ดร.โปรดปราน บุณยพุกกณะ

พื้นฐานการออกแบบฐานข้อมูลเชิงสัมพันธ์

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

โปรดปราน บุณยพุกกณะ

พื้นฐานการออกแบบฐานข้อมูลเชิงสัมพันธ์ / โปรดปราน บุณยพุกกณะ

1. ระบบการจัดการฐานข้อมูล
2. ฐานข้อมูลเชิงสัมพันธ์

พิมพ์ครั้งที่ 1 จำนวน 100 เล่ม มกราคม 2560

จัดพิมพ์โดย
โปรดปราน บุณยพุกกณะ
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
จุฬาลงกรณ์มหาวิทยาลัย
พญาไท กรุงเทพฯ 10330

ออกแบบปก : กวิน เมศร์ศิริตระกูล
ออกแบบรูปเล่ม : โปรดปราน บุณยพุกกณะ
ภาพประกอบ : ชลธร ขวัญขจรเกียรติ และ กวิน เมศร์ศิริตระกูล

พิมพ์ที่ บริษัท นีโอ ดิจิตอล จำกัด เลขที่ 666 ซ.สาธุประดิษฐ์ 58 แยก 22 (ประสานใจ)
แขวงบางโพงพาง เขตยานนาวา กรุงเทพฯ 10120 โทรศัพท์ 0-2683-8388

คำนำ

การออกแบบฐานข้อมูลเป็นทักษะที่สำคัญสำหรับวิศวกรคอมพิวเตอร์ เพราะองค์กรไม่ว่าขนาดเล็กหรือใหญ่ล้วนจำเป็นต้องใช้ฐานข้อมูลในการดำเนินธุรกิจหรือกิจกรรมใด ๆ ที่มีข้อมูลจำนวนมาก หากออกแบบได้ไม่เหมาะสมจะส่งผลให้การทำงานของระบบฐานข้อมูลนั้นล้มเหลวได้ง่าย

ตำราพื้นฐานการออกแบบฐานข้อมูลเชิงสัมพันธ์นี้ เป็นการรวบรวมความรู้พื้นฐาน แนวคิด เครื่องมือ วิธีการ และตัวอย่างการออกแบบฐานข้อมูลเชิงสัมพันธ์ เพื่อเป็นตำราประกอบรายวิชา 2110422 การออกแบบระบบการจัดการฐานข้อมูล (Database Management Systems Design) สำหรับนิสิตปริญญาตรี ของภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เนื้อหาครอบคลุมคำอธิบายรายวิชา แต่อาจมีศัพท์เปลี่ยนแปลงไปบ้างเพื่อให้เหมาะสมและสอดคล้องกับความเปลี่ยนแปลงของเทคโนโลยีฐานข้อมูลที่มีการพัฒนาอย่างรวดเร็ว

ตำรานี้เน้นภาคทฤษฎีและมีตัวอย่างให้พอฝึกฝนปฏิบัติได้ ผู้เขียนเชื่อว่าได้ถ่ายทอดความรู้เพียงพอต่อการศึกษาระดับปริญญาบัณฑิตด้านวิศวกรรมคอมพิวเตอร์ ที่ต้องเข้าใจการทำงานและการคำนวณภายในฐานข้อมูล จึงจะออกแบบฐานข้อมูลได้อย่างมีประสิทธิภาพ และเชื่อว่าจะมีพื้นฐานที่เข้มแข็งพอที่จะพัฒนาองค์ความรู้ใหม่ต่อไปในระดับบัณฑิตศึกษาได้ ความรู้ในตำรานี้เป็นพื้นฐานสำหรับวิศวกรคอมพิวเตอร์ที่จะประกอบวิชาชีพในองค์กรที่ใช้ฐานข้อมูลโดยเฉพาะฐานข้อมูลเชิงสัมพันธ์

สุดท้ายนี้ ผู้เขียนขอขอบคุณครอบครัวที่ให้เวลาและกำลังใจ ขอบคุณลูกศิษย์ทั้งหลาย โดยเฉพาะ กฤตย์ กังวาลพงศ์พันธุ์ ชลธร ขวัญขจรเกียรติ กวิน เมศร์ศิริตระกูล สัญชัย จักรธีรังกูร เมสินี นาคมณี วิสิฏฐ์ วงศ์ชัยอนุกูล ธัญนุช ฉันทานุรักษ์ และสุภัชชา เหมรัตนธร ที่เป็นแรงสำคัญผลักดันให้ตำราเล่มนี้สำเร็จลงได้

โปรดปราน บุณยพุกกณะ

มกราคม 2560

สารบัญ

สารบัญภาพ

สารบัญตาราง

บทที่ 1
ฐานข้อมูลเบื้องต้น

วัตถุประสงค์

1. เพื่อให้ทราบนิยามของฐานข้อมูล
2. เพื่อให้ทราบคุณสมบัติของฐานข้อมูล
3. เพื่อให้เข้าใจความแตกต่างระหว่างฐานข้อมูล ระบบการจัดการฐานข้อมูล และระบบฐานข้อมูล
4. เพื่อให้ทราบประวัติความเป็นมาของฐานข้อมูล

ข้อมูลเป็นสิ่งที่อยู่รอบตัวเรา มีอยู่มากมาย และต้องจัดเก็บอย่างมีระเบียบ เพื่อให้นำไปใช้งานได้อย่างมีประสิทธิภาพ ทำได้ด้วยการนำข้อมูลที่สนใจมาออกแบบแล้วจัดเก็บเป็นฐานข้อมูลตามเรื่องนั้น ๆ อย่างเหมาะสม ขอยกตัวอย่างระบบฐานข้อมูลทะเบียนนิสิตจุฬาฯ หรือที่นิสิตเรียกอย่างติดปากว่า เร็กจุฬาฯ (Reg Chula) เป็นสิ่งที่ใกล้ตัวและสำคัญต่อนิสิตจุฬาฯ ทุกคน เพราะเร็กจุฬาฯ เป็นฐานข้อมูลที่เก็บข้อมูลของนิสิตไว้เป็นจำนวนมาก ทั้งชื่อนามสกุล วิชาที่ลงเรียน ชื่ออาจารย์ที่ปรึกษา ชื่อผู้ปกครอง สถานที่ติดต่อ โรงเรียนมัธยมฯ ที่จบการศึกษามา หรือผลการเรียนสมัยมัธยมฯ แล้วผู้เรียนคิดว่าเร็กจุฬาฯ เป็นฐานข้อมูลที่ใหญ่ที่สุดในมหาวิทยาลัยหรือไม่ ปัจจุบันจุฬาฯ มีนิสิตประมาณ 40,000 คน และมีนิสิตจบการศึกษาเกือบ 10,000 คนต่อปี ยิ่งไปกว่านั้นจุฬาฯ ยังถูกก่อตั้งมานานกว่าร้อยปีแล้ว ดังนั้นฐานข้อมูลของเร็กจุฬาฯ จึงมีขนาดใหญ่มาก แต่ว่าขนาดฐานข้อมูลของเร็กจุฬาฯ ที่ว่าใหญ่นั้น เมื่อนำไปเทียบกับระบบฐานข้อมูลประชากร ระบบฐานข้อมูลของธนาคาร ระบบฐานข้อมูลบริษัทโทรคมนาคมแล้ว อาจพบว่าข้อมูลในเร็กจุฬาฯ อาจดูมีขนาดเล็กกว่ามาก เช่น ข้อมูลของบริษัทโทรคมนาคมนั้นจะต้องมีการจัดเก็บล็อก (log) การโทรศัพท์ทั้งหมด เมื่อข้อมูลที่เก็บมีจำนวนมากขึ้น ฐานข้อมูลก็จะมีขนาดใหญ่และซับซ้อนตามกันไป

ในบทนี้ ผู้เรียนจะได้ทราบนิยามของฐานข้อมูล คุณสมบัติของฐานข้อมูล ความแตกต่างระหว่างฐานข้อมูล ระบบฐานข้อมูล ระบบการจัดการฐานข้อมูล และประวัติความเป็นมาของฐานข้อมูล ตำราเล่มนี้อิงมาจากหนังสือ Fundamentals of Database Systems ของศาสตราจารย์เอลมาศรีและศาสตราจารย์นาวาเธ [1-3]

1.1 ฐานข้อมูล ระบบการจัดการฐานข้อมูล และระบบฐานข้อมูล

วิชาเกี่ยวกับการออกแบบฐานข้อมูลมีมาช้านาน เพราะเป็นศาสตร์สำคัญในวงการวิศวกรรมคอมพิวเตอร์ วิทยาศาสตร์คอมพิวเตอร์ และเทคโนโลยีสารสนเทศ และยังเกี่ยวข้องกับหลักสูตรทางธุรกิจ เพราะธุรกิจส่วนใหญ่มีความจำเป็นต้องใช้ฐานข้อมูล ดังนั้นจึงมีผู้นิยามคำว่าฐานข้อมูลไว้มากมาย นิยามที่ชัดเจนมาจากปรมาจารย์ผู้เขียนหนังสือวิชาระบบการจัดการฐานข้อมูลที่ใช้เรียนกันอย่างกว้างขวาง [1-3] กล่าวไว้ว่า ฐานข้อมูล (database) คือ “กลุ่ม (collection) ข้อมูลที่เป็นตัวแทนสถานการณ์จริงในมุมที่สนใจ ข้อมูลเหล่านั้นต้องมีความสอดคล้องกัน (coherent) และมีความหมายในตัวเอง (inherent meaning) นอกจากนี้ฐานข้อมูลจะต้องได้รับการออกแบบและสร้างเพื่อใช้บันทึกข้อมูลตามวัตถุประสงค์นั้น” ฐานข้อมูลอาจมีขนาดเท่าใดก็ได้และอาจมีความซับซ้อนไม่เท่ากัน ส่วนข้อมูล (data) คือข้อเท็จจริงที่สามารถบันทึกได้และมีความหมายโดยปริยาย

อาจกล่าวโดยสรุปได้ว่า ฐานข้อมูล คือกลุ่มของข้อมูลที่เกี่ยวข้องกัน เช่น ฐานข้อมูลร้านอาหาร อาจบันทึกข้อมูลตัวร้าน เมนูอาหาร จำนวนโต๊ะและที่นั่ง ราคาอาหาร พนักงานในร้าน แต่ไม่ควรมีข้อมูลบัญชีธนาคารของเจ้าของร้าน หรือข้อมูลการท่องเที่ยวของเจ้าของร้าน เป็นต้น กล่าวคือ ข้อมูลในฐานข้อมูลหนึ่ง ๆ ถ้าไม่มีความสัมพันธ์ซึ่งกันและกันก็ไม่ควรอยู่ในฐานข้อมูลเดียวกัน

ในฐานข้อมูลหนึ่ง ๆ อาจบันทึกข้อมูลได้หลากหลายชนิด นอกเหนือจากข้อมูลที่เป็นข้อความ (text) หรือตัวเลข (numeric) ที่เห็นกันทั่วไปแล้ว ยังมีข้อมูลชนิดอื่น เช่น ข้อมูลมัลติมีเดีย (multimedia) อันหมายถึงรูปภาพ คลิปวีดิทัศน์ (video clip) เสียง (voice) ข้อมูลทางภูมิศาสตร์หรือเชิงพื้นที่ (spatial data) เช่น ข้อมูลภูมิอากาศ หรือภาพถ่ายดาวเทียม และอื่น ๆ อย่างไรก็ตาม ฐานข้อมูลแต่ละฐานควรเก็บข้อมูลที่เกี่ยวข้องกันอย่างชัดเจน

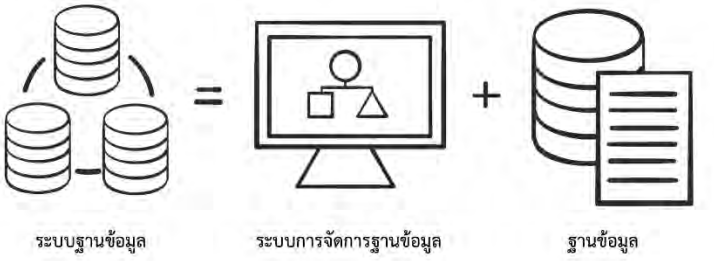
ส่วนการประมวลผลข้อมูลในฐานข้อมูลนั้นจะอาศัย ระบบการจัดการฐานข้อมูล (database management system: DBMS) [1] ซึ่งเป็นซอฟต์แวร์ที่มีความสามารถในการสร้างฐานข้อมูล จัดการฐานข้อมูล บำรุงรักษาข้อมูลและฐานข้อมูลในคอมพิวเตอร์ การจัดการฐานข้อมูลนั้นหมายรวมทั้งการบันทึก การสืบค้น การคำนวณ การปรับปรุงข้อมูล การจัดการผู้ใช้งาน การดูแลความปลอดภัยและอื่น ๆ ส่วนการบำรุงรักษานั้น มีตัวอย่างคือ การสำรองข้อมูล และการกู้คืน เป็นต้น

ซอฟต์แวร์ประเภทระบบการจัดการฐานข้อมูลนั้นมีผู้พัฒนาอยู่หลายบริษัท เช่น ออราเคิล (Oracle) หรือมายเอสคิวแอล (MySQL) ที่ใช้กันอย่างแพร่หลายในองค์กรต่าง ๆ [1] อย่างไรก็ตาม หน่วยงานต่าง ๆ จะต้องพิจารณาเลือกใช้ผลิตภัณฑ์ที่เหมาะกับงานของตน โดยในเบื้องต้น สามารถเลือกได้จากลักษณะของข้อมูลที่แตกต่างกัน ซึ่งจะต้องมีการออกแบบและจัดเก็บต่างกันไป ดังนั้นจำเป็นต้องมีเครื่องมือการจัดเก็บที่เฉพาะเจาะจงด้วย เช่น หากมีข้อมูลแผนที่เป็นหลัก ก็ควรพิจารณาระบบการจัดการฐานข้อมูลประเภทระบบการจัดการฐานข้อมูลเชิงพื้นที่เป็นต้น

นอกจากจะเลือกผลิตภัณฑ์ระบบการจัดการฐานข้อมูลตามลักษณะข้อมูลแล้ว ผู้ใช้ยังควรพิจารณาความรวดเร็วในการตอบสนองของระบบด้วย เช่น ระบบที่ต้องการการทำงานแบบเวลาจริง (real-time) จำเป็นต้องใช้ระบบการจัดการฐานข้อมูลที่ออกแบบมาโดยเฉพาะ เช่น งานควบคุมเครื่องจักรหรือยานพาหนะที่ต้องกระทำการ (execute) แบบทันที

ส่วนคำว่า ระบบฐานข้อมูล (database system) [1] นั้น หมายถึงซอฟต์แวร์ระบบการจัดการฐานข้อมูล (DBMS) พร้อมกับฐาข้อมูล ซึ่งรวมถึงแอปพลิเคชันด้วย ดังนั้นเร็กจุฬาฯ ที่กล่าวมาก่อนหน้านี้ ควรเรียกให้ถูกต้องว่า ระบบฐานข้อมูลทะเบียนนิสิตจุฬาฯ

จะเห็นได้ว่าระบบฐานข้อมูลหนึ่ง จะประกอบด้วยระบบการจัดการฐานข้อมูลหนึ่งระบบกับฐานข้อมูลหลายฐาน รูปที่ 1 แสดงความสัมพันธ์ระหว่างฐานข้อมูล ระบบการจัดการฐานข้อมูล และระบบฐานข้อมูล



รูปที่ 1 ความสัมพันธ์ระหว่าง ฐานข้อมูล ระบบการจัดการฐานข้อมูล และระบบฐานข้อมูล

ตัวอย่าง

สมมุติเราต้องการสร้างระบบฐานข้อมูลมหาวิทยาลัย (university database system) จะต้องเริ่มว่าระบบของเรามีความต้องการเก็บข้อมูลส่วนใดของมหาวิทยาลัย ในตัวอย่างนี้ ขอกำหนดว่าต้องการเก็บข้อมูลเกี่ยวกับวิชาและตอนเรียนที่นิสิตลงทะเบียนเรียน ก็จะได้ฐานข้อมูล 3 ฐาน ที่รองรับขอบเขตงานที่เราสนใจ ได้แก่ ฐานข้อมูลนิสิต ฐานข้อมูลวิชา และฐานข้อมูลตอนเรียน วิธีเขียนชื่อฐานข้อมูลนั้น นิยมเขียนด้วยตัวพิมพ์ใหญ่ ดังนั้น ขอตั้งชื่อฐานข้อมูลทั้ง 3 ฐาน ดังนี้

STUDENT
COURSE
SECTION

ระบบการจัดการฐานข้อมูลมีเครื่องมือแสดงโครงสร้างของข้อมูลแต่ละฐานให้ผู้ใช้งานเข้าใจได้ง่าย วิธีหนึ่งคือการแสดงข้อมูลในรูปแบบตาราง ดังตัวอย่างในรูปที่ 2

สำหรับระบบฐานข้อมูลขนาดใหญ่ มักมีผู้ใช้งานหลายคนพร้อมกัน สิ่งที่ต้องคำนึงถึงในการออกแบบคือการเข้าถึง (access) ที่เหมาะสม ตลอดจนการรักษาความปลอดภัยของข้อมูล และความถูกต้องของข้อมูล ในแง่ของการเข้าถึงข้อมูลนั้น ผู้ออกแบบฐานข้อมูลจำต้องคำนึงถึงการออกแบบให้ระบบเสถียร แม้มีผู้ใช้งานที่เข้าถึงข้อมูลหลายคนพร้อมกัน และยังต้องคำนึงถึงความปลอดภัยให้ผู้ใช้งานเข้าถึงข้อมูลเฉพาะที่ได้รับสิทธิ์ด้วย นอกเหนือจากนั้น ในแง่ของความปลอดภัยของข้อมูลยังต้องจัดการเรื่องการสำรอง (backup) ข้อมูลเป็นระยะ เพื่อให้การคืนสภาพ หรือการนำกลับมาใช้

(recovery) สามารถทำได้สะดวกหากระบบเกิดความขัดข้อง (disaster) อีกด้วย

STUDENT	Name	StudentID	Section	Major
	Kawin	5970105021	2110101	Computer Engineering
	Chonlatorn	5970136421	2110101	Computer Engineering

COURSE	CourseName	CourseNumber	Credits	Department
	Discrete Structures and Computability	2110202	4	Computer Engineering
	Introduction to Data Structures	2110211	3	Computer Engineering
	Programming Methodology I	2110215	3	Computer Engineering

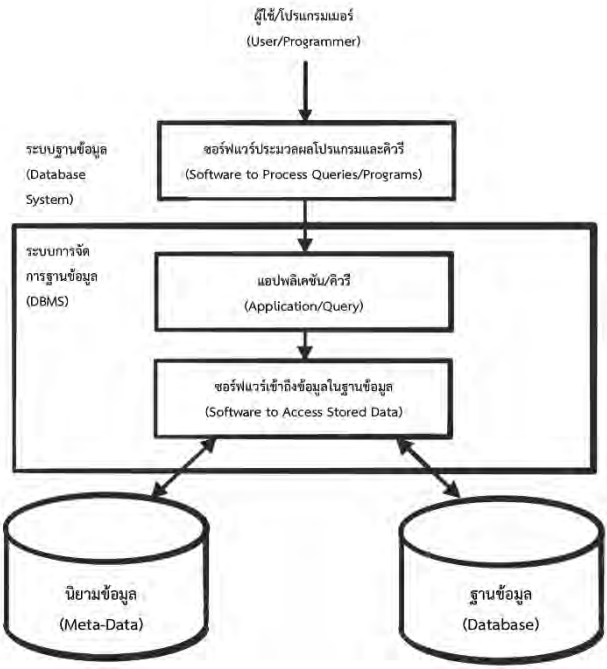
SECTION	SectionIdentifier	CourseNumber	Semester	Year	Instructor
	85	2110202	First	2018	Proadpran
	92	2110211	First	2018	Atiwong
	102	2110215	Second	2018	Athasit

รูปที่ 2 ตารางฐานข้อมูลมหาวิทยาลัย

1.2 ความสัมพันธ์ของระบบการจัดการฐานข้อมูลและระบบฐานข้อมูล

แนวคิดของความสัมพันธ์ของระบบการจัดการฐานข้อมูลและระบบฐานข้อมูลแสดงในรูปที่ 3 ซึ่งปรับมาจาก [2] แสดงความสัมพันธ์ของระบบการจัดการฐานข้อมูลและระบบฐานข้อมูลระดับแนวคิด เมื่อพิจารณารูปจากบนลงล่าง จะเริ่มจากผู้ใช้งานระบบฐานข้อมูล ซึ่งหมายรวมถึงผู้ใช้งานข้อมูลและโปรแกรมเมอร์ (programmer) โดยที่ผู้ใช้ข้อมูลมักใช้โปรแกรมสำเร็จรูปหรือแอปพลิเคชันในการเรียกดูข้อมูล ส่วนโปรแกรมเมอร์มักเรียกใช้ข้อมูล

ผ่านภาษาคิวรีหรือภาษาสอบถาม (query language) ในการสืบค้นข้อมูล เพื่อให้ได้ข้อคำถาม (query) จากจุดนี้จะเป็นหน้าที่ของระบบการจัดการฐานข้อมูล โดยคร่าว ๆ แล้วระบบการจัดการฐานข้อมูลมีหน้าที่นำข้อคำถามนั้น ๆ ไปประมวลผล แล้วส่งคำสั่งเพื่อทำให้รู้ว่าจะเข้าถึงข้อมูล หรือเรียกข้อมูลจากฐานข้อมูลผ่านนิยามข้อมูล (meta-data) ที่จัดเก็บไว้แยกกันอย่างไร ทั้งนี้ขอให้สังเกตว่าขั้นตอนการดึงข้อมูลจากหน่วยเก็บฐานข้อมูล (stored database) นั้นไม่จำเป็นว่าจะต้องเป็นงานของระบบการจัดการฐานข้อมูล



รูปที่ 3 ระบบการจัดการฐานข้อมูลและระบบฐานข้อมูล

จะเห็นได้ว่าหน้าที่ของระบบการจัดการฐานข้อมูลคือส่วนกลางของรูปที่ 3 ส่วนรายละเอียดของความสามารถของระบบการจัดการฐานข้อมูลมีหลายอย่าง ขอกล่าวถึงความสามารถที่สำคัญ ได้แก่

1. สามารถกำหนดคุณลักษณะของฐานข้อมูล ตั้งแต่ชนิดของข้อมูล (data type) โครงสร้างของข้อมูล (structure) และเงื่อนไขบังคับ (constraint) ต่าง ๆ
2. สามารถสร้างฐานข้อมูลและบรรจุข้อมูลลงฐานข้อมูลในหน่วยเก็บรอง (secondary storage)
3. สามารถจัดดำเนินการ (manipulate) ฐานข้อมูล เช่น การแทรกข้อมูล (insertion) การลบ (deletion) การแก้ไข (modification) การสอบถาม และการจัดทำรายงาน (report)
4. สามารถประมวลผลพร้อมกันและใช้ข้อมูลร่วมกัน (concurrent processing and sharing) ให้สามารถมีผู้ใช้งานหลายคนพร้อมกันได้ มีโปรแกรมหลายโปรแกรมทำงานพร้อมกันได้ โดยที่ข้อมูลยังคงความถูกต้อง
5. สามารถจัดการป้องกันและดูแลความปลอดภัย ผ่านทางการตรวจสอบการเข้าถึงที่ไม่ได้รับการอนุญาต (unauthorized access) และจัดการด้านความมั่นคงของข้อมูลเมื่อซอฟต์แวร์หรือฮาร์ดแวร์มีปัญหา
6. สามารถทำการปรับปรุงและบำรุงรักษาฐานข้อมูล (maintenance) และปรับเปลี่ยนระบบไปตามความต้องการที่เปลี่ยนไป

1.3 **คุณสมบัติของฐานข้อมูล**

ข้อมูลที่เก็บไว้ในฐานข้อมูลเดียวกันควรมีความเกี่ยวข้องกันและมีความหมายในตัวเอง ฐานข้อมูลแต่ละฐานจะต้องถูกออกแบบแล้วสร้างขึ้นตามแบบเพื่อใช้บรรจุข้อมูลสำหรับวัตถุประสงค์อันใดอันหนึ่งที่แน่ชัด ฐานข้อมูลสามารถเป็นตัวแทนเหตุการณ์หรือเป็นแบบจำลองของปัญหาจริงหรือส่วนหนึ่งของปัญหาจริงที่ข้อมูลในฐานข้อมูลนั้นเกี่ยวข้อง มีขอบเขตแน่ชัด เช่น ฐานข้อมูลการลงทะเบียนของนิสิตในมหาวิทยาลัย จะมีขอบเขตคือ ข้อมูลเกี่ยวกับการลงทะเบียนเรียนของนิสิต ชื่อนามสกุลนิสิต ปีที่นิสิตเข้าเรียน สาขาวิชาที่นิสิตเรียน วิชาที่ต้องการลงทะเบียน ปีการศึกษาที่ลงทะเบียน จำนวนนิสิตของแต่ละวิชา ห้องเรียน ตำราเรียน อาจารย์ผู้สอน อาจารย์ที่ปรึกษา และเกรด เป็นต้น ขอบเขตของฐานข้อมูลนี้ไม่ควรมีข้อมูลอื่นใดที่ไม่เกี่ยวข้อง เช่น ข้อมูลเงินเดือนของอาจารย์ ข้อมูลตำแหน่งเจ้าหน้าที่ ข้อมูลหนังสือในห้องสมุด เป็นต้น

ฐานข้อมูลที่ดีจะต้องสามารถจัดเก็บและจัดการข้อมูลให้มีข้อมูลที่ถูกต้อง ทันกาลอยู่เสมอ ต้องไม่มีความขัดแย้งกันเอง มีความซ้ำซ้อน (redundancy) น้อยที่สุด และหากจำเป็นต้องซ้ำซ้อนก็ต้องมีการจัดการให้ข้อมูลถูกต้องทั่วถึงกันทั้งระบบ สามารถเรียกข้อมูลมาใช้งานได้ในเวลาอันเหมาะสมกับงานนั้น ๆ

1.4 **คุณสมบัติของระบบการจัดการฐานข้อมูล**

คุณสมบัติของระบบการจัดการฐานข้อมูลมีหลายประการ จาก [1] ได้ระบุที่สำคัญ ได้แก่

1. ผู้ใช้งานสามารถเข้าใจข้อมูลได้ง่าย เพราะระบบการจัดการฐานข้อมูลจะทำสารบัญหรือแค็ตตาล็อก (catalog) ข้อมูลในระบบโดยเก็บในรูปแบบนิยามข้อมูล ทำให้ผู้ใช้งานสามารถเข้าใจความหมายของทุกข้อมูลได้เหมือน ๆ กัน โดยคำอธิบายนั้นประกอบด้วยโครงสร้างของข้อมูล ชนิดรูปแบบของข้อมูล พร้อมทั้งข้อบังคับต่าง ๆ ถ้ามี โดยระบบการจัดการฐานข้อมูลจะแยกเก็บนิยามข้อมูลไว้แยกจากหน่วยเก็บข้อมูลและแยกจากแอปพลิเคชัน จึงเป็นคุณสมบัติสำคัญที่ทำให้เราสามารถนำข้อมูลชุดเดิมไปใช้ในแอปพลิเคชันต่างกันได้ เช่น เราสามารถใช้ฐานข้อมูลนิสิตในระบบฐานข้อมูลศิษย์เก่าได้ แม้ว่าระบบฐานข้อมูลศิษย์เก่าจะแยกจากระบบฐานข้อมูลมหาวิทยาลัย เป็นต้น
2. มีความอิสระระหว่างแอปพลิเคชันกับข้อมูล หมายความว่าระบบฐานข้อมูลจะอนุญาตให้เปลี่ยนแปลงโครงสร้างของหน่วยเก็บข้อมูล และการจัดการต่าง ๆ โดยไม่จำเป็นต้องเปลี่ยนแอปพลิเคชัน เนื่องจากโครงสร้างของไฟล์ข้อมูลนั้นถูกแยกออกจากแอปพลิเคชัน ซึ่งไม่เหมือนกระบวนการประมวลผลไฟล์ (file processing) แบบดั้งเดิมที่โครงสร้างของไฟล์ข้อมูลจะถูกฝังติดกับแอปพลิเคชัน จึงต้องทำการเปลี่ยนแปลงโปรแกรมทั้งหมดเมื่อต้องการเปลี่ยนโครงสร้างไฟล์
3. มีความอิสระระหว่างโปรแกรมกับการจัดการ กรณีจะเกิดเมื่อใช้ฐานข้อมูลเชิงวัตถุ (object-oriented database) ซึ่งผู้ใช้สามารถระบุการจัดการหรือโอเปอเรชัน (operation) ให้กับข้อมูล โดยที่โอเปอเรชันประกอบด้วยอินเทอร์เฟซ (interface) และอิมพลิเมนเทชัน (implementation) คุณสมบัติข้อนี้คือเมื่อมีการเปลี่ยนแปลงส่วนอิมพลิเมนเทชันจะไม่กระทบต่ออินเทอร์เฟซ เราเรียกคุณสมบัตินี้ว่า โปรแกรม

กับการจัดการนั้นแยกกันได้หรือมีอิสระต่อกัน (program-operation independence) นั่นเอง

4. มีการใช้แบบจำลองข้อมูล (data model) ในระบบการจัดการฐานข้อมูล ทำให้ผู้ใช้สามารถกำหนดสาระสำคัญของข้อมูล (data abstraction) ได้ง่ายทั้งที่ข้อมูลอาจมีความสัมพันธ์ที่ซับซ้อน ซึ่งมีผลทำให้ผู้ใช้งานสามารถทำงานผ่านโมเดลระดับแนวคิด (conceptual model) โดยซ่อนรายละเอียดของหน่วยเก็บข้อมูลที่ซับซ้อนไว้ จากนั้นระบบการจัดการฐานข้อมูลจะทำหน้าที่ในการดึงข้อมูลได้อย่างถูกต้องโดยใช้รายละเอียดจากแค็ตตาล็อกและนิยามข้อมูลเป็นพอ
5. มีการรองรับมุมมองสำหรับผู้ใช้ต่างกันได้ กล่าวคือ ผู้ใช้แต่ละคนหรือแต่ละกลุ่มสามารถเรียกใช้ข้อมูลด้วยมุมมอง (view) ที่แตกต่างกัน แต่ละมุมมองจะทำให้ผู้ใช้เข้าถึงส่วนของข้อมูลที่ได้รับการอนุญาตได้อย่างสะดวกและสร้างความปลอดภัยต่อฐานข้อมูลด้วย
6. มีการจัดการการใช้ข้อมูลร่วมกัน (sharing) และประมวลผลรายการเมื่อมีผู้ใช้หลายคนพร้อมกัน (multiuser transaction processing) เพราะในระบบฐานข้อมูลใด ๆ มักมีผู้ใช้หลายคนพร้อมกันซึ่งอาจมีสิทธิ์ในการแก้ไขข้อมูลได้ ดังนั้นระบบการจัดการฐานข้อมูลจึงจำเป็นต้องรองรับการทำงานแบบพร้อมกันได้อย่างมีประสิทธิภาพ เช่น การจัดลำดับการทำรายการ (transaction) อย่างถูกต้องจนครบสมบูรณ์ ซึ่งนับว่าเป็นส่วนสำคัญมากสำหรับแอปพลิเคชันประเภทออนไลน์ ทั้งยังเป็นการป้องกันการเข้าถึงข้อมูลโดยไม่ได้รับอนุญาตอีกด้วย
7. สามารถควบคุมความซ้ำซ้อน ในการจัดเก็บข้อมูล ซึ่งความซ้ำซ้อนของข้อมูลนั้นเป็นสาเหตุของปัญหามากมาย เช่น สิ้นเปลืองพื้นที่ในการเก็บข้อมูล หรือข้อมูลที่เก็บไว้ไม่ตรงกัน

8. มีการจัดการการสอบถาม ได้อย่างมีประสิทธิภาพ เช่น สามารถกำหนดดัชนี (index) เพื่อให้เข้าถึงข้อมูลที่ใช้บ่อยได้รวดเร็วขึ้น
9. มีความสามารถในการสำรองและกู้คืนหากระบบฐานข้อมูลมีปัญหาหรือล่ม (disaster)
10. สามารถระบุเงื่อนไขบังคับต่าง ๆ ได้หลายรูปแบบและหลายลำดับที่ช่วยลดความผิดพลาดของข้อมูล เช่น บังคับให้ข้อมูลต้องไม่ซ้ำด้วยข้อบังคับของคีย์ (key constraint) ข้อมูลต้องมีข้อบังคับความเป็นบูรณภาพ (integrity constraint) เป็นต้น
11. มีศักยภาพในการควบคุมมาตรฐาน โดยเฉพาะการทำแอปพลิเคชันฐานข้อมูลในองค์กรขนาดใหญ่ที่มีโปรแกรมเมอร์จำนวนมากที่สร้างหรือใช้ฐานข้อมูลเดียวกัน เพราะทำให้ทุกคนได้เข้าใจข้อมูลเหมือน ๆ กันได้ด้วยการใช้พจนานุกรมข้อมูล (data dictionary)

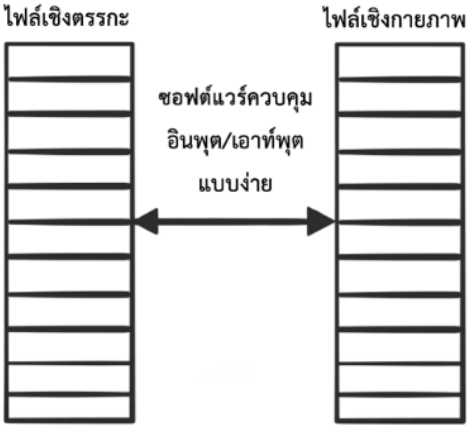
1.5 วิวัฒนาการของระบบการจัดการฐานข้อมูล

แนวคิดของระบบการจัดการฐานข้อมูลมีมาแต่ยาวนาน สามารถแบ่งได้เป็น 5 ช่วง โดยช่วงที่ 1-4 อ้างอิงจาก [4, 5] และช่วงที่ 5 คือ ยุคปัจจุบัน มีรายละเอียดดังนี้

ช่วงที่ 1 : ยุคการจัดการข้อมูลแบบไฟล์
ช่วงที่ 2 : ยุคการจัดการข้อมูลแบบการเข้าถึงไฟล์
ช่วงที่ 3 : ยุคระบบฐานข้อมูลในช่วงแรกเริ่ม
ช่วงที่ 4 : ยุคระบบฐานข้อมูลเชิงสัมพันธ์
ช่วงที่ 5 : ยุคปัจจุบัน

1.5.1 ช่วงที่ 1 : ยุคการจัดการข้อมูลแบบไฟล์

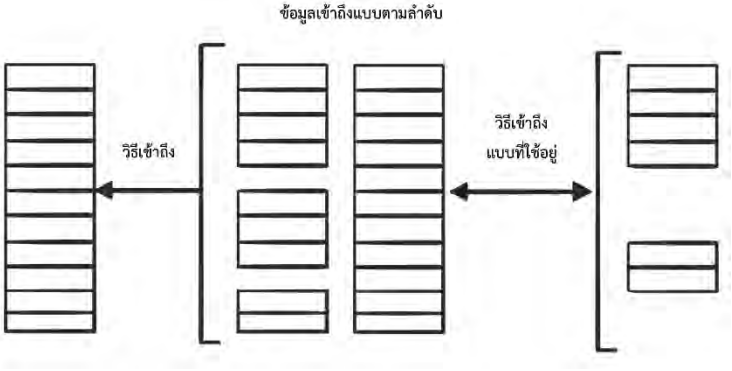
คือการจัดระบบไฟล์แบบลำดับอนุกรม (serial) หมายถึง โครงสร้างข้อมูลทางกายภาพ (physical data structure) มีลำดับคู่กับโครงสร้างข้อมูลเชิงตรรกะ (logical data structure) เสมอ และใช้กระบวนการประมวลผลแบบกลุ่ม (batch-processing) ทำให้ไม่สามารถเข้าถึงข้อมูลแบบเวลาจริงได้ จะเห็นได้ว่าการจัดเก็บไฟล์จะเกิดการซ้ำซ้อนโดยปริยาย โปรแกรมเมอร์จำเป็นต้องเขียนโปรแกรมสำหรับการใช้ข้อมูลชุดนั้น ๆ โดยเฉพาะ เมื่อมีการเปลี่ยนแปลงโครงสร้างข้อมูลหรือเปลี่ยนแปลงโปรแกรมใด ๆ โปรแกรมเมอร์จะต้องเขียนโปรแกรมใหม่และจัดเรียงข้อมูลใหม่เสมอ เกิดการซ้ำซ้อนระหว่างไฟล์ข้อมูลอย่างมาก รูปที่ 4 แสดงความสัมพันธ์ของข้อมูลแบบไฟล์หรือแบบลำดับอนุกรมนี้



รูปที่ 4 การจัดการข้อมูลแบบไฟล์

1.5.2 ช่วงที่ 2 : ยุคการจัดการข้อมูลแบบการเข้าถึงไฟล์

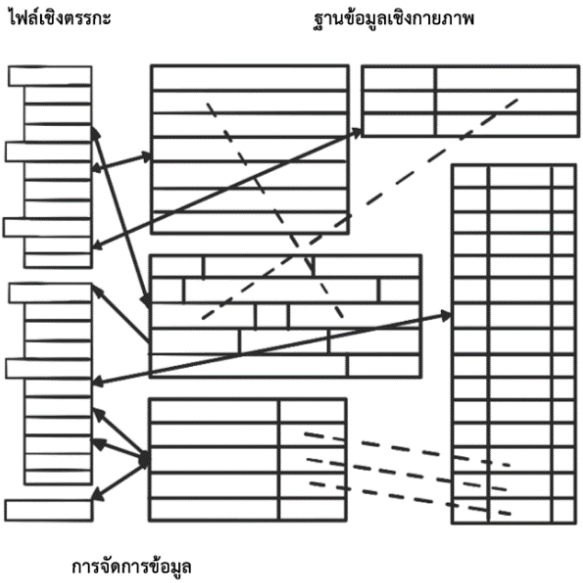
ยุคการจัดการข้อมูลแบบการเข้าถึงไฟล์ (File Access Method) ดังแสดงในรูปที่ 5 เกิดขึ้นในช่วงปลาย ค.ศ. 1960 ได้มีการเสนอการเข้าถึงข้อมูลแบบตามลำดับ (Serial access) หรือการเข้าถึงข้อมูลแบบสุ่ม (Random access) ซึ่งทำได้ด้วยการแยกไฟล์เชิงตรรกะ (Logical file) และไฟล์ข้อมูลจริง (Physical file) ออกจากกัน ส่วนการประมวลผลนั้น มีทั้งการประมวลผลแบบเดิม คือแบบกลุ่ม (Batch) และมีแบบใหม่เกิดขึ้น คือแบบอินไลน์ (In-line) ซึ่งนับเป็นการประมวลผลแบบเวลาจริง ด้วยวิธีนี้ ทำให้โปรแกรมเมอร์สามารถเปลี่ยนแปลงหน่วยเก็บข้อมูลได้โดยไม่จำเป็นต้องแก้ไขแอปพลิเคชัน การจัดโครงสร้างข้อมูลอาจอยู่ในรูปแบบลำดับอนุกรม ลำดับเชิงดัชนี (Index sequential) หรือเข้าถึงข้อมูลตรง ๆ ได้ (Simple direct access) ในยุคนี้ยังไม่มีการใช้คีย์หลายตัว (Multiple-key) ในการสืบค้นข้อมูล แม้จะเริ่มมีการตรวจสอบความปลอดภัยของข้อมูล แต่นับว่ายังไม่ปลอดภัยนัก ยังคงพบความซ้ำซ้อนของข้อมูลจำนวนมากอยู่



รูปที่ 5 การจัดการข้อมูลแบบการเข้าถึงไฟล์

1.5.3 ช่วงที่ 3 : ยุคระบบฐานข้อมูลในช่วงแรกเริ่ม

ตั้งแต่ช่วงต้น ค.ศ. 1970 ได้มีการสร้างไฟล์เชิงตรรกะมากกว่าหนึ่งไฟล์ ไฟล์เชิงตรรกะนี้จะมีความแตกต่างกัน เพื่อให้เข้าถึงข้อมูลเชิงกายภาพ (physical data) หรือเรียกว่าข้อมูลจริงชุดเดียวกันได้ ดังแสดงในรูปที่ 6 นับเป็นการรองรับให้แอปพลิเคชันต่างกันสามารถเข้าถึงข้อมูลชุดเดียวกัน ส่งผลให้ลดความซ้ำซ้อนของข้อมูลและทำให้เกิดบูรณภาพของข้อมูล (data integrity) และการแก้ไขโปรแกรมสามารถทำได้โดยไม่ต้องปรับแก้ข้อมูลตาม วิธีการที่เกิดขึ้นในยุคนี้ทำให้สามารถเข้าถึงข้อมูลได้ทั้งระดับเขตข้อมูลหรือฟิลด์ (field) หรือในระดับกลุ่ม (group) ได้ และยังรองรับการสืบค้นข้อมูลด้วยมัลติเพิลคีย์ หรือคีย์หลายตัวได้ด้วย ทำให้มีการจัดระบบข้อมูลที่ซับซ้อนโดยไม่ทำให้โปรแกรมแอปพลิเคชันซับซ้อนได้



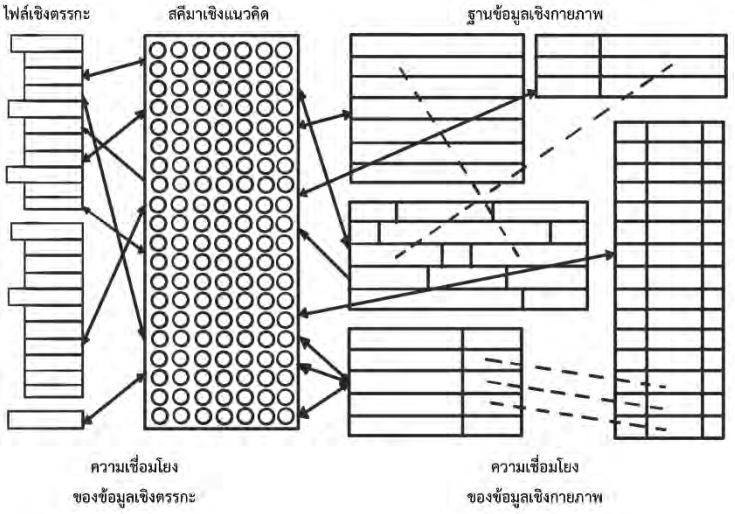
รูปที่ 6 ระบบฐานข้อมูลในช่วงแรกเริ่ม

1.5.4 ช่วงที่ 4 : ระบบฐานข้อมูลเชิงสัมพันธ์

ในยุคนี้ ระบบการจัดการฐานข้อมูลมีการแยกข้อมูลเชิงตรรกะ และข้อมูลจริง ออกจากกันอย่างอิสระ ดังแสดงในรูปที่ 7 ทำให้สามารถพัฒนาระบบฐานข้อมูลได้ง่ายขึ้นมาก อีกทั้งยังเพิ่มความสามารถในการติดตาม (monitor) การบำรุงรักษา การจัดการความปลอดภัย และความสามารถอื่น ๆ ดังได้กล่าวไว้ก่อนหน้านี้ รูปแบบของระบบฐานข้อมูลแบบนี้เรียกว่า ระบบฐานข้อมูลเชิงสัมพันธ์ (relational database system) ซึ่งให้ความสำคัญกับความสัมพันธ์ของข้อมูลในระบบเป็นหลัก เกิดจากบริษัท

ไอบีเอ็มในช่วงต้น ค.ศ. 1970 และตามมาด้วยระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ (relational DBMS) ตั้งแต่ช่วงทศวรรษที่ 1980

จากนั้น มีผู้เสนอระบบการจัดการฐานข้อมูลเชิงวัตถุขึ้นในช่วงปลายทศวรรษที่ 1990 เพื่อตอบสนองการเขียนโปรแกรมด้วยภาษาเชิงวัตถุ (object-oriented language) แต่การใช้งานไม่เป็นที่นิยมเท่ากับระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ ตัวอย่างระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ที่เป็นที่นิยม ได้แก่ ออราเคิล มายเอสคิวแอล และ เอสคิวแอลเซิร์ฟเวอร์ เป็นต้น นอกจากนั้น ยังมีผลิตภัณฑ์ที่ออกแบบเพื่อรองรับข้อมูลที่นอกเหนือจากตัวอักษรและตัวเลข เพื่อให้รองรับข้อมูลภาพนิ่ง ภาพเคลื่อนไหว ข้อมูลเสียง ข้อมูลเชิงพื้นที่ และอื่น ๆ



รูปที่ 7 ระบบฐานข้อมูลเชิงสัมพันธ์

1.5.5 ช่วงที่ 5 : ยุคปัจจุบัน

เมื่อ ค.ศ. 1998 ได้มีผู้เสนอแนวคิดฐานข้อมูลแบบโนเอสคิวแอล (NoSQL หรือ Not-Only-SQL) ที่ทำให้เกิดการเปลี่ยนแปลงครั้งใหญ่ในวงการฐานข้อมูล ตัวอย่างระบบการจัดการฐานข้อมูลแบบโนเอสคิวแอล ได้แก่ มองโกดีบี (MongoDB) เรดิส (Redis) คัสซานดรา (Cassandra) เป็นต้น ซึ่งแต่ละผลิตภัณฑ์ยังมีรายละเอียดต่างกันอีกด้วย โดยรวมแล้วแนวคิดแบบโนเอสคิวแอลนี้จะเหมาะกับข้อมูลที่ไม่มีโครงสร้างแน่ชัด และเหมาะกับข้อมูลขนาดใหญ่ในยุคบิ๊กดาต้า (big data) เช่น ข้อมูลในสังคมออนไลน์ เป็นต้น อย่างไรก็ดี ความนิยมในระบบฐานข้อมูลเชิงสัมพันธ์ยังสูงกว่าฐานข้อมูลแบบโนเอสคิวแอลอยู่มาก

1.6 บทสรุปและเรื่องชวนคิด

ถึงแม้การใช้ระบบการจัดการฐานข้อมูลจะมีข้อดีมากมายและเป็นที่นิยมแพร่หลาย แต่การเลือกใช้ระบบการจัดการฐานข้อมูลก็ต้องพิจารณาอย่างถี่ถ้วนก่อนจะใช้ เนื่องจากการใช้ระบบการจัดการฐานข้อมูลมีค่าใช้จ่ายสูง เพราะมักต้องการการลงทุนทางฮาร์ดแวร์และซอฟต์แวร์ค่อนข้างสูง ดังนั้น สำหรับกรณีที่ฐานข้อมูลและแอปพลิเคชันไม่ซับซ้อนมากนัก หรือมีการเปลี่ยนแปลงน้อย หรือมีผู้ใช้แค่คนเดียว ก็ไม่มีความจำเป็นใด ๆ ที่ต้องใช้ระบบการจัดการฐานข้อมูล

สำหรับองค์กรขนาดใหญ่ คงหลีกเลี่ยงการใช้ฐานข้อมูลไม่ได้ แต่ด้วยรูปแบบของข้อมูลสมัยใหม่ที่ไม่มีโครงสร้างที่แน่นอน ผู้เรียนคิดว่าฐานข้อมูลเชิงสัมพันธ์ที่ได้รับความนิยมมาราว 50 ปี จะถึงวันหมดอายุหรือยัง

แบบฝึกหัดท้ายบท

1. จงยกตัวอย่างโครงการที่พบเห็นและควรมีฐานข้อมูลในการทำงานนั้น ๆ มา 3 โครงการ พร้อมยกตัวอย่างข้อมูลที่ฐานข้อมูลนั้นควรมี
2. จงอธิบายความแตกต่างระหว่างระบบฐานข้อมูลและระบบการจัดการฐานข้อมูล
3. จงเปรียบเทียบความแตกต่างของฐานข้อมูลยุคการจัดการข้อมูลแบบไฟล์กับยุคปัจจุบัน
4. เมื่อใดถึงควรเลือกใช้ระบบการจัดการฐานข้อมูลแบบโนเอสคิวแอล
5. จงยกตัวอย่างสิ่งที่ควรคำนึงในการออกแบบฐานข้อมูล
6. หากมีโปรแกรมเมอร์ทำงานติดต่อกับฐานข้อมูลเดียวกันเป็นจำนวนมาก สิ่งใดช่วยให้ทุกคนสามารถทำงานร่วมกันได้โดยเข้าใจฐานข้อมูลตรงกัน
7. จงบอกคุณสมบัติของข้อมูลภายในฐานข้อมูลเดียวกัน
8. “ระบบการจัดการฐานข้อมูลหนึ่ง ๆ สามารถเก็บข้อมูลได้หลายชนิด ดังนั้นจึงสามารถใช้กับงานที่มีข้อมูลลักษณะใดก็ได้” จากข้อความข้างต้น ผู้เรียนเห็นด้วยหรือไม่ จงอภิปราย
9. ภาษาสอบถาม มีหน้าที่อะไรในระบบฐานข้อมูล
10. สิ่งใดทำให้หลายแอปพลิเคชันสามารถประมวลผลและใช้ข้อมูลร่วมกันได้

บทที่ 2
การพัฒนาระบบฐานข้อมูล

วัตถุประสงค์

1. เพื่อให้เข้าใจขั้นตอนการพัฒนาระบบฐานข้อมูล
2. เพื่อให้รู้จักสถาปัตยกรรมข้อมูลแบบต่าง ๆ
3. เพื่อให้รู้จักภาษาฐานข้อมูล
4. เพื่อให้รู้จักภาษาที่ใช้ในการจัดการฐานข้อมูล
5. เพื่อให้รู้จักระบบการจัดการฐานข้อมูลประเภทต่าง ๆ

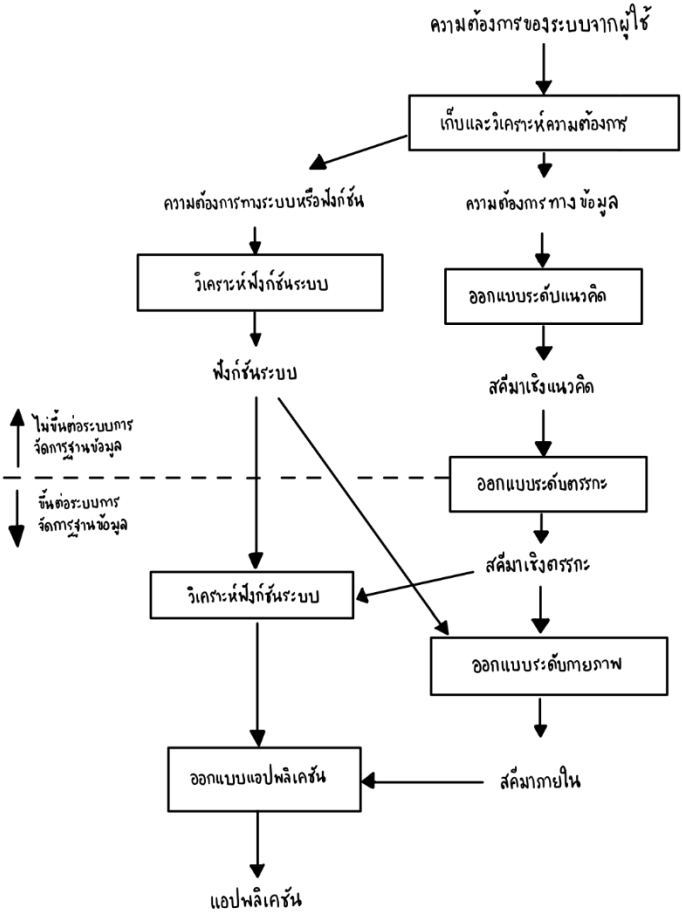
การพัฒนาระบบฐานข้อมูลมักเกิดควบคู่กับการพัฒนาระบบหรือแอปพลิเคชันที่ใช้ข้อมูลนั้น ๆ เพื่อตอบโจทย์จากเจ้าของงาน เนื่องจากแอปพลิเคชันส่วนใหญ่โดยเฉพาะในธุรกิจจำเป็นที่จะต้องใช้ฐานข้อมูลเป็นส่วนหนึ่งในการทำงานเสมอ

กระบวนการทางวิศวกรรมซอฟต์แวร์ [6-8] ระบุว่ากระบวนการสร้างซอฟต์แวร์หรือแอปพลิเคชัน (ต่อไปนี้จะเรียกว่า แอปพลิเคชัน) มักเริ่มจากการเก็บข้อมูลความต้องการทางระบบจากผู้ใช้ แล้วนำไปวิเคราะห์ความต้องการ ออกแบบ เขียนโปรแกรม ทดสอบ แล้วจึงจบกระบวนการ

ขอเน้นที่ขั้นตอนการวิเคราะห์ความต้องการ ซึ่งจะได้ผลการวิเคราะห์เป็น 2 ประเด็นหลัก ได้แก่ ความต้องการทางระบบหรือฟังก์ชัน และความต้องการทางข้อมูล

สำหรับแอปพลิเคชันที่ต้องใช้ฐานข้อมูล ก็จะต้องดำเนินการออกแบบฐานข้อมูลควบคู่กันไป โดยพิจารณาว่าแอปพลิเคชันนั้นจำเป็นต้องใช้ข้อมูลอะไรในขั้นตอนใดบ้าง

รูปที่ 8 แสดงความสัมพันธ์ระหว่างการพัฒนาแอปพลิเคชัน กับการพัฒนาระบบฐานข้อมูล ซึ่งเป็นสองกระบวนการที่มักต้องทำควบคู่กัน

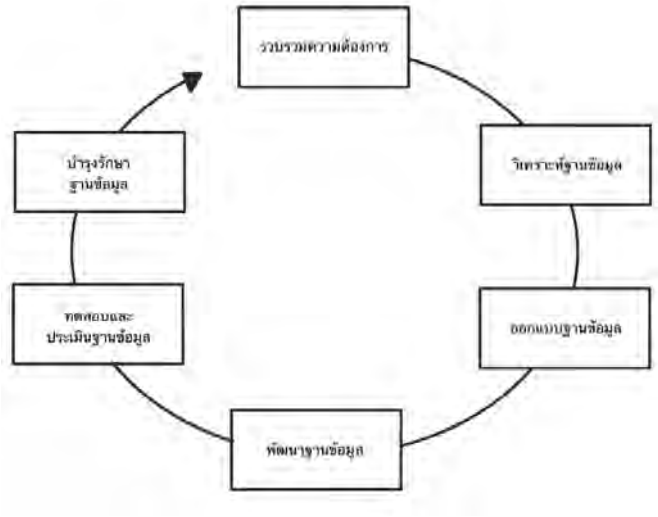


รูปที่ 8 ความสัมพันธ์ระหว่างการพัฒนาแอปพลิเคชัน กับการพัฒนาระบบฐานข้อมูล

2.1 การพัฒนาระบบฐานข้อมูล

ขั้นตอนการพัฒนาระบบฐานข้อมูล [1, 9-14] แสดงในรูปที่ 9 ประกอบด้วย 6 ขั้นตอนคือ

1. รวบรวมความต้องการ (requirement gathering)
2. วิเคราะห์ฐานข้อมูล (database analysis)
3. ออกแบบฐานข้อมูล (database design)
4. พัฒนาฐานข้อมูล (database implementation)
5. ทดสอบและประเมินฐานข้อมูล (database testing and evaluation)
6. บำรุงรักษาฐานข้อมูล (database maintenance)



รูปที่ 9 ขั้นตอนการพัฒนาระบบฐานข้อมูล

ขั้นตอนทั้งหกนี้ สามารถนำมาใช้ในการออกแบบระบบฐานข้อมูลได้ สำหรับตำรานี้ เราจะเน้นขั้นตอนที่ 3 คือการออกแบบฐานข้อมูล ซึ่งจะกล่าวโดยละเอียดต่อไป

อย่างไรก็ดี การออกแบบฐานข้อมูลยังต้องคำนึงถึงระบบคอมพิวเตอร์ที่จะใช้งานฐานข้อมูลนั้นเป็นหลักด้วย เพราะสถาปัตยกรรมของระบบคอมพิวเตอร์ที่ต่างยุคกันนั้นจะส่งผลถึงการทำงานร่วมกันระหว่างระบบทั้งหมดกับระบบฐานข้อมูล เราจึงต้องรู้จักสถาปัตยกรรมระบบฐานข้อมูลเพื่อให้เข้าใจว่าระบบทั้งหมดมีการติดต่อหรือทำงานร่วมกันกับระบบฐานข้อมูลอย่างไร

2.2 สถาปัตยกรรมระบบฐานข้อมูล

ยุคสมัยของคอมพิวเตอร์มีการเปลี่ยนแปลงอย่างรวดเร็ว สามารถเห็นความแตกต่างได้อย่างชัดเจน ทั้งในแง่ของขนาดและความสามารถในการประมวลผล เริ่มจากคอมพิวเตอร์ระบบเมนเฟรม (mainframe) ที่มีขนาดใหญ่มากจนถึงโทรศัพท์แบบสมาร์ตโฟนที่ใช้กันอย่างแพร่หลายทุกวันนี้

ในยุคแรก ๆ ที่เริ่มใช้แนวคิดแบบฐานข้อมูลในการทำแอปพลิเคชัน ฮาร์ดแวร์ทางคอมพิวเตอร์ที่ใช้ในองค์กรสมัยนั้นมีขนาดใหญ่ คือระบบเมนเฟรมที่ทำการประมวลผลแบบศูนย์รวม จากนั้น เมื่อมีความเปลี่ยนแปลงทางระบบฮาร์ดแวร์พร้อมทั้งมีระบบปฏิบัติการหลากหลายขึ้นเพื่อรองรับขนาดและรูปแบบของธุรกรรมที่ต่างกัน ทำให้การออกแบบชนิดโมดูลา (modular design) เกิดขึ้น นั่นคือการแบ่งการประมวลผลเป็นส่วน ๆ แทนการประมวลผลแบบรวมศูนย์ในระบบเมนเฟรม ต่อมา ก็เกิดสถาปัตยกรรมแบบ

ลูกข่ายแม่ข่าย (client/server) ที่เป็นที่นิยมอย่างแพร่หลาย พร้อมกับคอมพิวเตอร์ส่วนบุคคล (personal computer) มีความสามารถในการประมวลผลสูงขึ้นเรื่อย ๆ จึงมีการเชื่อมต่อระหว่างคอมพิวเตอร์ส่วนบุคคลไปยังเครื่องแม่ข่าย (server machine) เพื่อจัดให้การประมวลผลเร็วขึ้น ทั้งนี้องค์กรทั้งหลายจะต้องประเมินพื้นที่สำหรับการจัดเก็บข้อมูลให้เหมาะสมกับปริมาณงานอย่างมีประสิทธิภาพ

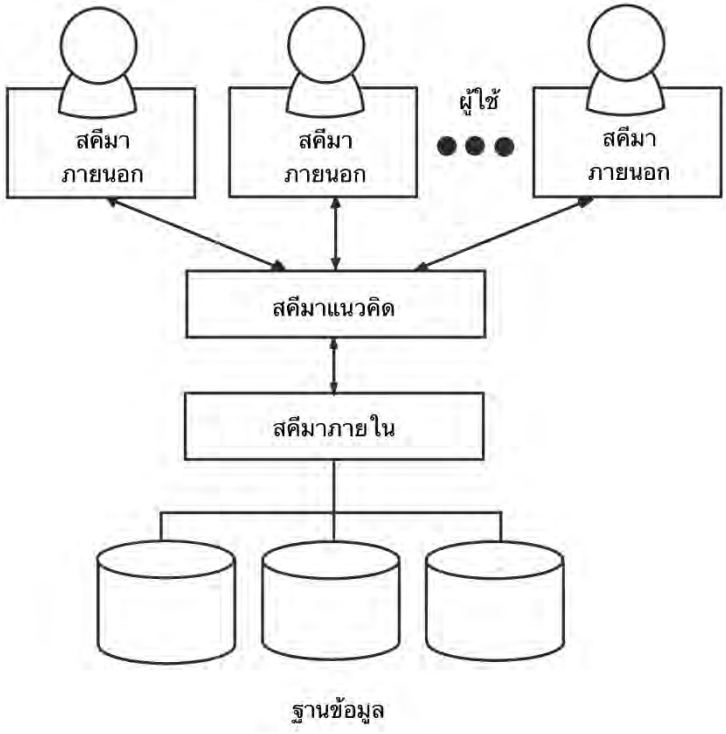
ในปัจจุบัน องค์กรขนาดใหญ่มีทางเลือกที่จะไม่เป็นเจ้าของแม่ข่าย (server) โดยอาจเลือกใช้บริการคลาวด์ (cloud service) [15-17] ซึ่งมีความยืดหยุ่นสูง โดยเฉพาะเรื่องพื้นที่การจัดเก็บข้อมูล ตลอดจนการให้บริการในการดูแลข้อมูล ทำให้ทางเลือกนี้เป็นที่นิยมอย่างมาก

สถาปัตยกรรมระบบฐานข้อมูล คือรูปแบบโครงสร้างและองค์ประกอบระดับต่าง ๆ ที่ประกอบเป็นระบบฐานข้อมูล มีหลายแบบ ในตำรานี้จะอธิบายสถาปัตยกรรม 4 แบบ ได้แก่ 1) สถาปัตยกรรมแบบทรีสคีมา 2) สถาปัตยกรรมแบบรวมศูนย์ 3) สถาปัตยกรรมแบบลูกข่ายแม่ข่าย 4) สถาปัตยกรรมแบบคลาวด์

2.2.1 สถาปัตยกรรมแบบทรีสคีมา

สถาปัตยกรรมแบบทรีสคีมา (three-schema architecture) ดังแสดงในรูปที่ 10 ซึ่งนำมาจาก [2] ประกอบด้วยสคีมา (schema) 3 ระดับ ซึ่งคำว่าสคีมานี้ ภาษาไทยแปลว่า “เค้าร่าง” อธิบายได้ว่าเป็นวิธีเสนอหรือแสดงความคิดบางอย่าง

