INTRODUCTION TO DATABASE II

Based on materials by Kevin C. Chang

Review

- ฐานข้อมูลเป็นแหล่งรวบรวมข้อมูล (data) ไว้อย่างเป็นระบบ
- ในคลาสนี้เราจะใช้แบบจำลอง **Entity-Relationship Model**
- **Entity** คือสิ่งของในชีวิตจริงที่สามารถบอกถึงความแตกต่างจากสิ่งของอื่น ๆ ได้ โดยเราจะอธิบาย Entity ได้ด้วยกลุ่มของ Attributes
- **Attributes** คือคุณลักษณะย่อย ๆ เช่น ข้อความ จำนวนจริง จำนวนเต็ม
- **Entity Set** คือกลุ่มของ Entity ที่มีกลุ่มของ Attributes คล้ายกัน
- **Key** คือ Attribute หรือ กลุ่มของ Attribute ใน Entity Set ที่ต้องมีค่าไม่ซ้ำกันสำหรับแต่ละ Entity ใน Entity Set นั้น

SQL

- SQL หรือ Structured Query Language คือภาษาที่ใช้ในการจัดการกับฐานข้อมูล

SELECT ชื่อ Attributes

FROM ชื่อ Entity Set หรือตาราง

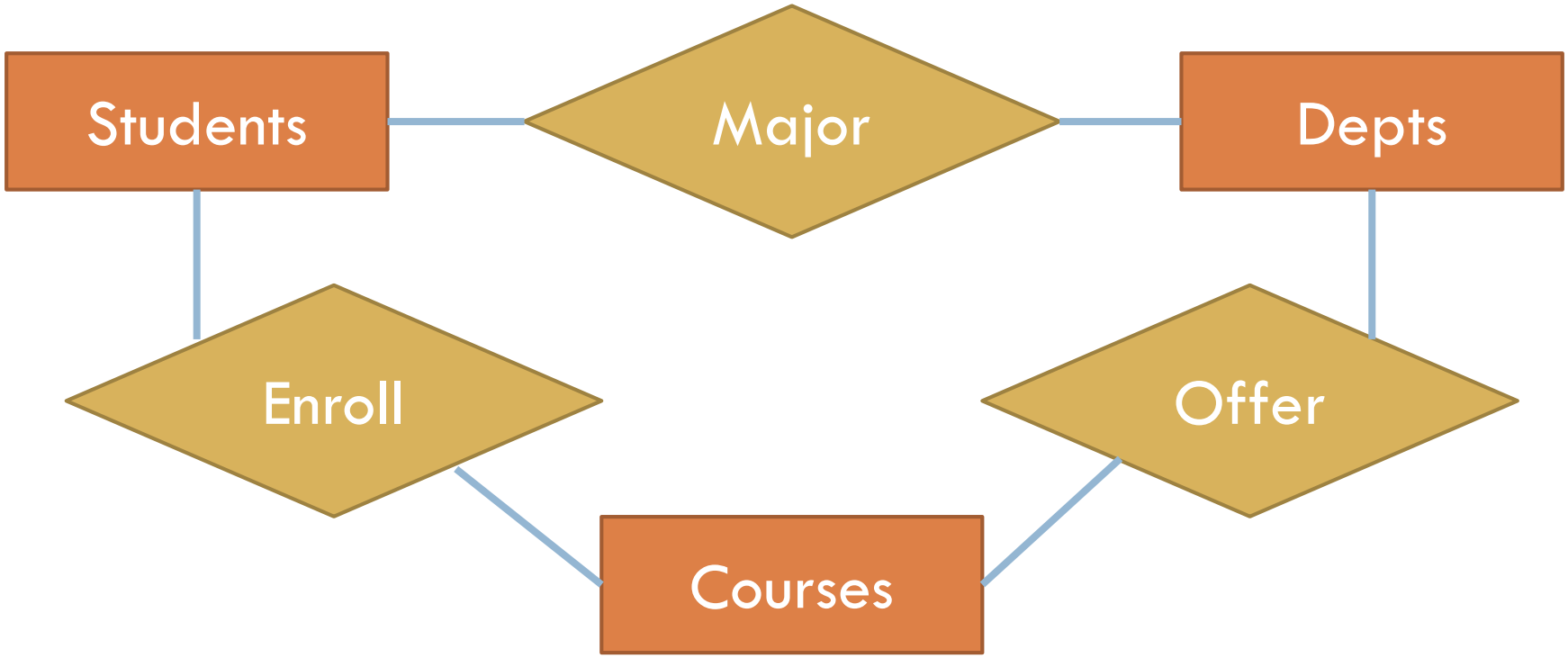
(WHERE เงื่อนไข)

