, last_name, salary, manager_id
FROM   employees
WHERE  manager_id IN (100, 101, 201) ;
```

	EMPLOYEE_ID	LAST_NAME	SALARY	MANAGER_ID
1	101	Kochhar	17000	100
2	102	De Haan	17000	100
3	124	Mourgos	5800	100
4	149	Zlotkey	10500	100
5	201	Hartstein	13000	100
6	200	Whalen	4400	101
7	205	Higgins	12000	101
8	202	Fay	6000	201

PATTERN MATCHING USING THE LIKE OPERATOR

การใช้คำสั่ง LIKE สำหรับค้นหาข้อมูลตัวอักษรที่ตรงเงื่อนไขที่กำหนด

เงื่อนไขในการค้นหา

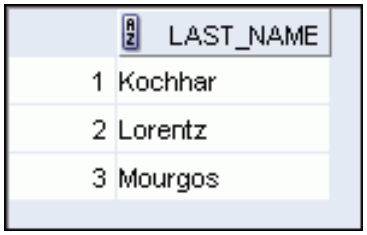
- % แทนตัวอักษรใด (ไม่มีก็ได้ หรือมีมากกว่า 1 ตัว)
- _ แทนตัวอักษรหนึ่งตัว

```
SELECT    first_name
FROM      employees
WHERE     first_name LIKE 'S%';
```

COMBINING WILDCARD CHARACTERS

สามารถใช้สัญลักษณ์ผสมระหว่าง % และ _ รวมกันได้

```
SELECT last_name
FROM   employees
WHERE  last_name LIKE '_o%';
```



	LAST_NAME
1	Kochhar
2	Lorentz
3	Mourgos

You can use the ESCAPE identifier to search for the actual % and _ symbols.

USING THE NULL CONDITIONS

การกำหนดเงื่อนไขค่าว่างในคำสั่ง SQL ด้วยคำสั่ง IS NULL

```
SELECT last_name, manager_id
FROM   employees
WHERE  manager_id IS NULL ;
```

	LAST_NAME	MANAGER_ID
1	King	(null)

DEFINING CONDITIONS USING THE LOGICAL OPERATORS

Operator	Meaning
AND	Returns TRUE if *both* component conditions are true
OR	Returns TRUE if *either* component condition is true
NOT	Returns TRUE if the condition is false

USING THE AND OPERATOR

เงื่อนไขเมื่อใช้คำสั่ง AND ทั้งสองเงื่อนไขจะต้องเป็นจริง

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary >= 10000
AND    job_id LIKE '%MAN%' ;
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	149	Zlotkey	SA_MAN	10500
2	201	Hartstein	MK_MAN	13000

USING THE OR OPERATOR

เงื่อนไขเมื่อใช้คำสั่ง OR เงื่อนไขใดเงื่อนไขหนึ่งจะต้องเป็นจริง

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary >= 10000
OR     job_id LIKE '%MAN%' ;
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	100	King	AD_PRES	24000
2	101	Kochhar	AD_VP	17000
3	102	De Haan	AD_VP	17000
4	124	Mourgos	ST_MAN	5800
5	149	Zlotkey	SA_MAN	10500
6	174	Abel	SA_REP	11000
7	201	Hartstein	MK_MAN	13000
8	205	Higgins	AC_MGR	12000

USING THE NOT OPERATOR

```
SELECT last_name, job_id
FROM   employees
WHERE  job_id
       NOT IN ('IT_PROG', 'ST_CLERK', 'SA_REP') ;
```

	LAST_NAME	JOB_ID
1	De Haan	AD_VP
2	Fay	MK_REP
3	Gietz	AC_ACCOUNT
4	Hartstein	MK_MAN
5	Higgins	AC_MGR
6	King	AD_PRES
7	Kochhar	AD_VP
8	Mourgos	ST_MAN
9	Whalen	AD_ASST
10	Zlotkey	SA_MAN

RULES OF PRECEDENCE

Operator	Meaning
1	Arithmetic operators
2	Concatenation operator
3	Comparison conditions
4	IS [NOT] NULL, LIKE, [NOT] IN
5	[NOT] BETWEEN
6	Not equal to
7	NOT logical condition
8	AND logical condition
9	OR logical condition

สามารถใช้วงเล็บเพื่อกำหนดลำดับการทำงานก่อน-หลังได้

RULES OF PRECEDENCE

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  job_id = 'SA_REP'
OR     job_id = 'AD_PRES'
AND    salary > 15000;
```