สถาปัตยกรรมแบบทรีสคีมา ออกแบบมาให้มีความสามารถในการรองรับการจัดการให้ข้อมูลมีความอิสระ เช่น สามารถแก้ไขโปรแกรมโดยไม่ต้องปรับแก้ฐานข้อมูล และสามารถแก้ไขโปรแกรมโดยไม่กระทบกับการดำเนินการ เป็นต้น นอกเหนือจากนี้สถาปัตยกรรมแบบทรีสคีมามีการรองรับมุมมองหรือวิวของข้อมูลได้จำนวนมาก เพื่อให้ผู้ใช้แต่ละประเภทหรือแต่ละคนสามารถเรียกใช้หรือแก้ไขข้อมูลได้ตามความเหมาะสมอีกด้วย



รูปที่ 10 สถาปัตยกรรมแบบทรีสคีมา

สถาปัตยกรรมแบบทรีสคีมาแยกเป็น 3 ระดับ ได้แก่ ระดับภายนอก (external level) ระดับแนวคิด (conceptual level) และระดับภายใน (internal level) ในการทำงานของระบบการจัดการฐานข้อมูลนั้น จะเริ่มด้วยการแปลงความต้องการ (request) ที่ระบุไว้ที่สคีมาภายนอก ไปเป็นสคีมาแนวคิด และแปลงเป็นสคีมาภายในเพื่อดำเนินการกับฐานข้อมูลที่ถูกเก็บไว้ต่อไป กระบวนการดังกล่าวนี้เรียกว่า การแม็ป (mapping) หรือการแปลง โดยมีรายละเอียดดังนี้

สคีมาภายใน – ใช้บรรยายโครงสร้างหน่วยเก็บข้อมูลทางกายภาพของข้อมูลทั้งหมดโดยใช้แบบจำลองเชิงกายภาพ มีการบรรยายรายละเอียดเกี่ยวกับหน่วยเก็บข้อมูลและเส้นทางการเข้าถึงฐานข้อมูลโดยสมบูรณ์

สคีมาแนวคิด – ใช้บรรยายโครงสร้างของฐานข้อมูลทั้งหมดในระดับแนวคิด ไม่มีข้อมูลเชิงกายภาพปรากฏในสคีมานี้ เน้นการบรรยายกลุ่มข้อมูล ความสัมพันธ์ระหว่างกลุ่มข้อมูล การดำเนินการที่ข้อมูลนั้นทำได้ และข้อบังคับต่าง ๆ โดยใช้แบบจำลองข้อมูลเชิงแนวคิด (Conceptual data model) หรือแบบจำลองข้อมูลเชิงปฏิบัติการ (Implementation data model หรือ Representation data model)

สคีมาภายนอก – ใช้บรรยายมุมมองหรือวิว (view) ที่หลากหลายของผู้ใช้ โดยทั่วไปแล้วมักใช้แบบจำลองข้อมูลเชิงปฏิบัติการ

2.2.2 สถาปัตยกรรมแบบรวมศูนย์

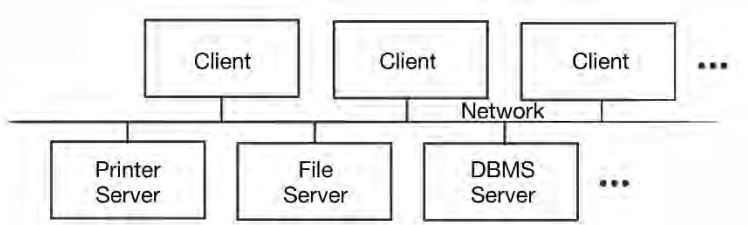
สถาปัตยกรรมสำหรับระบบการจัดการฐานข้อมูลนั้น ได้รับอิทธิพลจากสถาปัตยกรรมระบบคอมพิวเตอร์ทั่วไป ซึ่งในช่วงแรกนั้นมีการใช้คอมพิวเตอร์เมนเฟรมในกระบวนการฟังก์ชันทั้งหมดของระบบ โดยรวบรวมทุกอย่างไว้ในระบบเดียว ไม่ว่าจะเป็นระบบการจัดการฐานข้อมูล แอปพลิเคชัน หรือ คอมไพเลอร์ เรียกว่า สถาปัตยกรรมแบบรวมศูนย์ (centralized architecture)

2.2.3 สถาปัตยกรรมแบบลูกข่ายแม่ข่าย

สถาปัตยกรรมแบบลูกข่ายแม่ข่าย (client/server architecture) ใช้แนวคิดในการนำบริการกลางวางไว้ที่แม่ข่าย เช่น แม่ข่ายจัดการไฟล์ (file server) แม่ข่ายจัดการเครื่องพิมพ์ (printer server) แม่ข่ายเว็บ (web server) และแม่ข่ายอีเมล (email server) เป็นต้น ส่วนเครื่องลูกข่าย (client) จะประกอบด้วยส่วนติดต่อกับผู้ใช้ที่เหมาะสมในการเข้าถึงและใช้ทรัพยากรของเครื่องแม่ข่าย ซึ่งเครื่องลูกข่ายอาจเป็นเครื่องที่ไม่มีดิสก์ หรือเป็นคอมพิวเตอร์ส่วนบุคคลหรือเป็นเวิร์กสเตชัน (workstation) ที่ติดตั้งเฉพาะซอฟต์แวร์ลูกข่าย โดยเครื่องลูกข่ายจะถูกเชื่อมต่อสู่เครื่องแม่ข่ายผ่านทางระบบเครือข่าย (network) เช่น ระบบเครือข่ายท้องถิ่น (Local Area Network : LAN) หรือเครือข่ายไร้สาย (wireless network) เป็นต้น ซึ่งสถาปัตยกรรมแบบลูกข่ายแม่ข่ายนี้แบ่งได้ 2 ประเภท ได้แก่ 1) สถาปัตยกรรมแบบลูกข่ายแม่ข่ายสองชั้น 2) สถาปัตยกรรมแบบลูกข่ายแม่ข่ายสามชั้น

2.2.3.1 สถาปัตยกรรมแบบลูกข่ายแม่ข่ายสองชั้น

รูปที่ 11 แสดงให้เห็นการแบ่งชั้นเป็น 2 ชั้น (two-tier client/server architecture) คือชั้นลูกข่าย และชั้นแม่ข่าย ที่ทำงานร่วมกันผ่านระบบเครือข่าย โดยที่โปรแกรมติดต่อผู้ใช้ (user interface program) และแอปพลิเคชันต่าง ๆ จะทำงานอยู่บนฝั่งของเครื่องลูกข่าย และจะเข้าถึงฐานข้อมูลโดยใช้โอดีบีซี (Open Database Connectivity : ODBC) ซึ่งมีส่วนติดต่อเอพีไอ (Application Program Interface : API) ที่อนุญาตให้โปรแกรมฝั่งลูกข่ายสามารถเรียกฐานข้อมูลได้ ส่วนโปรแกรมลูกข่าย (client program) แต่ละโปรแกรมอาจเชื่อมต่อกับระบบฐานข้อมูลได้หลายระบบ นอกจากนี้ ลูกข่ายสามารถใช้ฟังก์ชันต่าง ๆ จากแม่ข่ายได้ เช่น ฟังก์ชันพจนานุกรมข้อมูล (data dictionary function) หรือการคืนสภาพ (recovery)

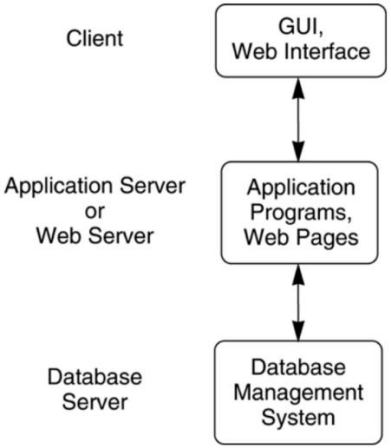


รูปที่ 11 สถาปัตยกรรมแบบลูกข่ายแม่ข่ายสองชั้น

2.2.3.2 สถาปัตยกรรมแบบลูกข่ายแม่ข่ายสามชั้น

รูปที่ 12 แสดงสถาปัตยกรรมแบบลูกข่ายแม่ข่ายสามชั้น (three-tier client/server architecture) มักใช้ในเว็บแอปพลิเคชัน (web application) โดยมีการเพิ่มชั้นกึ่งกลาง (intermediate layer) ระหว่างลูกข่ายและแม่ข่ายฐานข้อมูล (database server) ซึ่งเรียกชั้นกึ่งกลางนี้ว่าแม่ข่ายแอปพลิเคชัน (application server) หรือแม่ข่ายเว็บ (web server) โดยจะเป็นส่วนที่ทำหน้าที่เก็บซอฟต์แวร์การเชื่อมต่อเว็บ (web connectivity software) รวมถึงกฎและข้อบังคับต่าง ๆ ของแอปพลิเคชันในการเข้าถึงฐานข้อมูลในแม่ข่ายฐานข้อมูล

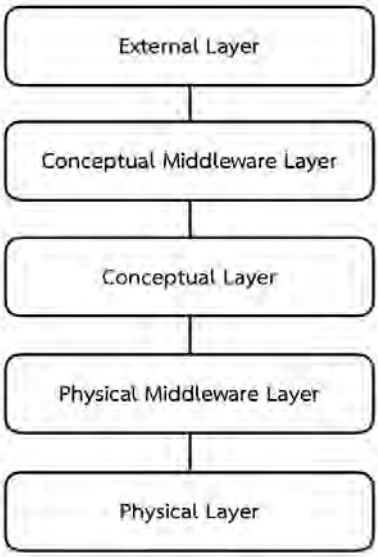
นอกจากนี้ยังมีลักษณะพิเศษเพิ่มเติมเกี่ยวกับความปลอดภัย โดยจะนำข้อมูลมาเข้ารหัส (encrypt) ที่เครื่องแม่ข่ายก่อนจะทำการส่งต่อออกไปถอดรหัส (decrypt) ที่เครื่องลูกข่ายด้วย



รูปที่ 12 สถาปัตยกรรมแบบลูกข่ายแม่ข่ายสามชั้น

2.2.4 สถาปัตยกรรมแบบคลาวด์

สถาปัตยกรรมแบบคลาวด์ (cloud architecture) หรือระบบการจัดการฐานข้อมูลแบบคลาวด์เป็นแนวปฏิบัติที่ได้รับความนิยมอย่างรวดเร็ว ระบบการจัดการฐานข้อมูลแบบคลาวด์นี้ทำงานบนแพลตฟอร์มการประมวลผลคลาวด์ (cloud computing platform) เช่น บริการเว็บของอเมซอน (Amazon Web Service : AWS) ที่ให้บริการทั้งการประมวลผล พื้นที่จัดเก็บข้อมูล การรักษาความปลอดภัย การสำรองข้อมูลและอื่น ๆ อีกมากมายที่ทำให้ลดความยุ่งยากในการติดตั้งและดูแลระบบ มีทั้งแบบบริการฟรีและแบบเก็บเงิน รูปที่ 13 แสดงสถาปัตยกรรมระบบการจัดการฐานข้อมูลแบบคลาวด์ [18]



รูปที่ 13 สถาปัตยกรรมแบบคลาวด์

2.3 ภาษาฐานข้อมูล

การจัดการข้อมูลภายในฐานข้อมูลนั้น เราจำเป็นต้องใช้ภาษาคอมพิวเตอร์ที่ออกแบบเฉพาะเจาะจงให้เหมาะกับการจัดการข้อมูล ภาษาฐานข้อมูลนี้แบ่งได้สองกลุ่ม [19, 20] คือ 1) ภาษาดีดีแอล และ 2) ภาษาดีเอ็มแอล

2.3.1 ภาษาดีดีแอล

ภาษาดีดีแอล (data definition language : DDL) เป็นภาษาที่ผู้ดูแลฐานข้อมูลและผู้ออกแบบฐานข้อมูลใช้ในการระบุสคีมาเชิงแนวคิดของฐานข้อมูล ซึ่งระบบการจัดการฐานข้อมูลส่วนใหญ่ก็มักใช้ภาษาดีดีแอลในการกำหนดสคีมาภายในและสคีมาภายนอกด้วยเช่นกัน นอกจากนี้ระบบการจัดการฐานข้อมูลบางตัว ยังมีการเพิ่มภาษาเอสดีแอล (storage definition language : SDL) และภาษาวีดีแอล (view definition language : VDL) ในการกำหนดสคีมาภายในและภายนอกโดยเฉพาะอีกด้วย แต่ก็ยังสามารถจัดอยู่ในดีดีแอลได้

2.3.2 ภาษาดีเอ็มแอล

ภาษาดีเอ็มแอล (data manipulation language : DML) ใช้ในการระบุการสืบค้นฐานข้อมูลต่าง ๆ เช่น การลบ การแทรก เป็นต้น โดยสามารถนำคำสั่งดีเอ็มแอลไปฝังติดไว้กับโปรแกรมที่เขียนด้วยภาษาใด ๆ เช่น ภาษาจาวา หรือภาษาซี นอกจากนี้ยังมีภาษาดีเอ็มแอลที่สามารถนำมาใช้สืบค้นข้อมูลโดยตรงคือ ภาษาสอบถาม ที่ใช้กันแพร่หลายในปัจจุบัน ภาษาดีเอ็มแอลสามารถแบ่งย่อยได้สองประเภทคือ 1) ดีเอ็มแอลระดับสูง เช่น ภาษาเอสคิว

แอลที่ทำงานแบบไม่ต้องระบุขั้นตอน (non-procedural) และ 2) ดีเอ็มแอลระดับต่ำ ที่ทำงานทีละคำสั่ง (record-at-a-time)

2.4 ประเภทของระบบการจัดการฐานข้อมูล

ประเภทของระบบการจัดการฐานข้อมูลสามารถพิจารณาได้จากหลายกฎเกณฑ์

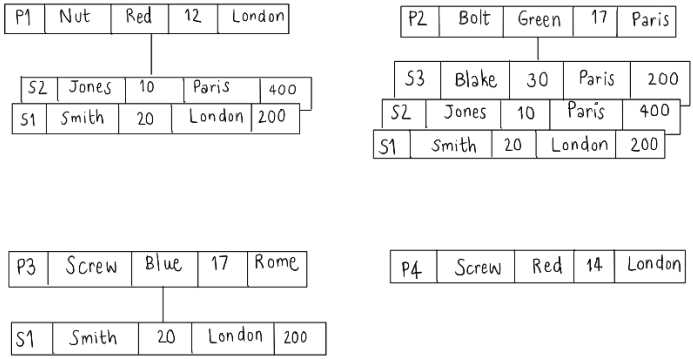
หากพิจารณาจากจำนวนผู้ใช้ จะแบ่งได้สองประเภทคือ 1) ผู้ใช้เดี่ยว ซึ่งมักใช้กับคอมพิวเตอร์ส่วนบุคคล และ 2) ผู้ใช้หลายคน ซึ่งระบบการจัดการฐานข้อมูลส่วนใหญ่จัดอยู่ในประเภทนี้

และหากพิจารณาจากลักษณะของไซต์ (site) หรือสถานที่ตั้ง [2] แบ่งได้สองประเภทคือ

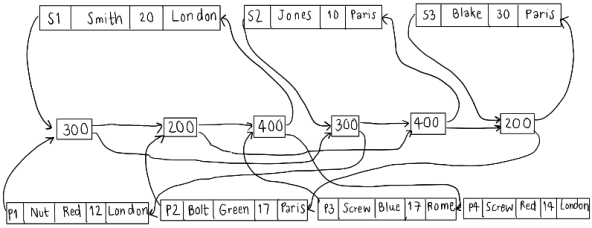
1) แบบรวมศูนย์ ใช้คอมพิวเตอร์หนึ่งเครื่องกับหนึ่งฐานข้อมูล
2) แบบกระจาย (distributed) ใช้คอมพิวเตอร์หลายเครื่องและหลายฐานข้อมูล สำหรับระบบฐานข้อมูลแบบกระจาย (distributed database system) นั้นก็คือระบบฐานข้อมูลที่ใช้สถาปัตยกรรมแบบลูกข่ายแม่ข่ายนั่นเอง ทั้งนี้ยังสามารถแบ่งย่อยได้เป็นแบบเอกพันธ์ (homogeneous) แบบวิวิธพันธ์ (heterogeneous) และแบบสหพันธ์ (federated หรือ multidatabase) ได้อีกด้วย

เมื่อพิจารณาจากโครงสร้างของฐานข้อมูล [21] จะแบ่งประเภทได้ 4 ประเภท ได้แก่

1) ฐานข้อมูลเชิงลำดับชั้น (hierarchical database) ดังแสดงตัวอย่างในรูปที่ 14
2) ฐานข้อมูลเชิงเครือข่าย (network database) แสดงตัวอย่างในรูปที่ 15
3) ฐานข้อมูลเชิงสัมพันธ์ (relational database) แสดงตัวอย่างในรูปที่ 16
4) ฐานข้อมูลเชิงวัตถุ (object-oriented database)



รูปที่ 14 ฐานข้อมูลเชิงลำดับชั้น



รูปที่ 15 ฐานข้อมูลเชิงเครือข่าย

S

S=	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris

SP

S=	P=	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S2	P1	300
S2	P2	400
S3	P2	200

P

P=	PNAME	COLOR	WEIGHT	CITY
P1	Nut	Red	12	London
P2	Bolt	Green	17	Paris
P3	Screw	Blue	17	Rome
P4	Screw	Red	14	London

รูปที่ 16 ฐานข้อมูลเชิงสัมพันธ์

อย่างไรก็ดี ได้มีระบบการจัดการฐานข้อมูลในยุคหลังเกิดขึ้นมาอีกหลายประเภท และก็หายหรือล้าสมัยไปอีกมาก ดังนั้น เราอาจแบ่งประเภทแบบใหม่ให้สอดคล้องกับยุคปัจจุบันได้ดังนี้ [22]

1. ระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ (relational DBMS) ที่พึ่งพาภาษาเอสคิวแอลเป็นหลักในการจัดการข้อมูล
2. ระบบการจัดการฐานข้อมูลโนเอสคิวแอล (NoSQL) ย่อมาจาก Not-Only-SQLหมายถึงไม่จำเป็นต้องพึ่งพาภาษาเอสคิวแอลเท่านั้น โดยที่โนเอสคิวแอลเป็นแนวทางในการจัดการข้อมูลที่เหมาะกับข้อมูลขนาดใหญ่ ที่อยู่อย่าง กระจัดกระจาย หลากหลายรูปแบบ ไม่มีโครงสร้างแน่ชัด เช่น ข้อมูลบนอินเทอร์เน็ต และยังประมวลผลได้รวดเร็วระดับเวลาจริง โนเอสคิวแอลแบ่งเป็นประเภทย่อยได้คือ

2.1. ฐานข้อมูลแบบเอกสาร (Document stores) เป็นฐานข้อมูลโนเอสคิวแอล ที่ออกแบบมาให้เหมาะกับการบันทึกข้อมูลรูปแบบใด ๆ เป็นชุดข้อความขนาดใหญ่ เช่น JSON structure ในลักษณะเอกสาร ตัวอย่างของ NoSQL database ผลิตภัณฑ์ที่เป็นที่รู้จักได้แก่ MongoDB และ CouchDB

2.2. ฐานข้อมูลค่าคีย์ (Key-value stores) เป็นฐานข้อมูลที่สามารถจัดเก็บข้อมูลรูปแบบใด ๆ ด้วยการระบุค่าคีย์ของข้อมูลนั้นๆ ตัวอย่างผลิตภัณฑ์ ได้แก่ Redis, Amazon S3 (Dynamo) และ Riak

2.3. ฐานข้อมูลคอลัมน์กว้าง (Wide-column stores) เป็นฐานข้อมูลที่ออกแบบให้จัดเก็บข้อมูลแบบคอลัมน์แทนการเก็บแบบบรรทัด ผลิตภัณฑ์ที่เป็นที่นิยมคือ Cassandra

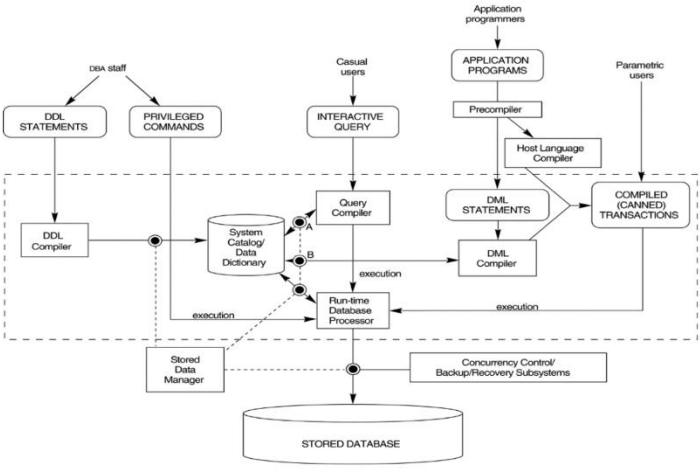
2.4. ฐานข้อมูลแบบกราฟ (Graph database) เป็นฐานข้อมูลที่ออกแบบมาให้ผู้ใช้สามารถระบุความสัมพันธ์ของข้อมูลแต่ละชุดในรูปแบบเครือข่ายหรือกราฟ ตัวอย่างของผลิตภัณฑ์ได้แก่ Neo4j

ทั้งนี้เรายังพบว่าฐานข้อมูลประเภทโนเอสคิวแอลจะเกิดแบบใหม่ ๆ ได้ค่อนข้างรวดเร็ว แต่อาจยังไม่เสถียรหรือแพร่หลายมากนัก จึงยังไม่นำมากล่าวในที่นี้

2.5 ภาพรวมของระบบการจัดการฐานข้อมูล

การทำงานภายในระบบการจัดการฐานข้อมูลมีรายละเอียดดังแสดงในรูปที่ 17 มาจาก [2] ให้สังเกตว่าส่วนบนของรูปคือผู้ใช้ระบบฐานข้อมูลทั้งหลาย ไม่ว่าจะเป็นโปรแกรมเมอร์ ผู้ดูแลฐานข้อมูล หรือผู้ใช้ทั่วไป เมื่อผู้ใช้ระดับต่าง ๆ ส่งคำสั่งเพื่อเรียกใช้ข้อมูลแล้ว ระบบการจัดการฐานข้อมูลจะส่งคำสั่งจากหน่วยย่อยหนึ่งไปยังหน่วยย่อยถัดไปตามลำดับในรูป

ขอยกตัวอย่างผู้ใช้ที่เป็นผู้ดูแลฐานข้อมูล (Database Administrator staff : DBA staff) ที่มักเขียนคำสั่งด้วยภาษาดีดีแอลเพื่อจัดการข้อมูล คำสั่งนั้นจะถูกส่งไปยังดีดีแอลคอมไพเลอร์ให้แปลงคำสั่งนั้นให้ระบบการจัดการฐานข้อมูลเข้าใจ แล้วส่งต่อไปยังแค็ตตาล็อกหรือพจนานุกรมข้อมูล เมื่อระบบการจัดการฐานข้อมูลเข้าใจแล้วว่าจะต้องไปดึงข้อมูลอะไรบ้าง ก็จะส่งต่อไปยังหน่วยประมวลผลฐานข้อมูลรันไทม์ (run-time database processor) ที่จะทำงานร่วมกับหน่วยจัดการการเก็บข้อมูล (stored data manager) และ หน่วยจัดการภาวะพร้อมกัน (concurrency control) ก่อนจะดึงข้อมูลจากหน่วยความจำแล้วแสดงคำตอบ



รูปที่ 17 การทำงานภายในระบบการจัดการฐานข้อมูล

ฟังก์ชันของระบบการจัดการฐานข้อมูลยังมีอีกมากมาย เช่น การแปลงข้อมูล (data conversion tool) การสำรองข้อมูล การปรับโครงสร้างฐานข้อมูล (reorganizing) การสร้างรายงาน (report generation) การติดตามประสิทธิภาพการทำงาน (performance monitoring) การเรียงข้อมูล (sorting) การดูแลตรวจสอบผู้ใช้ (user monitoring) หรือการบีบอัดข้อมูล (data compression) เป็นต้น

2.6 **บทสรุปและเรื่องชวนคิด**

การพัฒนาระบบฐานข้อมูลมีการเปลี่ยนแปลงไปมาก ผ่านมาหลายยุคหลายสมัย มีความซับซ้อนท้าทาย ทำให้สถาปัตยกรรมระบบคอมพิวเตอร์เปลี่ยนจนตามกันแทบไม่ทัน จนในปัจจุบันองค์กรขนาดใหญ่หลายแห่งก็ใช้บริการ

คลาวด์เพื่อเก็บและจัดการระบบการจัดการฐานข้อมูลของตนเอง เพราะอาจประหยัดทรัพยากรได้หลายด้าน ผู้เรียนคิดว่าปัจจัยอะไรบ้างที่ส่งผลในการตัดสินใจให้องค์กรเลือกใช้บริการระบบการจัดการฐานข้อมูลแบบคลาวด์ และแนวโน้มการเปลี่ยนแปลงนี้จะส่งผลต่องานในวงการคอมพิวเตอร์อย่างไรบ้าง

แบบฝึกหัดท้ายบท

1. จงอธิบายข้อแตกต่างระหว่างภาษาดีดีแอลและภาษาดีเอ็มแอล
2. ขั้นตอนการวิเคราะห์ความต้องการจากผู้ใช้ ประกอบด้วยสิ่งที่ต้องพิจารณาหลักกี่ประเด็น อะไรบ้าง เพราะเหตุใดจึงต้องเน้นที่ประเด็นหลักเหล่านี้
3. กระบวนการออกแบบและพัฒนาฐานข้อมูล ควรทำไปพร้อมช่วงใดในกระบวนการทางวิศวกรรมซอฟต์แวร์
4. การออกแบบและพัฒนาฐานข้อมูลควบคู่ไปกับการออกแบบและพัฒนาแอปพลิเคชันการพัฒนาระบบฐานข้อมูล มีกี่ขั้นตอน อะไรบ้าง
5. ระบบปฏิบัติการที่ออกแบบด้วยการออกแบบชนิดโมดูลา มีลักษณะการประมวลผลเป็นอย่างไร
6. เหตุใดองค์กรบางแห่งจึงไม่เลือกเป็นเจ้าของแม่ข่าย แต่ใช้บริการบนคลาวด์แทน
7. จงอธิบายความแตกต่างระหว่างสถาปัตยกรรมฐานข้อมูลแบบลูกข่ายแม่ข่ายสองชั้นกับสถาปัตยกรรมฐานข้อมูลแบบลูกข่ายแม่ข่ายสามชั้น
8. จงยกตัวอย่างเกณฑ์ในการแบ่งประเภทของระบบการจัดการฐานข้อมูลมา 3 ตัวอย่าง พร้อมบอกว่าถ้าแบ่งด้วยเกณฑ์นั้น จะแบ่งระบบการจัดการฐานข้อมูลได้กี่ประเภท อะไรบ้าง
9. ฐานข้อมูลแบบโนเอสคิวแอลเหมาะกับข้อมูลแบบใด
10. จงยกตัวอย่างฟังก์ชันของระบบการจัดการฐานข้อมูลมาพอสังเขป

บทที่ 3
การจำลองข้อมูล

วัตถุประสงค์

1. เพื่อให้เข้าใจวัตถุประสงค์และความสำคัญของการจำลองข้อมูล
2. เพื่อให้รู้จักแบบจำลองข้อมูลอีอาร์และอีอีอาร์
3. เพื่อให้สามารถระบุเอนทิตี รีเลชันชิป และแอตทริบิวต์
4. เพื่อให้สามารถออกแบบฐานข้อมูลด้วยแผนภาพอีอาร์และอีอีอาร์

การทำงานของระบบฐานข้อมูลในฮาร์ดแวร์ต่างระบบต่างยุคนั้น จำเป็นต้องใช้แบบจำลองข้อมูล (data model) ที่เหมาะสมกับอุปกรณ์และเครื่องมือที่ต่างกันไป แบบจำลองข้อมูลคือวิธีอธิบายโครงสร้างของฐานข้อมูล ซึ่งมีหลายแบบ สำหรับในตำรานี้ เราจะพูดถึงฐานข้อมูลเชิงสัมพันธ์ ดังนั้น แบบจำลองข้อมูลที่เราจะเน้น คือแบบจำลองข้อมูลเชิงสัมพันธ์ ซึ่งวิธีอธิบายโครงสร้างของฐานข้อมูลเชิงสัมพันธ์จะพูดถึง ชนิดของข้อมูล ความสัมพันธ์ของข้อมูล และเงื่อนไขบังคับต่าง ๆ ผลลัพธ์จากการทำแบบจำลองข้อมูลนี้ จะนำไปใช้ต่อในขั้นตอนถัดไป คือขั้นตอนการดำเนินการบนแบบจำลองฐานข้อมูล (data model operations) การดำเนินการ อาจแบ่งได้เป็น 2 ระดับคือ 1) การดำเนินการระดับพื้นฐาน (basic operation) ที่ระบบการจัดการฐานข้อมูลสามารถรองรับได้ เช่น การเพิ่ม การลบ การแก้ไข หรือการสืบค้นข้อมูล เป็นต้น และ 2) การดำเนินการตามที่ผู้ใช้ระบุ (user-defined operation) หมายถึง การดำเนินการที่โปรแกรมเมอร์เขียนตามความต้องการในการใช้ข้อมูล เช่น COMPUTE_GPA for STUDENT เป็นต้น

3.1 **การจำลองข้อมูล**

ในการออกแบบฐานข้อมูลนั้น เราจำเป็นที่จะต้องมีความเข้าใจและสามารถออกแบบฐานข้อมูลโดยใช้แบบจำลองข้อมูลเป็นเครื่องมือแสดงความคิดนั้น ๆ ใช้เป็นการสื่อสารระหว่างโปรแกรมเมอร์ด้วยกันหรือสื่อสารกับผู้ใช้ระบบ ซึ่งจะทำให้เราได้เห็นถึงพื้นฐานในการออกแบบโครงสร้างข้อมูลและเข้าใจถึงแบบจำลองของข้อมูลในแนวคิดระดับสูง

3.2 **แบบจำลองข้อมูล**

แบบจำลองข้อมูลมีหลากหลายแบบ มีความเหมาะสมในการใช้งานต่างกันไป ในที่นี้ ขอยกตัวอย่างแบบจำลองที่มักใช้กันอย่างแพร่หลาย 3 แบบ คือ

3.2.1 แบบจำลองข้อมูลเชิงแนวคิด

แบบจำลองข้อมูลเชิงแนวคิด (conceptual data model) เป็นแบบจำลองระดับสูง (high-level) ที่แสดงความหมาย (semantic) ของระบบฐานข้อมูลนั้น ๆ โดยแสดงแนวคิดของข้อมูลให้ใกล้เคียงกับความเข้าใจของผู้ใช้ทั่วไป ใช้ในการสื่อสารระหว่างโปรแกรมเมอร์กับผู้ใช้งานระบบ

เนื่องจากตำรานี้จะครอบคลุมการออกแบบระบบการจัดการฐานข้อมูล จึงเน้นการออกแบบเชิงแนวคิด

3.2.2 แบบจำลองข้อมูลเชิงกายภาพ

แบบจำลองข้อมูลเชิงกายภาพ (physical data model) เป็นแบบจำลองระดับต่ำ (low-level) หรือระดับภายใน (internal) เพื่อระบุรายละเอียดในการจัดเก็บข้อมูลลงในกายภาพของฐานข้อมูลนั้น ๆ เช่น ตำแหน่งของข้อมูลในหน่วยความจำ ขนาดและรูปแบบของข้อมูล เป็นต้น

3.2.3 แบบจำลองข้อมูลเชิงปฏิบัติการ

แบบจำลองข้อมูลเชิงปฏิบัติการ (implementation data model หรือ representational data model) แบบจำลองนี้ เป็นแบบจำลองที่อยู่

กึ่งกลางระหว่าง 2 แบบแรก มีการแสดงรวมรายละเอียดทั้ง 2 แบบแรกไว้ด้วยกัน อย่างไรก็ตาม แบบจำลองนี้ไม่ได้รับความนิยมเท่า 2 แบบแรก

3.3 **แบบจำลองอีอาร์**

ยุคของฐานข้อมูลเชิงสัมพันธ์ที่ใช้มาตั้งแต่ทศวรรษ 1970 มีพื้นฐานอยู่บนแบบจำลองเชิงสัมพันธ์ แบบจำลองนี้มีชื่อเรียกว่า แบบจำลองเอนทิตีรีเลชันชิป (Entity-Relationship model) [1, 10, 12-15, 23-25] หรือนิยมเรียกย่อได้ว่า แบบจำลองอีอาร์ (ER model) แบบจำลองอีอาร์นี้ออกแบบมาเพื่อให้ผู้ใช้อธิบายภาพรวมหรือแนวคิดของฐานข้อมูลนั้น โดยเน้นความสัมพันธ์ระหว่างข้อมูลในฐานข้อมูลนั้น ๆ องค์ประกอบสำคัญของรูปแบบอีอาร์ 3 ประการ คือ เอนทิตี รีเลชันชิป และแอตทริบิวต์ ตำราเล่มนี้เลือกใช้ทับศัพท์ภาษาอังกฤษ เพราะเชื่อว่าจะทำให้ลดความสับสนได้

1. เอนทิตี (entity) ศัพท์บัญญัติราชบัณฑิตยสถาน ให้ใช้ว่า เอนทิตี
2. แอตทริบิวต์ (attribute) ศัพท์บัญญัติราชบัณฑิตยสถานคือ ลักษณะประจำ แต่เนื่องจากเป็นศัพท์ที่ใช้แสดงสัญลักษณ์เฉพาะทาง ตำรานี้จึงขอใช้ทับศัพท์ว่า แอตทริบิวต์
3. รีเลชันชิป (relationship) ทั่วไปหมายถึงความสัมพันธ์ แต่เนื่องจากเป็นศัพท์ที่ใช้แสดงสัญลักษณ์เฉพาะทาง จึงขอทับศัพท์ว่า รีเลชันชิป

เราสามารถเขียนแบบจำลองความสัมพันธ์ระหว่างเอนทิตีกับรีเลชันชิป รวมทั้งแอตทริบิวต์ด้วยแผนภาพ และแผนภาพที่ถูกออกแบบมาสำหรับแบบจำลองนี้คือ แผนภาพความสัมพันธ์ระหว่างเอนทิตีกับรีเลชันชิป (Entity-Relationship diagram หรือ ER diagram) ต่อไปนี้จะเรียกว่า แผนภาพ

อีอาร์ที่แสดงให้เห็นถึงเอนทิตีและรีเลชันชิปของข้อมูลได้อย่างง่ายดาย ไม่ซับซ้อน

เพื่อให้เกิดความเข้าใจที่ชัดเจน ขอใช้แผนภาพอีอาร์เพื่อแสดงแบบจำลองฐานข้อมูลจากตัวอย่างระบบฐานข้อมูลของบริษัทแห่งหนึ่ง ให้ชื่อว่าบริษัทดีบี และใช้รูปแบบการเขียนที่นิยมในวงการฐานข้อมูล คือการใช้อักษรภาษาอังกฤษตัวใหญ่แสดงชื่อของข้อมูลโดยเฉพาะเอนทิตี ซึ่งโจทย์และผลลัพธ์จากนี้นำมาจาก [2]

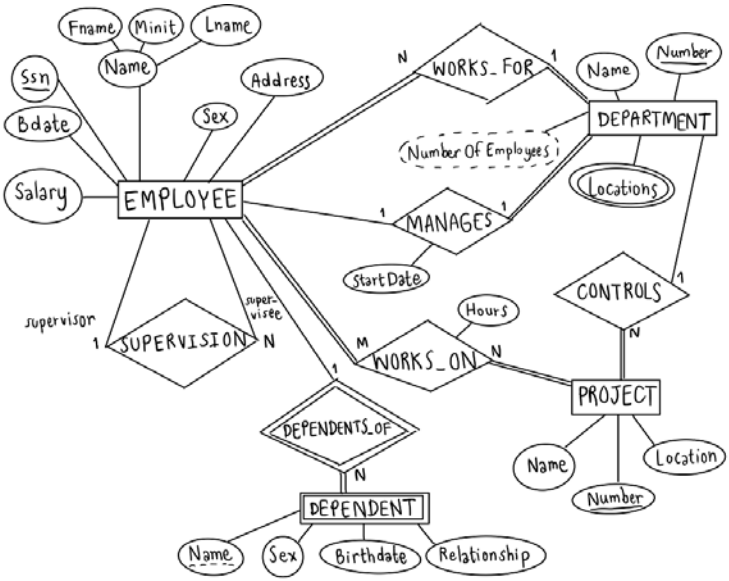
โจทย์กำหนดให้ดังนี้

- บริษัทดีบีมีการจัดระบบโดยเริ่มจากการแบ่งเป็นแต่ละแผนก ซึ่งเราจะเรียกว่า DEPARTMENT ถือว่าเป็นหน่วยย่อยของบริษัท โดยให้ DEPARTMENT มีชื่อและรหัสที่แตกต่างกัน
- แต่ละ DEPARTMENT มีพนักงานหลายคน ซึ่งเราจะเรียกว่า EMPLOYEE ทั้งนี้ EMPLOYEE แต่ละคนสามารถทำงานได้ใน DEPARTMENT เดียวเท่านั้น
- แต่ละ DEPARTMENT มีผู้จัดการ ซึ่งเราจะเรียกว่า MANAGER แต่ละ DEPARTMENT มีผู้จัดการหนึ่งคนเท่านั้น และเราต้องการเก็บข้อมูลวันที่เริ่มต้นเป็นผู้จัดการ DEPARTMENT นั้นไว้
- แต่ละ DEPARTMENT อาจมีสถานที่ตั้งได้หลายที่ ซึ่งเราจะเรียกสถานที่ว่า LOCATION
- นอกจากนั้น EMPLOYEE แต่ละคนจะทำงานในโครงการ ซึ่งเราจะเรียกว่า PROJECT แต่ละคนทำได้คนละหลาย PROJECT ซึ่งแต่ละ

PROJECT จะมีชื่อ เลขที่ และ LOCATION โดยแต่ละ PROJECT นั้นตั้งอยู่ใน LOCATION เดียวเท่านั้น

- บริษัทต้องการเก็บว่า EMPLOYEE แต่ละคนทำงานให้แต่ละ PROJECT กี่ชั่วโมงด้วย
- แต่ละ DEPARTMENT มีหน้าที่ควบคุม PROJECT ซึ่งมีได้หลาย PROJECT ต่อ DEPARTMENT
- ในส่วนของ EMPLOYEE บริษัทต้องการจัดเก็บข้อมูลของ EMPLOYEE คือ ชื่อพนักงาน รหัสประจำตัว ที่อยู่ เงินเดือน เพศ และวันเกิด
- บริษัทมีการกำหนดผู้คุมงาน ซึ่งเราจะเรียกว่า SUPERVISOR และต้องการบันทึกว่า EMPLOYEE คนใดเป็น SUPERVISOR ของคนใดอีกด้วย
- นอกจากนี้ยังมีการเก็บข้อมูลผู้อยู่ในอุปการะของ EMPLOYEE ซึ่งเราจะเรียกว่า DEPENDENT เช่น สามี ภรรยา บุตร หรือ คนในครอบครัว ซึ่ง EMPLOYEE แต่ละคน อาจมี DEPENDENT ได้หลายคน ข้อมูล DEPENDENT ที่สนใจคือ ชื่อ เพศ วันเกิด และความสัมพันธ์กับ EMPLOYEE

จากข้อมูลที่โจทย์กำหนด เราสามารถเขียนแผนภาพอีอาร์ได้ดังแสดงในรูปที่ 18



รูปที่ 18 แผนภาพอีอาร์แสดงระบบฐานข้อมูลของบริษัทดีบี

3.3.1 เอนทิตี

เอนทิตี (entity) แสดงในภาพด้วยกล่องสี่เหลี่ยมผืนผ้า พร้อมชื่อกำกับภายใน คือ สิ่งใด ๆ ที่เราให้ความสำคัญ และเป็นส่วนที่เราต้องการนำเสนอในฐานข้อมูล โดยอาจจะเป็นได้ทั้ง คน สัตว์ สิ่งของ หรือสรรพสิ่งใด ๆ ถ้ายกตัวอย่างตามแผนภาพอีอาร์ในรูปที่ 18 จะเห็นตัวอย่างของเอนทิตี เช่น EMPLOYEE คือพนักงานแต่ละคน เช่น

สมชาย รักการเรียน, สมหญิง รักการงาน, สมกมล รักการบ้าน

เมื่อนำแต่ละเอนทิตีที่มีลักษณะร่วมกันมารวมกัน เราจะเรียกว่า เอนทิตีไทป์ (entity type) แปลเป็นไทยว่าแบบเอนทิตี แต่เราขอทับศัพท์ว่า เอนทิตีไทป์ เพื่อให้ไม่ต้องสลับระหว่างไทยอังกฤษในศัพท์เดียวกันที่น่าจะทำให้เข้าใจยาก

ดังนั้น DEPARTMENT ที่ 1 หรือ DEPARTMENT ที่ 2 เรียกว่าเป็น เอนทิตี และเมื่อทุก DEPARTMENT มารวมกัน ให้เรียกว่าเป็น เอนทิตีไทป์ DEPARTMENT

ในทำนองเดียวกัน PROJECT แต่ละ PROJECT ก็เป็นเอนทิตี เมื่อเอา เอนทิตี PROJECT มารวมกันทั้งหมด ก็เรียกว่า เอนทิตีไทป์ PROJECT

สรุปจากตัวอย่างนี้ก็จะมี เอนทิตีไทป์ EMPLOYEE, DEPARTMENT, PROJECT และ DEPENDENT

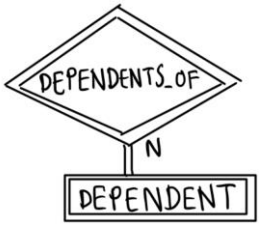
สังเกตจากแผนภาพ จะเห็นว่า เราแทนเอนทิตีไทป์ด้วยกล่องสี่เหลี่ยมผืนผ้า และเขียนชื่อเอนทิตีไทป์ไว้ในกล่องด้วยตัวพิมพ์ใหญ่

ตามที่ได้กล่าวไปแล้วข้างต้นว่า เอนทิตีที่มีแอตทริบิวต์พื้นฐานเหมือนกัน จะถูกจัดกลุ่มหรือจัดประเภทเป็นเอนทิตีไทป์เดียวกัน เช่น เอนทิตีไทป์ EMPLOYEE เอนทิตีไทป์ PROJECT ส่วนเอนทิตีแต่ละเอนทิตีเมื่อมารวมกัน จะเรียกว่า กลุ่มเอนทิตี (entity set) หรือ เอกซ์เทนชัน (extension)

จากแผนภาพอีอาร์ที่แสดงไว้ก่อนหน้านี้ จะพบเอนทิตีไทป์ 2 ประเภท ได้แก่

1. เอนทิตีไทป์ปกติ (regular entity type) ให้เรียกว่า เอนทิตีไทป์
2. วีคเอนทิตีไทป์ (weak entity type)

เอนทิตีไทป์ปกติที่ปรากฏในแผนภาพอีอาร์มี 3 ตัว คือ EMPLOYEE, DEPARTMENT และ PROJECT ส่วน DEPENDENT นั้นเป็นวีคเอนทิตีไทป์ แสดงด้วยกล่องสี่เหลี่ยมผืนผ้าเส้นคู่ หมายความว่าเป็นเอนทิตีที่ไม่มีแอตทริบิวต์ใดเป็นคีย์ (Key attribute) ของตนเองที่จะทำให้ข้อมูลในฐานนั้นไม่ซ้ำ แต่อาจมี คีย์บางส่วน (partial key) ได้ ให้แสดงคีย์บางส่วนด้วยเส้นใต้แบบประ ดังนั้นวีคเอนทิตีไทป์จำเป็นต้องใช้คีย์จากเอนทิตีไทป์ที่มีแอตทริบิวต์เชื่อมถึงกัน เรียกเอนทิตีไทป์ดังกล่าวนี้ว่า เอนทิตีไทป์เจ้าของ (owner entity type) หรือเอนทิตีไทป์ระบุ (identifying entity type) และเรียกความสัมพันธ์ที่เชื่อมต่อระหว่างวีคเอนทิตีไทป์กับชนิดระบุว่า รีเลชันชิปไทป์ระบุ (identifying relationship type) โดยวีคเอนทิตีไทป์นี้จะใช้คีย์จากเอนทิตีไทป์ระบุมารวมกับคีย์บางส่วน (partial key) ของตัวเอง เพื่อใช้เป็นคีย์ที่ทำให้ข้อมูลทุกบรรทัดในฐานข้อมูลนั้นไม่ซ้ำกัน จากรูปที่ 19 จะเห็นว่า DEPENDENT ต้องนำคีย์มาจาก EMPLOYEE คือ ID มาเป็นคีย์ร่วม เราเรียก EMPLOYEE ว่าเอนทิตีชนิดระบุ เรียก DEPENDENTS_OF ว่า รีเลชันชิปไทป์ระบุ



รูปที่ 19 วีคเอนทิตีไทป์

3.3.2 แอตทริบิวต์

ในแผนภาพอีอาร์ เราใช้วงรีพร้อมชื่อภายในวงรีเพื่อแสดงแอตทริบิวต์ เอนทิตีทุกเอนทิตีจะมีแอตทริบิวต์ จากตัวอย่าง จะเห็นได้ว่า สมชาย รักการเรียน มีเลขบัตรประจำตัว เพศ ที่อยู่ วันเกิด และเงินเดือน ที่มีค่าดังนี้ เลขบัตรประจำตัว = 1234567891012, เพศ = ชาย, ที่อยู่ = 10 ถนนพระรามสี่ บางรัก กทม 10500, วันเกิด = 10-5-1995 และเงินเดือน = 25,000

สมหญิง รักการงาน, เลขบัตรประจำตัว = 1234567891015, เพศ = หญิง, ที่อยู่ = 15 ถนนพระรามสี่ บางรัก กทม 10500, วันเกิด = 10-6-1992 และเงินเดือน = 35,000

สมกมล รักการบ้าน, เลขบัตรประจำตัว = 3234567891012, เพศ = หญิง, ที่อยู่ = 20 ถนนพระรามสี่ บางรัก กทม 10500, วันเกิด = 2-3-1985 และเงินเดือน = 250,000

ทั้งชื่อ นามสกุล เลขบัตรประจำตัว เพศ ที่อยู่ วันเกิด และเงินเดือน คือ แอตทริบิวต์ หรือในแผนภาพที่แสดงในรูปที่ 18 ให้ชื่อแอตทริบิวต์เหล่านี้ว่า Name, Lastname, ID, Address, Birthdate, Salary

แอตทริบิวต์จะมีค่า (value) และมีชนิดของข้อมูล (value set) ที่เหมาะสม เช่น จำนวนเต็ม (integer) ข้อความ (string) ตัวเลข (numeric) เป็นต้น

สังเกตจากแผนภาพในรูปที่ 18 จะเห็นว่า เราแทนแอตทริบิวต์ด้วยวงรี ส่วนชื่อแอตทริบิวต์จะขึ้นต้นด้วยตัวพิมพ์ใหญ่แล้วตามด้วยตัวพิมพ์เล็กหรืออาจแทรกตัวพิมพ์ใหญ่ในคำหลังเพื่อให้อ่านได้ง่าย เช่น StreetAddress เป็นต้น

ทั้งนี้ ทุก ๆ เอนทิตีควรมีแอตทริบิวต์ที่เป็นคีย์ เรียกว่า คีย์แอตทริบิวต์ (key attribute) ที่ทำให้ข้อมูลแต่ละบรรทัดในฐานข้อมูลนั้นไม่ซ้ำกัน เช่น เอนทิตี EMPLOYEE มี EmployeeID เป็นคีย์ของเอนทิตีนั่นเอง ให้ขีดเส้นใต้ชื่อแอตทริบิวต์เพื่อแสดงความเป็นคีย์

อย่างไรก็ดี มีความเป็นไปได้ที่เอนทิตีหนึ่ง ๆ อาจมีคีย์มากกว่าหนึ่งคีย์ เช่น เอนทิตีไทป์ CAR อาจมีคีย์คือ Vin หมายถึงเลขตัวถังรถ (Vehicle Identification Number), หรือ LicensePlate หมายถึง ทะเบียนรถอันประกอบด้วยเลขทะเบียนกับจังหวัด (RegistrationNumber, Province) เป็นต้น

นอกจากนั้น คีย์แอตทริบิวต์ ไม่จำเป็นต้องเป็นแอตทริบิวต์เดี่ยว หากแอตทริบิวต์เพียงตัวเดียวไม่ทำให้ข้อมูลแต่ละบรรทัดไม่ซ้ำ จึงจะต้องกำหนดแอตทริบิวต์หลายตัวมาเป็นคีย์ของเอนทิตีนั้นให้ได้ เช่น LicensePlate ที่ยกตัวอย่างข้างต้น เป็นคีย์ที่ประกอบด้วยแอตทริบิวต์ 2 แอตทริบิวต์ คือ RegistrationNumber กับ Province เป็นต้น

ดังนั้น จากแผนภาพอีอาร์ตัวอย่างในรูปที่ 18 เราสามารถแจกแจงเอนทิตีและแอตทริบิวต์ได้ดังแสดงในรูปที่ 20 คือมีเอนทิตีไทป์ 4 ตัว ได้แก่ DEPARTMENT, PROJECT, EMPLOYEE แบะ DEPENDENT หากเขียนแทนด้วยรูปแบบนี้จะไม่สามารถแยกได้ว่าเป็นเอนทิตีไทป์หรือวีคเอนทิตีไทป์

แต่ละเอนทิตีไทป์ทั้งชนิดธรรมดาและชนิดวีค จะประกอบด้วยแอตทริบิวต์ต่าง ๆ จากรูปนี้จะเห็นว่าเราสามารถใช้เครื่องหมายวงเล็บปีกกา {} เพื่อแสดงว่าเป็นแอตทริบิวต์หลายค่า และใช้เครื่องหมายวงเล็บ () แสดงแอตทริบิวต์ร่วมได้อีกด้วย

DEPARTMENT
Name, Number, {Locations}, Manager, ManagerStartDate

PROJECT
Name, Number, Location, ControllingDepartment

EMPLOYEE
Name (FName, MInit, LName), SSN, Sex, Address, Salary,
BirthDate, Department, Supervisor, {WorksOn (Project, Hours)}

DEPENDENT
Employee, DependentName, Sex, BirthDate, Relationship

รูปที่ 20 เอนทิตีและแอตทริบิวต์จากแผนภาพอีอาร์บริษัทดีบี

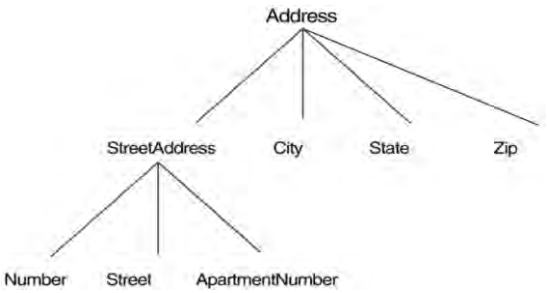
แอตทริบิวต์มีหลายประเภท ขึ้นอยู่กับมุมมองในการพิจารณาแยกประเภท เราจะสามารถแบ่งออกได้หลายแบบ

แบ่งตามลักษณะการแยกย่อย สามารถจำแนกได้เป็นสองประเภท คือ

1. แอตทริบิวต์อย่างง่าย (simple attribute)
2. แอตทริบิวต์ร่วม (composite attribute)

โดยแอตทริบิวต์อย่างง่ายจะไม่สามารถจะแบ่งเป็นส่วนประกอบย่อย ๆ ของแอตทริบิวต์ลงไปได้อีก เช่น เมื่อพิจารณาเอนทิตี EMPLOYEE เราจะพบว่าแอตทริบิวต์ ID และ SEX เป็นแอตทริบิวต์อย่างง่าย คือ ไม่สามารถแยกย่อยลงไปได้อีกในขณะที่แอตทริบิวต์ร่วมเป็นแอตทริบิวต์ที่สามารถแยกย่อยลงไปได้อีก เช่น ADDRESS จะสามารถแยกส่วนย่อยได้เป็น (StreetAddress,

City, State, Zip) เป็นต้น ให้แสดงแอตทริบิวต์ร่วมด้วยการเขียนชุดของแอตทริบิวต์ร่วมแต่ละตัวด้วยวงรีพร้อมชื่อเหมือนเดิม แต่ให้วงรีเหล่านี้แตกเส้นออกจากแอตทริบิวต์ต้นของชุดนั้น ในขณะเดียวกันลักษณะของแอตทริบิวต์ร่วมนั้นสามารถที่จะประกอบซ้อนกันได้มากกว่าหนึ่งชั้นด้วยเช่นเดียวกัน เพราะในกรณีของ StreetAddress ยังสามารถแยกย่อยออกได้เป็น (Number, Street, ApartmentNumber) ดังตัวอย่างในรูปที่ 21



รูปที่ 21 แอตทริบิวต์ร่วม

แบ่งตามค่า สามารถจำแนกได้เป็นสองประเภท คือ

1. แอตทริบิวต์ค่าเดี่ยว (single-valued attribute)
2. แอตทริบิวต์หลายค่า (multi-valued attribute)

โดยแอตทริบิวต์ค่าเดียวจะจัดเก็บค่าข้อมูลเพียงค่าเดียว เช่น AGE เก็บค่าอายุเพียงค่าเดียว หรือ SALARY เก็บค่าเงินเดือนเพียงค่าเดียว ในขณะที่ LOCATION เป็นแอตทริบิวต์หลายค่า จึงเก็บค่าสถานที่ตั้งได้หลายค่า เราแสดงแอตทริบิวต์หลายค่าด้วยวงรีเส้นคู่

ตัวอย่างเพิ่มเติมเช่น หากเราต้องการออกแบบให้เก็บข้อมูลเลขหมายโทรศัพท์ของคนคนหนึ่งได้มากกว่าหนึ่งเลขหมาย ก็สามารถกำหนดใช้แอตทริบิวต์แบบหลายค่าได้

แบ่งตามการจัดเก็บ จะได้แอตทริบิวต์สองประเภทคือ

1. แอตทริบิวต์ค่าบันทึก (stored attribute)
2. แอตทริบิวต์ค่าคำนวณ (derived attribute)

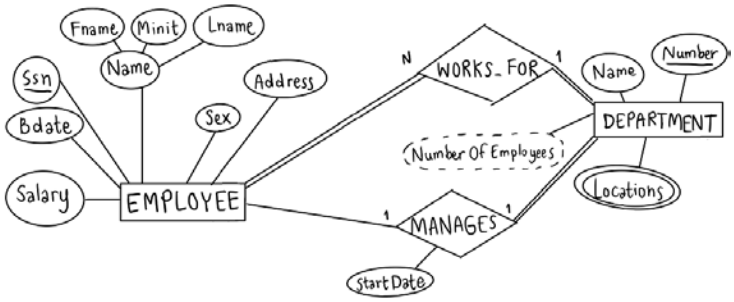
โดยแอตทริบิวต์ค่าบันทึกจะจัดเก็บค่าที่ผู้ใช้งานหรือโปรแกรมเมอร์ระบุหรือพิมพ์เข้ามาเก็บในฐานข้อมูล เช่น BIRTHDATE, NAME, LNAME จำเป็นต้องมีการระบุค่าจากผู้ใช้ ส่วนแอตทริบิวต์ค่าคำนวณ จะไม่มาจากผู้ใช้ แต่จะเกิดจากการคำนวณค่ามาจากแอตทริบิวต์อื่นแล้วมาเก็บไว้ เช่น AGE คำนวณจาก BIRTHDATE ได้เสมอ และทำให้ลดความผิดพลาดได้ดี ให้แสดงแอตทริบิวต์ค่าคำนวณด้วยวงรีเส้นประ

สังเกตว่าแต่ละแอตทริบิวต์ก็จะมีค่าเก็บไว้ ไม่ว่าจะมาจากการคำนวณหรือผู้ใช้ใส่เข้ามาก็ดี แต่ก็มีกรณีที่แอตทริบิวต์หนึ่ง ๆ อาจไม่มีค่าที่ควรจัดเก็บหมายถึง อาจไม่มีข้อมูลที่เหมาะสม หรืออาจไม่รู้ข้อมูลนั้นก็ได้ ในกรณีนี้ฐานข้อมูลจะเก็บค่าพิเศษเรียกว่า ค่าว่าง (null value) ค่าว่างนี้สำคัญมากในการจัดเก็บข้อมูลลงในฐานข้อมูล ในกรณีแอตทริบิวต์ใดไม่มีค่าที่สามารถจัดเก็บได้ ให้กำหนดเป็น ค่าว่าง หรือ นัล (null) เพื่อแสดงความว่างนั้น เช่น จากรูปที่ 21 มีแอตทริบิวต์ StreetAddress ซึ่งแตกเป็น Number, Street และ ApartmentNumber ที่ออกแบบไว้เก็บหมายเลขห้องของอพาร์ตเมนต์ที่อยู่ แต่เนื่องจากไม่ใช่ทุกคนจะมีที่อยู่อาศัยแบบอพาร์ตเมนต์ จึงอาจไม่มี

ข้อมูล ApartmentNumber เช่นกรณีอย่างนี้ ก็จะเก็บค่า นัล ไว้ในแอตทริบิวต์ดังกล่าวแทน ห้ามเก็บค่า 0 หรือเคาะวรรค เพราะฐานข้อมูลจะเข้าใจว่าเก็บค่าศูนย์หรือวรรคแล้วส่งผลให้ข้อมูลผิดพลาดได้

3.3.3 รีเลชันชิป

รีเลชันชิป ทำหน้าที่เชื่อมโยงระหว่างเอนทิตีไทป์ตั้งแต่ 2 เอนทิตีไทป์ขึ้นไป เขียนแสดงด้วยสี่เหลี่ยมข้าวหลามตัดพร้อมระบุชื่อข้างใน แต่ละรีเลชันชิปจะมีชื่อสื่อถึงวัตถุประสงค์ของรีเลชันชิปนั้น ๆ เช่น เราอาจสร้างรีเลชันชิประหว่าง EMPLOYEE กับ DEPARTMENT ให้มีความสัมพันธ์กันเมื่อ มี EMPLOYEE เป็นพนักงานอยู่ใน DEPARTMENT เป็นต้น ในกรณีนี้รีเลชันชิประหว่าง EMPLOYEE กับ DEPARTMENT ไม่จำเป็นต้องมีรีเลชันชิปแบบเดียว เราสามารถแสดงรีเลชันชิปที่สอดคล้องกับบทบาทรีเลชันชิปเหล่านั้นได้ ขอยกตัวอย่างประกอบดังแสดงในรูปที่ 22



รูปที่ 22 รีเลชันชิปที่สอดคล้องกับบทบาท

สมมุติว่าเราต้องการแสดงรีเลชันชิประหว่าง EMPLOYEE ว่าใครทำงานใน DEPARTMENT ใด ให้สร้างรีเลชันชิปชื่อว่า WORKS_FOR เชื่อมระหว่าง EMPLOYEE และ DEPARTMENT

ทีนี้เราอาจกำหนดรีเลชันชิประหว่างเอนทิตีไทป์ชุดเดิม ที่ต่างบทบาทความสัมพันธ์ก็ได้ เช่น ต้องการแสดงว่า EMPLOYEE สมหญิง เป็นผู้จัดการ DEPARTMENT การเงิน ก็ให้สร้างรีเลชันชิประหว่าง EMPLOYEE กับ DEPARTMENT ชื่อ MANAGES เป็นต้น

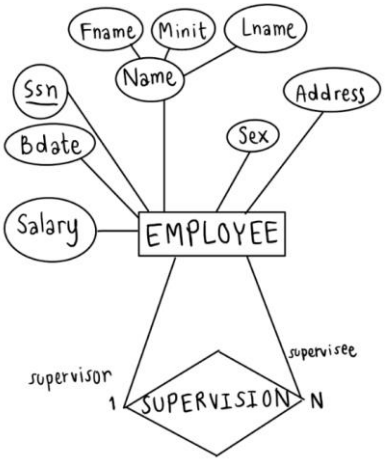
เมื่อนำข้อมูลรีเลชันชิปประเภทเดียวกันมารวมไว้ด้วยกัน จะเรียกว่า รีเลชันชิปไทป์ (relationship type) ซึ่งเป็นไปในทำนองเดียวกันกับเอนทิตีไทป์ จากตัวอย่างข้างบน เราต้องเรียก WORKS_ON และ MANAGES ว่าเป็นรีเลชันชิปไทป์

จากแผนภาพอีอาร์ก่อนหน้านี้ สรุปได้ว่ามีรีเลชันชิปไทป์ทั้งหมด 6 แบบความสัมพันธ์ คือ WORKS_FOR, MANAGES, CONTROLS, WORKS_ON, SUPERVISION และ DEPENDENTS_OF

สังเกตว่าการเขียนชื่อแบบความสัมพันธ์จะใช้ตัวพิมพ์ใหญ่ทั้งหมด เหมือนกับการเขียนชื่อเอนทิตีไทป์

ส่วนรีเลชันชิปไทป์ DEPENDENTS_OF นั้น เขียนด้วยสัญลักษณ์ สี่เหลี่ยมข้าวหลามตัดเส้นคู่ เพื่อแสดงความเชื่อมโยงกับ วีคเอนทิตีไทป์ เราจึงเรียกรีเลชันชิปนี้ว่า วีครีเลชันชิป (weak relationship)

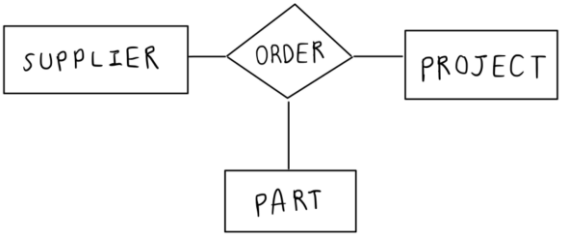
ยังมีความสัมพันธ์แบบพิเศษ คือแบบที่เชื่อมความสัมพันธ์วนกลับไปยังเอนทิตีไทป์เดิม เรียกเหตุการณ์นี้ว่า เกิดความสัมพันธ์แบบเวียนเกิด (recursive relationship) จากตัวอย่างในรูปที่ 23 จะเห็นรีเลชันชิปแบบเวียนเกิด คือ SUPERVISION ที่มีเอนทิตีไทป์ทั้งสองข้างเป็นเอนทิตีไทป์เดียวกัน คือ EMPLOYEE นั่นเอง