Running Example

- ผมต้องการเก็บข้อมูลการลงทะเบียนเรียนของนักศึกษา
- จากคำถามข้างต้น เราจะเห็นว่ามี **Entity Set** อยู่สามเซต
  - นักศึกษา
  - กระบวนวิชา
  - ภาควิชา
- จาก **Entity Set** ข้างต้น เราสามารถสร้างได้ **3** ความสัมพันธ์
  - นักศึกษาคนนี้เรียนวิชาไหน
  - นักศึกษาคนนี้อยู่ในภาควิชาไหน
  - กระบวนวิชานี้เปิดสอนโดยภาควิชาไหน

Running Example

- จาก **Entity Set** และ **Relationship** ข้างต้น เราสามารถนำมาเขียนเป็นแผนภาพ **ER Diagram** ได้ดังนี้



Running Example

- เพื่อให้ง่ายต่อการจำ เราจำเขียน **Entity Set** และ **Relationship** ดังนี้
  - Students(<u>name</u>, addr, tel)
  - Courses(<u>name</u>, type)
  - Depts(<u>name</u>, addr, tel)
  - Enroll(<u>student</u>, <u>course</u>)
  - Major(<u>student</u>, <u>dept</u>)
  - Offer(<u>dept</u>, <u>course</u>, credit)
- ในวงเล็บคือ **Attribute** ของตารางนั้น ๆ
- สังเกตว่าความสัมพันธ์จะมี 2 Key เนื่องจากเชื่อม 2 ความสัมพันธ์

One Entity Query

- ถ้าเราต้องการทราบว่ามีวิชาไหนบ้างที่มีการเรียนการสอนแบบ **Lecture**

```
SELECT name
FROM Courses
WHERE type = 'Lecture'
```

Students(name, addr, tel)
Courses(name, type)
Depts(name, addr, tel)
Enroll(student, course)
Major(student, dept)
Offer(dept, course, credit)

Renaming Attribute

- เราสามารถเปลี่ยนชื่อ Attribute (ชื่อคอลัมน์) โดยใช้ Keyword AS

```
SELECT name AS class
FROM Courses
WHERE type = 'Lecture'
```

Students(name, addr, tel)
Courses(name, type)
Depts(name, addr, tel)
Enroll(student, course)
Major(student, dept)
Offer(dept, course, credit)

- เวลาที่เราจะดึงค่าของตารางใน PHP เราต้องใช้ $row["class"] แทนที่จะเป็น $row["name"]

Expression in SELECT

- เราสามารถใช้สมการที่เหมาะสมในส่วนของ SELECT ได้โดย

```
SELECT dept, course, credit*20
                    AS fee
FROM Offer
```

Students(name, addr, tel)
Courses(name, type)
Depts(name, addr, tel)
Enroll(student, course)
Major(student, dept)
Offer(dept, course, credit)

- ถ้าหากเราไม่ได้เปลี่ยนชื่อ Attribute ชื่อคอลัมน์จะเป็น credit*20

Complex Conditions in WHERE

- ในส่วนของ WHERE เราสามารถใช้ตัวดำเนินการทางตรรกศาสตร์ได้
- ถ้าเราต้องการทราบว่าภาควิชาสถิติเปิดวิชาไหนบ้างที่เป็น 1 หน่วยกิจ หรือ 3 หน่วยกิจ

Students(name, addr, tel)
Courses(name, type)
Depts(name, addr, tel)
Enroll(student, course)
Major(student, dept)
Offer(dept, course, credit)

```
SELECT course
FROM Offer
WHERE dept = 'Statistics'
      AND (credit = 1 OR credit = 3)
```

Pattern Matching in WHERE

- นอกจากเราจะเทียบข้อความด้วยเครื่องหมายเท่ากับแล้ว เรายังใช้การเปรียบเทียบบางส่วนของข้อความได้โดยใช้ตัวดำเนินการ LIKE
- ถ้าเราต้องการทราบว่านักศึกษาคนใดบ้างเป็นผู้ชาย

```
SELECT name
FROM Students
WHERE name LIKE 'นาย%'
```

- ตัวเนินการ LIKE ใช้ได้กับข้อความเท่านั้น

Students(name, addr, tel)
Courses(name, type)
Depts(name, addr, tel)
Enroll(student, course)
Major(student, dept)
Offer(dept, course, credit)