1

	LAST_NAME	JOB_ID	SALARY
1	King	AD_PRES	24000
2	Abel	SA_REP	11000
3	Taylor	SA_REP	8600
4	Grant	SA_REP	7000

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  (job_id = 'SA_REP'
OR     job_id = 'AD_PRES')
AND    salary > 15000;
```

2

	LAST_NAME	JOB_ID	SALARY
1	King	AD_PRES	24000

USING THE ORDER BY CLAUSE

การเรียงลำดับแถวข้อมูลโดยใช้คำสั่ง ORDER BY

- ASC: Ascending order เรียงลำดับจากน้อยไปหามาก (ไม่ระบุก็ได้)
- DESC: Descending order เรียงลำดับจากมากไปหาน้อย

```
SELECT   last_name, job_id, department_id, hire_date
FROM     employees
ORDER BY hire_date ;
```

	LAST_NAME	JOB_ID	DEPARTMENT_ID	HIRE_DATE
1	King	AD_PRES	90	17-JUN-87
2	Whalen	AD_ASST	10	17-SEP-87
3	Kochhar	AD_VP	90	21-SEP-89
4	Hunold	IT_PROG	60	03-JAN-90
5	Ernst	IT_PROG	60	21-MAY-91
6	De Haan	AD_VP	90	13-JAN-93

SORTING

ตัวอย่างการเรียงลำดับจากมากไปหาน้อย

```
SELECT   last_name, job_id, department_id, hire_date
FROM     employees
ORDER BY hire_date DESC ;
```

1

การเรียงลำดับจากชื่อนามแฝงของคอลัมน์

```
SELECT employee_id, last_name, salary*12 annsal
FROM   employees
ORDER BY annsal ;
```

2

SORTING

ตัวอย่างการเรียงลำดับโดยใช้ลำดับที่ตำแหน่งคอลัมน์ข้อมูล

```
SELECT   last_name, job_id, department_id, hire_date
FROM     employees
ORDER BY 3;
```

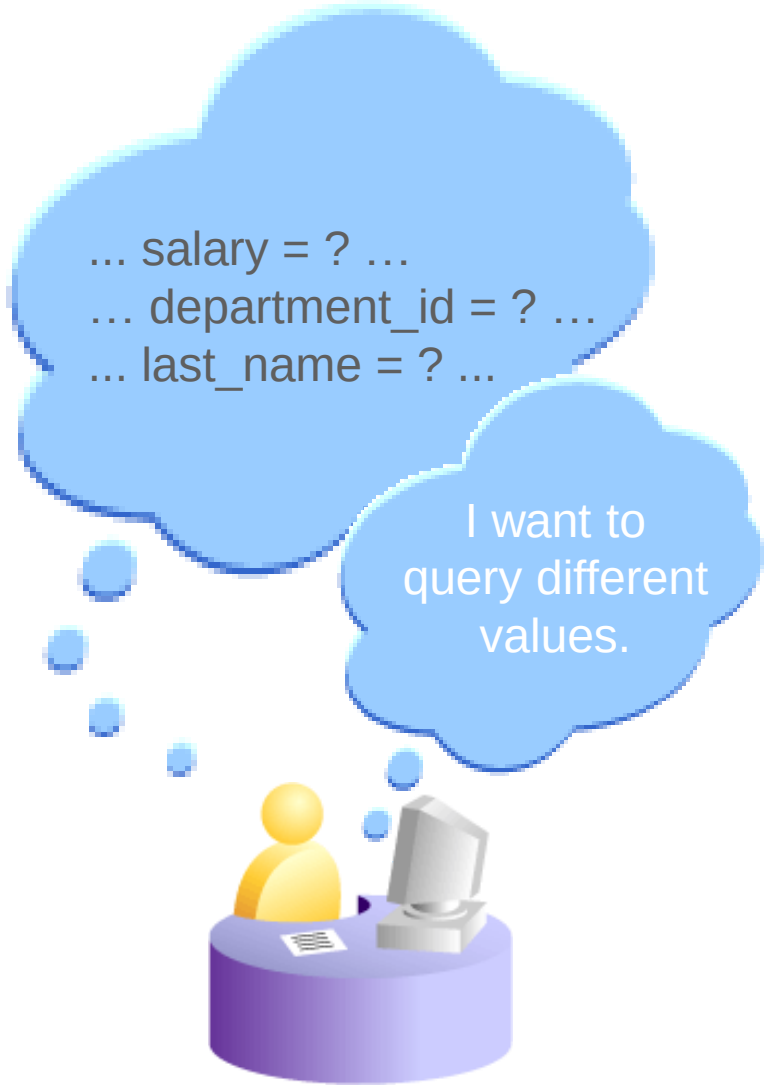
3

ตัวอย่างการเรียงลำดับหลาย ๆ คอลัมน์