รูปที่ 23 รีเลชันชิปแบบเวียนเกิด

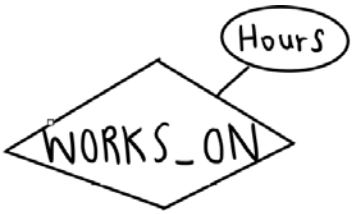
รีเลชันชิปไทป์จะมีระดับหรือดีกรีความสัมพันธ์ (degree of relationship type) กำหนดจากจำนวนเอนทิตีไทป์ที่อยู่รอบข้าง หากรีเลชันชิปไทป์นั้นเชื่อมเอนทิตีไทป์ไว้ 2 เอนทิตีไทป์ ก็จะกล่าวได้ว่ามีดีกรีรีเลชันชิปไทป์เป็น 2 หรือเรียกว่าแบบไบนารี (binary) ซึ่งจากแผนภาพอีอาร์ จะเห็นว่าทั้ง MANAGES และ WORKS_ON ต่างก็มีดีกรีเท่ากับ 2 หรือเรียกว่ารีเลชันชิปแบบไบนารี

หากรีเลชันชิปไทป์มีการเชื่อมโยง 3 เอนทิตีไทป์เข้าด้วยกัน จะมีดีกรีเท่ากับ 3 หรือเรียกว่าเทอนารี (ternary) แสดงในรูปที่ 24 และมากกว่าสาม ก็จะเรียกว่า เอนนารี (n-ary)



รูปที่ 24 รีเลชันชิปแบบเทอนารี

ที่ผ่านมาเราได้พูดถึงแต่แอตทริบิวต์ปกติทั่วไปก็คือแอตทริบิวต์ของเอนทิตี แต่ยังมีแอตทริบิวต์ที่เกิดจากรีเลชันชิปได้อีกด้วย สังเกตจากแผนภาพอีอาร์จะเห็นแอตทริบิวต์ Hours ดังแสดงในรูปที่ 25 ที่เกิดจากรีเลชันชิประหว่าง EMPLOYEE กับ PROJECT เพราะบันทึกเวลาการทำงานของ EMPLOYEE ใน PROJECT นั้น ๆ ข้อมูลนี้จะไม่เกิดขึ้น หากไม่มีรีเลชันชิประหว่างเอนทิตีไทป์ทั้งสอง



รูปที่ 25 แอตทริบิวต์ของรีเลชันชิป

ยังมีเรื่องที่สำคัญที่ยังไม่ได้กล่าวถึงอีกเรื่อง คือเราสามารถระบุเงื่อนไขบังคับให้กับรีเลชันชิปไทป์ใด ๆ ทั้งนี้เพื่อชี้ว่ารีเลชันชิปนั้น ๆ สามารถเกิดขึ้นได้กี่ครั้ง เงื่อนไขบังคับที่สำคัญมี 2 อย่างคือ

- คาร์ดินาลิตี
- เงื่อนไขบังคับเข้าร่วม

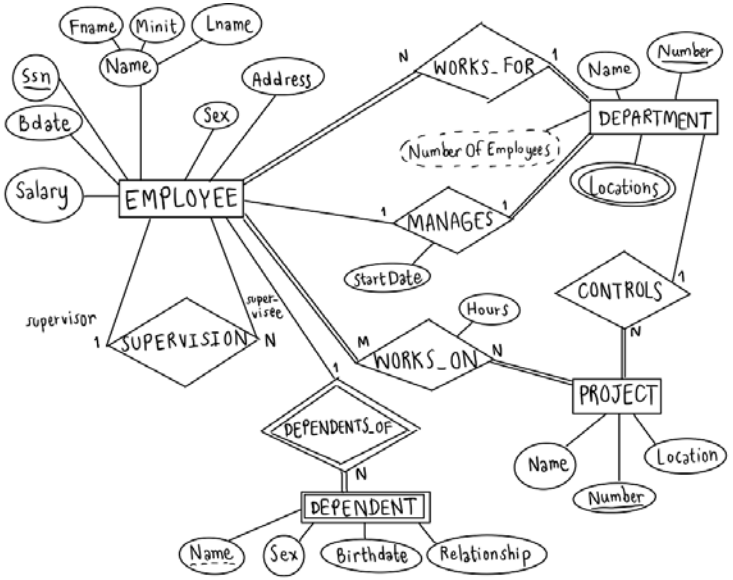
3.3.4 คาร์ดินาลิตี

คาร์ดินาลิตี (cardinality) คือสัดส่วน (ratio) ของรีเลชันชิป ซึ่งเป็นตัวบ่งชี้ว่าจำนวนรีเลชันชิปมากที่สุดที่แต่ละเอนทิตีสามารถเข้าร่วมในรีเลชันชิปนั้นได้มีค่าเท่าใด ซึ่งมีความเป็นไปได้ 3 แบบ คือ

- 1:1 แสดงรีเลชันชิปแบบหนึ่งต่อหนึ่ง นิยมอ่านว่า วันทูวัน (one-to-one)
- 1:N หรือ N:1 แสดงรีเลชันชิปแบบหนึ่งต่อหลายหรือหลายต่อหนึ่ง นิยมอ่านว่า วันทูเมนี (one-to-many) หรือ เมนีทูวัน (many-to-one)
- M:N แสดงรีเลชันชิปแบบหลายต่อหลาย นิยมอ่านว่า เมนีทูเมนี (many-to-many)

ขอยกตัวอย่างประกอบเพื่อให้เข้าใจได้ง่ายขึ้น ให้ดูแผนภาพอีอาร์เดิมที่นำมาแสดงอีกครั้งในรูปที่ 26 จะเห็นรีเลชันชิปไทป์ MANAGES ที่เชื่อมระหว่าง EMPLOYEE กับ DEPARTMENT ในกรณีนี้ จะเห็นเลข 1 กำกับอยู่บนเส้นที่เชื่อมระหว่างรีเลชันชิปไทป์นั้นไปยัง EMPLOYEE และ DEPARTMENT ซึ่ง

หมายความว่า EMPLOYEE 1 คน จะ MANAGE ได้อย่างมาก 1 DEPARTMENT และ แต่ละ DEPARTMENT มี EMPLOYEE ที่เป็นผู้จัดการ หรือ MANAGE ได้อย่างมากคนเดียวเท่านั้น



รูปที่ 26 แผนภาพอีอาร์แสดงระบบฐานข้อมูลของบริษัทดีบี (ซ้ำ)

ถัดไปให้ดูรีเลชันชิปไทป์ WORKS_FOR ที่มีเลข 1 กำกับบนเส้นรีเลชันชิปที่ชี้ไปยัง DEPARTMENT และ N ชี้ไปยัง EMPLOYEE ในกรณีนี้เป็นความสัมพันธ์แบบ 1:N ซึ่งหมายความว่า EMPLOYEE 1 คน อาจทำงานได้อย่างมาก 1 DEPARTMENT ทว่าแต่ละ DEPARTMENT อาจมี EMPLOYEE ได้หลายคน คือกี่คนก็ได้

จากนั้นให้ดูรีเลชันชิปไทป์ WORKS_ON จะเห็นอักษร M และ N กำกับบนเส้นเชื่อมระหว่าง WORKS_ON ที่ชี้ไปยัง EMPLOYEE และ PROJECT นั่นหมายความว่า EMPLOYEE คนหนึ่งอาจทำงานได้หลาย PROJECT และ PROJECT หนึ่ง ๆ จะมี EMPLOYEE ได้หลายคน สังเกตว่าการกำกับแบบ M:N ให้ใส่อักขระ M ไว้ข้างหนึ่ง และ N ไว้อีกข้างหนึ่ง อาจเป็นข้างใดก็ได้

3.3.5 เงื่อนไขบังคับเข้าร่วม

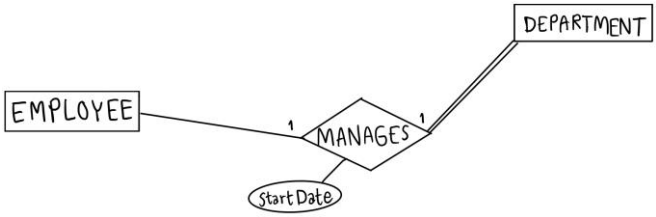
เงื่อนไขบังคับเข้าร่วม (participation constraint) เป็นการกำหนดค่าต่ำสุดของความสัมพันธ์นั้น หรืออาจมองว่าคือจำนวนรีเลชันชิปน้อยสุดที่ต้องมีก็ได้ ค่าที่ว่านี้มีความเป็นไปได้แค่ 2 แบบคือ

- การเข้าร่วมแบบไม่บังคับ
- การเข้าร่วมแบบบังคับ

การเข้าร่วมแบบไม่บังคับ (partial participation) ภาษาอังกฤษใช้คำว่า partial ที่แปลตรง ๆ ได้ว่าบางส่วน แต่ในตำรานี้ขอใช้คำว่าไม่บังคับ เพราะโดยความหมายนั้น ต้องการสื่อว่ารีเลชันชิปนั้นอาจเกิดหรือไม่เกิดก็ได้ ซึ่งตรงข้ามกับ การเข้าร่วมแบบบังคับ (total participation) ที่หมายความว่าต้องมีรีเลชันชิปนั้นเกิดขึ้นเสมอ อาจมองอีกนัยหนึ่งว่าเป็นค่าต่ำสุดก็ได้ ซึ่งก็คืออย่างน้อยต้องมี 1 รีเลชันชิปเกิดขึ้นนั่นเอง

การเข้าร่วมแบบไม่บังคับ ให้แสดงด้วยเส้นเชื่อมแบบเส้นเดี่ยว ส่วนการเข้าร่วมแบบบังคับ ให้แสดงด้วยเส้นคู่

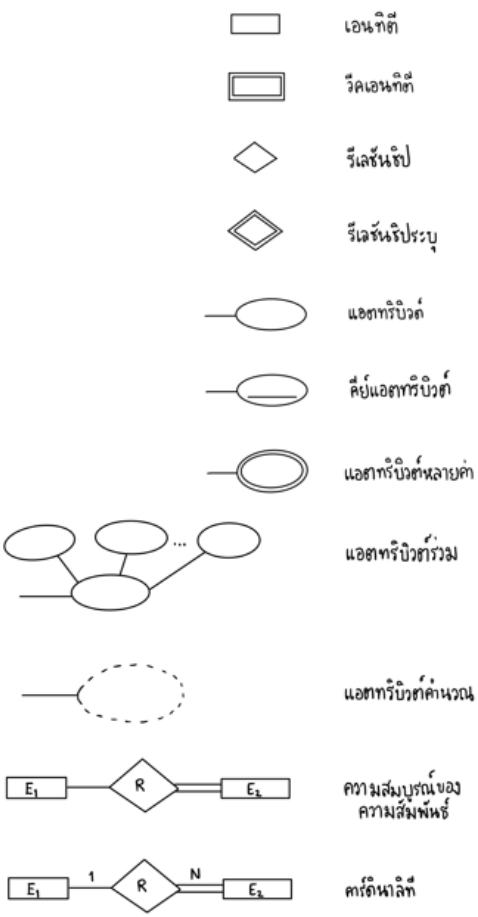
เพื่อให้เห็นภาพชัดเจน ขอยกตัวอย่างรีเลชันชิปไทป์ MANAGES แสดงในรูปที่ 27 ซึ่งเชื่อมระหว่าง EMPLOYEE กับ DEPARTMENT ซึ่งจะเห็นได้ว่ามีเส้นเดี่ยวเชื่อมระหว่างเอนทิตีไทป์ EMPLOYEE กับ รีเลชันชิปไทป์ MANAGES นั่นหมายความว่าอาจไม่มี EMPLOYEE ที่มีบทบาทผู้จัดการเลยก็ได้ แต่ในขณะเดียวกัน เราจะเห็นเส้นคู่กำกับระหว่าง MANAGES กับ DEPARTMENT นั่นหมายความว่า DEPARTMENT ต้องมีผู้จัดการอย่างน้อย 1 คน



รูปที่ 27 เงื่อนไขบังคับเข้าร่วมแบบบังคับและแบบไม่บังคับ

3.3.6 สรุปสัญกรณ์อีอาร์

สัญกรณ์ (notation) ที่ใช้ในตำรานี้ มีดังแสดงในรูปที่ 28

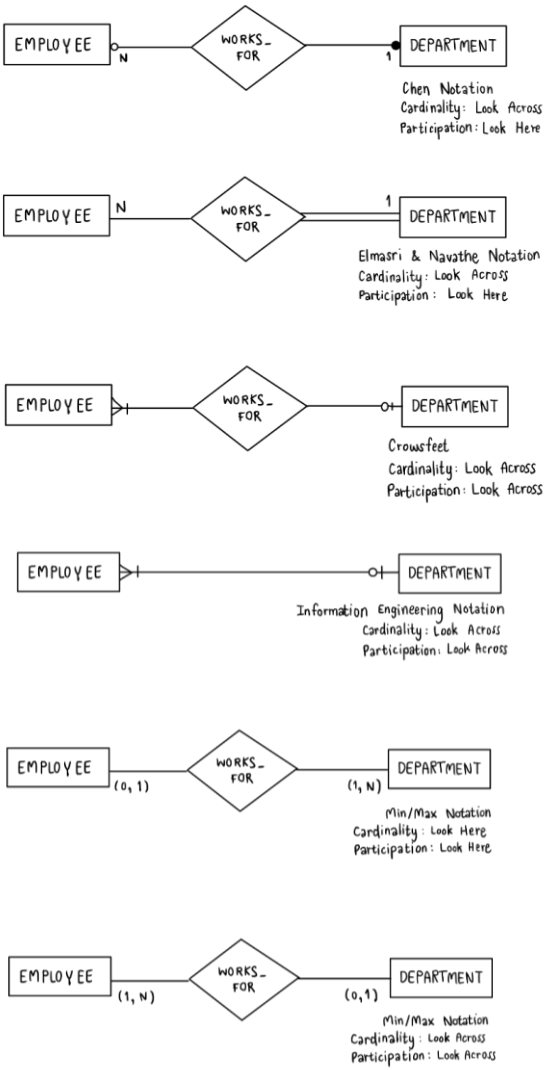


รูปที่ 28 สรุปสัญกรณ์อีอาร์

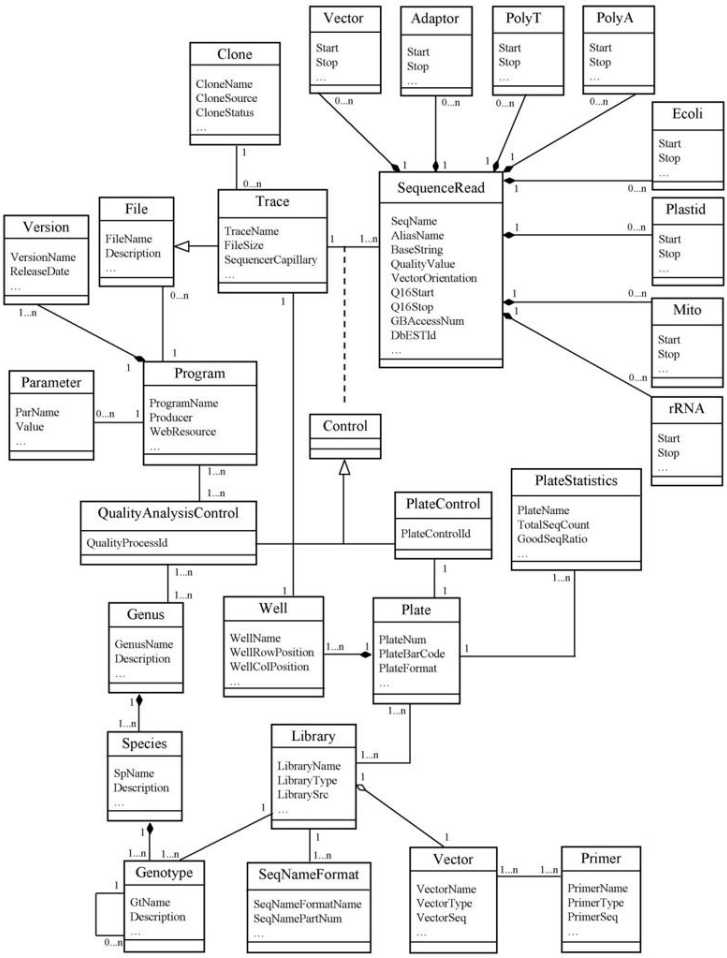
3.3.7 สัญกรณ์เพิ่มเติม

ตำราว่าด้วยเรื่องแผนภาพอีอาร์นั้น มีการนำเสนอการใช้สัญกรณ์ (notation) ต่างกันอยู่บ้าง ที่ใช้ในตำรานี้จะเป็นวิธีของศาสตราจารย์เอลมาศรีและศาสตราจารย์นาวาเธ [1] อย่างไรก็ดี ยังมีสัญกรณ์อีกหลายชุด ต่างก็สร้างมาเพื่อใช้แสดงแผนภาพระดับแนวคิดในการออกแบบฐานข้อมูลทั้งสิ้น แต่ก็อาจมีรายละเอียดต่างกันบ้าง

รูปที่ 29 จาก [26] เปรียบเทียบสัญกรณ์แบบต่าง ๆ และรูปที่ 30 แสดงตัวอย่างสัญกรณ์ยูเอ็มแอล (Unified Modeling Language : UML) [27] ซึ่งถ้านำไปเทียบกับแผนภาพอีอาร์ด้วยสัญกรณ์ชุดก่อนก็จะได้ความหมายคล้ายคลึงกัน



รูปที่ 29 เปรียบเทียบสัญกรณ์แบบต่าง ๆ



รูปที่ 30 สัญกรณ์ยูเอ็มแอล

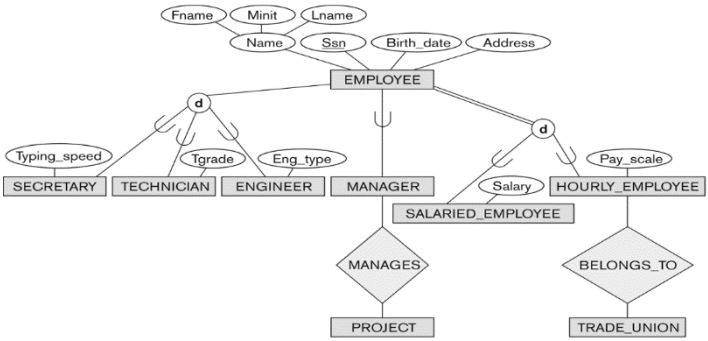
3.4 **แบบจำลองอีอีอาร์**

ที่ผ่านมาได้กล่าวถึงแบบจำลองอีอาร์เพื่อแสดงระบบฐานข้อมูลแบบที่นิยมใช้ตั้งแต่ยุคที่เริ่มใช้ฐานข้อมูลกันอย่างแพร่หลาย อย่างไรก็ดี ในยุคต่อ ๆ มาที่เทคโนโลยีมีการเปลี่ยนแปลงไป ก็มีการนำเสนอแบบจำลองใหม่ ๆ ที่รองรับแนวคิดที่ทันสมัยขึ้นเรื่อย ๆ ที่สำคัญคือยุคที่เปลี่ยนแปลงมานิยมการเขียนโปรแกรมด้วยภาษาเชิงวัตถุ (Object-Oriented language) ได้มีผู้ปรับปรุงแบบจำลองอีอาร์ให้รองรับแนวคิดเชิงวัตถุ เรียกว่า แบบจำลองอีอาร์แบบขยายหรือแบบปรับปรุง (Extended หรือ Enhanced ER model) [28, 29] หรือเรียกสั้น ๆ ว่า แบบจำลองอีอีอาร์ (EER model)

เพื่อรองรับแนวคิดเชิงวัตถุ แบบจำลองอีอีอาร์จึงต้องผนวกความสามารถเชิงวัตถุให้เข้ากับฐานข้อมูล แนวคิดดังกล่าว ได้แก่

1. ซับคลาส/ซูเปอร์คลาส (subclass/superclass)
2. สเปเชียลไลเซชัน/เจเนอรัลไลเซชัน (specialization/generalization)
3. การรับทอด (inheritance)

การแสดงแบบจำลองอีอีอาร์จะใช้แผนภาพเรียกว่าแผนภาพอีอีอาร์ มีลักษณะคล้ายแผนภาพอีอาร์ และมีสัญกรณ์เพิ่มเติมเพื่อรองรับแนวคิด 3 ข้อที่ได้กล่าวไป รูปที่ 31 แสดงตัวอย่างแผนภาพอีอีอาร์ อ้างอิงมาจาก [2]

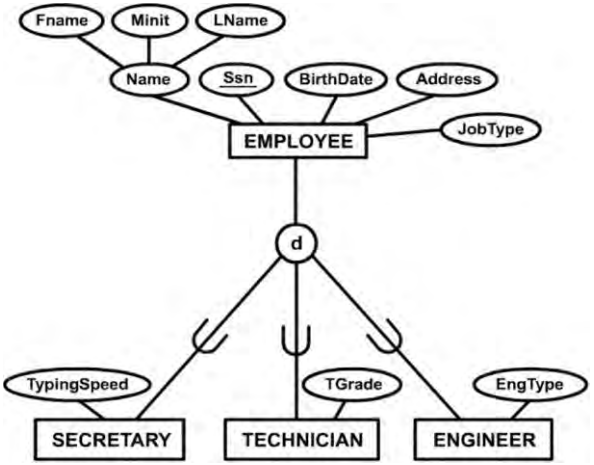


รูปที่ 31 แผนภาพอีอีอาร์

3.4.1 ซับคลาส/ซูเปอร์คลาส

ซับคลาส/ซูเปอร์คลาส (subclass/superclass) คือการประยุกต์แนวคิดเชิงวัตถุมาใช้กับฐานข้อมูล ในลักษณะคลาส โดยให้มองว่าเอนทิตีแต่ละเอนทิตีคือคลาส ดังนั้น เมื่อนำแนวคิดว่าคลาสสามารถแตกย่อยได้ ก็เรียก คลาสหลัก ว่า ซูเปอร์คลาส และเรียก คลาสย่อย ว่า ซับคลาส ตำรานี้จะเรียกว่าซูเปอร์คลาสและซับคลาส ซึ่งก็คือเอนทิตีไทป์นั่นเอง แต่เป็นเอนทิตีไทป์ที่แตกย่อยได้ ซูเปอร์คลาสก็คือเอนทิตีหลัก ส่วนซับคลาสคือเอนทิตีย่อยมาจากเอนทิตีหลัก ในการอ่านแผนภาพอีอีอาร์นี้ หากพบเอนทิตีไทป์ที่มีการแตกย่อย ให้เข้าใจว่าคือเอนทิตีไทป์ที่มีคุณสมบัติเป็นซูเปอร์คลาสหรือซับคลาสตามหน้าที่ที่กำหนด สำหรับเอนทิตีที่ไม่ได้มีการแตกย่อย ก็ให้อ่านว่าเอนทิตีไทป์เหมือนเดิม การแตกย่อยนี้ทำให้เกิดความสัมพันธ์แบบซูเปอร์คลาสซับคลาส

บางตำรา [13] ก็เรียกความสัมพันธ์ลักษณะซับคลาสซูเปอร์คลาสนี้ว่า ความสัมพันธ์อีสเอหรืออีสแอน (IS-A or IS-AN relationship) เช่น จากรูปที่ 32 อ่านได้ว่า SECRETARY IS-AN EMPLOYEE, TECHNICIAN IS-AN EMPLOYEE เป็นต้น



รูปที่ 32 ซูเปอร์คลาสซับคลาส

รูปที่ 32 แสดงซูเปอร์คลาส EMPLOYEE ที่แตกซับคลาสได้ 3 ซับคลาส คือ SECRETARY, TECHNICIAN และ ENGINEER นอกจากนั้น แต่ละซับคลาสยังมีแอตทริบิวต์เพิ่ม เรียกว่า แอตทริบิวต์เฉพาะ (specific attribute) คือ ซับคลาส SECRETARY มีแอตทริบิวต์เฉพาะคือ TypingSpeed, TECHNICIAN มีแอตทริบิวต์เฉพาะคือ TGrade และ ENGINEER มีแอตทริบิวต์เฉพาะคือ EngType

สังเกตความแตกต่างระหว่างแบบจำลองอีอาร์และแบบจำลองอีอีอาร์ จะเห็นได้ว่าแบบจำลองอีอาร์เดิมนั้นไม่สามารถรองรับเอนทิตีที่คล้ายกันได้ หากใช้แบบเดิมก็จำเป็นต้องนำแอตทริบิวต์ TypingSpeed, TGrade และ EngType ไปเพิ่มอยู่ในเอนทิตีไทป์ EMPLOYEE

ทั้งนี้เอนทิตีใด ๆ ไม่สามารถอยู่ในฐานข้อมูลโดยเป็นเพียงสมาชิกของซับคลาสได้ จะต้องเป็นสมาชิกของซูเปอร์คลาสด้วย ในขณะที่สมาชิกของซูเปอร์คลาสอาจเป็นสมาชิกของคลาสย่อยกี่ตัวก็ได้ เช่น วิศวกรที่เป็นลูกจ้างประเภทเงินเดือน อาจอยู่ในสองคลาสย่อยคือ ENGINEER และ SALARIED_EMPLOYEE แต่ทั้งนี้ไม่จำเป็นว่าเอนทิตีทุกตัวที่อยู่ในซูเปอร์คลาสจะต้องเป็นสมาชิกของคลาสย่อยเสมอไป

3.4.2 การรับทอด

การรับทอด (inheritance) ในแนวคิดเชิงวัตถุ คือ การคงความสามารถบางอย่างของวัตถุที่สัมพันธ์กัน เมื่อนำมาประยุกต์กับฐานข้อมูล ก็คือการกำหนดให้เอนทิตีที่เกี่ยวข้องกันจะต้องรับทอดคุณสมบัติซึ่งกันและกันด้วย การรับทอดนี้จะเกิดขึ้นเมื่อมีความสัมพันธ์แบบซับคลาสซูเปอร์คลาสทำให้ซับคลาสรับทอดคุณสมบัติของซูเปอร์คลาสไว้นั่นเอง

ความสำคัญคือ ซับคลาสทั้งหมดจะต้องคงคุณสมบัติจากซูเปอร์คลาสด้วยเสมอ แต่อาจมีแอตทริบิวต์พิเศษเพิ่มมาในแต่ละซับคลาสได้

ด้านซ้ายของรูปที่ 31 แสดงซูเปอร์คลาส EMPLOYEE และซับคลาส SECRETARY, TECHNICIAN และ ENGINEER ทั้ง 3 ซับคลาสนี้จะคง

แอตทริบิวต์ทั้งหมดของซูเปอร์คลาส EMPLOYEE ได้แก่ Name, SSN, Birthdate และ Address นอกจากนั้นแต่ละซับคลาสยังมีแอตทริบิวต์เพิ่ม คือซับคลาส SECRETARY เพิ่มแอตทริบิวต์ TypingSpeed, TECHNICIAN เพิ่มแอตทริบิวต์ TGrade และ ENGINEER เพิ่มแอตทริบิวต์ EngType

นั่นหมายความว่าจากการรับทอดจะทำให้ซับคลาส SECRETARY มีแอตทริบิวต์คือ Name, SSN, Birthdate, Address, TypingSpeed ส่วนซับคลาส TECHNICIAN แอตทริบิวต์คือ Name, SSN, Birthdate, Address, TGrade และ ENGINEER แอตทริบิวต์คือ Name, SSN, Birthdate, Address, EngType

3.4.3 สเปเชียลไลเซชันและเจเนอรัลไลเซชัน

สเปเชียลไลเซชัน (specialization) และ เจเนอรัลไลเซชัน (generalization) เป็นกระบวนการในการกำหนดสมาชิกของซับคลาสและซูเปอร์คลาสทั้งคู่ เพียงแต่ดำเนินการไม่เหมือนกัน กล่าวคือ กระบวนการสเปเชียลไลเซชันคือการแตกย่อยจากบนลงล่าง (top-down) คือกำหนดซูเปอร์คลาสก่อน แล้วแตกย่อยให้ได้ซับคลาส ส่วนเจเนอรัลไลเซชันนั้นเป็นการรวมซับคลาสหลายซับคลาสให้ได้ซูเปอร์คลาส หรือมองว่าเป็นการจัดกลุ่มจากล่างขึ้นบน (bottom-up) นั่นเอง

ยกตัวอย่างจากรูปที่ 31 ที่มีซูเปอร์คลาส EMPLOYEE ซึ่งมีความสัมพันธ์กับซับคลาส SALARIED_EMPLOYEE กับ HOURLY_EMPLOYEE หากกระบวนการกำหนดซูเปอร์คลาสซับคลาสนี้ เริ่มคิดจาก EMPLOYEE แล้วแยกเป็น SALARIED_EMPLOYEE กับ HOURLY_EMPLOYEE ก็ให้เรียกว่า

เป็นกระบวนการสเปเชียลไลเซชัน ในทางกลับกัน หากกระบวนการนี้เริ่มจากกำหนด SALARIED_EMPLOYEE กับ HOURLY_EMPLOYEE ก่อน แล้วนำมารวมเป็น EMPLOYEE ก็ให้เรียกว่าเป็นกระบวนการเจเนอรัลไลเซชัน

การเขียนแผนภาพอีอีอาร์ ให้เขียนสัญลักษณ์แสดงความสัมพันธ์แบบซูเปอร์คลาสซับคลาสด้วยตะขอ ลักษณะคล้ายตัว U กำกับบนเส้นความสัมพันธ์นั้น

เมื่อมีมากกว่าหนึ่งซับคลาส ให้เขียนสัญลักษณ์วงกลมเชื่อมซับคลาสทั้งหมดเข้าด้วยกัน แล้วเขียนตะขอบนเส้นความสัมพันธ์จากซับคลาสมายังวงกลมนั้น ดังตัวอย่างก่อนหน้า

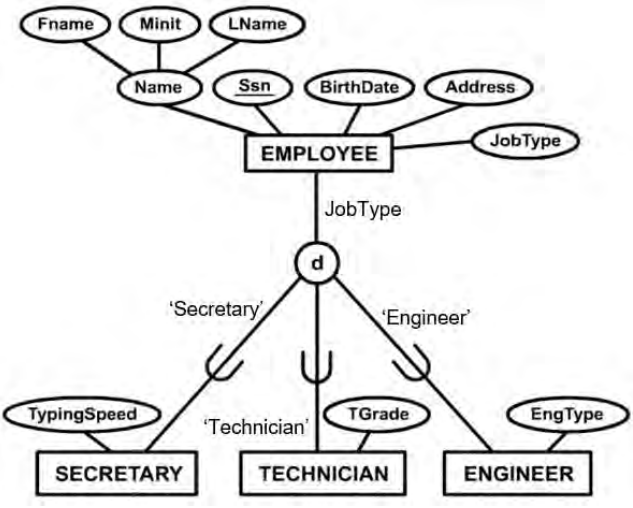
ขอให้สังเกตด้วยว่า ซับคลาสก็สามารถมีรีเลชันชิปกับเอนทิตีอื่นได้ จากรูปจะพบตัวอย่าง ซับคลาส HOURLY_EMPLOYEE มีรีเลชันชิปไทป์ BELONGS_TO เชื่อมกับเอนทิตีไทป์ TRADE_UNION และ ซับคลาส MANAGER รีเลชันชิปไทป์ MANAGES เชื่อมกับเอนทิตีไทป์ PROJECT ด้วย

3.4.4 การกำหนดสมาชิกซับคลาส

การกำหนดหรือระบุสมาชิกซับคลาสสามารถทำได้ 2 วิธี คือ 1) ใช้เงื่อนไขกำหนด หรือ 2) ผู้ใช้กำหนด

ซับคลาสที่ใช้เงื่อนไขกำหนด (predicate-defined subclass หรือ condition-defined subclass) คือซับคลาสที่มีค่าของแอตทริบิวต์ในซูเปอร์คลาสเป็นตัวระบุว่าเป็นซับคลาสใด เราเรียกแอตทริบิวต์นี้ว่า แอตทริบิวต์แบบเพรดิเคตกำหนด (defining predicate) วิธีแสดงซับคลาสที่ใช้เงื่อนไขกำหนดนี้ทำได้โดยการเขียนชื่อแอตทริบิวต์ที่เป็นตัวกำหนดลงบนเส้น

ระหว่างซูเปอร์คลาสกับวงกลม และเขียนค่าที่ใช้ในการกำหนดซับคลาสบนเส้นระหว่างวงกลมกับซับคลาส รูปที่ 33 แสดงตัวอย่างกรณีดังกล่าว จะเห็นว่า แอตทริบิวต์ Job_Type เป็นแอตทริบิวต์แบบเพรดิเคตกำหนด เมื่อค่าของ Job_Type เป็น 'Secretary' ให้กำหนดซับคลาสเป็น SECRETARY เมื่อค่าของ Job_Type เป็น 'Technician' ให้กำหนดซับคลาสเป็น TECHNICIAN และเมื่อค่าของ Job_Type เป็น 'Engineer' ให้กำหนดซับคลาสเป็น ENGINEER



รูปที่ 33 ซับคลาสที่ใช้เงื่อนไขกำหนด

จะเห็นว่ากระบวนดังกล่าวคือกระบวนการสเปเชียลไลเซชัน และเนื่องจากเป็นการสเปเชียลไลเซชันด้วยแอตทริบิวต์ จึงเรียกกระบวนการนี้ได้ว่ากระบวนการสเปเชียลไลเซชันแบบแอตทริบิวต์กำหนด (attribute defined-specialization)

ในกรณีที่ไม่มีแอตทริบิวต์แบบเพรดิเคตเป็นตัวกำหนดซับคลาส เราจะต้องให้ผู้ใช้เป็นผู้กำหนดว่าจะเป็นซับคลาสใด เราเรียกซับคลาสกรณีนี้ว่าเป็นซับคลาสที่ผู้ใช้กำหนด (user-defined subclass)

3.4.5 ความมีส่วนร่วมของซับคลาส

ความมีส่วนร่วมของซับคลาส (disjointness constraint) คือ การระบุประเภทความสัมพันธ์ระหว่างซูเปอร์คลาสซับคลาส มีเพียง 2 แบบคือแบบดิสจอยท์ (disjoint) และแบบ โอเวอร์แลป (overlap)

แบบดิสจอยท์นั้นหมายถึงซูเปอร์คลาสหนึ่ง ๆ จะสามารถแยกย่อยได้เป็นซับคลาสใดซับคลาสหนึ่งเท่านั้น ตัวอย่างจากรูปจะเห็นว่าซูเปอร์คลาส EMPLOYEE จะเป็นได้เพียง SECRETARY หรือ ENGINEER หรือ TECHNICIAN เท่านั้น ไม่สามารถเป็นสมาชิกได้มากกว่าหนึ่งซับคลาส

ส่วนแบบโอเวอร์แลปนั้นหมายถึงซูเปอร์คลาสหนึ่ง ๆ จะเป็นสมาชิกได้มากกว่าหนึ่งซับคลาส เช่น EMPLOYEE คนหนึ่งอาจเป็นทั้งพนักงานประเภทรับเงินเดือน คือ SALARIED_EMPLOYEE และ เป็นพนักงานประเภทรับค่าจ้างรายชั่วโมง คือ HOURLY_EMPLOYEE ก็ได้

ในแผนภาพอีอีอาร์ ให้เขียน d ในวงกลมเพื่อแสดงความมีส่วนร่วมแบบดิสจอยท์ หรือเขียน o ในวงกลม เพื่อแสดงความมีส่วนร่วมแบบโอเวอร์แลป ดังแสดงในรูปที่ 31

3.4.6 ความสมบูรณ์ของความสัมพันธ์

ความสมบูรณ์ของความสัมพันธ์ (completeness constraint) คือการระบุว่าเอนทิตีที่เป็นซูเปอร์คลาสนั้นมีความสัมพันธ์กับซับคลาสแบบสมบูรณ์หรือไม่ หรืออธิบายง่าย ๆ คือ ซูเปอร์คลาสนั้นจะต้องแยกเป็นซับคลาสหรือไม่ ค่าที่เป็นไปได้มีเพียง 2 แบบ คือแบบบังคับ (total) และแบบไม่บังคับ (partial) กรณีบังคับหมายความว่า เอนทิตีในซูเปอร์คลาสหนึ่ง ๆ จะต้องเป็นสมาชิกของซับคลาส ถ้าแบบไม่บังคับ หมายความว่าเอนทิตีในซูเปอร์คลาสนั้น ไม่จำเป็นต้องแตกย่อยเป็นซับคลาส

ในแผนภาพอีอีอาร์ ให้เขียนเส้นคู่จากซูเปอร์คลาสมายังวงกลม เพื่อแสดงความสัมพันธ์แบบบังคับ หรือเขียนเส้นเดี่ยว เพื่อแสดงความสัมพันธ์แบบไม่บังคับ

จะเห็นว่ากระบวนดังกล่าวคือกระบวนการสเปเชียลไลเซชันเพราะต้องเริ่มพิจารณาจากซูเปอร์คลาสก่อน และในกรณีที่กำหนดความสมบูรณ์ของความสัมพันธ์เป็นแบบบังคับนั้น ให้เรียกกระบวนการนี้ได้ว่า กระบวนการสเปเชียลไลเซชันแบบบังคับ (total specialization) หากเป็นแบบไม่บังคับให้เรียกว่า กระบวนการสเปเชียลไลเซชันแบบไม่บังคับ (partial specialization)

ดังนั้น เมื่อคำนึงถึงข้อกำหนดทั้งหมดที่ผ่านมา เราจะมีเจเนอรัลไลเซชันหรือสเปเชียลไลเซชัน 4 ประเภท คือ

1. ดิสจอยท์-บังคับ (disjoint-total)
2. ดิสจอยท์-ไม่บังคับ (disjoint-partial)
3. โอเวอร์แลป-บังคับ (overlapping-total)
4. โอเวอร์แลป-ไม่บังคับ (overlapping-partial)

3.4.7 กฎการแทรกและการลบข้อมูลในอีอีอาร์

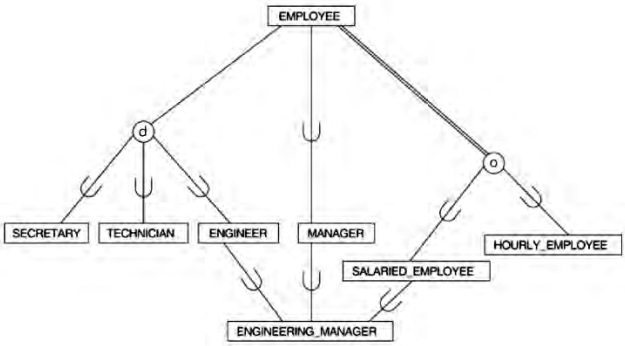
การปรับแก้ข้อมูลในฐานข้อมูลนั้น เมื่อใช้แบบจำลองอีอีอาร์ จำเป็นต้องคำนึงถึงคุณสมบัติการรับทอดเสมอ เราสามารถสรุปผลกระทบในการรับทอดเมื่อใดก็ตามที่มีการปรับปรุงข้อมูลดังนี้

1. การลบเอนทิตีออกจากซูเปอร์คลาส จะส่งผลให้ต้องลบเอนทิตีในซับคลาสทั้งหมดเสมอ
2. การแทรกเอนทิตีที่ซับคลาส จะส่งผลให้ต้องแทรกเอนทิตีที่ซูเปอร์คลาสทั้งหมดเสมอ
3. การแทรกเอนทิตีที่ซูเปอร์คลาส หากใช้กระบวนการสเปเชียลไลเซชันแบบแอตทริบิวต์กำหนด จะส่งผลให้เกิดการแทรกเอนทิตีที่ซับคลาสบางซับคลาส
4. การแทรกเอนทิตีที่ซูเปอร์คลาส หากใช้กระบวนการสเปเชียลไลเซชันแบบบังคับ จะส่งผลให้เกิดการแทรกเอนทิตีที่ซับคลาสอย่างน้อยหนึ่งซับคลาส (หากความมีส่วนร่วมของซับคลาสเป็นแบบดิสจอยท์จะมีการแทรกเอนทิตีเพียงหนึ่งซับคลาสเท่านั้น)

3.4.8 จำนวนชั้นของสเปเชียลไลเซชัน

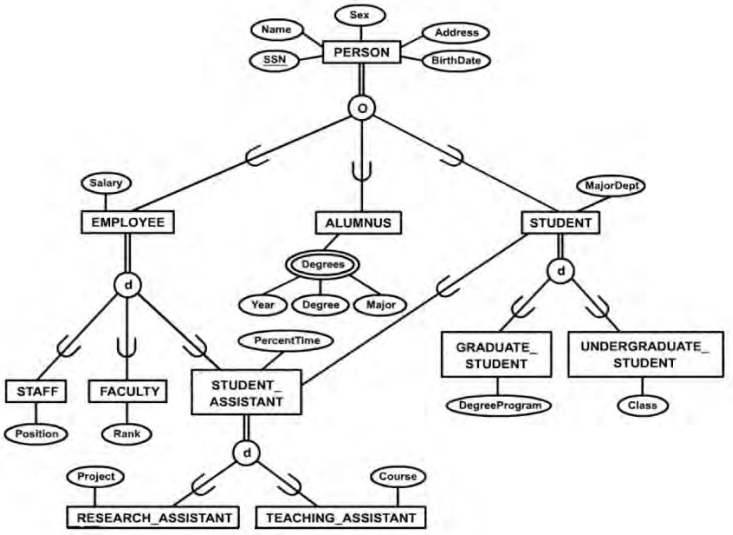
กระบวนการสเปเชียลไลเซชันหรือเจเนอรัลไลเซชันมักไม่เกิดเพียงชั้นเดียว หมายความว่าเมื่อซูเปอร์คลาสได้ถูกแตกย่อยเป็นซับคลาสแล้วยังอาจถูกแตกย่อยไปเรื่อย ๆ ได้ ถ้ามองกลับกันคือเจเนอรัลไลเซชันหลายทอดนั่นเอง

เมื่อมีการแตกย่อยแล้ว ให้พิจารณาที่ซับคลาส หากซับคลาสนั้นมีซูเปอร์คลาสเพียงซูเปอร์คลาสเดียว ก็จะรับทอดคุณสมบัติจากซูเปอร์คลาสนั้นและซูเปอร์คลาสก่อนหน้านั้นขึ้นไปทุกชั้นจนหมดซูเปอร์คลาสถึงต้นตอหรือราก (root) คุณสมบัติที่รับทอดในฐานข้อมูลก็คือแอตทริบิวต์ทั้งหมดจากทุกเอนทิตีไทป์และรีเลชันชิปทั้งหมด กรณีนี้เราเรียกว่า การรับทอดแบบเดี่ยว (single inheritance) และเรียกโครงสร้างความสัมพันธ์นี้ว่า สเปเชียลไลเซชันแบบลำดับชั้น (specialization hierarchies) ซึ่งมีลักษณะโครงสร้างแบบต้นไม้ ตัวอย่างในรูปที่ 34 จะพบว่า ENGINEER เป็นซับคลาสของ EMPLOYEE และยังมีซับคลาสต่อลงไปอีกชั้นคือ ENGINEERING_MANAGER



รูปที่ 34 สเปเชียลไลเซชันแบบแลตทิซ

อีกกรณีคือเมื่อพิจารณซับคลาสแล้วพบว่ามีซูเปอร์คลาสมากกว่าหนึ่ง ซับคลาสนั้นจะรับทอดคุณสมบัติจากซูเปอร์คลาสทุกซูเปอร์คลาสที่เกี่ยวข้อง คือแอตทริบิวต์ทั้งหมดจากทุกเอนทิตีไทป์และรีเลชันชิปทั้งหมด เราเรียกกรณีนี้ว่า การรับทอดแบบหลาย (multiple inheritance) และเรียกซับคลาสลักษณะนี้ว่า ซับคลาสร่วม (Shared subclass) และให้เรียกโครงสร้างความสัมพันธ์นี้ว่า สเปเชียลไลเซชันแบบแลตทิซ (Specialization Lattice) ซึ่งมีโครงสร้างแบบกราฟระบุทิศทาง (Directed acyclic graph) แสดงตัวอย่างซับคลาสร่วมคือ STUDENT_ASSISTANT ที่มีซูเปอร์คลาสหลัก 2 ซูเปอร์คลาส ได้แก่ EMPLOYEE และ STUDENT ดังรูปที่ 35 แสดงอีอีอาร์แบบซับซ้อน มีทั้งสเปเชียลไลเซชันแบบลำดับชั้นและแบบแลตทิซ



รูปที่ 35 สเปเชียลไลเซชันแบบลำดับชั้นและแบบแลตทิซ

3.4.9 แคทะกอรี

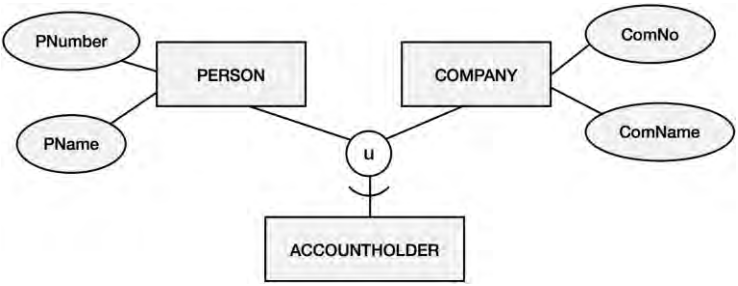
แคทะกอรี (Category) เป็นประเภทหนึ่งของซับคลาส เกิดเมื่อซับคลาสมีซูเปอร์คลาสหลายซูเปอร์คลาส แต่ซูเปอร์คลาสเหล่านั้นเป็นเอนทิตีที่ต่างกัน ดังนั้นการรับทอดจะไม่ใช่การรับทอดแบบหลาย แต่เป็นการรับทอดแบบเลือก (selective inheritance) ในฐานข้อมูลก็คือรับทอดแอตทริบิวต์และรีเลชันชิปไทป์ทั้งหมดจากเฉพาะซูเปอร์คลาสที่มีความสัมพันธ์เชื่อมโยงกับซับคลาสนั้น ๆ เท่านั้น

ย้ำความแตกต่างระหว่างแคทะกอรีกับซับคลาสร่วม ที่สำคัญคือกรณีซับคลาสร่วมนั้น เอนทิตีที่อยู่ที่ซับคลาสคือเอนทิตีที่ได้มาจากการอินเตอร์เซกชัน (intersection) กันระหว่างซูเปอร์คลาสของซับคลาสร่วมนั้น

ส่วนแคทะกอรีนั้น ถ้านำเอนทิตีจากซูเปอร์คลาสมาอินเตอร์เซกชันกันจะได้เซตว่าง เพราะซูเปอร์คลาสเหล่านั้นไม่มีความเกี่ยวข้องกัน ไม่มีเอนทิตีร่วมกัน การรับทอดจึงต้องรับทอดจากซูเปอร์คลาสที่ซับคลาสนั้นเป็นสมาชิก

ดังนั้นการรับทอดคุณสมบัติของซับคลาสประเภทแคทะกอรีจึงจำเป็นต้องใช้วิธียูเนียน (union) เพื่อรวมแอตทริบิวต์และรีเลชันชิปไทป์ทั้งหมดของซูเปอร์คลาส เมื่อเขียนแผนภาพอีอีอาร์ ให้แสดงการยูเนียนด้วยตัว U ในวงกลม

จากรูปที่ 36 พบว่าซับคลาส ACCOUNTHOLDER เป็นประเภทแคทะกอรี มีซูเปอร์คลาส PERSON และ COMPANY ที่เป็นเอนทิตีไทป์ที่ไม่เกี่ยวกัน ไม่มีแอตทริบิวต์ร่วมกัน ดังนั้นการรับทอดจึงต้องยูเนียนคุณสมบัติของซูเปอร์คลาสทั้งสองเข้ามาด้วยกัน



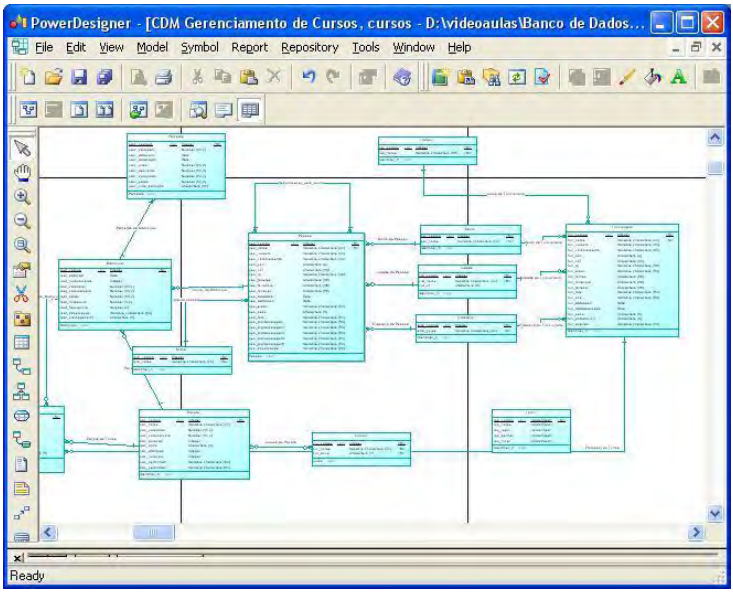
รูปที่ 36 ซับคลาสประเภทแคทะกอรีและการยูเนียน

อย่างไรก็ดี ความสัมพันธ์ระหว่างแคทะกอรีกับซูเปอร์คลาสนี้เป็นได้ทั้งแบบบังคับหรือไม่บังคับ หากเป็นแบบไม่บังคับ ก็จะจำลองได้ด้วยการยูเนียนเท่านั้น

หากเป็นแบบบังคับ นั่นหมายความว่าซับคลาสจะต้องรับทอดคุณสมบัติทั้งหมดของทุกซูเปอร์คลาส ซึ่งอาจจำลองด้วยกระบวนการสเปเชียลไลเซชันธรรมดาก็ได้ จากตัวอย่าง ก็จะเปลี่ยนเป็นให้ ACCOUNTHOLDER เป็นซูเปอร์คลาสที่มีซับคลาสคือ PERSON และ COMPANY

3.5 เครื่องมือสร้างแบบจำลองข้อมูล

มีเครื่องมือสร้างแบบจำลองข้อมูล (modeling tool) ที่ได้รับความนิยมเป็นจำนวนมากที่รองรับแบบจำลองระดับแนวคิด (conceptual modeling) และสามารถแปลงเอนทิตี รีเลชันชิป แอตทริบิวต์ ตลอดจนเงื่อนไขบังคับและรายละเอียดทั้งหมด เป็นตารางที่ระบบการจัดการฐานข้อมูลเข้าใจและนำไปใช้ได้ทันที ข้อดีคือ สะดวก ประหยัดเวลา และผิดพลาดน้อย อย่างไรก็ตาม ผู้ใช้จะต้องเลือกใช้เครื่องมือให้ถูกประเภทและถูกต้องตามรูปแบบหรือโมเดลของฐานข้อมูลด้วย รูปที่ 37 แสดงตัวอย่างหน้าจอของเครื่องมือชื่อ PowerDesigner [30]



รูปที่ 37 หน้าจอของ PowerDesigner

ตารางที่ 1 แสดงตัวอย่างเครื่องมือที่เป็นที่นิยมในปัจจุบัน [31] พร้อมแพลตฟอร์มและระบบปฏิบัติการ (operating system) ที่รองรับ มีทั้งที่ให้บริการฟรี จนถึงเครื่องมือราคาแพงที่มักมีความสามารถสูงตามไปด้วย

ตารางที่ 1 เครื่องมือแบบจำลองฐานข้อมูล

ชื่อเครื่องมือ	**บริษัท**	**แพลตฟอร์ม ที่รองรับ**	**OS ที่รองรับ**
Oracle SQL Developer Data Modeler	Oracle	Oracle, MS SQL Server, IBM DB2	Cross-platform
Navicat Data Modeler	PremiumSoft	MySQL, MS SQL Server, PostgreSQL, Oracle, SQLite	Windows, macOS, Linux
Database Deployment Manager	The Unauthorized Frog project	CUBRID, MySQL, SQLite	Windows, Linux
Software Ideas Modeler	Dusan Rodina	MS SQL Server, MySQL	Windows
Open ModelSphere	Grandite	MS SQL Server, MySQL, PostgreSQL, Oracle, DB2	Windows, macOS, Linux

3.6 **บทสรุปและเรื่องชวนคิด**

บทนี้เราได้เรียนเรื่องการจำลองข้อมูลโดยเฉพาะสำหรับฐานข้อมูลเชิงสัมพันธ์ แบบจำลองและแผนภาพที่เป็นเครื่องมือสำคัญ คือ แผนภาพอีอาร์และอีอีอาร์ อย่างไรก็ดี ด้วยเทคโนโลยีที่ก้าวหน้าอย่างรวดเร็ว ก็มีผู้เสนอแบบจำลองและแผนภาพเพิ่มมาอีกมากมายดังที่กล่าวเป็นตัวอย่างก่อนหน้านี้ ที่กำหนดใช้สัญลักษณ์ที่ต่างกันเพื่อสื่อความหมายคล้ายกัน ด้วยความรู้พื้นฐานที่ได้จากบทเรียนบทนี้ หวังว่าผู้เรียนจะสามารถประยุกต์ความรู้นี้เพื่ออ่านแผนภาพอื่นที่ใช้ออกแบบระบบฐานข้อมูลได้ และอยากให้คิดพัฒนาแบบที่เป็นมาตรฐานให้ดียิ่งขึ้นไป หรือคิดว่าในอนาคตเรายังจำเป็นต้องใช้แผนภาพเพื่อออกแบบฐานข้อมูลอีกหรือไม่ ถ้าเครื่องมือการออกแบบมีประสิทธิภาพสูงกว่านี้ จะเป็นไปได้ไหมว่าไม่จำเป็นต้องมีคนออกแบบระบบฐานข้อมูลอีกต่อไป

แบบฝึกหัดท้ายบท

1. จงอธิบายความแตกต่างระหว่างเอนทิตีและเอนทิตีไทป์
2. จงอธิบายความแตกต่างระหว่างเส้นแสดงความสัมพันธ์เชื่อมระหว่างเอนทิตีที่เป็นเส้นคู่กับเส้นเดี่ยว
3. จงอธิบายความสัมพันธ์แบบเวียนเกิด
4. จงอธิบายความแตกต่างระหว่างแบบจำลองอีอาร์กับแบบจำลองอีอีอาร์
5. แบบจำลองอีอีอาร์มีการเพิ่มความสามารถใดบ้างขึ้นมา เพื่อที่จะรองรับแนวคิดเชิงวัตถุให้เข้ากับฐานข้อมูล
6. จงอธิบายรีเลชันของเจเนอรัลไลเซชัน และ สเปเชียลไลเซชัน ในการทำงานแบบล่างขึ้นบน (bottom-up) และบนลงล่าง (top-down)
7. แบบจำลองข้อมูลหมายถึงอะไร จงยกตัวอย่างแบบจำลองที่มักใช้กันอย่างแพร่หลายมาอย่างน้อย 3 แบบ
8. จงบอกประโยชน์ของแบบจำลองข้อมูล
9. จงอธิบายความแตกต่างระหว่างแบบจำลองข้อมูลเชิงแนวคิดกับแบบจำลองข้อมูลเชิงกายภาพ
10. แบบจำลองอีอาร์ใช้อธิบายสิ่งใดของฐานข้อมูล และมีองค์ประกอบสำคัญอะไรบ้าง

บทที่ 4
ฐานข้อมูลเชิงสัมพันธ์

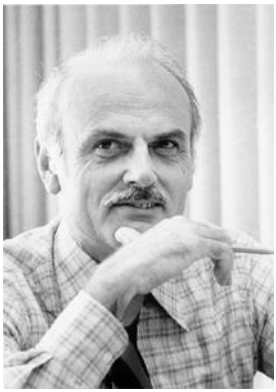
วัตถุประสงค์

1. เพื่อให้ทราบแนวคิดของรูปแบบฐานข้อมูลเชิงสัมพันธ์
2. เพื่อให้รู้จักองค์ประกอบของรีเลชัน
3. เพื่อให้เข้าใจสัญกรณ์คณิตศาสตร์ที่ใช้ในทฤษฎีรีเลชัน
4. เพื่อให้ทราบเงื่อนไขบังคับของฐานข้อมูลเชิงสัมพันธ์
5. สามารถสร้างข้อกำหนดการใช้งานในการแทรกข้อมูล การลบข้อมูลและการเปลี่ยนแปลงข้อมูล
6. เพื่อให้ทราบสิ่งที่ต้องพิจารณาเมื่อทำการปรับปรุงข้อมูลในรีเลชัน

แนวคิดฐานข้อมูลเชิงสัมพันธ์ (relational database) [32-40] นี้ถูกคิดค้นโดย ดร. เอดการ์ เอฟ คอดด์ (Dr. Edgar F. Codd) [41] ชาวอังกฤษ เมื่อ ค.ศ. 1970 ขณะที่ทำงานให้หน่วยวิจัยหน่วยหนึ่งของบริษัทไอบีเอ็ม นวัตกรรมครั้งนี้นับเป็นก้าวกระโดดในวงการคอมพิวเตอร์ที่สำคัญยิ่ง เป็นผลให้ ดร.คอดด์ได้รับรางวัลทัวริง (Turing award) [42] เมื่อ ค.ศ. 1981 และยังได้รับการพัฒนามาเรื่อย ๆ หลักการของฐานข้อมูลเชิงสัมพันธ์อาศัยหลักการจากคณิตศาสตร์เรื่องเซต (mathematical concept of set) เป็นพื้นฐาน และนำมาซึ่ง *ทฤษฎีรีเลชัน* (theory of relation)

หัวใจของฐานข้อมูลเชิงสัมพันธ์ คือ การมองข้อมูลในรูปแบบของตาราง หรือเรียกอย่างเป็นทางการในศาสตร์ด้านฐานข้อมูลว่า รีเลชัน ซึ่งสื่อถึงความสัมพันธ์ของข้อมูลทั้งหลาย

ตำราบทนี้จึงเป็นบทที่สำคัญมาก ผู้เรียนจะต้องเข้าใจรีเลชันและข้อบังคับกฎเกณฑ์ต่าง ๆ อย่างแม่นยำ เพื่อนำไปประยุกต์ใช้ได้ต่อไป



รูปที่ 38 ดร. เอดการ์ เอฟ คอดด์

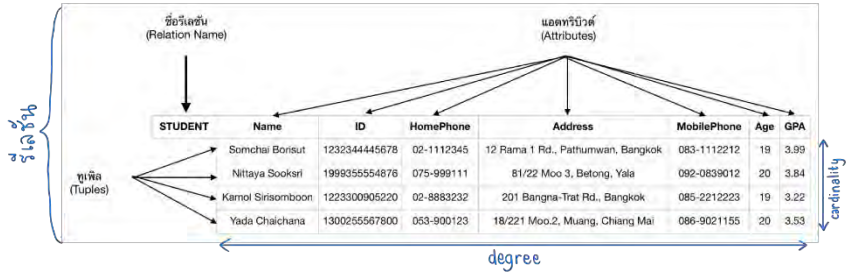
4.1 รีเลชัน

หัวใจของฐานข้อมูลเชิงสัมพันธ์คือการมองฐานข้อมูลในรูปแบบของ ตาราง (table) สองมิติของข้อมูลจำนวนมากที่มีความสัมพันธ์กัน เราเรียกตารางเหล่านี้ว่า รีเลชัน (relation)

รีเลชันมีลักษณะเป็นตารางที่ประกอบด้วย แถว (row) ของข้อมูลมาเรียงต่อกันเป็นรีเลชัน หรืออาจมองว่าประกอบด้วย คอลัมน์ (column) ของข้อมูลที่นำมาเรียงต่อกันก็จะได้รีเลชันเช่นกัน

บางตำราเรียกแถวข้อมูลว่า เรกคอร์ด (record) และเรียกคอมลัมน์ข้อมูลว่า ฟีลด์ (field)

ข้อมูลในแต่ละแถวต้องไม่ซ้ำกัน อาจกำหนดรหัสให้ข้อมูลในคอลัมน์ใดคอลัมน์หนึ่งแตกต่างกันทุกแถว รหัสที่ว่านี้มักถูกใช้ในการอ้างถึงข้อมูลในรีเลชัน โดยคอลัมน์ที่ถูกอ้างถึงนั้นจะเป็นคอลัมน์ที่สำคัญ เราเรียกชื่อคอลัมน์เหล่านี้ว่า แอตทริบิวต์ (attribute) ดังแสดงในรูปที่ 39



STUDENT	Name	ID	HomePhone	Address	MobilePhone	Age	GPA
	Somchai Borisut	1232344445678	02-1112345	12 Rama 1 Rd., Pathumwan, Bangkok	083-1112212	19	3.99
	Nittaya Sooksri	1999355554876	075-999111	81/22 Moo 3, Betong, Yala	092-0839012	20	3.84
	Kamol Sirisomboon	1223300905220	02-8883232	201 Bangna-Trat Rd., Bangkok	085-2212223	19	3.22
	Yada Chaichana	1300255567800	053-900123	18/221 Moo.2, Muang, Chiang Mai	086-9021155	20	3.53

รูปที่ 39 องค์ประกอบของรีเลชัน

นิยามทางคณิตศาสตร์ของรีเลชัน มีรูปแบบดังนี้

$R(A_1, A_2, ..., A_n)$

R คือ ชื่อรีเลชัน

A_i คือ ชื่อแอตตริบิวต์ตัวที่ i โดยที่ i มีค่าตั้งแต่ 1 ถึง n

แต่ละแอตทริบิวต์จะมีโดเมน (domain)กำกับ เขียนว่า $Dom(A_i)$

ตัวอย่าง

STUDENT(Name, ID, HomePhone, Address, MobilePhone, Age, GPA)

หมายถึง รีเลชันชื่อ STUDENT ประกอบด้วย 7 แอตทริบิวต์ ได้แก่ Name, ID, HomePhone, Address, MobilePhone, Age และ GPA โดยทุกแอตทริบิวต์มีโดเมนกำกับ เช่น กำหนดให้แอตทริบิวต์ ID มีโดเมนเป็นข้อความ (text) ความยาว 13 ตัวอักษร (digit) เป็นต้น ตัวอย่างข้อมูลแสดงในรูปที่ 40 รีเลชัน STUDENT

