เอกสารประกอบ 4.

ส่วนที่ 1.

เอกสารประกอบการสอนรายวิชา

204204 การออกแบบและพัฒนาฐานข้อมูล

ภาคการศึกษาที่ 2 ปีการศึกษา 2551

สถิตย์โชค โพธิ์สอาด

สาขาวิชาเทคโนโลยีสารสนเทศ

สำนักวิชาเทคโนโลยีสังคม

มหาวิทยาลัยเทคโนโลยีสุรนารี

กันยายน 2551

สารบัญ

บทที่ 1 แนวคิดทั่วไปเกี่ยวกับฐานข้อมูล

วัตถุประสงค์

- สามารถอธิบายแนวคิดเกี่ยวกับระบบจัดการฐานข้อมูล (database management system – DBMS) ได้
- สามารถอธิบายเกี่ยวกับระบบจัดการฐานข้อมูลเชิงสัมพันธ์ได้
- สามารถอธิบายถึงความจำเป็นในการใช้งานระบบจัดการฐานข้อมูล
- สามารถอธิบายว่าข้อมูลที่นำมาใช้ประโยชน์ถูกจัดเก็บไว้ในระบบจัดการฐานข้อมูลอย่างไร
- สามารถอธิบายว่าการค้นคืนและแก้ไขข้อมูลในระบบจัดการฐานข้อมูลทำได้อย่างไร
- สามารถอธิบายว่าระบบจัดการฐานข้อมูลรองรับการใช้งานของผู้ใช้หลายๆ คนพร้อมๆ กันได้อย่างไร
- สามารถอธิบายถึงวิธีการที่ระบบจัดการฐานข้อมูลใช้ป้องกันข้อมูลผิดพลาดในกรณีที่ระบบล้มเหลว
- สามารถอธิบายองค์ประกอบหลักของระบบจัดการฐานข้อมูล
- สามารถระบุผู้ที่มีส่วนเกี่ยวข้องกับการใช้งานระบบจัดการฐานข้อมูล

คำสำคัญ: การจัดการฐานข้อมูล (database management) ความเป็นอิสระของข้อมูล (data independence) การออกแบบฐานข้อมูล (database design) แบบจำลองข้อมูล (data model) ฐานข้อมูลเชิงสัมพันธ์และข้อคำถาม (relational database and queries) สคีมา (schema) ระดับของฐานข้อมูล (levels of abstraction) ธุรกรรม (transaction) การเกิดขึ้นพร้อมกันและการปิดกั้น (concurrency and locking) การคืนสภาพและการบันทึกกิจกรรม (recovery and logging) โครงสร้างของระบบจัดการฐานข้อมูล (DBMS structure) ผู้ดูแลฐานข้อมูล (database administrator) โปรแกรมเมอร์ (programmer) ผู้ใช้ (end user)

1.1 บทนำ

ข้อมูล (data) และสารสนเทศ (information) ในปัจจุบันนี้มีอยู่มากมายมหาศาล และเพิ่มจำนวนขึ้นอย่างรวดเร็ว ซึ่งเราไม่สามารถปฏิเสธได้เลยว่าข้อมูลและสารสนเทศนั้นมีประโยชน์มากมายสำหรับผู้นำไปใช้ ในแง่ขององค์กรต่างๆ ก็ตระหนักถึงความสำคัญของข้อมูลเป็นอย่างดี ซึ่งเราได้ให้ความสำคัญกับข้อมูลเปรียบเสมือนสินทรัพย์อย่างหนึ่งขององค์กรไปแล้ว การที่จะใช้ประโยชน์จากข้อมูลที่จำนวนมหาศาลและซับซ้อนนี้จำเป็นที่จะต้องอาศัยเครื่องมือเพื่อช่วยอำนวยความสะดวกในการจัดการข้อมูล ตลอดจนการนำข้อมูลไปใช้งานได้อย่างมีประสิทธิภาพ มิฉะนั้นแล้วค่าใช้จ่ายในการจัดหาและการจัดการข้อมูลจะมีมูลค่าสูงเกินกว่าประโยชน์ที่เราจะได้รับจากตัวข้อมูล

ฐานข้อมูล คือกลุ่มของข้อมูล ซึ่งโดยทั่วไปเป็นข้อมูลที่เกี่ยวข้องกับกิจกรรมต่างๆ ผู้กระทำกิจกรรมนั้นๆ และมีความสัมพันธ์กัน เช่น ฐานข้อมูลของมหาวิทยาลัยอาจจะประกอบไปด้วยข้อมูลดังนี้

- *เอนทิตี* (entity) เช่น นักศึกษา อาจารย์ รายวิชา ห้องเรียน
- *ความสัมพันธ์* (relationships) ระหว่างเอนทิตี เช่น นักศึกษาลงทะเบียนเรียนในรายวิชา อาจารย์สอนในรายวิชา และการใช้งานห้องต่างๆ สำหรับแต่ละรายวิชา

ระบบจัดการฐานข้อมูล (database management system – DBMS) คือซอฟต์แวร์ที่ช่วยในการจัดการและใช้งานกลุ่มของข้อมูลขนาดใหญ่ นอกจากการจัดการข้อมูลนั้นจำเป็นต้องใช้ระบบจัดการฐานข้อมูลดังกล่าวแล้ว ปัจจุบันเราได้ใช้ระบบจัดการฐานข้อมูลกันเป็นที่แพร่หลายซึ่งเกือบเป็นสิ่งหนึ่งที่ขาดไม่ได้ในการบริหารจัดการองค์กร ทางเลือกหนึ่งนอกเหนือจากการใช้ระบบจัดการฐานข้อมูลคือการจัดเก็บข้อมูลลงในไฟล์ข้อมูล และพัฒนาระบบเฉพาะโดยการเขียนโปรแกรมที่จัดการกับไฟล์ที่มีโครงสร้างที่กำหนดขึ้นมาโดยเฉพาะ ทั้งนี้เรามักจะเปรียบเทียบข้อดีและข้อเสียของการจัดการข้อมูลทั้ง 2 วิธี ซึ่งจะได้อธิบายในหัวข้อที่ 1.5 และ 1.6

วัตถุประสงค์ของการศึกษารายวิชาการออกแบบและพัฒนาฐานข้อมูลเน้นที่การออกแบบ พัฒนา และสามารถนำฐานนำระบบฐานข้อมูลที่พัฒนาไปใช้อย่างมีประสิทธิภาพ ทั้งนี้ การที่จะบรรลุวัตถุประสงค์ดังกล่าว เรามีความจำเป็นที่จะต้องเลือกใช้ระบบจัดการฐานข้อมูลที่ถูกต้องเหมาะสมกับการประยุกต์ใช้ ซึ่งการเลือกใช้ระบบจัดการฐานข้อมูล ตลอดจนการออกแบบ และพัฒนาฐานข้อมูลสำหรับระบบจัดการฐานข้อมูลใดๆ เรามีความจำเป็นอย่างยิ่งที่จะต้องเข้าใจถึงการทำงานของระบบจัดการฐานข้อมูล

ระบบจัดการฐานข้อมูลที่ใช้กันในปัจจุบันมีหลายประเภท แต่สำหรับรายวิชานี้จะเน้นที่ระบบจัดการฐานข้อมูลเชิงสัมพันธ์ (relational database management systems – RDBMSs) ซึ่งเป็นประเภทของระบบจัดการฐานข้อมูลที่แพร่หลายที่สุดในปัจจุบัน สำหรับเรื่องที่จะศึกษาสามารถแบ่งออกเป็นกลุ่มใหญ่ๆ ได้ดังต่อไปนี้

1) **แนวคิดพื้นฐานเกี่ยวกับฐานข้อมูล** ได้แก่เนื้อหาในบทนี้
2) **การออกแบบฐานข้อมูล** หัวข้อดังกล่าวศึกษาเกี่ยวกับ การที่ผู้ใช้อธิบายกิจกรรมต่างๆ ตลอดจนสิ่งที่เกี่ยวข้องกับกิจกรรมนั้นๆ ขององค์กร ทั้งในรูปสิ่งของที่เป็นรูปธรรมและนามธรรม รวมถึงบุคคลต่างๆ อย่างไร ในแง่ของการจัดเก็บเป็นข้อมูลในระบบจัดการฐานข้อมูล ปัจจัยใดบ้างที่ควรคำนึงถึงในการจัดการและจัดเก็บข้อมูล (บทที่ 2-5)

3) **การพัฒนาฐานข้อมูล** การพัฒนาระบบฐานข้อมูล การพัฒนาโปรแกรมประยุกต์หรือระบบสารสนเทศที่ใช้งานระบบจัดการฐานข้อมูลนั้นทำอย่างไร (บทที่ 7-8 และ 10)
4) **การวิเคราะห์และใช้งานข้อมูล** ผู้ใช้จะสามารถตอบคำถามเกี่ยวกับองค์กรในแง่ต่างๆ ได้อย่างไร โดยการใช้ข้อคำถาม (queries) กับระบบจัดการฐานข้อมูล (บทที่ 8)
5) **การจัดการธุรกรรมและความมั่นคงปลอดภัย** ระบบจัดการฐานข้อมูลรองรับการใช้งานของผู้ใช้หลายๆ คนพร้อมกันได้อย่างไร รวมทั้งระบบจัดการฐานข้อมูลทำอย่างไรในการป้องกันความผิดพลาดของข้อมูลในกรณีที่ระบบล้มเหลว (บทที่ 9)

หลังจากได้แนะนำเกี่ยวกับระบบฐานข้อมูลในเบื้องต้น และภาพรวมของเรื่องที่จะศึกษาในรายวิชาแล้ว ในบทนี้ยังจะกล่าวถึงข้อมูลและสารสนเทศ ซึ่งเน้นย้ำให้เห็นถึงความสำคัญของระบบฐานข้อมูลในหัวข้อที่ 2 วิวัฒนาการของการจัดการฐานข้อมูลและบทบาทของระบบจัดการฐานข้อมูลในระบบสารสนเทศสมัยใหม่ในหัวข้อที่ 3 ระบุถึงข้อเสียของการจัดเก็บข้อมูลด้วยระบบไฟล์ข้อมูลและระบบจัดการฐานข้อมูลสามารถลบข้อเสียของระบบไฟล์ได้อย่างไรในหัวข้อที่ 4 อธิบายถึงข้อดีของการใช้งานระบบจัดการฐานข้อมูลในหัวข้อที่ 5

สำหรับในหัวข้อที่ 6 นั้น ได้อธิบายว่าเราจะมีวิธีการอย่างไรในการจัดเก็บข้อมูลและสารสนเทศขององค์กรลงในระบบจัดการฐานข้อมูล ผู้ใช้จะมองฐานข้อมูลในระดับสูง กล่าวคือมองในเชิงการใช้งานจริงในสภาพแวดล้อมจริง ในขณะที่ระบบจัดการฐานข้อมูลจะจัดเก็บข้อมูลในรูปของข้อมูลดิจิตอลเป็นบิตลงในอุปกรณ์จัดเก็บ ทั้งนี้จะพบว่าระหว่างระดับของข้อมูลในระดับที่ผู้ใช้มองเห็นกับระดับจัดเก็บข้อมูลลงในฮาร์ดแวร์จะถูกขั้นและจัดการด้วยระดับต่างๆ ซึ่งเราเรียกว่าระดับของแอบแสตรคชั่นซึ่งถูกสร้างขึ้นมาเพื่อเอื้ออำนวยต่อกันจัดการฐานข้อมูลอย่างมีประสิทธิภาพ

หัวข้อที่ 7 อธิบายถึงวิธีการที่ผู้ใช้จะทำอย่างไรในการเรียกใช้ข้อมูลที่ถูกจัดเก็บอยู่ในระบบจัดการฐานข้อมูล รวมถึงความจำเป็นในการใช้เทคนิคต่างๆ ในการดึงข้อมูลออกมาจากฐานข้อมูลอย่างรวดเร็ว หัวข้อที่ 8 อธิบายภาพรวมเกี่ยวกับการทำงานของระบบจัดการฐานข้อมูลว่ามีวิธีการอย่างไรในการรองรับการทำงานของผู้ใช้หลายๆ คนพร้อมๆ กัน ตลอดจนระบบจัดการฐานข้อมูลป้องกันข้อมูลจากความผิดพลาดและเสียหายในกรณีที่ระบบล้มเหลวได้อย่างไร

1.2 ข้อมูลและสารสนเทศ

ข้อมูล คือ ข้อเท็จจริงเกี่ยวกับสิ่งต่างๆ

สารสนเทศ คือ ข้อมูลที่ผ่านการประมวลผล

ข้อมูลและสารสนเทศมีความจำเป็นในทุกวงการ เช่นในแง่ธุรกิจนั้นมีความสำคัญต่อผู้บริหารในการตัดสินใจดำเนินธุรกิจต่างๆ ทั้งนี้เราต้องเข้าใจความหมายของคำว่าข้อมูลและสารสนเทศก่อน**ข้อมูล (data)** คือข้อเท็จจริงที่

เกี่ยวข้องกับสิ่งที่เราสนใจจะจัดเก็บซึ่งได้แก่เอนทิตีและความสัมพันธ์ดังได้กล่าวถึงในหัวข้อที่แล้ว **สารสนเทศ (information)** หรือในบางครั้งเรียกว่าข้อมูลสารสนเทศ คือข้อมูลที่เรานำมาประมวลผล การประมวลผลคือกระบวนการใดๆ ที่จะได้มาซึ่งสารสนเทศที่นำมาใช้ประโยชน์ได้ เรานำสารสนเทศมาใช้ประกอบการตัดสินใจดำเนินการเพื่อให้บรรลุวัตถุประสงค์ขององค์กร

ข้อมูล (data) → ประมวลผล (process) → สารสนเทศ (information)

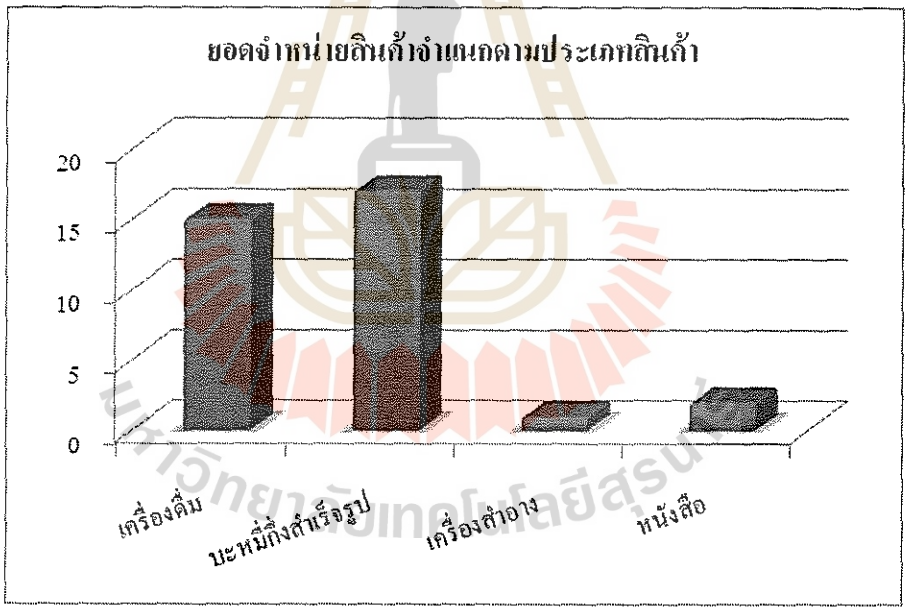
เราสามารถยกตัวอย่างความแตกต่างระหว่างข้อมูลและสารสนเทศ ตลอดจนการนำสารสนเทศไปใช้ประโยชน์ได้ด้วยตัวอย่างข้อมูลการจดบันทึกการขายของร้านขายของชำดังนี้ (สำหรับเป็นตัวอย่างเท่านั้น ข้อมูลมีปริมาณน้อยเพื่อความสะดวกในการอธิบาย)

รหัสสินค้า	ชื่อสินค้า	ประเภทสินค้า	จำนวนที่ขาย	วันที่
00001	มาม่ารสหมูสับ	บะหมี่กึ่งสำเร็จรูป	5	1/09/2007
00002	ยำยำรสหมูสับ	บะหมี่กึ่งสำเร็จรูป	5	5/09/2007
00001	มาม่าต้มยำกุ้ง	บะหมี่กึ่งสำเร็จรูป	3	10/09/2007
00005	ชาเขียวโออิชิรสข้าวญี่ปุ่น	เครื่องดื่ม	1	10/09/2007
00010	แชมพูแพนทีน 500 มล.	เครื่องสำอาง	1	17/09/2007
00001	มาม่ารสหมูสับ	บะหมี่กึ่งสำเร็จรูป	4	18/09/2007
00098	โค้กซีโร่กระป๋อง 325 มล.	เครื่องดื่ม	12	20/09/2007
00209	เธอกับฉัน กันยายน 2550	หนังสือ	1	22/09/2007
00005	ชาเขียวโออิชิรสข้าวญี่ปุ่น	เครื่องดื่ม	2	28/09/2007
00112	ขายหัวเราะ กันยายน 2550	หนังสือ	1	29/09/2007

จากข้อมูลดิบข้างต้น ยังไม่มีประโยชน์ต่อการช่วยในการนำมาพัฒนาการขาย เราสามารถนำรายการขายตลอดเดือนมาพิจารณาสินค้าที่ขายดีเป็นพิเศษ เพื่อซื้อสินค้านั้นๆ ในปริมาณมากขึ้น และสินค้าชิ้นใดที่ขายได้ไม่ดีซึ่งเราจะสามารถวิเคราะห์ในภายหลังได้ว่าเกิดจากอะไรและเราอาจจะไม่จำเป็นต้องสั่งซื้อสินค้านั้นๆ มาขาย หรือในทางกลับกันอาจจัดรายการส่งเสริมการขายสินค้าที่ขายได้น้อยแต่ได้กำไรมาก ให้มียอดจำหน่ายเพิ่มขึ้น ที่จะได้มาซึ่งสารสนเทศดังกล่าว เราสามารถประมวลผลได้ด้วยวิธีการต่างๆ เช่นการรวมจำนวนสินค้าที่เป็นสินค้าเดียวกันและเรียงลำดับข้อมูลจากมากไปน้อย

รหัสสินค้า	ชื่อสินค้า	ประเภทสินค้า	จำนวนที่ขาย
00098	โค้กซีโร่กระป๋อง 325 มล.	เครื่องดื่ม	12
00001	มาม่ารสหมูสับ	บะหมี่กึ่งสำเร็จรูป	9
00002	ยำยำรสหมูสับ	บะหมี่กึ่งสำเร็จรูป	5
00001	มาม่าต้มยำกุ้ง	บะหมี่กึ่งสำเร็จรูป	3
00005	ชาเขียวโออิชิรสข้าวญี่ปุ่น	เครื่องดื่ม	3
00010	แชมพูแพนทีน 500 มล.	เครื่องสำอาง	1
00209	เธอกับฉัน กันยายน 2550	หนังสือ	1
00112	ขายหัวเราะ กันยายน 2550	หนังสือ	1

เราสามารถประมวลผลให้สารสนเทศที่ได้มามีประโยชน์หรือสามารถเข้าใจได้ง่ายขึ้นอีกด้วยการทำให้เห็นภาพเช่นการสร้างแผนภูมิจากสารสนเทศที่ได้ ดังตัวอย่าง



เราจะเห็นภาพชัดเจนถึงประเภทสินค้าที่มียอดจำหน่ายสูงจากสารสนเทศที่ได้จากการประมวลผล ทำให้การวางแผนการบริหารร้านค้าสามารถทำได้อย่างมีกลวิธีและมีประสิทธิภาพเพิ่มขึ้น ในกรณีนี้ร้านค้าควรที่จะซื้อบะหมี่กึ่งสำเร็จรูปในปริมาณที่มากกว่าสินค้าอื่นๆ เพื่อให้เพียงพอต่อการจำหน่าย การจัดเก็บข้อมูล การนำข้อมูลมาใช้ประโยชน์ ตลอดจนการประมวลผลข้อมูลและนำสารสนเทศที่ได้มาใช้ประโยชน์นั้นเราสามารถใช้ระบบสารสนเทศ (information system) มาจัดการได้ ซึ่งระบบสารสนเทศในปัจจุบันอาศัยการจัดการข้อมูลที่มีประสิทธิภาพจากระบบจัดการฐานข้อมูล

1.3 วิวัฒนาการของฐานข้อมูล

การจัดเก็บ แก้ไข และบริหารจัดการข้อมูลเป็นหัวข้อการศึกษาและวิจัยที่สำคัญ ตลอดจนได้รับการพัฒนามาอย่างต่อเนื่อง ระบบจัดการฐานข้อมูลสำหรับใช้งานทั่วๆ ไประบบแรกได้รับการออกแบบโดยนายชาลส์ แบคแมน (Charles Bachman) แห่งบริษัทเจเนอรัลอิเล็กทริก ในต้นคริตศักราช 1960 ระบบดังกล่าวถูกเรียกว่า Integrated Data Store ซึ่งเป็นรากฐานของการจัดเก็บขอมูลโดยใช้แบบจำลองข้อมูลแบบเครือข่าย (network data model) ทั้งนี้ระบบดังกล่าวได้รับการยอมรับและกำหนดมาตรฐานขึ้นโดย the Conference on Data Systems Languages (CODASYL) รูปแบบของการจัดเก็บข้อมูลดังกล่าวได้รับการยอมรับและมีอิทธิอย่างมากต่อรูปแบบของสถาปัตยกรรมระบบจัดการฐานข้อมูลในช่วงทศวรรษ 1960 แบคแมนเป็นบุคคลแรกที่ได้รับ ACM's Turing Award ซึ่งเป็นรางวัลที่เปรียบได้กับรางวัลโนเบล สำหรับการศึกษาด้านฐานข้อมูล ในปี 1973

ในช่วงปลายทศวรรษที่ 1960 บริษัทไอบีเอ็มได้พัฒนาระบบจัดการฐานข้อมูลในชื่อ Information Management System (IMS) ซึ่งยังคงได้รับการใช้งานอยู่ในปัจจุบันสำหรับฐานข้อมูลขนาดใหญ่ โดย IMS นี้นำเสนอรูปแบบของการจัดการข้อมูลด้วยแบบจำลองข้อมูลแบบลำดับขั้น (hierarchical data model) ในช่วงเวลาเดียวกันระบบ SABRE ซึ่งเป็นระบบสำหรับการจองตั๋วเครื่องบินที่บริษัทไอบีเอ็มพัฒนาร่วมกับอเมริกันแอร์ไลนส์ได้ถูกพัฒนาขึ้น เพื่อให้สามารถใช้งานผ่านเครือข่ายคอมพิวเตอร์ได้ ระบบดังกล่าวมีความน่าสนใจตรงที่เป็นระบบสำคัญสำหรับการจองตั๋วเครื่องบินในปัจจุบัน เช่นเว็บไซต์ Travelocity ยังคงใช้งานระบบ SABRE

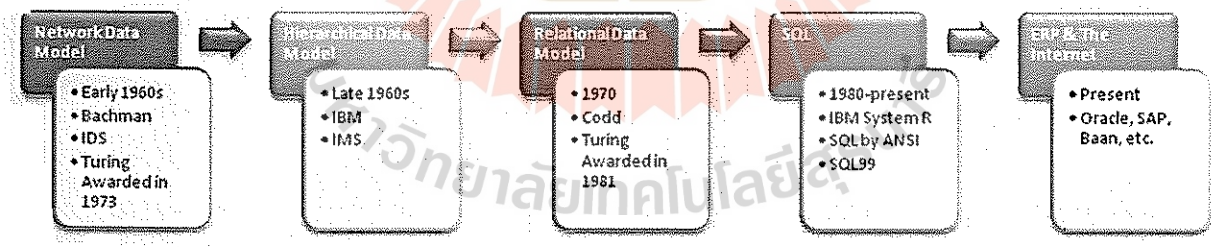
ในปี 1970 เอ็ดการ์ ค็อด (Edgar Codd) แห่งห้องปฏิบัติการวิจัยซานโฮเซของบริษัทไอบีเอ็ม ได้เสนอรูปแบบการจัดการข้อมูลด้วยแบบจำลองข้อมูลเชิงสัมพันธ์ (relational data model) ซึ่งรูปแบบดังกล่าวมีความเหมาะสมในการใช้งานเป็นอย่างยิ่งและได้รับความนิยมแพร่หลายอย่างรวดเร็ว นอกจากนี้มีผู้สนับสนุนและวิจัยเพื่อให้เกิดทฤษฎีมารองรับและประยุกต์ใช้อย่างเป็นระบบ นอกจากจะได้รับความสนใจในแวดวงวิชาการโดยถือเป็นศาสตร์แขนงหนึ่งแล้ว ระบบจัดการฐานข้อมูลเชิงสัมพันธ์ยังมีผลกระทบต่อแวดวงธุรกิจ การนำระบบจัดการฐานข้อมูลเชิงสัมพันธ์ไปช่วงในการบริหารองค์กรก่อให้เกิดประโยชน์อย่างสูง ได้รับการยอมรับ และถือเป็นเครื่องมือมาตรฐานอย่างหนึ่งที่ควรนำมาใช้เพื่ออำนวยความสะดวกในการปฏิบัติงานและเพิ่มประสิทธิภาพขององค์กร ค็อดได้รับรางวัล Turing ในปี 1981

ในช่วงทศวรรษ 1980 แบบจำลองเชิงสัมพันธ์ได้กลายเป็นมาตรฐานหลักของระบบจัดการฐานข้อมูล และมีการใช้ระบบจัดการฐานข้อมูลกันอย่างแพร่หลาย ภาษาที่ใช้สำหรับจัดการฐานข้อมูลเชิงสัมพันธ์ที่ได้รับการยอมรับเป็นมาตรฐานได้แก่ภาษา SQL ซึ่งถูกพัฒนาขึ้นโดยบริษัทไอบีเอ็มภายใต้โครงการ IBM's System R โดย SQL ดังกล่าวได้ถูกกำหนดมาตรฐานขึ้นในปลายทศวรรษที่ 1980 โดย SQL ที่ใช้ในปัจจุบันเป็นมาตรฐาน SQL 1999 โดยได้รับการยอมรับจาก American National Standards Institute (ANSI) ซึ่ง SQL 1999 ออกแบบมาเพื่อให้รองรับการใช้งานฐานข้อมูลโดยผู้ใช้ฐานข้อมูลหลายคน และธุรกรรมที่เกิดขึ้นพร้อมๆ กันได้ โดยเขียนคำสั่งเสมือนมีผู้ใช้งานฐานข้อมูล

ไม่พร้อมกัน แต่ระบบจัดการฐานข้อมูลสามารถรองรับการใช้งานฐานข้อมูลพร้อมจากผู้ใช้หลายคนได้ เจมส์ เกรย์ (James Gray) ได้รับรางวัล Turing สำหรับการศึกษาที่เป็นประโยชน์ต่อการจัดการธุรกรรมของฐานข้อมูล

ปรากฏการณ์ที่น่าสนใจของการจัดการฐานข้อมูลได้แก่การกำเนิดของระบบจัดการทรัพยากรองค์กร (enterprise resource planning – ERP) ซึ่งเป็นรูปแบบสมัยใหม่ของการใช้งานระบบจัดการฐานข้อมูล โดยส่วนใหญ่องค์กรธุรกิจจะประกอบไปด้วยหน่วยงานย่อยๆ ที่คล้ายคลึงกันแบ่งตามหน้าที่ทางธุรกิจ (business functions) เช่นฝ่ายทรัพยากรมนุษย์ ฝ่ายการเงินและบัญชี ฝ่ายผลิต ฝ่ายขาย และฝ่ายบริหารเป็นต้น หน่วยงานทางธุรกิจเหล่านี้มีระบบสารสนเทศของตนเองและจัดเก็บข้อมูลลงในระบบจัดการฐานข้อมูล นอกจากข้อมูลที่จัดเก็บในระบบจัดการฐานข้อมูลของหน่วยงานย่อยๆ จะมีความคล้ายคลึงกันในแต่ละบริษัทแล้ว กระบวนการทางธุรกิจอื่นๆ ก็มีความคล้ายคลึงเช่นเดียวกัน จึงมีผู้นำเสนอ ERP ขึ้นเพื่อเชื่อมโยงข้อมูลของหน่วยงานย่อยของบริษัทเข้าด้วยกัน ตลอดจนนำเสนอโปรแกรมประยุกต์ที่อำนวยความสะดวกในกระบวนการธุรกิจมาพร้อมกับระบบจัดการฐานข้อมูลด้วย บริษัทผู้นำด้านซอฟต์แวร์ ERP ได้แก่ Baan Oracle PeopleSoft SAP และ Siebel เป็นต้น ชุด ERP แต่ละชุดสามารถปรับแต่งเพื่อให้เข้ากับองค์กรธุรกิจใดๆ ได้โดยเฉพาะ จึงเป็นการประหยัดเวลาและค่าใช้จ่ายในการพัฒนาระบบสารสนเทศเพื่อช่วยในการดำเนินการและบริหารองค์กร

วิวัฒนาการที่สำคัญอย่างยิ่งยวดของระบบจัดการฐานข้อมูลได้แก่ความสามารถของการใช้งานฐานข้อมูลผ่านอินเตอร์เน็ต ในยุคแรกของอินเตอร์เน็ต หน้าเว็บต่างๆ เป็นไฟล์ที่มีเนื้อหาแน่นอนตามที่ได้สร้างขึ้นโดย HTML แต่ในปัจจุบันเราสามารถสร้างเว็บไซต์ที่สามารถตอบสนองตามการร้องขอของผู้ใช้ที่แตกต่างกันจากฟอร์มของเว็บ ค้นคืนข้อมูลจากฐานข้อมูล ประมวลผลและนำเสนอในรูปแบบของเอกสาร HTML เพื่อสามารถแสดงผลบนเว็บบราวเซอร์ได้ การประยุกต์ใช้งานดังกล่าวเป็นที่นิยมแพร่หลาย ผู้ขายระบบจัดการฐานข้อมูลทุกรายได้เพิ่มความสามารถด้านการใช้งานฐานข้อมูลบนเว็บให้กับผลิตภัณฑ์ของตนเอง



ระบบจัดการฐานข้อมูลทวีความสำคัญและประโยชน์มากขึ้นเรื่อยๆ เนื่องจากมีการเรียกใช้ข้อมูลและสารสนเทศผ่านทางเว็บมากขึ้นอย่างไม่หยุดยั้ง นอกจากข้อมูลที่เป็นภาพและตัวหนังสือแล้ว ปัจจุบันฐานข้อมูลมัลติมีเดียถูกใช้งานอย่างแพร่หลายเพื่อรองรับการดูภาพยนต์ ทีวี วีดีทัศน์ ตลอดจนฟังเพลงและข่าวสารผ่านทางเครือข่ายคอมพิวเตอร์ การใช้งานฐานข้อมูลเพื่อรองรับห้องสมุดดิจิตัล การใช้งานเพื่อรองรับโครงการวิจัยทางวิทยาศาสตร์ขนาดใหญ่เช่นแผนที่ยีนของมนุษย์ โครงการ Earth Observation System ขององค์การนาซ่า การใช้เพื่อรองรับกระบวนการตัดสินใจทางธุรกิจ การนำมาใช้ประกอบการขุดค้นข้อมูลที่น่าสนใจเพื่องานวิชาการและธุรกิจ ในแวดวงธุรกิจตลาดของระบบจัดการฐานข้อมูลยังมีขนาดที่ใหญ่เป็นอันดับต้นๆ ทั้งหมดที่กล่าวมานี้เป็นเพียงตัวอย่าง

บางส่วนของบทบาทที่สำคัญของระบบจัดการฐานข้อมูล การศึกษาเกี่ยวกับฐานข้อมูลจึงเป็นสิ่งที่น่าสนใจและมีประโยชน์อย่างกว้างขวาง

1.4 ระบบไฟล์และระบบจัดการฐานข้อมูล

เพื่อให้เข้าใจถึงความจำเป็นในการใช้งานระบบจัดการฐานข้อมูล เราสามารถยกตัวอย่างให้เห็นได้ดังนี้ บริษัทแห่งหนึ่งมีข้อมูลอยู่เป็นจำนวนมาก ประมาณ 500 GB ซึ่งเป็นข้อมูลของพนักงาน แผนก สินค้า การขาย ฯลฯ ข้อมูลถูกใช้งานโดยพนักงานหลายคนพร้อมกัน การเรียกใช้ข้อมูลจะต้องได้รับการตอบสนองอย่างรวดเร็ว การเปลี่ยนแปลงข้อมูลใดๆ โดยผู้ใช้งานคนละคนจะต้องยังคงความถูกต้องของข้อมูล และจะต้องมีการป้องกันการเข้าถึงข้อมูลบางข้อมล เช่น ข้อมูลเงินเดือนของพนักงาน เป็นต้น

เราสามารถลองที่จะจัดการข้อมูลดังกล่าวโดยจัดเก็บข้อมูลรูปแบบไฟล์ปกติ ซึ่งวิธีดังกล่าวมีข้อเสียที่เกิดขึ้นดังนี้

- เราไม่สามารถจัดเก็บข้อมูลไว้ในหน่วยความจำหลักได้ทั้ง 500 GB ข้อมูลจะต้องถูกจัดเก็บอยู่บนหน่วยความจำสำรองได้แก่ฮาร์ดดิกส์ หรือเทปแม่เหล็ก เมื่อจำเป็นจะต้องใช้งานข้อมูลใดๆ ข้อมูลส่วนที่เกี่ยวข้องจะถูกอ่านขึ้นสู่หน่วยความจำหลัก ซึ่งจะต้องมีการเขียนโปรแกรมเพื่อรองรับกระบวนการนี้เป็นพิเศษ
- ถึงแม้เราจะมีหน่วยความจำหลักขนาด 500 GB แต่ระบบปฏิบัติการแบบ 32 บิตที่ใช้กันแพร่หลายสามารถอ้างอิงหน่วยความจำได้เพียง 4 GB จึงต้องมีการเขียนโปรแกรมเพื่อรองรับกระบวนการนี้เป็นพิเศษเช่นเดียวกัน
- การตอบคำถามจากข้อมูล การจัดเก็บ การแก้ไขใดๆ จำเป็นต้องเขียนโปรแกรมเฉพาะกับข้อมูลชุดนั้นๆ ซึ่งโปรแกรมจะมีความซับซ้อนเนื่องจากขนาดที่ใหญ่โตของข้อมูล
- จะต้องมีการป้องกันความผิดพลาดในกรณีที่มีการแก้ไขข้อมูลเดียวกันพร้อมๆ กัน การจัดการเหตุการณ์ดังกล่าวในกรณีที่ใช้ระบบไฟล์ข้อมูล ยิ่งทวีความซับซ้อนในการเขียนโปรแกรม
- ระบบต้องรับประกันว่าข้อมูลจะคืนสู่สภาพที่ควรจะถูกต้องจากการแก้ไขครั้งสุดท้ายก่อนเกิดการล้มเหลวของระบบ
- ปกติแล้วระบบไฟล์จะมีการป้องกันการเข้าถึงไฟล์โดยการใช้รหัสผ่านสำหรับผู้ใช้ก่อนเข้าใช้ระบบหรือใช้รหัสผ่านเพียงรหัสเดียวสำหรับไฟล์ทั้งหมด ซึ่งวิธีการป้องกันไฟล์ดังกล่าวไม่เพียงพอต่อการใช้งานในรูปแบบของการอนุญาตให้ผู้ใช้แต่ละคนสามารถเข้าถึงข้อมูลชุดใดชุดหนึ่งเท่านั้นในชุดข้อมูลที่มีอยู่ทั้งหมด

ฝ่ายบุคคล

รหัสพนักงาน | ชื่อพนักงาน | แผนก | เงินเดือน | 00001 | นายสมชาย จดปลายเท้า | ฝ่ายขาย | 30000 | 00002 | นางสาวสมศรี จดปลายเท้า | ฝ่ายบุคคล | 12000 | ...

