Chapter 3B.

Structured Query Language (SQL II)

CSIS0278 / COMP3278

Introduction to

Database Management Systems

Department of Computer Science, The University of Hong Kong

Slides prepared by - **Dr. Chui Chun Kit**, http://www.cs.hku.hk/~ckchui/ For other uses, please email : ckchui@cs.hku.hk

Acknowledgement – Professor Nikos Mamoulis http://www.cs.hku.hk/~nikos/

Content

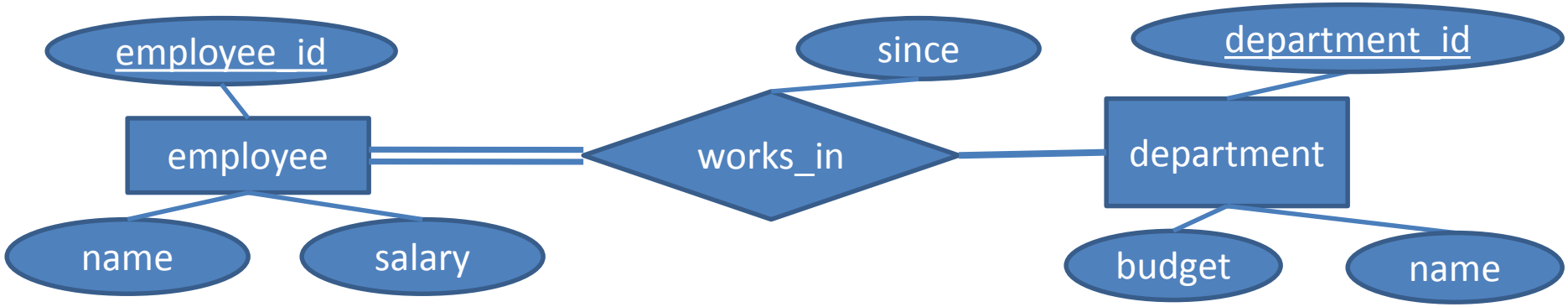
- Revision
- Set operations
- More on nested queries
- Null values
- Views
- Authorization
- Assertion
- Other SQL constructs

In this chapter…

- Outcome 1. **Information Modeling**
  - Able to understand the modeling of real life information in a database system.
- Outcome 2. **Query Languages**
  - Able to understand and use the languages designed for data access.
- Outcome 3. **System Design**
  - Able to understand the design of an efficient and reliable database system.
- Outcome 4. **Application Development**
  - Able to implement a practical application on a real database.

Revision

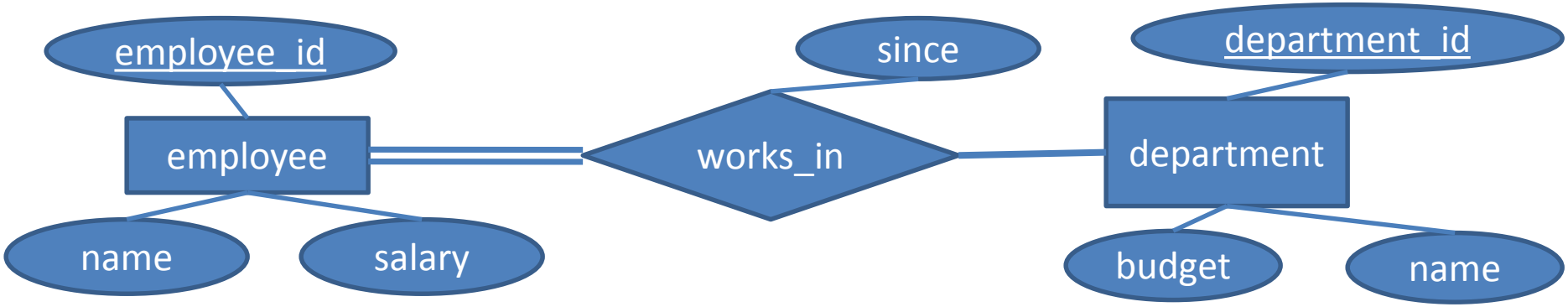
- **Step 1.** **Information modeling (Chapter 2A).**



Before we proceed to learning more SQL constructs, lets have a revision on what we have learned up to this chapter.

Revision

- **Step 1.** **Information modeling (Chapter 2A).**



- **Step 2.** **Reduce to database tables (Chapter 2B).**
  - **Employees** (employee_id, name, salary)
    Foreign key : none.
  - **Departments** (department_id, name, budget)
    Foreign key : none.
  - **Works_in**(employee_id, department_id, since)
    Foreign key : employee_id **REFERENCES** Employee (employee_id).
    department_id **REFERENCES** Department (department_id).

Revision

- **Step 3.** Create the database (Chapter 3A).

```
CREATE TABLE Employees (
    employee_id INT(12),
    name VARCHAR(30) NOT NULL,
    salary INT UNSIGNED NOT NULL,
    PRIMARY KEY(employee_id)
)ENGINE = INNODB;
```

INNODB storage engine, just for MySQL to support foreign key constraints.

```
CREATE TABLE Departments (
    department_id INT(12),
    name VARCHAR(30) NOT NULL,
    budget INT UNSIGNED NOT NULL,
    PRIMARY KEY(department_id)
) ENGINE = INNODB;
```

```
CREATE TABLE Works_in(
    employee_id INT(12),
    department_id INT(12),
    since DATE NOT NULL,
    PRIMARY KEY(employee_id, department_id),
    FOREIGN KEY (employee_id) REFERENCES Employees (employee_id),
    FOREIGN KEY (department_id) REFERENCES Departments (department_id)
) ENGINE = INNODB;
```

Revision

- **Step 3.** **Create the database (Chapter 3A).**

```
INSERT INTO Employees VALUES ( 1, 'Jones', 26000);
INSERT INTO Employees VALUES ( 2, 'Smith', 28000);
INSERT INTO Employees VALUES ( 3, 'Parker', 35000);
INSERT INTO Employees VALUES ( 4, 'Smith', 24000);
```

```
INSERT INTO Departments VALUES ( 1, 'Toys', 122000), ( 2, 'Tools', 239000), ( 3, 'Food', 100000);
```

```
INSERT INTO Works_in VALUES ( 1, 1, '2001-1-1'), ( 2, 1, '2002-4-1'), ( 2, 2, '2005-2-2'), ( 3, 3, '2003-1-1'), ( 4, 3, '2005-1-1');
```

Employees

employee_id	name	salary
1	Jones	26000
2	Smith	28000
3	Parker	35000
4	Smith	24000

Works_in

employee_id	department_id	since
1	1	2001-1-1
2	1	2002-4-1
2	2	2005-2-2
3	3	2003-1-1
4	3	2005-1-1

Departments

department_id	name	budget
1	Toys	122000
2	Tools	239000
3	Food	100000

Revision

- **Step 4. Design the SQL to access data for the application (Chapter 3A).**
  - **Query 1:** Find the names of all employees and remove duplicates.

```
SELECT DISTINCT name
FROM Employees;
```

Where can I find the name of all employees?

