แผนการสอนประจำบทเรียน

รายชื่ออาจารย์ผู้จัดทำ ศิรินุช เทียนรุ่งโรจน์

รายละเอียดของเนื้อหา

ตอนที่ 2.1 Key Rule

เรื่องที่ 2.1.1 แนวคิดเกี่ยวกับฐานข้อมูลเชิงสัมพันธ์

เรื่องที่ 2.1.2 คีย์

ตอนที่ 2.2 Algebra Rule

เรื่องที่ 2.2.1 แนวคิดเกี่ยวกับ Relation Algebra

เรื่องที่ 2.2.2 Relation Algebra พื้นฐาน

ตอนที่ 2.3 Constraints

เรื่องที่ 2.3.1 Constraints

ตอนที่ 2.4 Cartesian Rule

เรื่องที่ 2.4.1 Cartesian product

เรื่องที่ 2.4.2 Join

แนวคิด

1. ในปัจจุบันฐานข้อมูลเชิงสัมพันธ์เป็นที่รู้จักกันแพร่หลายและเป็นที่นิยมใช้ในการจัดเก็บข้อมูลอย่างมาก แนวความคิดที่สนับสนุนหรืออยู่เบื้องหลังของรูปแบบฐานข้อมูลเชิงสัมพันธ์มาจากรูปแบบทางคณิตศาสตร์ที่เรียกว่า relational algebra และ relational calculus
2. รูปแบบฐานข้อมูลเชิงสัมพันธ์ การเก็บข้อมูลจะเป็นแบบรูปตาราง 2 มิติคือ แถวและคอลัมน์นั้นจำเป็นต้องมีคอลัมน์หรือกลุ่มของคอลัมน์ในตารางที่ระบุแต่ละแถวได้อย่างชัดเจน คอลัมน์หรือกลุ่มของคอลัมน์ที่เห็นได้เด่นชัดนี้ที่ใช้ระบุแต่ละแถวและทำให้แถวทั้งหมดแตกต่างกันเรียกว่ากุญแจหลัก (key)
3. รูปแบบฐานข้อมูลเชิงสัมพันธ์ ผู้ใช้สามารถเลือกข้อมูลที่ต้องการดูได้โดยผู้ใช้สามารถใส่เงื่อนไขบังคับ (constraints) เพื่อได้คำตอบที่ต้องการ
4. ผู้ใช้สามารถนำข้อมูลจากหลายๆตารางมารวมกันได้ตามที่ต้องการโดยอาศัยกฎของความสัมพันธ์มาจากรูปแบบทางคณิตศาสตร์เข้ามาช่วย คือ Cartesian Rule

วัตถุประสงค์

หลังจากศึกษาบทเรียนที่ 1 แล้ว นักศึกษาสามารถ

1. เข้าใจแนวความคิดของรูปแบบฐานข้อมูลเชิงสัมพันธ์ บอกถึงรูปแบบและลักษณะที่สำคัญของ relational algebra และ relational calculus
2. เข้าใจหลักการและทฤษฎีทางคณิตศาสตร์ที่อยู่เบื้องหลังของรูปแบบฐานข้อมูลเชิงสัมพันธ์และการกระทำกับข้อมูลที่ต้องการที่ถูกจัดเก็บในฐานข้อมูล

กิจกรรมการเรียนการสอน

กิจกรรมที่นักศึกษาต้องทำสำหรับการเรียนการสอน ได้แก่

1. ศึกษาเอกสารการสอน
2. ปฏิบัติกิจกรรมตามที่ได้รับมอบหมายในเอกสารการสอนแต่ละตอน

สื่อการสอน

1. เอกสารการสอนของชุดวิชา
2. แบบฝึกปฏิบัติ
3. บทความ/ข้อมูลทางคอมพิวเตอร์
4. การให้คำปรึกษาทางโทรศัพท์
5. CD-ROM
6. Homepage ของชุดวิชาผ่านทางอินเตอร์เน็ต

เอกสารประกอบการสอน

1. Fundamentals of Database Systems, by Ramez Elmasri, Shamkant B. Navathe, The Second Edition, 1994
2. Database System Concepts, by Abraham Siberschaty, Henry F.Korth, S.Sudarshan, The Third Edition, 1991

ประเมินผล

1. ประเมินผลจากแบบฝึกหัด/ทดสอบ ในแต่ละบท
2. ประเมินผลจากการสอนประจำภาคการศึกษา

ตอนที่ 2.1 Key Rule

หัวเรื่อง

เรื่องที่ 2.1.1 แนวคิดเรื่องโมเดลเชิงสัมพันธ์

เรื่องที่ 2.1.2 คีย์

แนวคิด

1. รูปแบบฐานข้อมูลเชิงสัมพันธ์ การเก็บข้อมูลจะเป็นแบบรูปตาราง 2 มิติคือ แถวและคอลัมน์นั้น จำเป็นต้องมีคอลัมน์หรือกลุ่มของคอลัมน์ในตารางที่ระบุแต่ละแถวได้อย่างชัดเจน คอลัมน์หรือกลุ่มของคอลัมน์ที่เห็นได้เด่นชัดนี้ที่ใช้ระบุแต่ละแถวและทำให้แถวทั้งหมดแตกต่างกันเรียกว่า กุญแจหลัก (key)

วัตถุประสงค์

หลังจากที่ศึกษาตอนที่ 2.1 แล้ว นักศึกษาสามารถ

1. เข้าใจแนวความคิดของรูปแบบฐานข้อมูลเชิงสัมพันธ์

เรื่องที่ 2.1.1 แนวคิดเรื่องโมเดลเชิงสัมพันธ์

โมเดลเชิงสัมพันธ์ใช้รูปแบบของความสัมพันธ์ที่ชัดเจนมาเป็นตัวกำหนดลักษณะของข้อมูล โมเดลเชิงสัมพันธ์ได้ถูกนำไปใช้อย่างแพร่หลายในระบบการประมวลผลข้อมูล ความสัมพันธ์ในฐานข้อมูลจะอยู่ในรูปแบบของตาราง (table) ดังนั้น

- ชื่อของตารางคือชื่อของความสัมพันธ์
- แต่ละคอลัมน์ของตารางความสัมพันธ์ เรียกว่า แอททริบิว (attribute) ของความสัมพันธ์ ซึ่งแต่ละคุณสมบัติจะมีชื่อเรียกและค่าของแอททริบิวที่แตกต่างกัน ค่าและขอบเขตของข้อมูลของแอททริบิวเรียกว่าโดเมน (domain) เช่น ความสัมพันธ์ “player” ประกอบด้วยแอททริบิว 5 อย่าง คือ ชื่อ ตำแหน่ง อายุ ส่วนสูง น้ำหนัก
- แต่ละแถวของตารางความสัมพันธ์ เรียกว่า แถว หรือ ทูเพิล (tuple) ของความสัมพันธ์