Pattern Matching in WHERE

- เราสามารถใช้สัญลักษณ์ % และ _ เมื่อใช้ตัวดำเนินการ LIKE โดย
  - % แทนข้อความกี่ตัวอักษรก็ได้ (0 ตัวอักษรก็ได้)
  - _ แทนหนึ่งตัวอักษร

```
SELECT name
FROM Students
WHERE tel LIKE '%53-_____'
```

- ตัวอย่างด้านบนเป็นการค้นหานักศึกษา

ที่อยู่ในเชียงใหม่ โดยใช้เบอร์โทรศัพท์เป็นตัวระบุ

Students(name, addr, tel)
Courses(name, type)
Depts(name, addr, tel)
Enroll(student, course)
Major(student, dept)
Offer(dept, course, credit)

Multi-Relation Query

- เราสามารถสร้าง Query โดยเรียกใช้ความสัมพันธ์มากกว่าหนึ่งความสัมพันธ์พร้อมกันได้ โดยเพิ่มความสัมพันธ์ที่ต้องการในส่วน FROM
- เนื่องจากความสัมพันธ์ทั้งสองอาจจะมีชื่อ Attributes ซ้ำกันได้ ดังนั้นในการอ้างอิง Attributes ของความสัมพันธ์ใด เราต้องจะต้องอ้างอิงด้วย
  - ชื่อความสัมพันธ์.ชื่อAttribute

Multi-Relation Query Examples

- ถ้าต้องการทราบว่าวิชาไหนที่เปิดโดยภาควิชาสถิติแล้วมีนักศึกษาลงเรียนอย่างน้อยหนึ่งคน

Students(name, addr, tel)
Courses(name, type)
Depts(name, addr, tel)
Enroll(student, course)
Major(student, dept)
Offer(dept, course, credit)

```
SELECT Enroll.course
FROM Enroll, Offer
WHERE Offer.dept = 'Statistic'
AND Enroll.course = Offer.course
```

- สังเกตว่าใน Query นี้เรานำ Key ของทั้งสองความสัมพันธ์มาเทียบกัน

Multi-Relation Query Examples

- ถ้าต้องการทราบว่าแล้วมีนักศึกษาเอกสถิติลงเรียนวิชาใดบ้าง

Students(name, addr, tel)
Courses(name, type)
Depts(name, addr, tel)
Enroll(student, course)
Major(student, dept)
Offer(dept, course, credit)

```
SELECT Enroll.course
FROM Enroll, Major
WHERE Major.dept = 'Statistic'
AND Enroll.student = Major.student
```

- เราเรียนการรวมกันของสองความสัมพันธ์ในลักษณะนี้ว่า **Inner Join** โดยเงื่อนไข **Enroll.student = Major.student** จะถูกเรียกว่า **Join Condition**

Multi-Relation Query Examples

- จะเกิดอะไรขึ้นถ้าเราลืมใส่ Enroll.student = Major.student

```
SELECT Enroll.course
FROM Enroll, Major
WHERE Major.dept = 'Statistic'
```

Students(name, addr, tel)
Courses(name, type)
Depts(name, addr, tel)
Enroll(student, course)
Major(student, dept)
Offer(dept, course, credit)

- เราจะได้ชื่อวิชาทั้งหมดที่มีการเปิดสอนโดยแต่ละวิชาจะปรากฏเป็นจำนวนครั้งเท่ากับจำนวนนักศึกษาเอกสถิติ
- เราเรียกการรวมกันของสองความสัมพันธ์ในรูปแบบนี้ว่า Cross Join

More Examples

- อยากทราบว่านักศึกษาที่ชื่อ **John Smith** ลงวิชาใดบ้างที่เป็น **Lecture**

Students(name, addr, tel)
Courses(name, type)
Depts(name, addr, tel)
Enroll(student, course)
Major(student, dept)
Offer(dept, course, credit)

More Examples

- อยากทราบว่ามีนักศึกษาคนใดบ้างที่ลงวิชาเอก (เช่นนักศึกษามาจากภาคสถิติและเรียนวิชาที่สอนโดยภาคสถิติ

Students(<u>name</u>, addr, tel)
Courses(<u>name</u>, type)
Depts(<u>name</u>, addr, tel)
Enroll(<u>student</u>, <u>course</u>)
Major(<u>student</u>, <u>dept</u>)
Offer(<u>dept</u>, <u>course</u>, credit)

Explicit Tuple-Variables

- ในบางครั้ง Query อาจจะต้องใช้สองสำเนาของความสัมพันธ์เดียวกัน
- เราสามารถตั้งชื่อความสัมพันธ์ใหม่ในส่วน FROM ได้เพื่อให้สองสำเนาของความสัมพันธ์เดียวกันต่างกัน

