2020-2021

5COSC002W DATABASE SYSTEMS
Lecture 07

DATABASE QUERYING
SQL: Retrieving data from multiple tables using JOINs

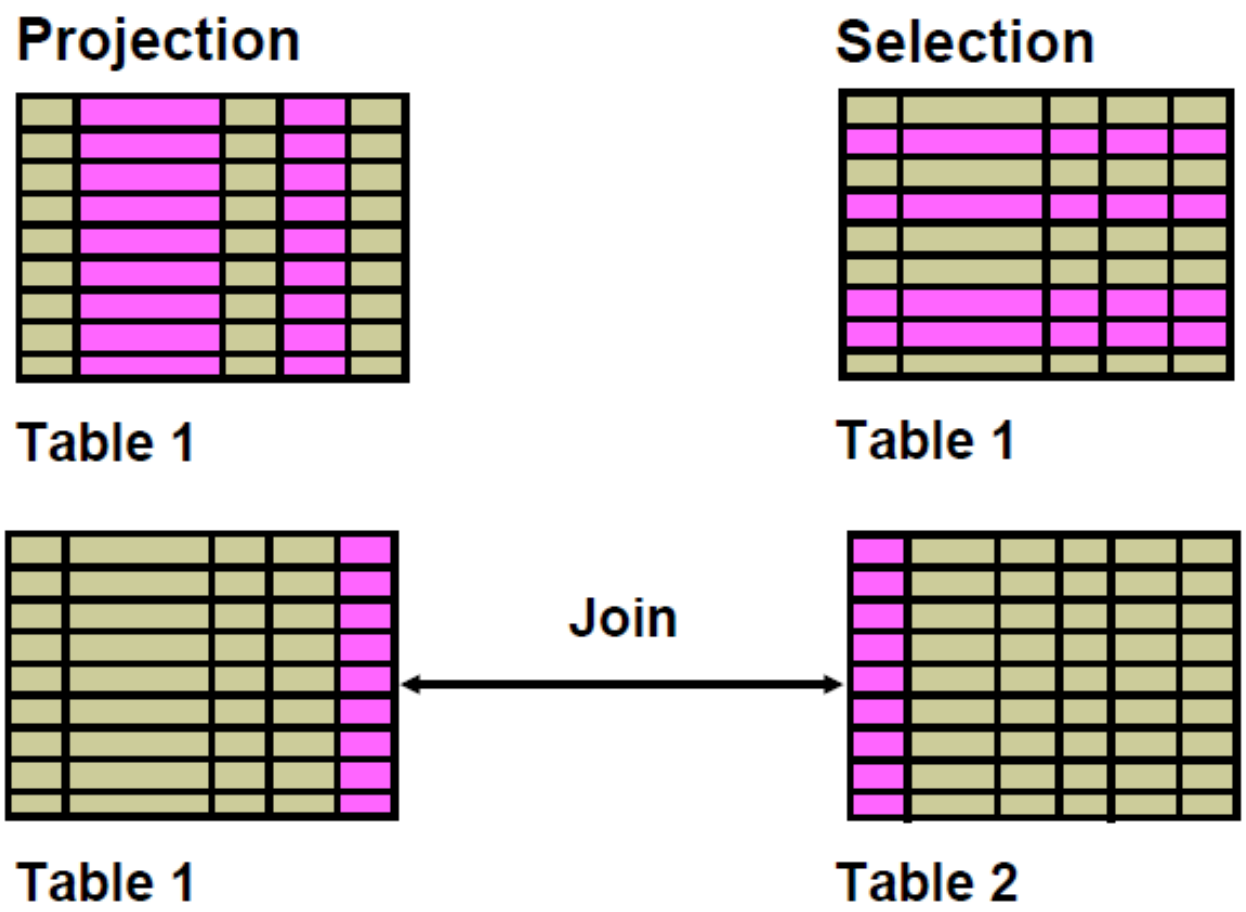
UNIVERSITY OF WESTMINSTER

UNIVERSITY OF TECHNOLOGICAL INNOVATION WESTMINSTER

Lecture 07 – Outline

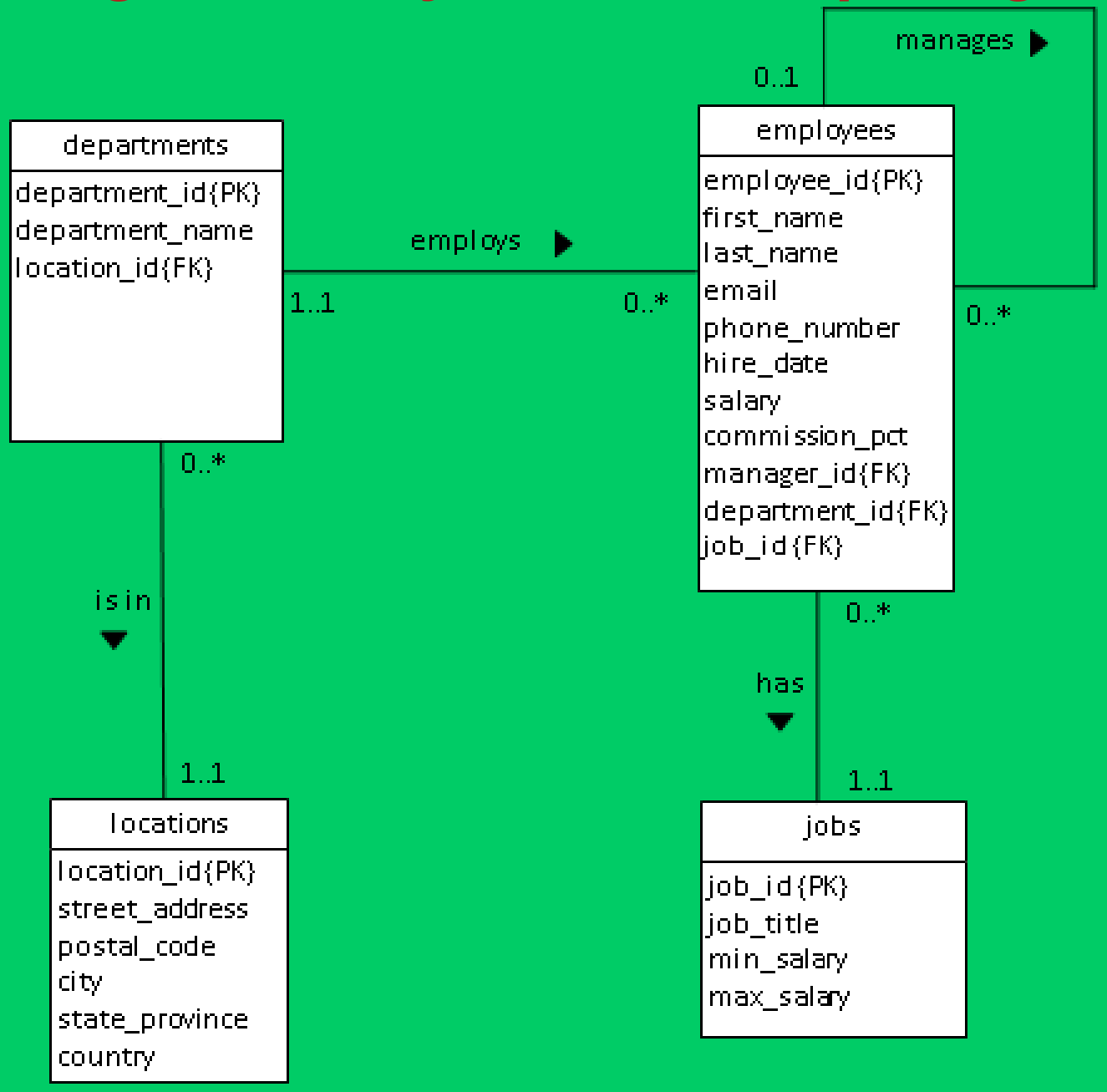
- *Write SELECT statements to access data from more than one table using:*
  - *equijoins*
  - *non-equijoins*
- *Join a table to itself by using:*
  - *self-joins*
- *View data that generally does not meet a join condition by using:*
  - *left outer joins*
  - *right outer joins*
  - *full outer joins*