Employees

employee_id	name	salary
1	Jones	26000
2	Smith	28000
3	Parker	35000
4	Smith	24000

Works_in

employee_id	department_id	since
1	1	2001-1-1
2	1	2002-4-1
2	2	2005-2-2
3	3	2003-1-1
4	3	2005-1-1

Departments

department_id	name	budget
1	Toys	122000
2	Tools	239000
3	Food	100000

Exercises

- **Query 2:** Find the employee_ids and names of employees who work in department with department_id=2.

```
SELECT
FROM Works_in W
WHERE W.department_id = 2
```

How to find the **employee_id(s)** of the employee in department #2?

Employees

employee_id	name	salary
1	Jones	26000
2	Smith	28000
3	Parker	35000
4	Smith	24000

Works_in

employee_id	department_id	since
1	1	2001-1-1
2	1	2002-4-1
2	2	2005-2-2
3	3	2003-1-1
4	3	2005-1-1

Departments

department_id	name	budget
1	Toys	122000
2	Tools	239000
3	Food	100000

Exercises

- **Query 2:** Find the employee_ids and names of employees who work in department with department_id=2.

```
SELECT E.employee_id, E.name
FROM Works_in W , Employees E
WHERE W.department_id = 2 AND
        E.employee_id = W.employee_id;
```

How to find the **employee_id(s)** of the employee in department #2?

Given an **employee_id**, find the name of the employee.

Employees

employee_id	name	salary
1	Jones	26000
2	Smith	28000
3	Parker	35000
4	Smith	24000

Works_in

employee_id	department_id	since
1	1	2001-1-1
2	1	2002-4-1
2	2	2005-2-2
3	3	2003-1-1
4	3	2005-1-1

Departments

department_id	name	budget
1	Toys	122000
2	Tools	239000
3	Food	100000

Exercises

- **Query 3:** Find the dept. names where employee with employee_id = 2 works.

```
SELECT
FROM  Works_in W
WHERE W.employee_id = 2
```

What are the **dpt. id(s)** of the dpt(s) where employee #2 works?

Employees

employee_id	name	salary
1	Jones	26000
2	Smith	28000
3	Parker	35000
4	Smith	24000

Works_in

employee_id	department_id	since
1	1	2001-1-1
2	1	2002-4-1
2	2	2005-2-2
3	3	2003-1-1
4	3	2005-1-1

Departments

department_id	name	budget
1	Toys	122000
2	Tools	239000
3	Food	100000

Exercises

- **Query 3:** Find the dept. names where employee with employee_id = 2 works.

```
SELECT D.name
FROM  Works_in W, Departments D
WHERE W.employee_id = 2 AND
      D.department_id = W.department_id;
```

What are the **dpt. id(s)** of the dpt(s) where employee #2 works?

Given a **department_id**, find the name of the department.

Employees

employee_id	name	salary
1	Jones	26000
2	Smith	28000
3	Parker	35000
4	Smith	24000

Works_in

employee_id	department_id	since
1	1	2001-1-1
2	1	2002-4-1
2	2	2005-2-2
3	3	2003-1-1
4	3	2005-1-1

Departments

department_id	name	budget
1	Toys	122000
2	Tools	239000
3	Food	100000

Exercises

- **Query 4:** Find the dept. ids where employees named Smith work.

```
SELECT
FROM Employees E
WHERE E.name = 'Smith'
```

How to find the **employee_id(s)** of employee named Smith?

Employees

employee_id	name	salary
1	Jones	26000
2	Smith	28000
3	Parker	35000
4	Smith	24000

Works_in

employee_id	department_id	since
1	1	2001-1-1
2	1	2002-4-1
2	2	2005-2-2
3	3	2003-1-1
4	3	2005-1-1

Departments

department_id	name	budget
1	Toys	122000
2	Tools	239000
3	Food	100000

Exercises

- **Query 4:** Find the dept. ids where employees named Smith work.

```
SELECT W.department_id
FROM Employees E, Works_in W
WHERE E.name = 'Smith' AND
      E.employee_id = W.employee_id;
```

How to find the **employee_id(s)** of employee named Smith?

Given an **employee_id**, find that employee's department id(s).

Employees

employee_id	name	salary
1	Jones	26000
2	Smith	28000
3	Parker	35000
4	Smith	24000

Works_in

employee_id	department_id	since
1	1	2001-1-1
2	1	2002-4-1
2	2	2005-2-2
3	3	2003-1-1
4	3	2005-1-1

Departments

department_id	name	budget
1	Toys	122000
2	Tools	239000
3	Food	100000

Exercises

- **Query 5:** Find the dept. names where employees named Smith work.

```
SELECT
FROM Employees E
WHERE E.name = 'Smith'
```

How to find the **employee_id(s)** of employee named Smith?

Employees

employee_id	name	salary
1	Jones	26000
2	Smith	28000
3	Parker	35000
4	Smith	24000

Works_in

employee_id	department_id	since
1	1	2001-1-1
2	1	2002-4-1
2	2	2005-2-2
3	3	2003-1-1
4	3	2005-1-1

Departments

department_id	name	budget
1	Toys	122000
2	Tools	239000
3	Food	100000

Exercises

- **Query 5:** Find the dept. names where employees named Smith work.

```
SELECT
FROM Employees E, Works_in W
WHERE E.name = 'Smith'  AND
        E.employee_id = W.employee_id
```

How to find the **employee_id(s)** of employee named Smith?

Given an **employee_id**, find the dept. id (s) of that employee.

Employees

employee_id	name	salary
1	Jones	26000
2	Smith	28000
3	Parker	35000
4	Smith	24000

Works_in

employee_id	department_id	since
1	1	2001-1-1
2	1	2002-4-1
2	2	2005-2-2
3	3	2003-1-1
4	3	2005-1-1

Departments

department_id	name	budget
1	Toys	122000
2	Tools	239000
3	Food	100000

Exercises

- **Query 5:** Find the dept. names where employees named Smith work.

```
SELECT D.name
FROM Employees E, Works_in W, Departments D
WHERE E.name = 'Smith'  AND
        E.employee_id = W.employee_id AND
        W.department_id = D.department_id;
```

How to find the employee_id(s) of employee named Smith?

Given an **employee_id**, find the **dept. id (s)** of that employee.

Given a **dept. id**, find the name of that department.

Employees

employee_id	name	salary
1	Jones	26000
2	Smith	28000
3	Parker	35000
4	Smith	24000

Works_in

employee_id	department_id	since
1	1	2001-1-1
2	1	2002-4-1
2	2	2005-2-2
3	3	2003-1-1
4	3	2005-1-1

Departments

department_id	name	budget
1	Toys	122000
2	Tools	239000
3	Food	100000

Exercises

- **Query 6:** Find the names of departments which have an employee named Smith and their budget is greater than 100000.

```
SELECT
FROM Departments D
WHERE D.budget > 100000
```

How to find the **dept.id(s)** of the dept with budget > 100000?