STUDENT	Name	ID	HomePhone	Address	MobilePhone	Age	GPA
	Somchai Borisut	1232344445678	02-1112345	12 Rama 1 Rd., Pathumwan, Bangkok	083-1112212	19	3.99
	Nittaya Sooksri	1999355554876	075-999111	81/22 Moo 3, Betong, Yala	092-0839012	20	3.84
	Kamol Sirisomboon	1223300905220	02-8883232	201 Bangna-Trat Rd., Bangkok	085-2212223	19	3.22
	Yada Chaichana	1300255567800	053-900123	18/221 Moo.2, Muang, Chiang Mai	086-9021155	20	3.53

รูปที่ 40 รีเลชัน STUDENT

นอกจากคำว่า “รีเลชัน” แล้ว จะเห็นได้ว่าศัพท์ที่เกี่ยวข้องมี ทูเพิล แอตทริบิวต์ และโดเมน จึงขออธิบายละเอียดดังนี้

4.1.1 ทูเพิล

เราเรียกแถวของข้อมูลว่า ทูเพิล (tuple) ทูเพิลในรีเลชันเดียวกันต้องมีจำนวนคอลัมน์เท่ากัน และแต่ละคอลัมน์ต้องมีลักษณะข้อมูลเดียวกัน เช่น

< “Somchai Borisud”, “1232344445678”, “021112345”, “12 Rama 1 Rd., Pathumwan, Bangkok”, “0831112212”, 19, 3.99 >

< “Nittaya Sooksri”, “1999355554876”, “075999111”, “81/22 Moo 3, Betong, Yala”, “0920839012”, 20, 3.84 >

< “Kamol Sirisomboon”, “1223300905220”, “028883232”, “201 Bangna-Trat Rd., Bangkok”, “0852212223”, 19, 3.22 >

< “Yada Chaichana”, “1300255567800”, “053900123”, “18/221 Moo 2, Muang, Chiang Mai”, “0869021155”, 20, 3.53 >

ทั้ง 4 ทูเพิลนี้มี 7 คอลัมน์ ที่ลักษณะข้อมูลที่เรียงต่อกันนั้นเหมือนกันทั้ง 4 ทูเพิล ซึ่งสอดคล้องกับรีเลชันตัวอย่าง STUDENT(Name, ID, HomePhone, Address, MobilePhone, Age, GPA) นั่นเอง

4.1.2 แอตทริบิวต์

แอตทริบิวต์ (attribute) ก็คือคอลัมน์ในตาราง จากตัวอย่างก่อนหน้า STUDENT(Name, ID, HomePhone, Address, MobilePhone, Age, GPA) นั้น มีอยู่ 7 แอตทริบิวต์ คือ Name, ID, HomePhone, Address, MobilePhone, Age, และ GPA

4.1.3 โดเมน

โดเมน (domain) คือ ลักษณะข้อมูล (data type) ที่กำหนดให้แต่ละแอตทริบิวต์ โดเมนดังกล่าวอาจเป็นได้ทั้ง integer, real, character, boolean, date, time, time-stamp หรืออื่นใดที่ใช้ในคอมพิวเตอร์ เช่น กำหนดให้ Name มีโดเมน 50 character ออกแบบมาเพื่อเก็บชื่อของ Student

การกำหนดโดเมนนั้น ให้กำหนดตามความหมายและวัตถุประสงค์ที่เราจะใช้แอตทริบิวต์นั้น ๆ เช่น เลขประจำตัวประชาชน ประกอบด้วยตัวเลข 13 หลัก ถ้าไม่พิจารณาให้ดีก็อาจกำหนดโดเมนเป็น เลขจำนวนเต็ม (integer) แต่ด้วยว่าเราไม่ได้จะนำเลขนี้ไปคำนวณและยอมให้หลักแรก ๆ เป็นศูนย์ได้ ดังนั้นวิธีที่ถูกต้องควรจะเลือกใช้โดเมนอักขระ (character) เป็นต้น

นอกเหนือจากนั้น เรายังสามารถกำหนดรูปแบบหรือหน้าตาการปรากฏของแอตทริบิวต์ได้ เช่น หมายเลขโทรศัพท์มือถือในประเทศไทย มีตัวเลข 10 ตัว ควรใช้ character ความยาว 10 และกำหนดให้แสดงรูปแบบเป็น xxx-xxx-xxxx ได้อีกด้วย แอตทริบิวต์ที่มักใช้รูปแบบกำกับอย่างนี้คือ date ที่

กำหนดให้แสดง xx-xx-xxxx ได้ ทั้งนี้ เมื่อกำหนดโดเมนและหน้าตาแล้ว ก็จะใช้โดเมนนี้ได้มากกว่าหนึ่งแอตทริบิวต์ เช่น ให้ใช้โดเมน date กับ BirthDate, EntryDate, SubscriptionDate เพื่อให้ข้อมูลมีความสอดคล้องกันทั้งฐานข้อมูล

4.2 สัญกรณ์คณิตศาสตร์

เราสามารถเขียนสัญกรณ์คณิตศาสตร์เพื่ออธิบายรีเลชันได้ดังนี้

$$R(A_1,A_2, ..., A_n)$$

R คือ สคีมา (schema) ของรีเลชัน r มีดีกรีเท่ากับ n ประกอบด้วยแอตทริบิวต์ A_1 ถึง A_n

r คือ สมาชิกของ R เมื่อเขียน r(R) หมายความว่า รีเลชัน r หนึ่งที่ใช้สคีมา R

ดังนั้น r ประกอบด้วยเซตของ m ทูเพิล เขียนได้ว่า

$$r = \{t_1, t_2, ..., t_m\}$$

แต่ละทูเพิล t ประกอบด้วย value v ที่เรียงกัน (ordered list)

$$t = \langle v_1, v_2, ..., v_n \rangle$$

แต่ละ value v_i ต้องมีโดเมนตามที่กำหนดให้แอตทริบิวต์ A_i เขียนว่า $dom(A_i)$ หรือค่าว่าง หรือ null

เราจะอ้างถึงค่าข้อมูลในทูเพิลด้วยการเขียน

$t[A_i] = V_i$ หรือ $t.A_i = V_i$

สรุปได้ว่า

- R คือ สคีมาของรีเลชัน เรียกได้อีกอย่างว่า อินเทนชัน (intension) ของรีเลชัน
- r(R) คือ สมาชิกของ R เรียกได้อีกอย่างว่า เอกซ์เทนชัน (extension) ของรีเลชัน

รีเลชันหนึ่ง ๆ เกิดจากผลการคาร์ทีเซียนของเซต แต่ละเซตคือค่าจากโดเมน ซึ่งโดเมนจะถูกกำหนดจากผู้ออกแบบรีเลชันนั้น

ตัวอย่าง

Let $S_1 = \{0,1\}$

Let $S_2 = \{a,b,c\}$

Let $R = S_1 \times S_2$

Then:

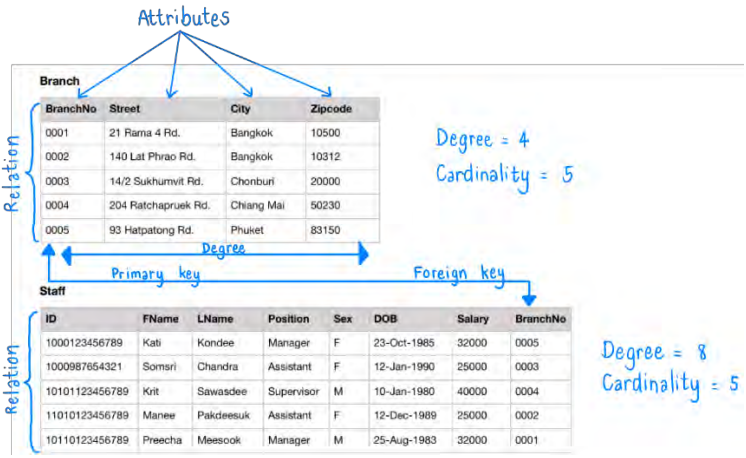
$r(R) = \{<0,a>, <0,b>, <1,c>\}$

เป็นเอกซ์เทนชันหรือสถานะ (state) หนึ่งที่เป็นไปได้ที่โจทย์ให้ ประกอบด้วย 3 ทูเพิล

ตารางที่ 2 และรูปที่ 41 สรุปศัพท์ที่ใช้ในฐานข้อมูลเชิงสัมพันธ์

ตารางที่ 2 ศัพท์ที่ใช้ในฐานข้อมูลเชิงสัมพันธ์

ศัพท์แบบจำลองเชิงสัมพันธ์	**ศัพท์อื่นที่มีความหมายเดียวกัน**
รีเลชัน (relation)	ตาราง (table)
ทูเพิล (tuple)	แถว (row) เรกคอร์ด (record)
แอตทริบิวต์ (attribute)	คอลัมน์ (column) หรือ ฟิลด์ (field)
โดเมน (domain)	ลักษณะข้อมูล (data type)
อินเทนชัน (intension) หรือ สคีมา (schema)	โครงสร้างตาราง (table definition)
เอกซ์เทนชัน (extension)	ตารางที่มีข้อมูล (populated table)
ดีกรี (degree)	จำนวนคอลัมน์หรือจำนวนฟิลด์ (number of column/field)
คาร์ดินาลิตี (cardinality)	จำนวนแถวหรือจำนวนเรกคอร์ด (number of rows/record)



รูปที่ 41 ศัพท์ที่ใช้ในฐานข้อมูลเชิงสัมพันธ์

4.3 เงื่อนไขบังคับของฐานข้อมูลเชิงสัมพันธ์

ฐานข้อมูลเชิงสัมพันธ์มีเงื่อนไขบังคับ (constraint) ที่ทุกรีเลชันต้องปฏิบัติดังต่อไปนี้

1. เงื่อนไขโดเมน (domain constraint)
2. เงื่อนไขคีย์ (key constraint)
3. เงื่อนไขบูรณภาพเอนทิตี (entity integrity constraint)
4. เงื่อนไขบูรณภาพอ้างอิง (referential integrity constraint)
5. เงื่อนไขอื่น ๆ

4.3.1 เงื่อนไขโดเมน

เงื่อนไขโดเมนระบุว่าแต่ละทูเพิลจะประกอบด้วยค่าข้อมูลของแต่ละแอตทริบิวต์ต้องเป็นค่าอะตอมมิก (atomic) คือไม่สามารถย่อยไปได้อีก และต้องอยู่ในรูปแบบที่โดเมนนั้น ๆ กำหนด (valid)

4.3.2 เงื่อนไขคีย์

เงื่อนไขคีย์ กล่าวว่าต้องไม่มีทูเพิลใดในรีเลชันที่มีค่าทั้งหมดซ้ำกัน ดังนั้นรีเลชันจึงควรจะมีคีย์ (key) ซึ่งคือแอตทริบิวต์ชุดหนึ่งซึ่งเป็นซับเซตของแอตทริบิวต์ทั้งหมดที่ทำให้ทูเพิลใด ๆ ไม่ซ้ำกัน เราเรียกแอตทริบิวต์ชุดนี้ว่าซูเปอร์คีย์

4.3.2.1 ซูเปอร์คีย์

ซูเปอร์คีย์ (superkey) ย่อด้วย SK มีคุณสมบัติทำให้ทูเพิลในรีเลชันไม่ซ้ำกัน

$$t_1.SK \neq t_2.SK$$

หมายความว่าซูเปอร์คีย์ของทูเพิลที่หนึ่งต้องไม่เท่ากับซูเปอร์คีย์ของทูเพิลที่สอง

แต่ละรีเลชันมีซูเปอร์คีย์เสมอ ค่าปริยาย (default) ซูเปอร์คีย์ คือ เซตของแอตทริบิวต์ทุกตัวของรีเลชัน

หากซูเปอร์คีย์นั้นประกอบด้วยเซตของแอตทริบิวต์จำนวนน้อยที่สุดที่ทำให้ทุกทูเพิลไม่ซ้ำ ก็จะเรียกเซตของแอตทริบิวต์นั้นว่า คีย์

4.3.2.2 คีย์

คีย์ (Key) คือ ซูเปอร์คีย์ที่เล็กที่สุดที่ยังทำให้รีเลชันใด ๆ ไม่มีทูเพิลซ้ำ

ตัวอย่าง

Car (LicenseNumber, EngineSerialNumber, Make, Model, Year)

CAR	LicenseNumber	EngineSerialNumber	Make	Model	Year
	ง 290 กทม.	A69352	Ford	Mustang	2016
	กข 2123 ราชบุรี	B43696	Toyota	Camry	2015
	ศ 1012 ลำพูน	X83554	Honda	Civic	2016
	ศง 5199 สุรินทร์	C43742	Nissan	Almera	2017
	ฆ 9999 นครราชสีมา	Y82935	BMW	I8	2018
	พจ 4163 ฉะเชิงเทรา	U028365	Toyota	Altis	2018

รูปที่ 42 รีเลชัน CAR

รีเลชัน CAR ดังแสดงในรูปที่ 42 มีคีย์ที่เป็นไปได้ 2 ตัวคือ LicenseNumber หมายถึงทะเบียนรถซึ่งประกอบด้วยจังหวัดกับเลขทะเบียน ซึ่งไม่มีรถคันไหนที่ทะเบียนรถซ้ำกัน อีกคีย์ที่เป็นไปได้คือ EngineSerialNo คือเลขตัวถังรถยนต์ ซึ่งไม่มีโอกาสซ้ำเช่นกัน ในกรณีนี้ ให้เลือกเซตใดเซตหนึ่งเป็นคีย์ก็ได้

จะเห็นว่า หากซูเปอร์คีย์มีแอตทริบิวต์เดียวเป็นสมาชิก ซูเปอร์คีย์นั้นก็จะเป็นคีย์แน่นอน แต่ถ้าซูเปอร์คีย์มีแอตทริบิวต์มากกว่า 1 ตัว จะต้องตรวจสอบว่าสามารถลดแอตทริบิวต์ใดออกแล้วทำให้ทูเพิลไม่ซ้ำหรือไม่

นอกจากนั้นถ้าในรีเลชันนั้นมีคีย์คู่แข่ง (candidate key) ต้องเลือกคีย์ที่เหมาะสม 1 ตัว ให้เป็นคีย์หลัก (primary key) แล้วทำการขีดเส้นใต้ใต้ชื่อแอตทริบิวต์ที่เป็นคีย์หลัก

ในกรณีที่มีคีย์ที่เป็นไปได้มากกว่า 1 คีย์ เช่น จากตัวอย่างคือ {LicenseNumber} หรือ {SerialNo} เราเรียกคีย์ทั้งสองนี้ว่า แคนดิเดตคีย์ (candidate key) หรือคีย์คู่แข่ง ในทางปฏิบัติให้เลือกชุดใดชุดหนึ่งก็ได้มาเป็นคีย์ แล้วเรียกตัวที่เลือกว่า ไพรมารีคีย์ (primary key) หรือ คีย์หลัก ย่อว่า PK

4.3.3 เงื่อนไขบูรณภาพเอนทิตี

เงื่อนไขบูรณภาพเอนทิตี (entity integrity constraint) กล่าวว่า การที่เอนทิตีจะมีความบูรณภาพได้นั้น คีย์ทุกตัวต้องไม่ใช่ค่าว่าง (null)

$S = \{R_1, R_2, R_3, \ldots R_n\}$

ให้ S เป็นชื่อฐานข้อมูล R เป็นสคีมาของรีเลชันทั้งหมดที่อยู่ใน S

แอตทริบิวต์ที่เป็นคีย์หลัก (primary key : PK) ทุกทูเพิลในแต่ละรีเลชันของฐานข้อมูล S ต้องไม่ใช่ค่าว่าง

$t.PK \neq null$

4.3.4 เงื่อนไขบูรณภาพอ้างอิง

เราจะพิจารณาเงื่อนไขบูรณภาพอ้างอิง (referential integrity) เฉพาะเมื่อมีรีเลชัน 2 รีเลชัน ที่มีความสัมพันธ์กันและต้องอ้างอิงถึงกัน เราเรียกรีเลชัน 2 รีเลชันนี้ว่า รีเลชันผู้อ้าง (referencing relation) กับรีเลชันรับอ้าง (referenced relation)

การอ้างอิงระหว่างกันต้องใช้คีย์ชนิดหนึ่ง เรียกว่า ฟอเรนคีย์ (foreign key) เขียนย่อว่า FK ทั้งนี้ ถ้าแปลเป็นไทย อาจแปลตรงตัวได้ว่าคีย์ต่างชาติหรือคีย์แปลกปลอม ซึ่งไม่ตรงกับความหมายของฟอเรนคีย์ในฐานข้อมูล ตำรานี้จึงขอเลือกทับศัพท์ว่า ฟอเรนคีย์ เขียนย่อว่า FK

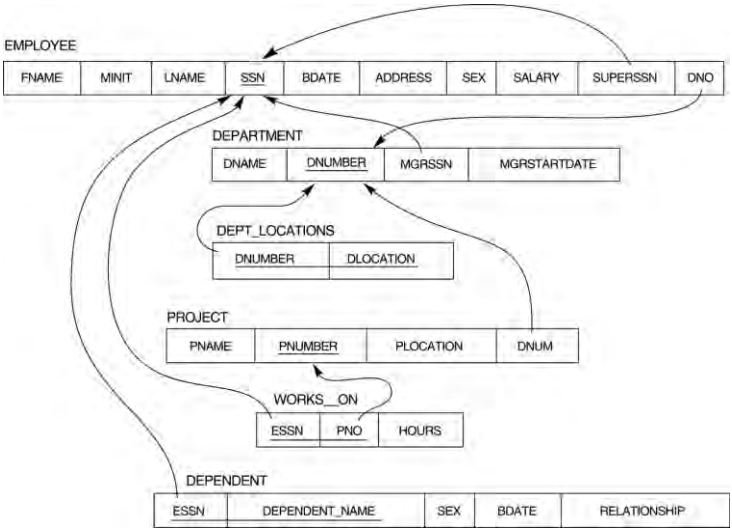
เงื่อนไขบูรณภาพอ้างอิง กำหนดว่า ค่าจาก FK ของรีเลชันผู้อ้าง มีความเป็นไปได้ 2 อย่างคือ 1) ต้องมีค่าเท่ากับ PK ของรีเลชันรับอ้างซึ่งต้องมีค่านั้นอยู่แล้ว หรือ 2) เป็นค่าว่าง ในกรณีที่เป็นค่าว่าง FK ต้องไม่ใช่ส่วนหนึ่งของคีย์หลักของรีเลชันนั้น นั่นคือ

t_1.FK = t_2.PK หรือ

t_1.FK = null

พิจารณารูปที่ 43 เส้นลูกศรแสดงความสัมพันธ์ระหว่างรีเลชันผู้อ้างไปยังรีเลชันรับอ้าง ดังนั้น แอตทริบิวต์ที่รีเลชันผู้อ้าง เรียกว่า ฟอเรนคีย์ FK ตัวอย่างจากรูปนี้มีจะเห็นว่ารีเลชัน PROJECT เมื่อเป็นรีเลชันผู้อ้าง จะมี DNUM เป็น FK อ้างอิงไปยัง DEPARTMENT ซึ่งมีหน้าที่เป็นรีเลชันรับอ้าง

ในกรณีนี้ค่าใน DNUM จะต้องมีปรากฏใน DNUMBER ที่ DEPARTMENT หรือเป็นค่าว่าง ซึ่งทำได้ เพราะ DNUM ไม่ใช่ PK ของ PROJECT



รูปที่ 43 รีเลชันบริษัทดีบีแปลงจากอีอาร์

4.3.5 เงื่อนไขบังคับอื่น ๆ

ยังมีเงื่อนไขบังคับที่สำคัญ เรียกว่า เงื่อนไขฟังก์ชันนัลดีเพนเดนซี (Functional dependency) หมายความว่า แอตทริบิวต์เซตหนึ่งขึ้นต่อแอตทริบิวต์อีกเซตหนึ่ง หรืออธิบายง่าย ๆ ว่า เมื่อรู้แอตทริบิวต์หนึ่งจะรู้อีกแอตทริบิวต์หนึ่งแน่ ๆ เขียนได้ว่า

$$X \longrightarrow Y$$

อ่านว่า X กำหนดฟังก์ชัน Y (X functionally determines Y) หรือ Y ขึ้นต่อ X (ฟังก์ชัน Y depends on X) อ่านอย่างย่อว่า X determines Y หรือ Y depends on X

รายละเอียดของเงื่อนไขนี้จะอยู่ในบทที่ 6 ของตำรานี้

4.4 การปรับปรุงข้อมูลในรีเลชัน

โดยธรรมชาติแล้วข้อมูลในฐานข้อมูลที่มีการใช้งานย่อมมีการเปลี่ยนแปลงเสมอ เรามองว่าข้อมูลต่าง ๆ ได้รับการปรับปรุง ซึ่งการปรับปรุงมีได้ 3 กรณี คือ

1) การแทรก (insert) คือ การเพิ่มทูเพิลใหม่ลงในรีเลชัน

2) การลบ (delete) คือ การลบทูเพิลออกจากรีเลชัน

3) การแก้ไข (update หรือ modify) คือ การเปลี่ยนแปลงข้อมูลของทูเพิลในรีเลชัน

สิ่งที่ต้องคำนึงถึงเสมอคือ เมื่อใดที่ปรับปรุงข้อมูลในรีเลชัน จะต้องไม่ผิดเงื่อนไขบังคับใด ๆ ทั้งสิ้น มิฉะนั้นข้อมูลในฐานข้อมูลอาจผิดพลาดได้

4.4.1 การแทรก

การแทรก (insert operation) คือการเพิ่มทูเพิลใหม่เข้าไปในรีเลชัน การแทรกข้อมูลมีความเสี่ยงต่อความผิดพลาดของข้อมูลที่ต้องระวัง คือ

- อาจกระทบเงื่อนไขโดเมน นั่นคือ ถ้าค่าใดในทูเพิลที่จะเพิ่มเข้าไปใหม่ไม่เป็นอะตอมมิก หรือไม่ตรงตามโดเมนของรีเลชันนั้น
- อาจกระทบเงื่อนไขคีย์ ถ้าค่าข้อมูลของคีย์ในทูเพิลใหม่มีอยู่แล้วในทูเพิลเดิม
- อาจกระทบเงื่อนไขบูรณภาพเอนทิตี ถ้าค่าในคีย์หลักของทูเพิลใหม่เป็นค่าว่าง
- อาจกระทบเงื่อนไขบูรณภาพอ้างอิง ถ้าค่าฟอเรนคีย์ไม่ตรงกับค่าคีย์หลักของรีเลชันที่อ้างถึง

4.4.2 การลบ

การลบ (delete operation) การลบข้อมูลมีความเสี่ยงต่อความผิดพลาดของข้อมูลที่ต้องระวัง เฉพาะเงื่อนไขบูรณภาพอ้างอิงเท่านั้น นั่นคือ หากทูเพิลถูกลบแล้วยังมีการอ้างอิงฟอเรนคีย์จากทูเพิลอื่นในฐานข้อมูล จะไม่สามารถลบข้อมูลได้ เหตุนี้สามารถแก้ไขได้ 4 วิธีคือ 1) ยกเลิกการลบ 2) ลบทูเพิลมาอ้างทูเพิลที่จะลบให้ครบ เรียกว่าแคสเคต (cascade) 3) แก้ค่าแอตทริบิวต์ให้เป็นค่าว่างหรือเปลี่ยนเป็นค่าอื่นที่ไม่กระทบ หรือ 4) เขียนโปรแกรมจัดการการลบนั้นโดยเฉพาะ

4.4.3 การแก้ไข

การแก้ไข (update operation) คือการปรับข้อมูลตามความต้องการ แต่การแก้ไขข้อมูลนี้มีความเสี่ยงต่อความผิดพลาดของข้อมูลที่ต้องระวังเหมือนกับแทรก คือทั้ง 4 กรณี ได้แก่ เงื่อนไขโดเมน เงื่อนไขคีย์ เงื่อนไขบูรณภาพเอนทิตี และเงื่อนไขบูรณภาพอ้างอิง

4.5 **บทสรุปและเรื่องชวนคิด**

บทนี้เราได้เรียนรู้แนวคิดและหลักการของฐานข้อมูลเชิงสัมพันธ์ ได้เข้าใจศัพท์ที่วงการฐานข้อมูลใช้ในการออกแบบฐานข้อมูลเชิงสัมพันธ์ ตลอดจนเงื่อนไขที่ต้องปฏิบัติเพื่อลดความผิดพลาดของข้อมูลให้ได้มากที่สุด ถึงแม้ในการทำงานอาจไม่ได้มีโอกาสได้คิดคำนวณเงื่อนไขต่าง ๆ ที่ระบุในบทนี้ แต่วิศวกรคอมพิวเตอร์ทุกคนต้องเข้าใจที่มา เพื่อให้นำไปประยุกต์ใช้ได้อย่างถูกต้อง และทำให้มีโอกาสหาวิธีใหม่ ๆ ในอนาคตได้ สำหรับบทนี้ ผู้เรียนคงเข้าใจแล้วว่าหลักการของรีเลชันนั้นอาศัยคณิตศาสตร์เรื่องเซต แล้วฐานข้อมูลใหม่ ๆ ตระกูลโนเอสคิวแอล จะใช้หลักการคณิตศาสตร์ใดเป็นพื้นฐานได้บ้าง

แบบฝึกหัดท้ายบท

1. หลักการของฐานข้อมูลเชิงสัมพันธ์อาศัยหลักการจากคณิตศาสตร์เรื่องใด
2. สิ่งใดถือเป็นสิ่งที่สำคัญที่สุดหรือเป็นหัวใจของฐานข้อมูลเชิงสัมพันธ์
3. การกำหนดโดเมนให้แต่ละแอตทริบิวต์ ควรพิจารณาจากสิ่งใด
4. จงอธิบายความแตกต่างระหว่างซูเปอร์คีย์และคีย์
5. จงบอกเงื่อนไขที่ทำให้ฟอเรนคีย์สามารถเป็นค่าว่างได้
6. “รีเลชันใด ๆ สามารถมีคีย์ได้เพียงตัวเดียวเท่านั้น” เป็นคำกล่าวที่ถูกต้องหรือไม่ จงอธิบาย
7. ฟอเรนคีย์ เป็นคำที่ใช้เรียกแอตทริบิวต์ในรีเลชันผู้อ้าง หรือรีเลชันผู้รับอ้าง
8. การลบข้อมูลออกจากฐานข้อมูลนั้น หากทูเพิลที่จะลบติดเงื่อนไขบูรณภาพอ้างอิง กล่างคือ มีการอ้างอิงฟอเรนคีย์จากทูเพิลอื่นในฐานข้อมูลมายังทูเพิลดังกล่าว สามารถแก้ไขได้กี่วิธี อย่างไรบ้าง
9. รีเลชัน Car_Rent ประกอบด้วยแอตทริบิวต์ Gear, Seat และ Fuel จงหาจำนวนทูเพิลสูงสุดของรีเลชัน Car_Rent ถ้า โดเมน(Gear) = {Manual, Auto} โดเมน (Seat) = {5, 7, 12} โดเมน (Fuel) = {Diesel, Benzine, LPG, NGV}
10. การปรับปรุงข้อมูลในรีเลชัน ไม่ว่าจะเป็นการแทรก การลบ การแก้ไข จะต้องพิจารณาถึงสิ่งใดเป็นสำคัญ

11. จงพิจารณารีเลชันนี้ แล้วทำให้อยู่ในรูปแบบการเชื่อมโยงของรีเลชัน

    Student(SSN, Name, Major, Bdate)

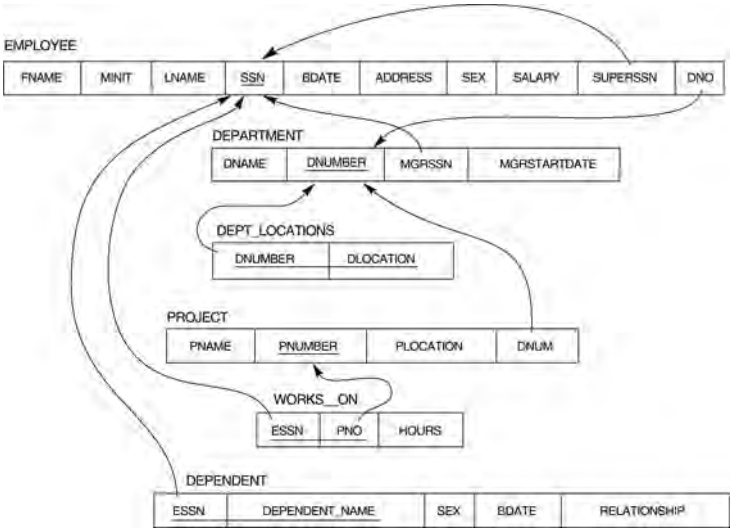
    Course(Course#, Cname, Dept)

    Enroll(SSN, Course#, Quarter, Grade)

    Book_Adoption(Course#, Quarter, Book_ISBN)

    Text(Book_ISBN, Book_title, Publisher, Author)

12. จงพิจารณาฐานข้อมูล Company แอตทริบิวต์ใดในรีเลชัน WORKS_ON ไม่สามารถมีค่าเป็น NULL ได้ เพราะเหตุใด



บทที่ 5
แบบจำลองกับฐานข้อมูลเชิงสัมพันธ์

วัตถุประสงค์

1. เพื่อให้เข้าใจกระบวนการออกแบบฐานข้อมูลจากระดับแนวคิดเป็นระดับตรรกะ
2. เพื่อให้สามารถแปลงแผนภาพอีอาร์เป็นรีเลชันได้
3. เพื่อให้สามารถแปลงแผนภาพอีออาร์เป็นรีเลชันได้

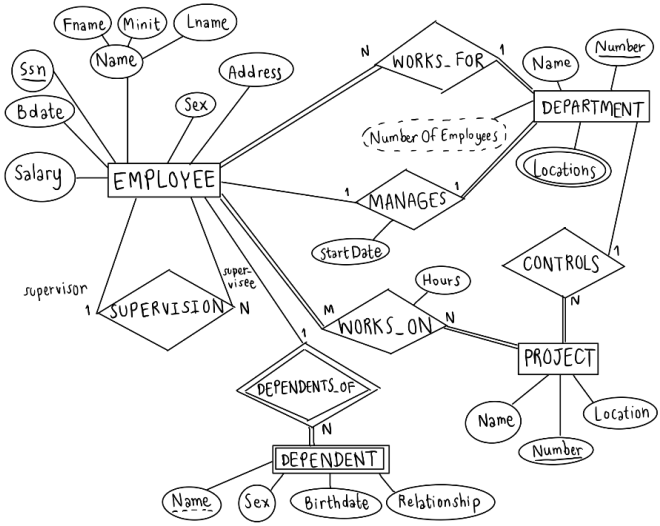
ที่ผ่านมาเราได้เรียนเรื่องการออกแบบฐานข้อมูลด้วยแบบจำลองอีอาร์และใช้แผนภาพอีอาร์เพื่อสื่อสารกับผู้ใช้ ส่วนนั้นคือแบบจำลองระดับแนวคิด (conceptual level)

จากนั้นเราได้เรียนเรื่องรีเลชัน คือวิธีการทำงานของฐานข้อมูลเชิงสัมพันธ์ ซึ่งเป็นแบบจำลองระดับตรรกะ (logical level)

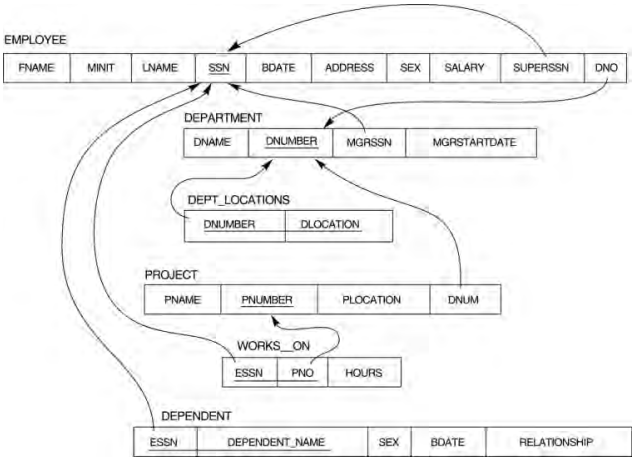
ในบทนี้ เราจะแปลงแบบจากอีอาร์หรืออีอีอาร์ให้อยู่ในรูปแบบของรีเลชันที่ฐานข้อมูลและคอมพิวเตอร์นำไปประมวลผลได้ [3, 13]

5.1 ขั้นตอนการแปลงอีอาร์และอีอีอาร์เป็นรีเลชัน

ขั้นตอนการแปลงอีอาร์มี 7 ขั้นตอน และการแปลงอีอีอาร์มีเพิ่มอีก 2 ขั้นตอน รูปที่ 45 แสดงรีเลชันผลลัพธ์ที่ได้จากการแปลงอีอาร์ระบบฐานข้อมูลของบริษัทดีบีจากรูปที่ 44 ตัวอย่างและขั้นตอนในบทนี้อ้างอิงมาจาก [2] การแปลงมีรายละเอียดดังต่อไปนี้



รูปที่ 44 แผนภาพอีอาร์แสดงระบบฐานข้อมูลของบริษัทดีบี



รูปที่ 45 รีเลชันแสดงระบบฐานข้อมูลของบริษัทดีบี

5.1.1 ขั้นตอนที่ 1 : แปลงเอนทิตี

แปลงเอนทิตีไทป์จากอีอาร์ให้เป็นรีเลชัน ในขั้นนี้ เราสนใจเฉพาะเอนทิตีไทป์ปกติ จะยังไม่รวมถึงเอนทิตีแบบวีค ขอเรียกเอนทิตีไทป์ปกตินี้ว่าเอนทิตีไทป์

การแปลงขั้นนี้ ให้สร้างรีเลชันใหม่ให้กับเอนทิตีไทป์ทุกตัว หนึ่งรีเลชันต่อหนึ่งเอนทิตีไทป์ จากนั้นให้นำแอตทริบิวต์ที่อยู่ในแต่ละเอนทิตีมาเป็นแอตทริบิวต์ของรีเลชันใหม่นั้น แล้วเลือกแอตทริบิวต์ที่เป็นคีย์ของเอนทิตี ให้เป็นคีย์หลัก (primary key) ของรีเลชันนั้น

ดังนั้น จากขั้นตอนที่ 1 เราจะได้รีเลชัน EMPLOYEE, DEPARTMENT และ PROJECT โดยมี SSN, DNumber และ PName เป็นคีย์หลักของรีเลชันตามลำดับ ดังแสดงในรูปที่ 46

EMPLOYEE

SSN	Fname	MInt	Lname	Bdate	Sex	Address	Salary

DEPARTMENT

Dname	Dnumber	DLocations

PROJECT

Pname	Pnumber	PLocation

รูปที่ 46 แปลงเอนทิตีไทป์

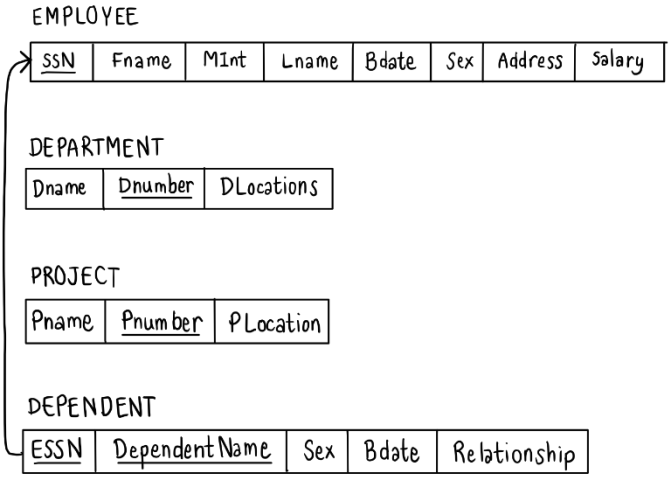
5.1.2 ขั้นตอนที่ 2 : แปลงวีคเอนทิตี

แปลงวีคเอนทิตีไทป์ (weak entity type) ในแผนภาพอีอาร์ ให้เป็นรีเลชัน โดยการสร้างรีเลชันใหม่ แล้วนำแอตทริบิวต์ทั้งหมดที่อยู่ในวีคเอนทิตีไทป์มาเป็นแอตทริบิวต์ของรีเลชันใหม่ให้ครบ

จากนั้นให้สร้างความสัมพันธ์ระหว่างรีเลชันใหม่กับรีเลชันที่เป็นเจ้าของวีคเอนทิตี กำหนดบทบาทของรีเลชันใหม่ว่าเป็น รีเลชันอ้างถึง และรีเลชันเจ้าของคือ รีเลชันรับอ้าง

จากนั้น ให้นำ PK จากรีเลชันรับอ้างมาเพิ่มเป็น PK ร่วมกับคีย์เดิมที่เป็นคีย์บางส่วน (partial key) แล้วกำหนดให้ PK นั้นเป็นฟอเรนคีย์ (foreign key : FK) จากรีเลชันอ้างถึง ชี้ไปยังคีย์หลัก (primary key : PK) ของรีเลชันรับอ้าง

จากตัวอย่าง ให้สร้างรีเลชันชื่อ DEPENDENT นำ SSN จาก EMPLOYEE มาเป็นคีย์ร่วมกับ DEPENDENT_NAME เปลี่ยนชื่อ SSN เป็น ESSN เพื่อไม่ให้ซ้ำ ดังนั้น PK ของรีเลชัน DEPENDENT คือ {ESSN, DEPENDENT_NAME} แล้วกำหนดให้ ESSN เป็น FK ชี้ไปยัง SSN ในรีเลชัน EMPLOYEE ดังแสดงในรูปที่ 47



รูปที่ 47 แปลงวีคเอนทิตีไทป์

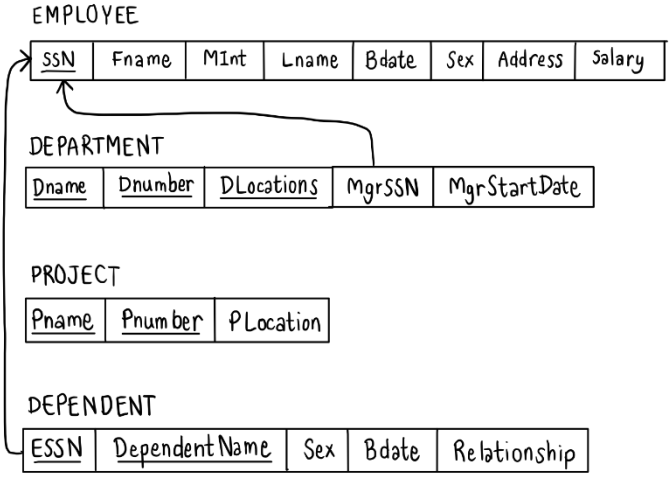
5.1.3 ขั้นตอนที่ 3 : แปลงรีเลชันชิปไทป์ 1:1

รีเลชันชิปใน ER แบบ 1:1 สามารถแปลงเป็นรีเลชันได้ 3 วิธี

วิธีที่ 1 ใช้ FK

เหมาะกับรีเลชันชิปเป็นแบบไบนารี ที่มีเงื่อนไขบังคับเข้าร่วมแบบบังคับ (total participation) ข้างหนึ่ง และแบบไม่บังคับ (partial participation) อีกข้างหนึ่ง ให้กำหนดรีเลชันฝั่งที่บังคับเป็นหลัก แล้วนำ PK จากรีเลชันอีกฝั่งหนึ่งมาเพิ่มเป็นแอตทริบิวต์ใหม่เข้าไปในรีเลชัน จากนั้นให้กำหนดแอตทริบิวต์นี้เป็น FK ชี้กลับไปยังรีเลชันอีกฝั่ง ถ้ารีเลชันชิปนั้นมีแอตทริบิวต์ของตัวเอง ให้รวมแอตทริบิวต์นั้นมาในรีเลชันฝั่งหลักด้วย

จากตัวอย่าง พบว่า รีเลชันชิป MANAGES เป็นแบบ 1:1 และมีเงื่อนไขบังคับเข้าร่วมแบบบังคับกับ DEPARTMENT และแบบไม่บังคับกับ EMPLOYEE ดังนั้น เราจึงยึด DEPARTMENT เป็นหลัก แล้วนำ PK ของ EMPLOYEE คือ SSN มาใส่เพิ่มใน DEPARTMENT เปลี่ยนชื่อจาก SSN เป็น MgrSSN เพื่อให้ไม่ซ้ำ แล้วโยง FK MgrSSN จาก DEPARTMENT ไปยัง SSN ใน EMPLOYEE สุดท้าย ให้รวมแอตทริบิวต์ MgrStartDate ไว้ในรีเลชัน DEPARTMENT ด้วย ดังแสดงในรูปที่ 48



รูปที่ 48 แปลงรีเลชันชิปไทป์ 1:1

วิธีที่ 2 ผนวกรีเลชัน

วิธีนี้เหมาะกับรีเลชันชิปที่เงื่อนไขบังคับเข้าร่วมเป็นแบบบังคับ (total participation) ทั้งสองข้างของรีเลชันชิป กรณีนี้ให้นำแอตทริบิวต์จากรีเลชัน

ทั้งสองมารวมกันทั้งหมด และหากรีเลชันชิปนั้นมีแอตทริบิวต์ก็ให้รวมไปด้วย กำหนดคีย์ของรีเลชันใหม่นี้ คือคีย์จากรีเลชันทั้งสองที่ผนวกกัน จะเกิดคีย์ร่วม

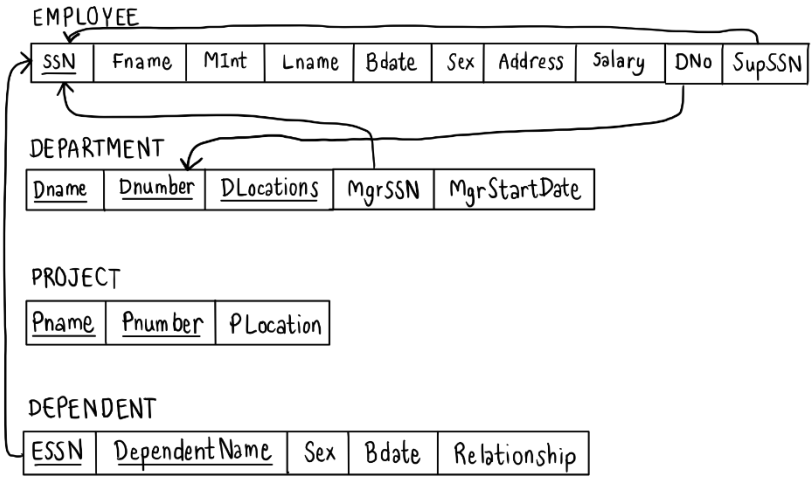
วิธีที่ 3 ครอสเรเฟอเรนซ์

วิธีครอสเรเฟอเรนซ์ (cross reference) นี้ ให้สร้างรีเลชันใหม่ที่แอตทริบิวต์ประกอบด้วยคีย์จากรีเลชันทั้งสอง และแอตทริบิวต์ของรีเลชันชิปนั้น ถ้ามีจากนั้น ให้สร้าง FK จากคีย์เหล่านั้น โยงกลับไปที่รีเลชันเดิม

5.1.4 ขั้นตอนที่ 4 : แปลงรีเลชันชิบไทป์ 1:N

ให้ยึดรีเลชันด้าน N เป็นหลัก แล้วนำ PK จากด้าน 1 มาเพิ่มเป็นแอตทริบิวต์ในข้าง N และสร้าง FK โยงกลับไปด้าน 1 หากมีแอตทริบิวต์ที่รีเลชันชิปนี้ก็ให้นำมารวมที่รีเลชั้นหลักนี้ด้วย

จากตัวอย่าง รีเลชัน EMPLOYEE เป็นด้าน N และ รีเลชัน DEPARTMENT เป็นด้าน 1 จึงนำ Dnumber ซึ่งเป็น PK ใน DEPARTMENT มาเพิ่มในรีเลชัน EMPLOYEE เปลี่ยนชื่อ Dnumber เป็น DNo แล้วโยง DNo เป็น FK กลับไปยัง Dnumber ใน DEPARTMENT ดังแสดงในรูปที่ 49

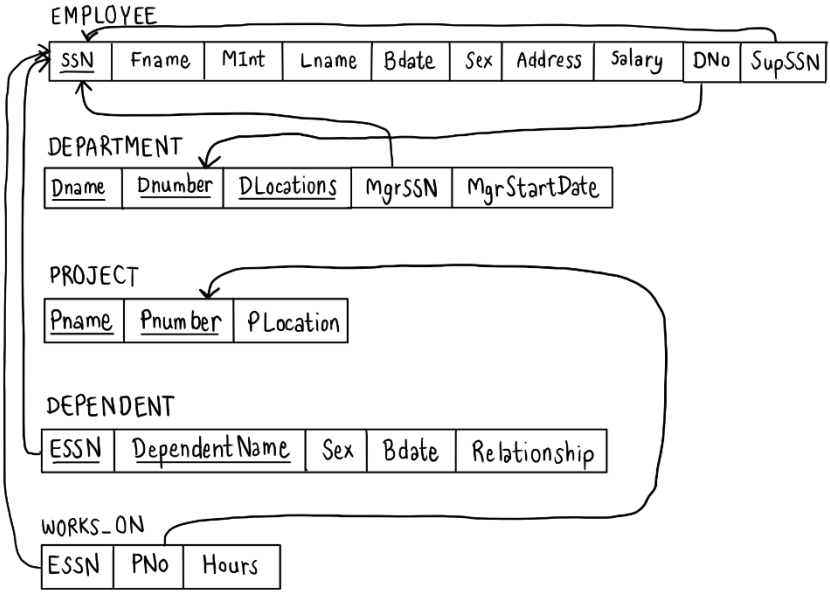


รูปที่ 49 แปลงรีเลชันชิบไทป์ 1:N

5.1.5 ขั้นตอนที่ 5 : แปลงรีเลชันชิบไทป์ M:N

ให้ใช้วิธีครอสเรเฟอเรนซ์ที่แสดงในขั้นตอนที่ 3

จากตัวอย่าง เราแปลงรีเลชันชิป WORKS_ON ที่เป็นแบบ M:N โดยสร้างรีเลชันใหม่ ชื่อ WORKS_ON ที่ประกอบด้วย SSN จาก EMPLOYEE เปลี่ยนชื่อเป็น ESSN และ PNumber จาก PROJECT เปลี่ยนชื่อเป็น PNo มาเป็นคีย์ร่วม โยง FK ทั้งสองกลับไปที่รีเลชันหลัก แล้วรวม Hours มาด้วย ดังแสดงในรูปที่ 50



รูปที่ 50 แปลงรีเลชันชิบไทป์ m:n

5.1.6 ขั้นตอนที่ 6 : แปลงแอตทริบิวต์หลายค่า

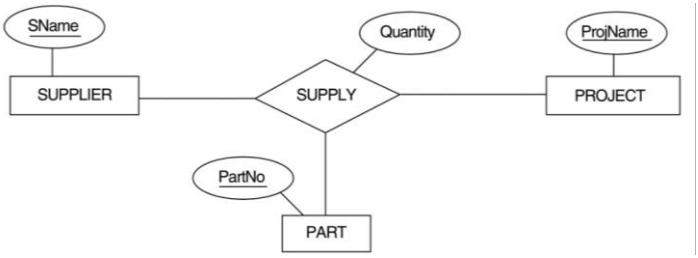
เมื่อพบแอตทริบิวต์หลายค่า (multivalued attribute) ให้สร้างรีเลชันใหม่ประกอบด้วยแอตทริบิวต์หลายค่านั้นพร้อมกับคีย์ของรีเลชันที่พบแอตทริบิวต์หลายค่า แล้วกำหนดคีย์จากรีเลชันเดิมร่วมกับแอตทริบิวต์หลายค่ามาเป็นคีย์ของรีเลชันใหม่นี้ แล้วโยง FK กลับไปยังรีเลชันหลักด้วย

จากตัวอย่าง แอตทริบิวต์ Location เป็นแอตทริบิวต์แบบหลายค่า ปรากฏอยู่ที่ DEPARTMENT ดังนั้น ให้สร้างรีเลชันใหม่ ชื่อ DEPT_LOCATIONS

ประกอบด้วย แอตทริบิวต์ Location กับ Dnumber กำหนดให้ทั้งสองแอตทริบิวต์เป็นคีย์ร่วม แล้วโยง FK กลับไป

5.1.7 ขั้นตอนที่ 7 : แปลงรีเลชันชิปแบบเอนนารี

ในกรณีที่รีเลชันชิปมีเอนทิตีมากกว่า 2 เอนทิตีขึ้นไปมาสัมพันธ์กัน ให้ใช้วิธีครอสเรเฟอเรนซ์ที่แสดงในขั้นตอนที่ 3 คือให้สร้างรีเลชันใหม่ แล้วนำคีย์จากทุกรีเลชันที่เกี่ยวข้องมากำหนดเป็นคีย์ร่วม พร้อมทั้งโยง FK กลับไปยังรีเลชันหลักด้วย ถ้าพบแอตทริบิวต์ของรีเลชันชิปก็ให้นำไปรวมด้วย จากตัวอย่าง พบเทอนารีหรือเอนนารีรีเลชันชิป ดังแสดงในรูปที่ 51 คือมี เอนทิตี SUPPLIER, PART และ PROJECT ดังนั้นให้สร้างรีเลชันใหม่ชื่อ SUPPLY ประกอบด้วย SName, ProjName และ PartNo แล้วโยง FK กลับไป พร้อมทั้งรวมแอตทริบิวต์ของรีเลชันชิปนั้นเข้าไปด้วย ซึ่งก็คือรวมแอตทริบิวต์ Quantity เข้าไปด้วย ดังแสดงในรูปที่ 52



รูปที่ 51 แผนภาพอีอาร์แสดงรีเลชันชิปแบบเทอนารี

SName	. . .

ProjName	. . .

PartNo	. . .

SName	ProjName	PartNo	Quantity

รูปที่ 52 ผลการแปลงรีเลชันชิปแบบเทอนารี

5.1.8 ขั้นตอนที่ 8 : แปลงซับคลาสซูเปอร์คลาส

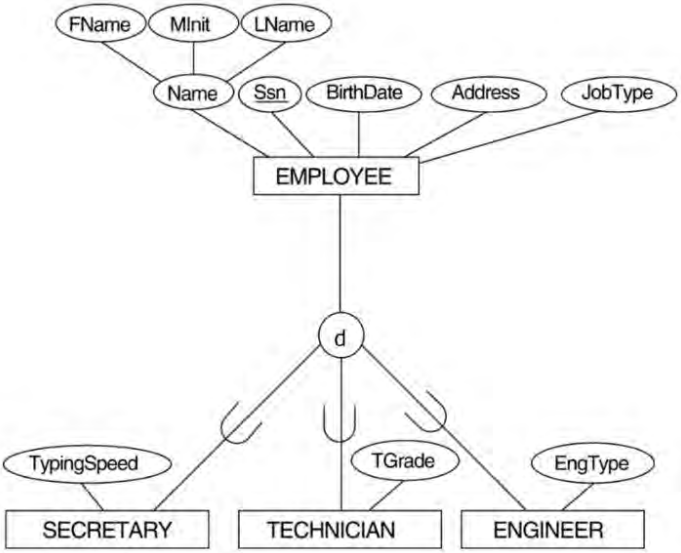
หากแผนภาพที่ใช้เป็นแผนภาพแบบอีอีอาร์ที่แสดงความสัมพันธ์ของข้อมูลแบบสเปเชียลไลเซชันหรือเจเนอรัลไลเซชัน จะมีวิธีให้เลือก 4 วิธี คือ

วิธีที่ 1 แยกรีเลชันซูเปอร์คลาสและซับคลาส

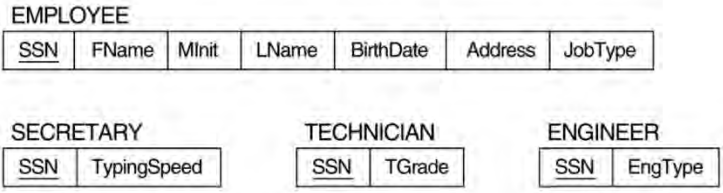
วิธีแยกรีเลชันซูเปอร์คลาสและซับคลาส (multiple relation – superclass and subclass) ให้สร้างรีเลชันใหม่สำหรับทุกซูเปอร์คลาสและซับคลาส สำหรับรีเลชันที่มาจากซูเปอร์คลาส ให้รวมแอตทริบิวต์ทั้งหมด แล้วกำหนด PK จากคีย์เดิม ส่วนรีเลชันที่มาจากซับคลาส จะประกอบด้วยแอตทริบิวต์ของตัวเองและคีย์ของซูเปอร์คลาส ดังแสดงการแปลงจากอีอีอาร์ในรูปที่ 53 เป็นรีเลชันในรูปที่ 54

ให้รวมแอตทริบิวต์จากทั้งซูเปอร์คลาสและซับคลาสมาไว้ในรีเลชันเดียว แล้วใช้คีย์จากซูเปอร์คลาสเป็นคีย์หลัก

วิธีนี้ใช้ได้กับสเปเชียลไลเซชันทุกแบบไม่ว่าจะเป็น Total, Partial, Disjoint หรือ Overlap



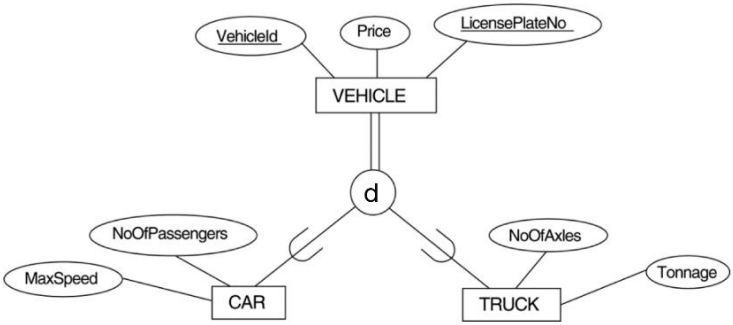
รูปที่ 53 แผนภาพอีอีอาร์แสดงซูเปอร์คลาสซับคลาส EMPLOYEE



รูปที่ 54 ตัวอย่างการแปลงด้วยวิธีวิธีแยกรีเลชันซูเปอร์คลาสและซับคลาส

วิธีที่ 2 แยกรีเลชันใช้ซับคลาส

วิธีแยกรีเลชันใช้ซับคลาส (multiple relation – subclass only) ให้สร้างรีเลชันใหม่จากซับคลาสทุกตัว แล้วนำแอตทริบิวต์ทั้งหมดจากซูเปอร์คลาสมารวม พร้อมทั้งระบุคีย์ที่มาจากซูเปอร์คลาสนั้น ๆ วิธีนี้ใช้ได้กับสเปเชียลไลเซชันแบบ total รูปที่ 55 และรูปที่ 56 แสดงการแปลงดังกล่าว



รูปที่ 55 แผนภาพอีอีอาร์แสดงซูเปอร์คลาสซับคลาส VEHICLE

CAR

VehicleId	LicensePlateNo	Price	MaxSpeed	NoOfPassengers

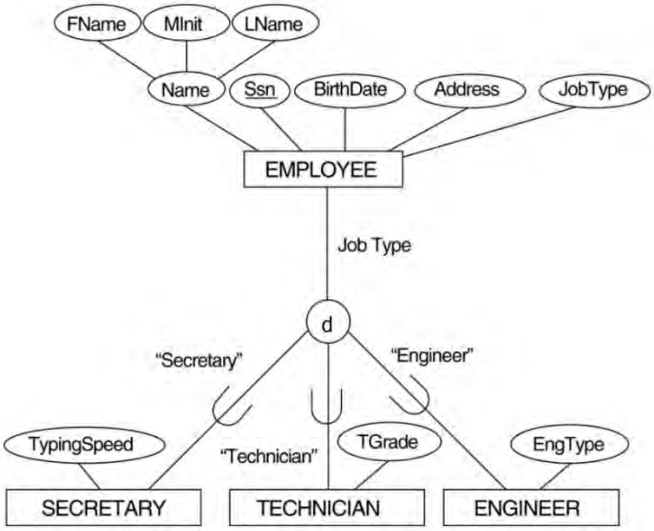
TRUCK

VehicleId	LicensePlateNo	Price	NoOfAxles	Tonnage

รูปที่ 56 ตัวอย่างการแปลงด้วยวิธีแยกรีเลชันใช้ซับคลาส

วิธีที่ 3 รวมรีเลชันพร้อมแอตทริบิวต์ระบุตัวเดียว

วิธีรวมรีเลชันพร้อมแอตทริบิวต์ระบุเดียว (single relation – one type attribute) ให้สร้างรีเลชันใหม่ รวมแอตทริบิวต์จากซูเปอร์คลาสและซับคลาสทั้งหมดไว้ ตลอดจนแอตทริบิวต์ที่เป็นเพรดิเคตกำหนด (predicate-defined attribute) บางตำราเรียกแอตทริบิวต์ประเภทนี้ว่า ไทป์ (type) วิธีนี้เหมาะกับสเปเชียลไลเซชันแบบแอตทริบิวต์กำหนด (attribute defined-specialization) รูปที่ 58 แสดงผลการแปลงจากรูปที่ 57



รูปที่ 57 แผนภาพอีอีอาร์แสดงซูเปอร์คลาสซับคลาสแบบแอตทริบิวต์กำหนด

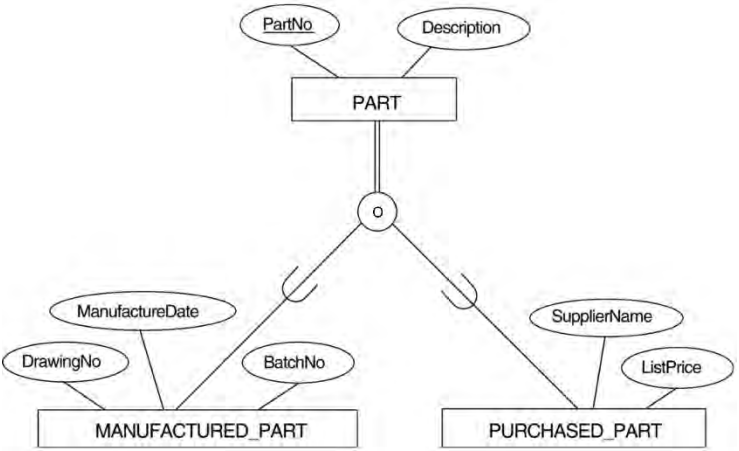
EMPLOYEE

SSN	FName	MInit	LName	BirthDate	Address	JobType	TypingSpeed	TGrade	EngType

รูปที่ 58 ตัวอย่างการแปลงด้วยวิธีรวมรีเลชันพร้อมแอตทริบิวต์ระบุตัวเดียว

วิธีที่ 4 รวมรีเลชันพร้อมแอตทริบิวต์ระบุหลายตัว

วิธีรวมรีเลชันพร้อมแอตทริบิวต์ระบุหลายตัว (Single Relation – Multiple Type Attribute) ให้รวมทุกแอตทริบิวต์เป็นรีเลชันเดียว แล้วเพิ่มแอตทริบิวต์เท่ากับจำนวนซับคลาส กำหนดโดเมนแอตทริบิวต์ที่เพิ่มมานี้ให้เป็นบูลีนทั้งหมด เช่น ถ้ามี 3 ซับคลาสก็เพิ่มแอตทริบิวต์ชนิดบูลีน 3 ตัว บูลีนใหม่นี้ทำหน้าที่ระบุว่าทูเพิลนั้น ๆ เป็นสมาชิกของซับคลาสใด รูปที่ 60 แสดงการแปลงจากรูปที่ 59 วิธีนี้ใช้ได้กับสเปเชียลไลเซชันทุกแบบไม่ว่าจะเป็น total, partial, disjoint หรือ overlap



รูปที่ 59 แผนภาพอีอีอาร์แสดงซูเปอร์คลาสซับคลาสแบบ overlap

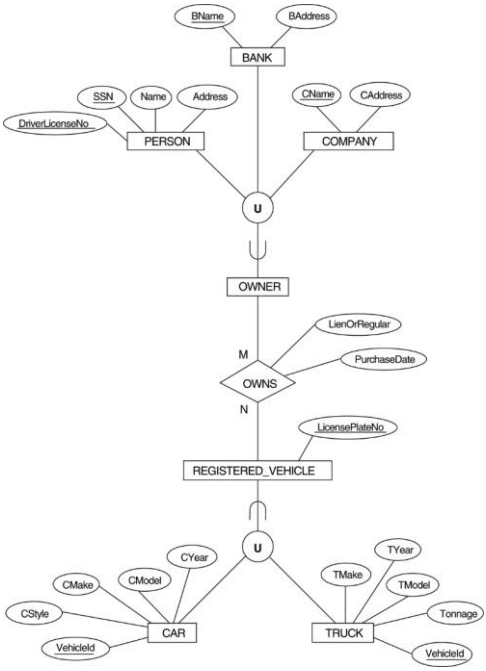
PART

PartNo	Description	MFlag	DrawingNo	ManufactureDate	BatchNo	PFlag	SupplierName	ListPrice

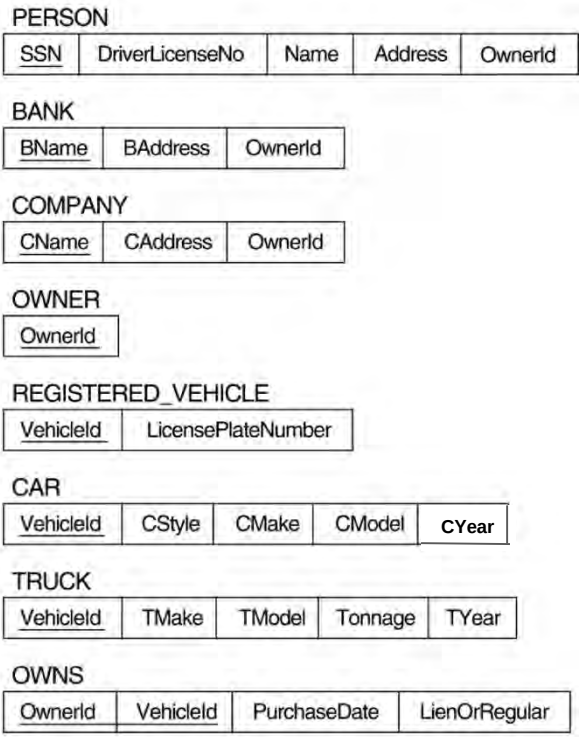
รูปที่ 60 วิธีรวมรีเลชันพร้อมแอตทริบิวต์ระบุหลายตัว