ฝ่ายขาย

รหัสพนักงาน | ชื่อพนักงาน | ยอดขาย | 00001 | นายสมชาย จดปลายเท้า | 300000 | 00004 | นางสาวสมหญิง จดปลายเท้า | 250000 | ...

ตัวอย่างไฟล์ข้อมูลพนักงาน 2 ไฟล์อยู่ในแต่ละแผนกแยกจากกัน จะพบว่าเรามีความจำเป็นต้องเขียนโปรแกรมสำหรับเข้าถึงระเบียนด้วยตนเองตามรูปแบบของไฟล์ ในกรณีตัวอย่างนี้เราใช้เครื่องหมาย "|" เป็นตัวแบ่งฟิลด์ เราจะไม่สามารถใช้โปรแกรมนี้อ่านไฟล์อื่นๆ ที่ไม่ได้ใช้เครื่องหมายแบ่งฟิลด์เดียวกัน นอกจากนี้ถ้ามีความจำเป็นที่ไฟล์จะต้องจัดเก็บฟิลด์เพิ่มเติมเช่น หมายเลขประกันสังคม เราจะต้องเขียนโปรแกรมใหม่ ไฟล์ข้อมูลซ้ำซ้อนกัน 2 ไฟล์นี้ยังจะก่อให้เกิดปัญหาความขัดแย้งกันของข้อมูลในกรณีที่เกิดการแก้ไขข้อมูลของพนักงานคนใดในไฟล์หนึ่ง แต่ไม่ได้แก้ในอีกไกล์หนึ่ง

คำตอบของข้อกำหนดของระบบทั้งหมดที่กล่าวมาคือการใช้ระบบจัดการฐานข้อมูล การจัดเก็บข้อมูลลงในฐานข้อมูลแทนที่จะจัดเก็บลงในไฟล์ธรรมดา เราสามารถที่จะจัดการก้อนข้อมูลได้อย่างมีประสิทธิภาพมากขึ้น ระบบจัดการฐานข้อมูลเป็นสิ่งที่จะต้องใช้งานอย่างหลีกเลี่ยงไม่ได้สำหรับหน่วยงานโดยทั่วไปที่ข้อมูลมีขนาดหลายร้อย GB (ซึ่งเป็นขนาดของข้อมูลโดยทั่วๆ ไปในปัจจุบัน) และการใช้งานพร้อมๆ กันของผู้ใช้เป็นพันคน ในหัวข้อต่อๆ ไปจะได้อธิบายถึงข้อดีและสถาปัตยกรรมของฐานข้อมูลเพื่อให้เข้าใจได้ถึงหลักการทำงานของฐานข้อมูลที่จะรองรับความต้องการจัดการข้อมูลได้ดีกว่าไฟล์อย่างไร

1.5 ข้อดีของระบบจัดการฐานข้อมูล

ระบบจัดการฐานข้อมูลมีข้อดีดังต่อไปนี้

- **ความเป็นอิสระของข้อมูล (data independence)** ความเป็นอิสระของข้อมูลหมายถึงการที่รูปแบบของข้อมูลไม่ผูกติดกับสิ่งใดสิ่งหนึ่งในกระบวนใช้งานฐานข้อมูล เช่น การที่นักพัฒนาระบบจัดพัฒนาระบบสารสนเทศหรือโปรแกรมประยุกต์ใดๆ เพื่อใช้ประโยชน์ ชุดคำสั่งของโปรแกรมไม่ควรผูกติดอยู่กับรูปแบบของการจัดเก็บข้อมูลในระดับฮาร์ดแวร์ แต่ควรมีกลไกในการขั้นกลางระหว่างการจัดเก็บข้อมูลกับการใช้งานข้อมูลในระดับสูง ซึ่งสามารถสร้างเป็นระดับๆ ของกลไกได้โดยเรียกว่าระดับทางนามธรรมของฐานข้อมูล (level of abstractions)
- **การเข้าถึงข้อมูลอย่างมีประสิทธิภาพ (efficient data access)** ระบบจัดการฐานข้อมูลใช้เทคนิคที่ซับซ้อนและชาญฉลาดที่จะจัดเก็บข้อมูลและเรียกใช้ข้อมูลอย่างมีประสิทธิภาพและรวดเร็ว เทคนิคดังกล่าวจำเป็น

อย่างยิ่งโดยเฉพาะสำหรับการจัดการข้อมูลที่อยู่ในหน่วยความจำสำรองที่ทำงานได้ช้ากว่าหน่วยความจำหลักอย่างมาก

- **บูรณภาพและความมั่นคงปลอดภัยของข้อมูล (data integrity and security)** ตราบเท่าที่ผู้ใช้จะต้องใช้งานข้อมูลผ่านระบบจัดการฐานข้อมูล เราสามารถที่จะกำหนดข้อกำหนดและควบคุมให้ข้อมูลมีบูรณภาพ ซึ่งหมายถึงการที่ข้อมูลมีความถูกต้องตรงตามสภาพที่ควรจะเป็นในชีวิตจริงเช่น การทำการขึ้นเงินเดือนให้พนักงาน ระบบจะสามารถตรวจสอบได้ว่าเงินเดือนของพนักงานไม่เกินไปจากงบประมาณของหน่วยงาน หรือการแก้ไขสถานะภาพสมรสของพนักงานคนใดๆ จะแก้ไขให้เป็นหม้ายไม่ได้หากยังไม่เคยแต่งงานเป็นต้น นอกจากนี้การเข้าถึงฐานข้อมูลโดยผ่านทางระบบจัดการฐานข้อมูลเท่านั้นยังอนุญาตให้มีการกำหนดระดับการเข้าถึงของข้อมูลว่าผู้ใช้คนใดหรือกลุ่มใด สามารถมองเห็น ข้อมูลชุดใดหรือทำกิจกรรมใดๆ กับฐานมูลนั้นได้บ้าง
- **การบริหารข้อมูล (data administrator)** เมื่อผู้ใช้หลายคน ใช้งานข้อมูลชุดเดียวกัน การจัดการข้อมูลโดยรวมศูนย์จะทำให้การบริหารข้อมูลมีประสิทธิภาพมากขึ้น ผู้ที่มีความชำนาญในการจัดการฐานข้อมูลจะทราบว่าจะปรับแต่งฐานข้อมูลอย่างไรให้ตรงตามความต้องการข้อมูลใช้ ในแง่ของการควบคุมการเข้าถึงข้อมูลชุดต่างๆ ของผู้ใช้อย่างเหมาะสม การลดความซ้ำซ้อนของข้อมูลที่ใช้งานร่วมกัน ตลอดจนการปรับแต่งฐานข้อมูลเพื่อให้สามารถเรียกใช้งานข้อมูลได้รวดเร็วขึ้น
- **การจัดการการใช้งานข้อมูลพร้อมกันและการคืนสภาพจากความล้มเหลวของระบบ (concurrent access and crash recovery)** ระบบจัดการฐานข้อมูลมีความสามารถในการจัดลำดับการทำงานของธุรกรรมที่เข้ามาพร้อมๆ กันให้เสมือนหนึ่งว่าฐานข้อมูลมีการใช้งานจากผู้ใช้เพียงคนเดียวและการปรับปรุงข้อมูลใดๆ มีความถูกต้องอยู่เสมอ ตลอดจนป้องกันข้อมูลสูญหายและผิดพลาดในกรณีที่ระบบล้มเหลว
- **ลดเวลาในการพัฒนาระบบที่ใช้งานฐานข้อมูล (reduced application development time)** ระบบจัดการฐานข้อมูลที่มีให้เลือกใช้มีความสามารถในการรองรับการทำงานที่ใช้กันโดยทั่วไป ดังนั้นจึงมีเครื่องมือและรูปแบบการเรียกใช้งานซึ่งมองได้ว่าเป็นการเรียกใช้งานฐานข้อมูลในระดับสูง เช่นมีมาตรฐาน SQL ในการจัดการข้อมูล ทำให้ไม่ต้องเขียนโปรแกรมเฉพาะที่ยุ่งยากซับซ้อนสำหรับข้อมูลแต่ละชุดเอง นอกจากนี้การจัดการข้อมูลลงในอุปกรณ์จัดเก็บยังทำงานอย่างถูกต้องและมีประสิทธิภาพอยู่แล้วโดยระบบจัดการฐานข้อมูล นักพัฒนาสามารถลดเวลาในการพัฒนาและแก้ไขข้อผิดพลาดจากการพัฒนาลงได้อย่างมาก

อย่างไรก็ตาม แม้ว่าระบบจัดการฐานข้อมูลจะมีประโยชน์มากมาย แต่เราจำเป็นต้องใช้งานระบบจัดการฐานข้อมูลเสมอไปหรือไม่ คำตอบคือไม่จำเป็นเสมอไป ในกรณีที่เราไม่จำเป็นต้องใช้ความสามารถที่กล่าวมาของระบบจัดการฐานข้อมูลเลยเช่น ไม่จำเป็นต้องใช้การจัดการกรณีที่มีผู้ใช้งานฐานข้อมูลพร้อมๆ กันหลายคน หรือมีผู้ใช้หลายระดับเข้าถึงข้อมูล เป็นต้น นอกจากนี้การวิเคราะห์ข้อมูลบางอย่างที่ซับซ้อน อาจไม่สามารถจัดเก็บได้ในฐานข้อมูล และในบางกรณีเราอาจมีความต้องการพิเศษในการใช้งานข้อมูล เช่นจะต้องมีความเร็วสูงกว่าที่ระบบจัดการ

ฐานข้อมูลจะสามารถทำได้ เราก็ไม่จำเป็นต้องใช้งานระบบจัดการฐานข้อมูล กล่าวโดยสรุปได้ว่าเราอาจจะไม่จำเป็นต้องใช้งานระบบจัดการฐานข้อมูลในกรณีที่เราไม่ต้องการความสามารถที่สูงเกินความจำเป็น ซึ่งอาจทำให้การใช้งานระบบจัดการฐานข้อมูลไม่คุ้มค่ากับค่าใช้จ่าย รวมถึงในกรณีที่เราไม่สามารถใช้งานระบบจัดการฐานข้อมูลได้ในกรณีที่ข้อมูลไม่อยู่ในรูปแบบที่ระบบจัดการฐานข้อมูลจัดการได้ รวมถึงความต้องการพิเศษที่เราจะต้องพัฒนาระบบสำหรับจัดการกับข้อมูลเอง แต่ในทางตรงกันข้าม การจัดการข้อมูลที่ซับซ้อนและมีขนาดใหญ่มักจะหลีกเลี่ยงไม่ได้ที่จะต้องใช้งานระบบจัดการฐานข้อมูล

1.6 การจัดเก็บข้อมูลในระบบจัดการฐานข้อมูลและสถาปัตยกรรมฐานข้อมูล

ผู้ใช้ข้อมูลโดยทั่วไปจะมองข้อมูลในมุมมองของการใช้งานในชีวิตจริง ในขณะที่การจัดเก็บข้อมูลลงในฐานข้อมูลเป็นวิธีในการที่จะทำอย่างไรเพื่อจัดเก็บข้อมูลในมุมมองนั้นๆ ลงในฐานข้อมูลให้ได้ เช่น ในมหาวิทยาลัยจะประกอบไปด้วยข้อมูลที่เกี่ยวกับนักศึกษา อาจารย์ และรายวิชา ตลอดจนกิจกรรมต่างๆ ที่เกิดขึ้นในมหาวิทยาลัยที่จะต้องจดบันทึกข้อมูลไว้ ในขณะที่การจัดเก็บข้อมูลลงในฐานข้อมูลจะอยู่ในรูปของการมองผู้กระทำกิจกรรมต่างๆ เป็นเอนทิตี การเก็บข้อมูลที่อธิบายเอนทิตีต่างๆ และจัดเก็บข้อมูลด้านความสัมพันธ์ต่างๆ ของเอนทิตีไว้

การที่จะจำลองมุมมองข้อมูลในโลกแห่งความเป็นจริงมาเป็นรูปแบบของการจัดเก็บข้อมูลในเครื่องคอมพิวเตอร์จำเป็นต้องแทนข้อมูลและเชื่อมโยงกันโดยอาศัยแบบจำลองข้อมูล (data model) ซึ่งแบบจำลองข้อมูลนี้ จะใช้เป็นเครื่องมือที่แทนรูปแบบของการจัดเก็บข้อมูลในระดับสูง และจะซ่อนรายละเอียดอื่นๆ ที่เกี่ยวข้องกับการจัดเก็บข้อมูลในระดับล่างที่เกี่ยวข้องกับการเก็บข้อมูลลงในอุปกรณ์จัดเก็บจริงๆ ซึ่งในปัจจุบัน แบบจำลองข้อมูลที่ใช้กันแพร่หลายที่สุดได้แก่แบบจำลองข้อมูลเชิงสัมพันธ์ (relational database) ซึ่งเป็นแบบจำลองหลักที่ใช้ศึกษาในรายวิชา

อย่างไรก็ตาม แม้แบบจำลองข้อมูลจะซ่อนรายละเอียดการจัดเก็บลองการจัดการกับอุปกรณ์จัดเก็บข้อมูล แต่แบบจำลองข้อมูลมีความเคียงกับการวิธีการจัดเก็บข้อมูลของระบบจัดการฐานข้อมูลมากกว่าการมองข้อมูลในโลกแห่งความเป็นจริง เราจึงต้องอาศัยเครื่องมืออื่นได้แก่แบบจำลองข้อมูลเชิงความหมาย (semantic data mode) ซึ่งเป็นธรรมชาติใกล้เคียงกับข้อมูลในมุมมองของการใช้งานในโลกแห่งความเป็นจริงมากขึ้น โดยแบบจำลองข้อมูลเชิงความหมายนี้จะใช้เป็นเครื่องมือเริ่มต้นในการแทนกิจกรรมและผู้เกี่ยวข้องต่างๆ ให้อยู่ในรูปแบบแผนภาพ หรือรูปแบบอื่นๆ ที่จะสามารถจดบันทึกและแลกเปลี่ยนทำความเข้าใจกันได้ แบบจำลองข้อมูลเชิงความหมายนี้สามารถแทนข้อกำหนดต่างๆ ของกิจกรรมในองค์กรได้ แต่ระบบจัดการฐานข้อมูลไม่สามารถรองรับของกำหนดนั้นๆ โดยตรง เพราะระบบจัดการฐานข้อมูลเชิงสัมพันธ์ใช้แบบจำลองข้อมูลเชิงสัมพันธ์ซึ่งต่างจากแบบจำลองในระดับสูงอย่างแบบจำลองข้อมูลเชิงความหมาย อย่างไรก็ตามเราแบบจำลองข้อมูลทั้งสองมีความสัมพันธ์กันและสามารถออกแบบให้สอดคล้องกันได้ เพื่อให้ระบบจัดการฐานข้อมูลเชิงสัมพันธ์สามารถรองรับการใช้งานข้อมูลในกิจการขององค์กรได้อย่างถูกต้อง

รูปแบบของแบบจำลองข้อมูลเชิงความหมายที่ได้รับการใช้งานอย่างกว้างขวางมีชื่อว่าแบบจำลองเอนทิตีและความสัมพันธ์ (entity-relationship model—ER model) แบบจำลองดังกล่าวทำให้เราสามารถแทนเอนทิตีและ

ความสัมพันธ์ของเอทิตีต่างๆ เป็นแผนภาพ แบบจำลองเอนทิตีและความสัมพันธ์เป็นเครื่องมือที่ใช้ในการออกแบบฐานข้อมูลขั้นต้นก่อนที่จะปรับแผนภาพเอนทิตีและความสัมพันธ์ไปเป็นแบบจำลองเชิงสัมพันธ์และจัดการในรูปแบบของฐานข้อมูลเชิงสัมพันธ์ในที่สุด การออกแบบฐานข้อมูลด้วยแบบจำลองเอนทิตีและความสัมพันธ์นั้นจะกล่าวถึงในบทที่ 3

1.6.1 แบบจำลองเชิงสัมพันธ์ (relational model)

ในหัวข้อนี้จะแนะนำแบบจำลองเชิงสัมพันธ์โดยย่อ โครงสร้างที่เป็นหลักของแบบจำลองเชิงสัมพันธ์ได้แก่ **รีเลชัน (relation)** ซึ่งรีเลชันนี้สามารถเปรียบได้กับชุดของระเบียน **(records)**

ในแบบจำลองข้อมูลนั้นจะมีการนิยามหรืออธิบายข้อมูลต่างๆ ไว้ ซึ่งเราเรียกการนิยามข้อมูลในแบบจำลองข้อมูลนี้ว่าเค้าร่าง (schema) ซึ่งเค้าร่างในแบบจำลองเชิงสัมพันธ์ประกอบไปด้วยชื่อของความสัมพันธ์ ชื่อของเขตข้อมูล หรือลักษณะประจำ หรือสดมภ์ (field or attribute or column ซึ่งจะใช้คำว่าฟิลด์ แอทริบิวท์ และคอลัมน์ตามลำดับแทนเนื่องจากสามารถเข้าใจได้ง่ายกว่า) ซึ่งทั้งฟิลด์ แอทริบิวท์ และคอลัมน์ ต่างก็คือการนิยามข้อมูลในรีเลชันเช่นเดียวกัน นอกจากนี้เค้าร่างในแบบจำลองเชิงสัมพันธ์ยังประกอบไปด้วยชนิดของข้อมูลแต่ละฟิลด์ ตัวอย่างต่อไปนี้แสดงรีเลชันของข้อมูลตัวอย่างของนักศึกษาในมหาวิทยาลัย ซึ่งรีเลชันมีเค้าร่างดังต่อไปนี้

Student(*sid:* string, *name:* string, *login:* string, *age:* integer, *gpa:* real)

Student แสดงถึงความสัมพันธ์ของนักศึกษา sid คือฟิลด์รหัสนักศึกษามีชนิดเป็นสตริงหรือสายอักขระ name คือชื่อจริงของนักศึกษา login คือชื่อที่ใช้ในการเข้าใช้งานระบบ age เป็นอายุซึ่งเป็นข้อมูชนิดจำนวนเต็ม และ gpa คือเกรดเฉลี่ยของนักศึกษามีชนิดของข้อมูลเป็นจำนวนจริง ทั้งนี้ในทางปฏิบัติแล้วเราไม่ควรออกแบบฐานข้อมูลโดยมีฟิลด์ที่สามารถเปลี่ยนแปลงข้อมูลไปตามกาลเวลาเช่นอายุ เนื่องจากในปีถัดไปอายุของนักศึกษาจะเพิ่มขึ้นได้ เราจึงควรกำหนดฟิลด์ดังกล่าวเป็นวันเดือนปีเกิด แต่เพื่อความง่ายต่อการแสดงตัวอย่าง จึงยกตัวอย่างฟิลด์ที่เป็นอายุ

กรณีตัวอย่าง (instance) ของข้อมูล 5 ตัวอย่างจากเค้าร่าง Student สามารถแสดงได้ดังนี้

sid	name	login	age	gpa
B5075666	สมชาย	somchai@it	18	3.44
B5075688	สมศรี	somsri@com	18	3.21
B5075650	สมศรี	somsri@it	19	3.82
B5075831	สมศักดิ์	somsak@med	11	1.80
B5075832	สมน้ำหน้า	somnamna@math	12	2.00

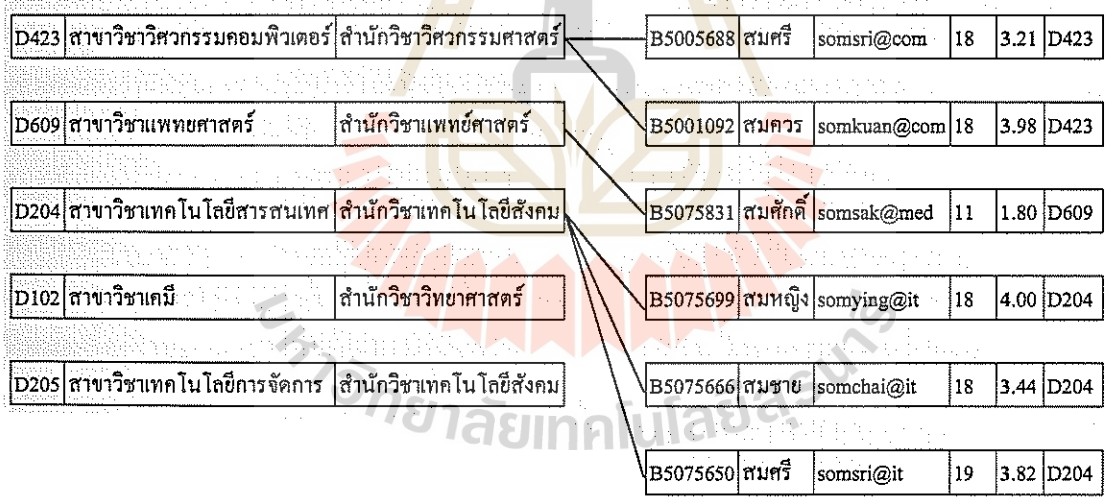
ข้อมูลที่ปรากฏในแต่ละแถวเป็นระเบียนข้อมูลของนักศึกษาแต่ละคน ข้อมูลที่กำหนดยังไม่ครบถ้วนเช่น ส่วนสูงของนักศึกษาหรือน้ำหนัก แต่ว่าข้อมูลเท่าที่มีอยู่เพียงพอต่อการใช้งานในระบบฐานข้อมูลมหาวิทยาลัย ของมูลทุกแถวเป็นไปตามรูปแบบของเค้าร่าง Student ดังนั้นเราสามารถมองได้ว่าเค้าร่างก็คือแม่แบบข้อมูลนั่นเอง

เราสามารถที่จะนิยามหรือระบุข้อกำหนดต่างๆ ของข้อมูลนักศึกษาให้เฉพาะเจาะจงได้อีกโดยการกำหนดกฎ **ความคงสภาพ (integrity constraints)** ซึ่งใช้ในการกำหนดเงื่อนไข ข้อบังคับ หรือกฎเกณฑ์ที่ข้อมูลในระเบียนนั้นต้องเป็นไปตามกฎ เช่นต้องไม่รหัสนักศึกษาที่ซ้ำกัน การกำหนดเงื่อนไขดังกล่าวควรที่จะสามารถระบุได้ในแบบจำลองข้อมูล

แบบจำลองข้อมูลแบบอื่นๆ

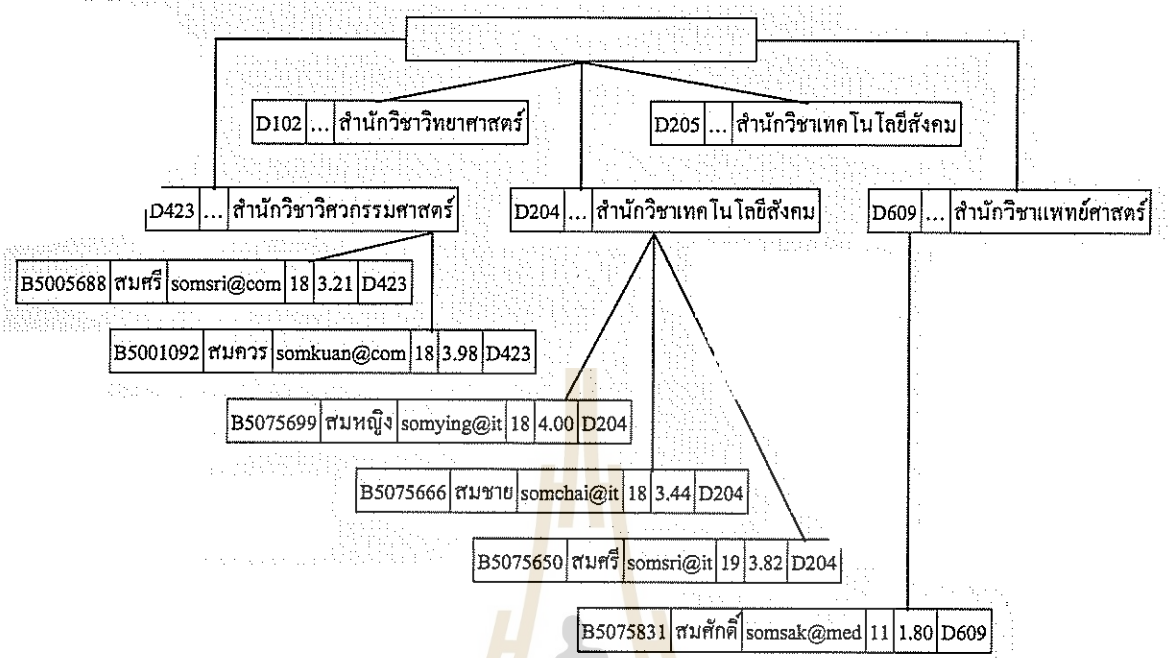
นอกจากแบบจำลองข้อมูลเชิงสัมพันธ์ซึ่งใช้ในระบบจัดการฐานข้อมูล IBM DB2, Oracle, Microsoft Access, Microsoft SQL Server, MySQL ฯลฯ แล้ว ยังมีแบบจำลองข้อมูลอื่นๆ ที่ใช้ประโยชน์บ้างในปัจจุบันแต่ไม่แพร่หลายเทียบเท่าแบบจำลองเชิงสัมพันธ์ ซึ่งสามารถใช้งานได้ดีในงานบางประเภทรวมถึงอาจเป็นระบบที่มีขนาดใหญ่โตและใช้มานานแล้ว แบบจำลองข้อมูลที่สำคัญที่ควรกล่าวถึงได้แก่แบบจำลองเชิงลำดับชั้น (hierarchical model) ซึ่งประยุกต์ใช้โดย IBM IMS แบบจำลองเชิงเครือข่าย (network model) ซึ่งใช้ใน IDS และ IDMS แบบจำลองเชิงวัตถุ (object-oriented model) มีใช้ใน Objectstore Versant ฯลฯ แบบจำลองเชิงวัตถุ-สัมพันธ์รองรับโดย IBM DB2, Oracle, Microsoft SQL Server ฯลฯ ต่อไปนี้จะเป็นการแนะนำฐานข้อมูลเชิงเครือข่ายและเชิงลำดับชั้นเบื้องต้น

- **แบบจำลองฐานข้อมูลเชิงเครือข่าย**



แบบจำลองข้อมูลเชิงเครือข่ายนั้นมีข้อมูลจัดเก็บเป็นระเบียนดังตัวอย่าง ซึ่งข้อมูลจะมีการเชื่อมโยงกับข้อมูลที่เกี่ยวข้องในลักษระเซตของความสัมพันธ์ เราอาจกำหนดเซ็ตของความสัมพันธ์ในคอลัมน์ใหม่เช่นคอลัมน์ที่กำหนดสาขาวิชาที่นักศึกษาแต่ละคนสังกัด ตัวอย่างข้อมูลนั้นเป็นตัวอย่างจำนวนน้อย ถ้าข้อมูลมีจำนวนมาก รูปแบบของการเชื่อมโยงข้อมูลจะอยู่ในรูปของเครือข่าย หรือที่เรียกว่าโครงสร้างข้อมูลกราฟ ข้อมูลแต่ละระเบียนเป็นโหนด (nodes) ของกราฟ (graph) และการเชื่อมโยงระหว่างระเบียนคือขอบ (edges)

แบบจำลองฐานข้อมูลเชิงลำดับชั้น



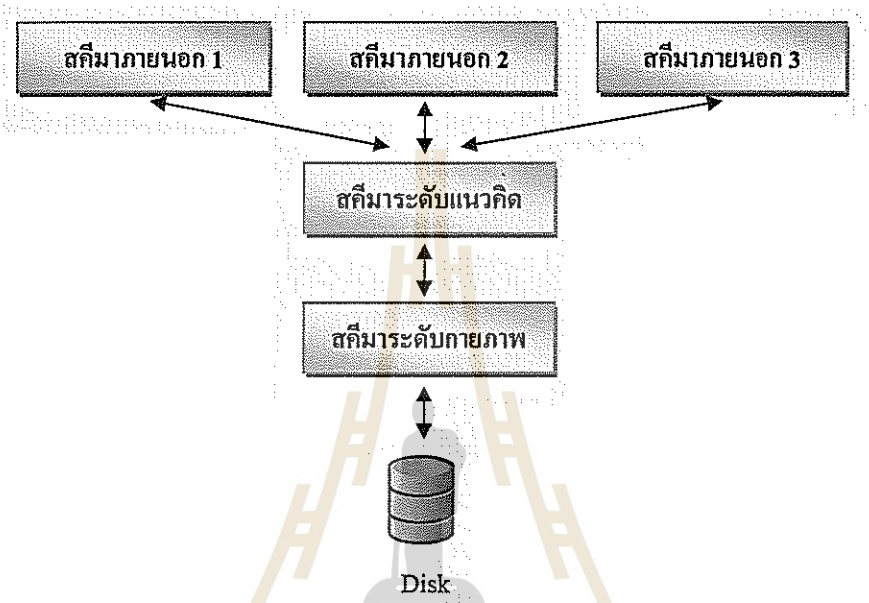
แบบจำลองฐานข้อมูลเชิงลำดับชั้นสามารถมองได้ว่าเป็นแบบจำลองเชิงเครื่อข่ายที่มีรูปแบบเฉพาะ โดยมีรูปแบบของการเชื่อมต่อเครือข่ายเป็นลำดับชั้น หรือเป็นโครงสร้างข้อมูลแบบต้นไม้ (tree) จากตัวอย่างข้อมูลจะพบว่าแบบจำลองเชิงลำดับชั้นจะอนุญาตให้มีโหนดที่เป็นต้นกำเนิดเพียงโหนดเดียว

แบบจำลองฐานข้อมูลทั้ง 2 แบบได้ถูกพัฒนาและใช้มากว่า 10 ปีก่อนที่จะมีการเสนอแบบจำลองข้อมูลเชิงสัมพันธ์ขึ้น ฐานข้อมูลที่ใช้แบบจำลองข้อมูลทั้ง 2 ได้ยกตัวอย่างไว้แล้ว แบบจำลองฐานข้อมูลทั้ง 2 สามารถรองรับการทำงานของข้อมูลได้ดีระดับหนึ่งเนื่องจากทำงานโดยใช้ระเบียนและรองรับความสัมพันธ์ระหว่างระเบียนแต่ข้อด้อยสำคัญของแบบจำลองทั้ง 2 คือการขาดความสามารถในการจัดการข้อมูลให้มีความถูกต้องโดยขาดการรองรับข้อกำหนดเพื่อความบูรณภาพของข้อมูล (integrity constraint) สำหรับแบบจำลองเชิงวัตถุนั้นรองรับข้อกำหนดทางธุรกิจต่างๆ แต่การแทนข้อมูลเพื่อจัดเก็บลงในระบบจัดการฐานข้อมูลนั้นไม่มีรูปแบบที่ชัดเจนและจะต้องอ้างอิงการจัดการข้อมูลเชิงระเบียนหรือเชิงสัมพันธ์