```
SELECT last_name, department_id, salary
FROM   employees
ORDER BY department_id, salary DESC;
```

4

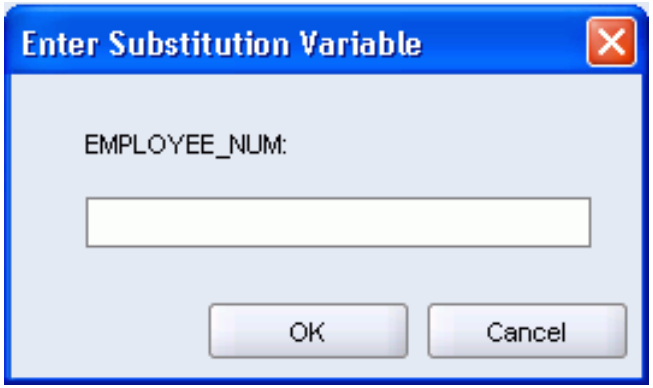
SUBSTITUTION VARIABLES



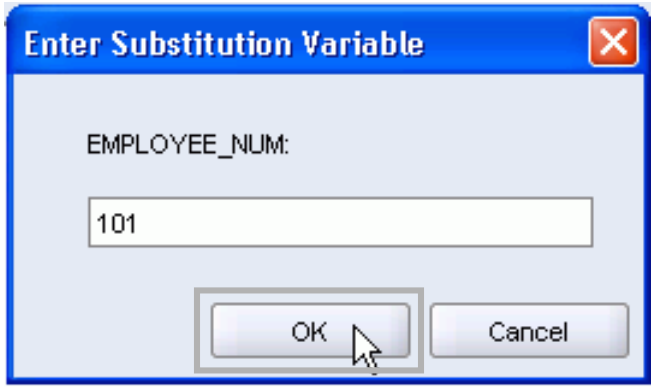
USING THE SINGLE-AMPERSAND SUBSTITUTION VARIABLE

ในโปรแกรม SQL Developer สามารถใช้เครื่องหมาย ampersand (&) เพื่อแสดงกล่องข้อความให้ผู้ใช้ป้อนค่าเงื่อนไขที่ต้องการ