Employees

employee_id	name	salary
1	Jones	26000
2	Smith	28000
3	Parker	35000
4	Smith	24000

Works_in

employee_id	department_id	since
1	1	2001-1-1
2	1	2002-4-1
2	2	2005-2-2
3	3	2003-1-1
4	3	2005-1-1

Departments

department_id	name	budget
1	Toys	122000
2	Tools	239000
3	Food	100000

Exercises

- **Query 6:** Find the names of departments which have an employee named Smith and their budget is greater than 100000.

```
SELECT
FROM Departments D, Employees E, Works_in W
WHERE D.budget > 100000 AND
      E.name = 'Smith' AND
      W.employee_id = E.employee_id
```

How to find the **dept.id(s)** of the dept with budget > 100000?

How to find the **dept.id(s)** of the dept with employee 'Smith'?

Employees

employee_id	name	salary
1	Jones	26000
2	Smith	28000
3	Parker	35000
4	Smith	24000

Works_in

employee_id	department_id	since
1	1	2001-1-1
2	1	2002-4-1
2	2	2005-2-2
3	3	2003-1-1
4	3	2005-1-1

Departments

department_id	name	budget
1	Toys	122000
2	Tools	239000
3	Food	100000

Exercises

- **Query 6:** Find the names of departments which have an employee named Smith and their budget is greater than 100000.

```
SELECT D.name
FROM Departments D, Employees E, Works_in W
WHERE D.budget > 100000 AND
      E.name = 'Smith' AND
      W.employee_id = E.employee_id AND
      D.department_id = W.department_id;
```

How to find the **dept.id(s)** of the dept with budget > 100000?

How to find the **dept.id(s)** of the dept with employee 'Smith'?

Given a **dept.id**, find the name of the department.

Employees

employee_id	name	salary
1	Jones	26000
2	Smith	28000
3	Parker	35000
4	Smith	24000

Works_in

employee_id	department_id	since
1	1	2001-1-1
2	1	2002-4-1
2	2	2005-2-2
3	3	2003-1-1
4	3	2005-1-1

Departments

department_id	name	budget
1	Toys	122000
2	Tools	239000
3	Food	100000

Exercises

- **Query 7:** Find the budgets of departments, who employ an employee called ‘Smith’ .

```
SELECT
FROM Employees E
WHERE E.name = 'Smith'
```

How to find the **employee_id(s)** of the employee named ‘Smith’?

Employees

employee_id	name	salary
1	Jones	26000
2	Smith	28000
3	Parker	35000
4	Smith	24000

Works_in

employee_id	department_id	since
1	1	2001-1-1
2	1	2002-4-1
2	2	2005-2-2
3	3	2003-1-1
4	3	2005-1-1

Departments

department_id	name	budget
1	Toys	122000
2	Tools	239000
3	Food	100000

Exercises

- **Query 7:** Find the budgets of departments, who employ an employee called ‘Smith’ .

```
SELECT
FROM Employees E, Works_in W
WHERE E.name = 'Smith' AND
      E.employee_id = W.employee_id
```

How to find the **employee_id(s)** of the employee named ‘Smith’?

Given an **employee_id**, find the department_id of the employee.

Employees

employee_id	name	salary
1	Jones	26000
2	Smith	28000
3	Parker	35000
4	Smith	24000

Works_in

employee_id	department_id	since
1	1	2001-1-1
2	1	2002-4-1
2	2	2005-2-2
3	3	2003-1-1
4	3	2005-1-1

Departments

department_id	name	budget
1	Toys	122000
2	Tools	239000
3	Food	100000

Exercises

- **Query 7:** Find the budgets of departments, who employ an employee called ‘Smith’ .

```
SELECT D.budget
FROM Employees E, Works_in W, Departments D
WHERE E.name = 'Smith' AND
      E.employee_id = W.employee_id AND
      W.department_id = D.department_id;
```

How to find the employee_id(s) of the employee named ‘Smith’?

Given an employee_id, find the **department_id** of the employee.

Given a **department_id**, find the budget of the department.

Employees

employee_id	name	salary
1	Jones	26000
2	Smith	28000
3	Parker	35000
4	Smith	24000

Works_in

employee_id	department_id	since
1	1	2001-1-1
2	1	2002-4-1
2	2	2005-2-2
3	3	2003-1-1
4	3	2005-1-1

Departments

department_id	name	budget
1	Toys	122000
2	Tools	239000
3	Food	100000

Exercises

- **Query 8:** For each department, find the total number of employees it employs.

```
SELECT W.department_id, COUNT(*)
FROM Works_in W
GROUP BY W.department_id;
```

Employees

employee_id	name	salary
1	Jones	26000
2	Smith	28000
3	Parker	35000
4	Smith	24000

Works_in

employee_id	department_id	since
1	1	2001-1-1
2	1	2002-4-1
2	2	2005-2-2
3	3	2003-1-1
4	3	2005-1-1

Departments

department_id	name	budget
1	Toys	122000
2	Tools	239000
3	Food	100000

Exercises

- **Query 9:** Find the dept. names with **at least 2** employee.

```
SELECT COUNT(*)
FROM Works_in W

GROUP BY D.department_id
```

Find the number of employees in each department.

Employees

employee_id	name	salary
1	Jones	26000
2	Smith	28000
3	Parker	35000
4	Smith	24000

Works_in

employee_id	department_id	since
1	1	2001-1-1
2	1	2002-4-1
2	2	2005-2-2
3	3	2003-1-1
4	3	2005-1-1

Departments

department_id	name	budget
1	Toys	122000
2	Tools	239000
3	Food	100000

Exercises

- **Query 9:** Find the dept. names with **at least 2** employee.

```
SELECT COUNT(*)
FROM Works_in W

GROUP BY D.department_id
HAVING COUNT(*) >= 2;
```

Find the number of employees in each department.

Find the **department** with #employee >= 2.

Employees

employee_id	name	salary
1	Jones	26000
2	Smith	28000
3	Parker	35000
4	Smith	24000

Works_in

employee_id	department_id	since
1	1	2001-1-1
2	1	2002-4-1
2	2	2005-2-2
3	3	2003-1-1
4	3	2005-1-1

Departments

department_id	name	budget
1	Toys	122000
2	Tools	239000
3	Food	100000

Exercises

- **Query 9:** Find the dept. names with **at least 2** employee.

```
SELECT COUNT(*), D.name
FROM Works_in W, Departments D
WHERE D.department_id = W.department_id
GROUP BY D.department_id
HAVING COUNT(*) >= 2;
```

Find the number of employees in each department.

Find the **department** with #employee >= 2.

Given a **department_id**, find the name of the department.