1.6.2 ระดับของฐานข้อมูล

ระดับชั้นของโครงสร้างในฐานข้อมูลสามารถแบ่งออกได้ 3 ระดับดังภาพในหน้าถัดไป ระดับของฐานข้อมูลแต่ละระดับนั้นเป็นระดับของการแทนการใช้งานข้อมูลจริงด้วยรูปแบบต่างๆ ที่เหมาะสมจนสามารถจัดการฐานข้อมูลได้อย่างมีประสิทธิภาพ แต่ละระดับบรรจุรายละเอียดของข้อมูลและสคีมาประจำแต่ละระดับ ฐานข้อมูลทั้ง 3 ระดับได้แก่ แนวคิด (conceptual) กายภาพ (physical) และภายนอก (external)

ภาษาที่ใช้ในการนิยามข้อมูล (data definition language—DDL) ใช้สำหรับนิยามและสร้างสคีมาในระดับภายนอกและระดับแนวคิด SQL เป็นภาษาที่ใช้งานมากที่สุดในการนิยามและจัดการข้อมูลซึ่งจะได้อธิบายถึงในบทที่ 7 และ 8 ระบบจัดการฐานข้อมูลเกือบทั้งหมดรองรับคำสั่ง SQL นอกจากนี้ยังใช้ SQL ในการจัดการข้อมูลในระดับกายภาพด้วย ต่อไปนี้เป็นการอธิบายถึงระดับของฐานข้อมูล



**

สคีมาระดับแนวคิด (Conceptual Schema)

สคีมาระดับแนวคิด ซึ่งในบางครั้งเรียกว่าสคีมาระดับตรรกกะ (logical schema) เป็นระดับของการอธิบายข้อมูลที่จะจัดเก็บให้มีรูปแบบสัมพันธ์กับแบบจำลองข้อมูลของระบบจัดการฐานข้อมูลโดยตรง ในระบบจัดการฐานข้อมูลเชิงสัมพันธ์ สคีมาระดับแนวคิดอธิบายถึงความสัมพันธ์ทั้งหมดที่จัดเก็บในฐานข้อมูล ตัวอย่างฐานข้อมูลของมหาวิทยาลัยต่อไปนี้บรรจุข้อมูลเกี่ยวกับเอนทิตีเช่น นักศึกษา (students) และอาจารย์ (faculty) และยังจัดเก็บความสัมพันธ์ (relationships) ของเอนทิตีต่างๆ เช่นนักศึกษาลงทะเบียนในรายวิชา (courses) เอนทิตีนักศึกษาทุกคนสามารถอธิบายข้อมูลได้จากระเบียนทั้งหมดในรีเลชัน เราสามารถแทนชุดข้อมูลของเอนทิตีและชุดข้อมูลของความสัมพันธ์ใดๆ ได้ด้วยรีเลชัน ซึ่งเราสามารถกำหนดสคีมาระดับแนวคิดได้ดังนี้

Students (*sid:* string, *name:* string, *login:* string, *age:* integer, *gpa:* real)

Faculty (*fid:* string, *fname:* string, *sal:* real)

Courses (*cid:* string, *cname:* string, *credits:* integer)

Rooms (*rno:* integer, *address:* string, *capacity:* integer)

Enrolled (*sid:* string, *cid:* string, *grade:* string)

Teaches (*fid:* string, *cid:* string)

Meets_In (*cid:* string, *rno*: integer, *time:* string)

การเลือกที่จะเก็บข้อมูลใดๆ และกำหนดขึ้นเป็นรีเลชัน ตลอดจนจะต้องจัดเก็บข้อมูลฟิลด์ใดบ้างในแต่ละรีเลชันนั้นไม่ชัดแจ้งและตายตัว เราสามารถสร้างสคีมาเชิงแนวคิดที่สมบูรณ์ดังที่ได้อธิบายในการออกแบบฐานข้อมูลในระดับแนวคิดในบทที่ 2

สคีมาระดับกายภาพ

สคีมาระดับกายภาพจะมีการระบุรายละเอียดเพิ่มเติมเกี่ยวกับการจัดเก็บข้อมูลในอุปกรณ์จัดเก็บ กล่าวคือ สคีมาระดับกายภาพเป็นสคีมาที่มีรองรับสคีมาในระดับแนวคิด และเพิ่มรายละเอียดในการจะจัดข้อมูลนั้นๆ รวมถึงการจัดการพิเศษใดๆ ในการจัดเก็บข้อมูลลงฮาร์ดดิสก์หรือเทปแม่เหล็ก

ตัวอย่างหนึ่งขององค์ประกอบของสคีมาระดับกายภาพได้แก่เราจะต้องเลือกการสร้างดรรชนีหรืออินเด็กซ์ (indexes) ซึ่งเป็นไฟล์เสริมในการที่จะเข้าถึงข้อมูลได้รวดเร็วขึ้นนอกจากที่เราจะต้องเลือกลักษณะการจัดเรียงของไฟล์บนฮาร์ดดิสก์ซึ่งเป็นสคีมาระดับกายภาพ สำหรับตัวอย่างสคีมาของฐานข้อมูลของมหาวิทยาลัยได้แก่

- จัดเก็บข้อมูลเป็นไฟล์ของระเบียนซึ่งไม่จำเป็นต้องเรียงลำดับข้อมูล นอกจากนี้อาจเป็นไฟล์หลายๆ ไฟล์จัดเก็บข้อมูลเป็นชุดๆ
- สร้างไฟล์ index เป็นไฟล์เสริม ซึ่งไฟล์ index นี้ใช้สำหรับเพิ่มความเร็วในการเข้าถึงข้อมูล ไฟล์ index จัดเก็บข้อมูลที่เรียงลำดับตามคอลัมน์ใดๆ พร้อมทั้งตำแหน่งของระเบียนที่สัมพันธ์กับข้อมูลคอลัมน์ที่ทำ index เช่น index ของคอลัมน์แรกในรีเลชันนักศึกษา อาจารย์ และห้องเรียน index ของคอลัมน์เงินเดือนของอาจารย์ index ของความจุของห้องเรียน

index เป็นไฟล์ที่เรียงลำดับข้อมูลซึ่งทำให้เราเข้าถึงข้อมูลได้รวดเร็วขึ้นมาก ทำให้เข้าถึงที่อยู่ของข้อมูลจริงได้รวดเร็วจำเป็นสำหรับข้อมูลจำนวนมหาศาลในอุปกรณ์หน่วยความจำสำรอง ซึ่งหลักการของ index จะได้กล่าวถึงในการสร้าง index ด้วย SQL ในบทที่ 7 การได้มาซึ่งสคีมาทางกายภาพที่ดีควรทำตามระเบียบวิธีในการออกแบบสคีมาระดับกายภาพ ซึ่งได้อธิบายไว้ในบทที่ 2

สคีมาระดับภายนอก

สคีมาระดับภายนอกหมายเป็นการระบุหรืออธิบายข้อมูลที่ใช้งานในเชิงแบบจำลองข้อมูลเช่นเดียวกันกับสคีมาระดับแนวคิด และมักจะมีโครงสร้างชนิดเดียวกันกับสคีมาระดับแนวคิดเช่นเป็นสคีมาในรูปแบบฐานข้อมูลเชิงสัมพันธ์ แต่สคีมาระดับภายนอกเป็น**มุมมองหรือวิว (views)** ของข้อมูลที่ผู้ใช้แต่ละคน แต่ละกลุ่มจะมองเห็น ในฐานข้อมูลแต่ละฐานข้อมูลจะมีสคีมาระดับแนวคิดและระดับกายภาพเพียงชุดเดียวเพราะข้อมูลที่จัดเก็บจะอยู่ในรูปของรีเลชันที่ออกแบบไว้ตายตัว แต่สคีมาระดับภายนอกเราสามารถสร้างได้โดยการสร้างวิวของผู้ใช้แต่ละคน โดยวิวนั้นเสมือนกับเป็นรีเลชันเช่นเดียวกัน แต่เป็นรีเลชันที่สร้างขึ้นมาจากรีเลชันที่มีอยู่จริงและจัดเก็บไว้ในฐานข้อมูล รีเลชันดังกล่าวอาจจะเกิดจากการคำนวณ หรือการกำหนดขอบเขตการมองเห็นข้อมูลของผู้ใช้ ซึ่งเราจะกล่าวถึงวิวในบทที่ ** การสร้างวิวด้วย SQL

ตัวอย่าง

สำหรับสคีมาระดับภายนอกนี้เราสามารถสร้างได้หลายๆ สคีมาตามข้อกำหนดที่ได้รับจากการวิเคราะห์ความต้องการการใช้ข้อมูลของผู้ใช้ ดังตัวอย่างรีเลชันข้อมูลรายวิชา Courseinfo ที่สร้างขึ้นจากวิวต่อไปนี้

Courseinfo (*cid:* string, *fname:* string, *enrollment:* integer)

Courseinfo เป็นวิวที่แสดงข้อมูลการของรายวิชาโดยระบุรหัสวิชา cid ชื่อของอาจารย์ผู้สอน fname และจำนวนนักศึกษาที่ลงทะเบียน วิวดังกล่าวอนุญาตให้นักศึกษาหรือผู้ใช้อื่นๆ ทราบว่าอาจารย์ท่านใดสอนในรายวิชาใดและมีผู้ลงทะเบียนเป็นจำนวนเท่าใด ผู้ใช้สามารถมองวิวดังกล่าเป็นรีเลชันหนึ่ง แต่รีเลชันดังกล่าวมีฟิลด์ enrollment เป็นค่าที่ได้จากการคำนวณว่ามีจำนวนระเบียนของการลงทะเบียนในแต่ละรายวิชาเท่าใด ซึ่งรีเลชัน Courseinfo มิได้มีข้อมูลอยู่จริงในฐานข้อมูล ถ้าเราสร้างรีเลชัน Courseinfo และบันทึกลงในฐานข้อมูลจริงๆ โดยจัดเก็บข้อมูลจำนวนผู้ลงทะเบียนด้วย จะเป็นการซ้ำซ้อนของข้อมูลซึ่งทำให้เสียพื้นที่จัดเก็บเพิ่มขึ้นโดยไม่จำเป็น และอาจจะนำไปสู่ปัญหาข้อมูลที่ขัดแย้งกัน ซึ่งสามารถยกตัวอย่างได้เช่น พิจารณาการเพิ่มระเบียนข้อมูลผู้ลงทะเบียนลงในรีเลชันต่อไปนี้

Enrolled (*sid:* string, *cid:* string, *grade:* string)

เราสามารถบันทึกการลงทะเบียนเรียนของนักศึกษาในรายวิชาโดยใช้รหัสนักศึกษา sid รหัสวิชา cid และอาจบันทึกเกรดที่ได้ในภายหลังในรีเลชัน Enrolled พิจารณาปัญหาว่าเมื่อนักศึกษาคนใดคนหนึ่งลงทะเบียนในรายวิชาใดๆ ระเบียนจะถูกบันทึกเพิ่มขึ้น 1 ระเบียน แต่ในถ้ามีการสร้างรีเลชัน Courseinfo จำนวนนักศึกษาที่ลงทะเบียนใหม่ไม่ได้มีการแก้ไขข้อมูลให้เพิ่มขึ้นอีก 1 คนจะยังมีค่าคงเดิมทำให้ข้อมูลนั้นขัดแย้งกัน เป็นตัวอย่างหนึ่งของสคีมาระดับภายนอก ซึ่งคำว่าภายนอกนั้นเป็นมุมมองจากการมองโดยใช้ระบบจัดการฐานข้อมูลเป็นตัวตั้ง

1.6.3 ความเป็นอิสระของข้อมูล

ความเป็นอิสระของข้อมูล (data independence) เป็นข้อดีข้อสำคัญของการใช้งานระบบจัดการฐานข้อมูล เราไม่จำเป็นต้องแก้ไขระบบสารสนเทศหรือโปรแกรมประยุกต์ที่เราพัฒนาขึ้นมาเรียกใช้งานฐานข้อมูลในกรณีที่มีการเปลี่ยนแปลงโครงสร้างหรือการจัดการไฟล์ข้อมูลบนอุปกรณ์หน่วยความจำสำรอง ความเป็นอิสระของข้อมูลเกิดขึ้นได้จากประโยชน์ของการสถาปัตยกรรม 3 ระดับของฐานข้อมูล ซึ่งเราจะได้รับประโยชน์อย่างมากโดยเฉพาะจากสคีมาระดับแนวคิด และสคีมาระดับภายนอก ซึ่งสามารถทำให้เกิดความเป็นอิสระของข้อมูลเชิงตรรกะ (logical data independence) และความเป็นอิสระของข้อมูลเชิงกายภาพ (physical data independence) ดังตัวอย่างต่อไปนี้

ตัวอย่าง

พิจารณาสคีมาระดับภายนอก Courseinfo แสดงข้อมูลรายวิชาที่เกี่ยวกับการลงทะเบียนในหัวข้อที่แล้ว ถ้าเกิดกรณีที่มีการเปลี่ยนแปลงของรีเลชันที่วิว Courseinfo อ้างอิงถึง เช่นจำนวนฟิลด์ในรีเลชัน Faculty เพิ่มขึ้น (Coruseinfo อ้างอิงถึง fname จากรีเลชัน Facluty) ได้แก่การเพิ่มฟิลด์คำนำหน้าหรือตำแหน่งทางวิชาการ title ผู้ใช้ซึ่งเข้าถึงข้อมูลผ่านทางวิวจะไม่มีผลกระทบใดๆ เนื่องจากเราไม่จำเป็นต้องแก้ไขสคีมาภายนอกซึ่งในที่นี้คือวิว

Courseinfo (*cid:* string, *fname:* string, *enrollment:* integer)

Faculty (*fid:* string, *fname:* string, *sal:* real)

Courseinfo (*cid:* string, *fname:* string, *enrollment:* integer)

Faculty (*fid:* string, *fname:* string, *sal:* real, *title:* string)

ตัวอย่าง

พิจารณาสคีมาระดับภายนอก Courseinfo แสดงข้อมูลรายวิชาที่เกี่ยวกับการลงทะเบียนในหัวข้อก่อนหน้า ถ้าเกิดกรณีที่มีการเปลี่ยนแปลงของรีเลชันที่วิว Courseinfo อ้างอิงถึง ซึ่งซับซ้อนกว่าในตัวอย่างที่แล้ว

Courseinfo (*cid:* string, *fname:* string, *enrollment:* integer)

Faculty (*fid:* string, *fname:* string, *sal:* real)

สมมติว่าเพื่อเพิ่มความเป็นส่วนตัวของข้อมูลอาจารย์อันได้แก่ข้อมูลเงินเดือน (sal) เราสามารถแยกข้อมูลของอาจารย์ออกเป็น 2 รีเลชันดังนี้

Courseinfo (*cid:* string, *fname:* string, *enrollment:* integer)

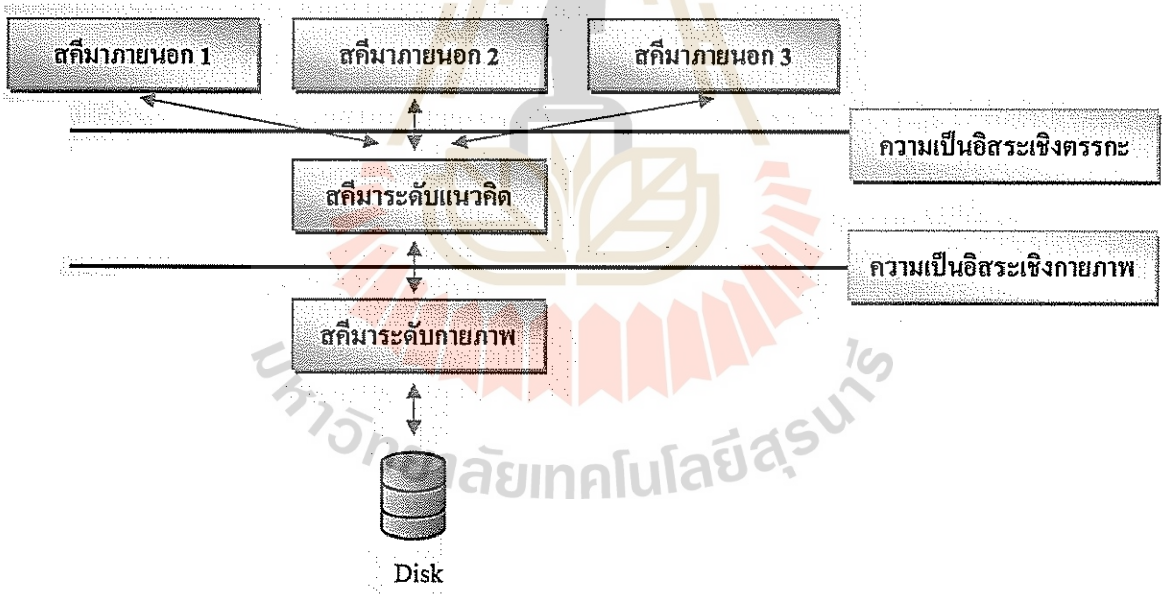
Faculty_public (*fid:* string, *fname:* string, *office:* integer)

Faculty_private (*fid:* string, *sal:* real)

ข้อมูลซึ่งเป็นส่วนตัวถูกแยกออกไปจัดเก็บในรีเลชันส่วนตัว Faculty_private ในขณะที่ข้อมูลที่ไม่ได้เป็นความลับจัดเก็บในชื่อของรีเลชัน Faculty_public การกระทำเช่นนี้เป็นการเพิ่มความเป็นส่วนตัวของข้อมูลมากขึ้น อย่างไรก็ตามแม้ว่าเราจะเปลี่ยนแปลงโครงสร้างของรีเลชันโดยการแตกรีเลชัน Faculty ออกเป็น 2 รีเลชันดังกล่าว ผู้ใช้ยังสามารถแสดงรายละเอียดข้อมูลการลงทะเบียนเรียนของรายวิชาได้จากวิว Courseinfo เช่นเดิม โดยไม่ต้องทำการเปลี่ยนแปลงสคีมาภายนอก เป็นผลให้เราไม่จำเป็นต้องแก้ไขโปรแกรมประยุกต์ที่เขียนขึ้นเพื่อนเรียกใช้ฐานข้อมูลนี้

ทั้ง 2 ตัวอย่างเป็นแสดงให้เห็นถึงการป้องกันรูปแบบการใช้ข้อมูลของผู้ใช้และสคีมาภายนอก จากการเปลี่ยนแปลงของสคีมาระดับแนวคิด ซึ่งเราเรียกคุณสมบัตินี้ว่าความเป็นอิสระของข้อมูลเชิงตรรกะ

นอกจากนี้ สถาปัตยกรรมของระบบจัดการฐานข้อมูลยังยอมให้สคีมาระดับกายภาพหรือรูปแบบการจัดการข้อมูลในอุปกรณ์หน่วยความจำสำรองสามารถเปลี่ยนแปลงได้ เช่นโครงสร้างไฟล์ข้อมูลที่เก็บระเบียน และการสร้างหรือลบ index ไม่มีผลกระทบต่อสคีมาระดับแนวคิดหรือรีเลชันยังคงเดิม ซึ่งเราเรียกคุณสมบัตินี้ว่าความเป็นอิสระของข้อมูลเชิงกายภาพ



1.7 ข้อคำถาม (queries) ในระบบจัดการฐานข้อมูล

ข้อดีที่สำคัญยิ่งอีกประการหนึ่งของระบบจัดการฐานข้อมูลนั้นคือระบบจัดการฐานข้อมูลเชิงสัมพันธ์มีความสามารถในเรียกใช้ แก้ไข และจัดการข้อมูลด้วยชุดคำสั่งที่มีโครงสร้างมาตรฐานและทรงพลัง ยกตัวอย่างเช่นเราต้องการทราบข้อมูลที่ใช้กันเป็นประจำจากฐานข้อมูลของมหาวิทยาลัยดังนี้

1) นักศึกษารหัส B5075666 ชื่ออะไร
   ตัวอย่างการใช้งาน ใช้สำหรับให้คะแนนนักศึกษา
2) นักศึกษาคนใดบ้างที่ลงทะเบียนเรียนในรายวิชา 204204 การออกแบบและพัฒนาฐานข้อมูล
   ตัวอย่างการใช้งาน ใช้สำหรับพิมพ์รายชื่อนักศึกษาเพื่อการลงชื่อเข้าเรียน
3) นักศึกษาที่ลงทะเบียนรายวิชา 204204 มีจำนวนเท่าใด
   ตัวอย่างการใช้งาน เพื่อให้นักศึกษาสามารถเลือกลงทะเบียนในกลุ่มที่ยังว่างอยู่
4) เกรดเฉลี่ยทั้งห้องเรียนของนักศึกษาในรายวิชา 204204 คือเท่าใด
   ตัวอย่างการใช้งาน สำหรับพิจารณาปรับปรุงรูปแบบการเรียนการสอนเพื่อเพิ่มคุณภาพการศึกษา
5) นักศึกษาที่มีเกรดเฉลี่ยน้อยกว่า 2.00 คือนักศึกษาคนใดบ้าง
   ตัวอย่างการใช้งาน พิจารณาสถานะการเป็นนักศึกษาของนักศึกษาว่าควรสิ้นสภาพหรือไม่

การสอบถามข้อมูลที่สนใจเพื่อนำไปใช้ประโยชน์ตามวัตถุประสงค์ และรองรับการทำงานขององค์กรใดๆ นั้น เราจะสร้างข้อคำถาม (queries) และป้อนเข้าสู่ระบบจัดการฐานข้อมูล ระบบจัดการฐานข้อมูลจะเป็นผู้จัดการขั้นตอนที่สลับซับซ้อนทั้งหมดในการประมวลผลเพื่อให้ได้มาซึ่งผลลัพธ์ตามข้อคำถาม ทั้งนี้ระบบจัดการฐานข้อมูลจะรองรับข้อคำถามที่มีโครงสร้างที่แน่นอน ซึ่งเป็นภาษาคอมพิวเตอร์ภาษาหนึ่ง ระบบจัดการฐานข้อมูลที่ใช้แบบจำลองข้อมูลแต่ละแบบมักจะมีภาษาสำหรับข้อคำถาม (query language) เฉพาะ สำหรับระบบจัดการฐานข้อมูลเชิงสัมพันธ์นั้นเราใช้ SQL เป็นภาษาหลักสำหรับข้อคำถามในการเรียกแสดงข้อมูล SQL เป็นภาษาที่มีโครงสร้างชัดเจน มีความสามารถและสิทธิภาพสูง ซึ่ง SQL มีพื้นฐานมาจากแคลคูลัสเชิงสัมพันธ์ (relational calculus) และพีชคณิตเชิงสัมพันธ์ (relational algebra) ซึ่งแคลคูลัสเชิงสัมพันธ์นั้นเป็นการใช้ทฤษฎีทางตรรกะของคณิตศาสตร์ในขณะที่พีชคณิตเชิงสัมพันธ์อยู่บนพื้นฐานของการใช้งานตัวดำเนินการบนรีเลชัน คล้ายกับกับตัวดำเนินการบนตัวเลข เช่นเครื่องหมายบวก ลบ ที่ดำเนินการบนตัวเลข แต่ในกรณีนี้เรานำมาใช้กับรีเลชัน เราสามารถใช้ SQL ในการแสดงข้อมูลได้ทั้ง SQL ที่มีพื้นฐานมาจากทั้งแคลคูลัสเชิงสัมพันธ์และพีชคณิตเชิงสัมพันธ์ เพื่อให้เข้าใจถึงหลักการสร้างข้อคำถาม SQL ที่ถูกต้อง การศึกษาแนวคิดและที่มาของ SQL จะทำให้การสร้าง SQL นั้นมีประสิทธิภาพ เราจึงจะกล่าวถึงแคลคูลัสเชิงสัมพันธ์และพีชคณิตเชิงสัมพันธ์ในบทที่ 6

ตัวอย่าง

ข้อคำถาม: นักศึกษารหัส B5075666 ชื่ออะไร

คำสั่ง SQL: SELECT name FROM student WHERE sid = 'B5075666'

ตัวอย่างดังกล่าวเป็นคำสั่ง SQL ที่แสดงชื่อของนักศึกษารหัส B5075666 ซึ่งเป็นการแสดงฟิลด์ name จากรีเลชัน Student สำหรับการสร้างข้อคำถามด้วย SQL อธิบายไว้โดยละเอียดในบทที่ 8

นอกจากระบบจัดการฐานข้อมูลจะรองรับการใช้ข้อคำถามที่มีโครงสร้างมาตรฐานแล้ว ความสามารถในการเลือกวิธีที่จะเข้าถึงตัวข้อมูลในอุปกรณ์จัดเก็บข้อมูลเพื่อนำมาประมวลผลเป็นผลลัพธ์ยังเป็นข้อดีสำคัญของระบบจัดการ

ฐานข้อมูล ทั้งนี้ระบบจัดการฐานข้อมูลจะนำข้อคำถามจากผู้ใช้มาวิเคราะห์ถึงข้อมูลที่จำเป็นในการสร้างผลลัพธ์ข้อมูลที่เกี่ยวข้องและต้องนำมาใช้ถูกจัดเก็บอยู่ในอุปกรณ์หน่วยความจำสำรองในรูปแบบของไฟล์ที่เป็นกลุ่มของระเบียนซึ่งอาจจะกระจายอยู่ในอุปกรณ์จัดเก็บ หรือแม้แต่ข้อมูลนั้นไม่กระจัดกระจาย แต่ความใหญ่โตของไฟล์ข้อมูลนั้นเราจำเป็นต้องมีเทคนิคในการเข้าถึงข้อมูลที่ชาญฉลาดเพื่อการสร้างผลลัพธ์ที่รวดเร็ว ระบบจัดการฐานข้อมูลจะเปรียบเทียบรูปแบบของการเข้าถึงข้อมูลแบบต่างๆ ที่เป็นไปได้ รวมถึงใช้ไฟล์พิเศษที่ช่วยในการเข้าถึงข้อมูลได้รวดเร็วได้แก่ไฟล์ indexและเลือกวิธีการที่ดีที่สุดในการเข้าถึงข้อมูลและสร้างผลลัพธ์

ข้อคำถามนั้นจัดอยู่ในกลุ่มของภาษาที่ใช้ในการเพิ่มข้อมูล แก้ไข ลบ และแสดงข้อมูล ซึ่งเราเรียกกลุ่มของภาษานี้ว่าภาษาที่ใช้ในการจัดการข้อมูล (data manipulation language—DML) DML เป็นกลุ่มของภาษาที่ SQL มีความสามารถในการรองรับ เราสามารถนำ SQL ที่เน้นการจัดการข้อมูลไปรวมกับภาษาคอมพิวเตอร์อื่นๆ เช่น C, C++, C#, Java, Visual Basic, ASP หรือ PHP ภาษาคอมพิวเตอร์เหล่านี้ความสามารถสูงในการจัดการสร้างโปรแกรมประยุกต์เพื่อรองรับกิจการใดๆ ขององค์กร การฝังคำสั่ง SQL เข้ากับภาษาหลักใดๆ นี้ทำให้เราสามารถสร้างโปรแกรมประยุกต์ได้อย่างรวดเร็วและมีประสิทธิภาพ เราเรียก SQL ในการประยุกต์แบบนี้ว่าภาษาย่อย (sub language) และภาษาคอมพิวเตอร์ที่ SQL ฝังอยู่ว่าภาษาหลัก (host language) ตัวอย่างการประยุกต์ใช้ SQL เป็นภาษาย่อยเพื่อประโยชน์ในการสร้างระบบสารสนเทศหรือโปรแกรมประยุกต์ ได้อธิบายไว้อย่างละเอียดในบทที่ 12 ในการสร้างเว็บที่ใช้งานระบบจัดการฐานข้อมูลซึ่งเป็นรูปแบบที่กำลังได้รับความนิยมในปัจจุบัน

1.8 การจัดการธุรกรรม

ระบบจัดการฐานข้อมูลมีความสามารถในการจัดการธุรกรรมที่เกิดขึ้นพร้อมๆ กัน ซึ่งแม้ว่าระบบจัดการฐานข้อมูลจะจัดการเรื่องดังกล่าวให้โดยอัตโนมัติ การศึกษากลไกและเทคนิคที่ใช้ในการจัดการธุรกรรมที่เกิดขึ้นพร้อมๆ จะทำให้เราเข้าใจและออกแบบฐานข้อมูลที่มีประสิทธิภาพสูงสุด เนื่องความสามารถในการอนุญาตให้ผู้ใช้ๆ ระบบได้พร้อมๆ กันนั้นทำให้ความรวดเร็วใจการทำงานของระบบลดลง พิจารณาตัวอย่างกรณีที่เรามีความจำเป็นต้องจัดการการทำงานพร้อมกันของธุรกรรมทางการเงินดังนี้

ตัวอย่าง

นายสมชายและนางสาวสมศรีเป็นเจ้าของบัญชีเงินฝากร่วมกันซึ่งแต่ละคนสามารถถอนเงินได้ด้วยบัตร ATM หรือทำธุรกรรมที่ธนาคารด้วยตนเอง สมมติว่านายสมชายต้องการถอนเงินจากบัญชี โดยนายสมชายเรียกดูยอดเงินคงเหลือในบัญชีก่อนการถอนเงิน มีเงินในบัญชี 5,000 บาท และต้องการจะถอนเงินทั้งหมด ในขณะเดียวกันนางสาวสมศรีทำการถอนเงินจำนวน 5,000 บาทจากบัญชีเช่นเดียวกันโดยทำธุรกรรมที่เคาท์เตอร์ที่สาขาของธนาคาร แต่เป็นเรื่องบังเอิญที่การถอนเงินนั้นทำในเวลาใกล้เคียงกันมาก โดยระบบถอนเงินที่สาขาของธนาคารทำงานได้เร็วกว่าและตัดยอดเงินในบัญชีไปแล้ว หากเครื่อง ATM ที่นายสมชายกำลังถอนเงินอยู่นั้นอ้างอิงยอดเงินเก่าที่ได้แสดงไปแล้วคือ

5,000 บาทและตัดยอดในบัญชีได้สำเร็จเช่นเดียวกันโดยคิดว่ายังมียอดเงินในบัญชี 5,000 จะทำให้ธนาคารได้รับความเสียหายเนื่องจากลูกค้าได้รับเงินไปถึง 10,000 บาท จากที่มีเงินในบัญชีเพียง 5,000 บาท

ระบบจัดการฐานข้อมูลจะต้องป้องกันการเกิดเหตุการณ์ดังกล่าวให้ได้ทั้งนี้เทคนิคที่นิยมใช้กันในการป้องกันปัญหาการเกิดขึ้นพร้อมกันของธุรกรรมคือเทคนิคการปิดกั้น (lock technique) ซึ่งจะทำการปิดกั้นข้อมูลที่เฉพาะเจาะจงให้ผู้ใช้งานเข้าถึงได้ทีละคน และผลลัพธ์ของการจัดการการเกิดขึ้นพร้อมกันของธุรกรรมคือข้อมูลที่ถูกต้อง ผลลัพธ์หลังจากทุกธุรกรรมเสร็จสิ้นเสมือนกับผลลัพธ์ที่ได้จากการดำเนินการกับธุรกรรมที่ละธุรกรรม สำหรับรายละเอียดในการจัดการการเกิดขึ้นพร้อมกันของธุรกรรมได้อธิบายในบทที่ 9

โดยปกติแล้วการดำเนินการธุรกรรมหนึ่งๆ มีขั้นย่อยๆ ที่จะเกิดขึ้น อาจเป็นไปได้ว่าระหว่างที่ธุรกรรมยังไม่สำเร็จเสร็จสิ้นแต่เกิดความผิดพลาดหรือความล้มเหลวของระบบขึ้น ดังนี้

ตัวอย่าง

นางสาวสมศรีจองตั๋วเครื่องบินโดยโทรไปจองกับบริษัทตัวแทนจำหน่าย บริษัทตัวแทนจำหน่ายทำการจองตั๋วเครื่องบินให้นางสาวสมศรีผ่านทางระบบการจองตั๋วออนไลน์ของตัวแทนจำหน่าย ระบบดังกล่าวส่งรายการการจองไปที่ฐานข้อมูลของสายการบินเพื่อจองที่นั่น ระบบจัดการฐานข้อมูลของสายการบินทำการเพิ่มระเบียนการจองที่นั่ง ลดจำนวนที่ว่างคงเหลือของเที่ยวบินเสร็จแล้วแต่ยังไม่ทันที่ระบบจัดการฐานข้อมูลของสายการบินจะตอบรับผลการจองที่นั่งสำเร็จกลับไปยังระบบจองตั๋วของตัวแทนจำหน่าย ระบบจัดการฐานข้อมูลของสายการบินเกิดการล้มเหลว (crash) ซึ่งอาจเกิดจากมีธุรกรรมเกิดขึ้นพร้อมกันมากเกินไปจนหน่วยความจำหลักไม่เพียงพอ ในกรณีที่ตัวแทนจำหน่ายไม่ได้รับการตอบรับจากสายการบินเราจะถือว่าธุรกรรมไม่เสร็จสมบูรณ์ ระบบจัดการฐานข้อมูลจะต้องทำการคืนสภาพตัวเอง (recovery) ให้ถูกต้องหลังจากที่แก้ไขความล้มเหลวของระบบได้แล้ว ในกรณีนี้ฐานข้อมูลต้องลบระเบียนการจองเที่ยวบินของนางสาวสมศรีและเพิ่มที่นั่งที่เหลือในเที่ยวบินนั้น