จากที่กล่าวมาข้างต้น สรุปโดยใช้หลักการทางคณิตศาสตร์ได้ว่า

- โดเมน D คือ ชุดของข้อมูลที่มีค่าเฉพาะเจาะจง และมีความหมายเป็นอย่างใดอย่างหนึ่ง
- แอททริบิว A คือ ตัวแทนของโดเมน เขียนได้เป็น dom(A) เมื่อกล่าวถึงแอททริบิว A จะหมายความถึงชุดของข้อมูลชุดหนึ่งที่เป็นที่รู้กันว่ามีขอบเขตเพียงไร
- รูปแบบของความสัมพันธ์ R ซึ่งสามารถเขียนได้เป็น R(A1, A2, ..., An) คือ ชุดของแอททริบิว โดยที่เมื่อกล่าวถึง R จะหมายถึง {A1, A2, ..., An} โดยปริยาย

- ดังนั้น ความสัมพันธ์ r คือ กลุ่มของแถว ที่เป็น subset ของกลุ่มของแอททริบิว dom(A1) x dom(A2) x ... x dom(An)
- หนึ่งแถวในความสัมพันธ์ r จึงเขียนได้เป็น t(A1, A2, ..., An) และมีความหมายเป็น subset หนึ่งของ dom(A1) x dom(A2) x ... x dom(An)

รูปแบบทางคณิตศาสตร์ของความสัมพันธ์ดังข้างต้นได้ถูกนำไปใช้ในการอธิบายรูปแบบของความสัมพันธ์ ซึ่งเป็นการกำหนดชนิดและความหมายของข้อมูล ในฐานข้อมูลสิ่งที่มักทำให้เกิดความสับสนก็คือ รูปแบบของความสัมพันธ์ (Relational schema) และตัวความสัมพันธ์ (Relational instance) รูปแบบของความสัมพันธ์เป็นการกำหนดลักษณะโครงสร้างของความสัมพันธ์ แต่ความสัมพันธ์เป็นข้อมูลที่เกิดความสัมพันธ์กันของแอททริบิว ทั้งรูปแบบและความสัมพันธ์มีหลักในการตั้งชื่อที่ไม่เหมือนกัน นั่นคือ รูปแบบของความสัมพันธ์มักจะตั้งชื่อโดยใช้ตัวอักษรพิมพ์ใหญ่ทุกตัวหรือเฉพาะตัวแรกของชื่อ และตั้งชื่อความสัมพันธ์โดยใช้ตัวอักษรพิมพ์เล็กทั้งหมด ตัวอย่าง รูปแบบของความสัมพันธ์ PLAYER_SCHEMA และ ความสัมพันธ์ player

PLAYER_SCHEME = (ชื่อ, ตำแหน่ง, อายุ, ส่วนสูง, น้ำหนัก)

player(PLAYER_SHEMA)

หมายความว่า รูปแบบของความสัมพันธ์ PLAYER_SCHEMA ประกอบไปด้วยแอททริบิว ชื่อ ตำแหน่ง อายุ ส่วนสูง น้ำหนัก และความสัมพันธ์ player เป็นความสัมพันธ์ที่เกิดจากกลุ่มของแอททริบิวใน PLAYER_SCHEMA ตัวอย่างของความสัมพันธ์ player แสดงดังตารางต่อไปนี้

Name	*position*	*age*	*height*	*weight*
สุเมธ	กองหน้า	26	183	82
ผลดี	ปีกซ้าย	21	175	78
ชาติชาย	กองหลัง	30	169	71
ก้องเกียรติ	กองกลาง	24	180	78

ตาราง 2-1 ความสัมพันธ์ player

คุณสมบัติของความสัมพันธ์

- ลำดับของแถวและคอลัมน์ไม่ทำให้ความหมายของข้อมูลเปลี่ยนไป ดังนั้นความสัมพันธ์ดังตาราง 2-2 จึงมีความหมายเหมือนตาราง 2-1
- แต่ละแถวในความสัมพันธ์จะเป็นเอกลักษณ์ คือจะไม่มีสองแถวที่ซ้ำกัน
- แอททริบิวทุกตัวจะเป็น atomic เท่านั้น ไม่มีแอททริบิวที่เป็น multivalue หรือ composite
- ดีกรีของความสัมพันธ์ คือ จำนวนแอททริบิวที่มีในความสัมพันธ์นั้น

Position	*age*	*name*	*height*	*weight*

กองหน้า	26	สุเมธ	183	82
กองหลัง	30	ชาติชาย	169	71
กองกลาง	24	ก้องเกียรติ	180	78
ปีกซ้าย	21	ผลดี	175	78

ตาราง 2-2 ความสัมพันธ์ player

เรื่องที่ 2.1.2 **คีย์**

คุณสมบัติหนึ่งที่สำคัญของความสัมพันธ์ก็คือ ความเป็นเอกลักษณ์ (Uniqueness property) สิ่งที่ใช้กำหนดความเป็นเอกลักษณ์ของแถวในความสัมพันธ์ เรียกว่า คีย์ (key)

ฐานข้อมูลหนึ่งๆ จะมีข้อมูลอยู่มากมาย ยิ่งฐานข้อมูลมีขนาดใหญ่ขึ้นก็จะมีข้อมูลจำนวนมากขึ้นเป็นเงาตามตัวข้อมูลเหล่านี้อาจมีค่าแตกต่างกัน คล้ายกัน หรือแม้กระทั่งเหมือนกัน ทำให้การแยกแยะโดยอาศัยเพียงตัวข้อมูลอย่างเดียวทำได้อย่างยากลำบาก ดังนั้นจึงมีการกำหนดค่า Keys ประจำข้อมูลเพื่อทำให้การแยกแยะข้อมูลในฐานข้อมูลเป็นไปอย่างถูกต้อง

คีย์มีหลายประเภท ได้แก่ คีย์หลัก, Secondary key, Foreign key, Candidate key, Super key ดังจะได้กล่าวในรายละเอียดต่อไป

2.1.2.1 **คีย์หลัก** (Primary key)

คีย์หลัก คือ Key หลักที่ใช้ในการอ้างถึง Entity ในฐานข้อมูล การเลือกคีย์หลักสามารถเลือกได้จาก Record ใดๆ ก็ได้ที่ไม่มีโอกาสซ้ำซ้อนกันบนฐานข้อมูลนั้น

เลขประจำตัวประชาชน	ชื่อ	นามสกุล	อายุ
3501552150054	สมชาย	แซ่ตั้ง	25
3210077565107	สมศรี	แซ่อึ้ง	42
4110597520235	สมชาย	แซ่ตั้ง	25
2156800512473	สมปอง	แซ่แต้	16
7812350453784	สมชัย	แซ่เล้ง	50