Employees

employee_id	name	salary
1	Jones	26000
2	Smith	28000
3	Parker	35000
4	Smith	24000

Works_in

employee_id	department_id	since
1	1	2001-1-1
2	1	2002-4-1
2	2	2005-2-2
3	3	2003-1-1
4	3	2005-1-1

Departments

department_id	name	budget
1	Toys	122000
2	Tools	239000
3	Food	100000

Exercises

- **Query 10:** Find the employee_id of all employees whose name includes the substring “one”.

```
SELECT employee_id
FROM Employees
WHERE name LIKE '%one%';
```

Employees

employee_id	name	salary
1	Jones	26000
2	Smith	28000
3	Parker	35000
4	Smith	24000

Works_in

employee_id	department_id	since
1	1	2001-1-1
2	1	2002-4-1
2	2	2005-2-2
3	3	2003-1-1
4	3	2005-1-1

Departments

department_id	name	budget
1	Toys	122000
2	Tools	239000
3	Food	100000

Exercises

- **Query 11:** Find the employee_id and name of the employees who worked in the departments with budget more than 100,000.

Find the department_id(s) of the department with budget > 100000.

```
SELECT department_id
FROM Departments
WHERE budget > 100000
```

Employees

employee_id	name	salary
1	Jones	26000
2	Smith	28000
3	Parker	35000
4	Smith	24000

Works_in

employee_id	department_id	since
1	1	2001-1-1
2	1	2002-4-1
2	2	2005-2-2
3	3	2003-1-1
4	3	2005-1-1

Departments

department_id	name	budget
1	Toys	122000
2	Tools	239000
3	Food	100000

Exercises

- **Query 11:** Find the employee_id and name of the employees who worked in the departments with budget more than 100,000.

```
SELECT
FROM Works_in W
WHERE
        W.department_id IN (
                SELECT department_id
                FROM Departments
                WHERE budget > 100000);
```

Find the department_id(s) of the department with budget > 100000.

Given a dpt. id, find the id of the employees in that dpt.

Employees

employee_id	name	salary
1	Jones	26000
2	Smith	28000
3	Parker	35000
4	Smith	24000

Works_in

employee_id	department_id	since
1	1	2001-1-1
2	1	2002-4-1
2	2	2005-2-2
3	3	2003-1-1
4	3	2005-1-1

Departments

department_id	name	budget
1	Toys	122000
2	Tools	239000
3	Food	100000

Exercises

- **Query 11:** Find the employee_id and name of the employees who worked in the departments with budget more than 100,000.

```
SELECT DISTINCT E.employee_id, E.name
FROM Works_in W, Employees E
WHERE E.employee_id = W.employee_id AND
      W.department_id IN (
            SELECT department_id
            FROM Departments
            WHERE budget > 100000);
```

Find the department_id(s) of the department with budget > 100000.

Given a dpt. id, find the id of the employees in that dpt.

Given an employee_id, find the name of the employee.

Employees

employee_id	name	salary
1	Jones	26000
2	Smith	28000
3	Parker	35000
4	Smith	24000

Works_in

employee_id	department_id	since
1	1	2001-1-1
2	1	2002-4-1
2	2	2005-2-2
3	3	2003-1-1
4	3	2005-1-1

Departments

department_id	name	budget
1	Toys	122000
2	Tools	239000
3	Food	100000

Exercises

- **Query 12:** Find the name and budget of the department with the greatest budget.

```
SELECT MAX(D.budget)
FROM Departments D
```

Find the **greatest budget** among all departments.

Employees

employee_id	name	salary
1	Jones	26000
2	Smith	28000
3	Parker	35000
4	Smith	24000

Works_in

employee_id	department_id	since
1	1	2001-1-1
2	1	2002-4-1
2	2	2005-2-2
3	3	2003-1-1
4	3	2005-1-1

Departments

department_id	name	budget
1	Toys	122000
2	Tools	239000
3	Food	100000

Exercises

- **Query 12:** Find the name and budget of the department with the greatest budget.

```
SELECT D2.name, D2.budget
FROM Departments D2
WHERE D2.budget = (
            SELECT MAX(D.budget)
            FROM Departments D
      );
```

Find the **greatest budget** among all departments.

Find the department with budget equals to the **greatest budget**.

Employees

employee_id	name	salary
1	Jones	26000
2	Smith	28000
3	Parker	35000
4	Smith	24000

Works_in

employee_id	department_id	since
1	1	2001-1-1
2	1	2002-4-1
2	2	2005-2-2
3	3	2003-1-1
4	3	2005-1-1

Departments

department_id	name	budget
1	Toys	122000
2	Tools	239000
3	Food	100000

Exercises

- **Query 13:** Find the names of employees who work in at least 2 departments.

```
SELECT E.name
FROM Employees E
WHERE 2 <= (

  // Find the number of dpts that he/she works in.

        );
```

For each employee, find the number of departments that he/she works in.

Employees

employee_id	name	salary
1	Jones	26000
2	Smith	28000
3	Parker	35000
4	Smith	24000

Works_in

employee_id	department_id	since
1	1	2001-1-1
2	1	2002-4-1
2	2	2005-2-2
3	3	2003-1-1
4	3	2005-1-1

Departments

department_id	name	budget
1	Toys	122000
2	Tools	239000
3	Food	100000

Exercises

- **Query 13:** Find the names of employees who work in at least 2 departments.

```
SELECT E.name
FROM Employees E
WHERE 2 <= (
        SELECT COUNT(W.department_id)
        FROM Works_in W
        WHERE W.employee_id = E.employee_id
      );
```

For each employee, find the number of departments that he/she works in.

Employees

employee_id	name	salary
1	Jones	26000
2	Smith	28000
3	Parker	35000
4	Smith	24000

Works_in

employee_id	department_id	since
1	1	2001-1-1
2	1	2002-4-1
2	2	2005-2-2
3	3	2003-1-1
4	3	2005-1-1

Departments

department_id	name	budget
1	Toys	122000
2	Tools	239000
3	Food	100000

Exercises

- **Query 14:** In each department, find the highest salary of the employee in that department.

```
SELECT MAX(E.salary) , W.department_id
FROM Employees E, Works_in W
WHERE E.employee_id = W.employee_id
GROUP BY W.department_id;
```

Employees

employee_id	name	salary
1	Jones	26000
2	Smith	28000
3	Parker	35000
4	Smith	24000

Works_in

employee_id	department_id	since
1	1	2001-1-1
2	1	2002-4-1
2	2	2005-2-2
3	3	2003-1-1
4	3	2005-1-1