Explicit Tuple-Variables

- ตัวอย่าง Query นี้จะให้ผลลัพท์อะไร

```
SELECT o1.course
FROM Offer o1, Offer o2
WHERE o1.course = o2.course
AND o1.credit <> o2.credit
```

Students(name, addr, tel)
Courses(name, type)
Depts(name, addr, tel)
Enroll(student, course)
Major(student, dept)
Offer(dept, course, credit)

- ผลลัพธ์ที่ได้คือรายชื่อวิชาที่มีการสอนมากกว่าหนึ่งภาควิชาซึ่งแต่ละภาควิชาให้จำนวนเครดิตไม่เท่ากัน

More Examples

- ถ้าอยากทราบว่ามีนักศึกษาคนไหนบ้างที่มีวิชาเอกมากกว่าหนึ่งวิชา

Students(name, addr, tel)
Courses(name, type)
Depts(name, addr, tel)
Enroll(student, course)
Major(student, dept)
Offer(dept, course, credit)

SQL Function

- ภาษา SQL ก็มีฟังก์ชันต่าง ๆ เพื่ออำนวยความสะดวกในการทำงาน เช่น
  - AVG() หาค่าเฉลี่ยของ Attribute นั้น
  - COUNT() นับจำนวนแถว
  - MAX() หาค่าสูงสุดของ Attribute นั้น
  - MIN() หาค่าต่ำสุดของ Attribute นั้น
  - SUM() หาผลรวมของ Attribute นั้น
- การใช้ฟังก์ชันเหล่านี้สามารถลดการเขียนโปรแกรมเพื่อคำนวณได้

More Examples

- ถ้าอยากทราบว่านักศึกษาที่ชื่อ **John Smith** ลงเรียนทั้งหมดกี่หน่วยกิจ

Students(name, addr, tel)
Courses(name, type)
Depts(name, addr, tel)
Enroll(student, course)
Major(student, dept)
Offer(dept, course, credit)

More Examples

- ถ้าอยากทราบว่าวิชา IT2 มีนักศึกษาที่ลงเรียนกี่คน

Students(name, addr, tel)
Courses(name, type)
Depts(name, addr, tel)
Enroll(student, course)
Major(student, dept)
Offer(dept, course, credit)

COUNT with GROUP BY

- ถ้าเราต้องการทราบว่าแต่ละวิชาที่คนเรียนกี่คน

```
SELECT course, COUNT(student)
FROM Enroll
```

- Query ด้านบนจะให้ผลลัพธ์เป็นชื่อวิชาแรกและจำนวนแถวทั้งหมดในตาราง ถ้าเราต้องการให้ได้ผลลัพธ์ที่เราต้องการ เราต้องใช้ GROUP BY

```
SELECT course, COUNT(student)
FROM Enroll
GROUP BY course
```

Duplicate Elimination

- ในบางครั้งตารางผลลัพธ์อาจจะมีข้อมูลที่ซ้ำกัน ซึ่งเราอาจจะไม่ต้องการ
- เราสามารถกำจัดข้อมูลซ้ำได้ด้วยคำว่า DISTINCT

```
SELECT DISTINCT type
FROM Courses
```

- จาก Query ด้านบน เราจะได้ผลลัพธ์เป็นประเภทของวิชาทั้งหมด ซึ่งถ้าเราไม่ใช้คำว่า DISTINCT เราก็จะได้คำว่า Lecture และ Lab ซ้ำกันหลาย ๆ รอบ

Students(name, addr, tel)
Courses(name, type)
Depts(name, addr, tel)
Enroll(student, course)
Major(student, dept)
Offer(dept, course, credit)

Summary

- เราสามารถเติมสมการและใช้ฟังก์ชันในส่วน SELECT ได้
- เราสามารถเรียกใช้ความสัมพันธ์เดิมหลายครั้งได้
- LIKE ใช้สำหรับการเปรียบเทียบบางส่วนของข้อมูลที่เป็นข้อความ
- การสร้าง Query ที่ดีจะทำให้เขียนโปรแกรมน้อยลง
- การเปรียบเทียบในภาษา PHP ใช้เท่ากับสองตัว แต่ใน SQL ใช้ตัวเดียว
- ยังมีการใช้คำสั่ง SELECT ในอีกหลายรูปแบบ เช่น Subquery, Set Operation, ANY/ALL และอื่น ๆ ซึ่งจะไม่สอนและไม่ออกสอบ

References

- http://www.w3schools.com/sql/sql_functions.asp