ตารางที่ 2-3 ข้อมูลทั่วไปของประชาชน

ตารางที่ 1 แสดง Entity ของประชาชนซึ่งประกอบด้วยเลขประจำตัวประชาชน ชื่อ นามสกุล และอายุ โดยจะสามารถเห็นได้ว่า นอกจาก Field เลขประจำตัวประชาชนแล้ว Field อื่นๆ คือชื่อ นามสกุล และ

อายุ อาจซ้ำกันได้ทั้งนั้น ในกรณีนี้ Field ที่เหมาะสมที่จะเป็นคีย์หลักที่สุดก็คือ Field เลขประจำตัวประชาชนนั่นเอง

คีย์หลักเป็นข้อมูลสำคัญที่จะทำให้การเข้าถึงข้อมูลบนฐานข้อมูลเป็นไปได้อย่างรวดเร็ว ดังนั้นผู้ใช้จึงควรกำหนดคีย์หลักให้ชัดเจนตั้งแต่ขั้นตอนออกแบบฐานข้อมูล หากไม่มีข้อมูลใดเลยในฐานข้อมูลที่เหมาะที่จะเป็นคีย์หลักก็ควรที่จะกำหนด Record ใหม่สำหรับให้เป็นคีย์หลักโดยเฉพาะ

ชื่อ	นามสกุล	อายุ	เพศ
สมชาย	แซ่ตั้ง	25	ชาย
สมศรี	แซ่อึ้ง	42	หญิง
สมชาย	แซ่ตั้ง	25	ชาย
สมปอง	แซ่แต้	16	ชาย
สมชัย	แซ่เล้ง	50	ชาย

ตารางที่ 2-4 ข้อมูลทั่วไปของพนักงานบริษัท

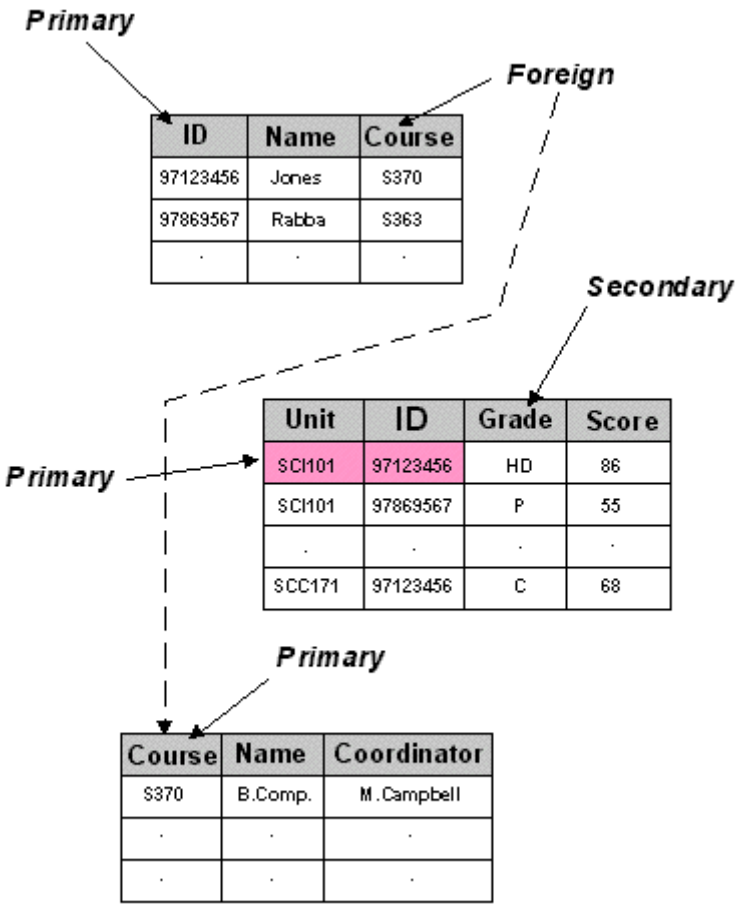
จากตารางที่ 2 จะเห็นได้ว่าฐานข้อมูลพนักงานบริษัทนี้ไม่มี Field ใดเลยที่เหมาะสมที่จะใช้เป็นคีย์หลัก ดังนั้น ผู้ออกแบบฐานข้อมูลจึงควรเพิ่ม Field เฉพาะสำหรับใช้เป็นคีย์หลักของฐานข้อมูลดังตารางที่ 3

ID	ชื่อ	นามสกุล	อายุ	เพศ
1	สมชาย	แซ่ตั้ง	25	ชาย
2	สมศรี	แซ่อึ้ง	42	หญิง
3	สมชาย	แซ่ตั้ง	25	ชาย
4	สมปอง	แซ่แต้	16	ชาย
5	สมชัย	แซ่เล้ง	50	ชาย

ตารางที่ 2-5 ข้อมูลทั่วไปของพนักงานบริษัทหลังจากเพิ่ม คีย์หลัก แล้ว

หลังจากเพิ่ม Field พิเศษคือ ID เข้าไปในตารางที่ 2 เพื่อใช้เป็นคีย์หลักโดยเฉพาะ จะสามารถทำให้การอ้างถึงข้อมูลในฐานข้อมูลเป็นไปได้สะดวกและมีประสิทธิภาพมากขึ้น

โครงสร้างของคีย์



รูป 2-6 โครงสร้างของคีย์

2.1.2.2 **คีย์รอง** (Secondary Key)

คีย์สำรอง คือ คีย์เดี่ยวหรือคีย์ผสม (Single or Composite key) ซึ่งเมื่อใช้ในการค้นหาข้อมูลจากความสัมพันธ์จะได้มากกว่าหนึ่งเรคคอร์ด ต่างจากคีย์หลักที่ทำให้ข้อมูลในตารางไม่ซ้ำกัน ดังนั้นคีย์รองจึงไม่จำเป็นจะต้องเป็นเอกลักษณ์

2.1.2.3 **คีย์นอก** (Foreign key)

คีย์นอก คือ คีย์เดี่ยวหรือคีย์ผสม ซึ่งปรากฏเป็นคีย์ทั่วไปของความสัมพันธ์หนึ่ง แต่ไปปรากฏเป็นอีกคีย์หลักในอีกความสัมพันธ์หนึ่ง คีย์นอกเป็นอีกคีย์หนึ่งที่มีความสำคัญมากในฐานข้อมูลเชิงสัมพันธ์ เนื่องจากเป็นตัวที่ใช้สร้างการเชื่อมต่อระหว่างความสัมพันธ์ การเปลี่ยนแปลงค่าของคีย์นอกจะต้องอาศัยความระมัดระวังเป็นอย่างมากเนื่องจากจะมีผลกระทบโดยตรงต่อข้อมูลในความสัมพันธ์อื่นที่มีการอ้างอิงถึงคีย์นอกตัวนี้ จึงมีกฎและเงื่อนไขที่บังคับใช้เพื่อทำให้ข้อมูลมีความถูกต้องอยู่เสมอ ดังจะกล่าวในตอนที่ 2.3