Departments

department_id	name	budget
1	Toys	122000
2	Tools	239000
3	Food	100000

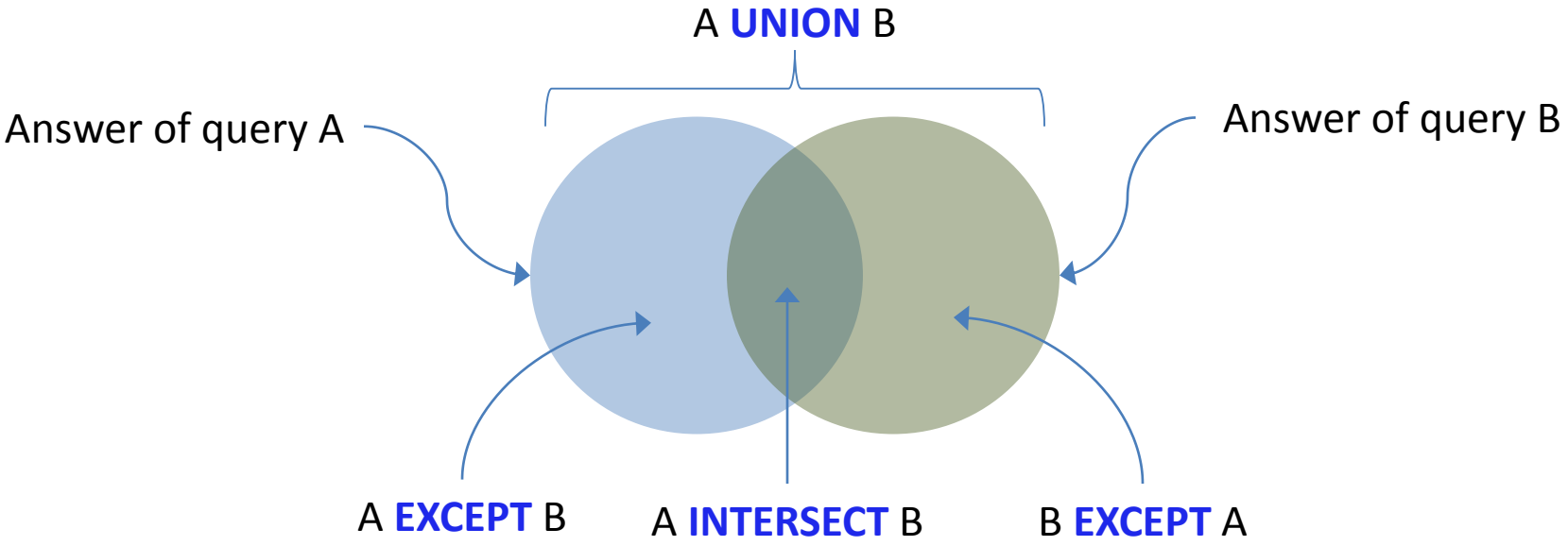
Section 1

Set Operations

Slides prepared by - **Dr. Chui Chun Kit**, http://www.cs.hku.hk/~ckchui/ For other uses, please email : ckchui@cs.hku.hk
Acknowledgement – Professor Nikos Mamoulis http://www.cs.hku.hk/~nikos/

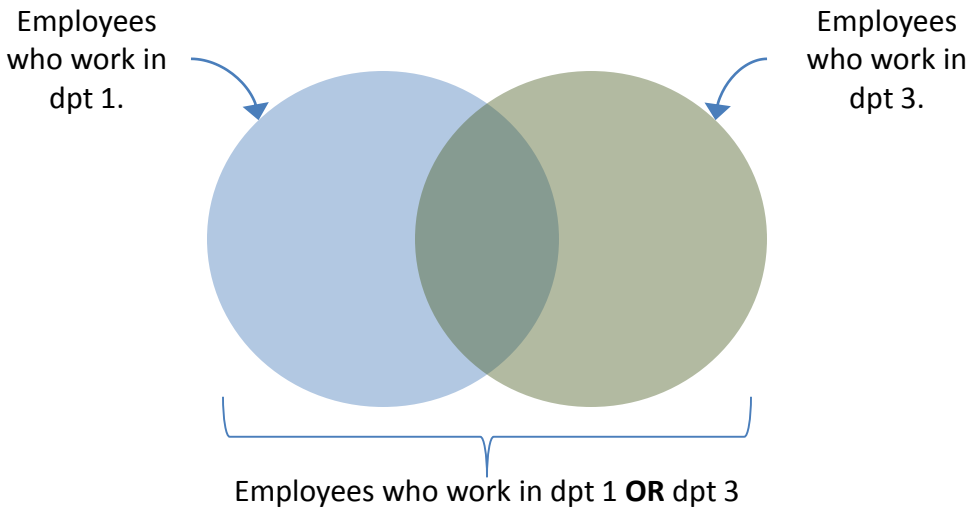
Set operations

- Set operations can be expressed in SQL using clauses **UNION**, **INTERSECT**, **EXCEPT**.
- Using the set operations can ease the design of SQL by breaking down a complex query to a number of simpler sub-queries.



The UNION clause

- **Query:** Find the names of employees who work in department 1 **or** department 3.



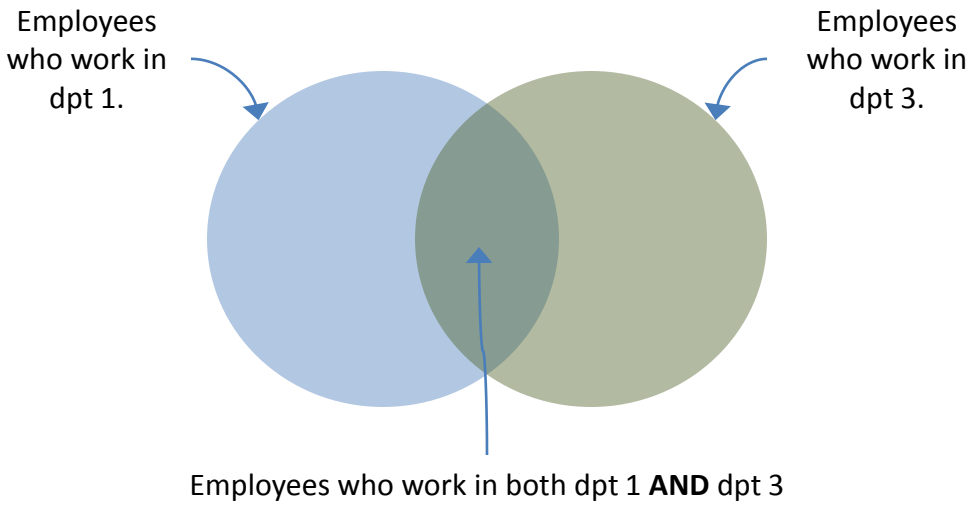

```
SELECT E.name
FROM Employees E, Works_in W
WHERE E.employee_id= W.employee_id AND
        (W. department_id = 1 OR
         W. department_id = 3)
```

```
SELECT E.name
FROM Employees E, Works_in W
WHERE E.employee_id = W.employee_id AND
        W.department_id = 1

UNION

SELECT E.name
FROM Employees E, Works_in W
WHERE E.employee_id = W.employee_id AND
        W. department_id = 3
```

Note : The two SQLs are NOT equivalent to each other!
Duplicates are eliminated when two sets are unified.

The INTERSECT clause

- **Query:** Find the name of employees who work in department 1 **and** department 3.