```
SELECT employee_id, last_name, salary, department_id
FROM   employees
WHERE  employee_id = &employee_num;
```



USING THE SINGLE-AMPERSAND SUBSTITUTION VARIABLE

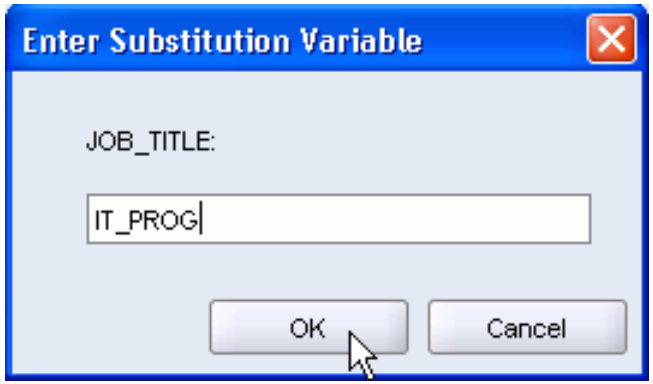


	EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_ID
1	101	Kochhar	17000	90

CHARACTER AND DATE VALUES WITH SUBSTITUTION VARIABLES

การใช้เครื่องหมาย single quotation marks สำหรับข้อมูลที่เป็นวันที่และข้อความ

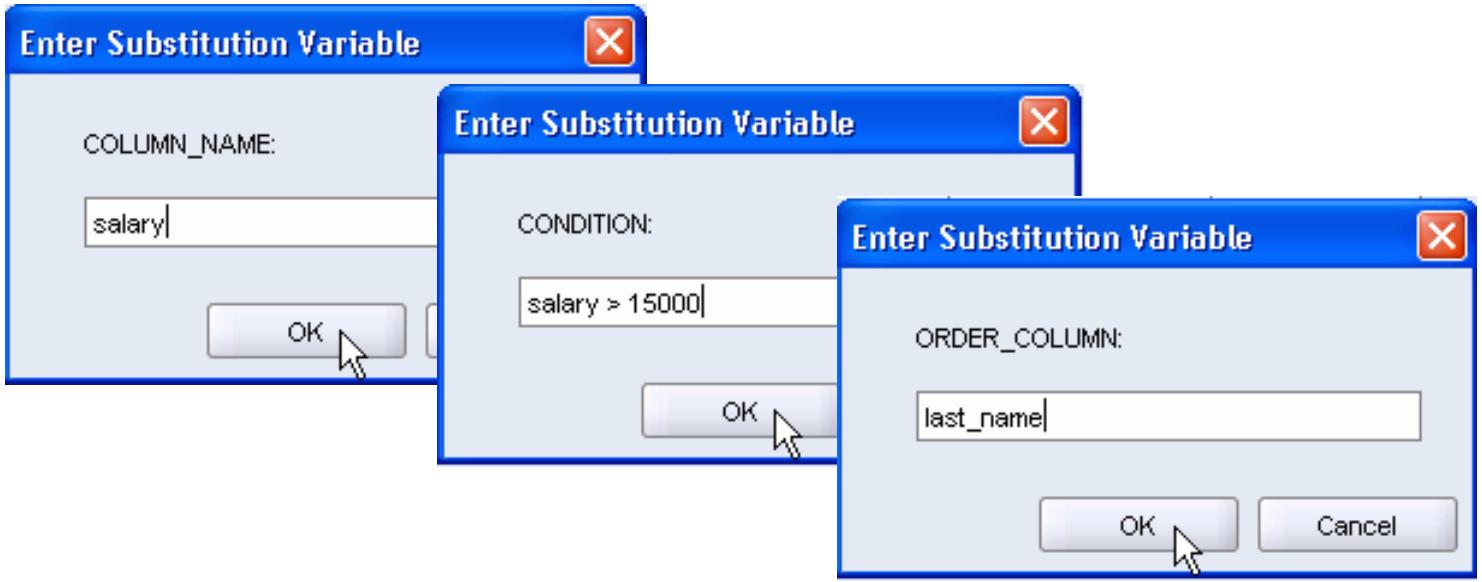
```
SELECT last_name, department_id, salary*12
FROM   employees
WHERE  job_id = '&job_title' ;
```



	LAST_NAME	DEPARTMENT_ID	SALARY*12
1	Hunold	60	108000
2	Ernst	60	72000
3	Lorentz	60	50400

SPECIFYING COLUMN NAMES, EXPRESSIONS, AND TEXT