การกำหนดค่าให้กับคีย์นอกของความสัมพันธ์ที่อ้างอิงถึงจะต้องกำหนดค่าของคีย์ให้อยู่ในโดเมนเดียวกันกับความสัมพันธ์ที่คีย์นอกนั้นเป็นคีย์หลัก แต่คีย์นอกนั้นไม่จำเป็นจะต้องเป็นส่วนหนึ่งในคีย์หลักของความสัมพันธ์อื่น

2.1.2.4 **ซุปเปอร์คีย์** (Superkey)

คือกลุ่มของแอททริบิวที่สามารถนำไปใช้ในการค้นหาข้อมูลที่เป็นเอกลักษณ์ได้

2.1.2.5 **คีย์แข่งขัน** (Candidate key)

คีย์แข่งขัน ก็คือ ซุปเปอร์คีย์ และไม่มีกลุ่มย่อยของคีย์ใดในคีย์แข่งขันที่จะสามารถเป็นซุปเปอร์คีย์ได้อีก

ตอนที่ 2.2 Algebra Rule

หัวเรื่อง

เรื่องที่ 2.2.1 Relational Algebra เบื้องต้น

เรื่องที่ 2.2.2 Relational Algebra Operations พื้นฐาน

เรื่องที่ 2.2.3 Relational Calculus

แนวคิด

ในปัจจุบันฐานข้อมูลเชิงสัมพันธ์เป็นที่รู้จักกันแพร่หลายและเป็นที่นิยมใช้ในการจัดเก็บข้อมูลอย่างมาก แนวความคิดที่สนับสนุนหรืออยู่เบื้องหลังของรูปแบบฐานข้อมูลเชิงสัมพันธ์มาจากรูปแบบทางคณิตศาสตร์ที่เรียกว่า relational algebra และ relational calculus

วัตถุประสงค์

หลังจากศึกษาตอนที่ 2.2 แล้ว นักศึกษาสามารถ

1. บอกถึงรูปแบบและลักษณะที่สำคัญของ relational algebra

เรื่องที่ 2.2.1 พื้นฐาน Relational Algebra

Relational Algebra ใช้ในการจัดการข้อมูลโดยระบุตัวกระทำ (operand) กับความสัมพันธ์ที่ต้องการจัดการ และจะได้ผลลัพธ์ออกมาเป็นความสัมพันธ์ใหม่ Relational Algebra เรียกได้อีกอย่างว่าเป็น Relational query language ที่ประกอบไปด้วยรูปแบบที่ใช้ในการจัดการและค้นหาข้อมูลจากฐานข้อมูลเชิงสัมพันธ์ Query language ในที่นี้ไม่ใช่ Programming language ดังนั้นจึงไม่ได้สนับสนุนการคำนวณที่ซับซ้อนนัก แต่สนับสนุนการเข้าถึงข้อมูลขนาดใหญ่ด้วยวิธีง่ายๆ และมีประสิทธิภาพ

Query language ที่ใช้กับฐานข้อมูลเชิงสัมพันธ์มี 2 รูปแบบ คือ

1. Relational Algebra ประกอบด้วยตัวกระทำต่างๆ ซึ่งมีประโยชน์มากในการจัดการฐานข้อมูลเชิงสัมพันธ์
2. Relational Calculus เป็นภาษาที่อาศัยหลักการทางตรรกะคณิตศาสตร์เพื่อการจัดการข้อมูล

ตัวกระทำกับความสัมพันธ์ใน Relational Algebra แบ่งตามประเภทการใช้งาน แยกได้เป็น 2 ประเภท คือ

1. การใช้งานขั้นพื้นฐาน ได้แก่
   - Selection (σ). เลือกแถวจากความสัมพันธ์
   - Projection (π) เลือกเฉพาะคอลัมน์ที่ต้องการจากความสัมพันธ์
   - Cross- product (X) สามารถรวมความสัมพันธ์ได้
   - Set difference (–) หาความแตกต่างระหว่าง 2 ความสัมพันธ์

- Union ($\cup$) เชื่อมความสัมพันธ์ 2 อันเข้าด้วยกัน

2. การใช้งานขั้นสูง ได้แก่ Intersection, join , division, renaming

ตัวกระทำกับความสัมพันธ์ใน Relational Algebra แบ่งตามการกระทำกับความสัมพันธ์แยกได้เป็น 2 ประเภท คือ

- Unary operators คือ ตัวกระทำที่ต้องการเพียงความสัมพันธ์เดียว เช่น select, project และ rename
- Binary operators คือ ตัวกระทำที่ต้องการสองความสัมพันธ์ เช่น Union, Intersection, Difference และ Cartesian product.

เรื่องที่ 2.2.2 Relational Algebra Operations **พื้นฐาน**

การกระทำเกี่ยวกับ Relational Algebra ขั้นพื้นฐานได้แก่ Select, Project, Composition, Union, Set difference, Cartesian-product และ Rename

2.2.2.1 Selection operator

ตัวกระทำในการ Select คือ σ (sigma) ทำหน้าที่เป็นตัวเลือกข้อมูลความสัมพันธ์จากเงื่อนไขที่กำหนด ผลลัพธ์จาก Select คือแถวในความสัมพันธ์จำนวนหนึ่งซึ่งจะมีจำนวนดีกรีเท่ากับความสัมพันธ์ที่นำมาผ่านการเลือก จำนวนแถวที่เป็นผลลัพธ์อาจจะน้อยกว่าจำนวนแถวทั้งหมดที่มีในความสัมพันธ์แต่จะไม่มากกว่า

เงื่อนไขในการเลือกจะถูกเขียนไว้เป็นตัวห้อยจากเครื่องหมาย σ การระบุเงื่อนไขสามารถทำได้โดยใช้เครื่องหมายในการเปรียบเทียบและตัวกระทำทางตรรกะร่วมกันได้

- เครื่องหมายเปรียบเทียบ ได้แก่ $=$ (เท่ากับ), $\neq$ (ไม่เท่ากับ), $<$ (น้อยกว่า), $\leq$ (น้อยกว่าหรือเท่ากับ), $>$ (มากกว่า), $\geq$ (มากกว่าหรือเท่ากับ)

- ตัวกระทำทางตรรกะ ได้แก่ $\wedge$ (and) , $\vee$ (or) และ $\neg$ (not)