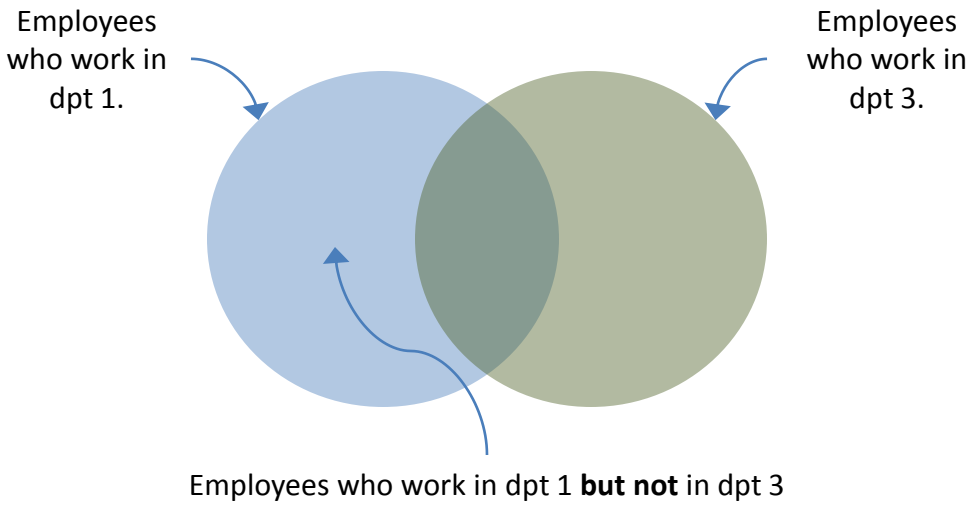



```
SELECT E.name
FROM Employees E, Works_in W
WHERE E.employee_id = W.employee_id AND
        W.department_id = 1

INTERSECT

SELECT E.name
FROM Employees E, Works_in W
WHERE E.employee_id = W.employee_id AND
        W. department_id = 3
```

Note : MySQL doesn't support the keyword INTERSECT.
But we can replace INTERSECT by joining tables.

The EXCEPT clause

- **Query:** Find the name of employees who work in department 1 **but not** department 3.




```
SELECT E.name
FROM Employees E, Works_in W
WHERE E.employee_id = W.employee_id AND
        W.department_id = 1

EXCEPT

SELECT E.name
FROM Employees E, Works_in W
WHERE E.employee_id = W.employee_id AND
        W. department_id = 3
```

Note : MySQL doesn't support the keyword EXCEPT.
But we can replace EXCEPT by using NOT IN.

Section 2

More on Nested Queries

Slides prepared by - **Dr. Chui Chun Kit**, http://www.cs.hku.hk/~ckchui/ For other uses, please email : ckchui@cs.hku.hk
Acknowledgement – Professor Nikos Mamoulis http://www.cs.hku.hk/~nikos/

Nested queries

- Nested queries have other **subqueries** embedded in them.
- Used for the ease of expressing a natural language request in SQL.
- Subqueries are usually nested under **WHERE** clauses.
  - May also be enclosed under **FROM** or **HAVING** clauses

The IN clause (also mentioned in Ch. 3A)

- **Query:** Find the names of the employees in department 1.

```
SELECT E.name
FROM Employees E, Works_in W
WHERE E.employee_id = W.employee_id AND
        W.department_id = 1;
```

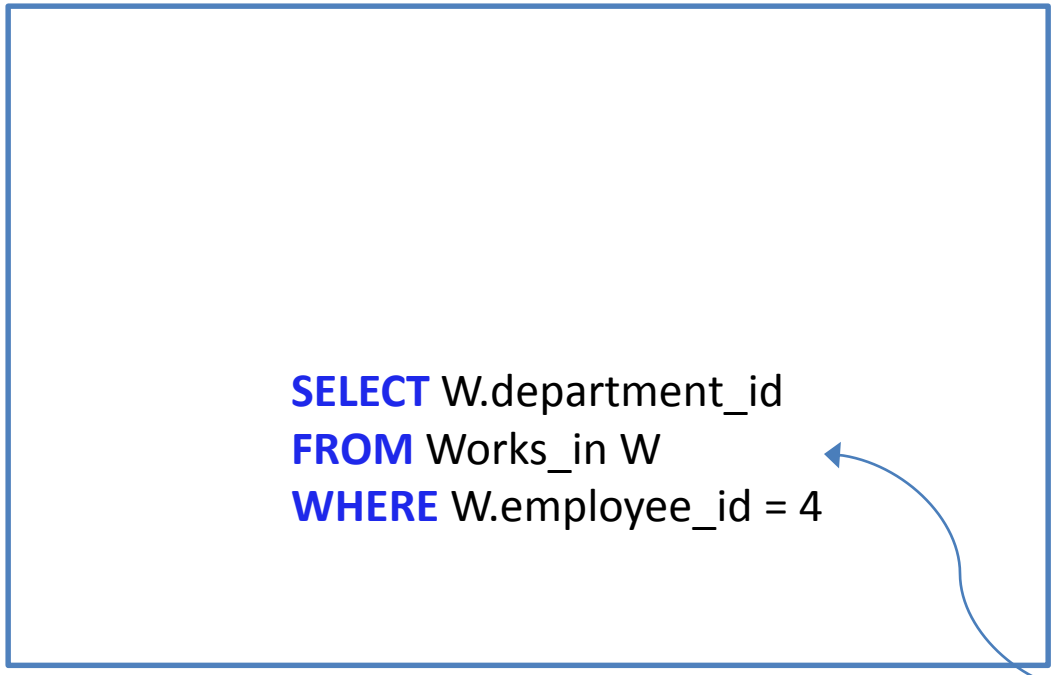
- **In natural language :** Find employee names whose employee_id **appears in the set of** employee_ids working for department 1.

```
SELECT E.name
FROM Employees E
WHERE E.employee_id IN (
        SELECT W.employee_id
        FROM Works_in W
        WHERE W.department_id = 1);
```

Just like searching the employee_id in the **result** of the nested query.

The SOME clause

- **Query:** Find department names that have greater budget than **some** department where employee 4 works.



The SOME clause

- **Query:** Find department names that have greater budget than **some** department where employee 4 works.

```
SELECT D2.budget
FROM Departments D2
WHERE D2.department_id IN (
        SELECT W.department_id
        FROM Works_in W
        WHERE W.employee_id = 4
 )
```

Find the budget of those departments.

Find the department ID where employee 4 works.

The SOME clause

- **Query:** Find department names that have greater budget than **some** department where employee 4 works.

```
SELECT D.name
FROM Departments D
WHERE D.budget > SOME(
        SELECT D2.budget
        FROM Departments D2
        WHERE D2.department_id IN (
            SELECT W.department_id
            FROM Works_in W
            WHERE W.employee_id = 4
        )
);
```

Find the name of the department with budget > some budgets (those returned by inner query).