```
SELECT employee_id, last_name, job_id, &column_name
FROM   employees
WHERE  &condition
ORDER BY &order_column;
```



OBTAINING DATA FROM MULTIPLE TABLES

EMPLOYEES

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	100	King	90
2	101	Kochhar	90
3	102	De Haan	90
18	202	Fay	20
19	205	Higgins	110
20	206	Gietz	110

DEPARTMENTS

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
1	10	Administration	1700
2	20	Marketing	1800
3	50	Shipping	1500
4	60	IT	1400
5	80	Sales	2500
6	90	Executive	1700
7	110	Accounting	1700
8	190	Contracting	1700

	EMPLOYEE_ID	DEPARTMENT_ID	DEPARTMENT_NAME
1	200	10	Administration
2	201	20	Marketing
3	202	20	Marketing
4	124	50	Shipping
5	144	50	Shipping
18	205	110	Accounting
19	206	110	Accounting

TYPES OF JOINS

รูปแบบของการเรียกข้อมูลร่วมกับตารางอื่น ๆ ตามมาตรฐาน SQL:1999 มีดังนี้

- Natural joins:
  1. NATURAL JOIN clause
  2. USING clause
  3. ON clause
- Outer joins:
  1. LEFT OUTER JOIN
  2. RIGHT OUTER JOIN
  3. FULL OUTER JOIN
- Cross joins

QUALIFYING AMBIGUOUS COLUMN NAMES

ใช้คำนำหน้าตาราง เพื่อระบุคอลัมน์ที่ต้องการเลือก กรณีที่มีชื่อคอลัมน์ซ้ำกันหลายตาราง

การใช้**คำนำหน้าตาราง**จะช่วยให้ประสิทธิภาพดีขึ้น

สามารถใช้**นามแฝง**ของตารางเข้าช่วยได้ ทำให้ชื่อตารางสั้นลง และ ทำให้การใช้ทรัพยากรหน่วยความจำน้อยลงด้วย

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	100	King	90
2	101	Kochhar	90
3	102	De Haan	90

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
1	10	Administration	1700
2	20	Marketing	1800
3	50	Shipping	1500
4	60	IT	1400
5	80	Sales	2500
6	90	Executive	1700
7	110	Accounting	1700
8	190	Contracting	1700

Select Employee .Employee_ID,

Department.Department_ID

From Employee Join Department ....

CREATING NATURAL JOINS

NATURAL JOIN clause มีเงื่อนไข**คือชื่อคอลัมน์ของทั้งสองตารางจะต้องตรงกัน**

มันจะเลือกแถวข้อมูลจากทั้งสองตารางที่มีค่าของคอลัมน์นั้น ๆ ตรงกัน

หากคอลัมน์ที่มีชื่อตรงกัน แต่ ชนิดของข้อมูลไม่ตรงกันจะคืนค่าผิดพลาดออกจากคำสั่ง SQL

RETRIEVING RECORDS WITH NATURAL JOINS

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

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID	CITY
1	60	IT	1400	Southlake
2	50	Shipping	1500	South San Francisco
3	10	Administration	1700	Seattle
4	90	Executive	1700	Seattle
5	110	Accounting	1700	Seattle
6	190	Contracting	1700	Seattle
7	20	Marketing	1800	Toronto
8	80	Sales	2500	Oxford

CREATING JOINS WITH THE USING CLAUSE

หากคอลัมน์ที่มีชื่อเดียวกันจากทั้งสองตารางมีชนิดของข้อมูลไม่ตรงกันสามารถใช้คำสั่ง USING เพื่อมาระบุเงื่อนไขได้ การใช้ USING สามารถจับคู่ได้เพียง 1 คอลัมน์เท่านั้น

สามารถใช้งานระหว่าง NATURAL JOIN กับ USING clauses ร่วมกันได้

ตาราง A

Column	Type
AA1	Number
A2	Number

AA1	A2
1	1
2	2

ตาราง B

Column	Type
AB1	Varchar2(10)
A2	Varchar2(10)

AB1	A2
TH	1
SG	2

SELECT * FROM A NATURAL JOIN B; ERROR

SELECT * FROM A JOIN B USING (A2);

JOINING COLUMN NAMES - USING

EMPLOYEES

EMPLOYEE_ID	DEPARTMENT_ID
100	90
101	90
102	90
103	60
104	60
107	60
124	50
141	50
142	50
143	50
144	50
149	80
174	80
176	80

...

DEPARTMENTS

	DEPARTMENT_ID	DEPARTMENT_NAME
1	10	Administration
2	20	Marketing
3	50	Shipping
4	60	IT
5	80	Sales
6	90	Executive
7	110	Accounting
8	190	Contracting

Primary key

Foreign key

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
1	200	Whalen	1700	10
2	201	Hartstein	1800	20
3	202	Fay	1800	20
4	124	Mourgos	1500	50
5	144	Vargas	1500	50
6	143	Matos	1500	50
7	142	Davies	1500	50
8	141	Rajs	1500	50
9	107	Lorentz	1400	60
10	104	Ernst	1400	60

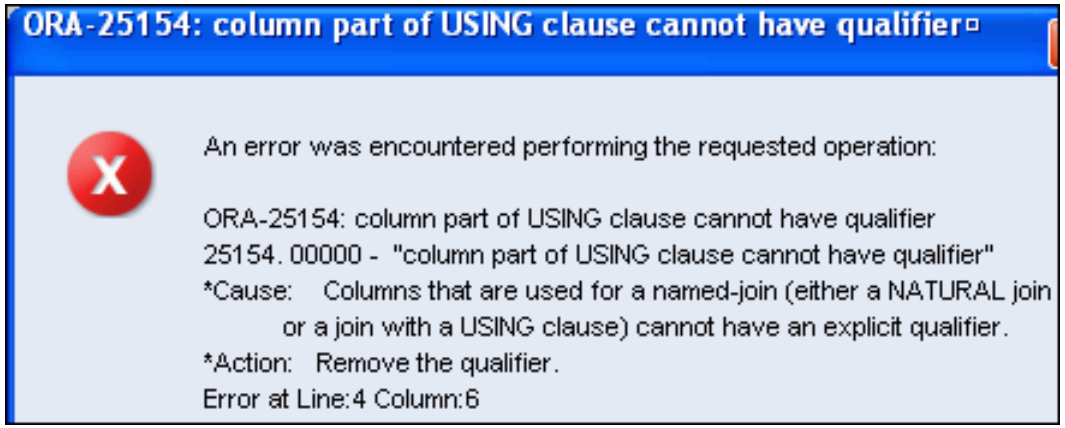
19	205	Higgins	1700	110

USING TABLE ALIASES WITH THE USING CLAUSE

ไม่สามารถใช้คอลัมน์ที่ได้มาจาก การตั้งนามแฝงของตาราง โดยใช้ในคำสั่ง USING

หากคอลัมน์นั้น ๆ ถูกนำไปใช้ในส่วนอื่น ๆ ของคำสั่ง SQL ก็ไม่สามารถใช้นามแฝงของตารางได้เช่นกัน