ตัวอย่างเช่น การค้นหาข้อมูลของผู้เล่นที่ส่วนสูงมากกว่า 178 จากความสัมพันธ์ Player ในตาราง 2-1 สามารถเขียนโดยใช้ Selection operator ได้เป็น

$$\sigma_{height > 178} \text{Player}$$

การค้นหาข้อมูลของผู้เล่นที่อายุมากกว่า 25 และเป็นผู้เล่นในตำแหน่งกองหน้า

$$\sigma_{age > 178 \wedge position = \text{กองหน้า}} \text{Player}$$

ผลลัพธ์ที่ได้แสดงดังตาราง 2-3

name	*position*	*age*	*Height*	*weight*
สุเมธ	กองหน้า	26	183	82

ก้องเกียรติ	กองกลาง	24	180	78

ตาราง 2-7 ผลลัพธ์ของ selection : $\sigma_{height > 178}$ Player

การเลือกข้อมูลโดยใช้ σ เทียบได้กับการใช้คำสั่ง SELECT ... FROM ... WHERE ในภาษา SQL

```
SELECT name, position, age, height, weight
FROM Player
WHERE height > 178
```

2.2.2.2 Projection operator

Projection คือ การเลือกเฉพาะบางคอลัมน์ของความสัมพันธ์ที่สนใจขึ้นมาแสดง โดยใช้สัญลักษณ์ π (pi) เขียนในรูปแบบของ $\pi_{attributes} r$ ผลลัพธ์จากการเลือกโดยใช้ π จะได้จำนวนแถวเท่ากับความสัมพันธ์ r แต่จำนวนดีกรีอาจจะเท่ากันหรือน้อยกว่าก็ได้ เทียบได้กับการใช้คำสั่ง SELECT ... FROM ในภาษา SQL ตัวอย่างเช่น การเลือกดูข้อมูลเฉพาะน้ำหนักและส่วนสูงจากความสัมพันธ์ Player ผลลัพธ์จากการ project นี้แสดงได้ดังตาราง 2-4

$\pi_{weight, height}$ Player

```
SELECT weight, height
FROM Player
```

weight	*height*
82	183
78	175
71	169
78	180

ตาราง 2-8 ผลลัพธ์ของการใช้ projection : $\pi_{weight, height}$ Player

2.2.2.3 **การใช้** Selection operator **ร่วมกับ** Projection operator

Selection operator () และ Projection operator () สามารถนำมาใช้ร่วมกันได้เพื่อเลือกข้อมูลจากความสัมพันธ์ที่ต้องการได้ เช่น การเลือก ชื่อ และ ตำแหน่งของผู้เล่นที่ส่วนสูงมากกว่า 178 เทียบได้กับคำสั่ง SQLดังข้างล่าง และผลลัพธ์แสดงดังตาราง 2-5

$\pi_{name, position}$ ($\sigma_{height > 178}$ Player)

```
SELECT name, position
FROM Player
WHERE height > 178
```

Name	*position*
สุเมธ	กองหน้า
ก้องเกียรติ	กองกลาง

ตาราง 2-9 ความสัมพันธ์ที่เกิดจากการใช้ selection และ projection ร่วมกัน

2.2.2.4 Union

เราสามารถกล่าวได้ว่าความสัมพันธ์ R1 และ R2 เป็นความสัมพันธ์ที่สามารถนำทำมา Union กันได้(Union compatible) ก็ต่อเมื่อมีรูปแบบของความสัมพันธ์เหมือนกัน การ Union ความสัมพันธ์ R1 และ R2 สามารถเขียนได้ในรูปของ $R1 \cup R2$

ผลลัพธ์ที่ได้นี้จะมีรูปแบบของเหมือนกับความสัมพันธ์ต้นแบบ R1 และ R2 และประกอบไปด้วยแถว t ใดๆ ที่อยู่ใน R1 หรือ R2 เมื่อพิจารณาการ Union ความสัมพันธ์ Player2000 และ Player2001 ต่อไปนี้ ความสัมพันธ์ที่เป็นผลลัพธ์จะประกอบไปด้วย 4 แถวจากความสัมพันธ์ Player2000 และแถวใหม่ 3 แถวจากความสัมพันธ์ Player2001 แสดงดังตาราง 2-7 โปรดสังเกตว่าแถวที่ซ้ำกันจะปรากฏเพียงครั้งเดียวหลังจากการ Union

Name	*position*	*age*	*height*	*weight*
สุเมธ	กองหน้า	26	183	82
ผลดี	ปีกซ้าย	21	175	78
ชาติชาย	กองหลัง	30	169	71
ก้องเกียรติ	กองกลาง	24	180	78

ความสัมพันธ์ Player2000

Name	*position*	*age*	*height*	*weight*
เอกพจน์	ปีกขวา	20	180	78
สุชาติ	กองหน้า	19	179	80
ชาติชาย	กองหลัง	30	169	71
พิเชษฐ์	ปีกซ้าย	22	177	76

ความสัมพันธ์ Player2001

ตาราง 2-10 ความสัมพันธ์ Player2000 และ Player2001

Name	*position*	*age*	*height*	*weight*
สุเมธ	กองหน้า	26	183	82
ผลดี	ปีกซ้าย	21	175	78
ชาติชาย	กองหลัง	30	169	71
ก้องเกียรติ	กองกลาง	24	180	78
เอกพจน์	ปีกขวา	20	180	78
สุชาติ	กองหน้า	19	179	80
พิเชษฐ์	ปีกซ้าย	22	177	76

ตาราง 2-11 ความสัมพันธ์ผลลัพธ์ของ Player2000 ∪ Player2001

การ Union โดยใช้คำสั่ง SQL สามารถกระทำได้โดยการใช้คำสั่ง UNION เพื่อเชื่อมระหว่างคำสั่ง SELECT

```
SELECT name, position, age, height, weight
FROM Player2000
UNION
SELECT name, position, age, height, weight
FROM Player2001
```

2.2.2.5 Difference

เราสามารถกล่าวได้ความสัมพันธ์ R1 และ R2 เป็นความสัมพันธ์ที่มีความแตกต่าง (Difference Compatible) ก็ต่อเมื่อ R1 และ R2 มีรูปแบบของความสัมพันธ์เหมือนกัน ความแตกต่างของความสัมพันธ์ R1 และ R2 หมายความว่า พบใน R1 และไม่พบใน R2 เขียนได้ในรูป R1 – R2 ตัวอย่าง การหาความแตกต่างของความสัมพันธ์ Player2000 และ Player 2001 แสดงดังตาราง 2-8