การคืนสภาพของตัวเองนั้นระบบจัดการฐานข้อมูลอาศัยเทคนิคที่สำคัญที่เรียกว่าการบันทึกกิจกรรมของระบบ (log) ไว้ เพื่อให้ทราบว่าระบบจัดการฐานข้อมูลกำลังทำกิจกรรมใดๆ กับฐานข้อมูลและอยู่ในขั้นตอนใดของธุรกรรมในกรณีที่ระบบล้มเหลว ระบบจัดการฐานข้อมูลจะทำการอ่าน log เพื่อตัดสินใจดำเนินการกับข้อมูลต่อไป ทั้งนี้การจัดการการคืนสภาพของข้อมูลยังเกี่ยวข้องกับการจัดการความมั่นคงปลอดภัยของข้อมูล ซึ่งได้อธิบายเพิ่มเติมในบทที่ 9

1.9 โครงสร้างของระบบจัดการฐานข้อมูล

การทำงานของระบบจัดการฐานข้อมูลโดยรวมมีขั้นตอนคร่าวๆ ดังนี้ ระบบจัดการฐานข้อมูลรับข้อคำถามในที่นี้คือ SQL จากนั้นระบบจัดการฐานข้อมูลจะวิเคราะห์แผนในการเข้าถึงข้อมูลที่จะนำมาประมวลผลเพื่อแสดงเป็น

ผลลัพธ์ ทำการดำเนินการประมวลผลตามแผนและแสดงผลลัพธ์ รูปที่ แสดงโครงสร้างที่สำคัญของระบบจัดการฐานข้อมูลเชิงสัมพันธ์ สามารถอธิบายขั้นตอนการทำงานและหน้าที่ของแต่ละองค์ประกอบได้ดังนี้

เราแบ่งผู้ใช้ออกเป็น 2 กลุ่มใหญ่ๆ คือกลุ่มผู้ใช้ทั่วไปที่ไม่จำเป็นต้องมีความรู้ด้านระบบจัดการฐานข้อมูล (เช่น ลูกค้าทั่วไป ตัวแทนจำหน่ายตั๋วเครื่องบิน นักศึกษาผู้ใช้งานระบบลงทะเบียน ฯลฯ) และกลุ่มผู้ใช้ที่มีความรู้และทักษะเชิงเทคนิคในการจัดการฐานข้อมูล (เช่น โปรแกรมเมอร์ ผู้บริหารฐานข้อมูล ฯลฯ) ผู้ใช้ในกลุ่มแรกจะใช้งานฐานข้อมูลผ่านส่วนติดต่อกับผู้ใช้ที่เป็นโปรแกรมประยุกต์และระบบสารสนเทศต่างๆ เช่นเว็บฟอร์ม หน้าจอของโปรแกรมฯลฯ ส่วนติดต่อกับผู้ใช้นั้นจะใช้งานได้ง่ายและอำนวยความสะดวกในการใช้ข้อมูลของผู้ใช้เช่นระบบลงทะเบียนจะมีส่วนติดต่อกับผู้ใช้แบบกราฟฟิคเพื่ออำนวยความสะดวกให้นักศึกษาค้นหารายวิชาและลงทะเบียนเรียนในรายวิชาใดๆ โปรแกรมประยุกต์จะนำข้อมูลที่ได้รับจากผู้ใช้ผ่านทางส่วนติดต่อกับผู้ใช้ไปสร้างคำสั่ง SQL สำหรับส่งเข้าไปประมวลผลในระบบจัดการฐานข้อมูลต่อไป สำหรับผู้ใช้งานระบบจัดการฐานขอมูลที่มีความเชี่ยวชาญเฉพาะด้านฐานข้อมูลอันได้แก่โปรแกรมเมอร์ หรือผู้บริหารฐานข้อมูลสามารถเรียกใช้งานข้อมูลโดยสร้างคำสั่ง SQL ด้วนตนเอง จะกรอกคำสั่ง SQL ผ่านหน้าจอรับคำสั่ง SQL ซึ่งหน้าจอรับคำสั่งอาจมีความสามารถในการวิเคราะห์ประสิทธิภาพของรูปแบบของ SQL ต่างๆ พร้อมทั้งเสนอแนะให้ผู้บริหารฐานข้อมูลเลือกใช้ข้อคำสั่งที่เหมาะสม หรือแม้แต่การเปรียบเทียบประสิทธิภาพหากสร้างไฟล์ index เพิ่มเติม โปรแกรมเมอร์จะใช้หน้าจอรับคำสั่ง SQL เพื่อทดสอบ SQL ที่จะต้องสร้างขึ้นเพื่อในไปฝังในโปรแกรมประยุกต์ให้ผู้ใช้ทั่วไปได้ใช้งาน

คำสั่ง SQL ที่ได้จากส่วนติดต่อกับผู้ใช้แบบต่างๆ จะถูกส่งเข้าไปประมวลผลยังระบบจัดการฐานข้อมูลโดยส่วนประมวลผลข้อคำถาม (query processor) เป็นส่วนประมวลข้อคำถามโดยการแปลง SQL ให้เป็นอยู่ในรูปแบบของแผนการเข้าถึงข้อมูลโดยใช้องค์ประกอบย่อยที่เรียกว่าตัว parser ** แผนการเข้าถึงข้อมูลมักจะอยู่ในรูปของต้นไม้ตัวดำเนินการเชิงสัมพันธ์ซึ่งจะได้กล่าวถึงในเรื่องของแคลคูลัสเชิงสัมพันธ์และพีชคณิตเชิงสัมพันธ์ ** จากนั้นแผนของการเข้าถึงข้อมูลจะถูกสร้างขึ้นหลายๆ รูปแบบและถูกเปรียบเทียบเพื่อเลือกรูปแบบที่เหมาะสมกับข้อมูลที่จัดเก็บอยู่ในหน่วยความจำสำรองด้วย optimizer และ query evaluator แผนการเข้าถึงและดึงข้อมูลขึ้นมาใช้งานจะถูกดำเนินการต่อไปด้วย plan executor

หลังจากที่เราได้แผนการแสดงข้อมูลที่อยู่รูปแบบของตัวดำเนินการทางแบบจำลองข้อมูล ต่อไปจะเป็นการเข้าถึงข้อมูลในอุปกรณ์จัดเก็บ ซึ่งจะอยู่ในรูปของไฟล์ที่มีโครงสร้างเฉพาะของแต่ละระบบจัดการฐานข้อมูล ทั้งนี้โดยทั่วไปจะจัดเก็บในรูปของไฟล์ข้อมูลที่เป็นกลุ่มของระเบียนหลายๆ ไฟล์ ซึ่งระเบียนมักไม่ได้เรียงลำดับของมูลอยู่เพราะระเบียนมีจำนวนมาก การเพิ่มและลบระเบียนเกิดขึ้นอยู่เสมอ ระเบียนต่างๆ จึงไม่ได้เรียงลำดับ แต่เรายังมีไฟล์เสริมในการเข้าถึงข้อมูลได้รวดเร็วเช่นไฟล์ index ที่เรียงลำดับเฉพาะฟิลด์ข้อมูลใดๆ เพื่อระบุที่อยู่ของระเบียบได้อย่างรวดเร็ว การเข้าถึงข้อมูลอย่างรวดเร็วโดยกลวิธีทางไฟล์ต่างๆ เป็นหน้าที่ของ file and access methods

ข้อมูลจะต้องถูกอ่านจากหน่วยความจำสำรองเข้าสู่หน่วยความจำหลักเพื่อประมวลผลและสร้างผลลัพธ์ buffer manager เป็นผู้จัดการเกี่ยวกับการนำข้อมูลจากหน่วยความจำสำรองเข้าสู่หน่วยความจำหลักและ disk space manager เป็นส่วนที่ใช้ในการควบคุมการจัดเก็บและอ่านข้อมูลในระดับล่างสุดหรือระดับฮาร์ดแวร์อันได้แก่การจองเนื้อที่ใน

อุปกรณ์จัดเก็บ การเขียนข้อมูลลงอุปกรณ์จัดเก็บตามที่ได้จองพื้นที่และจัดสรรตำแหน่งไว้ การอ่านข้อมูลจากอุปกรณ์จัดเก็บในระดับฮาร์ดแวร์และการคืนพื้นที่ของอุปกรณ์จัดเก็บให้ว่างลง

เมื่อข้อมูลในหน่วยความจำหลักถูกประมวลผลตาม plan executor แล้วผลลัพธ์ที่ได้จะถูกส่งกลับไปยังส่วนติดต่อกับผู้ใช้เพื่อนำไปใช้งานในที่สุด นอกเหนือกจากกระบวนการโดยทั่วไปของการใช้งานระบบจัดการฐานข้อมูลที่ได้อธิบายมา ระบบจัดการฐานข้อมูลยังมีส่วนของ transaction manager ที่อนุญาตให้ผู้ใช้งานใช้ฐานข้อมูลพร้อมกับได้หลายๆ คน การป้องกันความผิดพลาดที่จะเกิดจากการดำเนินธุรกรรมพร้อมๆ กันด้วย lock manager และการคืนสภาพของข้อมูลในกรณีที่ระบบเกิดการล้มเหลวโดย recovery manager

1.10 บุคคลที่เกี่ยวข้องกับฐานข้อมูล

บุคคลซึ่งเกี่ยวข้องกับการพัฒนาและใช้งานฐานข้อมูลได้แก่ผู้ใช้ (end user) ผู้สร้างระบบจัดการฐานข้อมูล (database implementer/vendor) ผู้บริหารฐานข้อมูล (database administrator—DBA) นักวิเคราะห์และออกแบบระบบ (system analyst) โปรแกรมเมอร์ (programmer)

1) **ผู้สร้างระบบจัดการฐานข้อมูล**

ผู้สร้างระบบจัดการฐานข้อมูลได้แก่บริษัทผู้สร้างระบบจัดการฐานข้อมูล เช่น ไอบีเอ็ม ไมโครซอฟต์ ออราเคิล ซึ่งเป็นผู้สร้างระบบจัดการฐานข้อมูล IBM DB/2, Microsoft SQL Server/Microsoft Access และ Oracle ตามลำดับ ผู้สร้างระบบจัดการฐานข้อมูลนั้นมีความสำคัญในการประยุกต์ทฤษฎีและเทคโนโลยีต่างๆ ที่เกี่ยวข้องและมีวิวัฒนาการอยู่ตลอดเวลา ในการจัดการฐานข้อมูลเพื่อสร้างระบบจัดการฐานข้อมูลที่เหมาะสมกับลักษณะการใช้งานปัจจุบัน อันเป็นหัวใจสำคัญของการจัดการข้อมูลอย่างมีประสิทธิภาพ

2) **ผู้ใช้**

ผู้ที่จะใช้งานฐานข้อมูล ซึ่งเป็นผู้บันทึกและเรียกใช้ข้อมูลจากฐานข้อมูล ทั้งนี้ผู้ใช้งานจะใช้งานผ่านระบบสารสนเทศหรือโปรแกรมประยุกต์ที่ออกแบบและพัฒนาไว้สำหรับกิจกรรมใดๆ ซึ่งผู้ใช้งานฐานข้อมูลไม่จำเป็นต้องมีความรู้เรื่องระบบจัดการฐานข้อมูล ทั้งนี้ระบบสารสนเทศดังกล่าวใช้ระบบจัดการฐานข้อมูลสำหรับจัดการข้อมูล ตัวอย่างของผู้ใช้ เช่นผู้ใช้งานระบบลงทะเบียนของมหาวิทาลัยได้แก่ นักศึกษาค้นหารายวิชาเพื่อทำการลงทะเบียนและแสดงเกรดของตนเอง อาจารย์เรียกดูข้อมูลตารางสอนและรายละเอียดการลงทะเบียนของวิชาที่ตนเองสอน เป็นต้น

3) **นักวิเคราะห์ระบบ**

เป็นผู้ทำหน้าที่สำรวจ รวบรวม และวิเคราะห์ความต้องการการใช้งานของผู้ใช้ เพื่อนำมาออกแบบมิใช่เพียงฐานข้อมูลเท่านั้น แต่เป็นการออกแบบองค์ประกอบของระบบสารสนเทศหรือโปรแกรมประยุกต์ที่จะอำนวยความสะดวกในการทำงานและเรียกใช้งานฐานข้อมูล ในส่วนของการออกแบบฐานข้อมูล นักวิเคราะห์ระบบจะทำการวิเคราะห์ความต้องการและออกแบบฐานข้อมูลในรูปแบบจำลองข้อมูลเชิงความหมาย ได้แก่แผนภาพ ER สคีมาของฐานข้อมูลระดับภายนอก โดยทั่วไปนักวิเคราะห์ระบบยังเป็นผู้ออกแบบสคีมาระดับแนวคิดและระดับกายภาพอีกด้วย นอกจากนี้แล้วนักวิเคราะห์ระบบยังออกแบบขั้นตอนการทำงานทั้งหมดของโปรแกรมอีกด้วย

4) โปรแกรมเมอร์

โปรแกรมเมอร์คือผู้ที่นำระบบที่ได้จากการวิเคราะห์และออกแบบมาเขียนโปรแกรมเพื่อพัฒนาเป็นระบบสานสนเทศหรือโปรแกรมประยุกต์โดยเครื่องมือที่ใช้ในการพัฒนาโปรแกรมต่างๆ ให้ออกมาเป็นระบบที่สามารถใช้งานได้

5) ผู้บริหารฐานข้อมูล

ผู้บริหารฐานข้อมูลเรามักเรียกว่า DBA เป็นผู้มีบทบาทสำคัญอย่างยิ่งในมุมมองของการออกแบบและพัฒนาฐานข้อมูล ทั้งนี้การศึกษารายวิชาการออกแบบและพัฒนาฐานข้อมูลนี้สามารถให้ผู้เรียนมีความสามารถระดับ DBA ซึ่งเป็นผู้เชี่ยวชาญด้านการจัดการฐานข้อมูล DBA มีหน้าที่ดังต่อไปนี้

- ออกแบบสคีมาระดับแนวคิดและระดับกายภาพ หน้าที่ของ DBA ในข้อนี้คล้ายคลึงกับนักวิเคราะห์ระบบ แต่มุ่งเน้นที่การวิเคราะห์และออกแบบฐานข้อมูล และเน้นที่การออกแบบสคีมาในระดับแนวคิดลงมา
- กำหนดสิทธิการเข้าใช้ข้อมูลของผู้ใช้
  กำหนดสิทธิการเข้าใช้ข้อมูลให้ผู้ใช้แต่ละกลุ่ม แต่ละคน ตลอดจนชุดข้อมูลที่เข้าถึงได้และความสามารถที่ผู้ใช้แต่ละคนสามารถกระทำได้กับข้อมูล ตลอดจนตรวจสอบความมั่นคงปลอดภัยและการบุกรุกการใช้งานฐานข้อมูล
- ดูแลให้การใช้งานฐานข้อมูลเป็นไปโดยปกติและแก้ไขในกรณีที่ระบบล้มเหลว
  ทำการตรวจสอบความเป็นปกติของการใช้งานฐานข้อมูลอย่างสม่ำเสมอ สำรองข้อมูล และสามารถทำให้ระบบคืนสู่สภาพปกติได้โดยข้อมูลมีความถูกต้องในกรณีระบบเกิดการติดขัดหยุดชะงัก ล้มเหลว
- ปรับแต่งในการใช้งานฐานข้อมูลถูกต้อง และรวดเร็วขึ้น
  เช่นการออกแบบการสร้าง index สำหรับตารางใดๆ DBA จะตรวจสอบการใช้งานระบบฐานข้อมูลและวิเคราะห์ว่าข้อมูลกลุ่มหรือตารางใดที่มีการใช้งานมาก หรือน้อย ควรเพิ่มหรือลด

index สำหรับตารางใด หรือการแยกข้อมูลชุดเก่าและไม่ได้ใช้ออกจากข้อมูลที่ใช้ในปัจจุบันเพื่อ เป็นต้น DBA ยังทำงานร่วมกับนักวิเคราะห์ระบบและโปรแกรมเมอร์เพื่อเสนอแนะ และปรับปรุงระบบให้สอดคล้องกับลักษณะการใช้งานฐานข้อมูลเมื่อใช้งานจริง

1.11 แบบฝึกหัดท้ายบท

ให้นักศึกษาใช้วัตถุประสงค์การเรียนของบทนี้เป็นคำถามท้ายบท

บทที่ 2 ขั้นตอนการออกแบบฐานข้อมูล

วัตถุประสงค์

- สามารถระบุขั้นตอนหลักของวงจรการออกแบบระบบฐานข้อมูล (database system development lifecycle—DSDLC) และอธิบายสิ่งที่ต้องปฏิบัติในแต่ละขั้นตอนได้
- สามารถอธิบายวิธีการออกแบบฐานข้อมูลในระดับแนวคิด ระดับตรรกะและระดับภายภาพ
- สามารถอธิบายถึงประโยชน์ของ CASE tools ได้

คำสำคัญ: วงจรระบบสารสนเทศ (information system lifecycle—ISDLC); วงจรการพัฒนาระบบ (software development lifecycle—SDLC); การวางแผนฐานข้อมูล (database planning); การนิยามระบบ (system definition); วิวของผู้ใช้ (user views); (การรวบรวมและวิเคราะห์ความต้องการ) requirements collection and analysis; (วิธีการแบบรวมศูนย์) centralized approach; (วิธีการแบบบูรณาการมุมมอง) view integration; การออกแบบฐานข้อมูลจากล่างขึ้นบน (bottom-up database design); การออกแบบฐานข้อมูลจากบนลงล่าง (top-down database design); การออกแบบฐานข้อมูลระดับแนวคิด (conceptual database design); การออกแบบฐานข้อมูลระดับตรรกะ (logical database design); การออกแบบฐานข้อมูลระดับการภาย (physical database design); การเลือกระบบจัดการฐานข้อมูล (DBMS selection); การออกแบบโปรแกรมประยุกต์ (application design); การพัฒนาต้นแบบ (prototyping); การพัฒนาระบบ (implementation); การแปลงและโหลดข้อมูล (data conversion and loading); การทดสอบ (testing); การบำรุงรักษา (operational maintenance); เครื่องมือทางคอมพิวเตอร์ที่ช่วยในงานวิศวกรรมซอฟต์แวร์ (CASE tools)

2.1 บทนำ

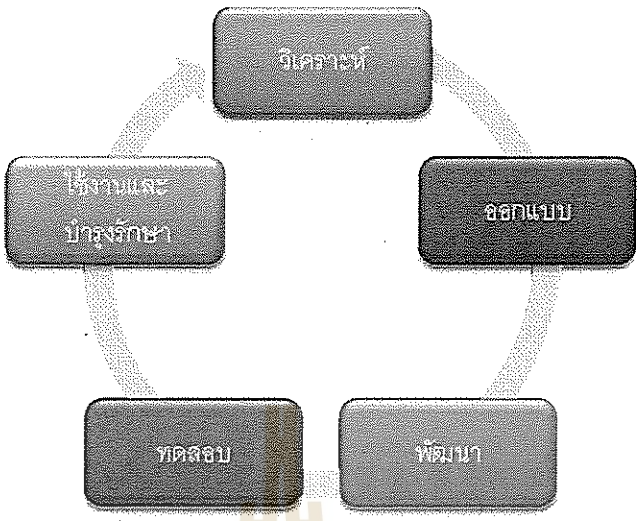
ระบบฐานข้อมูลนั้นมีความจำเป็นอย่างยิ่งยวดในปัจจุบันดังที่ได้กล่าวถึงในบทที่ 1 ซึ่งระบบฐานข้อมูลนั้นจะถูกนำไปใช้ประโยชน์อย่างสมบูรณ์ในรูปของการเป็นเป็นองค์ประกอบพื้นฐานอย่างหนึ่งของระบบสารสนเทศ ซึ่งระบบสารสนเทศคือการนำเทคโนโลยีคอมพิวเตอร์และการสื่อสารมาอำนวยความสะดวกให้การดำเนินการในองค์กรบรรลุวัตถุประสงค์อย่างมีประสิทธิภาพ ซึ่งการนำระบบสารสนเทศมาใช้จะต้องได้รับการออกแบบให้เหมาะสมกับกิจกรรมที่จะนำมาอำนวยความสะดวก

การได้มาซึ่งระบบสารสนเทศที่เหมาะสมกับองค์กร เราจำเป็นต้องอาศัยระเบียบวิธีที่รัดกุมในการวิเคราะห์ออกแบบ และพัฒนาระบบอย่างมีประสิทธิภาพ มิเช่นนั้นแล้วระบบอาจจะไม่สามารถรองรับความต้องการการใช้งานของผู้ใช้ ไม่คุ้มค่าเป็นอย่างมาก รวมถึงอาจเกิดการล้มเหลวของการพัฒนาและใช้งานระบบสารสนเทศซึ่งเป็นสิ่งที่ไม่ควรจะเกิดขึ้น แต่เป็นเรื่องที่สามารถเกิดขึ้นได้และบ่อยครั้งทีเดียว ดังข้อมูลของการสำรวจและวิจัยโดย OASIG (OASIG, 1996) ซึ่งเป็นหน่วยงานที่ศึกษาเกี่ยวกับการใช้เทคโนโลยีสารสนเทศในองค์กร พบว่า 80-90% ของระบบสารสนเทศที่พัฒนาขึ้นในการสำรวจมีประสิทธิภาพต่ำกว่าเป้า 80% นั้นมีความล่าช้าในการพัฒนาและงบประมาณบานปลาย ระบบมากถึง 40% ล้มเหลวและไม่ได้ใช้งาน ซึ่งมีเพียง 10% เท่านั้นที่ประสบความสำเร็จ จึงมีความจำเป็นอย่างยิ่งยวดที่จะมีการศึกษาถึงกระบวนการที่จะได้มาซึ่งระบบสารสนเทศ

ในบทนี้จะได้กล่าวถึงขั้นตอนและกระบวนการในการพัฒนาฐานข้อมูล และวงจรการใช้งานระบบฐานข้อมูลซึ่งมีความสัมพันธ์กับวงจรการพัฒนาระบบสารสนเทศอย่างแยกจากกันได้ยาก วิศวกรรมซอฟต์แวร์เป็นศาสตร์ที่รองรับการได้มาซึ่งระบบสารสนเทศที่มีประสิทธิภาพ

2.2 วงจรการพัฒนาระบบ (System Development Lifecycle—SDLC)

วิศวกรรมซอฟต์แวร์เป็นศาสตร์ที่เกี่ยวกับการพัฒนาซอฟต์แวร์ คือการใช้กระบวนการทางวิศวกรรมในการควบคุมการผลิต วิเคราะห์ความต้องการ การออกแบบ พัฒนา การทดสอบ การประเมินผล ฯลฯ ในการใช้งานระบบสารสนเทศ ซึ่งกลวิธีในการออกแบบและพัฒนาซอฟต์แวร์นั้นมีหลายวิธี ซึ่งโดยรวมแล้ววงจรการพัฒนาซอฟต์แวร์ (software development lifecycle—SDLC) หรือการพัฒนาระบบสารสนเทศจะประกอบไปด้วยขั้นตอนหลัก 5 ขั้นตอนได้แก่ การวิเคราะห์ การออกแบบ การพัฒนา การทดสอบ และการใช้งาน/การบำรุงรักษา **



การออกแบบระบบฐานข้อมูลเป็นส่วนหนึ่งของการออกแบบระบบ ซึ่งทั้งระบบฐานข้อมูลและระบบสารสนเทศต่างก็มีความสำคัญแก่กันและกัน ระบบสารสนเทศจะต้องอาศัยระบบจัดการฐานข้อมูลเพื่อรองรับการจัดการข้อมูลอย่างมีประสิทธิภาพ และในทางกลับกัน ระบบฐานข้อมูลจะไม่ได้ถูกนำไปใช้ประโยชน์อย่างเหมาะสมหากไม่มีโปรแกรมประยุกต์มาเรียกใช้ประโยชน์จากฐานข้อมูล โดยทั่วไปการออกแบบระบบสารสนเทศและการออกแบบฐานข้อมูลจะทำไปพร้อมๆ กัน ในหัวข้อต่อไปจะถึงกล่าวขั้นตอนของการออกแบบฐานข้อมูลที่มีประสิทธิภาพ ซึ่งมีขั้นตอนที่คล้ายคลึงกับวงจรของการพัฒนาซอฟต์แวร์ ที่มีวิธีการและข้อควรคำนึงถึงเป็นพิเศษรวมทั้งมีขั้นตอนเฉพาะที่สำคัญในส่วนของการออกแบบระบบฐานข้อมูล ได้แก่การออกแบบฐานข้อมูระดับแนวคิด ระดับตรรกะ และระดับกายภาพ

2.3 วงจรการพัฒนาฐานข้อมูล (Database System Development Lifecycle—DSDLC)

ขั้นตอนการพัฒนาฐานข้อมูลที่สำคัญมี 13 ขั้นตอน ได้แก่ 1) การวางแผนฐานข้อมูล 2) การนิยามระบบ 3) การรวบรวมและวิเคราะห์ความต้องการ 4) การออกแบบฐานข้อมูลระดับแนวคิด 5) การออกแบบฐานข้อมูลระดับตรรกะ 6) การออกแบบฐานข้อมูลระดับกายภาพ 7) การเลือกระบบจัดการฐานข้อมูล 8) การออกแบบระบบ 9) การพัฒนาระบบ 10) การพัฒนาต้นแบบ 11) การแปลงและบรรจุข้อมูล 12) การทดสอบ 13) การใช้งานและบำรุงรักษา โดยวงจรการพัฒนาฐานข้อมูลนั้นไม่จำเป็นต้องทำทุกขั้นตอนและบางขั้นตอนยังสามารถทำไปพร้อมๆ กันได้

ขั้นตอนการออกแบบฐานข้อมูลควรเริ่มจากการวางแผนฐานข้อมูล ซึ่งเกี่ยวข้องกับการวางแผนโครงการพัฒนาฐานข้อมูล จากนั้นเป็นการนิยามระบบ ซึ่งเกี่ยวข้องกับการระบุผู้ที่จะใช้งานฐานข้อมูลและวัตถุประสงค์ของการพัฒนาระบบฐานข้อมูล หลังจากนั้นการรวบรวมและวิเคราะห์ความต้องการเป็นการปฏิบัติในรายละเอียดเพื่อที่จะระบุข้อมูลทั้งหมดที่จะต้องจัดเก็บรวมถึงความสามารถของระบบสารสนเทศที่จะใช้งานระบบฐานข้อมูล หลังจากนั้นแล้วเรานำเอกสารที่ได้จากการวิเคราะห์ระบบมาออกแบบฐานข้อมูล การออกแบบฐานข้อมูลนั้นเราสามารถกระทำไป

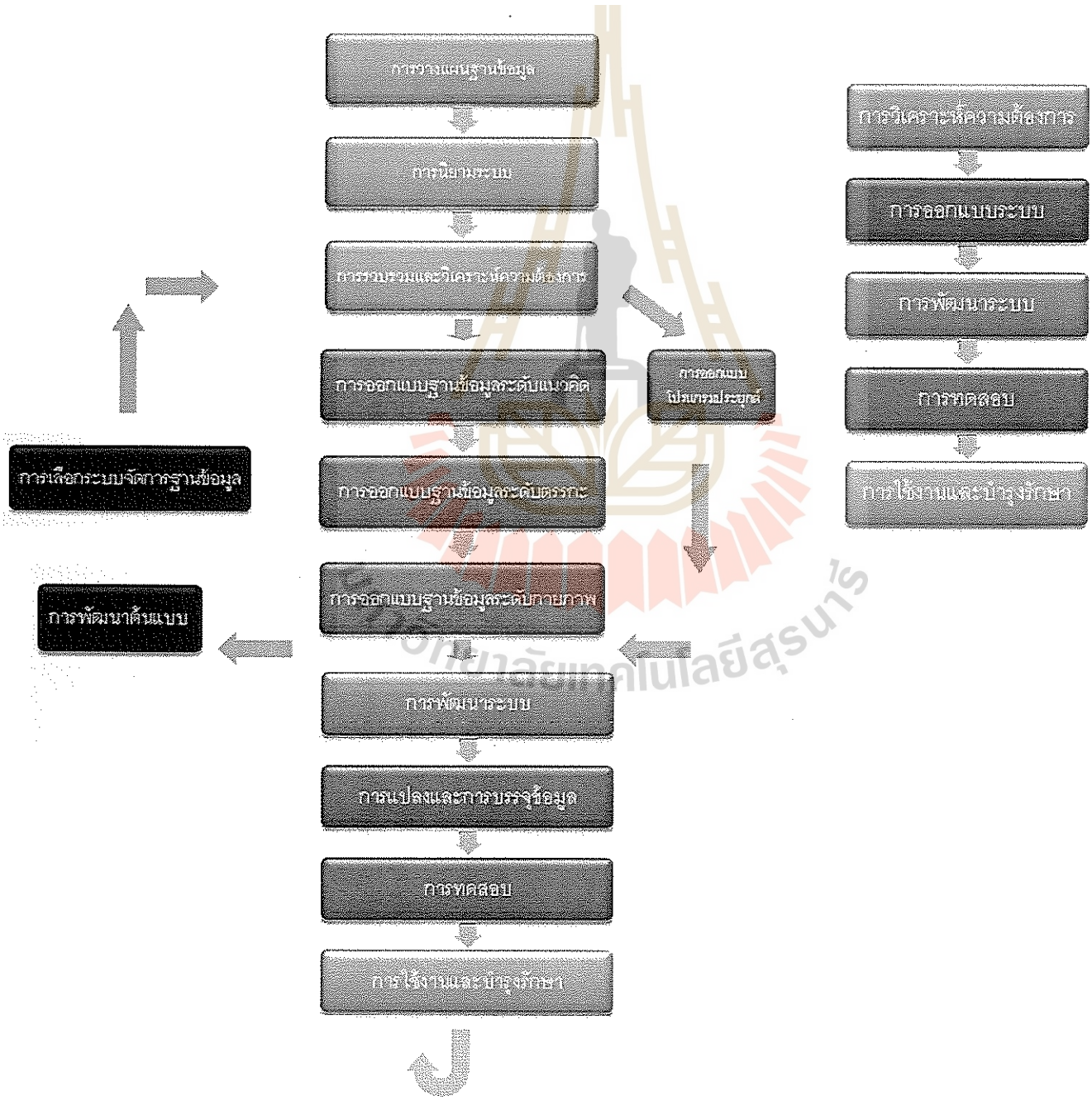
พร้อมๆ กับการออกแบบโปรแกรมประยุกต์ได้ ในที่นี้คือโปรแกรมหรือระบบสารสนเทศที่จะเรียกใช้งานฐานข้อมูล การออกแบบฐานข้อมูลมีระเบียบวิธีการออกแบบที่ชัดเจนโดยเริ่มจากการออกแบบระดับแนวคิด เป็นการระบุเอนทิตี และความสัมพันธ์ของเอนทิตี ตลอดจนข้อกำหนดหรือเงื่อนไขใดๆ ของข้อมูล ที่ไม่ยึดติดกับระบบจัดการฐานข้อมูล เรานำผลลัพธ์ที่ได้จากการออกแบบฐานข้อมูลระดับแนวคิดมาวิเคราห์เพื่อออกแบบฐานข้อมูลระดับตรรกะ จากนั้นเป็นการออกแบบฐานข้อมูลระดับกายภาพ หลังจากเสร็จสิ้นกระบวนการออกแบบทั้งฐานข้อมูลและโปรแกรมประยุกต์แล้วเราอาจพัฒนาระบบต้นแบบขึ้นเพื่อตรวจสอบความถูกต้องของการออกแบบ ทั้งนี้อาจมีการพิจารณาปรับแก้การออกแบบให้ถูกต้องหรือเพิ่มสิ่งที่ตกไปจากการออกแบบ จากนั้นทำการพัฒนาระบบตามที่ได้ออกแบบไว้ เมื่อฐานข้อมูลได้รับการพัฒนาแล้วเสร็จ เราจะต้องทดสอบโปรแกรมและระบบฐานข้อมูลที่ได้พัฒนาขึ้นก่อนการนำไปใช้จริง ในขั้นตอนสุดท้ายหลังจากการทดสอบ เรานำระบบฐานข้อมูลมาใช้ ระบบต้องการการดูแลบำรุงรักษาเพื่อให้สามารถใช้งานได้อย่างไม่ติดขัดตลอดจนการปรับแต่งหรือเพิ่มความสามารถใหม่ๆ ในระบบ ซึ่งหลักจากการใช้งานระบบฐานข้อมูลไปได้ระยะเวลาหนึ่ง เราอาจจะต้องมีการปรับแต่ง แก้ไข หรือเพิ่มความสามารถเพื่อรองรับความต้องการข้อผู้ใช้ที่เพิ่มขึ้น การเพิ่มความสามารถของระบบฐานข้อมูลนั้นเริ่มต้นจากการวางแผนการพัฒนาระบบฐานข้อมูล นั่นเป็นการกลับสู่ขั้นตอนเริ่มต้นของวงจรการพัฒนาระบบฐานข้อมูลนั่นเอง