5.1.9 ขั้นตอนที่ 9 : แปลงยูเนียน

เนื่องจากความสัมพันธ์แบบยูเนียน มีซับคลาสประเภทแคทะกอรีซึ่งมีซูเปอร์คลาสหลายซูเปอร์คลาสที่เป็นเอนทิตีที่ต่างกัน ดังนั้นการรับทอดเป็นแบบเลือก (selective inheritance) ดังนั้น การแปลงอีอีอาร์แบบยูเนียนจำเป็นต้องสร้างคีย์ใหม่ เรียกว่า คีย์เซอโรเกต (surrogate key) ให้กับแคทะกอรีนั้น ๆ จากนั้นให้สร้างรีเลชันใหม่ให้กับซูเปอร์คลาสทั้งหมด แล้วนำคีย์เซอโรเกตไปใช้เป็นคีย์หลักสำหรับทุกรีเลชันที่เกิดขึ้นใหม่ รูปที่ 62 แสดงการแปลงจากรูปที่ 61



รูปที่ 61 แผนภาพอีอีอาร์แสดงแคทะกอรี



รูปที่ 62 แสดงการแปลงแบบยูเนียน

5.2 **บทสรุปและเรื่องชวนคิด**

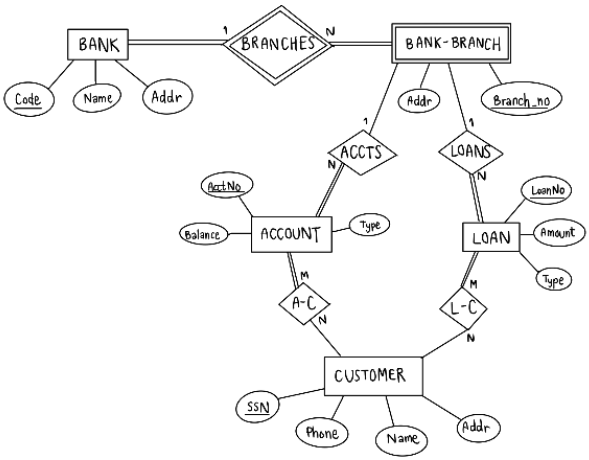
บทนี้เราได้เรียนรู้กระบวนการออกแบบฐานข้อมูลจากระดับแนวคิดเป็นระดับตรรกะ ตลอดจนขั้นตอนการแปลงแผนภาพอีอาร์และอีอีอาร์เป็นรีเลชันได้ จะเห็นได้ว่าคณิตศาสตร์ที่ใช้กับการทำงานของฐานข้อมูลเชิงสัมพันธ์นั้นไม่ซับซ้อนมากนัก จึงเป็นเหตุผลว่าเครื่องมือออกแบบฐานข้อมูลมักมีความสามารถที่จะแปลงแบบที่เราวาดด้วยแผนภาพอีอาร์หรืออีอีอาร์ให้

อยู่ในรูปแบบรีเลชันได้ จากบทนี้ อยากให้ผู้เรียนได้เห็นว่าการออกแบบที่ดีเป็นสิ่งสำคัญ เมื่อต้องการแก้ไขแบบ ควรแก้ที่แผนภาพก่อนเสมอ และอยากให้ผู้เรียนศึกษาต่อว่ามีแบบจำลองและเครื่องมืออื่นที่รองรับฐานข้อมูลแบบโนเอสคิวแอลหรือไม่

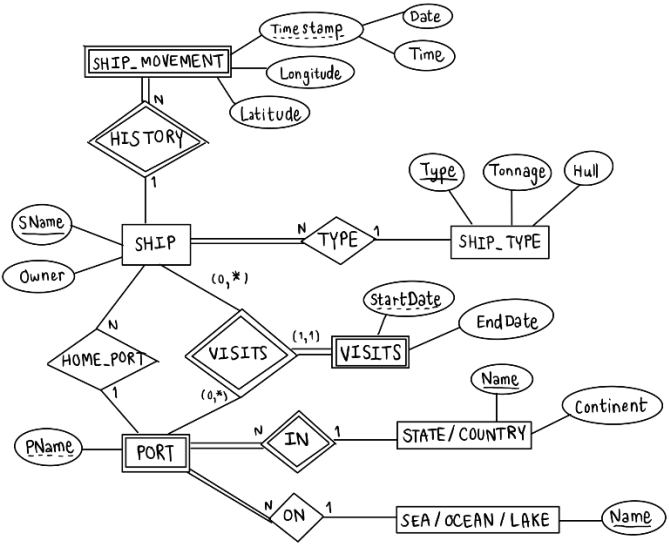
แบบฝึกหัดท้ายบท

1. ขั้นตอนการแปลงอีอาร์และอีอีอาร์เป็นรีเลชัน มีกี่ขั้นตอน อะไรบ้าง
2. วิธีการแปลงซับคลาสซูเปอร์คลาสในขั้นตอนที่ 8 สำหรับแผนภาพแบบอีอีอาร์นั้นมีกี่วิธี อะไรบ้าง และแต่ละวิธีเหมาะกับสเปเชียลไลเซชันแบบใดบ้าง
3. การแปลงรีเลชันชิปแบบใด ระหว่าง 1:1, 1:N หรือ M:N จำเป็นต้องสร้างรีเลชันใหม่เพิ่มเสมอหรือไม่
4. การแปลงรีเลชันชิปแบบ 1:N ฝั่งใดที่จะต้องเพิ่มฟอเรนคีย์ ซึ่งเก็บคีย์หลักของอีกฝั่งหนึ่ง
5. การแปลงรีเลชันแบบ 1:1 มีกี่วิธี อะไรบ้าง และแต่ละวิธีเมื่อแปลงแล้วจะเกิดรีเลชันสุดท้ายจำนวนกี่รีเลชัน
6. การเลือกวิธีการแปลงรีเลชันแบบ 1:1 แต่ละวิธี ควรใช้หลักเกณฑ์ใดในการพิจารณาเลือกวิธีการแปลง และพิจารณาอย่างไร
7. การแปลงรีเลชันชิปแบบเอนนารี หากกำหนดให้มีค่าดีกรีความสัมพันธ์เป็น N แล้ว เมื่อแปลงเสร็จจะมีจำนวนรีเลชันเท่าใด

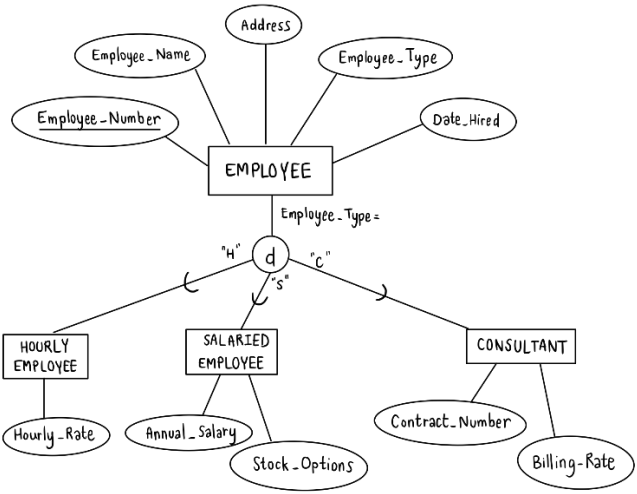
8. จงแปลงจากแผนภาพอีอาร์ให้เป็นรีเลชัน



9. จงแปลงจากแผนภาพอีอาร์ให้เป็นรีเลชัน



10. จงแปลงจากแผนภาพอีอีอาร์ให้เป็นรีเลชัน



บทที่ 6
การนอร์มาไลซ์

วัตถุประสงค์

1. เพื่อให้ทราบถึงคุณสมบัติของฐานข้อมูลที่ดี
2. เพื่อให้เข้าใจเรื่องฟังก์ชันนัลดีเพนเดนซี
3. เพื่อให้เข้าใจรูปแบบนอร์มัลฟอร์มที่หนึ่ง ที่สอง ที่สาม และแบบบีซีเอ็นเอฟ
4. เพื่อให้เข้าใจกระบวนการนอร์มาไลซ์

ฐานข้อมูลที่ดีคือฐานข้อมูลที่มีความผิดพลาดน้อยที่สุด และมีความซ้ำซ้อน (redundancy) น้อยที่สุด เพราะเมื่อข้อมูลซ้ำซ้อนก็จะทำให้เปลืองพื้นที่การจัดเก็บ และอาจส่งผลให้ข้อมูลผิดพลาดหรือข้อมูลเพี้ยน (anomaly) ไปได้ เมื่อปรับปรุงเปลี่ยนแปลงข้อมูลซึ่งเกิดได้เมื่อมีการแทรก ลบ หรือแก้ไขข้อมูลนั่นเอง

วิธีการลดความซ้ำซ้อนทำได้ด้วยการแตกรีเลชันออกเป็นรีเลชันย่อยซึ่งเรียกว่า กระบวนการย่อย (decomposition) และปัจจัยที่ต้องคำนึงถึงในกระบวนการย่อยนี้ ก็คือความขึ้นต่อกันของข้อมูลในฐานข้อมูล เราเรียกปัจจัยนี้ว่า ฟังก์ชันนัลดีเพนเดนซี

ส่วนการ นอร์มาไลซ์ คือกระบวนการย่อยที่คำนึงถึงฟังก์ชันนัลดีเพนเดนซีร่วมกับคีย์ ให้ตรงกับ นอร์มัลฟอร์ม ระดับต่าง ๆ ซึ่งเราจะกล่าวถึงต่อไป [3, 10, 12, 13, 15, 25, 43]

6.1 ปัญหาจากความซ้ำซ้อน

เมื่อมีความซ้ำซ้อน (redundancy) ของข้อมูลในรีเลชัน จะมีโอกาสเกิดความผิดพลาดเมื่อมีการปรับปรุงข้อมูลได้มาก ความผิดพลาดจากการปรับปรุงข้อมูล แบ่งได้เป็น 3 กรณีคือ

6.1.1 ความผิดพลาดจากการแก้ไข

ความผิดพลาดจากการแก้ไข (update anomaly) อาจเกิดขึ้นได้เมื่อเราแก้ไขข้อมูลไม่ครบถ้วน

ตัวอย่าง

EMP_PROJ(Emp#, Proj#, Ename, Pname, No_hours)

ความผิดพลาดจากการแก้ไขอาจเกิดขึ้นได้ เช่น สมมุติว่าหากต้องการเปลี่ยนชื่อ Pname จาก “Human Resource” เป็น “Human Development” นั้น เราจะต้องไล่แก้ข้อมูลดังกล่าวในทุกทูเพิล หากแก้ไม่ครบ ข้อมูลก็จะผิดแน่นอน

6.1.2 ความผิดพลาดจากการแทรก

ขอยกตัวอย่างความผิดพลาดจากการแทรก (insert anomaly) จากตัวอย่างก่อนหน้า นั่นคือ เราไม่สามารถแทรก Project ใหม่เข้าไปในรีเลชัน จนกว่าจะกำหนด Employee ให้มี Project เสียก่อน

6.1.3 ความผิดพลาดจากการลบ

ความผิดพลาดจากการลบ (delete anomaly) จากตัวอย่างก่อนหน้า อาจเกิดขึ้นเมื่อต้องการลบ Project หนึ่งออกไปจะส่งผลให้มีการลบ Employee ที่ทำงานอยู่ใน Project นั้นทั้งหมด หรือ อีกทางหนึ่งถ้า Employee นั้นทำงานอยู่เพียง Project เดียว การลบ Employee นั้นจะส่งผลให้ Project นั้นหายไปจากรีเลชัน

ดังนั้น วิธีการที่ดีในการออกแบบฐานข้อมูลที่จะลดปัญหาความผิดพลาดนี้ได้คือการย่อยรีเลชันนั่นเอง

6.2 ฟังก์ชันนัลดีเพนเดนซี

ฟังก์ชันนัลดีเพนเดนซี (functional dependency) คือ เงื่อนไขความขึ้นต่อกันของแอตทริบิวต์ในรีเลชัน ได้มาจากการตีความในขั้นตอนการรวบรวมและวิเคราะห์ความต้องการของระบบ เป็นความหมายของข้อมูลที่ตีความโดยผู้ออกแบบตามความตั้งใจของเจ้าของระบบ โดยเน้นที่ความสัมพันธ์ระหว่างแอตทริบิวต์ในฐานข้อมูลนั้น ๆ

ฟังก์ชันนัลดีเพนเดนซี อาจแปลเป็นไทยได้ว่าความขึ้นต่อกันทางฟังก์ชัน แต่เนื่องจากฟังแล้วเข้าใจยาก จึงขอใช้ทับศัพท์ว่า ฟังก์ชันนัลดีเพนเดนซี ย่อว่า FD ต่อไปนี้จะเรียกว่า FD

ศัพท์คำว่า กำหนด (determine) และคำว่า ขึ้นกับ (depend) เป็นหัวใจสำคัญของ FD เมื่อเขียนสัญกรณ์คณิตศาสตร์แสดงฟังก์ชันนัลดีเพนเดนซี ให้ใช้เครื่องหมายลูกศร ดังนี้

$$X \rightarrow Y$$

อ่านว่า X กำหนดฟังก์ชัน Y (X functionally determines Y) หรืออ่านว่า ฟังก์ชัน Y ขึ้นกับ X (Y depends on X) เมื่อ X และ Y คือเซตของแอตทริบิวต์ X และ Y ตามลำดับ

ความหมายคือ เมื่อรู้ค่า X แล้วจะรู้ค่า Y หรือค่า Y ขึ้นกับค่า X โดย X และ Y อาจเป็นแอตทริบิวต์หรือเซตของแอตทริบิวต์ก็ได้

เราสามารถอธิบายฟังก์ชันนัลดีเพนเดนซีในเชิงคณิตศาสตร์ได้ดังนี้

FD: X → Y สำหรับทุกอินสแทนซ์ (instance) r ในรีเลชัน R เมื่อ

$t_1 \in r, t_2 \in r, \pi_x(t_1) = \pi_x(t_2)$ implies $\pi_y(t_1) = \pi_y(t_2)$

คือ หากทูเพิล t_1 และ t_2 เป็นสมาชิกของ r และ ค่า x ใน t_1 เท่ากับ x ใน t_2 แล้ว ค่า y ใน t_1 ย่อมเท่ากับ y ใน t_2 ด้วย โดย x และ y เป็นเซตของแอตทริบิวต์

จากตัวอย่างอินสแทนซ์ที่เป็นไปได้ เราสามารถตรวจสอบได้ว่า อินสแทนซ์นั้นสอดคล้องกับเซตของ FD หรือไม่ แต่ไม่สามารถสรุปได้ว่า FD นั้นทรง (hold) สำหรับ r นั้น ๆ

เมื่อ K เป็นคีย์คู่แข่ง (candidate key) ของ R เราบอกได้ว่า K → R แต่ไม่สามารถสรุปได้ว่า K เป็นคีย์คู่แข่งที่เล็กที่สุด (minimal)

$A_1, A_2, ..., A_n \rightarrow B$

อ่านว่า $A_1, A_2, ..., A_n$ กำหนด B

ถ้า

$A_1, A_2, ..., A_n \rightarrow B_1$
$A_1, A_2, ..., A_n \rightarrow B_2$
...
$A_1, A_2, ..., A_n \rightarrow B_m$

ดังนั้น $A_1, A_2, ..., A_n \rightarrow B_1, B_2 ... B_m$

6.3 การหาฟังก์ชัลนัลดีเพนเดนซีจากข้อมูล

จากข้อมูลในรีเลชัน Employee เราอาจกำหนด F ด้วยการสังเกตความขึ้นต่อกันของข้อมูล ตัวอย่างและรูปในส่วนนี้นำมาจาก [2]

ตัวอย่าง

Employee

emp_no	name	dept_no	dept_name	skills
1	Chonlathorn Kwankajornkiet	201	R&D	C
1	Chonlathorn Kwankajornkiet	201	R&D	Perl
1	Chonlathorn Kwankajornkiet	201	R&D	Java
2	Kailerk Treetipsounthorn	224	IT	Linux
2	Kailerk Treetipsounthorn	224	IT	Mac
3	Kawin Metsiritrakul	201	R&D	DB2
3	Kawin Metsiritrakul	201	R&D	Oracle
3	Kawin Metsiritrakul	201	R&D	Java

รูปที่ 63 รีเลชัน Employee

จากรีเลชัน Employee ในรูปที่ 63 จะเห็นได้ว่า emp_no เดียวกันจะได้ name, dept_no, และ dept_name เดียวกันเสมอ แต่ไม่ได้ skills เดียวกัน จึงเขียน F ได้ว่า

F = {emp_no → name, dept_no, dept_name}

ตัวอย่าง

ให้ Hourly_Emps=(SSN, Name, Level, Rate, WageRate, Hours)

เขียนย่อว่า Hourly_Emps=(SNLRWH)

S	N	L	R	W	H
123-22-3666	Chonlathorn	48	8	10	40
231-31-5368	Kailerk	22	8	10	30
131-24-3650	Kawin	35	5	7	30
434-26-3751	Sanchai	35	5	7	32
612-67-4134	Thanyanuch	35	8	10	40

รูปที่ 64 Hourly_Emps=(SNLRWH)

จากข้อมูลในรูปที่ 64 จะเห็นได้ว่า S ไม่ซ้ำ จึงอาจอนุมานว่า S เป็นคีย์ นอกจากนั้น จะเห็นว่า ค่า R เดียวกันจะได้ W เดียวกันเสมอ จึงเขียน F ได้ดังนี้

F = {S→SNLRWH, R→W}

อย่างไรก็ดี จาก 2 ตัวอย่างข้างบนจะพบว่าข้อมูลมีความซ้ำซ้อนเกิดขึ้น ดังนั้นเราจึงควรย่อยรีเลชันเพื่อลดความซ้ำซ้อนนั้น จากตัวอย่างนี้ เราสามารถย่อยรีเลชันได้ดังแสดงในรูปที่ 65

S	N	L	R	H
123-22-3666	Chonlathorn	48	8	40
231-31-5368	Kailerk	22	8	30
131-24-3650	Kawin	35	5	30
434-26-3751	Sanchai	35	5	32
612-67-4134	Thanyanuch	35	8	40

R	W
8	10
5	7

รูปที่ 65 ย่อยรีเลชัน Hourly_Emps

6.4 การหาฟังก์ชันนัลดีเพนเดนซีทั้งหมด

เราจำเป็นที่จะต้องทราบฟังก์ชันนัลดีเพนเดนซีทั้งหมดของรีเลชันเพื่อให้เข้าใจความขึ้นต่อกันของข้อมูล แล้วนำไปออกแบบฐานข้อมูลให้ละเอียดและสมบูรณ์ที่สุด การที่จะทราบฟังก์ชันนัลดีเพนเดนซี F ทั้งหมดทำได้โดยการอนุมานจากกฎที่เรียกว่า สัจพจน์ของอาร์มสตรอง (Armstrong's axioms) [44] แล้วจะได้ F^+ (F-closure) ซึ่งคือ F ทั้งหมดที่เป็นไปได้ นอกจากนั้น เรายังสามารถใช้สัจพจน์ของอาร์มสตรองเพื่อหา X^+ (X-closure) ซึ่งหมายถึงแอตทริบิวต์ที่อนุมานได้จากแอตทริบิวต์ตั้งต้นโดยคำนึงถึงดีเพนเดนซีของรีเลชันนั้น ๆ ได้อีกด้วย

6.4.1 เอฟโคลเซอร์

FD หนึ่ง ๆ แทนด้วย f

เซตของ FD แทนด้วย F

การที่ f หนึ่ง ๆ จะทรง (hold) ได้นั้น ก็ต่อเมื่อ F ทั้งหมดทรงด้วย เราจึงจำเป็นต้องหา F ทั้งหมดให้ได้ วิธีการหา F ทั้งหมดนี้ทำได้ด้วยการอนุมาน แล้วเรียกเซตของ F ทั้งหมดนี้ว่า F^+ อ่านว่า เอฟโคลเซอร์ (F-closure)

เอฟโคลเซอร์ให้แทนด้วย F^+ คือ F ทั้งหมดที่ได้มาจากการอนุมาน เมื่อเราทราบ F เราสามารถอนุมาน F ทั้งหมดได้จากกฎ 3 กฎ เรียกว่า**สัจพจน์ของอาร์มสตรอง** อันได้แก่

1. กฎการสะท้อน (reflexivity) ถ้า $X \subseteq Y$ แล้ว $Y \rightarrow X$
2. กฎการแต่งเติม (augmentation) ถ้า $X \rightarrow Y$ แล้ว $XZ \rightarrow YZ$ สำหรับ Z ใด ๆ
3. กฎการถ่ายทอด (transitivity) ถ้า $X \rightarrow Y$ และ $Y \rightarrow Z$ แล้ว $X \rightarrow Z$

ทั้ง 3 กฎที่กล่าวมาเรียกได้ว่าเป็นกฎการอนุมานที่สมบูรณ์ในตัวแล้ว กฎอื่นใดที่ใช้ในการอนุมานล้วนมีรากฐานจาก 3 กฎนี้ทั้งสิ้น เช่น

- กฎการยูเนียน (union) ถ้า $X \rightarrow Y$ และ $X \rightarrow Z$ แล้ว $X \rightarrow YZ$
- กฎการย่อย (decomposition) ถ้า $X \rightarrow YZ$ แล้ว $X \rightarrow Y$ และ $X \rightarrow Z$

ตัวอย่าง

ให้ R(A,B,C)

F = {A→B, B→C}

เราสามารถอนุมาน F อื่น ๆ จากสัจพจน์ของอาร์มสตรองได้มากมาย ขอยกตัวอย่างผลจากการอนุมาน ดังนี้

A→B (จาก *f* ที่ให้มา)

B→C (จาก *f* ที่ให้มา)

A→A (Reflexive)

B→B (Reflexive)

C→C (Reflexive)

A→C (Transitive)

AC→BC (Augmentation)

BA→CA (Augmentation)

ทั้งนี้ จำนวนของ *f* ใน F^+ เป็นฟังก์ชันเอกซ์โพเนนเชียลของจำนวนแอตทริบิวต์ ซึ่งจะมีขนาดใหญ่มาก ทำให้การคำนวณหา F^+ เป็นเรื่องที่สิ้นเปลืองหากจะต้องหาทุก *f*

ตัวอย่าง

ให้ Contracts(cid, sid, jid, did, pid, qty, value)

ขอย่อว่า Contracts(CSJDPQV)

และ C เป็นคีย์ ดังนั้น

C → CSJDPQV

ถ้าเราทราบว่าเมื่อแต่ละ Project จะซื้อ Part จะต้องสร้างสัญญาขึ้นมา เราจะได้ f คือ JP → C

ถ้าเราระบุว่าแต่ละ Department สามารถซื้อ Part ได้อย่างมาก 1 จาก 1 Supplier จะได้ f คือ SD → P

ดังนั้น F = {C → CSJDPQV, JP → C, SD → P}

เมื่ออนุมานด้วยสัจพจน์ของอาร์มสตรองจะได้ว่า

JP → C, C → CSJDPQV imply JP → CSJDPQV

SD → P implies SDJ → JP

SDJ → JP, JP → CSJDPQV imply SDJ → CSJDPQV

6.4.2 เอกซ์โคลเชอร์

เราจะแทนแอตทริบิวต์แต่ละตัวด้วย X แล้วคำนวณ เอกซ์โคลเชอร์ (X-closure หรือ X^+) ด้วยแนวคิดเดียวกันกับ F^+ นั่นคือการอนุมาน X ทั้งหมดด้วยสัจพจน์ของอาร์มสตรอง

ประโยชน์ของการคำนวณ F^+ และ X^+ ในที่นี้ ก็เพื่อหาเซตของแอตทริบิวต์ X ที่เป็นคีย์ที่เหมาะสมของรีเลชันนั้น ๆ

ตัวอย่าง

R(A, B, C, D, E)
F = {A→D, D→B, B→C, E→B}

A^+ = {A (reflexive), D, B (transitive), C (transitive)}

B^+ = {B, C}

C^+ = {C}

D^+ = {D, B, C}

E^+ = {E, B, C}

6.5 **การหาคีย์จาก** F^+ **และ** X^+

เหตุผลที่จะคำนวณ F^+ และ X^+ ก็เพื่อจะหาคีย์ที่เหมาะสมกับรีเลชันนั้น นั่นหมายความว่า คีย์ที่ได้มานั้น จะกำหนดแอตทริบิวต์ทั้งหมดของรีเลชันได้ ซึ่งเราจะสามารถทราบได้ว่าคีย์นั้นจะกำหนดได้ทุกแอตทริบิวต์โดยการคำนวณด้วย F นั่นเอง

ดังนั้น ถ้าให้ X เป็นคีย์ของรีเลชัน R จะเขียนได้ว่า $X \rightarrow R$

นอกเหนือจากนั้น การที่จะเป็นคีย์ได้ เซตของแอตทริบิวต์ X ต้องเล็กที่สุดด้วย

ขั้นตอนการหาคีย์และคู่แข่งทั้งหมด ให้เริ่มจาก X ที่อยู่ซ้ายของลูกศร เพราะแอตทริบิวต์ใด ๆ ที่อยู่ข้างซ้ายย่อมเป็นตัวกำหนดแอตทริบิวต์อื่นได้ ในขณะที่แอตทริบิวต์ที่อยู่ขวาของลูกศรย่อมเป็นแอตทริบิวต์ที่ขึ้นต่อตัวอื่น

ตัวอย่าง

ให้ R(A, B, C)

$F = \{A \rightarrow B, B \rightarrow C\}$

จะเห็นว่า A และ B อาจเป็นคีย์ได้ เพราะอยู่ข้างซ้ายของลูกศร แต่ C ไม่มีโอกาสเป็นคีย์

จากนั้นให้คำนวน A^+ และ B^+ จาก F

A^+ = {A, B, C}

ดังนั้น A ต้องเป็นคีย์คู่แข่งแน่ ๆ เพราะสามารถกำหนดแอตทริบิวต์ทุกตัวในรีเลชันได้ และถ้าไม่มีคีย์คู่แข่งอื่น ก็จะสรุปได้ว่า A เป็นคีย์

ต่อไปให้คำนวณ B^+

B^+ = {B, C}

เห็นได้ว่า B ไม่ใช่คีย์คู่แข่ง เพราะไม่สามารถกำหนด A ได้

สรุปได้ว่า รีเลชันนี้ มีคีย์คือ A

6.6 **การนอร์มาไลซ์**

การนอร์มาไลซ์ (normalization) [3, 13, 43] คือการย่อยรีเลชันเพื่อลดปัญหาความซ้ำซ้อนและลดโอกาสที่จะเกิดความผิดพลาดเมื่อมีการปรับแก้ การย่อยนี้จะย่อยเพื่อให้อยู่ในรูปแบบต่าง ๆ ที่มีความเข้มงวดมากขึ้นตามลำดับ

รูปแบบที่ว่านี้เรียกว่า นอร์มัลฟอร์ม (normal form) ซึ่งตำรานี้จะครอบคลุม 4 นอร์มัลฟอร์ม ได้แก่ นอร์มัลฟอร์มที่ 1, นอร์มัลฟอร์มที่ 2, นอร์มัลฟอร์มที่ 3, และนอร์มัลฟอร์มบีซี โดยการย่อยเพื่อให้ได้นอร์มัลฟอร์มทั้งสี่นี้จะคำนึงถึง 2 ปัจจัยคือ 1) ฟังก์ชันนัลดีเพนเดนซี และ 2) คีย์

ในการย่อยรีเลชันนี้ เราจะต้องคำนึงถึงเป้าหมายสำคัญ 2 เป้าหมายคือ

1. ไม่มีปัญหาเมื่อเชื่อมรีเลชันคืน (lossless-join decomposition) กล่าวคือ เมื่อใดที่ย่อยรีเลชัน เราจะต้องมั่นใจว่า เมื่อเชื่อมรีเลชันย่อยเหล่านั้นกลับคืนแล้วจะได้ข้อมูลชุดเดิม ข้อมูลไม่หายและข้อมูลไม่เกิน (non-additive) หรือไม่มีข้อมูลแปลกปลอม (spurious) หรืออีกแง่ คือมั่นใจว่าเรายังสามารถค้นคืนข้อมูลใด ๆ ที่มีอยู่ในรีเลชันก่อนถูกย่อยจากตารางที่ย่อยแล้วได้เสมอ ปัญหานี้มักไม่เกิดหากการย่อยรีเลชันนั้นได้ยึดคีย์เป็นหลักสำคัญในการย่อย
2. ฟังก์ชันนัลดีเพนเดนซียังคงสภาพ (dependency-preserving decomposition) กล่าวคือ เมื่อย่อยรีเลชันแล้วยังคงเก็บฟังก์ชันนัลดีเพนเดนซีเดิมไว้ได้ทั้งหมด

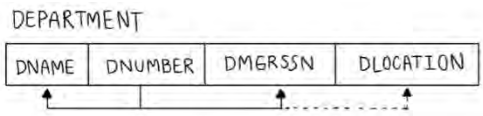
6.7 ขั้นตอนการนอร์มาไลซ์

ตามที่ได้กล่าวไปแล้วว่าการนอร์มาไลซ์คือกระบวนการย่อยรีเลชันให้อยู่ในรูปแบบต่าง ๆ ทั้ง 4 แบบ กระบวนการนอร์มาไลซ์มีรายละเอียดดังนี้

6.7.1 นอร์มัลฟอร์มที่ 1

นอร์มัลฟอร์มที่ 1 (First Normal Form) มักเขียนย่อว่า 1NF ไม่อนุญาตให้รีเลชันมีแอตทริบิวต์ร่วม หรือแอตทริบิวต์หลายค่า หรือความสัมพันธ์ซ้อนใน (nested) และแอตทริบิวต์ทั้งหมดต้องเป็นอะตอมมิก แยกต่อไปไม่ได้

รูปที่ 66 แสดงตัวอย่างแอตทริบิวต์หลายค่า คือ DLOCATIONS วิธีการแก้ไขมี 3 วิธีคือ



DEPARTMENT

DNAME	DNUMBER	DMGRSSN	DLOCATION
Research	5	333445555	{Bangkok,Phuket,Chiang Mai}
Administration	4	967654321	{Pathum Thani}
Headquaters	1	888665555	{Bangkok}

DEPARTMENT

DNAME	DNUMBER	DMGRSSN	DLOCATION
Research	5	333445555	Bangkok
Research	5	333445555	Phuket
Research	5	333445555	Chiang Mai
Administration	4	967654321	Pathum Thani
Headquaters	1	888665555	Bangkok

รูปที่ 66 รีเลชันที่มีแอตทริบิวต์หลายค่า

1. ให้สร้างรีเลชันใหม่ แล้วนำแอตทริบิวต์หลายค่า DLOCATION ไปใส่ไว้ พร้อมกับคีย์ของรีเลชันนั้น จะได้ผลดังแสดงใน

DEPARTMENT

DNAME	DNUMBER	DMGRSSN
Research	5	333445555
Administration	4	987654321
Headquarters	1	888665555

DEPT_LOCATION

DNUMBER	DLOCATION
1	Bangkok
4	Phuket
5	Chiang Mai
5	Pathum Thani
5	Bangkok

รูปที่ 67 DEPT_LOCATION 1NF

2. ขยายคีย์ให้ครอบคลุมถึงแอตทริบิวต์หลายค่านั้น ผลคือ

   DEPARTMENT (DNAME, DNUMBER, DMGRSSN, DLOCATION)

DEPARTMENT

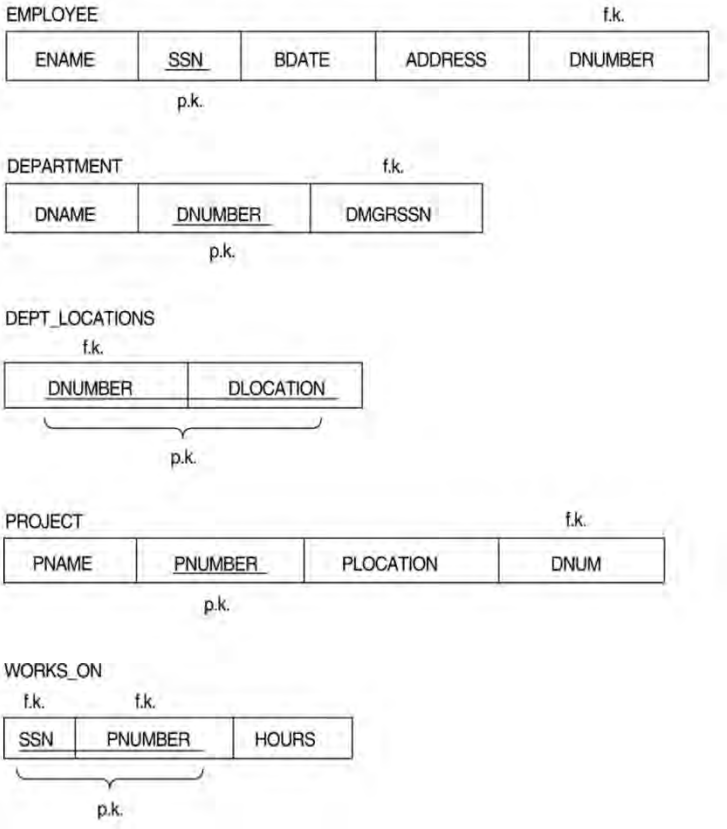
DNAME	DNUMBER	DMGRSSN	DLOCATION
Research	5	333445555	Bangkok
Research	5	333445555	Phuket
Research	5	333445555	Chiang Mai
Administration	4	967654321	Pathum Thani
Headquaters	1	888665555	Bangkok

รูปที่ 68 รีเลชันในรูปแบบ 1NF ด้วยวิธีขยายคีย์

3. หากทราบจำนวนที่เป็นไปได้ของแอตทริบิวต์หลายค่า ให้เพิ่มแอตทริบิวต์ตามจำนวนนั้น เช่น หากทราบว่ามี LOCATION 3 แห่ง ผลคือ

   DEPARTMENT (DNAME, DNUMBER, DMGRSSN, DLOC1, DLOC2, DLOC3)

เมื่อปรับรีเลชันให้อยู่ในรูปแบบ 1NF แล้วจะมีสคีมาคร่าว ๆ ดังแสดงในรูปที่ 69 ส่วนรูปที่ 70 แสดงสถานะหนึ่งของข้อมูลในฐานข้อมูลบริษัทดีบี



รูปที่ 69 สคีมาของบริษัทดีบี

EMPLOYEE

ENAME	SSN	BDATE	ADDRESS	DNUMBER
Smith,John B.	123456789	1965-01-09	731 Fondren,Houston,TX	5
Wong,Franklin T.	333445555	1955-12-08	638 Voss,Houston,TX	5
Zelaya,Alicia J.	999887777	1968-07-19	3321 Castle,Spring,TX	4
Wallace,Jennifer S.	987654321	1941-06-20	291 Berry,Bellaire,TX	4
Narayan,Ramesh K.	666884444	1962-09-15	975 Fire Oak,Humble,TX	5
English,Joyce A.	453453453	1972-07-31	5631 Rice,Houston,TX	5
Jabbar,Ahmad V.	987987987	1969-03-29	980 Dallas,Houston,TX	4
Borg,James E.	888665555	1937-11-10	450 Stone,Houston,TX	1

DEPARTMENT

DNAME	DNUMBER	DMGRSSN
Research	5	333445555
Administration	4	987654321
Headquarters	1	888665555

DEPT_LOCATIONS

DNUMBER	DLOCATION
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

SSN	PNUMBER	HOURS
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	null

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

รูปที่ 70 สถานะของข้อมูลในฐานข้อมูลบริษัทดีบี

6.7.2 นอร์มัลฟอร์มที่ 2

รีเลชันที่ผ่าน 1NF แต่ไม่ผ่านนอร์มัลฟอร์มที่ 2 (2NF) จะมีปัญหาเรื่องความซ้ำซ้อนเกิดขึ้น เช่น เมื่อเชื่อมรีเลชัน EMPLOYEE กับ PROJECT ได้เป็น EMP_PROJ จะเกิดความซ้ำซ้อนของข้อมูลในคอลัมน์ 2 ชุด ดังแสดงในรูปที่ 71 คือ คือข้อมูลในคอลัมน์ HOURS และ ENAME ซ้ำกันอยู่หลายทูเพิล และข้อมูล PNAME PLOCATION ก็ซ้ำกันหลายทูเพิล

ซ้ำซ้อน ซ้ำซ้อน

EMP_PROJ

SSN	PNUMBER	HOURS	ENAME	PNAME	PLOCATION
123456789	1	32.5	Smith,John B.	ProductX	Bellaire
123456789	2	7.5	Smith,John B.	ProductY	Sugarland
666884444	3	40.0	Narayan,Ramesh K.	ProductZ	Houston
453453453	1	20.0	English,Joyce A.	ProductX	Bellaire
453453453	2	20.0	English,Joyce A.	ProductY	Sugarland
333445555	2	10.0	Wong,Franklin T.	ProductY	Sugarland
333445555	3	10.0	Wong,Franklin T.	ProductZ	Houston
333445555	10	10.0	Wong,Franklin T.	Computerization	Stafford
333445555	20	10.0	Wong,Franklin T.	Reorganization	Houston
999887777	30	30.0	Zelaya,Alicia J.	Newbenefits	Stafford
999887777	10	10.0	Zelaya,Alicia J.	Computerization	Stafford
987987987	10	35.0	Jabbar,Ahmad V.	Computerization	Stafford
987987987	30	5.0	Jabbar,Ahmad V.	Newbenefits	Stafford
987654321	30	20.0	Wallace,Jennifer S.	Newbenefits	Stafford
987654321	20	15.0	Wallace,Jennifer S.	Reorganization	Houston
888665555	20	null	Borg,James E.	Reorganization	Houston

รูปที่ 71 รีเลชัน EMP_PROJ

ดังนั้น เราจะย่อยรีเลชันโดยคำนึงถึงคีย์ทั้งหมดของรีเลชัน เพราะถ้าทุกแอตทริบิวต์ขึ้นต่อคีย์แล้วจะทำให้ตารางลดความซ้ำซ้อนแน่นอน

นอร์มัลฟอร์มที่ 2 (Second Normal Form) หรือ 2NF ระบุว่า ต้องผ่านนอร์มัลฟอร์มที่ 1 และทุกแอตทริบิวต์ที่ไม่ใช่ไพรม์แอตทริบิวต์จะต้องขึ้นต่อทุกคีย์ของรีเลชันแบบทั้งหมด

ไพรม์แอตทริบิวต์ (prime attribute) คือ แอตทริบิวต์ที่เป็นส่วนหนึ่งของคีย์ รวมถึงคีย์คู่แข่ง (candidate key) ด้วย

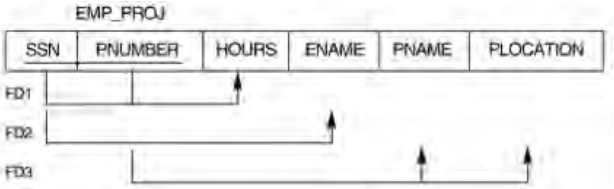
การขึ้นต่อกันแบบทั้งหมด (full functional dependency) หมายถึง เมื่อให้ FD Y → Z แล้ว จะไม่สามารถลดแอตทริบิวต์จากเซตของแอตทริบิวต์ Y แล้ว FD นั้นยังคงถูกต้องหรือทรง (hold) อยู่ เช่น

FD = ({SSN, PNUMBER} → HOURS)

เป็นการขึ้นต่อกันแบบทั้งหมด เนื่องจากไม่สามารถลด SSN หรือ PNUMBER ออกได้ หมายความว่าหากใช้เพียงตัวใดตัวหนึ่งเป็นตัวกำหนด รีเลชันนี้ก็จะทรงไม่ได้ เพราะตัวใดตัวหนึ่งไม่สามารถระบุค่า HOURS ได้

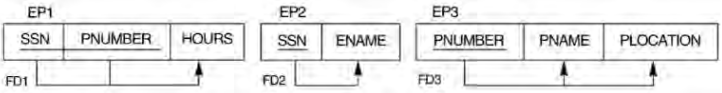
แต่ถ้า FD = ({SSN, PNUMBER} → ENAME) นี้จะไม่ใช่แบบขึ้นต่อกันทั้งหมด เพราะ ENAME ขึ้นต่อ SSN เท่านั้น เราจึงสามารถลด PNUMBER ออก แล้ว FD ยังคงถูกต้องได้

ตัวอย่าง 2NF แสดงในรูปที่ 72 จะเห็นว่า จาก FD1 นั้น HOURS ขึ้นต่อ {SSN, PNUMBER} จึงขึ้นต่อคีย์หลักทั้งหมด เพราะคีย์หลักคือ {SSN, PNUMBER} แต่ FD2 ไม่ผ่าน 2NF เพราะ ENAME ขึ้นต่อเพียง SSN ซึ่งเป็นเพียงส่วนหนึ่งของคีย์ (partial key) และ FD3 ก็ไม่ผ่าน 2NF ด้วยเหตุผลเดียวกัน คือ PNAME, PLOCATION ขึ้นต่อ PNUMBER ซึ่งเป็นส่วนหนึ่งของคีย์หลักเช่นกัน



รูปที่ 72 FD ของ EMP_PROJ

การแก้ไขคือให้ย่อยเป็น 3 รีเลชัน โดยแยก partial key ออกไปทั้ง FD2 และ FD3 ดังแสดงในรูปที่ 73



รูปที่ 73 EMP_PROJ ย่อยให้อยู่ในรูปแบบ 2NF

6.7.3 นอร์มัลฟอร์มที่ 3

พิจารณารูปที่ 74 จะเห็นรีเลชัน EMP_DEPT ที่เกิดจากการเชื่อมรีเลชัน EMPLOYEE กับ DEPARTMENT ซึ่งจะเกิดความซ้ำซ้อนของข้อมูล คือข้อมูลในคอลัมน์ DNAME และ DMGRSSN ซ้ำกันหลายทูเพิล

รีเลชัน EMP_DEPT ผ่าน 2NF เพราะมีคีย์เดี่ยว คือ SSN และไม่มีคีย์คู่แข่ง นั่นหมายความว่าแอตทริบิวต์ที่ไม่ใช่ไพรม์ย่อมขึ้นต่อ SSN ทั้งหมด อย่างไรก็ดี จะเห็นว่ายังมีความซ้ำซ้อนที่ไม่จำเป็น เราจึงควรย่อยตารางนี้ต่อเพื่อลดความซ้ำซ้อนลักษณะนี้

ซ้ำซ้อน

EMP_DEPT

ENAME	SSN	BDATE	ADDRESS	DNUMBER	DNAME	DMGRSSN
Smith,John B.	123456789	1965-01-09	731 Fondren,Houston,TX	5	Research	333445555
Wong,Frankin T.	333445555	1955-12-08	638 Voss,Houston,TX	5	Research	333445555
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle,Spring,TX	4	Administration	987654321
Wallace,Jennifer S.	987654321	1941-06-20	291 Berry,Bellaire,TX	4	Administration	987654321
Narayan,Ramesh K.	666884444	1962-09-15	975 FireOak,Humble,TX	5	Research	333445555
English,Joyce A.	453453453	1972-07-31	5631 Rice,Houston,TX	5	Research	333445555
Jabbar,Ahmad V.	987987987	1969-03-29	980 Dallas,Houston,TX	4	Administration	987654321
Borg,James E.	888665555	1937-11-10	450 Stone,Houston,TX	1	Headquarters	888665555

รูปที่ 74 รีเลชัน EMP_DEPT

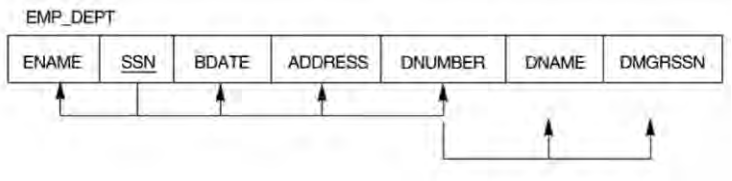
นอร์มัลฟอร์มที่ 3 (Third Normal Form) หรือ 3NF ระบุว่า รีเลชันใด ๆ จะอยู่ใน 3NF ได้นั้น จะต้องผ่าน 2NF และต้องไม่ขึ้นต่อทุกคีย์แบบส่งผ่าน (transitive dependency) ซึ่งรวมถึงคีย์คู่แข่งด้วย

ความขึ้นต่อกันแบบส่งผ่าน (Transitive dependency) จะเกิดขึ้นเมื่อ FD $X \rightarrow Z$ มาจาก $\{X \rightarrow Y$ และ $Y \rightarrow Z\}$

หรืออธิบาย 3NF ได้อีกแบบว่า 3NF กำหนดให้รีเลชันเป็น 3NF เมื่อ FD $X \rightarrow A$ ทรงในรีเลชัน R แล้ว

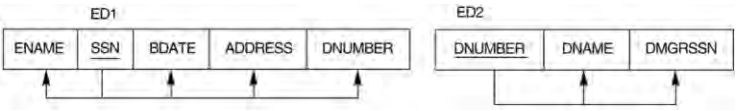
1. X เป็นซูเปอร์คีย์ของ R หรือ
2. A เป็นไพรม์แอตทริบิวต์

จากตัวอย่างจะเห็นว่า DNAME และ DMGRSSN ขึ้นต่อ DNUMBER และ DNUMBER ขึ้นต่อ SSN อีกที หรือมองกลับกันจะเห็นว่า SSN กำหนด DNUMBER แล้ว DNUMBER กำหนด DNAME และ DMGRSSN ดังแสดงในรูปที่ 75 ซึ่งทำให้ไม่ผ่าน 3NF



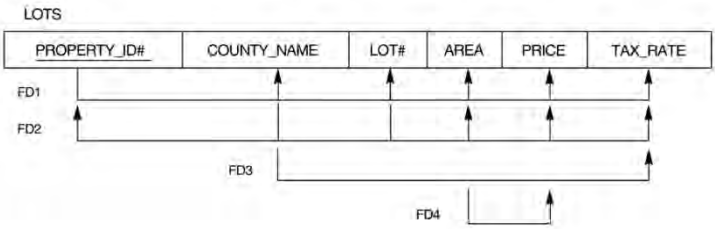
รูปที่ 75 FD ของ EMP_DEPT

เมื่อเกิดเหตุการณ์เช่นนี้ ให้ย่อยตารางดังตัวอย่างในรูปที่ 76



รูปที่ 76 EMP_PROJ ย่อยให้อยู่ในรูปแบบ 3NF

ตัวอย่างการแปลงจาก 1NF ไป 2NF และ 3NF



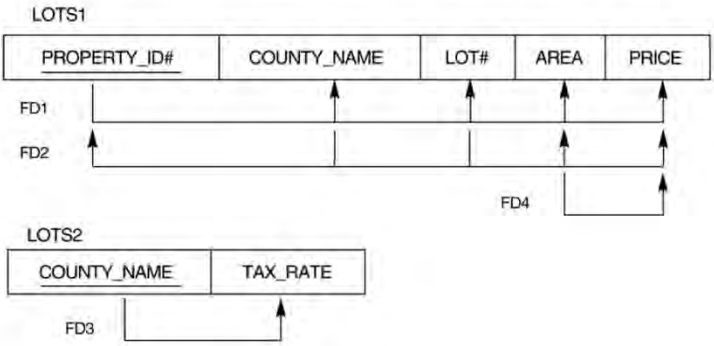
รูปที่ 77 รีเลชัน LOTS กับ 4 FD

จากรูปที่ 77 พบว่า FD ที่ส่งผลให้รีเลชันตั้งต้นที่อยู่ในรูปแบบ 1NF แต่ไม่ผ่าน 2NF คือ FD3 เพราะเราสามารถตีความได้ว่ารีเลชันนี้มีคีย์คู่แข่ง 2 ชุด ได้แก่

1. PROPERTY_ID# จากขีดเส้นใต้ และที่ FD1 ระบุว่า PROPERTY_ID# สามารถกำหนดแอตทริบิวต์ในรีเลชันได้ทุกตัว
2. {COUNTY_NAME, LOT#} แม้จะไม่ได้ขีดเส้นใต้ระบุว่าเป็นคีย์หลัก แต่จาก FD2 ที่ระบุว่า {COUNTY_NAME, LOT#} สามารถ

กำหนดแอตทริบิวต์ในรีเลชันได้ทุกตัวนั้น ชี้ให้เห็นว่า {COUNTY_NAME, LOT#} เป็นคีย์คู่แข่ง

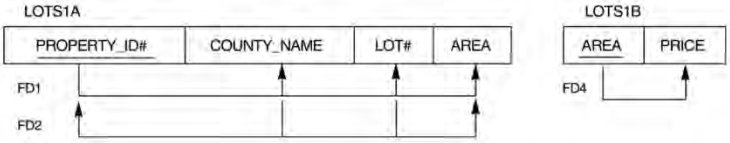
การแก้ไขกรณีนี้ ให้ย่อยรีเลชัน โดยแยก FD3 ออกมาเป็น LOTS2 (COUNTY_NAME, TAX_RATE) ดังแสดงใน รูปที่ 78



รูปที่ 78 รีเลชัน LOTS1 และ LOTS2 ในรูปแบบ 2NF

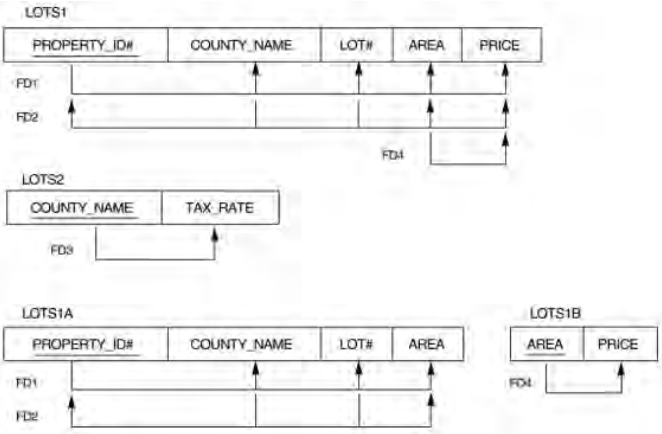
ส่วน FD ที่ส่งผลให้รีเลชันที่ย่อยแล้วไม่ผ่าน 3NF คือ FD4 เพราะ PRICE ขึ้นต่อ AREA แล้ว AREA ขึ้นต่อคีย์ คือ PROPERTY_ID# อีกต่อหนึ่ง จึงเกิดการขึ้นต่อกันแบบส่งผ่าน

การแก้ไขในกรณีนี้ ให้ย่อยรีเลชัน โดยแยก FD4 ออกมาอีกรีเลชัน คือ LOTS1B(AREA, PRICE) ดังแสดงในรูปที่ 79



รูปที่ 79 ย่อยรีเลชัน LOTS1 เป็น LOTS1A และ LOTS1B ในรูปแบบ 3NF

สรุปรีเลชันทั้งหมดที่อยู่ในรูปแบบ 3NF แสดงในรูปที่ 80

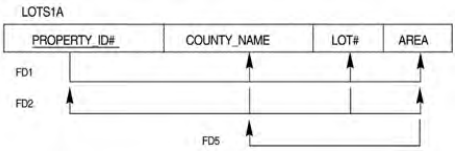


รูปที่ 80 รีเลชัน LOTS ย่อยให้อยู่ในรูปแบบ 3NF

6.7.4 นอร์มัลฟอร์มบอยซ์คอดด์

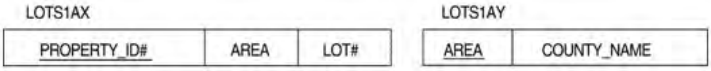
นอร์มัลฟอร์มบอยซ์คอดด์ (Boyce-Codd Normal Form) นิยมเรียกแบบย่อว่า บีซีเอ็นเอฟ (BCNF) กำหนดว่ารีเลชันต้องอยู่ใน 3NF และเมื่อ FD $X \rightarrow A$ ทรงในรีเลชัน R แล้ว X ต้องเป็นซูเปอร์คีย์ของ R เท่านั้น

เมื่อพิจารณารูปที่ 81 จะพบว่า FD5 ส่งผลให้ไม่ผ่านบีซีเอ็นเอฟ เพราะ AREA → COUNTY_NAME แต่ AREA ไม่ใช่ซูเปอร์คีย์ของรีเลชันนี้



รูปที่ 81 รีเลชัน LOTS1A ที่ไม่ผ่านบีซีเอ็นเอฟ

วิธีการแก้คือให้ย่อยรีเลชันโดยยึดแอตทริบิวต์ที่เป็นไพรม์แต่ไม่ใช่ซูเปอร์คีย์นั้นเป็นหลัก ให้มีอยู่ในรีเลชันย่อยทั้งสอง ดังแสดงในรูปที่ 82



รูปที่ 82 รีเลชัน LOTS1AX และ LOTS1AY ในรูปแบบ BCNF

การแยกย่อยรีเลชันให้เป็นไปตามที่บีซีเอ็นเอฟกำหนดนั้น จำเป็นต้องยึดแอตทริบิวต์ที่เป็นไพรม์แต่ไม่ใช่ซูเปอร์คีย์เป็นหลัก นั่นเป็นเพราะการแยกเพียงแบบนี้เท่านั้นที่จะทำให้การเชื่อมรีเลชันคืนแล้วจะไม่เกิดปัญหาข้อมูลเกินหรือขาด อย่างไรก็ดี เพื่อย่อยรีเลชันในกรณีนี้แล้ว จะทำให้ไม่สามารถคงสภาพ FD ไว้ได้

ตัวอย่างรีเลชันในรูปที่ 83 กำหนด

FD1: {student, course} → instructor

FD2: instructor → course

เป็นรีเลชันที่ไม่ผ่านบีซีเอ็นเอฟ เพราะจาก FD1 อาจบอกได้ว่า {student, course} เป็นคีย์ จึงเป็นไพรม์ แต่เมื่อพบว่า instructor ย้อนไปกำหนดไพร์ม จึงทำให้ไม่ผ่านบีซีเอ็นเอฟ

STUDENT	COURSE	INSTRUCTOR
Kailerk	Database	Proadpran
Kawin	Database	Chotirat
Pranee	Operating Systems	Chonlatorn
Sanchai	Theory	Athasith
Somsak	Database	Wiwat
Supachart	Operating Systems	Chonlatorn
Thanyanuch	Database	Tarathip

รูปที่ 83 รีเลชัน TEACH

เราอาจย่อยรีเลชันได้ 3 คู่คือ

- {student, instructor} และ {student, course}
- {course, instructor} และ {course, student}
- {instructor, course} และ {instructor, student}

ให้สังเกตว่า ไม่ว่าจะย่อยแบบคู่ใดก็จะส่งผลให้ FD1 นั้นไม่คงสภาพแล้ว ซึ่งบีซีเอ็นเอฟยอมให้เกิดได้

อย่างไรก็ดี มีเพียงแค่ {instructor, course} และ {instructor, student} เท่านั้น ที่การเชื่อมตารางคืนแล้วจะไม่เกิดปัญหาข้อมูลเกิน

6.8 **บทสรุปและเรื่องชวนคิด**

หัวใจสำคัญของบทนี้คือการนอร์มาไลซ์รีเลชันเพื่อลดความซ้ำซ้อนของข้อมูล ซึ่งจะส่งผลให้ลดโอกาสเกิดความผิดพลาดเมื่อแก้ไขข้อมูลได้ แต่ก็ยังมีอีกหลายปัจจัยที่ต้องคำนึงถึง เช่น ปัญหาค่าว่าง (null value) ว่าควรย่อยตารางโดยพิจารณาให้มีการเก็บค่าว่างอย่างเหมาะสมอีกด้วย เพราะค่าว่างอาจทำให้เพิ่มพื้นที่จัดเก็บและทำให้ลดประสิทธิภาพของฐานข้อมูลโดยไม่จำเป็น หากออกแบบได้ไม่ดี ในประเด็นนี้ ขอให้ผู้เรียนลองคิดต่อว่านอกจากค่าว่างแล้ว จะมีปัจจัยอะไรอีกบ้าง ที่อาจส่งผลให้ประสิทธิภาพของฐานข้อมูลลดลงได้

นอกเหนือจากนั้นเราได้เรียนเรื่องนอร์มัลฟอร์มและขั้นตอนการย่อยรีเลชันให้อยู่ในรูปแบบที่แต่ละนอร์มัลฟอร์มระบุ อย่างไรก็ดี ยังมีนอร์มัลฟอร์มอีกหลายรูปแบบที่ไม่ได้ครอบคลุมในตำรานี้ หากผู้เรียนมีโอกาส อยากให้สังเกตและเรียนรู้ว่าองค์กรต่าง ๆ ได้กำหนดระดับนอร์มัลฟอร์มของฐานข้อมูลของเขาที่ระดับใด และหากสนใจเรื่องนี้ก็อาจศึกษานอร์มัลฟอร์มรูปแบบอื่น ๆ ต่อไปด้วย

แบบฝึกหัดท้ายบท

1. เอฟโคลเซอร์และเอกซ์โคลเซอร์คืออะไร และทำอย่างไรจึงจะหาสองสิ่งนี้ได้
2. เหตุใดเราจึงต้องคำนวณหาเอฟโคลเซอร์และเอกซ์โคลเซอร์
3. สัจพจน์ของอาร์มสตรองมีกฎพื้นฐานกี่ข้อ อะไรบ้าง
4. เพราะเหตุใดการคำนวณหาเอฟโคลเซอร์จึงเป็นเรื่องที่สิ้นเปลือง หากจะต้องหาทุก f
5. ขั้นตอนการหาคีย์และคู่แข่งทั้งหมด ควรเริ่มจากเซตแอตทริบิวต์ใดในฟังก์ชันนัลดีเพนเดนซี เพราเหตุใด
6. ฐานข้อมูลที่ดีควรมีลักษณะอย่างไร
7. การนอร์มาไลซ์คืออะไร และเหตุใดเราจึงควรทำการนอร์มาไลซ์
8. เป้าหมายสำคัญที่ต้องคำนึงถึงเมื่อจะทำการนอร์มาไลซ์รีเลชันมีกี่เป้าหมาย อะไรบ้าง จงอธิบาย
9. ไพรม์แอตทริบิวต์คืออะไร
10. “การนอร์มาไลซ์เป็นนอร์มัลฟอร์มทุกแบบ จะสามารถคงฟังก์ชันนัลดีเพนเดนซีไว้ได้ครบเสมอ” จากข้อความข้างต้น ผู้เรียนเห็นด้วยหรือไม่ จงอภิปราย

บทที่ 7
พีชคณิตเชิงสัมพันธ์และแคลคูลัสเชิงสัมพันธ์

วัตถุประสงค์

1. เพื่อให้เข้าใจการสอบถามฐานข้อมูลเชิงสัมพันธ์ด้วยพีชคณิตเชิงสัมพันธ์
2. เพื่อศึกษาโอเปอเรเตอร์พีชคณิตเชิงสัมพันธ์
3. เพื่อให้เข้าใจความแตกต่างระหว่างพีชคณิตเชิงสัมพันธ์กับแคลคูลัสเชิงสัมพันธ์

การสอบถาม (query) ฐานข้อมูลเชิงสัมพันธ์อาศัยภาษาทางคณิตศาสตร์ (mathematical query language) สองภาษาคือ พีชคณิตเชิงสัมพันธ์และแคลคูลัสเชิงสัมพันธ์ [3, 13, 43]

พีชคณิตเชิงสัมพันธ์มีลักษณะการทำงานแบบปฏิบัติการ (operational) คือ แปลคำสั่งแล้วทำงานทีละคำสั่ง ทำงานทีละขั้นตอน เน้นการสอบถามข้อมูลในรีเลชัน

ส่วนแคลคูลัสเชิงสัมพันธ์นั้นเป็นเชิงพรรณนา (declarative) ผู้เขียนคำสั่งโดยใช้แคลคูลัสเชิงสัมพันธ์มักระบุนิยามของผลลัพธ์ โดยไม่ต้องคำนึงถึงลำดับการทำงานของคำสั่งนั้น

กฎของคอดด์ (Codd's theorem) กล่าวว่า มีนิพจน์พีชคณิตเชิงสัมพันธ์และนิพจน์แคลคูลัสเชิงสัมพันธ์ ที่ให้ผลลัพธ์เดียวกันได้เสมอ (logically equivalent)

7.1 หลักการของพีชคณิตเชิงสัมพันธ์

พีชคณิตเชิงสัมพันธ์มีรากฐานมาจากทฤษฎีเซต (set theory) การดำเนินการใด ๆ กับข้อมูลในฐานข้อมูล หากใช้พีชคณิตเชิงสัมพันธ์ (relational algebra) แล้ว ผลที่ได้จากการทำโอเปอเรชันพีชคณิต (algebra operation) หรือคำสั่งพีชคณิตนั้นจะทำให้เกิดรีเลชันใหม่ อาจเกิดรีเลชันเดียวหรือหลายรีเลชันก็ได้

เซตของโอเปอเรชันพีชคณิตเชิงสัมพันธ์ เรียกว่านิพจน์พีชคณิตเชิงสัมพันธ์ (relational algebra expression) ซึ่งผลลัพธ์ก็คือรีเลชันที่เป็นผลของ

การสอบถาม (query) ชุดนั้น ๆ หรืออาจพูดได้ว่า การสอบถามคือโอเปอเรชันพีชคณิตที่เรียงต่อกันนั่นเอง

7.2 โอเปอเรชันเชิงสัมพันธ์

โอเปอเรชันเชิงสัมพันธ์ (relational operation) มีโอเปอเรเตอร์ (operator) หลายตัว สามารถจัดแบ่งตามการใช้งานได้ดังนี้

โอเปอเรชันเชิงสัมพันธ์จากรีเลชันเดียว (unary relational operations) ตัวโอเปอเรเตอร์ ได้แก่ SELECT, PROJECT และ RENAME

โอเปอเรชันเชิงสัมพันธ์จาก 2 รีเลชัน (binary relational operations) ตัวโอเปอเรเตอร์ ได้แก่ JOIN และ DIVISION

โอเปอเรชันเชิงสัมพันธ์จากหลายรีเลชัน (n-ary relation operations) จะนำเอาตัวโอเปอเรชันหลายตัวมาใช้ร่วมกัน