```
SELECT l.city, d.department_name
FROM   locations l JOIN departments d
USING (location_id)
WHERE d.location_id = 1400;
```



CREATING JOINS WITH THE ON CLAUSE

เงื่อนไขของการเชื่อมตารางแบบ natural join ถูกใช้เมื่อคอลัมน์มีชื่อตรงกันเท่านั้น หากต้องการ **เชื่อมระหว่างคอลัมน์ที่มีชื่อไม่ตรงกัน**จะไม่สามารถทำได้

ดังนั้นเงื่อนไขคำสั่ง ON เข้ามาจัดการคอลัมน์ที่มีชื่อไม่ตรงกันนี้

คำสั่ง ON ทำให้คำสั่ง SQL อ่านง่ายขึ้นและเข้าใจได้ง่ายขึ้นด้วย

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_1	LOCATION_ID
1	200	Whalen	10	10	1700
2	201	Hartstein	20	20	1800
3	202	Fay	20	20	1800
4	124	Mourgos	50	50	1500
5	144	Vargas	50	50	1500
6	143	Matos	50	50	1500
7	142	Davies	50	50	1500
8	141	Rajs	50	50	1500
9	107	Lorentz	60	60	1400
10	104	Ernst	60	60	1400

...

CREATING THREE-WAY JOINS WITH THE ON CLAUSE

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

	EMPLOYEE_ID	CITY	DEPARTMENT_NAME
1	100	Seattle	Executive
2	101	Seattle	Executive
3	102	Seattle	Executive
4	103	Southlake	IT
5	104	Southlake	IT
6	107	Southlake	IT
7	124	South San Francisco	Shipping
8	141	South San Francisco	Shipping

APPLYING ADDITIONAL CONDITIONS TO A JOIN

การใช้คำสั่ง AND หรือ WHERE ในการ**กำหนดเงื่อนไขเพิ่มเติม**