แผนภาพที่ 1** แสดงภาพรวมของขั้นตอนในการพัฒนาระบบฐานข้อมูล และมีการเปรียบเทียบกับวงจรพัฒนาซอฟต์แวร์ในขั้นตอนที่มีการดำเนินงานในลักษณะเดียวกัน ตารางที่ 1** สรุปขั้นตอนและกิจกรรมหลักที่ปฏิบัติสำหรับแต่ละขั้นตอนของการพัฒนาระบบฐานข้อมูล พร้อมทั้งตัวอย่างของผลลัพธ์หรือเอกสารที่ได้จากการปฏิบัติในแต่ละขั้นตอน ในหัวข้อต่อๆ ไปเป็นการอธิบายถึงวิธีการปฏิบัติของขั้นตอนการพัฒนาระบบฐานข้อมูลแต่ละขั้นตอน ตัวอย่างของการปฏิบัติในแต่ละขั้นตอนตั้งแต่ขั้นตอนแรกจนถึงขั้นตอนสุดท้าย ได้ถูกแสดงเป็นกรณีศึกษาในการพัฒนาระบบฐานข้อมูลร้าน 7-Elephant ในภาคผนวก

ขั้นตอน	กิจกรรมหลัก
การวางแผนฐานข้อมูล	การวางแผนโครงการการพัฒนาระบบฐานข้อมูล การกำหนดขั้นตอน และวางแนวทางเพื่ออำนวยความสะดวกให้การพัมนาฐานข้อมูลเกิดประสิทธิภาพสูงสุด
การนิยามระบบ	กำหนดขอบเขตของการใช้งานระบบฐานข้อมูล
การรวบรวมและวิเคราะห์ความต้องการ	รวบรวบ และวิเคราะห์ความต้องการของผู้ใช้ เพื่อให้ระบบฐานข้อมูลรองรับ
การออกแบบฐานข้อมูล	การออกแบบฐานข้อมูลระดับแนวคิด ระดับตรรกะ และระดับกายภาพ
การเลือกระบบจัดการฐานข้อมูล	เลือกใช้งานระบบจัดการฐานข้อมูลที่เหมาะสม
การออกแบบโปรแกรมประยุกต์	ออกแบบส่วนติดต่อกับผู้ใช้และโปรแกรมประยุกต์ที่เรียกใช้งานฐานข้อมูล
การพัฒนาต้นแบบ	สร้างต้นแบบที่สามารถใช้งานได้
การพัฒนาระบบ	ลงมือสร้างฐานข้อมูลและพัฒนาโปรแกรมประยุกต์เพื่อใช้งานจริง

การแปลงและการบรรจุข้อมูล	การนำข้อมูลบรรจุลงฐานข้อมูลใหม่ที่ได้พัฒนาขึ้นรวมถึงการปรับเปลี่ยนโปรแกรมประยุกต์ที่อาจใช้อยู่เพื่อให้ทำงานได้กับระบบฐานข้อมูลใหม่
การทดสอบ	ทดสอบระบบเพื่อหาข้อผิดพลาดและยืนยันว่าระบบรองรับความต้องการทั้งหมดของผู้ใช้
การใช้งานและบำรุงรักษา	ดูแลและบำรุงรักษาระบบฐานข้อมูลในขณะใช้งานจริง

ขั้นตอนแต่ละขั้นตอนจะสำเร็จได้โดยการสร้างเอกสารที่เป็นผลลัพธ์สำหรับขั้นตอนนั้นๆ 1**



2.4 การวางแผนฐานข้อมูล

การวางแผนฐานข้อมูลคือการวางแผนเกี่ยวกับฐานข้อมูลเชิงการจัดการในองค์กร เป็นการกำหนดวัตถุประสงค์ของการใช้งานฐานข้อมูลในองค์กรให้สอดคล้องกับเป้าหมายการดำเนินการ ตลอดจนให้องค์กรได้ตระหนักถึงความสำคัญของระบบฐานข้อมูลเพื่อให้การพัฒนาและใช้งานมีประสิทธิภาพสูงสุด

การวางแผนฐานข้อมูลเป็นขั้นตอนเชิงการบริหารจัดการขององค์กร โดยเป็นการวางแผนการปฏิบัติงานสำหรับโครงการการพัฒนาระบบฐานข้อมูลเพื่อใช้ในองค์กร เปรียบได้กับกรณีเช่น หากบริษัทมีโครงการที่จะจัดกิจกรรมส่งเสริมการขายสินค้า บริษัทต้องวางแผนว่าจะต้องจัดกิจกรรมส่งเสริมสินค้าตัวใด ที่ใด เมื่อใด อย่างไร เพราะเหตุใดเป็นต้น การวางแผนฐานข้อมูลนั้นต้องสอดคล้องกับการบูรณาการเพื่อใช้กับระบบสารสนเทศขององค์กร การวางแผนฐานข้อมูลยังครอบคลุมถึงการติดตามและเร่งรัดการปฏิบัติงานในแต่ละขั้นตอน ซึ่งการวางแผนฐานข้อมูลให้สอดคล้องกับการใช้งานในองค์กรนั้นอาจมีแรงผลักดังหรือสามารถพิจารณาได้จาก

- การพิจารณาภาพรวมขององค์การ เช่น พัฒนาระบบฐานข้อมูลเพื่อนำมาใช้รองรับการทำงานให้บรรลุเป้าหมายขององค์กร
- การประเมินระบบสารสนเทศที่ใช้อยู่ในปัจจุบัน ข้อเด่น ข้อด้อยของระบบ ซึ่งนำมาสู่การปรับปรุงหรือสร้างระบบใหม่
- พิจารณาความได้เปรียบในการแข่งขัน เราอาจนำระบบสารสนเทศมาใช้งานเพื่อมีความสามารถเหนือคู่แข่ง

หลังจากพิจารณาถึงความจำเป็นในนำระบบฐานข้อมูลมาใช้ องค์กรต้องกำหนดถ้อยแถลงภาระกิจ (mission statement) ที่ถูกต้องสำหรับระบบฐานข้อมูล เพื่อกำหนดทิศทางที่แน่ชัดในการพัฒนาและใช้งานระบบฐานข้อมูล การกำหนดภาระกิจที่ชัดเจนและถูกต้องดังกล่าวเป็นการทำความเข้าใจของผู้มีส่วนร่วมในระบบฐานข้อมูลให้ตรงกัน และหลีกเลี่ยงการพัฒนาระบบที่สร้างขึ้นมาแล้วไม่ได้ผลักดันให้องค์กรบรรลุวัตถุประสงค์การดำเนินงาน โดยส่วนใหญ่ผู้บริหารระดับสูงหรือผู้ที่มีความเชี่ยวชาญด้านระบบสารสนเทศขององค์กรจะเป็นผู้กำหนดภารกิจและนโยบายเกี่ยวกับระบบฐานข้อมูลในองค์กร จากนั้นวัตถุประสงค์ของการสร้างและใช้งานระบบฐานข้อมูลต้องถูกกำหนดขึ้นให้สอดคล้องกับภาระกิจของระบบฐานข้อมูล หลังจากที่ระบบฐานข้อมูลพัฒนาแล้วเสร็จและได้ถูกใช้งาน การประเมินผลการใช้งานในภายหลังสามารถพิสูจน์ได้ว่าระบบฐานข้อมูลที่ได้พัฒนาขึ้นนั้นปฏิบัติภาระกิจได้สมบูรณ์ก็ต่อเมื่อการใช้งานบรรลุวัตถุประสงค์แต่ละข้อที่ได้กำหนดไว้ ถ้อยแถลงภาระกิจยังสามารถใช้ในการอธิบายภาพรวมวัตถุประสงค์ของระบบฐานข้อมูลในกรณีที่ชื่อของระบบฐานข้อมูลไม่ได้ระบุลักษณะการใช้งานโดยตรง

ตัวอย่างถ้อยแถลงภาระกิจของระบบฐานข้อมูลทะเบียนนักศึกษาของมหาวิทยาลัยๆ หนึ่ง ได้แก่

"ระบบฐานข้อมูลทะเบียนนักศึกษาของมหาวิทยาลัยเป็นระบบฐานข้อมูลการเรียนการสอนและผลการศึกษาของนักศึกษา ที่รองรับการใช้งานของนักศึกษา คณาจารย์ และบุคลากร เพื่ออำนวยความสะดวกในการจัดการการเรียนการสอน"

ตัวอย่างวัตถุประสงค์ของระบบฐานข้อมูลที่มหาวิทยาลัยกำหนดขึ้น และสอดรับกับภาระกิจ ได้แก่

เพื่อจัดเก็บ (เพิ่ม แก้ไข ลบ) ข้อมูลของนักศึกษา
เพื่อจัดเก็บ (เพิ่ม แก้ไข ลบ) ข้อมูลของคณาจารย์
เพื่อจัดเก็บ (เพิ่ม แก้ไข ลบ) ข้อมูลของรายวิชาที่เปิดสอน
เพื่อจัดเก็บ (เพิ่ม แก้ไข ลบ) ข้อมูลของทรัพยากรการเรียนการสอนเช่น ห้องเรียน
เพื่อจัดเก็บ (เพิ่ม แก้ไข ลบ) ข้อมูลการลงทะเบียนและผลการเรียนในแต่ละรายวิชาของนักศึกษา
เพื่อค้นหาและแสดงรายละเอียดของรายวิชา เช่น ห้องเรียน อาจารย์ผู้สอนและจำนวนผู้ลงทะเบียน
เพื่อค้นหาและแสดงรายละเอียดของนักศึกษารายบุคคล
เพื่อรายงานข้อมูลนักศึกษา คณาจารย์ และรายวิชา
เพื่อรายงานผลการเรียนของนักศึกษารายคน
เพื่อรายงานผลการเรียนเฉลี่ยของนักศึกษาทั้งรายวิชา
ฯลฯ

การวางแผนฐานข้อมูลยังเป็นการกำหนดแนวทางในการจัดสรรทรัพยากรเพื่อนำมาใช้ในการสร้างและใช้งานระบบฐานข้อมูลซึ่งได้แก่การจัดสรร บุคลากร งบประมาณ และทรัพยากรอื่นๆ เช่น เวลา หรืออุปกรณ์ทางด้านฮาร์ดแวร์และซอฟต์แวร์ ฯลฯ การกำหนดมาตรฐานอื่นๆ ในการปฏิบัติงานเพื่อพัฒนาระบบฐานข้อมูลเป็นสิ่งที่ควรกระทำอีกด้วย เช่น การกำหนดมาตรฐานและแนวการปฏิบัติในการเก็บรวบรวมข้อมูลให้ใช้เอกสารที่มีรูปแบบอย่างไร เราจะใช้แบบจำลองใดในการพัฒนาโปรแกรมประยุกต์ เป็นต้น การกำหนดมาตรฐานดังกล่าวอาจใช้เวลามากและยุ่งยากในช่วงเริ่มต้น แต่จะมีประโยชน์ในระยะยาวเช่นการกำหนดมาตรฐานในการตั้งชื่อข้อมูล จะทำให้การระบุข้อมูลที่ใช้ในการจัดเก็บไม่ซ้ำซ้อน เป็นต้น

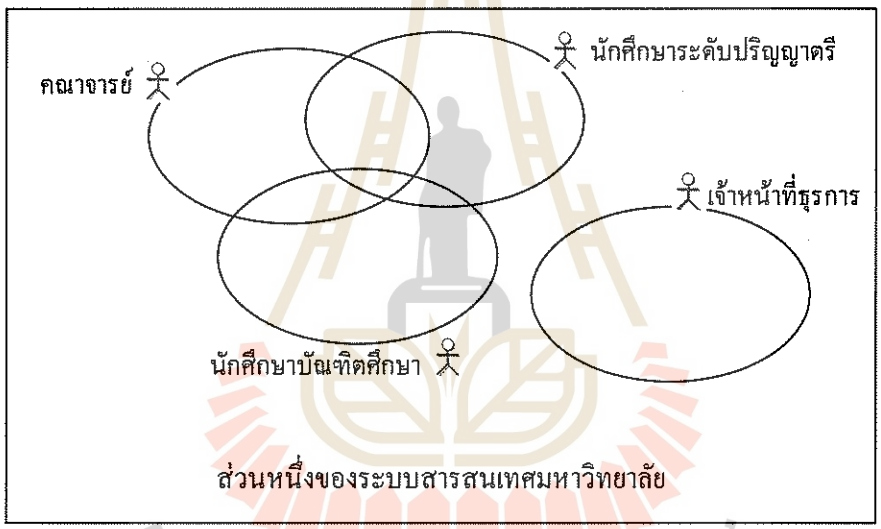
หลังจากวัตถุประสงค์ของการสร้างและใช้งานระบบฐานข้อมูลได้ถูกกำหนดขึ้น ตลอดจนการวางแผนขั้นตอนการปฏิบัติงาน การจัดสรรทรัพยากรและกำหนดมาตรฐานในการปฏิบัติงานแล้ว ขั้นต่อไปของการพัฒนาระบบฐานข้อมูลได้แก่การกำหนดขอบเขตของระบบ

2.5 การนิยามระบบ

กำหนดขอบเขตของการพัฒนาและใช้งานฐานข้อมูล รวมถึงระบุมุมมองของใช้งานทั้งหมด

การพัฒนาระบบฐานข้อมูลใดๆ ต้องมีการระบุขอบเขตของการพัฒนาและการใช้งานระบบ ตลอดจนรูปแบบการเชื่อมต่อกับระบบอื่น การกำหนดของเขตของระบบฐานข้อมูลนอกจากจะต้องคำนึงถึงผู้ใช้ในปัจจุบันแล้วยังจะต้องคำนึงถึงการรองรับการขยายขอบเขตการใช้งานในอนาคตอีกด้วย

กิจกรรมที่สำคัญอันหนึ่งของการนิยามระบบคือการระบุมุมมองของผู้ใช้ ซึ่งเป็นการระบุตัวผู้ใช้ทั้งหมดที่จะใช้งานระบบฐานข้อมูล ตลอดจนข้อมูลสำหรับผู้ใช้แต่ละคนและลักษณะหรือกลุ่มของกิจกรรมที่ผู้ใช้จะกระทำกับข้อมูล การระบุมุมมองของผู้ใช้เป็นแนวปฏิบัติเพื่อยืนยันว่าไม่มีบุคคลใดที่เราละเลยหรือทำตกหล่นในฐานะผู้ใช้งานระบบ มิเช่นนั้นการวิเคราะห์ความต้องการของผู้ใช้อาจตกหล่นไม่สมบูรณ์ได้ การระบุมุมมองของผู้ใช้แต่ละคนยังจะช่วยให้การวิเคราะห์ความต้องการมีความสะดวกขึ้นโดยเราสามารถวิเคราะห์ความต้องการของผู้ใช้ทีละกลุ่มแล้วนำความต้องการมารวมกันภายหลัง จะลดความซับซ้อนของการออกแบบขนาดใหญ่ได้ การระบุมุมมองของผู้ใช้ยังครอบคลุมถึงการระบุข้อมูลที่จะต้องจัดเก็บ และธุรกรรมของผู้ใช้ซึ่งหมายถึงโดยแท้จริงแล้วเราจะทำอะไรกับข้อมูลที่เราจัดเก็บบ้าง



รูปที่ 1** แสดงตัวอย่างแนวคิดของมุมมองการใช้ข้อมูลของผู้ใช้ที่สามารถมีข้อมูลที่ใช้ร่วมกัน

จากแผนภาพแสดงมุมมองของผู้ใช้ของระบบฐานข้อมูล ผู้ใช้แต่ละคนจะมีมุมมองในการใช้งานข้อมูลไม่เหมือนกัน อย่างไรก็ตามผู้ใช้บางกลุ่มอาจใช้งานกลุ่มข้อมูลเดียวกันเช่น นักศึกษามีความจำเป็นในการเรียกดูข้อมูลรายวิชาเพื่อทำการลงทะเบียนเรียนในขณะที่อาจารย์ผู้สอนมีความต้องการแสดงข้อมูลรายวิชาที่ตนเองสอนเพื่อดูห้องสอน

2.6 การรวบรวมและวิเคราะห์ความต้องการ

การรวบรวมและวิเคราะห์ข้อมูลตลอดจนสารสนเทศที่องค์กรมีอยู่รวมถึงที่ต้องการใช้เพื่อระบุว่าจะต้องจัดเก็บและใช้งานข้อมูลอะไรบ้าง ใครเป็นผู้จัดเก็บและเรียกใช้ ตลอดจนจะจัดเก็บและใช้งานอย่างไร ในระบบฐานข้อมูล

การรวบรวมและวิเคราะห์ข้อมูลที่จะต้องจัดเก็บ จัดการและเรียกใช้นั้นเรารวบรวมข้อมูลจากผู้ใช้โดยจำแนกกลุ่มผู้ใช้ตามบทบาทของการใช้งาน และอาจจำแนกตามกลุ่มงาน (ลักษณะของกลุ่มงานที่ต่างกัน เช่น แผนก ฝ่าย หน่วยงาน รวมถึงการแบ่งกลุ่มงานที่อาจเฉพาะเจาะจงหรือกว้างกว่านี้ เป็นต้น) การรวบรวมข้อมูลอาศัยเทคนิคที่เกรียกว่าการหาข้อเท็จจริง (fact-finding) ดังต่อไปนี้

เทคนิคการหาข้อเท็จจริงในการรวบรวมข้อกำหนดของระบบ

1) การตรวจสอบเอกสาร
2) การสัมภาษณ์
3) การสังเกตการณ์
4) การวิจัย
5) การใช้แบบสอบถาม

**

การรวบรวมและวิเคราะห์ความต้องการของผู้ใช้จะต้องได้มาซึ่งคำอธิบายข้อมูลต่างๆ ของข้อมูลที่สร้างขึ้นและคำอธิบายข้อมูลที่ใช้ทั้งหมด วิธีการสร้างและการใช้งานข้อมูลทำอย่างไรโดยละเอียด มีข้อกำหนดอื่นใดอีกหรือไม่ที่ระบบฐานข้อมูลต้องรองรับ

ผลลัพธ์ที่ได้จากการรวบรวมและวิเคราะห์ความต้องการของผู้ใช้คือรายการข้อกำหนดที่ระบบต้องรองรับ โดยทั่วไปผลลัพธ์ที่ได้อยู่ในรูปแบบของเอกสารข้อกำหนดด้านความต้องการ (requirement specifications) สำหรับฐานข้อมูลใหม่ที่จะพัฒนาขึ้น การรวบรวมและวิเคราะห์ข้อมูลควรมีการกำหนดขอบเขต วางแผน และจัดสรรเวลาอย่างเหมาะสม ไม่ควรใช้เวลากับการรวบรวมความต้องการที่มากเกินไปหรือไม่จบไม่สิ้นในบางครั้ง ทำให้การใช้เวลาไม่คุ้มค่า ในทางกลับกันเราไม่ควรใช้เวลาน้อยเกินไปจนการรวบรวมและวิเคราะห์ความต้องการของผู้ใช้ไม่ครอบคลุมความต้องการทั้งหมด

การวิเคราะห์ระบบนั้นมีความสำคัญเป็นอย่างมาก หากการวิเคราะห์ระบบไม่สมบูรณ์ การรองรับความต้องการของผู้ใช้ไม่ครบถ้วน ระบบฐานข้อมูลที่พัฒนาขึ้นจะไม่สามารถรองรับการใช้งานได้อย่างมีประสิทธิภาพ ผู้ใช้อาจรำคาญ ระบบอาจได้รับการใช้งานน้อย หรือไม่มีผู้ใช้เลย ซึ่งเป็นการล้มเหลวของการสร้างระบบฐานข้อมูล การกำหนดความสามารถของระบบที่เกินความต้องการของผู้ใช้มากเกินไปอาจทำให้การพัฒนาซับซ้อน การใช้งานและบำรุงรักษายากลำบาก ด้วยความสำคัญนี้การวิเคราะห์ระบบจึงเป็นศาสตร์ที่ได้รับการพัฒนามาอย่างกว้างขวางและยาวนาน มีการเสนอแนวปฏิบัติที่หลากหลาย และมีเครื่องมือสำหรับช่วยในการวิเคราะห์ เช่นการใช้แผนภาพกระแสข้อมูล (data flow diagram—DFD) การวิเคราะห์และออกแบบอย่างมีโครงสร้าง (structured analysis and design—SAD) การใช้แผนภาพ unified modeling language—UML เป็นต้น ซึ่งการวิเคราะห์ระบบสารสนเทศและโปรแกรม

ประยุกต์นั้นเกินขอบเขตของการพัฒนาระบบฐานข้อมูล และไม่ได้อธิบายในรายละเอียด อย่างไรก็ตามในที่นี้จะกล่าวถึงทางเลือกหนึ่งของแนวปฏิบัติที่ใช้ในการจัดการกับการรวบรวมและวิเคราะห์ความต้องการของผู้ใช้งานระบบฐานข้อมูลที่มีมุมมองของผู้ใช้หลายมุมมอง ซึ่งแบ่งออกเป็น 3 วิธี ซึ่งมีความเกี่ยวข้องกับการออกแบบฐานข้อมูลระดับแนวคิด (ขั้นตอนถัดไปของวงจรพัฒนาระบบฐานข้อมูล) ได้แก่

- วิธีการแบบรวมศูนย์ (centralized approach)
- วิธีการแบบบูรณาการมุมอง (view integration approach)
- วิธีการแบบผสาน (combination approach)

วิธีการแบบรวมศูนย์ (centralized approach)

เป็นการนำความต้องการของผู้ใช้ทุกมุมมองมารวมกันก่อนแล้วจึงนำไปออกแบบฐานข้อมูลระดับแนวคิด กล่าวคือ เรานำความต้องการของผู้ใช้ที่รวบรวมได้จากแต่ละมุมมองมารวมกันเป็นความต้องการของผู้ใช้ทั้งระบบ ความต้องการที่ซ้ำซ้อนกันจะผสานเข้าด้วยกัน ผลลัพธ์ที่ได้คือเอกสารระบุรายการความต้องการของผู้ใช้ที่ได้ผสานความต้องการของผู้ใช้จากทุกๆ มุมมองเข้าด้วยกันแล้ว จากนั้นเรานำความต้องการของผู้ใช้ในลักษณะรวมศูนย์ที่ได้ไปใช้ประโยชน์ในขั้นต่อไปของวงจรการออกแบบระบบฐานข้อมูล ได้แก่การออกแบบฐานข้อมูลระดับแนวคิด ในที่นี้คือแบบจำลอง ER จากนั้นทำการออกแบบระดับกายภาพต่อไป

รูป** Central

จากภาพ 1** แสดงให้เห็นถึงขั้นตอนการรวบรวมและวิเคราะห์ความต้องการโดยใช้วิธีการรวมศูนย์ ผู้พัฒนาระบบฐานข้อมูลทำการรวบรวมความต้องการของผู้ใช้มุมมองต่างๆ ด้วยเทคนิคการหาข้อเท็จจริง จากนั้นวิเคราะห์ความต้องการของผู้ใช้ทุกมุมมองเป็นข้อๆ นำมาประมวลรวมกัน ความต้องการ 1 ข้ออาจรองรับมุมมองของผู้ใช้หลายๆ มุมมอง วิธีการนี้เหมาะสมกับระบบที่มีขนาดไม่ใหญ่และไม่มีความซับซ้อนมากนัก โดยเฉพาะในกรณีที่มุมมองของผู้ใช้ไม่มีการซ้อนทับกัน

วิธีการแบบบูรณาการมุมอง (view integration approach)

วิธีการรวบรวมและวิเคราะห์ความต้องการของผู้ใช้แบบบูรณาการมุมมอง ต่างจากแบบรวมศูนย์ตรงที่หลังจากวิเคราะห์ความต้องการของผู้ใช้แต่ละมุมมองแล้วเราจะยังไม่นำความต้องการทั้งหมดมารวมกัน แต่จะนำไปออกแบบฐานข้อมูลระดับแนวคิดหรือแบบจำลอง ER ก่อน จากนั้นค่อยๆ บูรณาการแบบจำลอง ER ที่ได้มา โดยรวมเอนทิตีและความสัมพันธ์ต่างๆ เข้าด้วยกัน อธิบายได้ดังแผนภาพ 1** วิธีนี้เหมาะสมกับระบบฐานข้อมูลที่มีขนาดใหญ่และซับซ้อน อาจมีผู้ใช้หลายๆ มุมมองและมีการซ้อนทับกันของข้อมูลหรือกิจกรรมที่ใช้ข้อมูล การรวมกันของ

ความต้องการของแต่ละคนจัดการยาก เนื่องจากมีรายละเอียดปลีกย่อยของข้อมูลและธุรกรรมที่ต่างกันในแต่ละมุมมองที่ใช้ข้อมูลซ้อนทับกัน

**

วิธีการแบบผสมผสาน (combination approach)

เราสามารถนำเอาวิธีทั้งแบบรวมศูนย์และแบบบูรณาการมุมมองผู้ใช้มาประยุกต์ใช้ร่วมกันได้ในกรณีที่ระบบฐานข้อมูลมีความซับซ้อนมากๆ ผู้ใช้บางกลุ่มอาจมีมุมมองการใช้ข้อมูลที่คล้ายคลึงกันเช่นการทำงานคนละหน้าที่แต่อยู่ในแผนกเดียวกัน ทำให้เราสามารถรวมมุมมองนั้นๆ เข้าด้วยกันได้ก่อน อาจจะ 2-3 มุมมอง ทำให้เราได้รายการความต้องการของผู้ใช้เป็นกลุ่มๆ จากนั้นทำการออกแบบฐานข้อมูลระดับแนวคิดโดยแยกการออกแบบแบบจำลองข้อมูลสำหรับผู้ใช้เป็นชุดๆ สุดท้ายเราสามารถนำแบบจำลองข้อมูลหรือแบบจำลอง ER มารวมกันได้ในภายหลัง อย่างไรก็ตามเราไม่จำเป็นต้องรวมกลุ่มของมุมมองก่อนเสมอไป ตัวอย่างการรวบรวมและวิเคราะห์ความต้องการของผู้ใช้ด้วยวิธีผสมผสานนี้ แสดงได้ดังภาพ 1** วิธีการรวบรวมและวิเคราะห์ความต้องการแบบนี้เหมาะสำหรับระบบขนาดใหญ่และซับซ้อนมากๆ เช่นการออกแบบฐานข้อมูลของทั้งบริษัท รองรับการทำงานของแผนกหลายๆ แผนก

2.7 การออกแบบฐานข้อมูล

การออกแบบฐานข้อมูลให้รองรับวัตถุประสงค์และเป้าหมายการดำเนินการขององค์กร

หลังจากได้เอกสารระบุรายการความต้องการของผู้ใช้และข้อกำหนดของระบบทั้งหมดแล้ว เราข้อกำหนดมาออกแบบระบบฐานข้อมูล และออกแบบโปรแกรมประยุกต์ให้สามารถรองรับข้อกำหนด วิธีการออกแบบฐานข้อมูลมี 2 วิธีได้แก่การออกแบบจากล่างขึ้นบน (bottom-up) และการออกแบบจากบนลงล่าง (top-down)

วิธีการออกแบบจากล่างขึ้นบนสามารถทำได้โดยการวิเคราะห์ข้อมูลในหน่วยย่อยที่สุดก่อนที่เรียกว่าแอทริบิวต์ แล้วนำมารวมเป็นความสัมพันธ์ขนาดใหญ่ จากนั้นทำการแยกกลุ่มของข้อมูลออกจากกันให้เป็นเอนทิตีต่างๆ วิธีดังกล่าวต้องอาศัยเทคนิคที่เรียกว่าการทำให้เป็นรูปแบบบรรทัดฐาน (normalization)

**

เทคนิคการออกแบบจากล่างขึ้นบนนี้เหมาะกับฐานข้อมูลขนาดเล็กๆ เท่านั้น สำหรับฐานข้อมูลขนาดใหญ่ข้อมูลมีอยู่เป็นจำนวนมาก การนำหน่วยย่อยของข้อมูลจำนวนมหาศาลมาสร้างความสัมพันธ์รวมเป็นเรื่องที่ยุ่งยากและซับซ้อน

วิธีการออกแบบจากบนลงล่างเป็นการมองภาพรวมของระบบก่อน ได้แก่มุมมองของผู้ใช้ เอนทิตี ระบุความความสัมพันธ์ของเอนทิตี จากนั้นค่อยลงลึกถึงการระบุข้อมูลหน่วยย่อยของเอนทิตี ซึ่งตรงข้ามกับการออกแบบด้วยวิธีจากล่างขึ้นบน การออกแบบจากบนลงล่างนี้คือการออกแบบด้วยแบบจำลองข้อมูลเชิงความหมาย หรือการสร้างแบบจำลอง ER ที่นิยมกระทำกันนั่นเอง อย่างไรก็ตามเราสามารถนำเอาเทคนิคการทำให้เป็นรูปแบบบรรทัดฐานมาช่วยในการออกแบบจากบนลงล่างได้เช่นเดียวกัน การทำให้เป็นรูปแบบบรรทัดฐานเป็นการตรวจสอบแบบจำลองข้อมูล เพื่อกำจัดความซ้ำซ้อนของข้อมูลออกไป ขั้นตอนของการทำให้เป็นรูปแบบบรรทัดฐานได้อธิบายไว้ในบทที่ 5

**

ขั้นตอนการออกแบบฐานข้อมูลโดยมาตรฐานนั้นมี 3 ขั้นตอน เราจะเริ่มจากการสร้างแบบจำลองข้อมูลเชิงความหมาย ซึ่งจะได้มาซึ่งฐานข้อมูลระดับแนวคิด ซึ่งไม่ยึดติดกับอุปกรณ์ฮาร์ดแวร์และซอฟต์แวร์ใดๆ จากนั้นเป็นการแปลงแบบจำลองข้อมูลที่ได้เป็นแบบจำลองข้อมูลในระดับตรรกะ แบบจำลองข้อมูลในระดับนี้อาศัยแบบจำลองข้อมูลที่สามารถจัดการข้อมูลได้อย่างมีประสิทธิภาพ เช่น แบบจำลองข้อมูลเชิงสัมพันธ์ แบบจำลองข้อมูลเชิงเครือข่าย แบบจำลองข้อมูลเชิงวัตถุ เป็นต้น จากนั้นเป็นการกำหนดรูปแบบต่างๆ ของการจัดเก็บข้อมูลในระดับโครงสร้างของไฟล์ที่จะจัดเก็บในอุปกรณ์จัดเก็บข้อมูลซึ่งเรียกว่าการออกแบบระดับกายภาพ การออกแบบฐานข้อมูลทั้ง 3 ขั้นตอนมีระเบียบวิธีที่ชัดเจนสามารถนำไปปฏิบัติดังที่อธิบายต่อไปนี้ สำหรับตัวอย่างการออกแบบโดยระเบียบวิธีการออกแบบฐานข้อมูลนี้แสดงในภาคผนวกเรื่องการออกแบบฐานข้อมูล 7-Elephant

2.7.1 การออกแบบฐานข้อมูลระดับแนวคิด

การสร้างแบบจำลองข้อมูลเชิงความหมายที่แสดงถึงความสัมพันธ์ของข้อมูลต่างๆ ที่จะใช้ในองค์กร โดยเป็นอิสระจากฮาร์ดแวร์และซอฟต์แวร์ที่จะนำมาใช้

ขั้นตอนแรกของการออกแบบฐานข้อมูล คือการออกแบบระดับแนวคิดเกี่ยวกับข้อมูลที่ต้องใช้ของหน่วยงานใดหน่วยงานหนึ่งในองค์กรที่ต้องการใช้งานระบบฐานข้อมูล ผลลัพธ์ที่ได้จากการออกแบบระดับนี้ได้แก่แบบจำลองข้อมูลในระดับแนวคิดโดยทั่วไปคือแบบจำลอง ER ซึ่งไม่ยึดติดกับรูปแบบการจัดเก็บของระบบจัดการฐานข้อมูลใดๆ ฮาร์ดแวร์ ซอฟต์แวร์ ระบบปฏิบัติการ โปรแกรมประยุกต์ และภาษาการเขียนโปรแกรมใดๆ แบบจำลอง ER ที่ได้มานั้นสามารถนำมาใช้เป็นเครื่องมือในการทำความเข้าใจระหว่างผู้พัฒนาระบบฐานข้อมูลด้วยกัน นำมายืนยันความถูกต้องของการออกแบบโดยการตรวจสอบการรองรับความต้องการของผู้ใช้ รวมถึงใช้สื่อสารกับผู้ใช้ได้