Capabilities of SQL SELECT Statements



UNIVERSITY OF TECHNOLOGICAL INNOVATION WESTMINSTER

Logical Entity Relationship Diagram



Equijoins

EMPLOYEES

EMPLOYEE_ID	DEPARTMENT_ID
200	10
201	20
202	20
124	50
141	50
142	50
143	50
144	50
103	60
104	60
107	60
149	80
174	80
176	80

...

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME
10	Administration
20	Marketing
20	Marketing
50	Shipping
50	Shipping
50	Shipping
50	Shipping
50	Shipping
60	IT
60	IT
60	IT
80	Sales
80	Sales
80	Sales

...

Foreign key

Primary key

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);
```

Child.FK= Parent.PK

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
124	Mourgos	50	50	1500
141	Rajs	50	50	1500
142	Davies	50	50	1500
143	Matos	50	50	1500

...

19 rows selected.

UNIVERSITY OF TECHNOLOGICAL INNOVATION WESTMINSTER

Retrieving records from multiple tables

– **2 syntaxes are acceptable for MySQL**

```
SELECT department_id, department_name,
street_address, postal_code, city
FROM departments JOIN locations
ON locations.location_id = departments.location_id;
```

Parent.PK= Child.FK

```
SELECT department_id, department_name,
street_address, postal_code, city
FROM departments, locations
WHERE locations.location_id = departments.location_id;
```

Parent.PK= Child.FK

Applying Additional Conditions to a Join

```
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 ;
```

Child.FK= Parent.PK

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID	LOCATION_ID
174	Abel	80	80	2500
176	Taylor	80	80	2500

UNIVERSITY OF TECHNOLOGICAL INNOVATION WESTMINSTER

Additional condition with a join

– **2 syntaxes are acceptable for MySQL**

```
SELECT department_id, department_name,
street_address, postal_code, city
FROM departments JOIN locations
ON locations.location_id = departments.location_id
AND department_name LIKE 'A%';
```

Parent.PK= Child.FK

```
SELECT department_id, department_name,
street_address, postal_code, city
FROM departments, locations
WHERE locations.location_id = departments.location_id
AND department_name LIKE 'A%';
```

Parent.PK= Child.FK

Self-Joins Using the ON Clause

EMPLOYEES (WORKER)

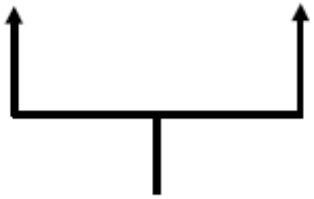
EMPLOYEE_ID	LAST_NAME	MANAGER_ID
100	King	
101	Kochhar	100
102	De Haan	100
103	Hunold	102
104	Ernst	103
107	Lorentz	103
124	Mourgos	100

...

EMPLOYEES (MANAGER)

EMPLOYEE_ID	LAST_NAME
100	King
101	Kochhar
102	De Haan
103	Hunold
104	Ernst
107	Lorentz
124	Mourgos

...



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

Self-Joins Using the ON Clause

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

Child.FK= Parent.PK

EMP	MGR
Hartstein	King
Zlotkey	King
Mourgos	King
De Haan	King
Kochhar	King

...

19 rows selected.

UNIVERSITY OF TECHNOLOGICAL INNOVATION WESTMINSTER

Self Join

– **2 syntaxes are acceptable for MySQL**