การเขียนคำสั่งโอเปอเรชันนั้น ให้ระวังการใช้อักษรตัวเล็กและตัวใหญ่ ในโอเปอเรชันพีชคณิตเชิงสัมพันธ์เราจะใช้อักษรตัวเล็ก เช่น

$$r - s$$

หากเมื่อใดใช้อักษรตัวใหญ่ ให้เข้าใจว่าเป็นโอเปอเรชันบนสคีมา เช่น

$$R - S$$

7.2.1 SELECT

คำสั่งหรือโอเปอเรชันที่ใช้เมื่อมีรีเลชันเดียว ได้แก่ โอเปอเรเตอร์ SELECT ซึ่งมีหน้าที่เลือกซับเซตของทูเพิลจากรีเลชันตามเงื่อนไขที่ต้องการ สามารถเขียนได้ในรูปแบบนี้

$$\sigma_p(r)$$

สัญลักษณ์ σ อ่านว่า ซิกมา (sigma) เป็นสัญลักษณ์ที่ใช้แทนคำสั่ง SELECT

p คือเงื่อนไขที่ต้องการ เขียนด้วยสมการแคลคูลัสที่เชื่อมต่อแต่ละเทอมด้วย $\wedge$ (and), $\vee$ (or), $\neg$ (not)

แต่ละเทอมเขียนได้ดังนี้

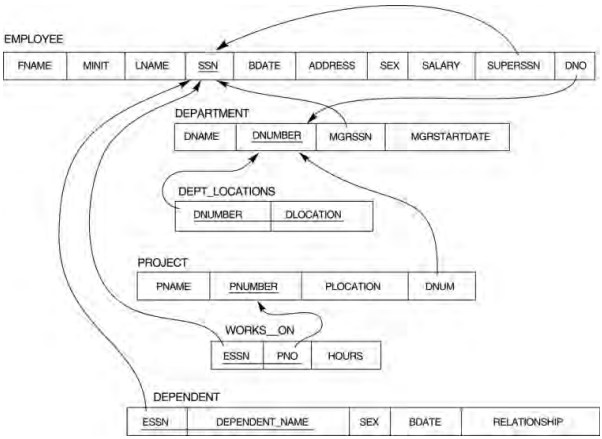
<attribute> *op* <attribute> or <constant>

โดย *op* อาจเป็น =, $\neq$, >, $\geq$, <, $\leq$

r เป็นชื่อของรีเลชันที่ต้องการดำเนินการ จะได้ผลลัพธ์

$$\sigma_p(r) = \{t \mid t \in r \textbf{ and } p(t)\}$$

ตัวอย่าง



รูปที่ 84 รีเลชันบริษัทดีบี

จากรีเลชันบริษัทดีบีในรูปที่ 84 เราต้องการสอบถามหา EMPLOYEE ที่อยู่ DEPARTMENT No 4 และ EMPLOYEE ที่มีเงินเดือนมากกว่า 30,000

$\sigma_{DNO=4}(employee)$

$\sigma_{SALARY>30000}(employee)$

ตัวอย่าง

$$\sigma_{(DNO=4 \wedge SALARY>25000) \vee (DNO=5 \wedge SALARY>30000)}(employee)$$

หมายถึง ต้องการเลือกทูเพิลของรีเลชัน EMPLOYEE เฉพาะรายการที่ DNo = 5 และเงินเดือนมากกว่า 30,000 หรือ DNO = 4 และเงินเดือนมากกว่า 25,000

จะได้ผลลัพธ์ดังแสดงในรูปที่ 85

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
Franklin	T	Wong	333445555	1955-12-08	638 Voss,Houston,TX	M	40000	888665555	5
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry,Bellaire,TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 FireOak,Humble,TX	M	38000	333445555	5

รูปที่ 85 ผลลัพธ์การ SELECT

คุณสมบัติของ SELECT มีดังนี้

โอเปอเรเตอร์ SELECT นั้น ผลที่ได้สคีมาเหมือนกับรีเลชันตั้งต้น คือ

$\sigma_p(r)$ จะได้สคีมาเดียวกันกับ r

โอเปอเรเตอร์ SELECT มีคุณสมบัติการเรียงสับเปลี่ยน (commutative) หมายความว่าการเรียงต่างแบบไม่ส่งผลให้ผลลัพธ์ต่างไป

$$\sigma_{p1}(\sigma_{p2}(r))=\sigma_{p2}(\sigma_{p1}(r))$$

7.2.2 PROJECT

โอเปอเรเตอร์ PROJECT เป็นโอเปอเรชันที่ใช้เมื่อมีรีเลชันเดียว ใช้เลือกข้อมูลเฉพาะคอลัมน์ที่ต้องการจากตารางหรือรีเลชันนั้น

$$\pi_{<\text{attribute list}>}(r)$$

สัญลักษณ์ π อ่านว่า พาย (pi) เป็นสัญลักษณ์ที่ใช้แทนคำสั่ง PROJECT

attribute list คือแอตทริบิวต์หรือคอลัมท์ที่ต้องการเลือก

R เป็นชื่อของรีเลชันที่ต้องการดำเนินการ

ผลลัพธ์จากการ PROJECT คือทูเพิลทั้งหมด หากผลการ PROJECT ทำให้มีทูเพิลซ้ำ ก็จะลดทูเพิลที่ซ้ำกันออก เพราะรีเลชันคือเซตของทูเพิลนั่นเอง ดังนั้นจำนวนทูเพิลที่ได้จะน้อยกว่าหรือเท่ากับรีเลชันตั้งต้นเสมอ

ตัวอย่าง

$$\pi_{\text{LNAME,FNAME,SALARY}}(\text{employee})$$

หมายถึง ต้องการเลือกข้อมูลทุกทูเพิลจากคอลัมน์หรือแอตทริบิวต์ LNAME,FNAME และ SALARY จากรีเลชัน EMPLOYEE

ผลลัพธ์แสดงในรูปที่ 86

LNAME	FNAME	SALARY
Smith	John	30000
Wong	Franklin	40000
Zelaya	Alicia	25000
Wallace	Jennifer	43000
Narayan	Ramesh	38000
English	Joyce	25000
Jabbar	Ahmad	25000
Borg	James	55000

รูปที่ 86 ผลลัพธ์การ PROJECT LNAME, FNAME, SALARY

ตัวอย่าง

$$\pi_{SEX,\ SALARY}(employee)$$

ผลลัพธ์แสดงในรูปที่ 87

SEX	SALARY
M	30000
M	40000
F	25000
F	43000
M	38000
M	25000
M	55000

รูปที่ 87 ผลลัพธ์การ PROJECT SEX, SALARY

7.2.3 RENAME

บางครั้งเราจำเป็นที่จะต้องนำผลลัพธ์จากโอเปอเรชันไปหาดำเนินการต่อ เราอาจสร้างรีเลชันระหว่างทางเพื่อเก็บผลลัพธ์จากโอเปอเรชันหนึ่งไปใช้อ้างอิง

ต่อได้ โดยการให้ชื่อหรือเปลี่ยนชื่อรีเลชันนั้นด้วยโอเปอเรเตอร์ RENAME นอกจากนั้น คำสั่ง RENAME นี้ยังสามารถใช้เปลี่ยนชื่อแอตทริบิวต์ได้ด้วย รูปแบบการเขียนมีดังนี้

$$\rho_{X(oldname1 \rightarrow newname1 \text{ or } position \rightarrow newname1, \ldots)}(e)$$

หรือ

$$\rho_{X(newname1, newname2, \ldots, newnamen)}(e)$$

สัญลักษณ์ ρ อ่านว่า โรห์ (rho) ใช้แสดงคำสั่ง RENAME

e คือ expression ใด ๆ

oldname คือ ชื่อเดิมของแอตทริบิวต์

newname คือ ชื่อใหม่ของแอตทริบิวต์

X คือ ชื่อใหม่ของรีเลชันผลลัพธ์

ในกรณีที่ไม่ได้ระบุ oldname นั้น newname จะแทนชื่อของแอตทริบิวต์ตามตำแหน่งที่เขียนในโอเปอเรชัน RENAME

ตัวอย่าง

$$\rho_{myRelation(A \rightarrow E, 2 \rightarrow K)}(r - s)$$

จากคำสั่งข้างบน จะมีการลบรีเลชัน r ด้วย s แล้วนำผลลัพธ์ไปเก็บไว้ในตารางหรือรีเลชันชื่อ myRelation และยังมีการเปลี่ยนชื่อแอตทริบิวต์ A เป็น E และ เปลี่ยนชื่อแอตทริบิวต์ตำแหน่งที่ 2 เป็น K

อีกวิธีที่ใช้เปลี่ยนชื่อรีเลชัน คือการใช้โอเปอเรเตอร์ ASSIGNMENT แทนด้วยลูกศรชี้ซ้าย $\leftarrow$ เช่น

$$\text{newRelation} \leftarrow \sigma_{DNO=5}(\text{employee})$$

$$\text{result} \leftarrow \pi_{LNAME,FNAME,SALARY}(\text{newRelation})$$

หมายความว่า ให้ SELECT ทูเพิลที่ DNO = 5 จากรีเลชัน EMPLOYEE ไปเก็บไว้ในรีเลชันใหม่ ชื่อว่า NEWRELATION จากนั้นก็อ้างอิงชื่อรีเลชันใหม่นั้นได้เลย

7.2.4 UNION

โอเปอเรชันยูเนียน (UNION) เป็นการดำเนินการในโอเปอเรชันเชิงสัมพันธ์จาก 2 รีเลชัน (binary relational operations) เพื่อเชื่อมตาราง 2 ตาราง ด้วยการนำทูเพิลจาก 2 ตารางมารวมกัน มีรูปแบบการเขียนคือ

$$r \cup s$$

สัญลักษณ์ $\cup$ อ่านว่า ยูเนียน (union)

R และ S คือชื่อรีเลชัน

ผลลัพธ์จากการยูเนียนคือ

$$r \cup s = \{t \mid t \in r \text{ or } t \in s\}$$

ทั้งนี้ การยูเนียนบังคับว่า 2 ตารางที่นำมายูเนียนกันต้องมีไทป์เดียวกัน (type compatibility) นั่นหมายความว่าทั้งสองตารางมีจำนวนแอตทริบิวต์และโดเมน (domain) เดียวกัน และเรียงแอตทริบิวต์เหมือนกันด้วย แต่ไม่จำเป็นต้องมีชื่อแอตทริบิวต์เหมือนกัน ดังนั้นรีเลชันที่เป็นผลลัพธ์ของ r1 ∪ r2 นั้นจะมีชื่อแอตทริบิวต์ตามรีเลชันตัวแรก คือ r1

การยูเนียนจะมีการสร้างรีเลชันใหม่ โดยจะนำทูเพิลของ R และทูเพิลของ S มารวมกัน ถ้ามีทูเพิลที่ซ้ำกันก็จะตัดทูเพิลนั้นออก

ตัวอย่าง

RESULT ← result1 ∪ result2

RESULT1	SSN
	123456789
	333445555
	666884444
	453453453

RESULT2	SSN
	333445555
	888665555

RESULT	SSN
	123456789
	333445555
	666884444
	453453453
	888665555

รูปที่ 88 โอเปอเรชันยูเนียน

จะเห็นได้ว่า ค่า 333445555 มีอยู่ทั้งในรีเลชัน RESULT1 และ RESULT2 เมื่อ UNION แล้วจะตัดทูเพิลซ้ำนั้นออก

7.2.5 INTERSECTION

โอเปอเรชันอินเตอร์เซกชัน เป็นการดำเนินการในโอเปอเรชันเชิงสัมพันธ์จาก 2 รีเลชัน (binary relational operations) เพื่อหาส่วนร่วมของข้อมูลจากตาราง 2 ตาราง มีรูปแบบการเขียนคือ

$$r \cap s$$

สัญลักษณ์ $\cap$ อ่านว่า อินเตอร์เซก (intersect)

ได้ผลลัพธ์

$$r \cap s = \{t \mid t \in r \text{ and } t \in s\}$$

ผลลัพธ์จากโอเปอเรชันอินเตอร์เซกชันจะได้รีเลชันใหม่ โดยจะนำทูเพิลที่อยู่ในรีเลชัน R และรีเลชัน S มาเก็บไว้ในตารางผลลัพธ์

การอินเตอร์เซก มีข้อบังคับเหมือนกับการยูเนียน คือ 2 ตารางที่นำมาอินเตอร์เซกกันต้องมีไทป์เดียวกัน นั่นหมายความว่าทั้งสองตารางมีจำนวนแอตทริบิวต์และโดเมนเดียวกัน เรียงเหมือนกัน แต่ไม่จำเป็นต้องมีชื่อแอตทริบิวต์เหมือนกัน

7.2.6 MINUS

โอเปอเรชันไมนัส หรือค่าต่างของเซต (set difference) คือการลบของรีเลชัน 2 รีเลชัน จึงเป็นโอเปอเรเตอร์ที่ใช้ในโอเปอเรชันเชิงสัมพันธ์จาก

2 รีเลชัน (binary relational operations) เพื่อหาส่วนต่างของตาราง 2 ตาราง มีรูปแบบการเขียนคือ

r – s

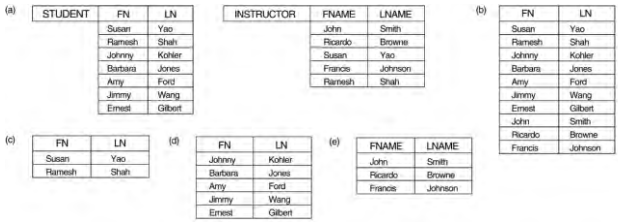
สัญลักษณ์ – อ่านว่า ไมนัส (minus)

ได้ผลลัพธ์

$r - s = \{t \mid t \in r \textbf{ and } t \notin s\}$

ผลลัพธ์จากการลบนี้จะได้รีเลชันใหม่ ที่โดยจะนำทูเพิลผลต่างระหว่างตัวตั้ง R และตัวลบ S มาสร้างรีเลชันใหม่ ซึ่งถ้าทูเพิลนั้นอยู่เฉพาะใน S อย่างเดียว หรือทูเพิลนั้นอยู่ทั้งใน R และ S ให้ตัดทิ้ง

การลบนี้มีข้อบังคับเหมือนกับการยูเนียนและอินเตอร์เซก คือ สองตารางที่นำมาลบกันต้องมีไทป์เดียวกัน นั่นหมายความว่าทั้งสองตารางมีจำนวนแอตทริบิวต์และโดเมนเดียวกัน เรียงเหมือนกัน แต่ไม่จำเป็นต้องมีชื่อแอตทริบิวต์เหมือนกัน



รูปที่ 89 การ UNION INTERSECTION และ MINUS

รูปที่ 89 แสดงการ UNION INTERSECTION และ MINUS

(a) แสดงรีเลชัน STUDENT กับ รีเลชัน INSTRUCTOR

(b) แสดงผลลัพธ์ student $\cup$ instructor

(c) แสดงผลลัพธ์ student $\cap$ instructor

(d) แสดงผลลัพธ์ student – instructor

(e) แสดงผลลัพธ์ instructor – student

คุณสมบัติเพิ่มเติม

UNION กับ INTERSECTION มีคุณสมบัติ commutative หรือสมบัติการสลับที่ คือสลับที่ได้ ผลลัพธ์ยังคงเดิม

$$r \cup s = s \cup r$$

$$r \cap s = s \cap r$$

UNION กับ INTERSECTION มีคุณสมบัติ associative หรือสมบัติการเปลี่ยนกลุ่ม คือสามารถทำโอเปอเรชันกี่รีเลชันพร้อมกันก็ได้และจัดกลุ่มอย่างไรก็ได้ ผลลัพธ์จะเท่าเดิม

$$r \cup (s \cup t) = (r \cup s) \cup t$$

$$r \cap (s \cap t) = (r \cap s) \cap t$$

MINUS ไม่มีคุณสมบัติ commutative ดังนั้น

$$r - s \neq s - r$$

ย้ำอีกครั้งว่าการ UNION, INTERSECT และ MINUS นั้น รีเลชันต้องมีไทป์เดียวกัน คือมีจำนวนแอตทริบิวต์เท่ากัน แต่ละแอตทริบิวต์มีโดเมนเดียวกัน และเรียงลำดับแอตทริบิวต์เหมือนกัน แต่ชื่อแอตทริบิวต์ไม่จำเป็นต้องเหมือนกัน สามารถอธิบายด้วยคณิตศาสตร์ว่า

$$R1(A_1, A_2, ..., A_n)$$

$$R2(B_1, B_2, ..., B_n)$$

$$dom(A_i) = dom(B_i) \text{ for } i=1, 2, ..., n$$

รีเลชันผลลัพธ์จะมีชื่อแอตทริบิวต์ตาม R1

7.2.7 CARTESIAN

โอเปอเรชันคาร์ทีเซียน CARTESIAN หรือ ครอสโปรดักต์ (cross-product) คือการเชื่อมข้อมูล 2 รีเลชันเข้าด้วยกันโดยพิจารณาข้อมูลทุกคู่ในรีเลชันทั้งสอง โดยที่รีเลชันทั้งสองไม่จำเป็นต้องเป็นไทป์เดียวกัน หมายถึงไม่จำเป็นต้องมีจำนวนแอตทริบิวต์เท่ากันและโดเมนเดียวกัน เขียนรูปแบบได้ดังนี้

$$r \times s$$

ผลลัพธ์ของ CARTESIAN จะเกิดรีเลชันใหม่ ประกอบด้วยแอตทริบิวต์จาก R และ S ทุกคู่ของทูเพิลที่เป็นไปได้ (all possible combinations of rows)

ดังนั้นดีกรี (degree) หรือจำนวนแอตทริบิวต์ของ r x s จะเท่ากับ จำนวนแอตทริบิวต์ R + จำนวนแอตทริบิวต์ S

และจำนวนทูเพิลของ r x s เขียนแทนว่า |r x s| = จำนวนทูเพิลใน R x จำนวนทูเพิลใน S

ตัวอย่าง

$R(A_1, A_2, ..., A_n)$

$S(B_1, B_2, ..., B_m)$

$q \leftarrow r \times s$

ได้ผลลัพธ์จากการ CARTESIAN ไปเก็บไว้ในรีเลชัน Q เรียงลำดับแอตทริบิวต์ดังนี้

$Q(A_1, A_2, ..., A_n, B_1, B_2, ..., B_m)$

ซึ่งมีดีกรี (degree) = n + m แอตทริบิวต์ และมีจำนวนทูเพิล n x m

อย่างไรก็ดี การ CARTESIAN เพียงอย่างเดียวจะสร้างรีเลชันที่มีข้อมูลเกินจากเดิม เป็นข้อมูลแปลกปลอม (spurious) ที่เกิดจากการคาร์ทีเซียน แต่เราจำเป็นต้อง CARTESIAN เพื่อเชื่อมรีเลชันที่สคีมาต่างกัน ดังนั้นเราต้องใช้

CARTESIAN แล้วตามด้วย SELECT เพื่อเลือกข้อมูลที่ถูกต้องมาเก็บไว้ ซึ่งเรียกว่า คำสั่ง JOIN

ตัวอย่าง

$$female\_emps \leftarrow \sigma_{SEX='F'}(employee)$$

$$empnames \leftarrow \pi_{FNAME,\ LNAME,\ SSN}\ (female\_emps)$$

รูปที่ 90 แสดงผลการ SELECT และ PROJECT เพื่อเตรียมข้อมูลไว้ดำเนินการต่อ

FEMALE_EMPS	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle,Spring,TX	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	1941-06-20	291 Berry,Bellaire,TX	F	43000	888665555	4
	Joyce	A	English	453453453	1972-07-31	5631 Rice,Houston,TX	F	25000	333445555	5

EMPNAMES	FNAME	LNAME	SSN
	Alicia	Zelaya	999887777
	Jennifer	Wallace	987654321
	Joyce	English	453453453

รูปที่ 90 ผลการ SELECT และ PROJECT

$$emp\_dependents \leftarrow empnames \times dependent$$

ได้ผลลัพธ์คือรูปที่ 91

EMP_DEPENDENTS	FNAME	LNAME	SSN	ESSN	DEPENDENT_NAME	SEX	BDATE	• • •
	Alicia	Zelaya	999887777	333445555	Alice	F	1986-04-05	• • •
	Alicia	Zelaya	999887777	333445555	Theodore	M	1983-10-25	• • •
	Alicia	Zelaya	999887777	333445555	Joy	F	1958-05-03	• • •
	Alicia	Zelaya	999887777	987654321	Abner	M	1942-02-28	• • •
	Alicia	Zelaya	999887777	123456789	Michael	M	1988-01-04	• • •
	Alicia	Zelaya	999887777	123456789	Alice	F	1988-12-30	• • •
	Alicia	Zelaya	999887777	123456789	Elizabeth	F	1967-05-05	• • •
	Jennifer	Wallace	987654321	333445555	Alice	F	1986-04-05	• • •
	Jennifer	Wallace	987654321	333445555	Theodore	M	1983-10-25	• • •
	Jennifer	Wallace	987654321	333445555	Joy	F	1958-05-03	• • •
	Jennifer	Wallace	987654321	987654321	Abner	M	1942-02-28	• • •
	Jennifer	Wallace	987654321	123456789	Michael	M	1988-01-04	• • •
	Jennifer	Wallace	987654321	123456789	Alice	F	1988-12-30	• • •
	Jennifer	Wallace	987654321	123456789	Elizabeth	F	1967-05-05	• • •
	Joyce	English	453453453	333445555	Alice	F	1986-04-05	• • •
	Joyce	English	453453453	333445555	Theodore	M	1983-10-25	• • •
	Joyce	English	453453453	333445555	Joy	F	1958-05-03	• • •
	Joyce	English	453453453	987654321	Abner	M	1942-02-28	• • •
	Joyce	English	453453453	123456789	Michael	M	1988-01-04	• • •
	Joyce	English	453453453	123456789	Alice	F	1988-12-30	• • •
	Joyce	English	453453453	123456789	Elizabeth	F	1967-05-05	• • •

รูปที่ 91 ผลการ CARTESIAN

จะเห็นว่า EMP_DEPENDENTS ซึ่งเป็นผลจากการ CARTESIAN นั้นจะเกิดทูเพิลแปลกปลอมเกินเลยไปมากมาย ไม่มีความหมาย นำผลจากการ CARTESIAN มาใช้ทันทีไม่ได้ จำเป็นต้อง SELECT ต่อไปเพื่อเลือกเฉพาะทูเพิลที่ถูกต้องเท่านั้น

เช่น จากตัวอย่างข้างบน เราต้อง SELECT เพื่อให้ได้มาซึ่งทูเพิลที่ถูกต้อง นั่นคือ มี SSN และ ESSN ที่ตรงกัน ดังนี้

ACTUAL_DEPENDENTS	FNAME	LNAME	SSN	ESSN	DEPENDENT_NAME	SEX	BDATE	• • •
	Jennifer	Wallace	987654321	987654321	Abner	M	1942-02-28	• • •

RESULT	FNAME	LNAME	DEPENDENT_NAME
	Jennifer	Wallace	Abner

รูปที่ 92 CARTESIAN ตามด้วย SELECT

7.2.8 DIVISION

การหารรีเลชัน หรือ โอเปอเรชัน DIVISION อาจไม่ค่อยเห็นใช้บ่อยนัก แต่ก็มีประโยชน์อยู่มาก เช่น ต้องการหานักเรียนทุกคนที่ลงวิชา 5 วิชา เป็นต้น

รูปแบบ DIVISION เขียนได้ดังนี้

r / s หรือ r ÷ s

ให้ $R = (A_1, ..., A_m, B_1, ..., B_n)$

$S = (B_1, ..., B_n)$

$r / s = \{ \langle x \rangle \mid \exists \langle x, y \rangle \in r \land \forall \langle y \rangle \in s \land x \in \pi_{R-S}(R) \}$

ผลลัพธ์ของ r / s จะเป็นรีเลชันที่มีสคีมา = $R - S = (A_1, ..., A_m)$

หรืออธิบายได้อีกแบบ ดังนี้

ให้ r(z) / s(x) โดยที่ X เป็นซับเซตของ Z

ให้ y = z – x (ดังนั้น z = x $\cup$ y)

จะได้ว่า y คือ เซตของแอตทริบิวต์ในรีเลชันที่เป็นผลลัพธ์ของ r / s

ตัวอย่าง

รูปที่ 93 แสดงขั้นตอนและผลลัพธ์ของ ssns ← ssn_pnos ÷ smith_pnos

SSN_PNOS	ESSN	PNO
	123456789	1
	123456789	2
	666884444	3
	453453453	1
	453453453	2
	333445555	2
	333445555	3
	333445555	10
	333445555	20
	999887777	30
	999887777	10
	987987987	10
	987987987	30
	987654321	30
	987654321	20
	888665555	20

SMITH_PNOS	PNO
	1
	2

SSNS	SSN
	123456789
	453453453

รูปที่ 93 ขั้นตอนและผลลัพธ์ของ DIVISION ssns

รูปที่ 94 แสดงผลลัพธ์ของ t ← r ÷ s

R	A	B
	a1	b1
	a2	b1
	a3	b1
	a4	b1
	a1	b2
	a3	b2
	a2	b3
	a3	b3
	a4	b3
	a1	b4
	a2	b4
	a3	b4

S	A
	a1
	a2
	a3

T	B
	b1
	b4

รูปที่ 94 ขั้นตอนและผลลัพธ์ของ DIVISION R/S

7.2.9 JOIN

เมื่อต้องการเชื่อมสองรีเลชันเข้าด้วยกัน ที่ไม่จำเป็นต้องเป็นไทป์เดียวกัน เราเชื่อมได้ด้วยการ CARTESIAN แล้วตามด้วย SELECT ซึ่งก็คือโอเปอเรชัน JOIN ที่รวมคำสั่ง CARTESIAN ตามด้วย SELECT เข้าไว้ด้วยกันนั่นเอง

กรณีที่รีเลชันทั้งสองฝั่งมีคอมมอนแอตทริบิวต์หนึ่งตัว หมายถึง มีแอตทริบิวต์คู่หนึ่งที่มีโดเมนเดียวกัน จะ JOIN โดยทำคำสั่ง CARTESIAN บนสองรีเลชันแล้ว SELECT ข้อมูลทุกทูเพิลที่แอตทริบิวต์คู่นั้นมีค่าเท่ากัน

กรณีที่รีเลชันทั้งสองฝั่งไม่มีคอมมอนแอตทริบิวต์ ผลลัพธ์ก็คือ CARTESIAN ของสองรีเลชันนั้น

กรณีที่รีเลชันทั้งสองฝั่งมีคอมมอนแอตทริบิวต์มากกว่าหนึ่งตัว ผลลัพธ์จากการ JOIN จะ SELECT ทุกทูเพิลที่ทุกคอมมอนแอตทริบิวต์มีค่าเท่ากัน

รูปแบบการ JOIN เขียนได้ดังนี้

$$r \bowtie_{\text{<join condition>}} s$$

สัญลักษณ์ $\bowtie$ อ่านว่า จอยน์ (join)

การทำงานของ JOIN จะเริ่มด้วยการคาร์ทีเซียนแล้วตามด้วย SELECT ด้วยแอตทริบิวต์อ้างอิงเสมอ

การ JOIN นี้ แบ่งได้หลายแบบ คือ

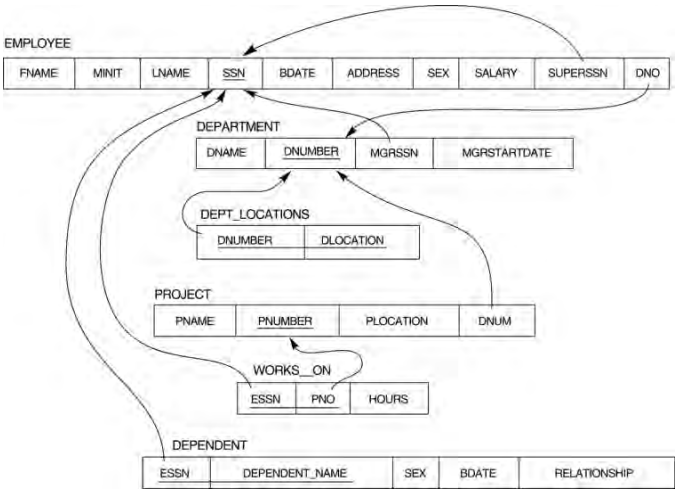
7.2.9.1 EQUI JOIN

โอเปอเรชัน EQUI JOIN เป็นการเชื่อมรีเลชันเมื่อมีอย่างน้อยหนึ่งแอตทริบิวต์จากสองรีเลชันที่มีค่าเท่ากันเท่านั้น (equality comparison)

ผลลัพธ์ของ EQUI JOIN จะต้องมีแอตทริบิวต์อย่างน้อย 1 คู่จากแต่ละรีเลชันที่มีค่าในทูเพิลเท่ากันทั้งสองตาราง

ทั้งนี้ ชื่อของแอตทริบิวต์ไม่จำเป็นต้องเหมือนกัน แต่ต้องมีโดเมนเดียวกัน

ตัวอย่าง



รูปที่ 95 รีเลชันบริษัทดีบี

จากรูปที่ 95 สมมุติว่าเราต้องการสอบถามชื่อของ Manager ของแต่ละ Department เราจำเป็นต้องเชื่อมรีเลชัน DEPARTMENT กับรีเลชัน

EMPLOYEE เพราะ ใน DEPARTMENT มีแต่ MGRSSN ไม่มีชื่อ EMPLOYEE ดังนั้น เราจึงสนใจจะเชื่อมด้วยเงื่อนไขว่า MGRSSN ในรีเลชัน DEPARTMENT มีค่าเท่ากับ SSN ในรีเลชัน EMPLOYEE โดยจะเขียนโอเปอเรชัน EQUI JOIN ได้ว่า

$$dept\_mgr \leftarrow department \bowtie_{MGRSSN=SSN} employee$$

จะได้ผลลัพธ์ดังนี้

DEPT_MGR	DNAME	DNUMBER	MGRSSN	...	FNAME	MINT	LNAME	SSN	...
	Research	5	333445555	...	Proadpran	P	Punyabukkana	333445555	...
	Admin	4	123456789	...	Kawin	T	Meisiritrakul	123456789	...
	HR	1	001001001	...	Sanchai	T	Jakteerangkur	001001001	...

รูปที่ 96 ผลลัพธ์ของ EQUI JOIN

สังเกตว่าคอลัมน์ MGRSSN และ SSN จะเหมือนกันทุกทูเพิล ซึ่งคือความซ้ำซ้อนที่ฟุ่มเฟือย (superfluous) เกินจำเป็น

7.2.9.2 NATURAL JOIN

เป็นการเชื่อมรีเลชันด้วยแอตทริบิวต์จากสองรีเลชันที่ชื่อเหมือนกันและโดเมนเดียวกัน หากชื่อแอตทริบิวต์ที่ใช้เชื่อมไม่เหมือนกัน ให้ RENAME เสียก่อน ผลลัพธ์ของ NATURAL JOIN ต่างกับ EQUI JOIN คือจะตัดแอตทริบิวต์หรือคอลัมน์ซ้ำทิ้งไป ทำให้รีเลชันผลลัพธ์ไม่พบปัญหาข้อมูลฟุ่มเฟือยและไม่จำเป็น (superfluous) ที่เกิดในกรณีของ EQUI JOIN

การเขียนโอเปอเรชัน NATURAL JOIN สามารถใช้ $\bowtie$ ก็ได้ หรือในบางตำราให้ใช้ * (สัญกรณ์ star) [2] เพื่อให้ต่างจาก EQUI JOIN

ตัวอย่าง

R = (A, B, C, D)

S = (E, B, D)

$r \bowtie s = \pi_{r.A,\ r.B,\ r.C,\ r.D,\ s.E} (\sigma_{r.B = s.B\ \ r.D = s.D} (r \times s))$

สคีมาของผลลัพธ์ = (*A, B, C, D, E*)

(a) รูปที่ 97 แสดงขั้นตอนและผลลัพธ์ของ NATURAL JOIN

proj_dept ← project ⋈ dept

(b) dept_locs ← department ⋈ dept_location

(a)

PROJ_DEPT	PNAME	PNUMBER	PLOCATION	DNUM	DNAME	MGRSSN	MGRSTARTDATE
	ProductX	1	Bellaire	5	Research	333445555	1988-05-22
	ProductY	2	Sugarland	5	Research	333445555	1988-05-22
	ProductZ	3	Houston	5	Research	333445555	1988-05-22
	Computerization	10	Stafford	4	Administration	987654321	1995-01-01
	Reorganization	20	Houston	1	Headquarters	888665555	1981-06-19
	Newbenefits	30	Stafford	4	Administration	987654321	1995-01-01

(b)

DEPT_LOCS	DNAME	DNUMBER	MGRSSN	MGRSTARTDATE	LOCATION
	Headquarters	1	888665555	1981-06-19	Houston
	Administration	4	987654321	1995-01-01	Stafford
	Research	5	333445555	1988-05-22	Bellaire
	Research	5	333445555	1988-05-22	Sugarland
	Research	5	333445555	1988-05-22	Houston

รูปที่ 97 ขั้นตอนและผลลัพธ์ของ NATURAL JOIN

7.2.9.3 THETA JOIN

โอเปอเรชัน THETA JOIN คือการ JOIN ที่ไม่จำกัดเฉพาะเงื่อนไขเท่ากับ จึงเป็นการ JOIN ที่เปิดกว้างกว่า NATURAL JOIN และ EQUI JOIN

$$r \bowtie_c s = \sigma_c(r \times s)$$

c คือเงื่อนไขการ JOIN ซึ่งจะเป็นเพรดิเคตใด ๆ ก็ได้

หากมีทูเพิลซ้ำก็จะถูกลดไป แต่คอลัมน์ซ้ำจะยังคงอยู่

7.2.10 OUTER JOIN

โอเปอเรชัน JOIN ใด ๆ ที่ผ่านมา ได้แก่ NATURAL JOIN, EQUI JOIN และ THETA JOIN จะได้ผลลัพธ์คือทูเพิลที่พบในทั้งสองรีเลชันที่ JOIN กันเท่านั้น ซึ่งอาจทำให้ไม่สามารถดึงข้อมูลที่ต้องการออกมาได้ในบางกรณี หรือเมื่อทูเพิลที่ JOIN มีค่าว่างที่แอตทริบิวต์ที่ใช้ JOIN ก็จะไม่ปรากฏในผลลัพธ์เช่นกัน ดังนั้น เมื่อต้องการเก็บข้อมูลจากบางรีเลชันไว้แม้จะ join แล้วไม่พบ เราจะใช้โอเปอเรเตอร์ OUTER JOIN ที่จะเก็บค่าทั้งหมดที่เกิดจากโอเปอเรชัน JOIN แม้ JOIN ไม่พบไว้ได้ โดยถ้าค่าในข้อมูลใดไม่มีก็จะกำหนดให้เป็นค่าว่าง

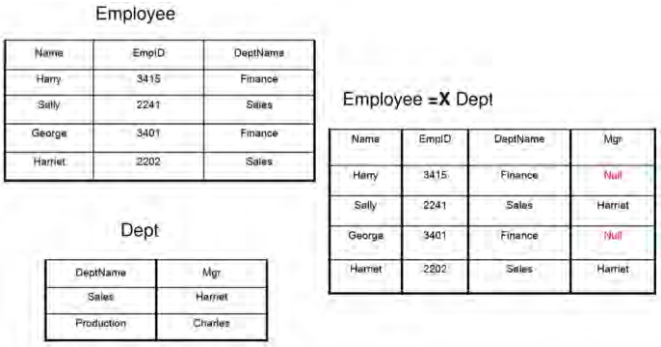
7.2.10.1 LEFT OUTER JOIN

R ⟕ S

หรือ R =x S

อ่านว่า R LEFT-OUTER-JOIN S ส่งผลให้ผลลัพธ์การ JOIN นี้เก็บทุกทูเพิลข้างซ้ายของโอเปอเรชันไว้ คือทุกทูเพิลใน R จะคงอยู่ และถ้าไม่พบว่ามีทูเพิลใน S ที่ JOIN ได้ ก็กำหนดค่าว่างให้ ดังแสดงในรูปที่ 98

ตัวอย่าง



รูปที่ 98 ขั้นตอนและผลลัพธ์การ LEFT OUTER JOIN

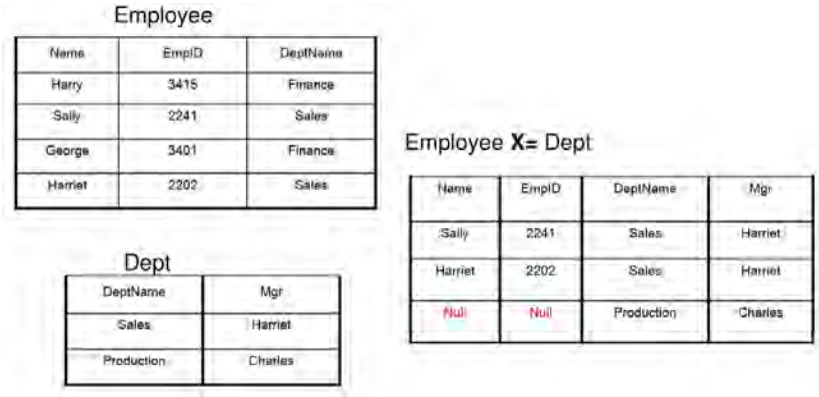
7.2.10.2 RIGHT OUTER JOIN

R ⟖ S

หรือ R x= S

อ่านว่า R RIGHT-OUTER-JOIN S ส่งผลให้ผลลัพธ์การ JOIN นี้เก็บทุกทูเพิลข้างขวาของโอเปอเรชันไว้ คือทุกทูเพิลใน S จะคงอยู่ และถ้าไม่พบว่ามีทูเพิลใน R ที่ JOIN ได้ ก็กำหนดค่าว่างให้ ดังแสดงในรูปที่ 99

ตัวอย่าง



Employee

Name	EmpID	DeptName
Harry	3415	Finance
Sally	2241	Sales
George	3401	Finance
Harriet	2202	Sales

Dept

DeptName	Mgr
Sales	Harriet
Production	Charles

Employee **X=** Dept

Name	EmpID	DeptName	Mgr
Sally	2241	Sales	Harriet
Harriet	2202	Sales	Harriet
Null	Null	Production	Charles

รูปที่ 99 ขั้นตอนและผลลัพธ์การ RIGHT OUTER JOIN

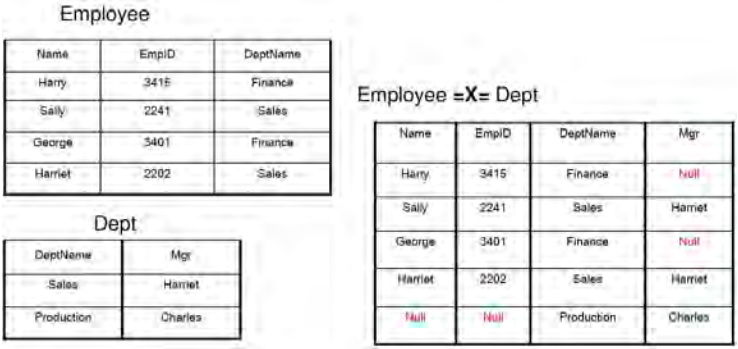
7.2.10.3 FULL OUTER JOIN

R ⟗ S

หรือ R =x= S

อ่านว่า R FULL-OUTER-JOIN S ส่งผลให้ผลลัพธ์การ JOIN นี้เก็บทุกทูเพิล ทั้งจาก R และ S พร้อมเติมค่าว่างให้กับค่าที่ JOIN ไม่พบ ดังแสดงในรูปที่ 100

ตัวอย่าง



รูปที่ 100 ขั้นตอนและผลลัพธ์การ FULL OUTER JOIN

7.2.11 แอกกรีเกตฟังก์ชันและการกรุป

ความต้องการในการสอบถามข้อมูลจากฐานข้อมูลบางอย่างไม่สามารถทำได้ด้วยนิพจน์พีชคณิตเชิงสัมพันธ์ เช่น การหาผลรวม การนับ หรือการหาค่าเฉลี่ย ซึ่งต้องอาศัยการคำนวณข้อมูลจากตารางด้วย ดังนั้น จึงได้มีแอกกรีเกตฟังก์ชันทางคณิตศาสตร์ (mathematical aggregate function) ที่ทำให้สามารถคำนวณข้อมูลในรีเลชันได้

แอกกรีเกตฟังก์ชันที่ใช้บ่อย ได้แก่ การหาผลรวม การหาค่าเฉลี่ย การหาค่าสูงสุด การหาค่าต่ำสุด และการนับ

มีรูปแบบการเขียนดังนี้

การหาค่าสูงสุด (maximum) ของแอตทริบิวต์ในรีเลชัน ใช้คำสั่ง MAX สามารถเขียนได้เป็น

$$\Im_{\text{MAX } \langle Attribute \rangle}(r)$$

การหาค่าต่ำสุด (minimum) ของแอตทริบิวต์ในรีเลชัน ใช้คำสั่ง MIN สามารถเขียนได้เป็น

$$\Im_{\text{MIN } \langle Attribute \rangle}(r)$$

การหาค่าผลรวม (summation) ของแอตทริบิวต์ในรีเลชัน ใช้คำสั่ง SUM สามารถเขียนได้เป็น

$$\Im_{\text{SUM } \langle Attribute \rangle}(r)$$

การนับจำนวนทูเพิล (count) ในรีเลชันใช้คำสั่ง COUNT สามารถเขียนได้เป็น

$$\Im_{\text{COUNT } \langle Attribute \rangle}(r)$$

ส่วนการกรุป (group) ให้เขียนชื่อแอตทริบิวต์ที่ต้องการกรุปไว้หน้า $\Im$ เช่น

$$_{\text{DNO}}\Im \text{COUNT}_{\text{SSN}} (\text{employee})$$

ตัวอย่าง

$\sigma_{r(DNO,\ NO\_OF\_EMPLOYEES,\ AVERAGE\_SAL)DNO}(_{DNO}\ \Im\ COUNT_{SSN},\ AVERAGE_{SALARY}(employee))$

ได้ผลลัพธ์ดังแสดงในรูปที่ 101 ผลลัพธ์การแอกกรีเกตรูปที่ 101

R	DNO	NO_OF_EMPLOYEES	AVERAGE_SAL
	5	4	33250
	4	3	31000
	1	1	55000

รูปที่ 101 ผลลัพธ์การแอกกรีเกต

$_{DNO}\ \Im\ COUNT_{SSN},\ AVERAGE_{SALARY}(employee)$

ได้ผลลัพธ์ดังแสดงในรูปที่ 102

COUNT_SSN	AVERAGE_SALARY
8	35125

รูปที่ 102 ผลลัพธ์การกรุป

7.3 เซตสมบูรณ์ของโอเปอเรชันเชิงสัมพันธ์

โอเปอเรชันเชิงสัมพันธ์ประกอบด้วย SELECT σ, PROJECT π, UNION $\cup$, INTERSECTION $\cap$, MINUS – และ CARTESIAN X ถือว่าเป็นเซตสมบูรณ์ เพราะโอเปอเรชันอื่นใดในพีชคณิตเชิงสัมพันธ์ล้วนเป็นการผนวก 6 โอเปอเรชันนี้ทั้งสิ้น

ตัวอย่าง

$R \cap S$ มีค่าเท่ากับ $(R \cup S) - ((R - S) \cup (S - R))$

$R \bowtie$ <join condition>S มีค่าเท่ากับ σ<join condition>$(R \times S)$

7.4 แคลคูลัสเชิงสัมพันธ์

แคลคูลัสเชิงสัมพันธ์ (relational calculus) เป็นภาษาสอบถามแบบฟอร์มัล (formal query language) สำหรับแบบจำลองเชิงสัมพันธ์

การเขียนนิพจน์ด้วยภาษานี้จะเป็นลักษณะพรรณา ไม่สนใจลำดับการทำงานแบบพีชคณิตเชิงสัมพันธ์ ผู้เขียนสามารถเขียนนิพจน์อธิบายผลลัพธ์ที่ต้องการโดยไม่ต้องระบุขั้นตอนใด ๆ

ตัวอย่างแนวคิดแคลคูลัสเชิงสัมพันธ์เปรียบเทียบกับพีชคณิตเชิงสัมพันธ์ เป็นดังนี้

สมมุติว่าเราต้องการค้นหาชื่อและหมายเลขโทรศัพท์ของร้านหนังสือที่วางขายหนังสือแฮร์รี พอตเตอร์

แคลคูลัสเชิงสัมพันธ์ แบ่งได้เป็น 2 แนวคิดคือ

1. แคลคูลัสเชิงสัมพันธ์อิงทูเพิล (tuple relational calculus)
2. แคลคูลัสเชิงสัมพันธ์อิงโดเมน (domain relational calculus)

7.4.1 แคลคูลัสเชิงสัมพันธ์อิงทูเพิล

ดร. เอดการ์ เอฟ คอดด์ เป็นผู้เสนอแนวคิดและกรรมวิธีแคลคูลัสเชิงสัมพันธ์อิงทูเพิล เมื่อ ค.ศ. 1972 [33]

แคลคูลัสเชิงสัมพันธ์อิงทูเพิล มีรูปแบบดังนี้

$$\{t|p(t)\}$$

โดย t คือ ตัวแปรทูเพิล

$p(t)$ คือ เงื่อนไขของ t

ผลลัพธ์ของนิพจน์นี้คือรีเลชันที่ประกอบด้วยทูเพิล t ที่ ทุก t เป็นจริงตามเงื่อนไข $p(t)$ ที่กำหนด

ตัวอย่าง

เมื่อต้องการค้นหา EMPLOYEE ที่เงินเดือนมากกว่า 30000 จากรีเลชัน S เขียนได้ดังนี้

$$\{S.salary|S.salary > 30000\}$$

สำหรับเงื่อนไข เราสามารถใช้ตัวบ่งปริมาณ (quantifier) ได้สองตัวคือ

1. existential quantifier $\exists$ (there exists/for some)
2. universal quantifier $\forall$ (For all)

ตัวอย่าง

ให้รีเลชัน BOOK (B) และ รีเลชัน STORE (S)

{S.Name, S.Phone|∃B(B.StoreID=S.StoreID∧B.Title='Harry Potter')}

เงื่อนไขที่ต้องเป็นจริงคือ มีบางทูเพิลใน รีเลชัน BOOK ที่มีรหัสร้านเท่ากับรหัสร้านในรีเลชัน STORE และ ชื่อหนังสือ Title มีค่าเท่ากับ 'Harry Potter'

ผลลัพธ์ที่ได้คือชื่อร้านหนังสือและหมายเลขโทรศัพท์ของร้านที่มีหนังสือแฮรีพอตเตอร์วางขาย

7.4.2 แคลคูลัสเชิงสัมพันธ์อิงโดเมน

แคลคูลัสเชิงสัมพันธ์อิงโดเมนนี้ เสนอโดย Michel Lacroix and Alain Pirotte [45] นิพจน์แคลคูลัสเชิงสัมพันธ์อิงโดเมนมีรูปแบบดังนี้

$$\{\langle x_1,x_2,...,x_n\rangle | p[\langle x_1,x_2,...,x_n\rangle\}$$

โดยแต่ละ x_i คือตัวแปรโดเมน (domain variable) และ $p[\langle x_1,x_2,...,x_n\rangle]$ คือเงื่อนไข (condition) ของตัวแปรโดเมน เรียกว่า ฟอร์มูลา

ผลลัพธ์ของนิพจน์นี้ จะได้ทุกทูเพิล $\langle x_1,x_2,...,x_n\rangle$ ที่ทำให้ฟอร์มูลา $p[\langle x_1,x_2,...,x_n\rangle]$ เป็นจริง

ฟอร์มูลาอาจซับซ้อนได้ด้วยการฟอร์มูลาซ้อนกัน สามารถใช้ logical connector $\wedge$ (and), $\vee$ (or), $\neg$ (not) ได้

ฟอร์มูลาที่เล็กที่สุดเรียกว่า อะตอมมิกฟอร์มูลา s อาจอยู่ในรูปแบบ

$$\langle x_1, x_2, \ldots, x_n \rangle \in r$$

หรือ

x op y

หรือ

x op ค่าคงที่ (constant)

op ที่ใช้ได้คือ $<, \leq, >, \geq, =, \neq$

หนึ่งอะตอมมิกฟอร์มูลาคือหนึ่งเงื่อนไข

ส่วนฟอร์มูลานั้น อาจเป็นอะตอมมิกฟอร์มูลา หรือ อะตอม

$\neg p$, $p \wedge q$, $p \vee q$ เมื่อ p, q เป็นฟอร์มูลา หรือ

$\exists x(p(x))$ เมื่อ x เป็นตัวแปรทูเพิลอิสระ

$\forall x(p(x))$ เมื่อ x เป็นตัวแปรทูเพิลอิสระ

ตัวแปรทูเพิลอิสระ (free tuple variable) หมายถึง ตัวแปรทูเพิล ที่ไม่มี $\exists$ หรือ $\forall$ กำกับ

หรืออธิบายได้ว่า $\exists x$ และ $\forall x$ ยึดเหนี่ยว (bind) x

ตัวแปรใด ๆ ที่ไม่ได้ถูกยึดเหนี่ยว ให้ถือว่าอิสระ

ดังนั้น ตัวแปร $x_1, x_2, ..., x_n$ ที่อยู่ข้างซ้ายของเครื่องหมาย | เท่านั้น ที่เป็นตัวแปรอิสระในฟอร์มูลา $p(...)$ ได้

7.5 เปรียบเทียบพีชคณิตเชิงสัมพันธ์กับแคลคูลัสเชิงสัมพันธ์

ขอเปรียบเทียบนิพจน์พีชคณิตเชิงสัมพันธ์กับแคลคูลัสเชิงสัมพันธ์จากโจทย์ว่า จงหาชื่อลูกค้าทุกคนที่กู้เงินจากธนาคารสาขาจามจุรี

นิพจน์พีชคณิตเชิงสัมพันธ์

$$\pi_{\text{customer-name}}(\sigma_{\text{branch-name='Chamchuri'}}(\sigma_{\text{borrower.loan-number=loan.loan-number}}(\text{borrower x loan})))$$

นิพจน์แคลคูลัสเชิงสัมพันธ์

$$\{\langle X\rangle \mid \langle X,Y\rangle \in \text{borrower} \wedge \exists A,B,C(\langle A,B,C\rangle \in \text{loan} \wedge B=\text{'Chamchuri'} \wedge Y=A)\}$$

7.6 **บทสรุปและเรื่องชวนคิด**

เราได้เรียนรู้ว่าฐานข้อมูลเชิงสัมพันธ์มีวิธีการคำนวณอย่างไร ทำไมจำเป็นต้องเรียนรู้ ก็เพื่อให้ในขั้นตอนการออกแบบได้คำนึงถึงการคำนวณที่เหมาะสมที่สุด หากออกแบบได้ไม่ดีแล้ว จะส่งผลให้ขั้นตอนการคำนวณซับซ้อน เพิ่มเวลาในการประมวลผลโดยไม่จำเป็น วิศวกรคอมพิวเตอร์และโปรแกรมเมอร์ที่เชี่ยวชาญเรื่องฐานข้อมูลจะสามารถปรับเขียนนิพจน์พีชคณิตที่มีประสิทธิภาพได้ หรือใช้ query optimizer ให้เลือกนิพจน์ที่ดีที่สุดก็ได้ จึงอยากชวนผู้เรียนไปศึกษาว่า optimizer ทำงานอย่างไร และเราอาจมีวิธีปรับปรุงให้ optimizer เก่งขึ้นได้อย่างไรบ้าง

แบบฝึกหัดท้ายบท

1. ผลลัพธ์ของนิพจน์พีชคณิตเชิงสัมพันธ์ หรือนิพจน์แคลคูลัสเชิงสัมพันธ์มีความเหมือน หรือต่างกัน ซึ่งผลลัพธ์นั้นคืออะไร
2. จงอธิบายความแตกต่างของโอเปอเรเตอร์ SELECT และ PROJECT โอเปอเรเตอร์ SELECT จะเป็นการเลือกบางแถวของรีเลชัน ส่วนโอเปอเรเตอร์ PROJECT จะเป็นการเลือกบางคอลัมน์ของรีเลชัน
3. ผู้เรียนคิดว่า การดำเนินการโอเปอเรเตอร์ SELECT ก่อน PROJECT และ PROJECT ก่อน SELECT มีความแตกต่างกันหรือไม่ อย่างไร ($\sigma_p(\pi_q(r))$ และ $\pi_q(\sigma_p(r))$ มีความแตกต่างกันหรือไม่ อย่างไร)
4. กำหนดรีเลชัน MOVIE เก็บข้อมูลเกี่ยวกับภาพยนตร์ในระบบชมภาพยนตร์ออนไลน์แห่งหนึ่ง ซึ่งอธิบายโดย

   MOVIE (TITLE, SYNOPSIS, DURATION, GENRE, DIRECTOR, YEAR, COMPANY)

   จงเขียนนิพจน์พีชคณิตเชิงสัมพันธ์ที่ให้ผลลัพธ์ดังต่อไปนี้

   4.1. แสดงรายละเอียดทั้งหมดของภาพยนตร์ที่เป็นประเภท comedy
   4.2. แสดงเฉพาะชื่อภาพยนตร์ทั้งหมด
   4.3. แสดงเฉพาะชื่อภาพยนตร์ที่เป็นของบริษัท Marvel เท่านั้น
   4.4. แสดงชื่อ และเรื่องย่อภาพยนตร์ ที่มีประเภท horror และระยะเวลาของภาพยนตร์ไม่เกิน 120 นาที
   4.5. แสดงชื่อภาพยนตร์ และชื่อผู้กำกับของทุกภาพยนตร์ โดยแสดงเป็นรีเลชันใหม่ ชื่อ DIRECTOR ที่มีคอลัมน์ชื่อว่า Movie และ Director

5. กำหนดรีเลชันดังต่อไปนี้

   INSTRUCTOR (FNAME, LNAME, INSTRUCTOR_ID, YEAR)
   STUDENT (FNAME, LNAME, STUDENT_ID, YEAR)
   แทนข้อมูลอาจารย์ และนิสิตของมหาวิทยาลัยแห่งหนึ่ง จงเขียนนิพจน์พีชคณิตเชิงสัมพันธ์ที่ให้ผลลัพธ์ดังต่อไปนี้

   5.1. แสดงข้อมูลทั้งหมดของอาจารย์ และนักเรียนทุกคน รวมไว้ในรีเลชันใหม่ที่อธิบายโดย

   PEOPLE (FNAME, LNAME, PEOPLE_ID, YEAR)

   5.2. แสดงข้อมูลเฉพาะชื่อ นามสกุล ของคนที่เป็นทั้งอาจารย์ และนักเรียน
   5.3. แสดงข้อมูลเฉพาะชื่อ นามสกุล ของคนที่เป็นอาจารย์ หรือนักเรียน อย่างใดอย่างหนึ่ง
   5.4. แสดงข้อมูลเฉพาะชื่อ นามสกุลของอาจารย์ ที่มีญาติเป็นนักเรียน (ยกเว้นตัวเอง) กล่าวคือ เป็นอาจารย์ที่มีนักเรียนนามสกุลเดียวกับอาจารย์ แต่ชื่อต่างกัน

6. กำหนดรีเลชันดังต่อไปนี้

   MENU (MENU_ID, MENU_NAME, MENU_PRICE)
   ORDER (ORDER_ID, CUSTOMER_NAME)
   ORDER_ITEM (ORDER_ID, MENU_ID)
   RECOMMENDED_MENU (MENU_ID)

แทนข้อมูลเมนู ข้อมูลการสั่งอาหาร รายการอาหารที่สั่ง และเมนูแนะนำ ของร้านอาหารแห่งหนึ่ง จงเขียนนิพจน์พีชคณิตเชิงสัมพันธ์ที่ให้ผลลัพธ์ดังต่อไปนี้

6.1. แสดงรายการชื่อเมนู และหมายเลขการสั่งซื้อ ที่ตรงกัน (ORDER_ID และ MENU_NAME)

6.2. แสดงรายชื่อเมนู ราคา ที่ลูกค้าสั่ง ตามรายชื่อของลูกค้า (CUSTOMER_NAME, MENU_NAME และ MENU_PRICE)

6.3. แสดงรายชื่อลูกค้าทั้งหมด ที่เคยสั่งเมนูแนะนำครบทุกเมนู

6.4. แสดงราคารวม แยกตามแต่ละ ORDER_ID

6.5. แสดงชื่อลูกค้า และจำนวนเงินที่จ่ายไป สำหรับลูกค้าที่ใช้จ่ายรวมมากที่สุดของร้าน

7. จงแปลงนิพจน์พีชคณิตเชิงสัมพันธ์ต่อไปนี้ เป็นนิพจน์แคลคูลัสเชิงสัมพันธ์

7.1. $\pi_{NAME,SALARY}$ ($\sigma_{SALARY > 20000}$ (employee))

7.2. $\pi_{STUDENT\_NAME,\ COURSE\_NAME}$ (student $\bowtie$ enrollment $\bowtie$ course)

โดยที่ student.id = enrollment.student_id และ enrollment.course_id = course.id

บทที่ 8
ภาษาเอสคิวแอล

วัตถุประสงค์

1. เพื่อให้รู้จักภาษาเอสคิวแอลเบื้องต้นจากตัวอย่าง
2. เพื่อให้ใช้ภาษาเอสคิวแอลสอบถามข้อมูลจากฐานข้อมูลแบบง่ายได้

ภาษาเอสคิวแอล (structured query language : SQL) เป็นภาษาชนิดโดเมนจำเพาะ (domain-specific) กล่าวคือ เป็นภาษาที่ออกแบบมาเพื่องานบางอย่างโดยเฉพาะ

ภาษาเอสคิวแอลถูกออกแบบมาเพื่อให้ผู้ใช้สามารถจัดการข้อมูลในระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ และเหมาะกับข้อมูลแบบมีโครงสร้างและมีความสัมพันธ์ต่อกัน สามารถเข้าถึงข้อมูลหลายทูเพิลพร้อม ๆ กันในเพียงคำสั่งเดียว โดยที่ผู้ใช้เอสคิวแอลไม่จำเป็นต้องรู้ว่าข้อมูลถูกจัดเก็บอย่างไรหรืออยู่ที่ไหน [46-50] ภาษาเอสคิวแอลสร้างมาบนพื้นฐานของแคลคูลัสเชิงสัมพันธ์

วัตถุประสงค์ของบทนี้คือให้รู้จักภาษาเอสคิวแอลเบื้องต้นเท่านั้น เน้นตัวอย่างคำสั่งพื้นฐานเพื่อให้เห็นรูปแบบของภาษา หากผู้เรียนต้องการเรียนวากยสัมพันธ์ (syntax) หรือคำสั่งภาษาเอสคิวแอลอย่างละเอียด ขอแนะนำให้ศึกษาจากตำราเอสคิวแอลต่อไป

8.1 รู้จักภาษาเอสคิวแอล

เอสคิวแอลเขียนง่าย มีวากยสัมพันธ์และไวยากรณ์ยืดหยุ่น ไม่ซับซ้อน แต่ทรงพลังในการค้นหาและจัดการข้อมูล เราสามารถเข้าใจและเขียนเอสคิวแอลได้อย่างเป็นธรรมชาติ (intuitive) เพราะเป็นภาษาที่ใกล้เคียงกับภาษามนุษย์ เมื่อเราเขียนคำสั่งเอสคิวแอลแล้วก็จะเป็นหน้าที่ของระบบการจัดการฐานข้อมูลที่จะประมวลผลและนำผลลัพธ์มาแสดง นอกเหนือจากนั้น เรายังสามารถใช้ตัวออปทิไมซ์คิวรี (query optimizer) จัดเรียงคำสั่งเอสคิวแอลให้ได้ประสิทธิภาพที่สุดแล้วความหมายคงเดิมได้ด้วย

ภาษาเอสคิวแอล เสนอโดยบริษัทไอบีเอ็มเมื่อช่วง ค.ศ. 1974 ชื่อว่า Structured Query Language หรือ SEQUEL เพื่อใช้กับระบบจัดการฐานข้อมูลเชิงสัมพันธ์ของไอบีเอ็ม ชื่อ System R ต่อมาเปลี่ยนชื่อเป็น SQL

เมื่อเอสคิวแอลได้รับความนิยมอย่างแพร่หลาย จึงจำเป็นต้องกำหนดมาตรฐาน ในที่สุด เมื่อ ค.ศ. 1986 สถาบันมาตรฐานแห่งสหรัฐอเมริกา (American National Standards Institute : ANSI) ได้ให้การรับรองมาตรฐานเอสคิวแอล เรียกว่า SQL-86 ต่อมา ISO ก็รับรองมาตรฐานเอสคิวแอลใน ค.ศ. 1987 จากนั้นต่อไป มาตรฐานนี้ก็ได้ถูกปรับปรุงมาเรื่อย ๆ โดยหน่วยงานมาตรฐานข้ามชาติหลายหน่วยงานร่วมกัน

ภาษาเอสคิวแอล ทำให้ผู้ใช้

- สามารถสร้างฐานข้อมูลโดยกำหนดรีเลชันได้
- ทำให้ผู้ใช้สามารถจัดการข้อมูลได้ ทั้งการแทรกข้อมูล ลบข้อมูล หรือแก้ไขข้อมูล
- ทำให้ผู้ใช้สามารถสอบถามข้อมูลได้ง่ายอีกด้วย

รูปแบบในการเขียนคำสั่งเอสคิวแอล ให้ใช้ตัวอักษรตัวใหญ่เมื่อเขียนวากยสัมพันธ์ที่เป็นคำสงวนในเอสคิวแอล (reserved word) และใช้ตัวอักษรตัวเล็กเมื่อเขียนศัพท์อื่นใดที่ผู้เขียนกำหนดเอง เช่น ชื่อตาราง ชื่อฟิลด์

ทั้งนี้ ขอเทียบศัพท์ที่ใช้ในแบบจำลองเชิงสัมพันธ์และพีชคณิตเชิงสัมพันธ์กับศัพท์ที่ใช้ในเอสคิวแอลที่อาจทำให้สับสนได้ดังแสดงในตารางที่ 3

ตารางที่ 3 ศัพท์ที่ใช้ในแบบจำลองเชิงสัมพันธ์และพีชคณิตเชิงสัมพันธ์กับศัพท์ที่ใช้ในเอสคิวแอล