ระเบียบวิธีการออกแบบฐานข้อมูลระดับแนวคิดมีขั้นตอนดังนี้

1. ระบุเอนทิตีที่เกี่ยวข้อง
2. ระบุความสัมพันธ์ของเอนทิตี

3. ระบุข้อมูลหน่วยย่อยหรือที่เรียกว่าแอทริบิวต์ที่สัมพันธ์กับเอนทิตีและความสัมพันธ์ของเอนทิตี
4. ระบุขอบเขตของแอทริบิวต์
5. ระบุคีย์หลักที่ใช้แทนข้อมูลแต่ละระเบียนในเอนทิตีและความสัมพันธ์
6. กำหนดข้อกำหนดหรือเงื่อนไขของข้อมูลโดยการใช้คุณสมบัติที่แบบจำลอง ER รองรับ ขั้นตอนนี้เป็นขั้นตอนเสริมเท่านั้นซึ่งไม่จำเป็นต้องปฏิบัติ
7. ตรวจสอบแบบจำลองที่ได้มาเพื่อกำจัดความซ้ำซ้อนของข้อมูล
8. ตรวจสอบแบบจำลองระดับแนวคิดที่ออกแบบว่าสามารถรองรับธุรกรรมของผู้ใช้ได้ทั้งหมด
9. ยืนยันกับผู้ใช้

ผลลัพธ์ที่ได้เป็นแบบจำลองข้อมูลระดับแนวคิดซึ่งเป็นสิ่งที่จะนำไปใช้สำหรับขั้นตอนถัดไปของการออกแบบฐานข้อมูล

2.7.2 การออกแบบฐานข้อมูลระดับตรรกะ

การสร้างแบบจำลองข้อมูลตามแบบจำลองข้อมูลที่เลือก เช่นแบบจำลองข้อมูลเชิงสัมพันธ์ แต่ยังเป็นอิสระกับผู้ผลิตระบบจัดการฐานข้อมูล

ขั้นตอนหลักที่สองของระเบียบวิธีการออกแบบฐานข้อมูลคือการออกแบบแบบจำลองข้อมูลของข้อมูลที่ต้องใช้สำหรับหน่วยงานใดๆ ที่จะใช้ระบบฐานข้อมูล แบบจำลองข้อมูลที่จะต้องออกแบบในขั้นนี้เป็นแบบจำลองข้อมูลเชิงตรรกะโดยการจับคู่หรือแปลงสิ่งที่สัมพันธ์กันจากแบบจำลองระดับแนวคิดมาเป็นแบบจำลองข้อมูลเชิงระดับตรรกะ แบบจำลองข้อมูลในระดับนี้ต่างจากแบบจำลองข้อมูลระดับแนวคิดในลักษณะที่แบบจำลองระดับนี้ถูกออกแบบอ้างอิงกับแบบจำลองข้อมูลแบบใด เช่น แบบจำลองข้อมูลเชิงเครือข่าย แบบจำลองข้อมูลเชิงวัตถุ หรือแบบจำลองข้อมูลเชิงสัมพันธ์ เป็นต้น ในที่นี้คือการแปลงแบบจำลอง ER เป็นแบบจำลองข้อมูลเชิงสัมพันธ์ แบบจำลองที่ได้ยังคงเป็นอิสระจากโครงสร้างที่จะมารองรับฐานข้อมูลอยู่ เช่น ไม่ยึดติดกับผู้ผลิตของระบบจัดการฐานข้อมูล รูปการการจัดการและการเข้าถึงไฟล์ การสร้างไฟล์ดัชนีเป็นต้น

ผู้ออกแบบต้องทำการตรวจสอบแบบจำลองที่ได้มาว่าสามารถรองรับความต้องการตลอดจนธุรกรรมของผู้ใช้เรายังสามารถอาศัยเทคนิคการกำจัดความซ้ำซ้อนของข้อมูลด้วยการทำให้เป็นรูปแบบบรรทัดฐานในขั้นตอนนี้ การซ้ำซ้อนของข้อมูลจะทำให้เกิดความผิดพลาดของข้อมูลได้ในกรณีที่ข้อมูลหน่วยเดียวกันแต่ปรากฏอยู่ 2 ครั้งในที่ๆ ต่างกัน การแก้ไขข้อมูลหนึ่งโดยไม่แก้ไขข้อมูลอีกที่หนึ่งให้ตรงกันทำให้ข้อมูลเกิดความขัดแย้งกันได้
**

ระเบียบวิธีการออกแบบฐานข้อมูลระดับแนวคิดมีระเบียบวิธีดังนี้

1. สร้างรีเลชันสำหรับแบบจำลองระดับตรรกะจากแบบจำลองระดับแนวคิด

2. ตรวจสอบความซ้ำซ้อนของแบบจำลองด้วยการทำให้เป็นรูปแบบบรรทัดฐาน
3. ตรวจสอบรีเลชันว่าสามารถรองรับธุรกรรมของผู้ใช้ได้
4. ตรวจสอบข้อกำหนดหรือเงื่อนไขเพื่อความคงสภาพของข้อมูล **
5. ยืนยันแบบจำลองกับผู้ที่จะใช้ระบบฐานข้อมูล
6. ผนวกแบบจำลองข้อมูลย่อยเป็นแบบจำลองข้อมูลโดยรวม (ในกรณีที่ใช้วิธีการวิเคราะห์และออกแบบแบบบูรณการมุมมองที่ได้กล่าวถึงในหัวข้อที่ 6** ขั้นตอนนี้ไม่จำเป็นต้องปฏิบัติ)
7. ตรวจสอบว่าแบบจำลองสามารถรองรับการขยายขอบเขตการใช้งานในอนาคตหรือไม่อย่างไร

แบบจำลองข้อมูลระดับตรรกะนี้มีความสำคัญอย่างมากในการพัฒนาระบบฐานข้อมูล นอกจากจะเป็นตัวแบบในการออกแบบฐานข้อมูลในขั้นถัดไปแล้ว แบบจำลองนี้ยังใช้ในการรองรับการบำรุงรักษาระบบฐานข้อมูล การปรับปรุง เปลี่ยนแปลง และพัฒนาระบบฐานข้อมูลในกรณีที่มีการเปลี่ยนความต้องการของผู้ใช้อีกด้วย แบบจำลองข้อมูลระดับแนวคิดนี้จะได้รับการพัฒนาไปให้สอดคล้องกับสภาพปัจจุบันของระบบจัดการฐานข้อมูล

พจนานุกรมข้อมูล**

2.7.3 การออกแบบฐานข้อมูลระดับกายภาพ

การสร้างรายละเอียดทั้งหมดสำหรับจัดเก็บข้อมูลลงในหน่วยความจำสำรอง ได้แก่การสร้างรีเลชันหลัก รูปแบบการจัดเก็บของไฟล์ในระบบจัดการฐานข้อมูล การสร้างไฟล์ดรรชนี เพื่อให้การเข้าถึงข้อมูลเป็นไปอย่างมีประสทธิการ รวมถึงการกำหนดข้อกำหนดเพื่อความคงสภาพข้อมูลและข้อกำหนดด้านความปลอดภัยต่างๆ

ขั้นตอนที่ 3 และเป็นขั้นตอนสุดท้ายของการออกแบบฐานข้อมูลได้แก่การระบุรายละเอียดของรูปแบบการจัดการกับไฟล์ข้อมูลที่จะจัดเก็บบนหน่วยความจำสำรอง ซึ่งต่างจากขั้นตอนการออกแบบฐานข้อมูลหลักใน 2 ขั้นตอนแรก ขั้นตอนการออกแบบฐานข้อมูลระดับกายภาพนี้จะอ้างอิงกับระบบจัดการฐานข้อมูลที่เฉพาะเจาะจงลงไปถึงผลิตภัณฑ์หรือยี่ห้อของระบบจัดการฐานข้อมูล การออกแบบฐานข้อมูลในระดับนี้มีความสัมพันธ์กับการออกแบบฐานข้อมูลเชิงแนวคิดระดับหนึ่ง กล่าวคือในกรณีที่มีการเปลี่ยนแปลงรีเลชันในฐานข้อมูลระดับตรรกะแล้ว เราต้องปรับแก้รีเลชันที่จัดเก็บในตัวฐานข้อมูลด้วย

การออกแบบฐานข้อมูลระดับกายภาพเป็นการกำหนดเกี่ยวกับการจัดการระบบฐานข้อมูลในเรื่องต่อไปนี้

- การสร้างตัวรีเลชันสำหรับเอนทิตีและความสัมพันธ์ในฐานข้อมูล และการสร้างข้อกำหนดหรือเงื่อนไขเพื่อความคงสภาพข้อมูล
- กำหนดโครงสร้างไฟล์ข้อมูลที่ใช้ในการจัดเก็บข้อมูลในระบบฐานข้อมูล ปรับแต่งวิธีการเข้าถึงข้อมูลให้มีประสิทธิภาพ
- การกำหนดค่าของระบบในด้านความมั่นคงปลอดภัยของข้อมูล

ระเบียบวิธีการออกแบบฐานข้อมูลระดับกายภาพมีขั้นตอนดังนี้

1. แปลงแบบจำลองข้อมูลระดับแนวคิดสำหรับการจัดเก็บข้อมูลในระบบจัดการฐานข้อมูลที่เลือกใช้
    1. ออกแบบรีเลชันหลัก
    2. Design representation of derived data**
    3. ออกแบบข้อกำหนดหรือเงื่อนไขเพื่อความคงสภาพของข้อมูล
2. ออกแบบโครงสร้างไฟล์ที่ใช้จัดเก็บข้อมูลในระบบจัดการฐานข้อมูลและการกำหนดไฟล์ดัชนี
    1. วิเคราะห์ธุรกรรม การวิเคราะห์ธุรกรรมที่ใช้งานทำให้เราทราบว่าเราใช้งานฐานข้อมูลในลักษณะใดบ่อยครั้ง ทำให้เรากำหนดโครงสร้างข้อมูลและดัชนีได้ถูกต้อง
    2. เลือกโครงสร้างไฟล์
    3. เลือกสร้างไฟล์ดัชนี
    4. ประเมินขนาดความจุของหน่วยคำวามจำที่ต้องใช้
3. ออกแบบวิวของผู้ใช้
4. ออกแบบกลไกเพื่อควบคุมความมั่นคงปลอดภัยของระบบฐานข้อมูล
5. Consider the introduction of controlled redundancy
6. ดูแลและปรับแต่งระบบที่ใช้งานจริง

** ANSI SPARC

2.8 การเลือกระบบจัดการฐานข้อมูล

การเลือกระบบจัดการฐานข้อมูลสำหรับระบบฐานข้อมูลที่จะพัฒนา เป็นการเลือกในแง่ของผู้ผลิตหรือยี่ห้อของระบบจัดการฐานข้อมูล

ในกรณีที่ไม่มีการใช้งานระบบจัดการฐานข้อมูลอยู่เลย การเลือกระบบจัดการฐานข้อมูลที่เหมาะสมเป็นสิ่งที่จำเป็น ทั้งนี้การเลือกระบบจัดการฐานข้อมูลมารองรับระบบฐานข้อมูลที่จะพัฒนามักกระทำในขั้นตอนนี้ อย่างไรก็ตามการเลือกระบบจัดการฐานข้อมูลสามารถทำ ณ ขณะใดก็ได้หากมีข้อมูลที่เพียงพอในการตัดสินใจ ขั้นตอนหลักในการเลือกฐานข้อมูลได้แก่

- **กำหนดวัตถุประสงค์และขอบเขตของการเลือกใช้ระบบจัดการฐานข้อมูล**
  เป็นขั้นตอนเกี่ยวกับการจัดการและการวางแผนในการคัดเลือกระบบจัดการฐานข้อมูล กล่าวคือ กำหนดวัตถุประสงค์ของการเลือกฐานข้อมูลคืออะไร เพื่ออะไร และจะมีขั้นตอนอย่างไร โดยอาศัยข้อมูลที่ได้จากขั้นตอนการพัฒนาระบบฐานข้อมูล

- **สร้างรายการระบบจัดการฐานข้อมูลที่คาดว่าจะเลือกใช้**
  ระบบจัดการฐานข้อมูลมีมากมายให้เลือกใช้ เรามีความจำเป็นต้องคัดเลือกระบบจัดการฐานข้อมูลที่มีแนวโน้มว่าจะใช้มาจำนวนหนึ่ง เช่น 3-4 ราย ซึ่งการได้มาซึ่งรายการระบบจัดการฐานข้อมูลในขั้นแรกนี้มักจะสามารถกระทำได้ง่าย สาเหตุมาจากระบบฐานข้อมูลที่นิยมในระดับแนวหน้ามีจำกัด ราคา ข้อจำกัดด้านระบบที่มีอยู่แล้ว ซอฟต์แวร์ ฮาร์ดแวร์ นโยบายของบริษัทที่เราไม่สามารถหลีกเลี่ยงได้ ข้อมูลที่มีอยู่เช่นบริการหลังการขายกับรายที่เคยใช้บริการ
- **ประเมินระบบจัดการฐานข้อมูลที่ได้เลือกมาแล้ว**
  ในกรณีที่มีรายการของระบบจัดการฐานข้อมูลในตัดสินใจเลือกใช้ มีคุณสมบัติใกล้เคียงกัน เราสามารถนำมาเปรียบเทียบกันโดยกำหนดเกณฑ์ที่จะใช้พิจารณา (เช่น ราคา การบริการหลังการขาย ความน่าเชื่อถือ ประสิทธิภาพ การรองรับการสำรองข้อมูล การเชื่อมต่อกับระบบอื่น ความสามารถด้านเทคนิคอื่นๆ ฯลฯ) ระบุระดับความสำคัญของเกณฑ์แต่ละเกณฑ์ จากนั้นให้คะแนนผลิตภัณฑ์แต่ละผลิตภัณฑ์
- **สรุป ระบุระบบจัดการฐานข้อมูลที่ควรเลือกใช้ และนำเสนอ**
  นำคะแนนรวมของแต่ละผลิตภัณฑ์ที่ได้จากการประเมินมาเปรียบเทียบ ในกรณีที่ไม่มีเงื่อนไขอื่นๆ อีกเราสามารถเลือกใช้ระบบฐานข้อมูลที่ได้รับคะแนนประเมินมากที่สุด ทั้งนี้ต้องทำการสรุป และการนำเสนอในรูปแบบที่เข้าใจได้ง่าย โดยอธิบายตั้งแต่การวางแผนการเลือกระบบจัดการฐานข้อมูล วิธีการ ข้อสรุปและข้อเสนอแนะ

2.9 การออกแบบระบบ

การออกแบบส่วนติดต่อกับผู้ใช้และออกแบบโปรแกรมประยุกต์ที่จะมาใช้ระบบฐานข้อมูล

การออกแบบระบบหรือการออกแบบโปรแกรมประยุกต์ที่จะมาใช้งานระบบฐานข้อมูล สามารถทำคู่ขนานกับการออกแบบฐานข้อมูลได้ โดยปกติออกแบบระบบจะไม่เสร็จสมบูรณ์ถ้าไม่ทราบโครงสร้างของฐานข้อมูลที่แน่นอน และในทางกลับกัน ระบบฐานข้อมูลเป็นระบบที่รับข้อมูลมาจากโปรแกรมประยุกต์ ซึ่งเป็นผู้ระบุกระบวนการการไหลของข้อมูลซึ่งฐานข้อมูลต้องรองรับ ดังนั้นการออกแบบฐานข้อมูลและการออกแบบระบบจะต้องทำไปพร้อมๆ กัน และสอดคล้องสัมพันธ์กัน

การออกแบบระบบครอบคลุมถึงการออกแบบโปรแกรมประยุกต์เพื่ออำนวยความสะดวกให้กับผู้ใช้ดำเนินธุรกรรมได้ครบถ้วนตามความต้องการของผู้ใช้ และการออกแบบหน้าตาของระบบซึ่งเป็นส่วนสำคัญอย่างยิ่งของระบบ ส่วนติดต่อกับผู้ใช้ควรเป็นมิตรกับผู้ใช้ ใช้งานได้ง่าย ไม่สลับซับซ้อน ฯลฯ แนวปฏิบัติในการออกแบบธุรกรรมและส่วนติดต่อกับผู้ใช้อธิบายได้ดังนี้

2.9.1 การออกแบบธุรกรรม

ธุรกรรมคือการดำเนินการใดๆ ของผู้ใช้เพื่อให้กิจกรรมๆ หนึ่งสำเร็จ โดยการเรียกใช้หรือเปลี่ยนแปลงข้อมูลในระบบฐานข้อมูล

ธุรกรรมคือกิจกรรมที่เกิดขึ้นจริงในโลกแห่งความเป็นจริง เช่นธุรกรรมการลงทะเบียนะเรียน การฝาก ถอนเงิน การเรียกดูยอดบัญชี ธุรกรรมอาจประกอบไปด้วยการดำเนินงานย่อยหามากกว่า 1 ขั้นตอนใน 1 ธุรกรรมเช่น การโอนเงินประกอบไปด้วยขั้นตอนการหักบัญชีของผู้โอน และการเพิ่มยอดบัญชีของบัญชีปลายทางถือเป็น 1 ธุรกรรม

การออกแบบธุรกรรมต้องคำนึงถึง

- ข้อมูลที่ใช้
- ลักษณะการใช้งานข้อมูลสำหรับกิจกรรมนั้นๆ กล่าวคือเราจะทำอะไรกับข้อมูลหรือนำข้อมูลมาใช้ประโยชน์อย่างไร
- ผลลัพธ์ของธุรกรรม
- ความสำคัญต่อผู้ใช้
- การประมาณการความถี่ในการใช้งานธุรกรรม

ผู้พัฒนาระบบฐานข้อมูลจะต้องนำรายการความต้องการของผู้ใช้และข้อกำหนดของระบบฐานข้อมูลมาทำการออกแบบธุรกรรมโดยระบุสิ่งที่กล่าวถึงข้างต้นให้ครบและครอบคลุมความต้องการของผู้ใช้ทั้งหมด ธุรกรรมบางประเภทอาจจะต้องได้รับการออกแบบเพื่อจัดการเป็นพิเศษ ประเภทของธุรกรรมแบ่งออกเป็น 3 ประเภท ได้แก่

- ธุรกรรมในการค้นคืนข้อมูล (retrieval transaction) เช่น การแสดงยอดเงินในบัญชี
- ธุรกรรมการปรับปรุงข้อมูล (update transaction) เช่น การเพิ่มข้อมูลบัญชีใหม่จากการเปิดบัญชี
- ธุรกรรมแบบผสม (mixed transaction) คือธุรกรรมที่ประกอบด้วยขั้นตอนย่อยทั้งการค้นคืนและปรับปรุงข้อมูล แสดงก่อนแล้วเพิ่ม เช่น การฝากเงิน เราจะต้องทำการค้นคืนข้อมูลของบัญชีเงินฝากเพื่อแสดงยอดคงเหลือในบัญชีก่อนทำการรวมเงินที่ฝากเข้าไป

ขั้นตอนย่อยในธุรกรรมที่มีขั้นตอนหลายขั้นตอนนั้นจะต้องได้ดำเนินการให้สำเร็จทุกขั้นตอน ธุรกรรมนั้นทั้งธุรกรรมจึงจะเรียกว่าสำเร็จ ดังตัวอย่างธุรกรรมการโอนเงินที่เป็นธุรกรรมเดียว หากการดำเนินการธุรกรรมการโอนเงินได้หักเงินออกจากบัญชีต้นทางแล้ว แต่ยังไม่ทันที่ระบบจัดเพิ่มยอดเงินกับบัญชีปลายทาง ระบบเกิดการล้มเหลวด้วยสาเหตุต่างๆ ธุรกรรมนั้นๆ จะต้องถูกยกเลิกและเงินของบัญชีต้นทางต้องเพิ่มกลับให้คงเดิม อย่างไรก็ตามหากการโอนเงินเป็นไปในรูปแบบ 2 ธุรกรรม กล่าวคือเจ้าของบัญชีทำการถอนเงินก่อน แล้วค่อยฝากเงินเข้าบัญชีปลายทาง เราจะต้องแยกธุรกรรมนี้ออกจากกันและไม่สัมพันธ์กัน

2.9.2 การออกแบบส่วนติดต่อกับผู้ใช้

Shneiderman, 1992 ได้เสนอแนะแนวปฏิบัติในการออกแบบส่วนติดต่อกับผู้ใช้ที่พึงกระทำไว้ดังนี้

- **ชื่อของแบบฟอร์มต้องสื่อความหมาย**
  ชื่อของแบบฟอร์มมักจะถูกแสดงในแถบหัวเรื่องของแบบฟอร์ม การตั้งชื่อต้องสื่อความหมายถึงข้อมูลและกิจกรรมที่กำลังปฏิบัติ เช่น เราตั้งชื่อแบบฟอร์มว่า "เพิ่มนักศึกษาใหม่" แทนที่จะใช้คำว่า "ข้อมูลนักศึกษา" สำหรับแบบฟอร์มกรอกข้อมูลนักศึกษาใหม่ที่ยังไม่เคยมีอยู่ในระบบฐานข้อมูล
- **คำสั่งที่เข้าใจได้ง่าย**
  ข้อความอธิบายการทำงานของฟอร์มหรือข้อความบนปุ่มคำสั่ง จะต้องชัดเจนเข้าใจได้ง่าย เช่น "คลิกที่นี่เพื่อไปยังขั้นตอนถัดไป"
- **การจัดลำดับช่องกรอกข้อมูลและการรวมกลุ่มของประเภทข้อมูล**
  ข้อมูลที่ควรจะกรอกก่อนควรจะมาก่อนเช่นชื่อ ตามด้วยนามสกุล ข้อมูลที่เป็นชุดเดียวกันควรจะจัดไว้ในกลุ่มช่องกรอกเดียวกัน เช่นข้อมูลของนักศึกษาประกอบด้วยกลุ่มของประวัติส่วนตัว ได้แก่ชื่อ นามสกุล ฯลฯ และกลุ่มของประวัติการเรียน เช่น ปีที่เข้าศึกษา หลักสูตรที่สังกัด เกรดเฉลี่ย ฯลฯ
- **การวางตำแหน่งขององค์ประกอบบนฟอร์มและรายงานได้สัดส่วน**
  พื้นที่ของหน้าจอควรถูกจัดสรรให้เหมาะสม ไม่มีช่องกรอกข้อมูลมากเกินไปจนเบียดอัด การวางเนื้อหาเป็นสัดส่วนและสวยงาม
- **ตั้งชื่อของช่องกรอกข้อมูลอย่างเหมาะสม**
  เช่นเราใช้คำว่า "ชื่อ" จะสื่อความหมายได้ดีกว่าคำว่า "นาม" การใช้คำว่า "sex" สามารถเข้าใจได้ง่ายกว่าคำว่า "gender" เป็นต้น
- **การใช้คำและคำย่อคงที่**
  เช่น "ว/ด/ป" แทน "วัน/เดือน/ปี" ไม่เปลี่ยนเป็น "ว/ด/พ.ศ." หรือ "วัน/เดือน/พ.ศ." ในหน้าอื่นๆ ให้เลือกใช้อย่างใดอย่างหนึ่งและใช้ให้คงที่
- **การใช้สีที่คงที่**
  ทั้งสีโดยภาพรวมของหน้าจอและสีขององค์ประกอบย่อยในหน้าจอ เช่นข้อมูลปกติมีสีดำ ข้อมูลที่กรอกแล้วไม่ตรงตามเงื่อนไขเป็นสีแดง และเป็นอย่างนี้ในทุกๆ หน้า หรือช่องกรอกของมูลที่ผู้ใช้คนนั้นๆ แก้ไขข้อมูลได้มีพื้นหลังสีขาว ในขณะที่ช่องที่ไม่สามารถแก้ไขข้อมูลได้มีพื้นหลังเป็นสีเทา เป็นต้น
- **ช่องกรอกข้อมูลมีเนื้อที่เพียงพอที่จะแสดงข้อมูลที่ผู้ใช้กรอก**
  ช่องกรอกข้อมูลที่ต้องการเนื้อที่มาก ควรมีความยาวหรือขนาดใหญ่กว่าช่องอื่นๆ เช่นช่องกรอกที่อยู่ควรมีพื้นที่มากกว่าช่องกรอกชื่อ เพื่อแสดงข้อมูลที่ผู้ใช้กรอกทั้งหมด เป็นการป้องกันความผิดพลาดในการกรอกข้อมูลซ้ำซ้อน กรอกข้อมูลเกิน หรือไม่ครบถ้วน
- **การเลื่อนของตัวชี้มีความสะดวกและเป็นลำดับ**
  หลายครั้งที่ผู้ใช้มักจะเลื่อนตัวชี้เพื่อกรอกข้อมูลในช่องกรอกถัดไปด้วยแป้นพิมพ์ โดยใช้ปุ่ม Tab หรือ Enter โดยเฉพาะการกรอกข้อมูลจำนวนมากที่ไม่สะดวกในการละมือไปควบคุมตัวชี้ด้วยเมาส์ ระบบที่

ออกแบบควรรองรับการใช้งานในลักษณะดังกล่าว และลำดับของการเลื่อนตัวชี้ไปยังช่องกรอกถัดไปควรเรียงลำดับตามข้อมูลที่จะกรอกก่อน-หลัง

- **Error correction for individual characters and entire fields**
  การออกแบบระบบต้องอนุญาตให้ผู้ใช้แก้ไขข้อมูลทั้งแบบแก้ไขส่วนใดส่วนหนึ่งของช่องกรอกได้โดยไม่ต้องกรอกข้อมูลทั้งช่องใหม่ หรือแก้ไขทีเดียวทั้งช่องกรอกได้ แล้วแต่กรณีและความประสงค์ของผู้ใช้
- **แสดงข้อความเตือนที่สื่อความหมายในกรณีที่มีการกรอกข้อมูลไม่ถูกต้อง**
  เช่น ในกรณีที่เกรดของนักศึกษามีเพียง A, B, C, D และ F เท่านั้น หากผู้ใช้กรอกข้อมูลผิดเป็น B+ ระบบควรแสดงข้อความเตือนว่า "คุณกรอกเกรดไม่ถูกต้อง เกรดที่สามารถกำหนดได้คือ A, B, C, D และ F" แทนที่จะแสดงข้อความเพียง "คุณกรอกข้อมูลไม่ถูกต้อง" ผู้ใช้จะไม่ทราบว่าข้อมูลช่องใดที่กรอกผิดและผิดเพราะอะไรจะแก้ไขได้อย่างไร ไม่ทำให้เสียเวลาในการหาที่ผิดและวิธีการแก้ไข
- **ระบุช่องกรอกข้อมูลที่ไม่จำเป็นต้องกรอกให้ชัดเจน**
  ในกรณีที่ช่องกรอกข้อมูลส่วนใหญ่ไม่จำเป็นต้องกรอก เราสามารถเปลี่ยนมาเป็นการระบุช่องกรอกที่จำเป็นต้องกรอกให้ชัดเจนแทน เช่น ชื่อ – นามสกุลเป็นช่องที่ต้องกรอก โดยใช้เครื่องหมาย "*" และอธิบายในบริเวณที่เห็นได้ง่ายว่าช่องที่กำกับด้วยเครื่องหมาย "*" เป็นช่องที่เว้นว่างไม่ได้ ช่องเงินเดือนอาจจะไม่จำเป็นต้องกรอก จึงไม่ต้องใส่เครื่องหมาย "*" กำกับไว้
- **คำอธิบายข้อมูลในแต่ละช่องกรอก**
  บางครั้งชื่อที่ใช้กำกับช่องกรอกข้อมูลไม่สามารถสื่อความหมายได้เช่น "ยืนยันรหัสผ่าน" หน้าจออาจแสดงคำอธิบายท้ายช่องกรอกหรืออนุญาตให้ผู้ใช้เลือกแสดงคำอธิบายว่าเป็นช่องกรอกข้อมูลที่จะต้องกรอกเหมือนกับการกรอกรหัสผ่านเพื่อยืนยันว่ากรอกรหัสผ่านถูกต้องในกรณีที่ช่องกรอกปิดบังตัวอักษรที่ผู้ใช้พิมพ์
- **มีวิธีการทำให้ผู้ใช้ทราบถึงการเสร็จสิ้นของกิจกรรม**
  ไม่ว่าธุรกรรมนั้นสำเร็จหรือไม่สำเร็จ ระบบควรแจ้งแก่ผู้ใช้ รวมถึงหน้าจอลักษณะที่ต้องกรอกเป็นขั้นตอนต่อเนื่องควรบอกจุดสิ้นสุด เช่น หลังจากผู้ใช้กรอกข้อมูลแล้วเสร็จ และทำการบันทึกข้อมูลโดยกดปุ่ม "บันทึก" ระบบควรแสดงข้อความว่าบันทึกข้อมูลแล้วที่แถบสถานะด้านล่างหน้าจอ หรือด้วยกล่องข้อความที่เปิดใหม่ หรือด้วยบริเวณอื่นๆ ของหน้าจอ

2.10 การสร้างระบบต้นแบบ

การสร้างระบบต้นแบบที่สามารถใช้งานได้ โดยทั่วไปต้นแบบจะไม่ครอบคลุมการทำงานทั้งหมดแต่ระบบดังกล่าวจะนำมาใช้เพื่อสื่อสารกับผู้ใช้งานว่าระบบที่จะพัฒนาขึ้นนั้นได้รับการออกแบบอย่างถูกทิศทาง ขั้นตอนนี้เป็นขั้นตอนที่ไม่จำเป็นต้องปฏิบัติ

ขั้นตอนนี้เป็นขั้นตอนที่ไม่จำเป็นต้องปฏิบัติก็ได้ แต่มีข้อดีคือสามารถทำให้เรายืนยันกับผู้ใช้ได้ว่าระบบที่จะพัฒนาขึ้นนั้นมีลักษณะเป็นไปตามที่ผู้ใช้คาดหวัง ทั้งในเรื่องของส่วนติดต่อกับผู้ใช้ที่ผู้ใช้พึงพอใจ ขั้นตอนการใช้งานตลอดจนการรองรับความต้องการทางธุรกรรมที่ถูกต้อง

การพัฒนาระบบต้นแบบนั้นเป็นการพัฒนาระบบที่สามารถใช้งานได้แต่อาจจะไม่ครอบคลุมความต้องการข้อผู้ใช้ทั้งหมด บางครั้งเป็นการทำงานจากข้อมูลที่ได้จำลองขึ้นไม่ได้มีการปฏิบัติธุกรรมจริงๆ ก็ได้ เราใช้ระบบต้นแบบเพื่อทำความเข้าใจระหว่างผู้ใช้และผู้พัฒนา เปิดโอกาสให้ผู้ใช้เพิ่ม ลดความต้องการที่ขาดหรือเกิน ระบุถึงสิ่งที่ดีและไม่ดีในระบบต้นแบบ

การพัฒนาระบบต้นแบบนั้นใช้เวลาและความมานะไม่มาก รูปแบบของการพัฒนาระบบต้นแบบมี 2 วิธีได้แก่ แบบใช้แล้วทิ้ง (throw-away prototyping) กับแบบพัฒนาต่อเนื่อง (evolutionary prototyping) การพัฒนาต้นแบบแบบใช้แล้วทิ้งเป็นการพัฒนาต้นแบบแล้วนำมาให้ผู้ใช้ลองใช้เพื่อทราบความคิดเห็น จากนั้นทำการทิ้งต้นแบบดังกล่าวไปไม่ได้นำมาใช้พัฒนาต่อ ในขณะที่การพัฒนาต้นแบบแบบพัฒนาต่อเนื่องนั้น เราจะนำต้นแบบที่ผ่านการใช้งานแล้วมาพัฒนาต่อยอดไปเรื่อยๆ ซึ่งอาจกลายเป็นระบบที่ใช้งานจริงในที่สุด

2.11 การพัฒนาระบบ

การลงมือปฏิบัติในการพัฒนาฐานข้อมูลและโปรแกรมประยุกต์เพื่อเรียกใช้งานฐานข้อมูล และรองรับความต้องการของผู้ใช้

ทำการสร้างฐานข้อมูลโดยใช้เครื่องมือต่างๆ ในระบบจัดการฐานข้อมูล เราสามารถใช้ภาษาที่ใช้ในการนิยามข้อมูล (data definition language—DDL) เป็นคำสั่งที่ใช้ในการสร้างโครงสร้างของฐานข้อมูล หรือใช้เครื่องมืออำนวยความสะดวกที่เป็นกราฟฟิคในการสร้างโครงสร้างฐานข้อมูล สิ่งที่สร้างนั้นยังรวมถึงรีเลชัน ไฟล์ดัชนี ฯลฯ นอกจากตัวฐานข้อมูลแล้วขั้นตอนนี้เป็นขั้นตอนของการพัฒนาระบบสารสนเทศ/โปรแกรมประยุกต์ที่ใช้งานระบบฐานข้อมูล เครื่องมือที่ใช้พัฒนาระบบได้แก่เครื่องมือ 3GL/4GL (third and fourth generation language) ซึ่งเป็นภาษาคอมพิวเตอร์ระดับสูงและเครื่องมือกราฟฟิคในการพัฒนาซอฟต์แวร์ เช่นภาษา C, C#, C++, Java, Pascal, Visual Basic, PHP และ ASP เป็นต้น