```
SELECT e.last_name, e.first_name, e.salary,
m.last_name, m.first_name, m.salary
FROM employee e JOIN employee m
ON e.manager_id = m.employee_id
```

Child.FK= Parent.PK

```
SELECT e.last_name, e.first_name, e.salary,
m.last_name, m.first_name, m.salary
FROM employee e, employee m
WHERE e.manager_id = m.employee_id;
```

Child.FK= Parent.PK

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;
```

Parent.PK= Child.FK

Parent.PK= Child.FK

EMPLOYEE_ID	CITY	DEPARTMENT_NAME
103	Southlake	IT
104	Southlake	IT
107	Southlake	IT
124	South San Francisco	Shipping
141	South San Francisco	Shipping
142	South San Francisco	Shipping
143	South San Francisco	Shipping
144	South San Francisco	Shipping

...

19 rows selected.

UNIVERSITY OF TECHNOLOGICAL INNOVATION WESTMINSTER

Three-way joins

– **Syntax using ON clause**

```
SELECT department_name, last_name, salary, job_title
FROM departments
JOIN employees
ON departments.department_id = employees.department_id
JOIN jobs
ON jobs.job_id = employees.job_id
```

Parent.PK= Child.FK

Parent.PK= Child.FK

UNIVERSITY OF TECHNOLOGICAL INNOVATION WESTMINSTER

Three-way joins

– **Syntax using WHERE clause**

```
SELECT department_name, last_name, salary, job_title
FROM departments, employees, jobs
WHERE
departments.department_id = employees.department_id
AND
jobs.job_id = employees.job_id
```

Parent.PK= Child.FK

Parent.PK= Child.FK

Non-Equijoins

EMPLOYEES

LAST_NAME	SALARY
King	24000
Kochhar	17000
De Haan	17000
Hunold	9000
Ernst	6000
Lorentz	4200
Mourgos	5800
Rajs	3500
Davies	3100
Matos	2600
Vargas	2500
Zlotkey	10500
Abel	11000
Taylor	8600

...

20 rows selected.

JOB_GRADES

GRA	LOWEST_SAL	HIGHEST_SAL
A	1000	2999
B	3000	5999
C	6000	9999
D	10000	14999
E	15000	24999
F	25000	40000

Salary in the EMPLOYEES table must be between lowest salary and highest salary in the JOB_GRADES table.

Retrieving Records with Non-Equijoins

```
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	GRA
Matos	2600	A
Vargas	2500	A
Lorentz	4200	B
Mourgos	5800	B
Rajs	3500	B
Davies	3100	B
Whalen	4400	B
Hunold	9000	C
Ernst	6000	C

...

20 rows selected.

Outer Joins

DEPARTMENTS

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

8 rows selected.

EMPLOYEES

DEPARTMENT_ID	LAST_NAME
90	King
90	Kochhar
90	De Haan
60	Hunold
60	Ernst
60	Lorentz
50	Mourgos
50	Rajs
50	Davies
50	Matos
50	Vargas
80	Zlotkey

...

20 rows selected.

There are no employees in department 190.

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) ;
```

Child.FK= Parent.PK

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Fay	20	Marketing
Hartstein	20	Marketing
...		
De Haan	90	Executive
Kochhar	90	Executive
King	90	Executive
Gietz	110	Accounting
Higgins	110	Accounting
Grant		

20 rows selected.

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) ;
```

Child.FK= Parent.PK

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Fay	20	Marketing
Hartstein	20	Marketing
Davies	50	Shipping
...		
Kochhar	90	Executive
Gietz	110	Accounting
Higgins	110	Accounting
	190	Contracting

20 rows selected.

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) ;
```

Child.FK= Parent.PK

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Fay	20	Marketing
Hartstein	20	Marketing
...		
King	90	Executive
Gietz	110	Accounting
Higgins	110	Accounting
Grant		
	190	Contracting

21 rows selected.