Name	*position*	*age*	*height*	*weight*

สุเมธ	กองหน้า	26	183	82
ผลดี	ปีกซ้าย	21	175	78
ก้องเกียรติ	กองกลาง	24	180	78

ตาราง 2-12 ความสัมพันธ์ผลลัพธ์ของ Player2000 – Player2001

การหาความแตกต่างโดยใช้คำสั่ง SQL สามารถกระทำได้โดยการใช้คำสั่ง MINUS ระหว่างคำสั่ง SELECT

```
SELECT name, position, age, height, weight
    FROM Player2000
MINUS
SELECT name, position, age, height, weight
    FROM Player2001
```

2.2.2.6 Intersection

การทำ Intersection ระหว่างความสัมพันธ์ R1 และ R2 ผลลัพธ์ที่ได้จะเป็นความสัมพันธ์ที่เกิดจากแถวที่ซ้ำกันใน R1 และ R2 แถวใดที่ไม่ปรากฏใน R1 และ R2 เหมือนกันจะไม่อยู่ในความสัมพันธ์ผลลัพธ์ สามารถเขียนได้ในรูป R1 ∩ R2 ตัวอย่าง การทำ Intersection ระหว่างความสัมพันธ์ Player2000 และ Player 2001 จะได้ผลลัพธ์ดังตาราง 2-9 และการทำ Intersection ใน SQL ก็คล้ายกับตัวอย่างการทำ Union และ Difference ข้างต้น เพียงแต่ใช้คำสั่ง INTERSECT คั่นระหว่างคำสั่ง SELECT

Name	*position*	*Age*	*height*	*weight*
ชาติชาย	กองหลัง	30	169	71

ตาราง 2-13 ความสัมพันธ์ผลลัพธ์ของ Player2000 ∩ Player2001

การหาความแตกต่างโดยใช้คำสั่ง SQL สามารถกระทำได้โดยการใช้คำสั่ง MINUS ระหว่างคำสั่ง SELECT

```
SELECT name, position, age, height, weight
FROM Player2000
INTERSECT
SELECT name, position, age, height, weight
```

FROM Player2001

2.2.2.9 Division

การทำ Division ไม่ค่อยถูกใช้นักใน Relation Algebra แต่มีประโยชน์ในการค้นหาบางประเภท รูปแบบของ Division คือ R1/R2 เมื่อ R1 และ R2 เป็นความสัมพันธ์ ตัวอย่าง การค้นหาลูกจ้างที่ทำงานอยู่สองแผนกคือ CS และ Fin จากความสัมพันธ์ DEPTNAME ผลปรากฏว่า สิทธิ คือคนทำงานอยู่ทั้งสองแผนก สังเกตผลลัพธ์จากตาราง 2- 13

DEPTNAME

DName
CS
Fin

EMPDEPT

Name	Dept
สิทธิ	CS
โจ	Econ
เขียว	Econ
ตาล	CS
สิทธิ	Fin

EMPDEPT/DEPTNA

สิทธิ

รูป 2-14 EMPDEPT / DEPTNAME

เรื่องที่ 2.2.3 Relational Calculus

เมื่อผู้ใช้ทำการค้นหาข้อมูลโดยใช้ Relational algebra ผู้ใช้จะต้องกำหนดการกระทำและเงื่อนไขเพื่อให้ได้ข้อมูลตามที่ต้องการ จึงกล่าวได้ว่า Relational algebra เป็น procedural language เช่น การ Join แล้วตามด้วย Projection ด้วยการใช้ Relational calculus ผู้ใช้สามารถกำหนดรูปแบบการค้นหาในลักษณะของนิพจน์หรือสมการทางคณิตศาสตร์ที่มีตัวแปร ค่าคงที่ ตัวกระทำ ตัวเชื่อม และอื่นๆ Relational calculus จึงเป็นการใช้คณิตศาสตร์ในรูปของตรรกะเข้ามาช่วยในการค้นหาข้อมูล คำตอบที่ได้จากการใช้ Relational calculus ถือแถวของข้อมูลจากความสัมพันธ์ที่ทำให้ค่าของสมการคณิตศาสตร์นั้นมีค่าเป็น จริง

เมื่อ R1, ..., Rk เป็นชุดของความสัมพันธ์ และความสัมพันธ์ Ri ประกอบไปด้วยแอททริบิวจำนวน n ตัว (A1,...,An) แล้ว Relational Calculus จะประกอบด้วยสมการทางตรรกะที่สร้างได้โดยการใช้

สัญลักษณ์แทนเงื่อนไข : $\in$, =, <, <=, >, >= , $\neq$

รูปแบบของคิวรี

$$\{ (A1, ...,An) \mid p[(A1,...,An)] \}$$

คำตอบที่ได้จากคิวรี ประกอบไปด้วยแถวของข้อมูล (A1,...,An) ที่ทำให้สมการ p [(A1,...,An)] มีค่าเป็นจริง

สมการมีอยู่ 2 แบบ คือ

สมการเดี่ยว (Atomic formula) อยู่ในรูป p [(A1,...,An)]

สมการผสม (Formula) อยู่ในรูปของสมการเดี่ยวผสมกับสมการเดี่ยวอื่น โดยอาศัยตัวกระทำกับสมการ : $\neg$ (not) , $\wedge$ (intersection), $\vee$ (union), $\forall$ (for all) , $\exists$ (there exists)

ตัวอย่าง สมการต่อไปนี้หมายความว่า ให้หาข้อมูล t_1 ทุกแถวในความสัมพันธ์ EmpDept (รูปจากตาราง 2-10) ที่ทำให้มีบางแถวใน t2 ซึ่ง t2 อยู่ในความสัมพันธ์ Emp (รูปจากตาราง 2-11) และทั้ง t1 และ t2 มีค่าในคอลัมน์ Dept เหมือนกัน

$$\{t_1 \mid (\exists t_2)((t_1 \in \text{EmpDept}) \& (t_2 \in \text{Emp}) \& (t_1.\text{Dept} = t_2.\text{Dept})\}$$

ตอนที่ 2.3 Constraints

หัวเรื่อง

เรื่องที่ 2.3.1 Constraints

แนวคิด

รูปแบบฐานข้อมูลเชิงสัมพันธ์ ผู้ใช้สามารถเลือกข้อมูลที่ต้องการดูได้โดยผู้ใช้สามารถใส่เงื่อนไขบังคับ (constraints) เพื่อได้คำตอบที่ต้องการ

วัตถุประสงค์

หลังจากศึกษาตอนที่ 2.3 แล้ว นักศึกษาสามารถ

เข้าใจหลักการและทฤษฎีทางคณิตศาสตร์ที่อยู่เบื้องหลังของรูปแบบฐานข้อมูลเชิงสัมพันธ์และการกระทำกับข้อมูลที่ต้องการที่ถูกจัดเก็บในฐานข้อมูล