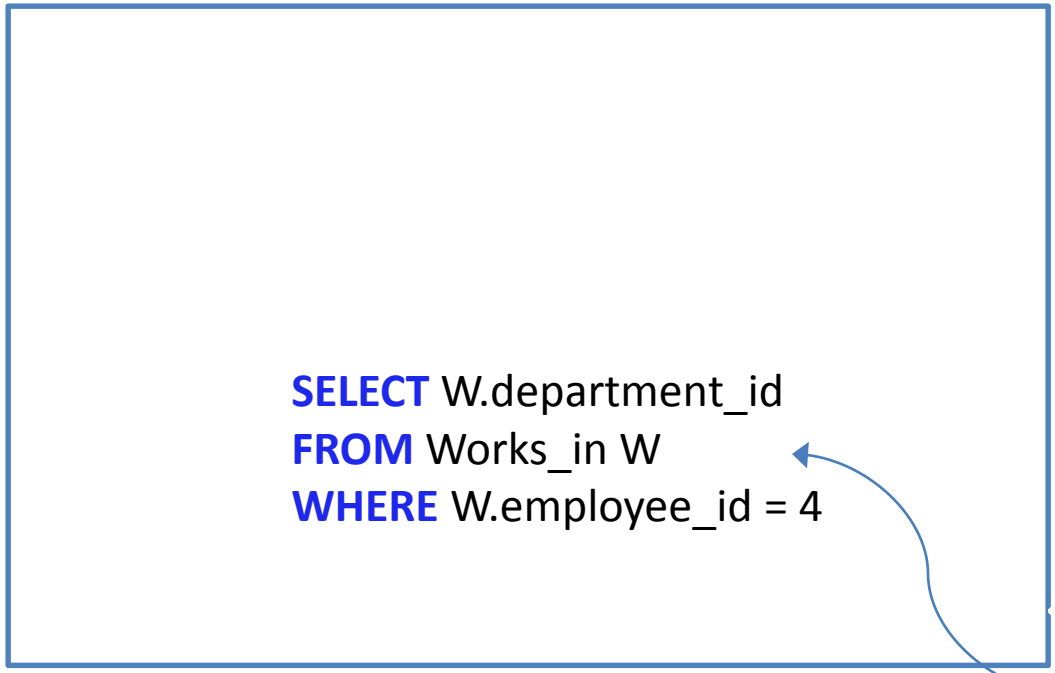
Find the budget of those departments.

Find the department ID where employee 4 works.

- **IMPORTANT NOTE:** If nested query result is empty, then **> SOME** will return **false** for every D.budget!

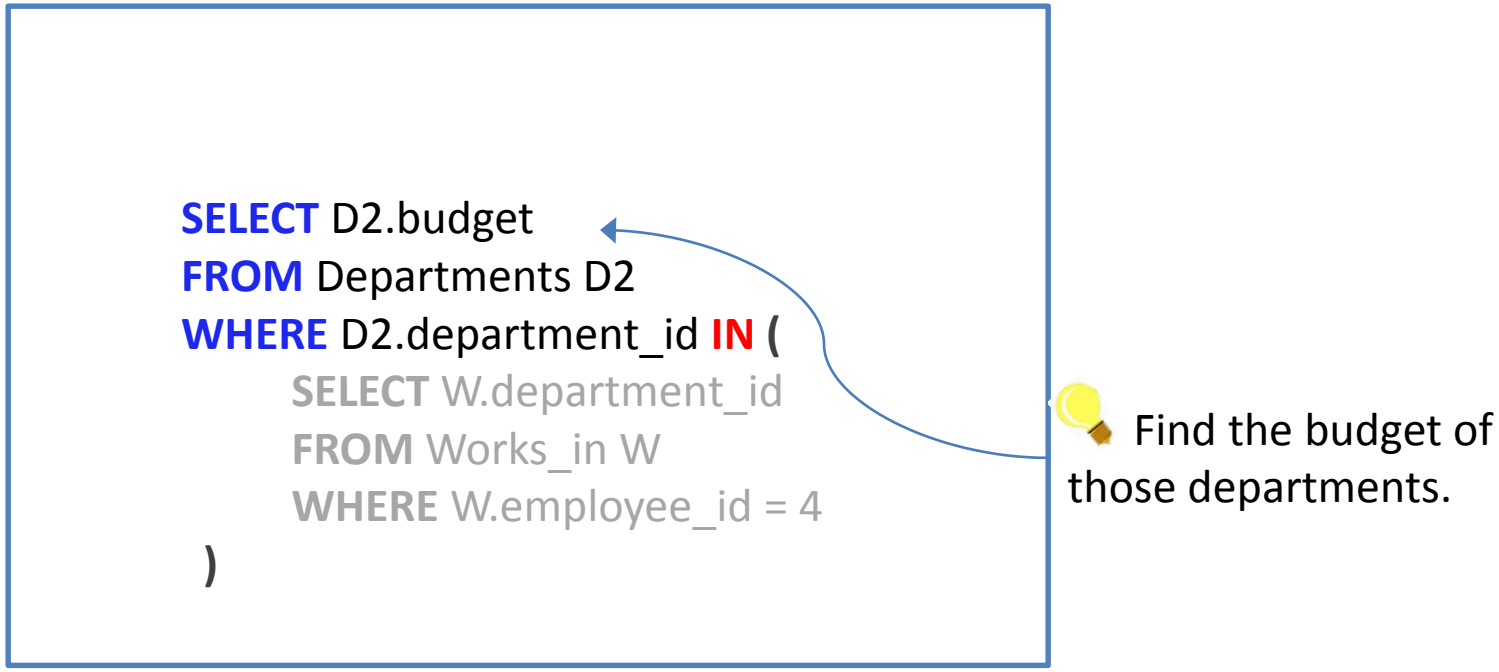
The ALL clause

- **Query:** Find department names that have greater budget than **all** departments where employee 4 works.



The ALL clause

- **Query:** Find department names that have greater budget than **all** departments where employee 4 works.



The ALL clause

- **Query:** Find department names that have greater budget than **all** departments where employee 4 works.

```
SELECT D.name
FROM Departments D
WHERE D.budget > ALL (
        SELECT D2.budget
        FROM Departments D2
        WHERE D2.department_id IN (
            SELECT W.department_id
            FROM Works_in W
            WHERE W.employee_id = 4
        )
);
```

Find the name of the department with budget > ALL budgets (those returned by inner query).

- **IMPORTANT NOTE:** If nested query result is empty, then **> ALL** will return **true** for every D.budget!

The ALL clause

- **Query:** Find department names that have the greatest budget than **all** departments.

```
SELECT D.name
FROM Departments D
WHERE D.budget >= ALL (
            SELECT D2.budget
            FROM Departments D2
);
```

- **Question:** What would the result be if **>ALL** is used?

Can you rewrite the above query using **Aggregate function MAX** in a **nested query**?

The EXISTS clause

- The inner subquery could depend on the row **currently examined** in the outer query.
  - **Query:** Find the names of employees who work in department with department_id=1.

```
SELECT E.name
FROM Employees E
WHERE EXISTS (
            SELECT *
            FROM Works_in W
            WHERE W.department_id = 1 AND
                        E.employee_id = W.employee_id);
```

For each employee record ***r***.
If the inner query can return some records, we will return the record ***r***.

- **EXISTS** is a boolean set-comparison operator that returns **false** if the input set is empty and **true** otherwise.