ศัพท์แบบจำลองเชิงสัมพันธ์และพีชคณิตเชิงสัมพันธ์	**ศัพท์เอสคิวแอล**
รีเลชัน (relation)	ตาราง (table)
แอตทริบิวต์ (attribute)	ฟิลด์ (field)
PROJECT	SELECT
SELECT	WHERE

ตัวอย่าง

STUDENTS	sid	name	login	age	gpa
	6120056721	Chonlathorn	chonlathorn.k@student.chula.ac.th	23	4.00
	6120061221	Kailerk	kailerk.t@student.chula.ac.th	18	3.80
	6121005121	Kawin	kawin.m@student.chula.ac.th	18	3.50
	6121007721	Panusorn	panusorn.p@student.chula.ac.th	24	3.75
	6121010121	Sanchai	sanchai.j@student.chula.ac.th	25	4.00
	6121026321	Thanyanuch	thanyanuch.c@student.chula.ac.th	28	3.98

รูปที่ 103 รีเลชัน STUDENTS

จากรีเลชัน STUDENTS ในรูปที่ 103 หากเราต้องการหาชื่อนักเรียนที่อายุ 18 ปี เราสามารถเขียนคำสั่งเอสคิวแอลดังนี้

```
SELECT  *
FROM    Students S
WHERE   S.age = 18
```

ผลลัพธ์คือ

STUDENTS	sid	name	login	age	gpa
	6120061221	Kailerk	kailerk.t@student.chula.ac.th	18	3.80
	6121005121	Kawin	kawin.m@student.chula.ac.th	18	3.50

รูปที่ 104 ผลลัพธ์การ SELECT *

หรือหากต้องการได้ผลลัพธ์เฉพาะชื่อและ login ให้แก้คำสั่งบรรทัดแรกเป็น

```
SELECT  S.name, S.login
FROM    Students S
WHERE   S.age = 18
```

ตัวอย่าง

จากรีเลชัน STUDENS ในรูปที่ 103 และรีเลชัน ENROLL ในรูปที่ 105

ENROLL	sid	cid	gpa
	6120056721	Database201	4.00
	6120061221	Java101	3.80
	6121005121	SA112	3.50
	6121007721	SE113	3.75

รูปที่ 105 รีเลชัน ENROLL

หากต้องการสอบถามนักเรียนที่ได้ gpa = 4.0 จะเขียนได้ดังนี้

```
SELECT  S.name, E.cid
FROM    Students S, Enroll E
WHERE   S.sid = E.sid AND E.gpa = 4.0
```

ผลลัพธ์แสดงในรูปที่ 106

S.name	E.cid
Chonlathorn	Database201

รูปที่ 106 ผลลัพธ์นักเรียน gpa = 4.0

จะเห็นได้ว่า เราสามารถเข้าใจคำสั่งได้โดยไม่ต้องอธิบาย

8.2 **คำสั่งที่ใช้บ่อย**

ในบทนี้ เราจะนำเสนอคำสั่งที่ใช้บ่อย โดยแสดงตัวอย่างคำสั่งเอสคิวแอล เพราะเชื่อว่าไม่จำเป็นต้องอธิบายวากยสัมพันธ์ (syntax) ก็สามารถเข้าใจได้ง่าย

8.2.1 สร้างรีเลชัน

เมื่อต้องการสร้างรีเลชัน ให้ใช้คำสั่ง CREATE TABLE แล้วกำหนดแอตทริบิวต์พร้อมทั้งโดเมน ดังนี้

```
CREATE TABLE Students    (sid CHAR(20),
                          name CHAR(20),
                          login CHAR(10),
                          age INTEGER,
                          gpa REAL)

CREATE TABLE Enrolled    (sid CHAR(20),
                          cid CHAR(20),
                          grade CHAR(2))
```

8.2.2 ลบรีเลชัน

เมื่อต้องการลบรีเลชัน ให้ใช้คำสั่ง DROP

```
DROP TABLE Students
```

คำสั่งนี้จะลบสคีมา Students และข้อมูลทุกทูเพิลทิ้งไป

8.2.3 แก้ไขรีเลชัน

การแก้ไขสคีมาของตาราง ใช้คำสั่ง ALTER เช่น ต้องการเพิ่มฟิลด์ จะส่งผลให้ทุกทูเพิลในตารางมีฟิลด์เพิ่มและให้ใส่ค่า null ไปที่ฟิลด์ใหม่นั้น

```
ALTER TABLE Students
        ADD COLUMN firstYear INTEGER
```

8.2.4 เพิ่มทูเพิล

เพิ่มทีละทูเพิล ใช้คำสั่ง INSERT

```
INSERT
INTO    Students (sid, name, login, age, gpa)
VALUES ('5931072221', 'Sanchai',
        'Sanchai.J@chula.ac.th', 18, 3.9)
```

8.2.5 ลบทูเพิล

ลบทูเพิลตามเงื่อนไขที่ระบุ ใช้คำสั่ง DELETE

เช่น ลบทูเพิลที่ name = Proadpran

```
DELETE
FROM    Students S
WHERE   S.name = 'Proadpran'
```

8.2.6 อัปเดตทูเพิล

อัปเดตทีละทูเพิล ใช้คำสั่ง UPDATE

```
UPDATE   Students S
SET      S.age = S.age + 1, S.gpa = S.gpa + 0.1
WHERE    S.sid = '5930136421'
```

8.2.7 กำหนดคีย์หลักและคีย์คู่แข่ง

หากมีคีย์คู่แข่ง ให้เลือกคีย์หนึ่งเป็นคีย์หลัก ใช้คำสั่ง PRIMARY KEY และหากต้องการให้ค่าในแอตทริบิวต์ใด ๆ ห้ามซ้ำกัน ให้ใช้คำสั่ง UNIQUE

```
CREATE TABLE Enrolled  (sid     CHAR(20)
                        cid     CHAR(20),
                        grade   CHAR(2),
                        PRIMARY KEY (sid, cid))
```

กรณีนี้ นักเรียน 1 คน ลงทะเบียน 1 วิชา จะได้เกรด 1 เกรด แต่นักเรียนคนนี้จะลงทะเบียนวิชาเดิมไม่ได้

```
CREATE TABLE Enrolled  (sid     CHAR(20)
                        cid     CHAR(20),
                        grade   CHAR(2),
                        PRIMARY KEY (sid),
                        UNIQUE (cid, grade))
```

ถ้าเขียนแบบนี้ นักเรียน 1 คน จะลงทะเบียนได้เพียงวิชาเดียว นอกเหนือจากนั้น จะไม่มีนักเรียนคนไหนได้เกรด ซ้ำกันด้วย

ดังนั้น การกำหนดคีย์นั้นต้องออกแบบให้ดี ตรงตามความหมายที่ต้องการ ให้รองรับข้อมูลตามที่ควรจะเป็น

8.2.8 กำหนดฟอเรนคีย์

การกำหนดฟอเรนคีย์ ใช้คำสั่ง FOREIGN KEY และ REFERENCES เพื่อโยงความสัมพันธ์ไปยังรีเลชันที่เชื่อมโยง

```
CREATE TABLE Enrolled (sid CHAR(20),
                       cid CHAR(20),
                       grade CHAR(2),
                       PRIMARY KEY (sid, cid),
                       FOREIGN KEY (sid)
                            REFERENCES Students)

CREATE TABLE Works_In (ssn      CHAR(13),
                       did   INTEGER,
                       since      DATE,
                       PRIMARY KEY (ssn, did),
                       FOREIGN KEY (ssn)
                            REFERENCES Employees,
                       FOREIGN KEY (did)
                            REFERENCES Departments)
```

8.2.9 กำหนดทางเลือกเมื่อลบหรืออัปเดตข้อมูลอ้างอิงข้ามตาราง

มีทางเลือก 4 ทางคือ

- NO ACTION ปฏิเสธการลบหรืออัปเดต
- CASCADE ลบทุกทูเพิลที่อ้างอิงทูเพิลที่ถูกลบ
- SET NULL กำหนดค่าว่างให้ฟอเรนคีย์ของทูเพิลที่จะลบ
- SET DEFAULT กำหนดค่าดีฟอลต์ให้ฟอเรนคีย์ของทูเพิลที่จะลบ

```
CREATE TABLE Enrolled (sid    CHAR(20),
                       cid    CHAR(20),
                       grade  CHAR(2),
                       PRIMARY KEY (sid,cid),
                       FOREIGN KEY (sid)
                       REFERENCES  Students
                            ON DELETE CASCADE
                            ON UPDATE SET DEFAULT)
```

8.2.10 สร้างวิว

คำสั่ง VIEW ใช้เพื่อกำหนดวิวให้ผู้ใช้

```
CREATE VIEW YoungActiveStudents (name, grade)
        AS SELECT  S.name, E.grade
           FROM    Students S, Enrolled E
           WHERE   S.sid = E.sid AND S.age < 21
```

8.3 **บทสรุปและเรื่องชวนคิด**

ภาษาเอสคิวแอลเป็นภาษาที่เข้าใจง่าย เรียนรู้ง่าย เขียนง่าย เพราะเป็นธรรมชาติ คล้ายภาษามนุษย์ สร้างบนพื้นฐานของแคลคูลัสเชิงสัมพันธ์ ผู้ใช้ภาษานี้สามารถสอบถามและจัดการข้อมูลในฐานข้อมูลได้ไม่ยาก ขอให้ผู้เรียนได้ลองคิดดูว่ามีเหตุการณ์ใดที่ภาษาเอสคิวแอลไม่สามารถรองรับได้บ้าง

แบบฝึกหัดท้ายบท

ระบบฐานข้อมูลนักเรียน อาจารย์ และการลงทะเบียนเรียนของมหาวิทยาลัยหนึ่ง ประกอบด้วย 5 รีเลชัน ซึ่งมีตัวอย่างข้อมูลดังแสดงด้านล่างนี้ จงเขียนภาษาเอสคิวแอลเพื่อทำตามคำสั่งที่กำหนดต่อไปนี้

STUDENT

STUDENT_ID (PK)	STUDENT_FNAME	STUDENT_LNAME	YEAR
6030000121	เกษม	สวัสดี	2017
6030000221	ขวัญชนก	ใจดี	2017
5930000321	คามิน	สวัสดี	2016
5930000421	จิรัฐ	สุขี	2016

INSTRUCTOR

INSTRUCTOR_ID (PK)	INSTRUCTOR_NAME	INSTRUCTOR_LNAME
100041	ชูใจ	เสียงเสนาะ
100042	เอก	ชัยชนะ
100043	ณภัทร	หมื่นลี้
100044	เมธัส	มานะวงศ์

COURSE

COURSE_ID (PK)	COURSE_NAME
2110322	DATABASE SYSTEMS
2110221	COMP ENG ESS
2110202	DISCRETE STRUC COM

COURSE_SECTION

SECTION_ID (PK)	COURSE_ID (FK)	INSTRUCTOR_ID (FK)	YEAR	SEMESTER	SECTION	DAY	TIME	CAPACITY
40001	2110322	100041	2016	1	1	MON	9:00 – 12:00	30
40002	2110322	100042	2016	1	2	TUE	13:00 – 16:00	30

ENROLLMENT

STUDENT_ID (PK, FK)	SECTION_ID (PK, FK)	GRADE
6030000121	40001	4.00
6030000221	40002	3.50
5930000321	40001	2.50

1. จงเขียนคำสั่งเพื่อสร้างตาราง ENROLLMENT โดยมีข้อกำหนดเพิ่มเติม คือ STUDENT_ID มีความยาว 10 อักขระ และ SECTION_ID มีความยาว 5 อักขระ และกำหนดให้ ถ้าหากข้อมูลนักเรียนในตาราง

STUDENT ถูกลบทิ้ง หรือ ข้อมูล SECTION ในตาราง COURSE_SECTION ถูกลบทิ้ง ข้อมูล ENROLLMENT ที่เกี่ยวข้องจะถูกลบทิ้งด้วย

2. เมื่อใช้งานระบบไปนานวันเข้า ความยาวของ SECTION_ID เพียง 5 อักขระไม่พอในการใช้งาน จงเขียนคำสั่งเพื่อแก้ไขรีเลชัน COURSE_SECTION โดยแก้ไขให้ SECTION_ID มีความยาว 6 อักขระ
3. ในปีการศึกษาใหม่ จะมีนักเรียนเข้าเรียนใหม่อยู่เสมอ จงเขียนคำสั่งเพื่อเพิ่มข้อมูลนักเรียน ที่มีรหัสนักเรียน 6039999921 ชื่อ นงนุช รักเรียน ซึ่งเข้าเรียนปี 2016 ลงไปในตาราง STUDENT
4. เมื่อนักเรียนมีการเปลี่ยนชื่อ หรือนามสกุล จะต้องแจ้งผู้ดูแลระบบเพื่อแก้ไขชื่อนามสกุลในระบบให้ถูกต้อง จงเขียนคำสั่งเพื่อแก้ไขข้อมูลชื่อนักเรียน ที่มีรหัสนักเรียน 6039999921 โดยเปลี่ยนแปลงเป็นชื่อ ขวัญชนก
5. เมื่อนักเรียนลาออก ผู้ดูแลระบบจะดำเนินการลบข้อมูลนักเรียนออกจากระบบ จงเขียนคำสั่งเพื่อลบนักเรียนที่มีรหัสนักเรียน 6039999921
6. จงเขียนคำสั่งเพื่อแสดงเฉพาะรหัสนักเรียน ชื่อ นามสกุล ของนักเรียนที่เข้าเรียนในปี 2017 ทุกคน
7. จงเขียนคำสั่งเพื่อแสดงเฉพาะรหัสนักเรียน ชื่อ นามสกุล ของนักเรียนที่เข้าเรียนในปี 2017 ลำดับที่ 16 – 25
8. จงเขียนคำสั่งเพื่อแสดง section ชื่อ-นามสกุลของอาจารย์ที่สอน วัน เวลา และจำนวนนิสิตที่รับทั้งหมดของวิชา 2110322 ในปีการศึกษา 2016 เทอม 1

9. จงเขียนคำสั่งเพื่อแสดงจำนวนนักเรียนที่ชื่อซ้ำกัน โดยแสดง ชื่อ และจำนวนคนที่ซ้ำกัน เรียงลำดับตามจำนวนที่ซ้ำกัน หากมีจำนวนที่ซ้ำกันมากกว่า 1 อันดับ
10. จงเขียนคำสั่งเพื่อแสดงเกรดเฉลี่ยของรายวิชา โดยแสดง รหัสวิชา (COURSE_ID) และเกรดเฉลี่ยรวมของรายวิชานั้น โดยเรียงลำดับตามเกรดเฉลี่ยมากไปหาน้อย

บทที่ 9
การจัดเก็บและการทำดัชนี

วัตถุประสงค์

1. เพื่อให้ทราบวิธีการจัดเก็บข้อมูลของฐานข้อมูล
2. เพื่อให้คำนวณเวลาที่ใช้ในการทำงานของโอเปอเรชัน
3. เพื่อให้ทราบประเภทของดัชนีและเลือกใช้ได้เหมาะสม

เมื่อเรามีข้อมูลในฐานข้อมูลจำนวนมาก เราจะต้องคำนึงถึงวิธีการจัดเก็บที่เหมาะสมด้วย โดยมีเป้าหมายหลักคือให้ตอบสนองผู้ใช้ได้รวดเร็ว ทันกาล วิธีการจัดเก็บข้อมูลมีหลายวิธี เราต้องออกแบบตามลักษณะการใช้งานของฐานข้อมูลนั้น ๆ นอกเหนือจากนั้น เรายังสามารถเพิ่มประสิทธิภาพในการเรียกใช้ให้ได้รวดเร็วด้วยกันเพิ่มดัชนีให้กับฐานข้อมูลอีกด้วย

ในบทนี้ เราจะพูดเรื่องการจัดเก็บข้อมูลและการทำดัชนีในฐานข้อมูล เพื่อให้การเข้าถึงข้อมูลมีประสิทธิภาพ เหมาะกับรูปแบบการสอบถามและการใช้งาน โดยเฉพาะงานที่ใช้บ่อย ดังนั้น เราจึงต้องเรียนรู้การใช้งานข้อมูลเป็นอย่างไร จึงจะออกแบบการจัดเก็บให้เหมาะสม และทำดัชนีที่ส่งผลให้ประสิทธิภาพการใช้งานของฐานข้อมูลที่ดีด้วย อย่างไรก็ดี ไม่ว่าเราจะออกแบบด้วยความระมัดเพียงใด เมื่อนำฐานข้อมูลไปใช้งานจริงก็ยังมีโอกาสที่จะต้องปรับวิธีการจัดเก็บและปรับดัชนีตามสถานการณ์ที่เปลี่ยนไป เพื่อให้เหมาะกับการใช้งานและตอบสนองความต้องการของผู้ใช้อยู่เสมอ หน้าที่ของผู้ดูแลฐานข้อมูล จึงเป็นผู้ที่มีความสำคัญอย่างยิ่ง โดยเฉพาะในองค์กรที่ใช้ฐานข้อมูลขนาดใหญ่และมีผู้ใช้ฐานข้อมูลจำนวนมาก

ตำรานี้อ้างอิงวิธีการดัชนีทั้งบทนี้จาก [1]

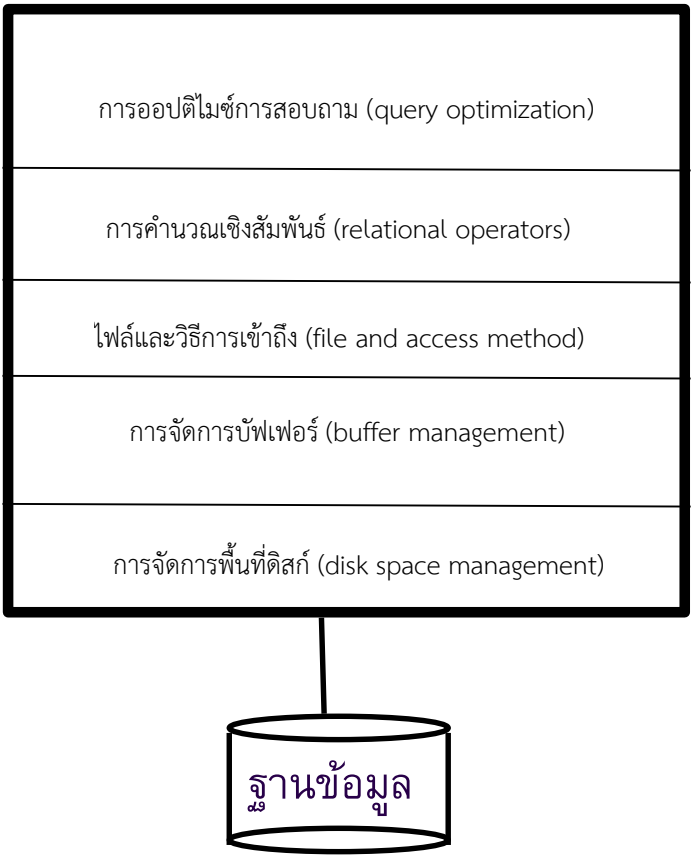
รูปที่ 107 แสดงโครงสร้างของระบบการจัดการฐานข้อมูล ประกอบด้วย 5 ชั้นคือ

1) การออปติไมซ์การสอบถาม (query optimization)
2) การคำนวณเชิงสัมพันธ์ (relational operators)
3) ไฟล์และวิธีการเข้าถึง (file and access method)

4) การจัดการบัฟเฟอร์ (buffer management)

5) การจัดการพื้นที่ดิสก์ (disk space management)

ส่วนที่เกี่ยวข้องการการจัดเก็บและดัชนีคือ 3 ชั้นหลัง ได้แก่ ชั้นไฟล์และวิธีการเข้าถึง



รูปที่ 107 โครงสร้างของระบบการจัดการฐานข้อมูล

เมื่อเราต้องการจะเก็บข้อมูล ระบบการจัดการฐานข้อมูลจะส่ง request ไปยังส่วนการจัดการบัฟเฟอร์ เพื่อจัดสรร (allocate) พื้นที่ให้ หรือเมื่อต้องการอ่านข้อมูลใด ส่วนการจัดการบัฟเฟอร์ก็จะทำหน้าที่หาว่าข้อมูลนั้นอยู่ที่ตำแหน่งใด แล้วดึงข้อมูลนั้นมาเก็บไว้ในหน่วยความจำหลัก (main memory)

9.1 หน่วยเก็บข้อมูลภายนอก

การทำงานในระบบการจัดการฐานข้อมูลส่วนใหญ่คือการอ่าน (read) ข้อมูลและเขียน (write) ข้อมูลลงบนจากหน่วยเก็บข้อมูลภายนอก (external storage)

หน่วย (unit) วัดความเร็วในการอ่านและเขียนหรืออินพุตเอาต์พุต ย่อว่าไอโอ (input/output – i/o) จึงไม่คิดเป็นหน่วยเวลา ไม่นับเป็นฟลอป (flop) แบบที่วัดความเร็วของระบบปฏิบัติการคอมพิวเตอร์ทั่วไป แต่จะคิดเป็น เพจ (page) โดยเฉพาะเจาะจงคือเพจอินพุตเอาต์พุต เรียกย่อว่า เพจไอโอ (page i/o) ว่ามีกี่ครั้งที่เราจะต้องเข้าถึงเพจ หรือเข้าถึงหน่วยจัดเก็บรอง (secondary storage) เพราะการอ่านเขียนข้อมูลเป็นงานที่กินเวลา เราจึงคำนวณเวลาที่ระบบปฏิบัติการฐานข้อมูลใช้เป็นหน่วยเพจ เพราะเวลาที่ซีพียูใช้จะไม่ต่างมากเท่ากับไอโอ ดังนั้น เพื่อให้ระบบฐานข้อมูลตอบสนองได้เร็ว เราจึงมีเป้าหมายลดเพจไอโอให้ต่ำที่สุด

9.1.1 ประเภทของหน่วยเก็บข้อมูลภายนอก

หน่วยเก็บข้อมูลภายนอก แบ่งออกเป็น 2 ประเภทได้แก่

1. ดิสก์ (disk) เป็นอุปกรณ์ที่บันทึกข้อมูลได้จำนวนมาก ใช้เทคโนโลยีแม่เหล็ก หรือแสง หรือทั้งสองอย่าง เช่น แมกเนติกดิสก์ แผ่นซีดี ดีวีดี และหน่วยความจำโซลิดสเตต ข้อดีคือ เก็บข้อมูลได้มาก และสามารถเข้าถึงข้อมูลด้วยแบบสุ่ม (random access) คือเข้าถึงข้อมูลตรงที่ที่ข้อมูลถูกบรรจุอยู่ได้ทันที
2. เทป (tape) เป็นหน่วยเก็บข้อมูลที่ใช้กันมานานตั้งแต่คอมพิวเตอร์ยุคต้น ปัจจุบันได้รับความนิยมน้อยลง มีลักษณะเป็นม้วน หรือตลับ (cartridge) การเข้าถึงข้อมูลเป็นแบบลำดับ (sequential access) จึงเข้าถึงได้ช้า ข้อดีคือราคาถูกกว่าดิสก์มากและเคลื่อนย้ายง่าย จึงเหมาะกับการเก็บข้อมูลที่ไม่ได้ใช้เป็นประจำ เช่นใช้สำรองข้อมูล (backup) หรือใช้เก็บข้อมูลถาวร (archive) ที่เมื่อนำกลับมาใช้ก็ให้อ่านข้อมูลทั้งหมดจากเทป

การอ่านข้อมูลแต่ละเพจจากดิสก์จึงใช้เวลาเท่ากัน (fixed cost) ดังนั้น การอ่านข้อมูลจากเพจแบบต่อเนื่อง (consecutive pages) จึงจะสิ้นเปลืองเวลาน้อยกว่าการแรนดอมอ่านที่ละเพจอยู่มาก

9.2 ไฟล์และวิธีการเข้าถึง

การเก็บรีเลชันลงในหน่วยเก็บข้อมูลจะเก็บในลักษณะไฟล์ของ เรกคอร์ด (record) ซึ่งเรกคอร์ดก็คือทูเพิลหรือแถวข้อมูลนั่นเอง การบันทึกรีเลชันเหล่านี้ลงในหน่วยเก็บข้อมูลจะเก็บเป็นเพจ ที่มี รหัสเรกคอร์ด (Record ID) ย่อว่า RID ต่อจากนี้จะเรียก RID ทำหน้าที่เป็นพอยน์เตอร์หรือตัวชี้ (pointer) บอกเพจของเรกคอร์ดนั้น

ชั้นไฟล์และวิธีการเข้าถึงนี้ มีหน้าที่จัดเก็บข้อมูลและระบุเพจของข้อมูล และติดตามสถานะการจัดสรรเพจให้แก่ไฟล์ ตรวจสอบเนื้อที่ว่างในแต่ละเพจที่ถูกจัดสรรแล้ว ขั้นตอนนี้จะเกิดขึ้นทุกครั้งที่มีการสร้าง (create) แทรก (insert) ลบ (delete) และ สแกน (scan) ข้อมูล ซึ่งการสแกนข้อมูลคือการค้นคืน (retrieve) ข้อมูลจำนวนมาก เช่น หาข้อมูลแล้วพิมพ์รายงานการลงทะเบียนของนิสิตทั้งมหาวิทยาลัย เป็นต้น

9.3 การจัดโครงสร้างไฟล์

การจัดโครงสร้างไฟล์ (file organization) คือกระบวนการจัดระเบียบเรกคอร์ดในไฟล์เพื่อนำไปเก็บในดิสก์

ข้อมูลในดิสก์อาจถูกเก็บแบบเรียงลำดับ (sorted) หรือไม่เรียงลำดับ (unsorted) การที่จะเก็บแบบเรียงหรือไม่เรียงลำดับนั้น ให้คำนึงว่าเป้าหมายของผู้ออกแบบฐานข้อมูลคือต้องการจะลดจำนวนเพจไอโอให้เหลือน้อยที่สุด เพื่อให้การทำงานของฐานข้อมูลใช้เวลาน้อยที่สุด ซึ่งจะขึ้นอยู่กับโอเปอเรชันที่ฐานข้อมูลนั้นใช้บ่อย โดยเฉพาะโอเปอเรชันการสอบถามข้อมูล

เราสามารถแบ่งโครงสร้างไฟล์ได้ 3 แบบ คือ

1. ไฟล์แบบไม่เรียงลำดับ (unsorted file) หรือเรียกอีกชื่อว่าไฟล์ฮีป (heap file) คือโครงสร้างไฟล์แบบไม่เรียงลำดับ ซึ่งทำให้การเข้าถึงข้อมูลเป็นแบบลำดับ เหมาะกับโอเปอเรชันสแกน ที่ต้องการอ่านข้อมูลทุกเรกคอร์ด

2. ไฟล์แบบเรียงลำดับ (sorted file) ข้อดีคือจะค้นหา (search) ข้อมูลได้เร็ว โดยเฉพาะการค้นหาข้อมูลเฉพาะช่วง (range search) เพราะจัดเรียงข้อมูลไว้แล้ว จึงไม่ต้องอ่านทุกเรกคอร์ด ส่วนข้อเสียคือ ทุกครั้งที่เพิ่มหรือแก้ข้อมูล จะต้องจัดเรียงข้อมูลทั้งไฟล์ใหม่เสมอเพื่อให้ลำดับการเรียงถูกต้อง
3. ดัชนี (index) คือการสร้างดัชนีแยกไว้ต่างหากจากข้อมูลในตาราง ซึ่งอาจสร้างดัชนีหลายชุด ส่งผลให้การค้นหาทำได้รวดเร็ว และเมื่อเพิ่มหรืออัปเดตเรกคอร์ดก็จะทำงานเร็วกว่าแบบไฟล์เรียงลำดับมาก เพราะอัปเดตเฉพาะดัชนี ไม่ใช่ทั้งไฟล์ข้อมูล โครงสร้างดัชนีมีแบบต้นไม้ (tree) กับแบบแฮช (hash)

9.4 **ดัชนี**

ตัวอย่างของดัชนีในชีวิตจริงที่ไม่ใช่ในระบบฐานข้อมูล ขอให้นึกถึงสารบัญหนังสือ ที่ชี้ไปยังหน้าที่เราสนใจ หรือดัชนีท้ายเล่มที่มีไว้ค้นหาเนื้อหาตามคำที่สนใจ

ดัชนีในฐานข้อมูลก็ใช้หลักการเดียวกัน คือมีการสร้างดัชนีไว้นอกเหนือข้อมูลจริง เพื่อให้เข้าถึงข้อมูลที่ต้องการได้รวดเร็วขึ้น โดยไม่จำเป็นจะต้องค้นหาข้อมูลทั้งตาราง ฐานข้อมูลสามารถจะมีดัชนีได้หลายชุดตามความต้องการด้วย

อย่างไรก็ตาม เราต้องเข้าใจภาระที่ตามมาจากการสร้างดัชนีด้วย คือการทำดัชนีต้องใช้พื้นที่เพิ่มขึ้น และเมื่อเกิดโอเปอเรชันใด ๆ กับข้อมูล ไม่ว่า UPDATE, INSERT หรือ DELETE ระบบการจัดการฐานข้อมูลก็จะคำนวณ

และจัดเรียงดัชนีใหม่ทุกครั้ง ทำให้สิ้นเปลืองเวลาในการประมวลผลด้วย ดังนั้น เราจึงต้องออกแบบการจัดเก็บและดัชนี และปรับปรุงให้เหมาะสมอยู่เสมอ

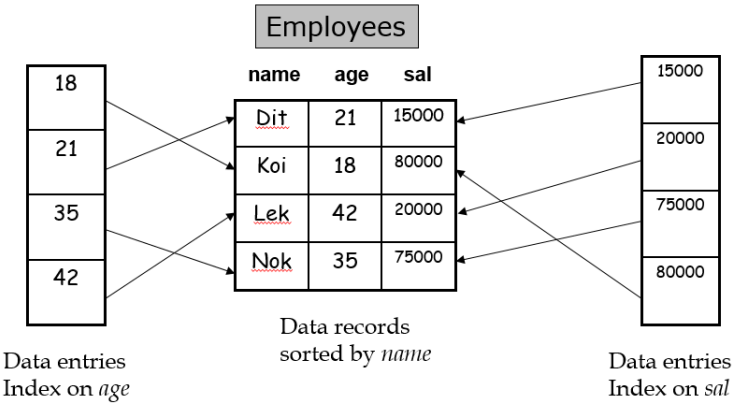
9.5 **คีย์ค้นหาและรายการข้อมูล**

เมื่อใช้ดัชนี เราสามารถค้นหาข้อมูลด้วย คีย์ค้นหา (search key) โดยระบุฟิลด์ใดฟิลด์หนึ่งในตารางให้เป็นคีย์ค้นหา ศัพท์คำว่า “คีย์ค้นหา” นี้ ไม่เกี่ยวข้องกับคีย์ในบทอื่นที่ผ่านมา อาจสับสนได้ คือคีย์ค้นหาไม่มีคุณสมบัติที่ทำให้ทูเพิลหรือเรกคอร์ดไม่ซ้ำกัน คีย์ค้นหาคือฟิลด์ที่ผู้ออกแบบฐานข้อมูลเลือกระบุว่าเป็นฟิล์ดที่ใช้ค้นคืนข้อมูลบ่อย และต้องการสร้างดัชนีสำหรับคีย์ค้นหานั้น

ให้แทนคีย์ค้นหาด้วย k

ผลการค้นหาด้วย k จะได้รายการข้อมูล (data entries) k* ซึ่งหมายถึงข้อมูลของคีย์ค้นหาที่นำมาทำดัชนี

รูปที่ 108 แสดงตัวอย่างรีเลชัน Employees ประกอบด้วย 3 ฟิลด์คือ name, age และ salary และเรียงตาม name จากน้อยไปมาก จะเห็นว่ามี data entries k* 2 ชุดที่สร้างขึ้นมาเป็นดัชนี ได้แก่ data entries ที่ดัชนีด้วย k=age กับ data entries ที่ดัชนีด้วย k=sal ซึ่ง k* แต่ละชุดจะจัดเรียงข้อมูลตาม k และเก็บพอยน์เตอร์ ชี้ไปยังที่อยู่ของข้อมูลนั้น ซึ่งในรูปแสดงด้วยลูกศร



รูปที่ 108 รีเลชัน Employees

การเรียงข้อมูลจะทำทีละฟิลด์ หากกำหนดไว้หลายฟิลด์ ก็จะเรียงข้อมูลในฟิลด์แรกก่อนแล้วค่อยเรียงฟิลด์ต่อ ๆ มา

9.6 วิธีการสร้างรายการข้อมูล

การสร้างรายการข้อมูล หรือ data entries มี 3 วิธี

9.6.1 วิธีที่ 1 โครงสร้างไฟล์แบบดัชนี

ให้นำข้อมูลจริงทุกเรกคอร์ดมาสร้างดัชนี หรือพูดว่า k* คือข้อมูลจริงทั้งหมดมาจัดเรียงตามคีย์ค้นหา k ที่กำหนด นั่นหมายความว่า k* ก็จะมีได้ชุดเดียว เพราะเราเรียงข้อมูลจริงได้เพียงแบบเดียว หากมีเรกคอร์ดจำนวนมากจะส่งผลให้จำนวนเพจใน k* สูงตามไปด้วย เราเรียกชื่อวิธีที่ 1 นี้ว่า โครงสร้างไฟล์แบบดัชนี (indexed file organization)

9.6.2 วิธีที่ 2 คู่ <k,rid>

ให้สร้างรายการข้อมูล k* ขึ้น โดยกำหนดคีย์ค้นหาและรหัสเรกคอร์ด <k,rid> ที่ชี้ไปยังเรกคอร์ดในข้อมูลจริง ดังนั้น k* จะมีคู่ <k,rid> สำหรับทุก k ในตารางข้อมูลจริง จะเห็นว่าวิธีนี้เราจะไม่ยุ่งเกี่ยวกับข้อมูลที่อยู่ในไฟล์จริง มีเพียงพอยน์เตอร์จาก rid ชี้ไปยังเรกคอร์ดในข้อมูลจริง

9.6.3 วิธีที่ 3 <k,rid-list>

คล้ายวิธีที่ 2 แต่แทนที่จะสร้าง k กับ rid เป็นคู่ ก็ให้สร้างเป็นรายการหรือ list <k,rid-list> นั่นหมายถึง k* วิธีที่ 3 นี้ สำหรับ k เดียวกัน อาจมี rid หลายค่าได้ จึงไม่ปรากฏเป็นคู่เหมือนวิธีที่ 2 ดังนั้น วิธีนี้จะเหมาะกับ k ที่ปรากฏอยู่หลายทูเพิลในตารางข้อมูลจริง

สรุปคือ วิธีที่ 2 และวิธีที่ 3 จะได้ k* ที่อิสระจากชุดข้อมูลจริง วิธีที่ 3 จะประหยัดพื้นที่กว่าวิธีที่ 2

9.7 ประเภทของดัชนี

เราสามารถแบ่งดัชนีได้หลายประเภท ขึ้นอยู่กับมุมมอง

หากพิจารณาจากคีย์ค้นหา จะแบ่งดัชนีได้ 2 ประเภท คือ

4. ดัชนีหลัก (primary index) หมายถึงคีย์ค้นหานั้นคือคีย์หลักหรือส่วนหนึ่งของคีย์หลักของรีเลชัน

5. ดัชนีรอง (secondary index) หมายถึงคีย์ค้นหานั้นไม่ใช่คีย์หลักของรีเลชัน รวมทั้งคีย์คู่แข่งด้วย

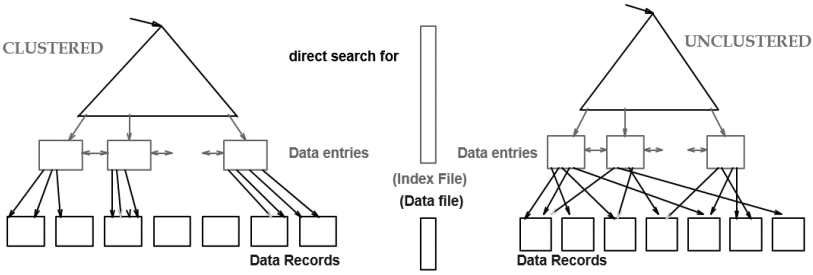
หากพิจารณาจากการเรียงของข้อมูลในดัชนี จะแบ่งดัชนีได้ 2 ประเภท คือ

6. ดัชนีคลัสเตอร์ (clustered index) หมายถึง ดัชนีที่รายการข้อมูล k* มีลำดับการเรียงตัวเหมือนหรือคล้ายกันกับเรกคอร์ดในรีเลชัน
7. ดัชนีไม่คลัสเตอร์ (unclustered index)

จะเห็นได้ว่า การสร้างรายการข้อมูลด้วยวิธีที่ 1 คือแบบโครงสร้างไฟล์แบบดัชนีนั้น จะอนุมานได้ว่าเป็นดัชนีแบบคลัสเตอร์แน่นอน

เวลาที่ใช้ในการค้นคืนข้อมูลอาจต่างกันได้มากเมื่อดัชนีเป็นแบบคลัสเตอร์หรือไม่คลัสเตอร์

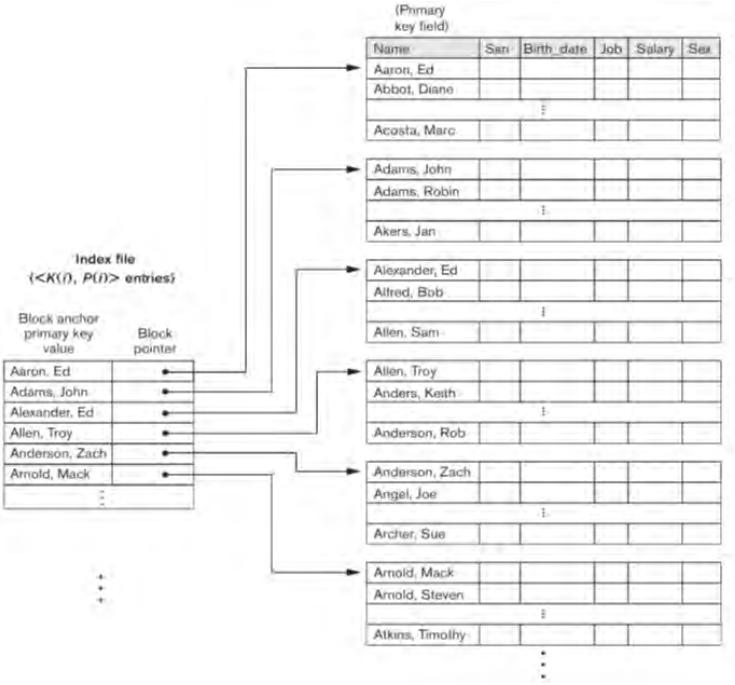
รูปที่ 109 เปรียบเทียบดัชนีคลัสเตอร์กับดัชนีไม่คลัสเตอร์



รูปที่ 109 ดัชนีคลัสเตอร์กับดัชนีไม่คลัสเตอร์

ตัวอย่าง

รูปที่ 110 จาก [2] แสดงตัวอย่างดัชนีหลัก หมายถึง k ของดัชนีนี้ คือคีย์หลักของรีเลชัน สังเกตว่า Name เป็นคีย์หลักของรีเลชัน ดังนั้น k ก็คือ Name จากนั้นให้สังเกตว่ารายการข้อมูล k* ซึ่งประกอบด้วย <name, rid> นั้น ก็เรียงตาม name จึงทำให้ดัชนีเป็นแบบคลัสเตอร์โดยปริยาย



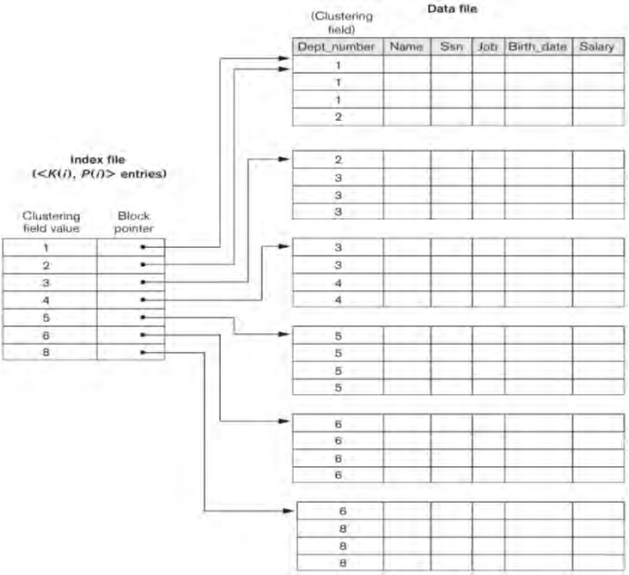
รูปที่ 110 ดัชนีหลัก

สังเกตรูปที่ 110 ต่อ จะเห็นว่า name ใน k* นั้น สัมพันธ์กับข้อมูลแถวแรกในเพจของเรกคอร์ดข้อมูล ดัชนีแบบนี้เรียกว่า ดัชนีเบาบาง (sparse index)

ที่ k* จะเก็บเฉพาะเรกคอร์ดแรกของแต่ละเพจเท่านั้น เมื่อค้นหาข้อมูล ก็จะคำนวณหาที่อยู่ที่ต้นเพจแล้วค่อยหาในเพจต่ออีกที

ดัชนีที่ตรงข้ามกับดัชนีเบาบาง เรียกว่า ดัชนีหนาแน่น (dense index) ซึ่ง k* ที่เป็นดัชนีหลักนั้น จะต้องมีจำนวนเท่ากับจำนวนเรกคอร์ดในรีเลชัน แล้วมี rid ชี้ไปยังทุกเรกคอร์ดที่ k มีค่าเท่ากัน ดังนั้นจะมีจำนวนคู่ <k,rid> หรือ k* จึงมีมากกว่าดัชนีเบาบางมาก แต่จะส่งผลให้ค้นหาข้อมูลได้เร็วกว่ามากเช่นกัน ข้อเสียเมื่อเทียบกับดัชนีเบาบางก็คือ ดัชนีแน่นหนาจะใช้พื้นที่จัดเก็บใหญ่มาก

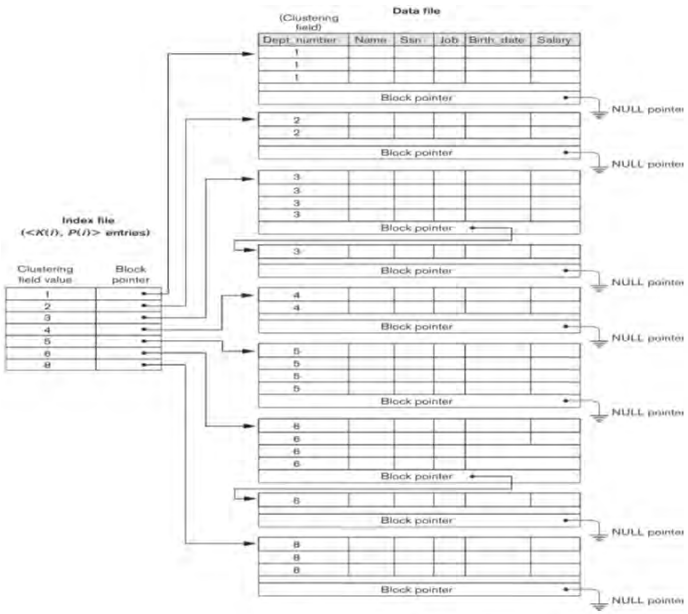
ตัวอย่าง



รูปที่ 111 ดัชนีคลัสเตอร์

รูปที่ 111 จาก [2] แสดงตัวอย่างดัชนีคลัสเตอร์ แต่ไม่ใช่แบบดัชนีหลัก จะเห็นว่าลำดับของเรกคอร์ดในรีเลชันเรียงตาม Dept_number แล้วสร้างดัชนี ให้ k คือ Dept_number จะเห็นว่าเป็นดัชนีเบาบางอีกเช่นกัน เมื่อต้องการค้นหาคนที่อยู่ Dept_number 1 จะมีบล็อกพอยน์เตอร์ชี้ไปยังเรกคอร์ดแรกของเพจที่ Dept_number = 1 เมื่อ เมื่อต้องการค้นหาคนที่อยู่ Dept_number 2 บล็อกพอยน์เตอร์ก็ชี้ไปยังเรกคอร์ดแรกของเพจที่ Dept_number = 2 ซึ่งก็ยังเป็นเพจแรก เพราะเรกคอร์ดแรกที่ Dept_number = 2 อยู่ในเพจนั้น ดังนั้น ดัชนีนี้จึงเป็นแบบเบาบาง

ตัวอย่าง



รูปที่ 112 ดัชนีคลัสเตอร์

ในรูปที่ 112 จาก [2] จะเห็นว่ามีการใช้ดัชนีคล้ายตัวอย่างก่อนหน้า ซึ่ง k คือ Dept_number เช่นเดียวกัน เป็นดัชนีคลัสเตอร์เหมือนกัน ส่วนที่ต่างคือ แต่ละเพจนั้นจะเก็บเรกคอร์ดที่มีค่าเท่ากัน เช่นเพจแรกสุดเก็บเร็คคอร์ด Dept_number 1 เพจต่อมาเก็บเร็คคอร์ด Dept_number 2 ไปเรื่อย ๆ ถ้าเพจนั้นเต็มแล้วยังเก็บข้อมูลไม่พอ หมายถึงยังมี Dept_number เท่าเดิมคงเหลืออยู่ ก็ให้โอเวอร์โฟลลงไปเพจถัดไป แล้วสร้างพอยน์เตอร์ชี้ต่อไปยังเพจเพิ่มเติมนี้ด้วย เมื่อ Dept_number มีค่าต่างไป ก็ให้ขึ้นเพจใหม่และยอมปล่อยที่ว่างในเพจนั้น พร้อมกับเซตนัลพอยน์เตอร์ไว้ที่ท้ายเพจ วิธีดัชนีแบบนี้ส่งผลให้การค้นหาข้อมูลทำได้เร็วขึ้น จากนั้นบล็อกพอยน์เตอร์จะชี้ตรงไปยังต้นเพจที่มี Dept_number ที่ต้องการค้นหาได้ทันที แต่ใช้พื้นที่มากกว่า

9.8 โครงสร้างของดัชนี

ที่ผ่านมาเราเห็นตัวอย่างของดัชนีแบบคลัสเตอร์ ซึ่งตัวดัชนีจะเก็บคีย์ค้นหา k กับ rid แต่ถ้าเราเลือกทำดัชนีของรีเลชันแบบไม่คลัสเตอร์แล้ว เราจะต้องเลือกโครงสร้างของตัวดัชนี คือรายการข้อมูล k* ด้วย ซึ่งตำรานี้จะกล่าวถึงโครงสร้างของดัชนี 2 แบบ คือ

1. โครงสร้างดัชนีแบบแฮช
2. โครงสร้างดัชนีแบบต้นไม้

9.8.1 โครงสร้างดัชนีแบบแฮช

โครงสร้างดัชนีแบบแฮช (hash-based index) เป็นการจัดเรียงข้อมูลโดยแยกเป็นถัง (bucket) แล้วแฮชรายการข้อมูล k* ค่าด้วยคีย์ค้นหา k ดัชนีก็

จะเก็บรายการถังซึ่งประกอบด้วย เพจหลัก (primary page) และพอยน์เตอร์ที่เชื่อมไปยังโอเวอร์โฟลวเพจถ้ามี

การจะหาว่าเรกคอร์ดที่ต้องการนั้นอยู่ที่ถังใด ให้คำนวณด้วยฟังก์ชันแฮช (Hashing function)

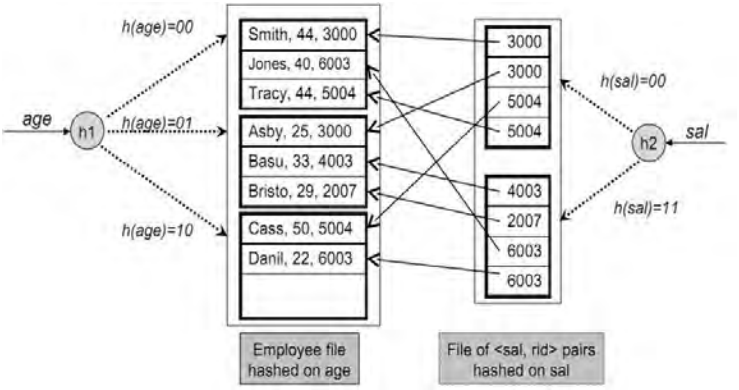
h: h(*r*)

จะได้หมายเลขของถังที่เก็บเรกคอร์ด *r*

ในกรณีที่การสร้างรายการข้อมูลใช้วิธีที่ 1 คือใช้โครงสร้างไฟล์แบบดัชนีนั้น ในถังก็จะเก็บเรกคอร์ดข้อมูลจริง ถ้าใช้วิธีคู่ <k,rid> ข้อมูลในถังก็คือ คู่ <k,rid> ถ้าใช้วิธีคู่ <k,rid-list> ข้อมูลในถังก็คือ คู่ <k,rid-list>

วิธีนี้เหมาะกับการค้นหาข้อมูลแบบเท่ากับ (equality search) จะทำได้เร็ว เพราะเราใช้โอเปเรชันเดียว แฮชคีย์ค้นหา แล้วจะทราบค่าถังของข้อมูลนั้นทันที

อย่างไรก็ดี ดัชนีแบบแฮชไม่เหมาะกับการค้นหาข้อมูลแบบส่วน (range search) อย่างยิ่ง เพราะผลลัพธ์ของการแฮชค่าที่ใกล้เคียงกันนั้น อาจทำให้ข้อมูลอยู่ห่างกันก็ได้



รูปที่ 113 ฟังก์ชันแฮช

รูปที่ 113 จาก [2] แสดงการใช้ฟังก์ชันแฮชเพื่อเก็บข้อมูล โดยแฮช 2 คีย์ค้นหา คือแฮช age แล้วตามด้วยแฮช sal

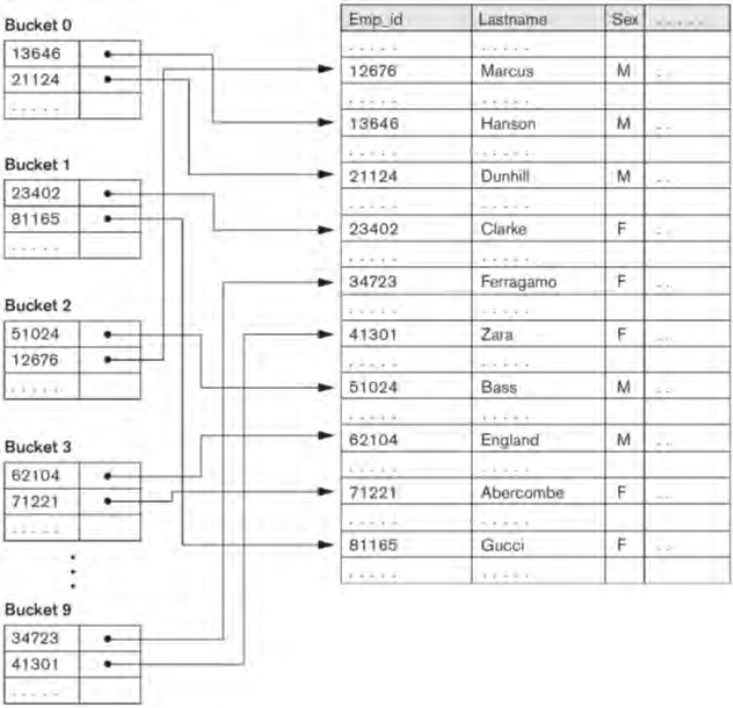
h1: h(*age*)

h2: h(*sal*)

ในการแฮช age นั้น ตัวอย่างนี้แฮชโดยการแปลง age เป็นเลขไบนารีแล้วตัดเลขมา 2 หลักสุดท้าย เช่น อายุ = 44 หรือ 40 จะได้ไบนารี = 00 (x mod 4 = 0) แล้วนำ age ใด ๆ ที่ได้ค่าแฮช = 00 ไปเก็บไว้ในถังเดียวกัน ต่อมาจะเห็นว่า age = 25, 33, 29 จะแฮชได้ 01 จึงนำข้อมูลที่แฮชแล้วได้ 01 ทั้งหมดไปไว้ในถังเดียวกัน และทำอย่างนี้ไปเรื่อย ๆ จะเห็นว่าข้อมูลในแต่ละถังนั้นจะได้ age ที่กระจายกันไป ซึ่งจะส่งผลให้การกระจายตัวของจำนวนข้อมูลในเพจแต่ละเพจ ส่วนการแทรกข้อมูลเข้าไปในถังนั้น จะใช้วิธีแอปเพนด์ (append) คือผนวกข้อมูลนั้นต่อท้ายโดยไม่ต้องจัดเรียงใด ๆ

เมื่อต้องการค้นหาข้อมูล ก็ให้คำนวณว่าข้อมูลอยู่ที่ถังใด แล้วเข้าไปค้นหาในถังอีกที

มาลองแฮช sal กันบ้าง จากตัวอย่างจะเห็นว่าเราใช้ฟังก์ชันแฮชเดียวกับที่ใช้กับ age ได้ผลการแฮชที่เป็นได้ 2 ค่า คือ 00 กับ 01 จึงจัดข้อมูลในถังตามผลแฮช



รูปที่ 114 ดัชนีหลัก

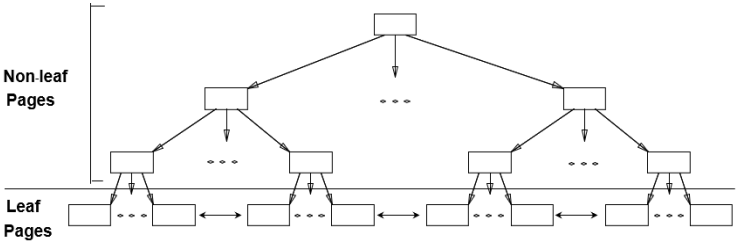
การแฮชส่วนมากแล้วจะเป็นดัชนีรอง (secondary index) แต่ตัวอย่างในรูปที่ 114 จาก [2] นี้จะเห็นว่าเรกคอร์ดในรีเลชันเรียงตาม Emp_id ซึ่งเป็นคีย์หลัก ดัชนีในกรณีนี้จึงเป็นดัชนีหลัก ดังนั้นเมื่อเราแฮช Emp_id จะทำให้ Emp_id ที่เดิมเรียงอยู่ติดกันกลับกระจายตัวออกด้วยผลของการแฮช ผลก็จะได้ดัชนีแบบไม่คลัสเตอร์

9.8.2 โครงสร้างดัชนีแบบต้นไม้

โครงสร้างดัชนีแบบต้นไม้ (tree-based index) ที่ตำรานี้จะครอบคลุมคือต้นไม้บีพลัส (B+ tree) ซึ่งต่างจากโครงสร้างแบบแฮชอย่างมาก เพราะรายการข้อมูล k* ของต้นไม้บีพลัสจะเรียงกันตามคีย์ค้นหาอยู่ก่อนแล้ว จากนั้นเราจะทำโครงสร้างต้นไม้ครอบรายการข้อมูลนี้อีกชั้น

โครงสร้างแบบต้นไม้บีพลัสนี้ จะมีความสมดุลอยู่เสมอ หมายถึงความสูงของต้นไม้จากราก (root) ถึงใบ (leaf) นั้นจะมีความยาวเท่ากันเสมอ คือเมื่อเพจนั้น ๆ เต็ม ก็จะแบ่งเพจ (split) แล้วจัดเรียงไฟล์ใหม่ (reorganize)

ต่อไป ขอทับศัพท์ ลีฟ (leaf) ดังนั้น เพจใบ จะเรียกว่า ลีฟเพจ (leaf page)

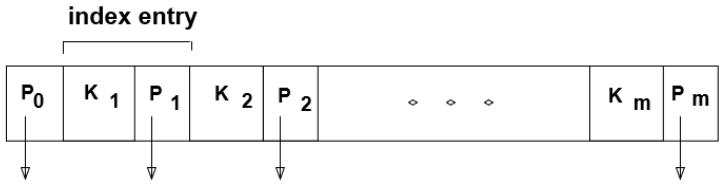


รูปที่ 115 โครงสร้างแบบต้นไม้บีพลัส

รูปที่ 115 จาก [2] แสดงอธิบายความสัมพันธ์ของลีฟเพจกับนอนลีฟเพจ (non-leaf page) ในโครงสร้างต้นไม้บีพลัส

ข้อมูลในลีฟเพจอาจเป็นรายการข้อมูล (data entries) k* ที่มีการโยงถึงกัน (chained) ในกรณีที่การสร้างรายการข้อมูลเป็นวิธีที่ 1 คือโครงสร้างไฟล์แบบดัชนี ข้อมูลตรงลีฟเพจจะเป็นเรกคอร์ดจริง

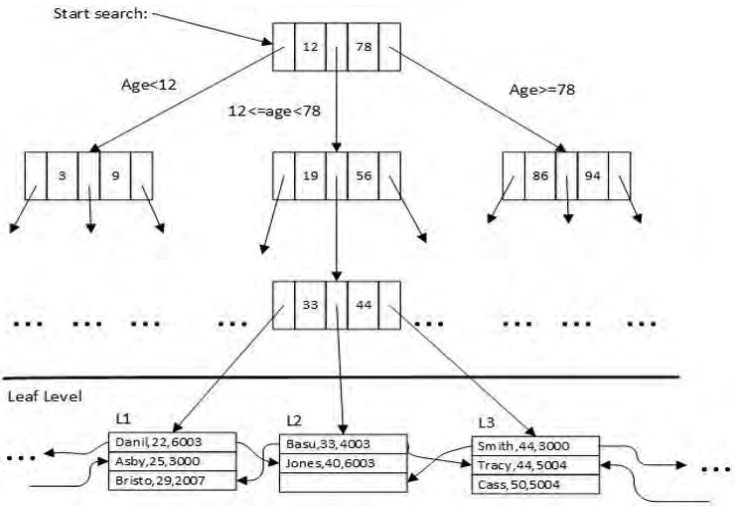
ส่วนข้อมูลที่อยู่ในนอนลีฟเพจนั้นก็คือรายการดัชนี (index entries) ดังแสดงในรูปที่ 116 ประกอบด้วยพอยน์เตอร์คั่นด้วยคีย์ค้นหา k เพื่อให้ชี้ไปยังลีฟเพจที่ถูกต้อง พอยน์เตอร์ชี้ลงไปยังชั้นถัดไปด้านซ้ายเพื่อนำไปยังรายการข้อมูลมีค่าน้อยกว่า k และไปด้านขวาเมื่อรายการข้อมูลมากกว่า k



รูปที่ 116 รายการดัชนี

รายการดัชนีแต่ละเพจ อาจมีขนาดใหญ่มากได้ อาจมากกว่า 100 รายการได้ ในทางปฏิบัติใช้กันอยู่ที่ 50-100 รายการ และมีพอยน์เตอร์จำนวนมากตามไปด้วย ในรูปแสดงด้วยลูกศร ทำให้สามารถรองรับเรกคอร์ดได้จำนวนมหาศาลโดยที่ความสูงของต้นไม้มีค่าน้อยได้ เช่นสามารถรองรับเรกคอร์ดจำนวนเป็นหลักล้านได้ด้วยความสูงของต้นไม้ไม่เกิน 4 ชั้น

คุณสมบัติสำคัญของต้นไม้บีพลัสคือจะมีฟิลแฟคเตอร์ (fill factor) อยู่ที่แต่ละเพจ เพื่อรับรองจำนวนข้อมูลในเพจนั้น ๆ ให้มีมากกว่าที่ผู้ใช้กำหนด (user-defined) เช่น กำหนดว่าให้มีข้อมูลไม่ต่ำกว่า 50% ของเพจ เป็นต้น หากจำนวนข้อมูลน้อยกว่าที่กำหนด ก็จะรวมเพจเข้าด้วยกัน



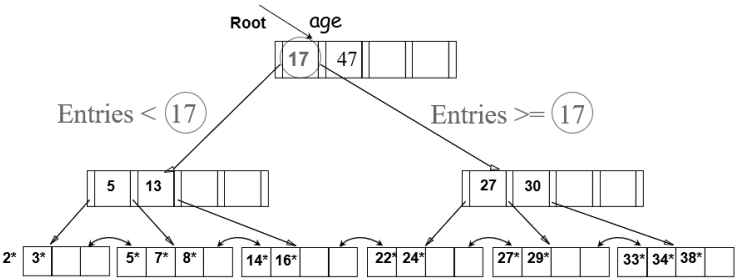
รูปที่ 117 การใช้ต้นไม้บีพลัส ดัชนีด้วย age

รูปที่ 117 จาก [2] แสดงตัวอย่างการใช้ต้นไม้บีพลัส ดัชนีด้วย age เริ่มจากรากที่มีช่องเก็บค่าคีย์ค้นหาได้ 2 ค่า สมมุติว่าเป็นค่าอายุ (age) จะเห็นว่าที่รากนั้นเก็บค่า 12 กับ 78 และมีพอยน์เตอร์ชี้ไปยังชั้นถัดไป ถ้าชี้ไปทางซ้ายหมายความว่าข้อมูลในเพจนั้นมีค่าน้อยกว่า 12 ไปด้านขวาหมายความว่ามากกว่า 78 ไปเรื่อย ๆ ดังนี้ จนกว่าจะถึงชั้นลีฟเพจ ก็จะพบรายการข้อมูลที่ต้องการ หากรายการนั้นมีอยู่ในฐานข้อมูล แต่ถ้าไม่พบรายการที่ต้องหา ณ เพจที่ควรจะอยู่ ก็สรุปได้ว่าเรกคอร์ดนั้นไม่ปรากฏในรีเลชัน คือไม่มีความ

จำเป็นใด ๆ ที่จะค้นหาข้อมูลในเพจอื่นต่อไปเพราะข้อมูลถูกจัดเรียงไว้อยู่แล้ว

สังเกตพอยน์เตอร์ในระดับลีฟนั้นจะมีทั้งพอยน์เตอร์ที่ชี้ไปยังเพจถัดไปและชี้กลับไปยังเพจก่อนหน้าด้วย