เรื่องที่ 2.3.1 Constraints

Constraints คือ ข้อบังคับหรือเงื่อนไขในการอนุญาติให้เก็บเฉพาะข้อมูลที่เหมาะสมลงในฐานข้อมูล เพื่อให้การเลือกข้อมูลจากฐานข้อมูลเป็นไปอย่างถูกต้อง ประโยชน์อย่างอื่นของการมีเงื่อนไข ได้แก่

1. เพื่อตรวจสอบข้อผิดพลาดในการกรอกข้อมูลลงในฐานข้อมูล
2. เพื่อความถูกต้องในการปรับปรุงข้อมูล
3. เพื่อรักษาความถูกต้องของข้อมูลโดยรวม
4. เพื่อบอกให้ฐานข้อมูลทราบว่าผู้ใช้ต้องการจะเก็บข้อมูลหรือค้นหาข้อมูล

รูปแบบของการแสดงฐานข้อมูล เช่น E/R model เงื่อนไขที่สามารถแสดงได้มีเพียงการประกาศคีย์ชนิดความสัมพันธ์ ลักษณะการเป็นสมาชิก และความซ้ำซ้อนเท่านั้น ต่างจาก Relational model หรือ SQL ที่สามารถรองรับเงื่อนไขที่มากกว่านี้ได้ คือ

- Key constraint
- Not Null constraint
- Referential Integrity constraint
- Check
- Assertion constraint

ในการประกาศและบังคับใช้เงื่อนไขเพื่อให้มีผลกับฐานข้อมูล สามารถทำได้โดยการประกาศไว้ในรูปแบบของความสัมพันธ์ หรือประกาศภายหลัง ซึ่งจะมีผลใช้กับฐานข้อมูลที่ใช้อยู่ปัจจุบัน เมื่อประกาศเงื่อนไขบังคับไปแล้ว และมีเหตุการณ์ใดที่ทำให้ผิดเงื่อนไข เช่น คำสั่ง SQL ระบบจัดการฐานข้อมูลก็จะต้อง

ยกเลิกเหตุการณ์หรือคำสั่งนั้นและแจ้งข้อผิดพลาดให้ผู้ใช้ทราบทันที ตัวอย่างที่จะกล่าวต่อไปจะอ้างอิงโดยใช้คำสั่ง SQL ซึ่งเป็นคำสั่งที่นิยมใช้มากในการค้นหาข้อมูลจากฐานข้อมูล

Key constraint

ในฐานข้อมูลเชิงสัมพันธ์มีเงื่อนไขเกี่ยวกับคีย์อยู่ 2 แบบ คือ

คีย์หลัก เมื่อมีการให้ค่าแก่ คีย์หลัก ในความสัมพันธ์ใดๆ ก็แล้วแต่ ค่าของคีย์ชนิดนี้จะไม่เป็น Null เสมอ (Not Null) และจะถูกใช้เป็นดัชนีในการเรียงลำดับเสมอ เมื่อผู้ใช้ค้นหาข้อมูลจากความสัมพันธ์โดยใส่ค่า คีย์หลัก เป็น Null ก็จะไม่พบข้อมูลที่ต้องการ ดังนั้นผู้ใช้จึงจำเป็นจะต้องทราบถึงเงื่อนไขนี้ เพื่อให้ได้ข้อมูลที่ถูกต้องจากการค้นหา

การกำหนดคีย์ใน SQL สามารถทำได้สองแบบ คือ การกำหนดให้แอททริบิวแต่ละตัว หรือกำหนดไว้ตั้งแต่ต้นเมื่อสร้างความสัมพันธ์ ดังตัวอย่าง

```
CREATE TABLE Emp (EmpID integer คีย์หลัก,
    Name char(40),
    Address char(80)
    Dept char(3)
    Office integer,
    Phone char(10)
    UNIQUE (name,address))
```

Unique ความสัมพันธ์ใดๆ สามารถมีแอททริบิวที่เป็นเอกลักษณ์ได้มากกว่าหนึ่งนอกจาก คีย์หลัก โดยการกำหนดให้แอททริบิวนั้นเป็น Unique และทุกแอททริบิวที่เป็น Unique นั้นจะถูกใช้เป็นดัชนีในการเรียงรายการในความสัมพันธ์นั้นโดยอัตโนมัติ ต่อจาก คีย์หลัก ตัวอย่างดังการสร้างตารางในภาษา SQL ข้างบน

Referential Integrity

การอ้างอิงถึงข้อมูลจากความสัมพันธ์อื่น มีเงื่อนไข คือ

แอททริบิวที่อ้างอิงมาจากความสัมพันธ์อื่นจะต้องเป็น คีย์หลัก เสมอ เพื่อป้องกันไม่ให้เกิดการซ้ำซ้อนของข้อมูลที่อ้างอิงได้

แอททริบิวที่ถูกอ้างอิงและนำมาอยู่ในความสัมพันธ์จะถูกเรียกว่า Foreign key

ในการบังคับใช้ Referential Integrity จาก SQL สามารถประกาศไว้กับตัวแอททริบิวโดยตรงหรือประกาศแยกไว้ในความสัมพันธ์ที่ต้องการอ้างอิงถึงแอททริบิวตัวนั้น

```
CREATE TABLE Apply(ID integer REFERENCES Emp(EmpID),
    Location char(20),
```

Dept char(3),

FOREIGN KEY (location) REFERENCES Building(location))

Check

การตรวจสอบค่าของข้อมูลในความสัมพันธ์เป็นเงื่อนไขอย่างหนึ่งที่จะต้องทำตาม เพื่อกรองเฉพาะข้อมูลที่เหมาะสมลงไปเก็บในฐานข้อมูล คำสั่ง CHECK ในภาษา SQL สามารถใช้ร่วมกับคำสั่งสร้างตารางความสัมพันธ์ CREATE TABLE ใน SQL เพื่อตรวจสอบการรับข้อมูลว่าสมควรจะเก็บข้อมูลแถวนั้นในตารางหรือไม่ ยกตัวอย่างเช่น ความสัมพันธ์ Emp มีการเพิ่มอายุของพนักงาน จึงต้องมีการตรวจสอบว่าพนักงานมีอายุเกิน 20 ปีหรือไม่

CREATE TABLE Emp (EmpID integer คีย์หลัก,

Name char(40),

Age integer,

Address char(80)

Dept char(3)

Office integer,

Phone char(10)

CHECK (age > 20))

Assertion

Assertion คือ เงื่อนไขที่ใช้ทั่วไปในการตรวจสอบความถูกต้องโดยรวมของทุกความสัมพันธ์หรือทุกฐานข้อมูล ซึ่งสามารถใส่เป็นเงื่อนไข เช่น การเปรียบเทียบได้โดยตรง การสร้าง Assertion โดยใช้ SQL สามารถทำได้โดยใช้คำสั่งในรูปแบบต่อไปนี้

CREATE ASSERTION <name> CHECK (<condition>)