Section 3

Null values

Slides prepared by - **Dr. Chui Chun Kit**, http://www.cs.hku.hk/~ckchui/ For other uses, please email : ckchui@cs.hku.hk
Acknowledgement – Professor Nikos Mamoulis http://www.cs.hku.hk/~nikos/

NULL value

- Handling null values is a non-trivial topic in database research.
- **null**: **unknown** value or value **does not exist**.
- Use predicate **IS NULL** to check for **null** values.
  - **Query:** Find all employee names for which the salary is unknown or undetermined.

Employees

employee_id	name	salary
1	Jones	
2	Smith	28000
3	Parker	
4	Smith	24000

```
SELECT name
FROM Employees
WHERE salary IS NULL
```

NULL value

- The result of any arithmetic expression involving **null** is **null**.
  - **5** + **null** returns **null**.
- Any comparison with **null** returns **UNKNOWN**.
  - Both **5** < **null** , **null** = **null** return **UNKNOWN**.
- Use ***P*** **IS UNKNOWN** to check if a predicate ***P*** is unknown or not.
- For the result of **WHERE** or **HAVING** clause, predicate is **false** if it evaluates to **UNKNOWN**.

Three valued logic

OR

	T	Un	F
T	T	T	T
Un	T	Un	Un
F	T	Un	F

AND

	T	Un	F
T	T	Un	F
Un	Un	Un	F
F	F	F	F

NOT

T	F
Un	Un
F	T

NULL value and aggregates

```
SELECT SUM (budget)
FROM Departments
```

- The statement above ignores **null** amounts.
- All aggregate operations except **COUNT(*)** ignore tuples with **null** values on the aggregated attributes.
  - **COUNT** counts not **null** values only.

Departments

department_id	name	budget
1	Toys	
2	Tools	
3	Food	100000

- **SUM**(budget) returns **100000**.
- **COUNT**(*) returns **3**.

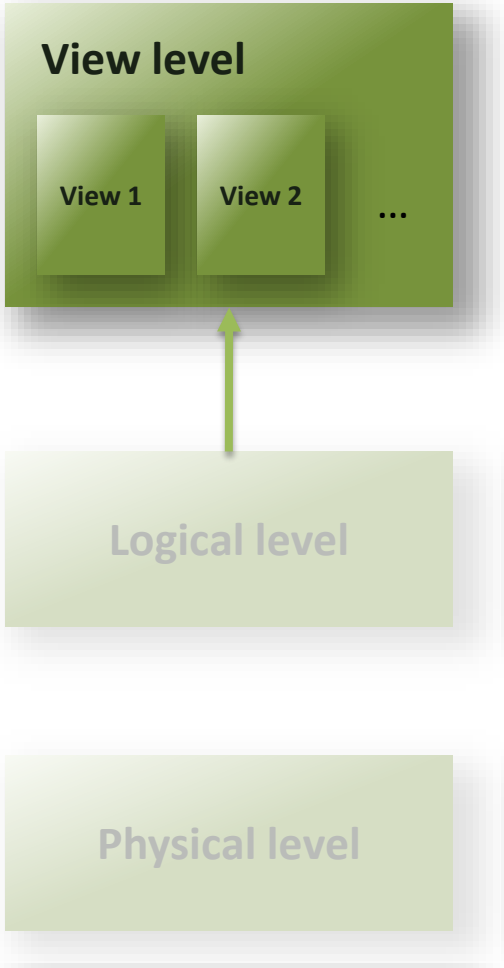
Section 4

Views

Slides prepared by - **Dr. Chui Chun Kit**, http://www.cs.hku.hk/~ckchui/ For other uses, please email : ckchui@cs.hku.hk
Acknowledgement – Professor Nikos Mamoulis http://www.cs.hku.hk/~nikos/

The CREATE VIEW clause

- Views provide a mechanism to hide certain data from the view of certain users.



- **Syntax:**

CREATE VIEW *view_name* **AS** *<expression>*

```
CREATE VIEW Employee_hide_salary AS (
        SELECT employee_id, name
        FROM Employees);
```

```
CREATE VIEW Dpt_size(name,num_of_employee) AS (
        SELECT D.name, COUNT(*)
        FROM Departments D, Works_in W
        WHERE D.department_id = W.department_id
        GROUP BY W.department_id );
```

Section 5

Authorization

Slides prepared by - **Dr. Chui Chun Kit**, http://www.cs.hku.hk/~ckchui/ For other uses, please email : ckchui@cs.hku.hk
Acknowledgement – Professor Nikos Mamoulis http://www.cs.hku.hk/~nikos/

Authorization

- The DBA can grant access/update authorization to users.
- **Syntax:**

GRANT <priviledge list>
ON <table name or view name>
TO <user/role list>

Johnson, Brown are usernames.

GRANT SELECT ON Departments **TO** *Johnson, Brown*

GRANT UPDATE(budget) **ON** Departments **TO** *Johnson*

GRANT UPDATE(budget) **ON** Departments **TO** *manager*

manager is not a username, it is a **role**.

Authorization

- Rights can be revoked.

```
REVOKE SELECT ON Departments FROM Johnson, Brown
```

- Create a role.

```
CREATE ROLE manager;
```

- Grant a role to a user.

```
GRANT manager TO Brown;
```

Section 6

Assertion

Slides prepared by - **Dr. Chui Chun Kit**, http://www.cs.hku.hk/~ckchui/ For other uses, please email : ckchui@cs.hku.hk
Acknowledgement – Professor Nikos Mamoulis http://www.cs.hku.hk/~nikos/

Assertions

- An **assertion** ensures a certain condition will always exist in the database.
  - Assume that we want to enforce that the number of departments cannot exceed the number of employees at any valid instance of our database.

```
CREATE ASSERTION EmpsNoLessThanDepts
CHECK (
        (SELECT COUNT(*) FROM Departments)
        <=
        (SELECT COUNT(*) FROM Employees)
      );
```

- **Assertions are checked every time the involved tables are updated** and they could be very expensive.

Section 7

Other topics in SQL

Slides prepared by - **Dr. Chui Chun Kit**, http://www.cs.hku.hk/~ckchui/ For other uses, please email : ckchui@cs.hku.hk
Acknowledgement – Professor Nikos Mamoulis http://www.cs.hku.hk/~nikos/

List of other topics

- Join types: Full outer join, Inner join, Cross join…
- **Transaction controls.**
- Derived relations and the **WITH** clause.
- Set operations: **UNIQUE**
- Triggers
- More on foreign key constraints: **ON DELETE CASCADE**
- More on DCL (Data Control Language) **WITH GRANT OPTION**

Chapter 3B.

END

CSIS0278 / COMP3278

Introduction to

Database Management Systems

Slides prepared by - **Dr. Chui Chun Kit**, http://www.cs.hku.hk/~ckchui/ For other uses, please email : ckchui@cs.hku.hk

Acknowledgement – Professor Nikos Mamoulis http://www.cs.hku.hk/~nikos/