ตัวอย่าง



รูปที่ 118 ต้นไม้บีพลัส ให้ age เป็น k

รูปที่ 118 แสดงตัวอย่างต้นไม้บีพลัสทรี ให้ age เป็น k สังเกต * ที่ชั้นลีฟหรือชั้นล่างสุด นั่นแสดงว่าข้อมูลชั้นนี้คือรายการข้อมูล

สมมุติว่าเราต้องการค้นหาคนที่อายุ 28 คือ k* = 28 ให้เริ่มดูที่ราก จะพบว่า 28 มากกว่า 17 แต่น้อยกว่า 47 จึงตามต้นไม้ลงมาชั้นถัดไปด้านขวา ซึ่ง k = 27, 30, ... ก็ให้ตามต้นไม้ลงมาชั้นถัดไปด้านขวาอีกครั้ง เพราะ 28 มากกว่า 27 แต่น้อยกว่า 30 จึงพบ k* = 27* ตามด้วย 29 ก็ให้สรุปได้ทันทีว่าไม่มี k* ที่มีค่าเท่ากับ 28*

สมมุติว่าเราจะค้นหาช่วงของข้อมูล (range search) เช่น ต้องการค้นหาคนที่อายุมากกว่า 15 หรือ k* > 15 ก็เริ่มจากรากเช่นเดียวกัน แล้วตามต้นไม้ชั้นถัดไปด้านซ้าย เนื่องจาก 15 < 17 แล้วแยกต่อลงไปด้านขวาของ 13 พบชั้นลีฟ มี k* = 16* ก็จะเริ่มรู้คำตอบว่า k* ตัวแรกคือ 16 แล้วให้ตามพอยน์เตอร์ไปจนจบไฟล์ (eof) ถือเป็นคำตอบ เพราะข้อมูลจัดเรียงไว้เรียบร้อยแล้ว

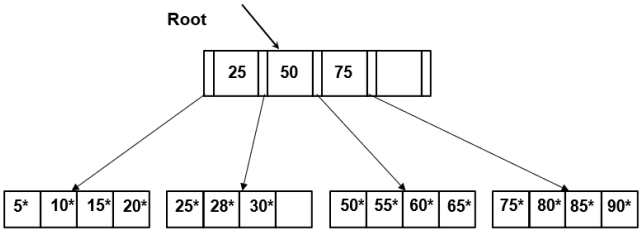
สมมุติว่าเราจะค้นหาช่วงของข้อมูล (range search) เช่น ต้องการค้นหาคนที่อายุมากกว่า 15 แต่น้อยกว่า 30 หรือ 15 < k* < 30 ก็ให้ใช้วิธีเดิมหาจุดเริ่มของ k* = 15* และ 30* แล้วกวาดคำตอบพี่พบตามพอยน์เตอร์ของทั้งสองค่านั่นเอง

จะเห็นว่าวิธีต้นไม้บีพลัสนี้ เหมาะกับการค้นหาช่วงของข้อมูล (range search) อย่างยิ่ง

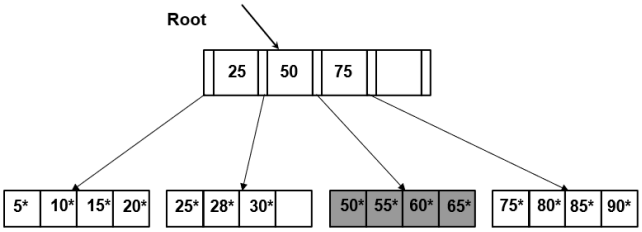
คุณสมบัติอีกอย่างของต้นไม้บีพลัสคือมีแฟนเอาต์ (fan-out) หรือจำนวนเฉลี่ยของพอยน์เตอร์ต่อโหนด (node) ในชั้นนอนลีฟ จำนวนมาก เมื่อเทียบกับต้นไม้ไบนารีที่มีค่าแฟนเอาต์เท่ากับ 2

9.9 การปรับความสมดุลของต้นไม้บีพลัส

ทุกครั้งมีการอัปเดต ลบ หรือแทรกข้อมูลลงในฐานข้อมูลที่ใช้ดัชนี จะส่งผลให้ต้องจัดเรียงดัชนีใหม่ เนื่องจากคุณสมบัติของต้นไม้บีพลัสกำหนดว่าต้องมีความสมดุลเสมอ เราจึงต้องมีอัลกอริทึมในการปรับความสมดุลของต้นไม้บีพลัส ให้ดูตัวอย่างการแทรกข้อมูลในรูปที่ 119



INSERT 70*

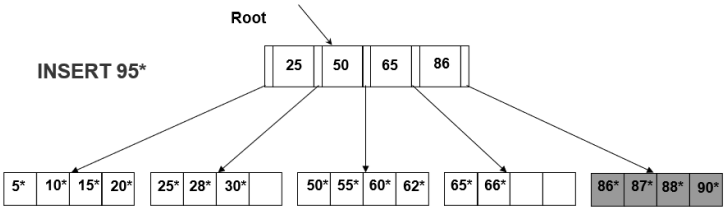


รูปที่ 119 แทรกข้อมูล age* = 70*

สมมุติว่าเราจะแทรกข้อมูล age* = 70* ให้เริ่มจากราก พบว่าจะต้องเพิ่มข้อมูลที่เพจที่ 3 ของลีฟเพจ คือเพจที่มี {50*, 55*, 60*, 65*} ก็พบว่าเพจนี้เต็มแล้ว จึงให้แบ่งโหนด (split node) ที่เต็มนั้นออกจากกัน โดยคำนึงถึงค่าฟิลแฟคเตอร์ที่กำหนด สมมุติกำหนดเป็น 50% คือทุกโหนดต้องถูกใช้อย่างน้อย 50% จึงแบ่งได้เป็น {50*, 55*, ...} กับ {60*, 65*, ...} เพราะ 50% ของช่องสำหรับแต่ละโหนดในรูปนี้คือ 4 คีย์ค้นหา หรือ 5 พอยน์เตอร์ จากนั้นก็สามารถแทรก k* = 70* จะได้ {50*, 55*, ...} กับ {60*, 65*, 70*, ...} สุดท้าย เราจะต้องปรับดัชนีของโหนดที่ชี้มายังลีฟโหนดที่ถูกแบ่งด้วย ซึ่งเดิมนั้น โหนดที่ชี้มามีค่า {25, 50, 75, ...} จึงอาจเลือกค่าใด ๆ ระหว่าง 55 กับ 60 ก็ได้ แต่วิธีที่แนะนำคือให้เลือกกำหนดเป็น {25, 50, 60, 75} หมายถึงให้ใช้ค่าน้อยที่สุดจากโหนดที่ถูกแบ่งมา แต่จะเห็นได้ว่าการที่เพิ่ม 60 ไปใน

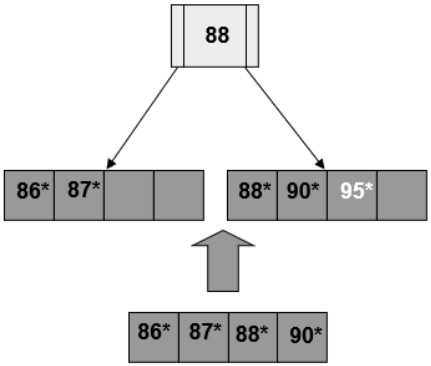
ดัชนีนี้ เราจำเป็นต้องจัดเรียงให้ลำดับถูกต้อง จึงต้องเลื่อน 75 ออกไปก่อนที่จะแทรก 60 แล้วด้วยการย้ายพอยน์เตอร์ให้สอดคล้องกัน

ตัวอย่าง



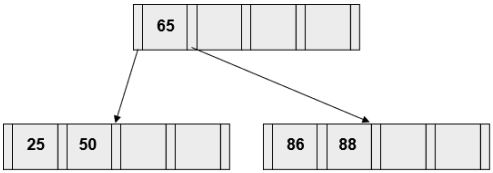
รูปที่ 120 แทรกข้อมูล age* = 95*

ตัวอย่างจากรูปที่ 120 นี้จะแสดงเหตุการณ์เมื่อจำเป็นต้องเพิ่มชั้นของต้นไม้ ในกรณีนี้ สมมุติว่าเราอยากแทรกข้อมูล age* = 95* ซึ่งพบว่า ต้องเพิ่มที่ลีฟเพจสุดท้าย แต่เพจเต็ม ก็ให้แบ่งเพจนั้น ดังรูปที่ 121



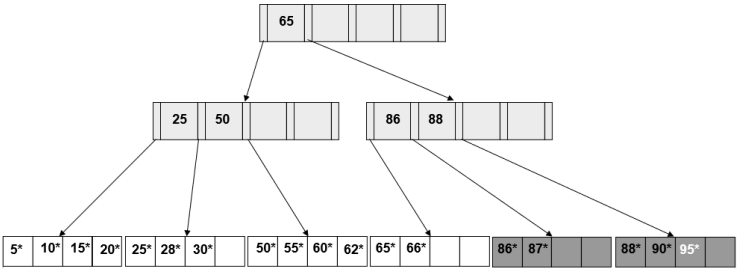
รูปที่ 121 เพิ่มชั้นของต้นไม้

หลังจากนั้น ให้เพิ่ม 88 ซึ่งคือค่าน้อยที่สุดของเพจใหม่ ไปยังโหนดที่ชี้มาเพจที่ 88 อยู่ ซึ่งก็พบว่าเต็ม จึงต้องแบ่งอีก ดังรูปที่ 122



รูปที่ 122 โหนดเพิ่มชั้น

จะเห็นว่าขั้นตอนนี้ทำให้เกิดโหนดเพิ่มชั้น เพราะรากถูกแบ่ง และในกรณีนี้การกำหนดค่าให้พอยน์เตอร์ก็จะทำต่างจากวิธีเดิม สังเกตว่า ถ้าเป็นวิธีเดิมนั้น k แรกของโหนดด้านขวาจะมีค่า 65 เพราะเราแบ่ง {25, 50, 65, 86} ออกจากกัน แล้วควรจะได้ {25, 50, ...} {65, 86, ...} แต่ถ้าเป็นกรณีการเกิดชั้นเพิ่ม คือรากถูกแบ่ง เราจะผลักค่า k น้อยสุดจากโหนดที่ถูกแบ่งขึ้นไปอยู่ชั้นต้นไม้ที่สูงขึ้นนั้นได้ทันที ผลลัพธ์จึงได้อย่างที่แสดงในรูปที่ 123



รูปที่ 123 ต้นไม้บีพลัสสมดุล

ด้วยวิธีนี้ ต้นไม้บีพลัสก็จะสมดุลได้ตลอดเวลา อย่างไรก็ดี หากมีการลบข้อมูลไปถึงระดับหนึ่ง ก็มีความเป็นไปได้ว่าโหนดจะถูกเชื่อมเพราะมีจำนวนข้อมูลน้อนกว่าฟิลแฟคเตอร์ และอาจส่งผลให้ชั้นของต้นไม้ลดลงได้ แต่ก็จะยังสมดุลเสมอ

9.10 **การคำนวณเวลา**

การจัดข้อมูลและการสร้างดัชนีแต่ละแบบ จะส่งผลให้การทำงานของฐานข้อมูลต่างไปโดยเฉพาะในด้านของความเร็ว เราจะวิเคราะห์ความสิ้นเปลือง (cost) หรือเวลา ในแง่เพจไอโอแบบเฉลี่ย และจะไม่นำเวลาซีพียูมาคำนวณในขั้นตอนนี้ ตัวแปรที่เราจะนำมาใช้คำนวณเวลามี 3 ตัวแปรคือ

B : จำนวนเพจข้อมูล (data page)
R : จำนวนเรกคอร์ดต่อเพจ
D : เวลาเฉลี่ยในการอ่านหรือเขียนดิสก์

ในการวิเคราะห์แบบเฉลี่ย (average case analysis) นี้จะไม่คำนึงกรณีที่ข้อมูลบางเพจอยากถูกดึงมารอพร้อมใช้ ดังนั้นเพจไอโอก็จะเป็นค่าประมาณเท่านั้น

การจัดโครงสร้างไฟล์ที่เรานำมาพิจารณาคำนวณ มีอยู่ด้วยกัน 5 แบบคือ

1. ไฟล์ฮีป หรือไฟล์แบบไม่เรียงลำดับ หมายถึงข้อมูลในไฟล์ไม่มีการจัดเรียงใด ๆ (random order) การแทรกข้อมูลให้แทรกที่ท้ายไฟล์เสมอ
2. ไฟล์แบบเรียงลำดับ เช่น เรียงลำดับตาม *<age, sal>*

3. ไฟล์แบบคลัสเตอร์ด้วยต้นไม้บีพลัส คือใช้โครงสร้างไฟล์แบบดัชนีและมีคีย์ค้นหา เช่น *<age, sal>*
4. ไฟล์ฮีปแบบไม่คลัสเตอร์และดัชนีแบบต้นไม้บีพลัส ด้วยคีย์ค้นหา เช่น *<age, sal>*
5. ไฟล์ฮีปแบบไม่คลัสเตอร์และดัชนีแบบแฮชด้วยคีย์ค้นหา เช่น *<age, sal>*

โอเปอเรชันที่นำมาคำนวณ มี 5 โอเปอเรชัน คือ

a) สแกน (scan) คือให้ดึงนำเรกคอร์ดทั้งหมดจากดิสก์
b) ค้นหาแบบเท่ากับ (equality search)
c) ค้นหาแบบช่วง (range selection)
d) แทรกเรกคอร์ด (insert a record)
e) ลบเรกคอร์ด (delete a record)

ในการคำนวณนี้ เพื่อให้ไม่ซับซ้อนเกินไป ขอตั้งสมมุติฐานดังนี้

1. ไฟล์ฮีป
    - ค้นหาค่าเท่ากันด้วยคีย์ค้นหา เมื่อค้นพบเพียงตัวแรกก็ให้หยุดนับเวลา
2. ไฟล์แบบเรียงลำดับ
    - ไฟล์จะถูกกระชับหรืออัดแน่น (compact) หลังการลบ เพื่อไม่ให้มีช่องโหว่ในไฟล์

3. ดัชนี
    - สำหรับวิธีที่ 2 และวิธีที่ 3 ให้ขนาดรายการข้อมูลเป็น 10% ของเรกคอร์ด นั่นเป็นเพราะว่าดัชนีควรมีขนาดเล็กกว่าข้อมูลจริงมาก ๆ และไม่ได้ต้องนำมาใช้ทั้งหมด นั่นคือเราเลือกใช้ดัชนีแบบเบาบาง (sparse index)
    - การแฮช เพื่อป้องกันการเกิดโอเวอร์โฟลว ให้ออกแบบไม่ให้มีข้อมูลเต็มเพจตั้งแต่แรก คือให้เหลือเนื้อที่ว่างพอให้แทรกข้อมูลได้ เราเรียกสถานการณ์นี้ว่า อัตราการครอบครองพื้นที่ (occupancy rate) เพื่อให้การคำนวณไม่ซับซ้อน เราจะใช้อัตราการครอบครองพื้นที่ที่ 80% ซึ่งจะส่งผลให้ไฟล์ (file size) มีขนาด 1.25 เท่าของขนาดข้อมูล (data size) ซึ่งคิดมาจากว่าขนาดไฟล์ต้องมีขนาดเท่าใดจึงจะรองรับข้อมูลได้ทั้งหมด ก็คือขนาดไฟล์ = 100 x 100 / 80 = 1.25
    - เมื่อใช้ต้นไม้ ก็ต้องคำนึงถึงฟิลแฟคเตอร์ ว่าแต่ละเพจควรใช้พื้นที่อย่างน้อยเท่าใด หรือมีอัตราการครอบครองพื้นที่เท่าใด ในกรณีนี้ เราจะใช้อัตราการครอบครองพื้นที่ที่ 2/3 หรือ 67% ดังนั้นไฟล์จะมีขนาดเท่ากับ 1.5 เท่าของขนาดข้อมูล หรือมองว่าขนาดไฟล์ = 3/2 หรือ 1.5 เท่าของขนาดข้อมูล

ตารางที่ 4 แสดงผลการวิเคราะห์เวลาที่ใช้สำหรับแต่ละโอเปอเรชัน a, b, c, d, e ในการทำงานกับไฟล์ที่มีโครงสร้างต่างกัน 5 แบบ 1, 2, 3, 4, 5

ตารางที่ 4 ผลการวิเคราะห์เวลา

	(a) Scan	(b) Equality	(c) Range	(d) Insert	(e) Delete
(1) Heap	BD	0.5BD	BD	2D	Search +D
(2) Sorted	BD	Dlog 2B	Dlog 2 B + # matches	Search + BD	Search +BD
(3) Clustered	1.5BD	Dlog F 1.5B	Dlog F 1.5B + # matches	Search + D	Search +D
(4) Unclustered Tree index	BD(R+0.15)	D(1 + log F 0.15B)	Dlog F 0.15B + # matches	D(3 + log F 0.15B)	Search + 2D
(5) Unclustered Hash index	BD(R+0.125)	2D	BD	4D	Search + 2D

1(a) การสแกนไฟล์ฮีป สำหรับไฟล์ฮีปนั้น เนื่องจากไม่มีการเรียงข้อมูลใด ๆ การสแกนจึงต้องอ่านข้อมูลทั้งหมด คือ B เพจ แต่ละเพจใช้เวลาอ่านเท่ากับ D ดังนั้น การสแกนไฟล์ฮีปจึงใช้เวลา BD ที่ไม่ต้องคำนึงถึง R เป็นเพราะเราต้องอ่านทั้งเพจไม่ว่าจะมีกี่เรกคอร์ด

1(b) การค้นหาข้อมูลแบบเท่ากับจากไฟล์ฮีป เนื่องจากเป็นไฟล์ฮีปที่ไม่มีการจัดเรียง เราจึงจำเป็นต้องค้นหาตั้งแต่ต้นไฟล์ (sequential search) จนกว่าจะพบข้อมูลที่ต้องการ กรณีที่ดีที่สุดคืออ่านเพจแรกแล้วพบข้อมูลที่ต้องการทันที คืออ่าน 1 เพจ กรณีแย่สุดคืออ่านจนเพจสุดท้าย คืออ่าน B เพจ แต่ด้วยว่าเราใช้วิธีวิเคราะห์แบบเฉลี่ย จึงเฉลี่ยว่าต้องอ่านข้อมูล 0.5B จึงจะพบข้อมูลที่ต้องการ แต่ละเพจใช้เวลาอ่านเท่ากับ D ดังนั้น การค้นหาข้อมูลแบบเท่ากับจากไฟล์ฮีปจึงใช้เวลา 0.5BD

1(c) การค้นหาข้อมูลแบบช่วงจากไฟล์ฮีป เนื่องจากไม่มีการเรียงข้อมูลใด ๆ การค้นหาข้อมูลแบบช่วงจึงต้องอ่านข้อมูลทุกเพจ คือ B เพจ เพราะข้อมูลที่

ต้องการอาจอยู่ท้ายไฟล์ได้ แต่ละเพจใช้เวลาอ่านเท่ากับ D ดังนั้น การค้นหาข้อมูลแบบช่วงไฟล์ฮีปจึงใช้เวลา BD

1(d) การแทรกข้อมูลในไฟล์ฮีป เนื่องจากไฟล์ไม่จัดเรียง การแทรกข้อมูลจึงทำได้ด้วยการต่อท้ายไฟล์ (append at eof) ได้ทันที ในโครงสร้างของไฟล์ฮีปจะมีเฮดเดอร์ (header) ระบุที่อยู่ของเพจ ทำให้เราสามารถเข้าถึงเพจที่มีที่ว่างได้ทันที การจะแทรกข้อมูลได้นั้น จะต้องอ่านเพจที่มีที่ว่างนั้นมาใส่ไว้ในหน่วยความจำก่อน การอ่านใช้เวลาเท่ากับ D แล้วเราก็แทรกข้อมูลไว้ที่เพจนั้น ขั้นต่อไปก็ต้องเขียนเพจนั้นกลับลงบนดิสก์ การเขียนใช้เวลาเท่ากับ D สรุปโอเปอเรชันนี้จึงใช้เวลาทั้งหมด 2D

1(e) การลบข้อมูลในไฟล์ฮีป ก่อนจะลบข้อมูลได้ ก็ต้องค้นหาข้อมูลที่จะลบเสียก่อน จึงใช้เวลาในการค้นหาเท่ากับ BD เมื่อพบข้อมูลที่จะลบแล้วก็ทำโอเปอเรชันลบได้เพราะโอเปอเรชันค้นหาได้นำเพจนั้นมาอยู่ในหน่วยความจำแล้ว เสร็จแล้วก็ต้องเขียนเพจนั้นกลับ ใช้เวลาเท่ากับ D โดยรวมโอเปอเรชันนี้ใช้เวลาเท่ากับ Search + D หรือ BD + D

2(a) การสแกนไฟล์แบบเรียงลำดับ อย่างไรก็ต้องสแกนทั้งไฟล์ จึงต้องอ่านข้อมูลทั้งหมด คือ B เพจ แต่ละเพจใช้เวลาอ่านเท่ากับ D ดังนั้น โอเปอเรชันนี้จึงใช้เวลา BD

2(b) การค้นหาข้อมูลแบบเท่ากับจากไฟล์แบบเรียงลำดับ เนื่องจากเป็นไฟล์ที่เรียงข้อมูลมาแล้ว จึงสามารถค้นหาข้อมูลได้ง่าย ถ้าเรามีทั้งหมด B เพจ แล้วค้นหาแบบไบนารี ก็จะต้องอ่านเพจ Log_2B เพจ ใช้เวลาเพจละ D จึงใช้เวลาทั้งหมด $D(log_2B)$

2(c) การค้นหาข้อมูลแบบช่วงจากไฟล์แบบเรียงลำดับ ให้เริ่มหาข้อมูลตัวแรกด้วยวิธีเดียวกับการค้นหาข้อมูลแบบเท่ากับ จึงใช้เวลาค้นหาตัวแรกเท่ากับ $D(\log_2 B)$ จากนั้นก็ยังต้องค้นหาข้อมูลต่อ ถ้าข้อมูลที่ต้องการอยู่ที่เพจเดียวกัน เวลาที่ใช้ทั้งหมดก็จะเท่าเดิม แต่เนื่องจากเราไม่สามารถคาดเดาได้ว่าข้อมูลที่อยู่ในช่วงที่ต้องการนั้นมีจำนวนเท่าใดและอยู่ในกี่เพจ จึงคำนวณเวลาแบบคร่าว ๆ ไว้ว่า เวลาทั้งหมดที่ใช้ในโอเปอชันนี้คือ $D(\log_2 B)$ รวมกับจำนวนข้อมูลที่พบ หรือ $D(\log_2 B)$ + #matches

2(d) การแทรกข้อมูลในไฟล์แบบเรียงลำดับ เนื่องจากไฟล์จัดเรียงไว้ การแทรกข้อมูลจึงทำไม่สามารถต่อท้ายไฟล์ได้ เราจำเป็นต้องหาที่ที่จะแทรกข้อมูล ซึ่งการค้นหาที่แทรกจะใช้เวลาการค้นหาข้อมูลแบบเท่ากับ แต่ก่อนจะแทรกข้อมูลได้ เราจะต้องสร้างช่องว่างให้แทรกข้อมูลใหม่นั้นได้ สมมุติว่าที่แทรกที่เหมาะสมนั้นอยู่ท้ายไฟล์ เราก็ไม่ต้องสร้างช่องว่างเลย แต่ถ้าที่แทรกอยู่ต้นไฟล์ นั่นหมายความว่าเราต้องเลื่อนข้อมูลทั้งไฟล์ให้มีช่องว่าต้นไฟล์ให้ได้ก่อนจะแทรก ในกรณีที่เราวิเคราะห์แบบเฉลี่ย ให้คิดว่าเราจะแทรกที่กลางไฟล์ คือต้องเลื่อนข้อมูลครึ่งไฟล์ออกทีละหนึ่ง ทุกครั้งที่เลื่อนข้อมูลเพื่อสร้างช่องนี้ เราจะต้องอ่านเพจแล้วเขียนกลับ ดังนั้น เราอ่านเพจ 0.5B เพจ และเขียน 0.5B เพจ รวมกับได้ B เพจ ใช้เวลา D ต่อเพจ จึงจะใช้เวลาทั้งหมดคือ Search + BD

2(e) การลบข้อมูลในไฟล์แบบเรียงลำดับ ก่อนจะลบข้อมูลได้ ก็ต้องค้นหาข้อมูลที่จะลบเสียก่อน จึงใช้เวลาในการค้นหาแบบเท่ากับ จากนั้น เมื่อพบข้อมูลที่จะลบแล้วก็ทำโอเปอเรชันลบได้ ซึ่งก็จะทำให้เกิดช่องว่างในเพจ เพราะข้อมูลถูกลบไป ดังนั้น เราจึงต้องกระชับไฟล์ด้วยวิธีเดียวกับการแทรก

คือเลื่อนข้อมูลทีละตัวจนครบ ต้องอ่าน 0.5B และเขียนทีละเพจ 0.5B โดยเฉลี่ยแล้วจึงใช้เวลาเท่ากับการแทรกข้อมูล คือ Search + BD

3(a) การสแกนไฟล์แบบคลัสเตอร์ ไฟล์แบบคลัสเตอร์นี้มีลักษณะคล้ายต้นไม้บีพลัสที่ลีฟเพจนั้นเก็บเรกคอร์ดจริง ดังนั้นเนื่องจากโครงสร้างเป็นแบบต้นไม้ เราจึงต้องคำนึงถึงฟิลแฟคเตอร์ที่เรากำหนดให้เป็น 2/3 หรือ 67% ทำให้จำนวนเพจที่ต้องสแกนทั้งหมดเพิ่มเป็น 1.5 เท่าจากของเดิม คือ 1.5B การอ่านข้อมูลใช้เวลาเพจละ D จึงทำให้โอเปอเรชันนี้ใช้เวลาทั้งหมด 1.5BD

3(b) การค้นหาข้อมูลแบบเท่ากับจากไฟล์แบบคลัสเตอร์ จะคล้ายกับการค้นหาข้อมูลแบบช่วงจากไฟล์แบบเรียงลำดับ แต่เนื่องจากเราไม่สามารถใช้การค้นหาแบบไบนารีแล้ว ด้วยเหตุผลว่าต้นไม้เป็นแบบบีพลัส ทำให้พอยน์เตอร์ไม่เท่ากับสองเหมือนเดิม ซึ่งต้นไม้แบบบีพลัสทรีมีพอยน์เตอร์เท่ากับจำนวนแฟนเอาต์ แทนด้วย F ดังนั้น เวลาที่ใช้ในการค้นหา B เพจ จะเป็น $\log_F B$ แทน $\log_2 B$ นอกเหนือจากนั้น จำนวนเพจในกรณีนี้เป็น 1.5B การอ่านข้อมูลใช้เวลาเพจละ D จึงทำให้เวลาทั้งสิ้นมีค่าเท่ากับ $D(\log_F 1.5B)$

3(c) การค้นหาข้อมูลแบบช่วงจากไฟล์แบบคลัสเตอร์ ให้คิดเหมือนกับการค้นหาข้อมูลแบบเท่ากับจากไฟล์แบบคลัสเตอร์ แล้วบวกเวลาในการหาค่าอื่นในช่วง ทำให้เวลาทั้งสิ้นมีค่าเท่ากับ $D(\log_F 1.5B)$ + #matches

3(d) การแทรกข้อมูลในไฟล์แบบคลัสเตอร์ ก็ต้องเริ่มจากการค้นหาแบบเท่ากับเสียก่อน พบแล้วก็แทรกข้อมูลได้เลย เพราะเราได้เผื่อที่ว่างจากฟิลแฟคเตอร์ไว้แล้ว การเขียนข้อมูลคืนจะใช้เวลาเท่ากับ D จึงทำให้เวลาทั้งสิ้นมีค่าเท่ากับ Search + D

3(e) การลบข้อมูลในไฟล์แบบคลัสเตอร์ เหมือนกับการแทรกข้อมูลในไฟล์แบบคลัสเตอร์ทุกกระบวนการ เพราะเมื่อค้นหาข้อมูลที่จะลบได้แล้วก็ทำโอเปอเรชันนั้นได้ จากนั้นก็ต้องเขียนข้อมูลคืน ใช้เวลาเท่ากับ D แต่ไม่ต้องเลื่อนข้อมูลเพื่อกระชับไฟล์ เนื่องจากเป็นต้นไม้และมีฟิลแฟคเตอร์ ทำให้เวลาที่ต้องใช้ทั้งสิ้นมีค่าเท่ากับ Search + D

4(a) การสแกนไฟล์แบบไม่คลัสเตอร์ที่ใช้ดัชนีต้นไม้บีพลัส ให้คิดว่าข้อมูลจริงคือไฟล์ฮีป แล้วเราสร้างดัชนีแบบต้นไม้เอาไว้ เนื่องจากไฟล์นี้ไม่คลัสเตอร์ นั่นหมายความว่าการเรียงของเรกคอร์ดไม่สอดคล้องกับรายการข้อมูลในดัชนีต้นไม้ และเนื่องจากเรากำหนดให้ต้นไม้บีพลัสมีฟิลแฟคเตอร์ 2/3 นั่นคือจำนวนเพจก็จะเป็น 1.5B เพจ แต่เนื่องจากเรากำหนดไว้ว่าเมื่อใช้วิธีดัชนีเบาบาง คือจะเลือกเพียง 10% ของเรกคอร์ดจริงมาทำดัชนี ทำให้ขนาดเพจที่เราต้องสแกนจะเหลือเพียง 10% ของ 1.5B คือ 0.15B คูณด้วยเวลา D ต่อเพจ จึงได้เวลาที่ใช้ในการอ่านดัชนีคือ 0.15BD

แต่เนื่องจากการทำงานแบบใช้ดัชนีนี้ เราก็จะสแกนดัชนีก่อนแล้วก็ตามไปอ่านเรกคอร์ดที่ดัชนีนั้นชี้ไป และเพราะเป็นไฟล์แบบไม่คลัสเตอร์ การอ่านจึงเกิดขึ้นที่ละเรกคอร์ด ไม่ใช่อ่านทีละเพจ มีจำนวน R เรกคอร์ดต่อเพจ อ่านทั้งหมด B เพจ จึงจะต้องใช้เวลาในการอ่านเรกคอร์ดทั้งหมดเท่ากับ RBD

รวมทั้งการสแกนดัชนีและอ่านเรกคอร์ดทั้งหมด จะใช้เวลาทั้งสิ้น RBD + 0.15BD หรือ BD(R+0.15)

4(b) การค้นหาข้อมูลแบบเท่ากับจากไฟล์แบบไม่คลัสเตอร์ที่ใช้ดัชนีต้นไม้บีพลัส ขั้นตอนการค้นหาข้อมูลก็ไม่ต่างไปจากการค้นหาข้อมูลแบบเท่ากับจากไฟล์แบบคลัสเตอร์ แต่เนื่องจากเราเริ่มต้นที่ 0.15B เพจ แทน 1.5B เพจแบบก่อนหน้า ทำให้เวลาในการค้นหาเป็น $D(\log_F 0.15B)$ ตอนนี้เราก็จะได้พอยน์เตอร์จากดัชนีชี้ไปยังข้อมูลที่ต้องการค้นหา ดังนั้น เราจึงใช้เวลาในการอ่านเพียงเพจนั้นเพจเดียวเท่ากับ D ส่งผลให้เวลาทั้งสิ้นเป็น $D + D(\log_F 0.15B)$ หรือ $D(1+\log_F 0.15B)$

4(c) การค้นหาข้อมูลแบบช่วงจากไฟล์แบบไม่คลัสเตอร์ที่ใช้ดัชนีต้นไม้บีพลัส ให้คิดเหมือนกับการค้นหาข้อมูลแบบเท่ากับจากไฟล์แบบไม่คลัสเตอร์ แล้วบวกเวลาในการหาค่าอื่นในช่วง ทำให้เวลาทั้งสิ้นมีค่าเท่ากับ $D(\log_F 0.15B) + \#matches$

4(d) การแทรกข้อมูลในไฟล์แบบไม่คลัสเตอร์ที่ใช้ดัชนีต้นไม้บีพลัส เนื่องจากเรกคอร์ดเป็นไฟล์ฮีป การแทรกใช้เวลาเท่ากับ 2D เพื่อเขียนข้อมูลเข้าที่ท้ายไฟล์ ใช้เวลาอ่านเท่ากับ D และเขียนเท่ากับ D จากนั้น ก็ต้องมาอัปเดตพอยน์เตอร์ที่ต้นไม้ให้มีพอยน์เตอร์ชี้ไปยังข้อมูลที่แทรกใหม่ให้ได้ จึงต้องค้นหาเพจที่ต้องอัปเดตพอยน์เตอร์ ใช้เวลาค้นหาเท่ากับ $D(1+\log_F 0.15B)$ ได้เพจที่ต้องอัปเดตมาอยู่ในหน่วยความจำ แล้วเขียนพอยน์เตอร์ใช้เวลาเท่ากับ D ทำให้เวลาทั้งสิ้นมีค่าเท่ากับ $D + D(1+\log_F 0.15B) + 2D$ หรือ $D(3+\log_F 0.15B)$

4(e) การลบข้อมูลในไฟล์แบบไม่คลัสเตอร์ที่ใช้ดัชนีต้นไม้บีพลัส ก็เริ่มด้วยการค้นหาเสียก่อน ก็จะได้เรกคอร์ดที่จะลบมาอยู่ในหน่วยความจำ เมื่อลบแล้วก็ต้องเขียนกลับ ใช้เวลาเขียน = D จากนั้นก็ต้องไปอัปเดตพอยน์เตอร์ในดัชนี ใช้เวลา D จึงทำให้เวลาทั้งสิ้นมีค่าเท่ากับ Search + 2D

5(a) การสแกนไฟล์แบบไม่คลัสเตอร์ที่ใช้ดัชนีแบบแฮช ข้อมูลเรกคอร์ดจริงคือไฟล์ฮีปเหมือนแบบที่แล้ว แต่เราสร้างดัชนีแบบแฮชไว้ เนื่องจากไฟล์นี้ไม่คลัสเตอร์ นั่นหมายความว่าการเรียงของเรกคอร์ดไม่สอดคล้องกับรายการข้อมูลในดัชนีแบบแฮช และเนื่องจากเรากำหนดให้แฮชมีอัตราการครอบครองพื้นที่เป็น 80% จึงส่งผลให้จำนวนเพจเพิ่มเป็น 1.5B เพจ แต่เนื่องจากเรากำหนดไว้ว่าเมื่อใช้วิธีดัชนีเบาบาง คือจะเลือกเพียง 10% ของเรกคอร์ดจริงมาทำดัชนี ทำให้ขนาดเพจที่เราต้องสแกนจะเหลือเพียง 10% ของ 1.25B เพจ คือ 0.125B เพจ คูณด้วยเวลา D ต่อเพจ จึงได้เวลาที่ใช้ในการสแกนดัชนีคือ 0.125BD

เช่นเดียวกับแบบที่แล้ว เมื่อเราสแกนดัชนีแล้วก็ต้องตามไปอ่านเรกคอร์ดที่ดัชนีนั้นแฮชแล้วชี้ไป และเพราะเป็นไฟล์แบบไม่คลัสเตอร์ การอ่านจึงเกิดขึ้นที่ละเรกคอร์ด ไม่ใช่อ่านทีละเพจ มีจำนวน R เรกคอร์ดต่อเพจ อ่านทั้งหมด B เพจ จึงจะต้องใช้เวลาในการอ่านเรกคอร์ดทั้งหมดเท่ากับ RBD

รวมทั้งการสแกนดัชนีและอ่านเรกคอร์ดทั้งหมด จะใช้เวลาทั้งสิ้น RBD + 0.125BD หรือ BD(R+0.125)

5(b) การค้นหาข้อมูลแบบเท่ากับจากไฟล์แบบไม่คลัสเตอร์ที่ใช้ดัชนีแบบแฮช ขั้นแรกก็ค้นหาดัชนีด้วยฟังก์ชันแฮช ก็จะได้ผลลัพธ์ว่าข้อมูลนั้นอยู่ถังใด ก็จะไปอ่านดัชนีเพจนั้นมาเก็บไว้ก่อน ใช้เวลาอ่านดัชนีเท่ากับ D แล้วก็ตามไปอ่านเรกคอร์ดจริง ใช้เวลาอ่านเรกคอร์ดเท่ากับ D ส่งผลให้เวลาทั้งสิ้นเป็น 2D

5(c) การค้นหาข้อมูลแบบช่วงจากไฟล์แบบไม่คลัสเตอร์ที่ใช้ดัชนีแบบแฮช กรณีนี้เป็นกรณีที่แปลกออกไป เพราะถ้าคำนวณการทำงานโดยใช้

ฟังก์ชันแฮชแล้วจะส่งผลให้ใช้เวลามาก ดังนั้น ในทางปฏิบัติในกรณีนี้จึงเลือกที่จะไม่ใช้แฮช แล้วเรียกใช้วิธีเดียวกับไฟล์ฮีป จึงได้เวลาเวลาทั้งสิ้นเท่ากับไฟล์ฮีป คือ BD

5(d) การแทรกข้อมูลในไฟล์แบบไม่คลัสเตอร์ที่ใช้ดัชนีแบบแฮช เนื่องจากเรกคอร์ดเป็นไฟล์ฮีป การแทรกใช้เวลาเท่ากับ 2D เพื่อเขียนข้อมูลเข้าที่ท้ายไฟล์ ใช้อ่านเท่ากับ D และเขียนเท่ากับ D จากนั้น ก็ต้องมาอัปเดตพอยน์เตอร์ ซึ่งทำได้ง่ายมากเพราะสามารถคำนวณจากฟังก์ชันแฮชได้ทันที ไม่ต้องเสียเวลาค้นหาดัชนี จึงทำให้ไปอ่านเพจ ใช้เวลาเท่ากับ D และเขียนกลับเท่ากับ D รวมเวลาทั้งสิ้นเท่ากับ 4D

5(e) การลบข้อมูลในไฟล์แบบไม่คลัสเตอร์ที่ใช้ดัชนีแบบแฮช ก็เริ่มด้วยการใช้ฟังก์ชันแฮชหาข้อมูลที่จะลบ แล้วอ่านขึ้นมา ใช้เวลาเท่ากับ D เมื่อลบแล้วเขียนคืน ใช้เวลาเท่ากับ D ค้นหาเสียก่อน ก็จะได้เรกคอร์ดที่จะลบมาอยู่ในหน่วยความจำ เมื่อลบแล้วก็ต้องเขียนกลับ ใช้เวลาเขียนเท่ากับ D จากนั้นก็ต้องไปอัปเดตพอยน์เตอร์ในดัชนี ใช้เวลา D จึงทำให้เวลาทั้งสิ้นมีค่าเท่ากับ Search + 2D

9.11 **ภาระของฐานข้อมูล**

เนื่องจากการทำดัชนีแต่ละแบบจะส่งผลต่อโอเปอเรชันแตกต่างกัน จึงต้องตรวจสอบเวิร์คโหลด (workload) ซึ่งหมายถึง ภาระของฐานข้อมูล ว่าภาระมักเกิดขึ้นที่รีเลชันใด โดยตรวจสอบผ่านการสอบถามที่เกิดขึ้นบ่อย

9.11.1 ภาระการสอบถาม

ให้ดูว่าการสอบถามข้อมูลส่วนใหญ่ภาระตกอยู่ที่ใด ประเด็นที่ต้องตรวจสอบเมื่อมีการสอบถามได้แก่

- รีเลชันใดมีการเข้าถึงบ่อยบ้าง
- แอตทริบิวต์ใดที่ถูกค้นคืน
- คำสั่ง SELECT และ JOIN นั้นมักมีแอตทริบิวต์ใดเกี่ยวข้องบ้าง และแอตทริบิวต์ตัวใดส่งผลให้การคัดเลือกข้อมูลต่างกันอย่างไร ถ้าเลือกข้อมูลได้ดี ก็นับว่ามีความซีเล็กทีฟ (selective) สูง

9.11.2 ภาระการอัปเดต

กรณีถัดไป ให้ดูว่าเมื่อมีการอัปเดตแล้ว ภาระตกอยู่ที่ใด

- เมื่อมีการอัปเดตพร้อมกับใช้คำสั่ง SELECT หรือ JOIN นั้น มักมีแอตทริบิวต์ใดเกี่ยวข้องบ้าง และแอตทริบิวต์ตัวใดมีความซีเล็กทีฟ (selective) อย่างไร
- ในการอัปเดตนั้น มักเป็นการแทรก หรือลบ หรือแก้ไขข้อมูล และมีแอตทริบิวต์ใดเกี่ยวข้องบ้าง

เมื่อเรารู้ภาระของรีเลชันและแอตทริบิวต์ที่ถูกเรียกใช้บ่อย ๆ ผ่านการสอบถามหรือการอัปเดต ก็ให้พิจารณาทำดัชนีให้กับรีเลชันเหล่านั้นด้วยแอตทริบิวต์ที่มักใช้ในการ SELECT หรือ JOIN นั่นเอง

การทำดัชนีไม่ได้ส่งผลทางบวกเสมอไป เพราะการทำดัชนีนั้นก็เพื่อให้การสอบถามได้ผลเร็วขึ้น แต่ก็จะทำให้เกิดภาระทุกครั้งที่อัปเดตฐานข้อมูล จึงต้องพิจารณาและออกแบบอย่างรอบคอบ นอกจากนั้น ผู้ดูแลฐานข้อมูลก็ต้องติดตามประเมินประสิทธิภาพและปรับแก้ดัชนีให้เหมาะสมอยู่เสมอ

9.12 **ข้อแนะนำในการเลือกดัชนี**

ขั้นตอนแรกให้สังเกตที่คำสั่ง WHERE เพื่อระบุแอตทริบิวต์ที่อาจนำมากำหนดเป็นดัชนี

- กรณีของการค้นหาแบบเท่ากับ ให้ใช้ดัชนีแบบแฮช
- สำหรับการค้นหาแบบช่วง ให้ใช้ดัชนีแบบต้นไม้ พร้อมดัชนีแบบคลัสเตอร์เสมอ
- หากกรณีของการค้นหาแบบเท่ากับ ที่มีข้อมูลซ้ำจำนวนมาก ควรให้ใช้ดัชนีแบบต้นไม้ พร้อมดัชนีแบบคลัสเตอร์ร่วมด้วย

เมื่อคำสั่ง WHERE มีเงื่อนไขการคัดเลือกข้อมูลมากกว่าหนึ่งเงื่อนไข ควรพิจารณาใช้คีย์ค้นหาที่มีแอตทริบิวต์มากกว่า 1 ตัวได้

- สำหรับการค้นหาแบบช่วงนั้น ลำดับการจัดเรียงแอตทริบิวต์จะส่งผลต่างกัน
- อาจพิจารณาเลือกใช้ดัชนีแบบอินเด็กซ์โอนลี (index-only) สำหรับการสอบถามที่สำคัญ ๆ ซึ่งหากเลือกใช้อินเด็กซ์โอนลี ก็ไม่ต้องใช้ดัชนีแบบคลัสเตอร์

อย่างไรก็ดี เราควรเลือกสร้างดัชนีที่ส่งผลดีต่อการสอบถามให้มากที่สุด ไม่ใช่จำเพาะเจาะจงที่การสอบถามหนึ่ง ๆ และเนื่องจากการทำดัชนีแบบคลัสเตอร์นั้นสามารถทำได้เพียงดัชนีเดียวต่อหนึ่งรีเลชัน เราจึงต้องพิจารณาให้ถี่ถ้วนว่าควรเลือกทำดัชนีกับแอตทริบิวต์ใดที่จะส่งผลดีต่อการสอบถามส่วนใหญ่

ตัวอย่าง

```
SELECT  E.dno
FROM    Emp E
WHERE  E.age>40
```

ขั้นแรก ให้ตรวจสอบแอตทริบิวต์ในคำสั่ง WHERE ซึ่งในกรณีนี้พบ age จากนั้นให้พิจารณาว่าควรสร้างดัชนีด้วยแอตทริบิวต์ age หรือไม่ ด้วยการดูว่าเงื่อนไขของ WHERE นี้ มีความซีเล็กทีฟมากน้อยเพียงใด เช่น ถ้าพนักงานส่วนใหญ่อายุมากกว่า 40 การทำดัชนีด้วย age ก็จะไม่มีความซีเล็กทีฟ ดังนั้น สมมุติว่าพนักงานส่วนใหญ่อายุน้อยกว่า 40 ก็ให้ใช้ดัชนีแบบต้นไม้พร้อมกับดัชนีแบบคลัสเตอร์

ตัวอย่าง

```
SELECT  E.dno,  COUNT (*)
FROM    Emp E
WHERE  E.age>10
GROUP BY E.dno
```

กรณีนี้มีคำสั่ง WHERE และ GROUP BY ดังนั้น เมื่อพิจารณา WHERE แต่เพียงอย่างเดียว ก็จะเลือกใช้ age ทำดัชนีแบบต้นไม้ พร้อมดัชนีแบบคลัสเตอร์ แต่ในกรณีนี้ ด้วยคำสั่ง GROUP จะส่งผลให้โอเปอเรชัน COUNT ทำได้ยาก เพราะต้องอ่านข้อมูลที่จัดเรียงตามคำสั่ง GROUP นั่นคือเรียงตาม dno หรือหมายเลขดีพาร์ทเมนต์ ดังนั้น จะต้องใช้ dno ทำดัชนีแบบต้นไม้พร้อมดัชนีแบบคลัสเตอร์

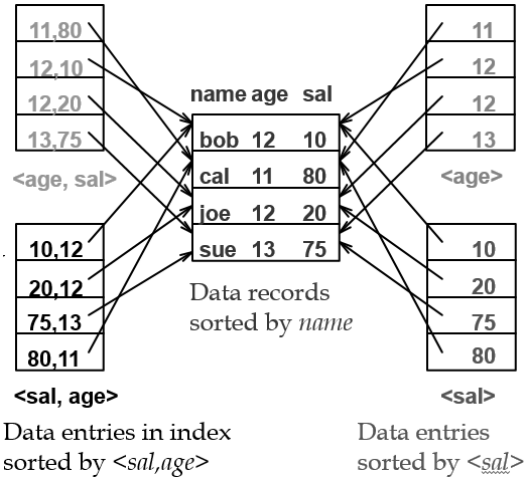
ตัวอย่าง

```
SELECT  E.dno
FROM    Emp E
WHERE  E.hobby=Stamps
```

ในกรณีนี้ เนื่องจากเป็นการค้นหาแบบเท่ากับ จึงดูเหมือนว่าควรจะเลือกใช้ดัชนีแบบแฮช แต่ถ้าพบว่า hobby = Stamps มีซ้ำจำนวนมาก เพราะไม่ใช่คีย์คู่แข่ง ควรใช้ดัชนีแบบต้นไม้พร้อมดัชนีแบบคลัสเตอร์แทน

ตัวอย่าง

เมื่อคีย์ค้นหาเป็นแบบคีย์ค้นหาร่วม (composite search key) ดังแสดงในรูปที่ 124 จะเห็นว่า <age, sal> และ <sal, age> เป็นคีย์ค้นหาร่วม ซึ่งจะส่งผลดีต่อการค้นหาแบบช่วง และลำดับของคีย์ค้นหาอาจเรียงตามตัวอักษร (lexicographic order) หรือตามตำแหน่งของข้อมูล (spatial order) อย่างไรก็ดี ดัชนีแบบคีย์ค้นหาร่วมจะมีขนาดใหญ่และอัปเดตบ่อยกว่า



รูปที่ 124 ดัชนีแบบคีย์ค้นหาร่วม

ตัวอย่าง

จากรูปที่ 124 เมื่อต้องการค้นหาพนักงานที่อายุเท่ากับ 30 และเงินเดือนเท่ากับ 75 ควรใช้ดัชนี <age, sal>

ถ้าต้องการค้นหา 20 < age < 30 และ 70 < sal < 90 ให้ใช้ดัชนีแบบต้นไม้ควบคู่กับดัชนีแบบคลัสเตอร์ โดยให้คีย์ค้นหาร่วมเป็น <age, sal> หรือ <sal, age> ก็ได้

หากต้องการค้นหา age = 30 และ 70 < sal < 90 ให้ใช้ดัชนีแบบคลัสเตอร์ โดยให้คีย์ค้นหาร่วมเป็น <age, sal> ซึ่งจะให้ผลลัพธ์เร็วกว่า <sal, age> มาก

9.13 **อินเด็กซ์โอนลี**

อินเด็กซ์โอนลี (Index only) หมายถึงสถานการณ์ที่สามารถค้นหาข้อมูลในฐานข้อมูลจากดัชนี โดยไม่ต้องเข้าถึงข้อมูลจริง ซึ่งจะทำให้เข้าถึงข้อมูลที่ต้องการได้เร็วมาก

ตัวอย่าง

```
SELECT  D.mgr
FROM    Dept D, Emp E
WHERE   D.dno=E.dno
```

ให้ใช้ <E.dno> สร้างดัชนีแบบต้นไม้

```
SELECT  D.mgr, E.eid
FROM    Dept D, Emp E
WHERE   D.dno=E.dno
```

ให้ใช้ <E.dno,E.eid> สร้างดัชนีแบบต้นไม้

```
SELECT      E.dno, COUNT(*)
FROM        Emp E
GROUP BY  E.dno
```

ให้ใช้ <E.dno> สร้างดัชนีแบบต้นไม้

```
SELECT      E.dno, MIN(E.sal)
FROM        Emp E
GROUP BY  E.dno
```

ให้ใช้ <E.dno,E.sal > สร้างดัชนีแบบต้นไม้

```
SELECT   AVG(E.sal)
FROM     Emp E
WHERE    E.age=25 AND
         E.sal BETWEEN 3000 AND 5000
```

ให้ใช้ <E.age,E.sal > หรือ <E.sal, E.age> สร้างดัชนีแบบต้นไม้

9.14 **บทสรุปและเรื่องชวนคิด**

การเรียนรู้การจัดเก็บข้อมูลจะทำให้ผู้ดูแลฐานข้อมูลสามารถออกแบบหรือปรับปรุงฐานข้อมูลให้ทำงานได้อย่างมีประสิทธิภาพ ซึ่งเป็นสิ่งที่จำเป็นอย่างยิ่งสำหรับวิศวกรคอมพิวเตอร์ที่จะต้องเข้าใจการทำงานของระบบ โดยรวม สำหรับบทนี้ อยากชวนผู้เรียนให้คิดต่อว่าวิธีการจัดเก็บข้อมูลและการทำดัชนีที่ได้เรียนในบทนี้สามารถนำไปใช้กับข้อมูลยุคบิ๊กดาต้าได้หรือไม่ ถ้าได้เพราะอะไร หรือถ้าคิดว่าไม่ได้ก็ขอให้หาความรู้เพิ่มพูนว่ามีวิธีการอะไรอีกบ้างในการจัดเก็บข้อมูลและการทำดัชนี

แบบฝึกหัดท้ายบท

1. เหตุใดที่เราต้องคำนึงถึงวิธีการจัดเก็บฐานข้อมูลที่เหมาะสม และเราควรคำนึงถึงประเด็นใดบ้าง
2. การวัดความเร็วในการอ่านเขียนฐานข้อมูลวัดเป็นอะไร และการที่ระบบฐานข้อมูลตอบสนองได้ไว จะมีค่าดังกล่าวเป็นอย่างไร
3. จงเปรียบเทียบข้อดีข้อเสีย ของโครงสร้างไฟล์แบบไม่เรียงลำดับ ไฟล์แบบเรียงลำดับ และดัชนี
4. หากข้อมูลที่ต้องการสร้างดัชนีมีลักษณะคือ เมื่อค้นหาตามดัชนีนี้แล้วอาจพบเรกคอร์ดมากกว่า 1 เรกคอร์ดได้ เราควรจะใช้วิธีการสร้างรายการข้อมูลแบบใด เพราะอะไร
5. จงเปรียบเทียบระหว่างดัชนีเบาบาง และดัชนีหนาแน่น ในประเด็นของความเร็วในการค้นหา และขนาดพื้นที่จัดเก็บ
6. จงเปรียบเทียบระหว่างโครงสร้างดัชนีแบบแฮช และโครงสร้างดัชนีแบบต้นไม้ ในประเด็นของการค้นหาข้อมูล

บทส่งท้าย

เราเดินทางมาจนถึงบทสุดท้ายของตำรานี้ เพื่อเรียนรู้เรื่องฐานข้อมูล โดยเฉพาะฐานข้อมูลเชิงสัมพันธ์ ความรู้ในระดับนี้นับว่าเป็นความรู้พื้นฐานที่ผู้เรียนวิชาว่าด้วยฐานข้อมูลในระดับปริญญาตรีต้องเข้าใจได้ อย่างไรก็ดี องค์ความรู้ด้านฐานข้อมูลยังมีอีกมาก เช่น การจัดการความปลอดภัยของข้อมูล การปรับสมรรถภาพฐานข้อมูล การประเมินขนาดฐานข้อมูล และอื่น ๆ อีกมากมาย อีกทั้งยังมีรูปแบบฐานข้อมูลที่ต่างไปจากเดิม สามารถเรียนได้จนถึงปริญญาเอก เพื่อเสนอวิธีการที่จะพัฒนาให้ฐานข้อมูลก้าวหน้าไปอีก ผู้เขียนเชื่อว่าพื้นฐานเหล่านี้เพียงพอที่จะให้ผู้เรียนสามารถทำงานด้านฐานข้อมูลในองค์กรหรือเรียนต่อได้

บรรณานุกรม

[1] R. Elmasri and S. Navathe, *Fundamentals of Database Systems - 7th edition*. Pearson London, 2016.

[2] R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems - 5th edition* (Database). 2003.

[3] R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems - 6th Edition*. 2011.

[4] J. Martin and James, *Computer data-base organization*. Prentice-Hall, 1977, pp. 713-713.

[5] J. Martin and J. Leben, *Client/server databases : enterprise computing*. Prentice Hall PTR, 1995, pp. 352-352.

[6] B. Boehm, "A view of 20th and 21st century software engineering," in *Proceedings of the 28th international conference on Software engineering*, 2006, pp. 12-29: ACM.

[7] C. Ghezzi, M. Jazayeri, and D. Mandrioli, *Fundamentals of software engineering*. Prentice Hall PTR, 2002.

[8] W. S. Humphrey, *A discipline for software engineering*. Addison-Wesley Longman Publishing Co., Inc., 1995.

[9] *The database development life cycle:* 1.1 *Introduction - OpenLearn - Open University - M*359_1. Available:

http://www.open.edu/openlearn/science-maths-technology/computing-and-ict/information-and-communication-technologies/the-database-development-life-cycle/content-section-1.1

[10] W. Adrienne and A. Watt, "Database Design - 2nd Edition," ed.

[11] H. Garcia-Molina *et al.*, *Database Systems: A Practical Approach to Design, Implementation, and Management*. 2010.

[12] M. L. Gillenson, *Fundamentals of database management systems*. Wiley, 2005, pp. 366-366.

[13] R. Ramakrishnan and J. Gehrke, *Database Management Systems*. 2003.

[14] T. Risch *et al.*, "Database Management System," 2009.

[15] C. Coronel and S. Morris, *Database systems: design, implementation, & management*. Cengage Learning, 2016.

[16] Y. E. Gelogo and S. Lee, "Database management system as a cloud service," *International Journal of Future Generation Communication and Networking,* vol. 5, no. 2, pp. 71-76, 2012.

[17] S. Ramanathan, S. Goel, and S. Alagumalai, "Comparison of cloud database: Amazon's SimpleDB and Google's Bigtable," in *Recent Trends in Information Systems*

(ReTIS), 2011 *International Conference on*, 2011, pp. 165-168: IEEE.

[18] B. Alam, M. Doja, M. Alam, and S. Mongia, "5-Layered Architecture of Cloud Database Management System," *AASRI Procedia,* vol. 5, pp. 194-199, 2013.

[19] D. M. DeKimpe, W. E. Malloy, and C. R. Tomlyn, "Extension of data definition language (DDL) capabilities for relational databases for applications issuing DML and DDL statements," ed: Google Patents, 2002.

[20] R. Elmasri and S. B. Navathe, "Fundamentals of Database Systems 5th edition," *Database,* 2003.

[21] *Types of Database Management Systems*. Available: https://www.c-sharpcorner.com/UploadFile/65fc13/types-of-database-management-systems/

[22] *Types of Databases and Dbms (With Examples)*. Available: https://codebots.com/data-management/types-of-databases-and-dbms-with-examples

[23] R. Elmasri, J. Weeldreyer, and A. Hevner, "The category concept: An extension to the entity-relationship model," *Data and Knowledge Engineering,* 1985.

[24] H. Garcia-Molina, J. D. Ullman, and J. Widom, "Database Systems: The Complete Book," *Education,* 2008.

[25] J. A. Hoffer, *Modern Database Management,* 10/*e*. Pearson Education India, 2011.

[26] I.-Y. Song, M. Evans, and E. K. Park, "A comparative analysis of entity-relationship diagrams," *Journal of Computer and Software Engineering,* vol. 3, no. 4, pp. 427-459, 1995.

[27] C. Liang *et al.*, "MAGIC-SPP: a database-driven DNA sequence processing package with associated management tools," *BMC bioinformatics,* vol. 7, no. 1, p. 115, 2006.

[28] G. Chen and E. Kerre, "Extending ER/EER concepts towards fuzzy conceptual data modeling," in *Fuzzy Systems Proceedings,* 1998. *IEEE World Congress on Computational Intelligence., The* 1998 *IEEE International Conference on*, 1998, vol. 2, pp. 1320-1325: IEEE.

[29] R. Elmasri and S. Navathe, *Fundamentals of database systems*. Addison-Wesley Publishing Company, 2010.

[30] *Top 6 Data Modeling Tools - Data Science Central*. Available: https://www.datasciencecentral.com/profiles/blogs/top-6-data-modeling-tools

[31] *Comparison of data modeling tools - Wikipedia*. Available:

https://en.wikipedia.org/wiki/Comparison_of_data_modeling_tools

[32] E. F. Codd, "A relational model of data for large shared data banks," *Communications of the ACM,* vol. 13, no. 6, pp. 377-387, 1970.

[33] E. F. Codd, "A data base sublanguage founded on the relational calculus," in *Proceedings of the* 1971 *ACM SIGFIDET (now SIGMOD) Workshop on Data Description, Access and Control*, 1971, pp. 35-68: ACM.

[34] E. F. Codd, "Normalized data base structure: A brief tutorial," in *Proceedings of the* 1971 *ACM SIGFIDET (now SIGMOD) Workshop on Data Description, Access and Control*, 1971, pp. 1-17: ACM.

[35] E. F. Codd, *Relational completeness of data base sublanguages*. Citeseer, 1972.

[36] E. F. Codd, "Extending the database relational model to capture more meaning," *ACM Transactions on Database Systems (TODS),* vol. 4, no. 4, pp. 397-434, 1979.

[37] E. F. Codd, "Data models in database management," *ACM Sigmod Record,* vol. 11, no. 2, pp. 112-114, 1981.

[38] E. F. Codd, "Missing information (applicable and inapplicable) in relational databases," *ACM Sigmod Record,* vol. 15, no. 4, pp. 53-53, 1986.

[39] E. F. Codd, "Relational database: a practical foundation for productivity," in *Readings in Artificial Intelligence and Databases*: Elsevier, 1988, pp. 60-68.

[40] E. F. Codd, *The relational model for database management: version* 2. Addison-Wesley Longman Publishing Co., Inc., 1990.

[41] M. Campbell-Kelly, "Edgar Codd," *The Independent,* 2003.

[42] E. F. Codd, 1981 *Turing Award Lecture – Relational Database: A Practical Foundation for Productivity*. 1981.

[43] C. J. Date, *The database relational model : a retrospective review and analysis : a historical account and assessment of E.F. Codd's contribution to the field of database technology*. Addison-Wesley, 2001, pp. 152-152.

[44] *Database Course, graduate database course, databases, DBMS, Graduate Database, Oracle*. Available: http://sce.uhcl.edu/boetticher/courses.html

[45] M. Lacroix and A. Pirotte, "Domain-oriented relational languages," in *Proceedings of the third international conference on Very large data bases-Volume* 3, 1977, pp. 370-378: VLDB Endowment.

[46] A. Beaulieu, *Learning SQL*, 2nd ed. Sebastapol, CA, USA: O'Reilly, 2009, pp. 320-320.

[47] M. Chapple, "SQL Fundamentals," *Databases*.

[48] M. Chatham, *Structured query language by example - volume i : data query language*. Lulu Com, 2012, pp. 8-8.

[49] C. J. Date and H. Darwen, *A guide to the SQL standard : a user's guide to the standard database language SQL*. Addison-Wesley, 1997, pp. 522-522.

[50] J. Melton and A. R. Simon, *Understanding the new SQL : a complete guide*. Morgan Kaufmann Publishers, 1993, pp. 536-536.

ดัชนี

O

P

Q

R

S

T

U

V

W

X

ก

ข

ค

ง

ผ

พ

ฟ

ภ

ม

ห

อ

ฮ

การออกแบบฐานข้อมูลเป็นทักษะที่สำคัญทุกคนในวงการไอที
เพราะองค์กรไม่ว่าขนาดเล็กหรือใหญ่ล้วนจำเป็นต้องใช้ฐานข้อมูล
ในการดำเนินธุรกิจหรือกิจกรรมใด ๆ ที่มีข้อมูลจำนวนมาก
หากออกแบบได้ไม่เหมาะสมอาจส่งผลให้การทำงานของระบบฐานข้อมูลนั้นล้มเหลวได้ง่าย
ตำราพื้นฐานการออกแบบระบบฐานข้อมูลนี้รวบรวมความรู้พื้นฐานแนวคิด เครื่องมือ วิธีการ
และตัวอย่างการออกแบบระบบฐานข้อมูลเชิงสัมพันธ์

เหมาะสำหรับนิสิตระดับปริญญาตรีที่สนใจความรู้ด้านคอมพิวเตอร์
ที่ต้องเข้าใจการทำงานและการคำนวณภายในฐานข้อมูล
จึงจะออกแบบฐานข้อมูลได้อย่างมีประสิทธิภาพ
และมีพื้นฐานที่เข้มแข็งพอที่จะพัฒนาองค์ความรู้ใหม่ต่อไป
หรือประกอบวิชาชีพในองค์กรที่ใช้ฐานข้อมูลโดยเฉพาะฐานข้อมูลเชิงสัมพันธ์