อาศัยตัวอย่างความสัมพันธ์ Emp ที่ปรับปรุงแล้วข้างต้น

CREATE ASSERTION ageover CHECK (NOT EXISTS (SELECT age FROM Emp) > 20)

Assertion ที่สร้างไว้จะเป็นเงื่อนไขที่ทำให้ระบบจัดการฐานข้อมูลคอยเช็คว่าเกิดการผิดเงื่อนไขตามที่กำหนดไว้ใน Assertion หรือไม่ ถ้าเกิดการผิดพลาดระบบจัดการฐานข้อมูลก็จะรายงานต่อผู้ใช้ด้วยชื่อของ Assertion ที่สร้างไว้ทันที

ตอนที่ 2.4 Cartesian Rule

หัวเรื่อง

เรื่องที่ 2.4.1 Cartesian Product

เรื่องที่ 2.4.2 Join

แนวคิด

1.ผู้ใช้สามารถนำข้อมูลจากหลายๆตารางมารวมกันได้ตามที่ต้องการโดยอาศัยกฎของความสัมพันธ์มาจากรูปแบบทางคณิตศาสตร์เข้ามาช่วย คือ Cartesian Rule

วัตถุประสงค์

หลังจากศึกษาตอนที่ 2.4 แล้ว นักศึกษาสามารถ

1. เข้าใจหลักการและทฤษฎีทางคณิตศาสตร์ที่อยู่เบื้องหลังของรูปแบบฐานข้อมูลเชิงสัมพันธ์และการกระทำกับข้อมูลที่ต้องการที่ถูกจัดเก็บในฐานข้อมูล

เรื่องที่ 2.4.1 Cartesian Product

พิจารณาความสัมพันธ์ R1 และ R2 ที่ต่างกัน R1(A1,A2,...,An) และ R2(A1',A2',...,Am') เมื่อต้องการจะเชื่อมเข้าด้วยกันจะต้องทำ Cartesian Product ของทั้งสองความสัมพันธ์ ซึ่งสามารถเขียนได้เป็น R1 X R2 ที่มีผลลัพธ์ในรูปแบบความสัมพันธ์ใหม่ (A1,A2,...,Am, A1', A2',..., Am') และประกอบไปด้วยแถวทั้งหมดจากทั้งสองความสัมพันธ์ (t1,..., tn, t1',..., tm') โดยที่ (t1,..., tn) อยู่ใน R1 และ (t1,...,tm') อยู่ใน R2 ความสัมพันธ์ในลักษณะของ Cartesian product จะถูกนำไปใช้ในการ Join ซึ่งเราจะได้เรียนต่อไป

การทำ Cartesian Product แท้ที่จริงแล้วคือการนำแถวจาก R1 มาต่อกับแถวจาก R2 ทำเช่นนี้ไปทีละแถว จนกระทั่งได้ผลลัพธ์ทั้งหมด ดังนั้นจำนวนแถวของความสัมพันธ์ผลลัพธ์จะเท่ากับจำนวนแถวของ R1 คูณด้วยจำนวนแถวของ R2 ตัวอย่าง ตาราง 2-10 แสดงความสัมพันธ์ R, S และ R X S

R

First	Last	Age
บิณฑ์	บันลือ	22
มะลิ	หอมชื่นใจ	23
โตน	ต้นตาล	32

S

Dinner	Dessert
ข้าวเหนียว	เฉาก๊วย
อาหารทะเล	รวมมิตร
อาหารทะเล	รวมมิตร

R X S

First	Last	Age	ข	Dessert
บิณฑ์	บันลือ	22	ข้าวเหนียว	เฉาก๊วย
บิณฑ์	บันลือ	22	อาหารทะเล	รวมมิตร
มะลิ	หอมชื่นใจ	23	ข้าวเหนียว	เฉาก๊วย

มะลิ	หอมชื่นใจ	23	อาหารทะเล	รวมมิตร
โตน	ต้นตาล	32	ข้าวเหนียว	เฉาก๊วย
โตน	ต้นตาล	32	อาหารทะเล	รวมมิตร

ตาราง 2-15 ความสัมพันธ์ R, S และ R X S

เรื่องที่ 2.4.2 Join

การรวมข้อมูลจากหลายความสัมพันธ์เข้าด้วยกัน เรียกว่า Join ซึ่งต่างจาก Projection และ Selection จากข้างต้นที่กระทำบนความสัมพันธ์เดียว Join จะรวมข้อมูลเข้าด้วยกันโดยอาศัยเงื่อนไขทางตรรกะในรูปแบบ $R \bowtie_C S$ โดยที่ R และ S เป็นความสัมพันธ์ C เป็นเงื่อนไข ตัวอย่างเช่น การ Join ความสัมพันธ์ EMP และ DEPTINFO เข้าด้วยกัน แสดงดังตาราง 2-11

EMP

Name	Office	Dept	Salary
สิทธิ	400	CS	45000
โจ	220	Econ	35000
เขียว	160	Econ	50000
ตาล	420	CS	65000
สิทธิ	500	Fin	60000

DEPT

Dept	MainOffice	Phone
CS	404	555-1212
Econ	200	555-1234
Fin	501	555-4321
Hist	100	555-9876

ตาราง 2-11 ความสัมพันธ์ EMP และ ความสัมพันธ์ DEPT

เราสามารถใช้ Join เพื่อค้นหารายละเอียดเกี่ยวกับที่ทำงานของพนักงานทุกคนได้โดยเขียนในรูปแบบ $EMP \bowtie_{DEPT=DEPT} DEPTINFO$ ผลลัพธ์จากการ Join นี้แสดงดังตาราง 2-12

EMP

Name	Office	Dept	Salary	Dept	MainOffice	Phone
สิทธิ	400	CS	45000	CS	404	555-1212
โจ	220	Econ	35000	Econ	200	555-1234
เขียว	160	Econ	50000	Econ	200	555-1234
ตาล	420	CS	65000	CS	404	555-1212
สิทธิ	500	Fin	60000	Fin	501	555-4321

ตาราง 2-16 ผลลัพธ์จากการ Join ความสัมพันธ์ EMP และ DEPTINFO โดยใช้เงื่อนไข DEPT=DEPT

ลักษณะการ Join มีหลายแบบ ได้แก่

Theta-Join คือ การ Join แบบปกติซึ่งทำให้เกิดชื่อคอลัมน์ที่ซ้ำกัน

Equi-Join คือ การ Join แบบที่เงื่อนไขเท่ากับ = เท่านั้น และผลลัพธ์ที่ได้คอลัมน์ที่ซ้ำกันจะถูกแสดงเพียงแค่ครั้งเดียว

Natural-Join คือ Equi-Join ที่ Join ทุกคอลัมน์ที่ซ้ำกัน