```
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 ;
```

OR

```
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)
WHERE  e.manager_id = 149 ;
```

JOINING A TABLE TO ITSELF - SELF-JOINS

EMPLOYEES (WORKER)

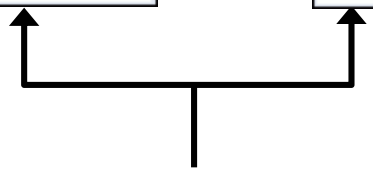
	EMPLOYEE_ID	LAST_NAME	MANAGER_ID
1	100	King	(null)
2	101	Kochhar	100
3	102	De Haan	100
4	103	Hunold	102
5	104	Ernst	103
6	107	Lorentz	103
7	124	Mourgos	100
8	141	Rajs	124
9	142	Davies	124
10	143	Matos	124

...

EMPLOYEES (MANAGER)

EMPLOYEE_ID	LAST_NAME
100	King
101	Kochhar
102	De Haan
103	Hunold
104	Ernst
107	Lorentz
124	Mourgos
141	Rajs
142	Davies
143	Matos

...



MANAGER_ID in the WORKER table is equal to EMPLOYEE_ID in the MANAGER table.

SELF-JOINS USING THE ON CLAUSE

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

	EMP	MGR
1	Hunold	De Haan
2	Fay	Hartstein
3	Gietz	Higgins
4	Lorentz	Hunold
5	Ernst	Hunold
6	Zlotkey	King
7	Mourgos	King
8	Kochhar	King
9	Hartstein	King
10	De Haan	King

NONEQUIJOINS

EMPLOYEES

	LAST_NAME	SALARY
1	King	24000
2	Kochhar	17000
3	De Haan	17000
4	Hunold	9000
5	Ernst	6000
6	Lorentz	4200
7	Mourgos	5800
8	Rajs	3500
9	Davies	3100
10	Matos	2600
...		
19	Higgins	12000
20	Gietz	8300

JOB_GRADES

	GRADE_LEVEL	LOWEST_SAL	HIGHEST_SAL
1	A	1000	2999
2	B	3000	5999
3	C	6000	9999
4	D	10000	14999
5	E	15000	24999
6	F	25000	40000

ตาราง JOB_GRADES เก็บข้อมูลLOWEST_SAL และ HIGHEST_SAL เป็นช่วงของแต่ละระดับ GRADE_LEVEL ดังนั้น สามารถกำหนดระดับเกรดให้กับพนักงานได้

RETRIEVING RECORDS WITH NONEQUIJOINS