การพัฒนาโปรแกรมประยุกต์เพื่อเรียกใช้ฐานข้อมูลคือการสร้างส่วนติดต่อกับผู้ใช้ ปุ่มคำสั่ง แบบฟอร์มกรอกข้อมูล ข้อความสื่อสารต่างๆ บนหน้าจอ รายงาน รวมถึงการสร้างโปรแกรมเพื่อรองรับธุรกรรมต่างๆ ซึ่งสามารถทำได้โดยการฝังคำสั่ง SQL ประเภทคำสั่งที่ใช้ในการจัดการข้อมูล (data manipulation language—DML) ไว้ในชุดคำสั่งของโปแกรมประยุกต์ ในปัจจุบันระบบจัดการฐานข้อมูลอาจมีเครื่องมือที่อำนวยความสะดวกในการสร้างฐานข้อมูลและโปรแกรมประยุกต์โดยไม่จำเป็นต้องใช้ DDL/DML หรือภาษาคอมพิวเตอร์ เช่นเครื่องมือสร้างฐานข้อมูลด้วยเครื่องมือแบบกราฟฟิค ตัวสร้างแบบฟอร์มอัตโนมัติ ตัวสร้างรายงานอัตโนมัติจากฐานข้อมูล หรือแม้แต่ตัวสร้างโปรแกรมประยุกต์ เป็นต้น

การพัฒนาระบบฐานข้อมูลในขั้นตอนนี้ยังรวมถึงการใช้ DLL ในการกำหนดนโยบายความมั่นคงปลอดภัยของข้อมูลในฐานข้อมูลด้วย SQL ซึ่งเป็นทั้ง DDL และ DML ได้อธิบายไว้อย่างละเอียดในบทที่ 7

2.12 การแปลงข้อมูลและการโหลดข้อมูล

การโอนย้ายข้อมูลจากระบบเก่า (ถ้ามี) เข้าสู่ระบบฐานข้อมูลใหม่ร่วมถึงการแปลงข้อมูลหรือโปรแกรมประยุกต์เดิมให้เข้ากับระบบฐานข้อมูลที่พัฒนาขึ้น

เป็นขั้นตอนที่ต้องดำเนินการในกรณีที่มีระบบฐานข้อมูลเดิมใช้งานอยู่เท่านั้น ระบบจัดการฐานข้อมูลมักจะมาพร้อมกับเครื่องมือที่ใช้ในการแปลงข้อมูลจากระบบจัดการฐานข้อมูลของผู้ผลิตรายอื่นเข้ามาบรรจุในระบบจัดการฐานข้อมูลของตน เพียงระบุข้อมูลต้นทางให้สัมพันธ์กับข้อมูลปลายทาง เครื่องมือแปลงข้อมูลจะโอนย้ายข้อมูล แปลงรูปแบบข้อมูลให้สามารถมาบรรจุในระบบจัดการฐานข้อมูลใหม่ได้ทันที ในกรณีที่มีระบบสารสนเทศ/โปรแกรมประยุกต์เดิมอยู่และจะต้องใช้งานต่อเนื่อง เราจะต้องทำการปรับแต่งเพื่อให้ระบบเดิมสามารถใช้ระบบฐานข้อมูลใหม่ได้ ทั้งนี้การย้ายการใช้งานจากระบบเก่าสู่ระบบใหม่จะต้องได้รับการวางแผนมาอย่างดี และเป็นสิ่งที่จะต้องคำนึงถึงตั้งแต่ต้นของวงจรการพัฒนาระบบฐานข้อมูล เพื่อให้การโอนย้ายการใช้งานเป็นไปอย่างราบรื่น

2.13 การทดสอบ

ทดสอบระบบฐานข้อมูล โปรแกรมประยุกต์ที่ได้พัฒนาขึ้น เพื่อกำจัดข้อบกพร่องและยืนยันว่าระบบตรงตามความต้องการของผู้ใช้ทุกข้อ

ผู้พัฒนาระบบฐานข้อมูลควรทำการทดสอบระบบอย่างละเอียดถี่ถ้วน โดยการสร้างข้อมูลเพื่อทดสอบ และจำลองสถานการณ์ ตลอดจนใช้อุปกรณ์ที่เหมือนจริงในการใช้งาน จากนั้นลองใช้งานระบบให้ครบทุกกรณีของการใช้งานจริง การทดสอบระบบมีวัตถุประสงค์เพื่อค้นหาข้อผิดพลาดของการพัฒนาระบบฐานข้อมูล ได้แก่ความผิดพลาดของโปรแกรมประยุกต์ และโครงสร้างของฐานข้อมูล พร้อมทั้งเป็นการยืนยันว่าระบบรองรับความต้องการของผู้ใช้ทุกข้อ รวมถึงมีประสิทธิภาพตามเกณฑ์ที่ได้วางไว้ การทดสอบควรทดสอบการสำรองข้อมูล การจำลองสถานการณ์ของการล้มเหลวของระบบและการกู้คืนระบบให้สำเร็จด้วย ผู้ใช้งานควรมีส่วนร่วมในการทดสอบด้วย เพื่อทดสอบความสามารถในการใช้งานได้ของระบบ ความยากง่ายของส่วนติดต่อกับผู้ใช้

2.14 การใช้งานและบำรุงรักษาระบบ

การใช้งานจริงรวมถึงการดูแลบำรุงรักษาระบบหลังการติดตั้ง มักเป็นหน้าที่ของผู้บริหารฐานข้อมูลหรือ DBA

หลังจากระบบผ่านการทดสอบเป็นที่เรียบร้อยแล้ว ผู้ใช้สามารถใช้งานระบบฐานข้อมูลในการปฏิบัติงานประจำวันได้ ทั้งนี้หากมีระบบที่ใช้อยู่แล้วควรใช้ระบบทั้งสองคู่ขนานกันเป็นระยะเวลาหนึ่ง เนื่องจากการทำงานของระบบใหม่อาจติดขัดหรือล้มเหลวได้ เมื่อระบบใหม่ทำงานได้อย่างไม่มีข้อบกพร่องแล้วจึงใช้งานเพียงระบบเดียว

ระบบฐานข้อมูลที่ได้ใช้งานจริงแล้ว จะต้องได้รับการดูแลให้ใช้งานได้เป็นปกติอยู่เสมอซึ่งเป็นหน้าที่ของผู้บริหารฐานข้อมูลหรือ DBA ทำการตรวจสอบความเป็นปกติของการใช้งานฐานข้อมูลอย่างสม่ำเสมอ สำรองข้อมูลและสามารถทำให้ระบบคืนสู่สภาพปกติได้โดยข้อมูลมีความถูกต้องในกรณีระบบเกิดการติดขัด หยุดชะงัก ล้มเหลว

มีการประเมินผลและติดตามประสิทธิการใช้งานของระบบ ปรับแต่งในการใช้งานฐานข้อมูลถูกต้อง และรวดเร็วขึ้น เช่นการออกแบบการสร้างดรรชนีสำหรับตารางใดๆ DBA จะตรวจสอบการใช้งานระบบฐานข้อมูลและวิเคราะห์ว่าข้อมูลกลุ่มหรือตารางใดที่มีการใช้งานมาก หรือน้อย ควรเพิ่มหรือลดดรรชนีสำหรับตารางใด หรือการแยกข้อมูลชุดเก่าและไม่ได้ใช้ออกจากข้อมูลที่ใช้ในปัจจุบันเพื่อเป็นต้น

ข้อสำคัญอีกประการหนึ่งคือ DBA ทำงานร่วมกับผู้ใช้เพื่อรับฟังข้อเสนอแนะ และรับฟังความต้องการของผู้ใช้ใหม่ๆ เพื่อประสานกับนักวิเคราะห์ระบบและโปรแกรมเมอร์ในการนำข้อเสนอแนะ และปรับปรุงระบบให้สอดคล้องกับลักษณะการใช้งานฐานข้อมูลเมื่อใช้งานจริงและความต้องการใหม่ๆ ของผู้ใช้ระบบ ซึ่งเป็นย้อนกลับไปสู่ขั้นตอนเริ่มต้นของวงจรการพัฒนาระบบฐานข้อมูลนั่นเอง

2.15 CASE Tools

Computer-Aided Software Engineering (CASE) tools คือเครื่องมือทางคอมพิวเตอร์ที่ช่วยในกระบวนการวิศวกรรมซอฟต์แวร์ ซึ่งซอฟต์แวร์ที่ออกแบบมาเพื่อช่วยอำนวยความสะดวกให้การวิเคราะห์ ออกแบบและพัฒนาฐานข้อมูล (ซึ่งเป็นส่วนหนึ่งของกระบวนการวิศวกรรมซอฟต์แวร์) ง่ายและมีประสิทธิภาพมากขึ้น

วงจรการออกแบบระบบฐานข้อมูลและวงจรการพัฒนาซอฟต์แวร์เป็นสิ่งที่ทำควบคู่กันไป การพัฒนาซอฟต์แวร์อย่างมีประสิทธิภาพนั้นเราอาศัยกระบวนการทางวิศวกรรมในการบริหารจัดการซึ่งเป็นศาสตร์ที่เรียกว่าวิศวกรรมซอฟต์แวร์ ในกระบวนการของวิศวกรรมซอฟต์แวร์ เราสามารถใช้เครื่องมือต่างๆ ในการอำนวยความสะดวกกิจกรรมที่ทำต่างๆ ซึ่งส่วนใหญ่เป็นโปรแกรมคอมพิวเตอร์

งานที่สามารถนำ CASE tools มาช่วยได้แก่

- การจัดเก็บคำอธิบายข้อมูลในระบบ ซึ่งเราเรียกว่าพจนานุกรมข้อมูล
- เครื่องมือที่นำมาช่วยในการวิเคราะห์ข้อมูล
- เครื่องมือที่ช่วยในการออกแบบแบบจำลองข้อมูล ได้แก่แบบจำลองข้อมูลระดับแนวคิด แบบจำลองข้อมูลระดับตรรกะเป็นต้น เครื่องมือประเภทนี้คือเครื่องมือที่ช่วยในการวาดแผนภาพตามข้อกำหนดของแบบจำลอง

- เครื่องมือที่นำมาช่วยในการสร้างระบบต้นแบบ

ประโยชน์ของ CASE tools ได้แก่

- รองรับการทำงานมาตรฐานเช่นการวาดแผนภาพเพื่อสร้างแบบจำลองข้อมูลต่างๆ อย่างสะดวกรวดเร็ว
- ทำให้เกิดมาตรฐานในการทำงาน ทั้งนี้ CASE tool จะกระตุ้นให้ผู้ใช้เครื่องมือใช้งานตามมาตรฐานกำหนดเช่นมาตรฐานของแผนภาพหรือแบบจำลองข้อมูลใดๆ
- การทำงานบางอย่างให้เป็นอัตโนมัติเช่นสามารถแปลงแบบจำลองข้อมูลที่วาดขึ้นไปเป็นคำสั่ง SQL ได้โดยอัตโนมัติ
- การพัฒนาระบบมีเอกภาพ มีรูปแบบเดียวกันทั้งองค์กร ในกรณีที่ใช้ CASE tool ที่รองรับมาตรฐานเดียวกัน แบบจำลอง รวมถึงระบบต่างๆ จะมีเอกภาพและนำมาบูรณาการเชื่อมต่อกันได้

2.16 แบบฝึกหัดท้ายบท

ให้นักศึกษาใช้วัตถุประสงค์ของบทเป็นแบบฝึกหัดท้ายบท**

บทที่ 3 แบบจำลองเอนทิตีและความสัมพันธ์ (ER model)

วัตถุประสงค์

- สามารถอธิบายเหตุผลของการนำแบบจำลองเอนทิตีและความสัมพันธ์ (ER model) มาใช้ในการออกแบบฐานข้อมูลระดับแนวคิดได้
- สามารถอธิบายแนวคิดหลักเกี่ยวกับ ER model ได้
- สามารถออกแบบฐานข้อมูลระดับแนวคิดด้วย ER model โดยสร้างแผนภาพเอนทิตีและความสัมพันธ์ (ERD) ได้
- สามารถระบุแนวปฏิบัติในการใช้ ER model อย่างมีประสิทธิภาพ

คำสำคัญ: การออกแบบฐานข้อมูลระดับแนวคิด (conceptual database design); แบบจำลองเอนทิตีและความสัมพันธ์ (entity-relationship model/ER model); แผนภาพเอนทิตีและความสัมพันธ์ (entity-relationship diagram/ERD); เอนทิตี (entity); ลักษณะประจำ/แอทริบิวต์ (attribute); ความสัมพันธ์ (relationship); คีย์ (key); เงื่อนไขบังคับบูรณภาพ (integrity constraints); ความสัมพันธ์แบบหนึ่งต่อกลุ่ม และกลุ่มต่อกลุ่ม (one-to-many and many-to-may relationship); เงื่อนไขบังคับการมีส่วนร่วม (participation constraints); เอนทิตีอ่อนแอ (weak entity); ลำดับชั้นของคลาส (class hierarchies); การรวมกลุ่ม (aggregation)

3.1 บทนำ

จากที่ได้อธิบายถึงวงจรของการพัฒนาฐานข้อมูลในบทที่แล้ว การออกแบบฐานข้อมูลเป็นขั้นตอนที่สำคัญที่สุดขั้นตอนหนึ่งเพื่อให้ได้มาซึ่งระบบฐานข้อมูล หลังจากที่ได้ทำการรวมรวมและวิเคราะห์ความต้องการของผู้ใช้และข้อกำหนดของระบบแล้ว ผลลัพธจากขั้นตอนเหล่านั้นนำมาใช้ในการออกแบบฐานข้อมูล ซึ่งการออกแบบฐานข้อมูลแบ่งเป็น 3 ขั้นตอนใหญ่ๆ ได้แก่ การออกแบบฐานข้อมูลระดับแนวคิด การออกแบบฐานข้อมูลระดับตรรกะและการออกแบบฐานข้อมูลระดับกายภาพ สำหรับในบทนี้เราจะได้กล่าวถึงแบบจำลองเอนทิตีและความสัมพันธ์ ซึ่งเป็นเครื่องมือในการออกแบบฐานข้อมูลระดับแนวคิด

แบบจำลองเอนทิตีและความสัมพันธ์ (ER model) เป็นเครื่องมือที่สามารถแทนเอนทิตีซึ่งเป็นวัตถุหรือนามธรรมใดๆ และความสัมพันธ์ระหว่างเอนทิตีในโลกแห่งความเป็นจริง แต่เป็นเอนทิตีและความสัมพันธ์ของสิ่งที่เราสนใจในการใช้งาน จัดการและจัดเก็บข้อมูลของเอนทิตีและความสัมพันธ์เหล่านั้นในระบบฐานข้อมูล การวิเคราะห์เอนทิตีและความสัมพันธ์เหล่านี้โดยอาศัย ER model จะทำให้รูปแบบข้อมูลที่ใช้ในองค์กรสามารถจดบันทึกได้ ในรูปแบบของแผนภาพเอนทิตีและความสัมพันธ์ (ER diagram/ERD) มีความเป็นทางการ มีคำอธิบายที่ชัดเจน สามารถนำมาใช้สื่อสารเพื่อความเข้าใจที่ตรงกัน และนำไปสร้างฐานข้อมูลต่อไป

เอนทิตีและลักษณะเฉพาะของเอนทิตี ตลอดจนความสัมพันธ์เป็นพื้นฐานของ ER model ได้อธิบายไว้ในหัวข้อที่ 3.2 และ 3.3 หัวข้อที่ 3.4 แนะนำคุณสมบัติอื่นๆ ที่ ERD รองรับ การออกแบบ ERD ให้ถูกต้องจำเป็นต้องอาศัยประสบการณ์และทักษะ การเลือกใช้รูปแบบของ ERD ในบางครั้งไม่ถูกต้อง หัวข้อที่ 3.5 อธิบายถึงข้อควรคำนึงถึงในการออกแบบด้วย ERD เพื่อที่จะสามารถเลือกใช้รูปแบบของ ERD ที่ถูกต้องเหมาะสม ตัวอย่างการออกแบบ ERD ด้วยกรณีศึกษาแสดงไว้ในหัวข้อที่ 3.6 โดยการใช้กรณีศึกษาร้าน 7-Elephant

ERD นั้นสามารถแทนรูปแบบโครงสร้างข้อมูลที่ใช้ในองค์กรเพื่อนำไปใช้ประโยชน์อื่นๆ นอกเหนือจากการออกแบบฐานข้อมูลได้ แบบจำลองที่ได้มาจะไม่ยึดติดกับแบบจำลองข้อมูลสำหรับฐานข้อมูลใดๆ ไม่ยึดติดกับระบบจัดการฐานข้อมูลของเจ้าใดเจ้าหนึ่ง โปรแกรมประยุกต์ ซอฟต์แวร์ หรือฮาร์ดแวร์ใดๆ ERD นั้นมีหลายรูปแบบ และไม่ได้มีการกำหนดมาตรฐานที่ตายตัว โดยหลักแล้ว ERD จะมีรูปแบบคล้ายคลึงกัน กล่าวคือมีเอนทิตี แอทริบิวต์ และความสัมพันธ์ ตลอดจนคุณสมบัติอื่นๆ ในการกำหนดเงื่อนไขบังคับต่างๆ แต่รูปแบบการนำมาแสดงเป็นภาพหรือ diagram มีความแตกต่างกัน เราจะได้ศึกษารูปแบบของ ERD พร้อมทั้งคุณสมบัติอื่นๆ จากรูปแบบของ Elmarsi และ Navathe** ซึ่งเป็น ERD ที่พัฒนามาจาก ERD ของ Chen** ผู้เสนอ ERD เป็นคนแรก ERD รูปแบบที่เลือกนำมาใช้นี้มีจุดเด่นที่สามารถเข้าใจได้ง่าย หากเข้าใจพื้นฐานของ ERD แล้วสามารถเทียบเคียงสัญลักษณ์กับมาตรฐาน ERD อื่นๆ ได้โดยง่าย สำหรับแบบจำลองเพื่อสร้างสคีมาระดับแนวคิดนี้นอกจาก ER model แล้วยังมีการใช้ UML ซึ่งเป็นอีกแบบจำลองหนึ่งที่ได้รับความนิยมแพร่หลายในปัจจุบัน เหมาะกับระบบจัดการฐานข้อมูลเชิงวัตถุ

3.2 เอนทิตี (Entity) และแอททริบิวต์ (Attribute)

เอนทิตี

เอนทิตี คือ วัตถุพื้นฐานที่ ER model ใช้แทนสิ่งต่างๆ ในโลกแห่งความเป็นจริง ซึ่งอาจอยู่ในรูปวัตถุที่เป็นรูปธรรมจับต้องได้ หรือในรูปของนามธรรม ซึ่งข้อมูลของสิ่งต่างๆ เหล่านี้อยู่ในขอบเขตที่เราจะนำมาใช้ประโยชน์

เอนทิตีที่เป็นรูปธรรม เช่น เอนทิตีในมหาวิทยาลัย ได้แก่ นักศึกษา อาจารย์ อาคารเรียน ห้องเรียน ฯลฯ เอนทีตีของบริษัท ได้แก่ พนักงาน สินค้า ฯลฯ เอนทิตีของอู่ซ่อมรถยนต์ ได้แก่ ช่าง รถยนต์ อะไหล่ ฯลฯ เป็นต้น

เอนทิตีที่เป็นนามธรรม เช่น มหาวิทยาลัย (ไม่ได้หมายถึงที่ตั้งแต่หมายถึงภาพรวมของมหาวิทยาลัย) รายวิชาที่เปิดสอน งาน บริษัท เป็นต้น

ความหมายของเอนทิตีโดยแท้จริงนั้นหมายถึงสิ่งใดสิ่งหนึ่งเพียงสิ่งเดียว เช่นนักศึกษาคนใดคนหนึ่ง ซึ่งนักศึกษาหลายๆ คนจัดเป็นเอนทิตีกลุ่มเดียวกัน เราเรียกว่า entity types อย่างไรก็ตามการกล่าวถึงเอนทิตีในที่นี้เป็นการกล่าวถึงกลุ่มของเอนทิตี

แอทริบิวต์

แอทริบิวต์ (ลักษณะประจำ) คือ คุณสมบัติต่างๆ ที่อธิบายเอนทิตี

ตัวอย่างของแอทริบิวต์ เช่น แอทริบิวต์ของเอนทิตีนักศึกษา ได้แก่ รหัสนักศึกษา ชื่อ นามสกุล หลักสูตรที่สังกัด เกรดเฉลี่ย ฯลฯ แอทริบิวต์ของเอนทิตีพนักงานในบริษัท ได้แก่ ชื่อ นามสกุล อายุ ที่อยู่ เงินเดือน ฯลฯ และตัวอย่างแอทริบิวต์ของเอนทิตีรถยนต์ ได้แก่ เลขทะเบียน ยี่ห้อ รุ่น สี ปีที่ผลิต ราคา ฯลฯ เป็นต้น เอนทิตีในโลกแห่งความเป็นจริงนั้นมีคุณสมบัติปลีกย่อยมากมาย เราสามารถระบุเฉพาะแอทริบิวต์ที่เกี่ยวข้องกับกิจกรรมที่เราดำเนินการและแอทริวบิวต์ที่เราประสงค์จะใช้ประโยชน์เท่านั้น เช่น เราไม่ได้จัดเก็บข้อมูลแอทริบิวต์ส่วนสูง หรือน้ำหนักของเอนทิตีนักศึกษาในกิจกรรมที่เกี่ยวกับการจัดการเรียนการสอน เป็นต้น นอกจากนี้แล้วแอทริบิวต์ยังมีคุณลักษณะอื่นๆ อีก ดังนี้

ขอบเขตของแอทริบิวต์ (Attribute Domain)

ขอบเขตของแทริบิวต์ (โดเมน) คือ ค่าของแอทริบิวต์ที่เป็นไปได้หรืออนุญาตให้เป็น

โดเมนของแอทริบิวต์สามารถเป็นได้ในลักษณะ

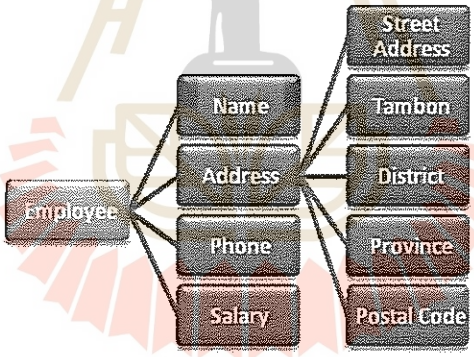
- กำหนดชนิดของข้อมูล เช่น ชื่อของพนักงานเป็นข้อมูลชนิดตัวอักษร อายุจะต้องเป็นตัวเลขจำนวนเต็มเท่านั้น

- กำหนดช่วงของข้อมูล เช่น เงินเดือนของพนักงานจะมีค่าติดลบไม่ได้ อายุของพนักงานสัญญาจ้างจะต้องอยู่ระหว่าง 15-60 ปีเท่านั้น เป็นต้น
- กำหนดรายการของค่าที่เป็นไปได้ เช่น เกรดของนักศึกษาจะต้องมีค่า A, B, C, D หรือ F อย่างหนึ่งอย่างใดเท่านั้น เป็นต้น
- กำหนดขนาดของข้อมูล เช่น หมายเลขโทรศัพท์ของพนักงานจะต้องมีความยาวไม่เกิน 10 ตัวเลข หมายเลขบัญชีธนาคารจะต้องมีความยาว 10 ตัวเลขเท่านั้น เป็นต้น

แอทริบิวต์ประกอบ (Composite Attribute)

แอทริบิวต์ประกอบ คือ แอทริบิวต์ที่สามารถแบ่งส่วนย่อยได้อีก

แอทริบิวต์โดยปกติทั่วไปจะมีค่าเพียงค่าเดียว เราเรียกว่าแอทริบิวต์ค่าเดียว (single value attribute) ซึ่งได้แก่ แอทริบิวต์ส่วนใหญ่ที่ได้ยกตัวอย่างมาแกล้ว แต่สำหรับแอทริบิวต์ประกอบ เป็นแอทริบิวต์ที่ประกอบด้วยข้อมูลย่อย เช่น ที่อยู่ ประกอบด้วยข้อมูลย่อย ได้แก่ เลขที่ ถนน ตำบล/แขวง อำเภอ/เขต จังหวัด และรหัสไปรษณีย์ ทะเบียนรถยนต์ ประกอบด้วยข้อมูลย่อยได้แก่ เลขทะเบียน และจังหวัด



รูปที่ 1** แสดงตัวอย่างเอนทิตีพนักงาน (Employee) ซึ่งมีแอทริบิวต์ Address เป็นแอทริบิวต์ประกอบ

คีย์ (Key)

คีย์ คือ แอทริบิวต์ที่ค่าของแอทริบิวต์นั้นสามารถระบุเฉพาะเจาะจงเอนทิตีหนึ่ง เอนทิตีใดในกลุ่มของเอนทิตีได้

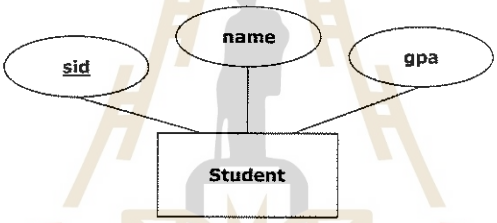
เช่น เอนทิตีนักศึกษาประกอบด้วย รหัสนักศึกษา ชื่อ นามสกุล หลักสูตรที่สังกัด และเกรดเฉลี่ย ในกลุ่มของเอนทิตีนักศึกษานี้ประกอบด้วยเอนทิตีนักศึกษาหลายๆ คน การระบุเอนทิตีให้เฉพาะเจาะจง เราต้องกำหนดให้แอทริบิวต์รหัสนักศึกษาเป็นคีย์ เพราะเมื่อเราระบุรหัสนักศึกษาใดๆ ข้อมูลของนักศึกษาคนนั้นจะถูกระบุได้ทั้งหมดตรงตาม

เจตนา เราไม่สามารถใช้แอทริบิวต์อื่นได้ เช่น แอทริบิวต์ชื่อ เนื่องจากนักศึกษาอาจมีชื่อซ้ำกัน เราจึงไม่สามารถระบุถึงเอนทิตีให้เฉพาะเจาะจงได้ คีย์ไม่จำเป็นต้องสร้างมาจากแอทริบิวต์เดียวเท่านั้น แอทริบิวต์หลายๆ แอทริบิวต์สามารถประกอบกันเป็นคีย์ได้ สำหรับรายละเอียดของคีย์นั้นได้อธิบายเพิ่มเติมในบทที่ 4 แบบจำลองเชิงสัมพันธ์

เราสามารถแสดงเอนทิตีและแอทริบิวต์ให้อยู่ในรูปแบบที่จดบันทึกและเห็นภาพได้โดยใช้ ERD เอนทีตีนั้นแทนด้วยรูปสี่เหลี่ยมกำกับด้วยชื่อของเอนทิตี ส่วนแอทริบิวต์จะแทนด้วยวงรี แอทริบวต์ของเอนทิตีใดๆ จะเชื่อมกับเอนทีตีนั้นๆ ด้วยเส้นเชื่อมต่อ เราสามารถกำกับโดเมนของแอทริบิวต์ในแผนภาพด้วยแต่ไม่นิยมเนื่องจากจะเปลืองเนื้อที่ ข้อมูลเกี่ยวกับโดเมนสามารถบันทึกแยกไว้ในพจนานุกรมข้อมูล แอทริบิวต์ใดที่เป็นคีย์สามารถกำกับได้โดยการขีดเส้นใต้ชื่อของแอทริบิวต์นั้น

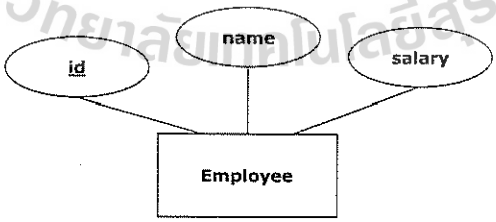
ตัวอย่าง

ตัวอย่างเอนทิตีในมหาวิทยาลัย ERD ของเอนทิตีนักศึกษา (Student) ประกอบด้วยแอทริบิวต์ รหัสนักศึกษา (sid) ชื่อ-นามสกุล (name) เกรดเฉลี่ย (gpa) ดังนี้



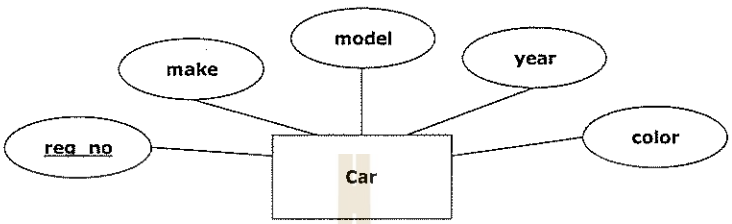
ตัวอย่าง

ตัวอย่างเอนทิตีในบริษัททั่วไป ERD ของเอนทิตีพนักงาน (Employee) ประกอบด้วยแอทริบิวต์ รหัสประจำตัวประชาชน (id) ชื่อ-นามสกุล (name) เงินเดือน (salary) ดังนี้



ตัวอย่าง

ตัวอย่างเอนทิตีของอู่ซ่อมรถ ERD ของเอนทิตีรถยนต์ (Car) ประกอบด้วยแอทริบิวต์ เลขทะเบียนรถยนต์ (reg_no) ยี่ห้อ (make) รุ่น (model) สี (color) ปีที่ผลิต (year) ดังนี้



3.3 ความสัมพันธ์ (Relationship)

ความสัมพันธ์

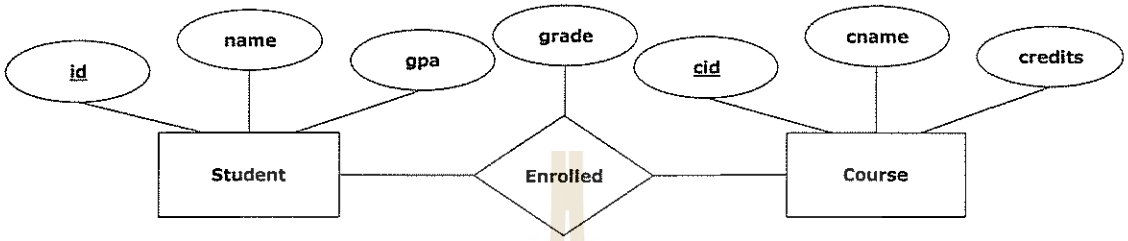
ความสัมพันธ์ คือ ความเกี่ยวข้องกันของเอนทิตี

เช่น สมชายซึ่งเป็นนักศึกษาลงทะเบียนเรียนรายวิชา 204203 การออกแบบและพัฒนาซอฟต์แวร์ และรายวิชา 204204 การออกแบบและพัฒนาฐานข้อมูล สมหญิงลงทะเบียนเรียนรายวิชา 204204 การออกแบบและพัฒนาฐานข้อมูล เป็นตัวอย่างความสัมพันธ์ระหว่างเอนทิตีนักศึกษากับรายวิชาที่เปิดสอน เราอาจระบุความสัมพันธ์นี้ได้ว่าเป็นการลงทะเบียนเรียน การลงทะเบียนเรียนของนักศึกษาแต่ละคนกับรายวิชาแต่ละรายวิชาเป็น 1 ความสัมพันธ์ ความสัมพันธ์ที่เหมือนๆ กันนี้รวมกันเป็นชุดความสัมพันธ์ (relationship types) ซึ่งในที่นี้เราจะเรียก "ชุดความสัมพันธ์" โดยสั้นๆ ว่า "ความสัมพันธ์" เท่านั้น เพื่อให้กระชับ ในความสัมพันธ์แต่ละความสัมพันธ์จะสามารถระบุคู่หรือชุดของเอนทิตีที่มีส่วนร่วมในความสัมพันธ์นั้น ดังนั้นคีย์ของเอนทิตีที่มีส่วนร่วมในความสัมพันธ์จะถูกบันทึกไว้ในความสัมพันธ์

เราสามารถระบุคุณสมบัติกำกับ หรือกำหนดข้อมูลเพื่ออธิบายความสัมพันธ์เพิ่มเติมได้ในลักษณะเดียวกันกับการอธิบายเอนทิตีด้วยแอทริบิวต์ เราเรียกแอทริบิวต์กำกับความสัมพันธ์ว่าแอทริบิวต์เชิงพรรณนา (descriptive attribute) การแทนความสัมพันธ์ด้วย ERD นั้นใช้สัญลักษณ์สี่เหลี่ยมขนมเปียกปูนเชื่อมระหว่างเอนทิตีที่มีความสัมพันธ์กันและเชื่อมกับแอทริบิวต์เชิงพรรณาลักษณะเดียวกับแอทริบิวต์ปกติ **