```
SELECT e.last_name, e.salary, j.grade_level
FROM   employees e JOIN job_grades j
ON     e.salary
       BETWEEN j.lowest_sal AND j.highest_sal;
```

	LAST_NAME	SALARY	GRADE_LEVEL
1	Vargas	2500	A
2	Matos	2600	A
3	Davies	3100	B
4	Rajs	3500	B
5	Lorentz	4200	B
6	Whalen	4400	B
7	Mourgos	5800	B
8	Ernst	6000	C
9	Fay	6000	C
10	Grant	7000	C

...

RETURNING RECORDS WITH NO DIRECT MATCH WITH OUTER JOINS

DEPARTMENTS

DEPARTMENT_NAME	DEPARTMENT_ID
Administration	10
Marketing	20
Shipping	50
IT	60
Sales	80
Executive	90
Accounting	110
Contracting	190

EMPLOYEES

	DEPARTMENT_ID	LAST_NAME
1	90	King
2	90	Kochhar
3	90	De Haan
4	60	Hunold
5	60	Ernst
6	60	Lorentz
7	50	Mourgos
8	50	Rajs
9	50	Davies
10	50	Matos
...		
19	110	Higgins
20	110	Gietz

There are no employees in department 190.

INNER VERSUS OUTER JOINS

ในคำสั่งมาตรฐาน SQL:1999 คำสั่ง join ที่ทำให้ได้ผลลัพธ์ที่คืนค่าแถวที่ตรงกันเรียกว่า inner join

การ join ระหว่างสองตารางที่คืนค่าผลลัพธ์แถวที่ตรงกัน และ **แถวที่ไม่ตรง**ของตารางทางด้านซ้าย หรือ ขวา เรียกว่า left (or right) outer join

การ join ที่ได้ผลลัพธ์ของทั้งหมด คือ ทั้งแถวที่ตรงกัน และ ไม่ตรงกันทั้งตารางด้านซ้าย และ ขวา เรียกว่า full outer join.

LEFT OUTER JOIN

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e LEFT OUTER JOIN departments d
ON   (e.department_id = d.department_id) ;
```

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Fay	20	Marketing
3	Hartstein	20	Marketing
4	Vargas	50	Shipping
5	Matos	50	Shipping

17	King	90	Executive
18	Gietz	110	Accounting
19	Higgins	110	Accounting
20	Grant	(null)	(null)

RIGHT OUTER JOIN

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e RIGHT OUTER JOIN departments d
ON    (e.department_id = d.department_id) ;
```

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Hartstein	20	Marketing
3	Fay	20	Marketing
4	Higgins	110	Accounting

19	Taylor	80	Sales
20	(null)	190	Contracting

FULL OUTER JOIN

```
SELECT e.last_name, d.department_id, d.department_name
FROM   employees e FULL OUTER JOIN departments d
ON   (e.department_id = d.department_id) ;
```

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Hartstein	20	Marketing
3	Fay	20	Marketing
4	Higgins	110	Accounting

19	Taylor	80	Sales
20	Grant	(null)	(null)
21	(null)	190	Contracting

CARTESIAN PRODUCTS

การเชื่อมตารางโดยไม่สนใจเงื่อนไขการใด ๆ เรียกว่าการคูณ

โดยที่ ทุก ๆ แถวในตารางแรก จะถูกเชื่อมต่อกับทุก ๆ แถวในตารางที่สอง

GENERATING A CARTESIAN PRODUCT

EMPLOYEES (20 rows)

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	100	King	90
2	101	Kochhar	90
3	102	De Haan	90
4	103	Hunold	60
...			
19	205	Higgins	110
20	206	Gietz	110

DEPARTMENTS (8 rows)

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
1	10	Administration	1700
2	20	Marketing	1800
3	50	Shipping	1500
4	60	IT	1400
5	80	Sales	2500
6	90	Executive	1700
7	110	Accounting	1700
8	190	Contracting	1700

Cartesian product:
20 x 8 = 160 rows

	EMPLOYEE_ID	DEPARTMENT_ID	LOCATION_ID
1	100	90	1700
2	101	90	1700
3	102	90	1700
4	103	60	1700
...			
159	205	110	1700
160	206	110	1700

CREATING CROSS JOINS

การ CROSS JOIN คือผลลัพธ์ที่ได้จากการคูณกันระหว่างสองตาราง ซึ่งบางครั้งถูกเรียกว่า Cartesian product เช่นกัน

```
SELECT last_name, department_name
FROM   employees
CROSS JOIN departments ;
```

	LAST_NAME	DEPARTMENT_NAME
1	Abel	Administration
2	Davies	Administration
3	De Haan	Administration
4	Ernst	Administration
5	Fay	Administration

159	Whalen	Contracting
160	Zlotkey	Contracting