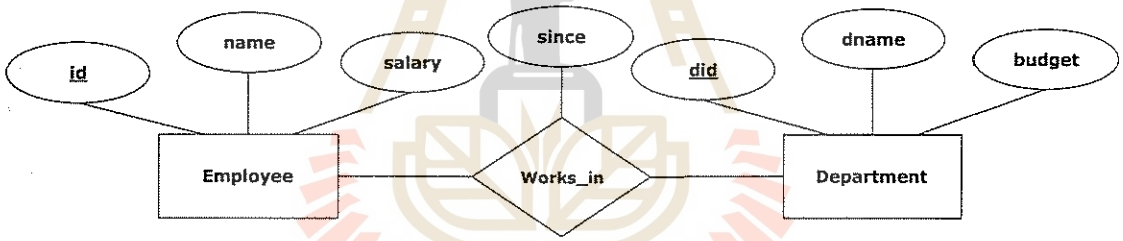
ตัวอย่าง

ตัวอย่างความสัมพันธ์ในมหาวิทยาลัย ความสัมพันธ์ของนักศึกษา (Student) **ลงทะเบียนเรียน (Enrolled)** รายวิชา (Course) ซึ่งความสัมพันธ์แต่ละความสัมพันธ์ในที่นี้นักศึกษาคนหนึ่งลงทะเบียนรายวิชาหนึ่ง ดังนี้



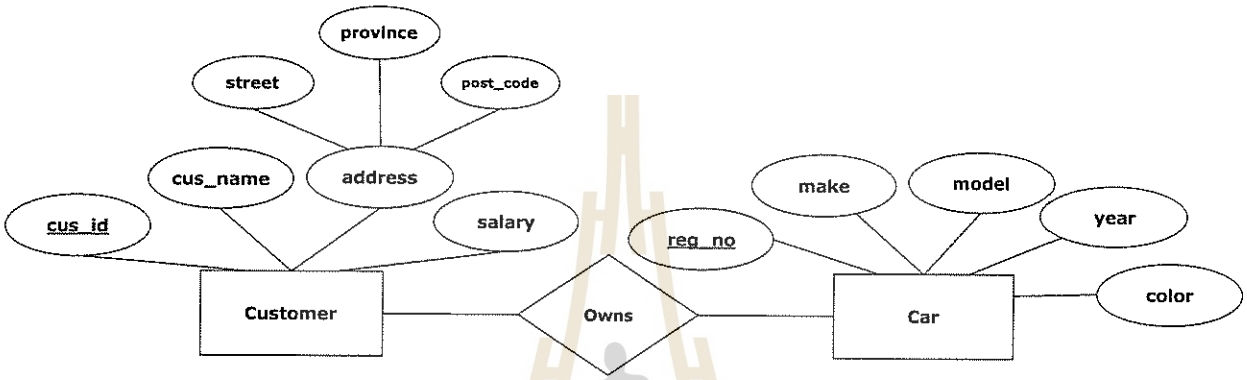
ตัวอย่าง

ตัวอย่างความสัมพันธ์ในบริษัททั่วไป ความสัมพันธ์ของพนักงาน (Employee) **ทำงานใน (Works_in)** แผนก (Department) ซึ่งความสัมพันธ์แต่ละความสัมพันธ์ในที่นี้คือพนักงานคนหนึ่งกับการทำงานให้แผนกๆ หนึ่ง ดังนี้



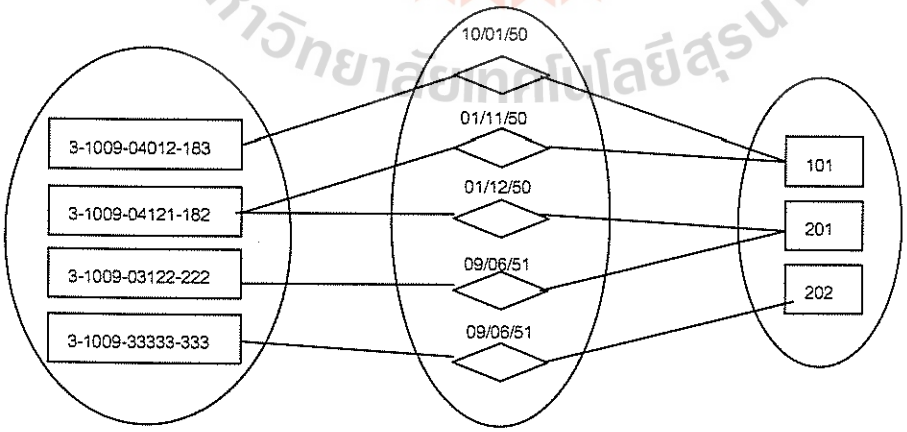
ตัวอย่าง

ตัวอย่างความสัมพันธ์ในอู่ซ่อมรถ (Garage) ความสัมพันธ์ของลูกค้า (Customer) **เป็นเจ้าของ (Owns)** รถยนต์ (Car) ซึ่งความสัมพันธ์แต่ละความสัมพันธ์ในที่นี้คือลูกค้าคนหนึ่งเป็นเจ้าของรถยนต์คันหนึ่ง ในตัวอย่างนี้ ที่อยู่ (address) เป็นตัวอย่างของแอทริบิวต์ประกอบและการแทนแอทริบิวต์ประกอบใน ER model ดังนี้



การแสดงตัวอย่างของข้อมูลจริงที่เป็นไปตามแบบจำลองข้อมูล จะอธิบายให้เห็นภาพชัดเจนขึ้นของการแทนข้อมูลในองค์กรด้วย ERD และสามารถใช้เป็นตัวอย่างในการศึกษาวิเคราะห์ถึงความจำเป็นที่เราอาจต้องกำหนดเงื่อนไขบังคับใดๆ สำหรับ ERD เพื่อรองรับให้ข้อมูลมีความถูกต้องตามการใช้งานจริง กรณีตัวอย่าง (instance) ของความสัมพันธ์ เป็นกรณีของข้อมูลที่เกิดขึ้นจริงของความสัมพันธ์ ณ ขณะใดขณะหนึ่ง ดังตัวอย่างรูปที่ 1** ที่แสดงกรณีตัวอย่างของความสัมพันธ์ "ทำงานใน" จากตัวอย่าง 1**

**



Employee Works_in Department

Total participation Many to Many Total participation

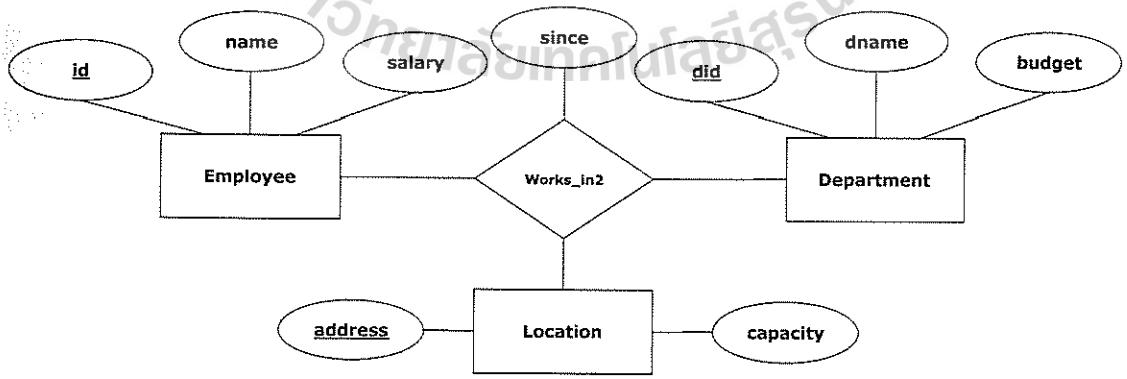
กรณีตัวอย่างของความสัมพันธ์พนักงาน "ทำงานใน" แผนกใดๆ อธิบายได้ว่าพนักงานที่มีรหัสประจำตัวประชาชน 3-1009-04012-183 ทำงานให้แผนกที่มีรหัสของหน่วยงาน 101 ตั้งแต่วันที่ 10/01/50 พนักงานที่มีรหัส 3-1009-04121-182 ทำงานให้แผนก 2 แผนก ได้แก่ทำงานให้แผนก 101 ตั้งแต่วันที่ 01/11/50 และแผนก 201 ตั้งแต่วันที่ 01/12/50 กรณีตัวอย่างของความสัมพันธ์ที่เหลือสามารถอธิบายได้ในทำนองเดียวกัน สำหรับ Total participation คือลักษณะการมีส่วนร่วมในความสัมพันธ์ และ Many-to-Many คือรูปแบบของความสัมพันธ์ ซึ่งอธิบายในหัวข้อ 3.4.1

ในบางกรณี ความสัมพันธ์ในโลกแห่งความเป็นจริงทำให้เกิด ความสัมพันธ์ที่มีรูปแบบนอกเหนือจากที่กล่าวมาแล้ว ดังนี้

ความสัมพันธ์แบบไตรภาค

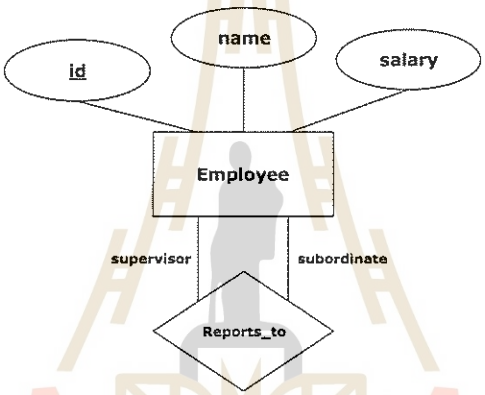
ความสัมพันธ์แบบไตรภาค คือ ความสัมพันธ์ๆ หนึ่งที่มีความเกี่ยวข้องกันของเอนทิตี 3 เอนทิตี

ความสัมพันธ์ระหว่างเอนทิตี 2 เอนทิตีเราสามารถเรียกชื่อเฉพาะได้ว่า**ความสัมพันธ์แบบทวิภาค (binary relationship)** อย่างไรก็ตามความสัมพันธ์ระหว่างเอนทิตีนั้นสามารถมีเอนทิตีที่เกี่ยวข้องกันได้มากกว่า 2 เอนทิตี สมมติว่าบริษัทฯ หนึ่งมีหลายแผนก แต่ละแผนกยังมีออฟฟิศหลายออฟฟิศมีที่ตั้งต่างๆ กัน ถ้าเราต้องการจัดเก็บข้อมูลโดยระบุว่าพนักงานแต่ละคนทำงานได้ในหลายแผนกและมีออฟฟิศที่ทำงานได้หลายออฟฟิศตามแผนกนั้นๆ ความสัมพันธ์ของเอนทิตีที่เกิดขึ้นจะเป็นความสัมพันธ์ของเอนทิตี 3 เอนทิตี เรียกว่า**ความสัมพันธ์แบบไตรภาค (ternary relationship)** ซึ่งได้แก่ความสัมพันธ์ของ พนักงาน (Employee) แผนก (Department) และที่ตั้งออฟฟิศ (Location) สามารถแสดงได้ด้วย ERD ในลักษณะคล้ายคลึงกับความสัมพันธ์แบบทวิภาค ดังนี้



ความสัมพันธ์ระหว่างเอนทิตียังไม่ได้จำกัดจำนวนของเอนทิตีที่เกี่ยวข้องกันเพียง 3 เอนทิตีเท่านั้น ความสัมพันธ์ยังสามารถเกิดจากเอนทิตีมากกว่า 3 เอนทิตี แล้วแต่กรณีของรูปแบบความสัมพันธ์นั้นๆ สำหรับความสัมพันธ์ที่เกิดจากเอนทิตีจำนวน 4 เอนทิตี เราเรียกว่า**ความสัมพันธ์แบบจตุภาค (quarterly relationship)**

ความสัมพันธ์ของเอนทิตีในบางครั้งสามารถที่จะขึ้นระหว่างเอนทิตีภายในชุดของเอนทิตีเดียวกันได้ เช่น พนักงานต่างก็อยู่ในชุดเอนทิตี Employee ทั้งสิ้น ความสัมพันธ์ ขึ้นตรงต่อ (Reports_to) เป็นตัวอย่างความสัมพันธ์ภายในกลุ่มเอนทิตี้ กล่าวคือโดยปกติพนักงานแต่ละคนจตะต้องมีผู้บังคับบัญชา ซึ่งผู้บังคับบัญชานั้นก็เป็นพนักงานเช่นกัน ความสัมพันธ์ Reports_to เป็นความสัมพันธ์ระหว่างเอนทิตีพนักงานในกลุ่มของเอนทิตีพนักงานเหมือนกัน โดยพนักงานคนหนึ่งเป็นผู้บังคับบัญชาของพนักงานอีกคนหนึ่ง เราสามารถแทนรูปแบบความสัมพันธ์ในลักษณะนี้ได้ตาม ERD ดังนี้



ซึ่งหลักการแทนความสัมพันธ์ด้วย ERD นี้เหมือนๆ กันกับการวาด ERD ของความสัมพันธ์ทั่วไป แต่การเชื่อมโยงความสัมพันธ์นั้นกลับไปยังเอนทิตีของตัวเอง อย่างไรก็ตามความสัมพันธ์ในลักษณนี้เอนทิตีในความสัมพันธ์อาจ**มีบทบาท (roles)** แตกต่างกัน เช่น บทบาท**ผู้บังคับบัญชา (supervisor)** และบทบาท**ผู้ใต้บังคับบัญชา (subordinate)** เราสามารถใช้ข้อความกำกับบทบาท (role indicator) เพื่อระบุบทบาทของเอนทิตีในความสัมพันธ์ได้ตามตัวอย่างที่แสดงไว้ ความสัมพันธ์ Reports_to นั้นจะเชื่อมโยงเอนทิตีที่สัมพันธ์กันด้วยแอทริบิวต์ที่แทนเอนทิตีทั้งสอง ได้แก่รหัสประจำตัวประชาชนของคู่พนักงานของผู้บังคับบัญชา (supervisor_id) และผู้ใต้บังคับบัญชา (subordinate_id)

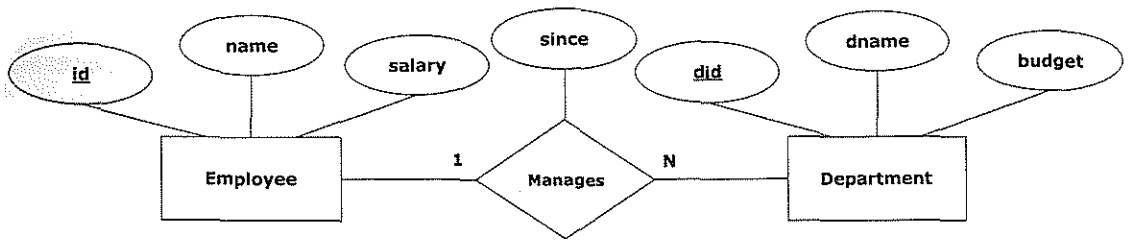
3.4 คุณสมบัติอื่นๆ ของ ER model

นอกจากการแทนเอนทิตีและความสัมพันธ์เป็นแบบจำลอง/แผนภาพได้แล้ว เราสามารถกำหนดคุณสมบัติเพื่อกำหนดข้อบังคับเพิ่มเติมเกี่ยวกับข้อมูลได้ โดยการเพิ่มสัญลักษณ์ใน ERD ความสามารถในการกำหนดข้อกำหนดของข้อมูลนี้ เป็นเหตุผลที่ทำให้ ERD ได้รับความนิยมอย่างแพร่หลาย คุณสมบัติที่สามารถระบุใน ERD ที่ได้รับความนิยมมีดังต่อไปนี้

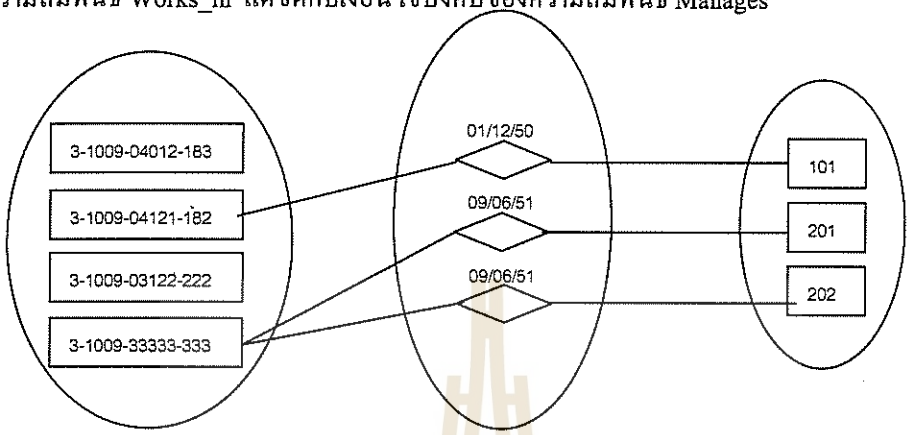
3.4.1 เงื่อนไขบังคับคีย์ (Key Constraints) หรือคาร์ดินัลลิตี (Cardinality)

พิจารณาความสัมพันธ์การทำงานในแผนก Works_in ในรูปที่ 1** พนักงานคนหนึ่งสามารถทำงานได้ในแผนกหลายแผนกพร้อมๆ กัน และแผนกๆ หนึ่งมีพนักงานได้หลายคน แสดงได้ดังกรณีตัวอย่างของความสัมพันธ์ Works_in ในรูป 1** พนักงานที่รหัสประจำตัวประชาชน 3-1009-04121-182 ทำงานในแผนกหมายเลข 101 ตั้งแต่วันที่ 01/11/50 และในแผนก 201 ตั้งแต่วันที่ 01/12/50 แผนก 201 มีพนักงาน 2 คน การที่เอนทิตีหนึ่งๆ จะมีความสัมพันธ์กับเอนทิตีอีกเอนทิตีหนึ่งเป็นจำนวนเท่าใดนั้นเราเรียกว่า **คาร์ดินัลลิตี (cardinality)** ของความสัมพันธ์ ความสัมพันธ์ที่พนักงานคนหนึ่งสามารถทำงานได้ในหลายๆ แผนก และในหนึ่งแผนกมีพนักงานได้หลายคนนั้นเราเรียกความสัมพันธ์แบบนี้ว่าเป็นความสัมพันธ์แบบกลุ่มต่อกลุ่ม **(manay-to-many)** หรือความสัมพันธ์มีคาร์ดินัลลิตีแบบ many-to-many

ลองพิจารณาดูความสัมพันธ์อีกรูปแบบหนึ่งซึ่งต่างออกไป ความสัมพันธ์ **จัดการ (Manages)** คือความสัมพันธ์ที่พนักงานคนหนึ่งซึ่งมีตำแหน่งเป็นผู้จัดการบริหารจัดการแผนกๆ หนึ่ง ในกรณีนี้ถ้าเรากำหนดว่าแผนกๆ หนึ่ง สามารถมีผู้จัดการได้เพียงคนเดียวเท่านั้น แต่ว่าพนักงานคนหนึ่งสามารถเป็นผู้จัดการของแผนกหลายๆ แผนก ข้อกำหนดที่อนุญาตให้แผนกๆ หนึ่งมีผู้จัดการเพียงคนเดียวเท่านั้นเป็นตัวอย่างของ**เงื่อนไขบังคับคีย์ (key constraints)** พิจารณาให้ลึกลงไปในโครงสร้างของแบบจำลองข้อมูลเชิงสัมพันธ์จะพบว่า เราอนุญาตให้เอนทิตีแผนกแต่ละแผนก อยู่ในความสัมพันธ์ Manages ได้เพียง 1 ครั้งเท่านั้น ในกรณีนี้รหัสแผนกแต่ละแผนกซึ่งเป็นคีย์ของเอนทิตีแผนกสามารถปรากฏในความสัมพันธ์ Manges ได้เพียง 1 ครั้งเท่านั้น การที่พนักงาน 1 คนสามารถบริหารจัดการแผนกได้หลายแผนก แต่แผนกๆ หนึ่งมีผู้จัดการได้เพียงคนเดียว เราเรียกลักษณะความสัมพันธ์ Manages ระหว่างเอนทิตีพนักงานและเอนทิตีแผนกว่าความสัมพันธ์แบบหนึ่งต่อกลุ่ม **(one-to-many)** ทั้งนี้ลำดับของการกล่าวถึงเอนทิตีมีผลต่อประเภทของความสัมพันธ์ หากเรากล่าวถึงประเภทความสัมพันธ์โดยนำเอนทิตีแผนกขึ้นก่อน เราเรียกความสัมพันธ์ระหว่างเอนทิตีแผนกกับเอนทิตีพนักงานว่าเป็นความสัมพันธ์แบบกลุ่มต่อหนึ่ง **(many-to-one)** ซึ่งเป็นคาร์ดินัลลิตีของความสัมพันธ์นั่นเอง เราสามารถกำหนดคาร์ดินัลลิตีของความสัมพันธ์ใน ERD ได้โดยใช้ตัวเลข 1 และตัวอักษร M กำกับที่เส้นเชื่อมระหว่างเอนทิตีและความสัมพันธ์ ตามรูปแบบความสัมพันธ์ระหว่างเอนทิตีในความสัมพันธ์นั้น โดย **1** และ **N** หรือ **M** แทนหนึ่งและกลุ่มในรูปแบบของคาร์ดินัลลิตีตามลำดับ

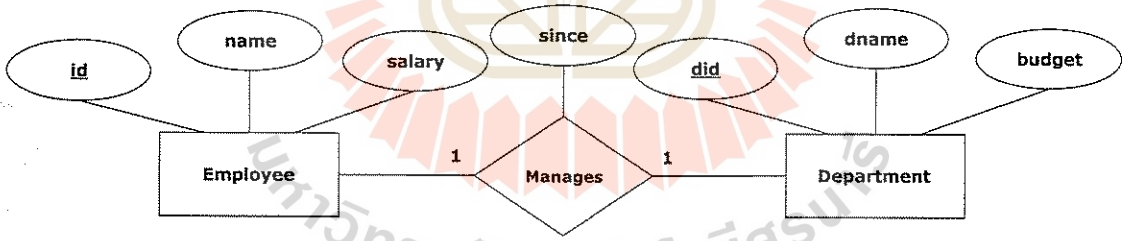


กรณีตัวอย่างในรูปที่ 1** ด้านล่าง แสดงกรณีตัวอย่างความสัมพันธ์ที่ไม่ขัดกับข้อกำหนดสำหรับความสัมพันธ์ Manages ถ้าลองพิจารณารูปที่ 1** ซึ่งเป็นกรณีตัวอย่างก่อนหน้า จะพบว่ากรณีตัวอย่างสามารถเป็นไปได้สำหรับความสัมพันธ์ Works_in แต่ขัดกับเงื่อนไขบังคับของความสัมพันธ์ Manages



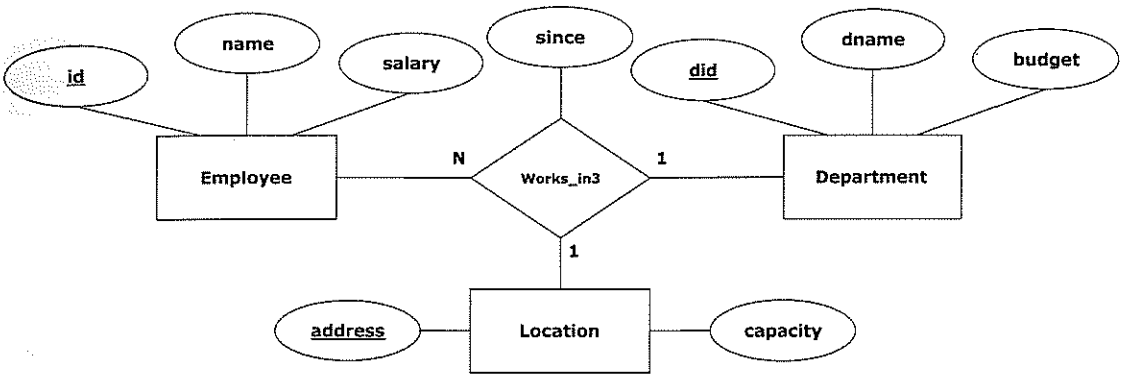
ตัวอย่าง

จากรูปที่ 1** หากเรากำหนดเงื่อนไขบังคับเพิ่ม โดยกำหนดให้นอกจากแผนกๆ หนึ่งสามาราถมีผู้จัดการได้เพียง 1 คนแล้ว เรายังอนุญาตให้พนักงานคนหนึ่งสามารถเป็นผู้จัดการให้กับแผนกๆ หนึ่งเท่านั้น ความสัมพันธ์ระหว่างเอนทิตีพนักงานและแผนกในความสัมพันธ์ Manages ที่มีเงื่อนไขบังคับเพิ่มเติมนี้มีคาร์ดินัลลิตีแบบ**หนึ่งต่อหนึ่ง (one-to-one)** เราสามารถแทนความสัมพันธ์ดังกล่าวใน ERD ดังนี้

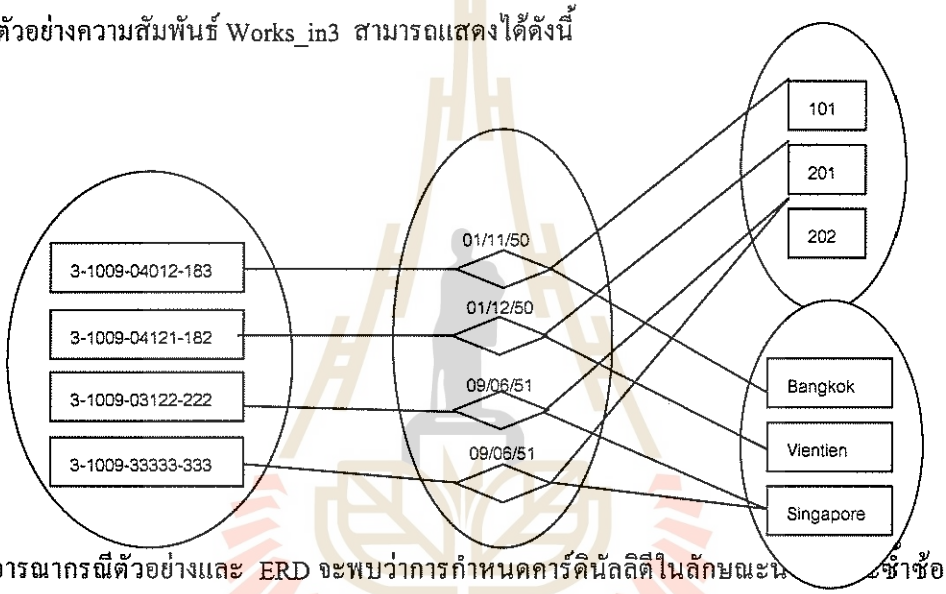


ตัวอย่าง

เงื่อนไขบังคับคีย์ สามารถใช้สำหรับความสัมพันธ์แบบไตรภาคได้เช่นเดียวกัน เช่นความสัมพันธ์ของพนักงาน แผนก และที่ตั้งออฟฟิศ ถ้าเรากำหนดเงื่อนไขเพิ่มเติมว่าพนักงานคนหนึ่งสามารถสังกัดแผนกใดแผนกหนึ่งและทำงาน ณ ออฟฟิศแห่งเดียวเท่านั้น ในขณะที่แผนกๆ หนึ่งยังคงมีพนักงานได้หลายคน ตลอดจนออฟฟิศแห่งหนึ่งเป็นที่ทำงานของพนักงานหลายคน แต่แผนกแต่ละแผนกจะต้องตั้งอยู่ที่ออฟฟิศเดียวเท่านั้นและออฟฟิศ 1 ออฟฟิศมีแผนกได้แผนกเดียว ERD ของความสัมพันธ์ดังกล่าวพร้อมทั้งเงื่อนไขบังคับสามารถแสดงได้ดังนี้



กรณีตัวอย่างความสัมพันธ์ Works_in3 สามารถแสดงได้ดังนี้



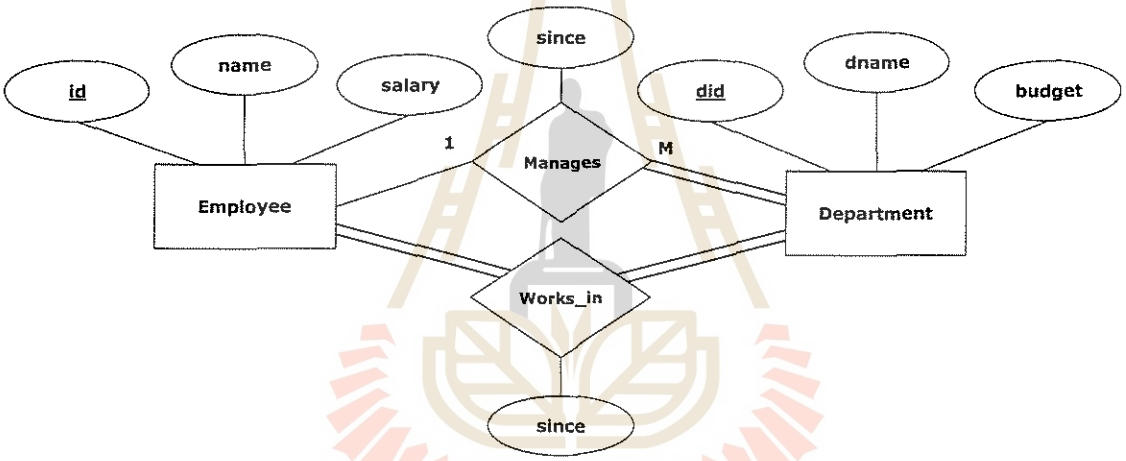
ถ้าพิจารณากรณีตัวอย่างและ ERD จะพบว่าการกำหนดคาร์ดินัลลิตีในลักษณะน[illegible]ซ้ำซ้อนกันโดยไม่จำเป็นกล่าวคือความสัมพันธ์ระหว่างแผนกและออฟฟิศที่ตั้งจะเป็นคู่เดียวกันตลอด ดังนั้นความสัมพันธ์แบบไตรภาคนั้นการกำหนดคาร์ดินัลลิตีใดๆ เราอาจต้องปรับ ERD ให้เป็นความสัมพันธ์แบบทวิภาค โดยจะได้กล่าวถึงในหัวข้อ 3.5.3 การเลือกใช้ระหว่างความสัมพันธ์ไตรภาคและทวิภาค โดยแท้จริงแล้วเรามีสัญลักษณ์อีกแบบหนึ่งที่ใช้กำหนดเงื่อนไขบังคับคีย์โดยตรงแทนคาร์ดินัลลิตี (ซึ่งมีใช้ใน Ramakrishnan, 2003 แต่ไม่ได้กล่าวถึงในที่นี้) การใช้สัญลักษณ์ทั้ง 2 รูปแบบมีวัตถุประสงค์เดียวกัน แต่รองรับเงื่อนไขบังคับในกรณีความสัมพันธ์แบบไตรภาคต่างกัน อย่างไรก็ตามในกรณีที่เราใช้สัญลักษณ์เงื่อนไขบังคับคีย์เราจะไม่สามารถรองรับเงื่อนไขบางประเภทได้ในขณะที่สัญลักษณ์คาร์ดินัลลิตีทำได้ จึงเลือกใช้สัญลักษณ์คาร์ดินัลลิตีที่เข้าใจได้ง่ายกว่าสำหรับ ERD ในที่นี้

3.4.2 เงื่อนไขบังคับการมีส่วนร่วม (Participation Constraints)

เงื่อนไขบังคับคีย์สามารถกำหนดว่าแผนกๆ หนึ่งสามารถมีผู้จัดการได้ 1 คนเท่านั้น แต่หากถามว่าแผนกๆ หนึ่งจำเป็นต้องมีผู้จัดการหรือไม่ ต้องมีการกำหนดเงื่อนไขตามลักษณะของข้อกำหนดจริงในการดำเนินธุรกิจ สมมติว่าจากคำถามดังกล่าว เรากำหนดว่าทุกแผนกจะต้องมีผู้จัดการ ข้อกำหนดลักษณะดังกล่าวเป็นตัวอย่างของ**เงื่อนไขบังคับการมีส่วนร่วม** สำหรับการมีส่วนร่วมของเอนทิตีในความสัมพันธ์ที่ทุกเอนทิตีต้องมีส่วนร่วมในความสัมพันธ์

นั้นๆ เราเรียกการมีส่วนร่วมแบบนี้ว่า**การมีส่วนร่วมทั้งหมด (total participation)** เช่น การมีส่วนร่วมของเอนทิตีแผนกในความสัมพันธ์ Manages2 ซึ่งทุกแผนกจะต้องได้รับการจัดการ สำหรับการมีส่วนร่วมของเอนทิตีที่เอนทิตีไม่ได้ปรากฏอยู่ในความสัมพันธ์ทั้งหมด เราเรียกการมีส่วนร่วมแบบนี้ว่าการมีส่วนร่วมแบบบางส่วน (partial participation) เช่น การมีส่วนร่วมของเอนทิตีพนักงานในความสัมพันธ์ Manages2 ซึ่งไม่จำเป็นที่พนักงานทุกคนต้องเป็นผู้จัดการ

หากเราเพิ่มเติมจากข้อกำหนด Manages2 ที่ได้กล่าวมาแล้ว โดยกำหนดให้พนักงานทุกคนต้องทำงานให้กับแผนกใดแผนกหนึ่งและทุกแผนกจะต้องมีพนักงานอย่างต่ำ 1 คน เราสามารถแทนข้อกำหนดความสัมพันธ์ดังกล่าวด้วย ERD ดังต่อไปนี้ ERD ใช้เส้น 2 เส้นหรือเส้นคู่เชื่อมความสัมพันธ์กับเอนทิตีจากที่ปกติเป็นเส้นเดี่ยว เส้นคู่นี้แทนเงื่อนไขบังคับสำหรับการมีส่วนร่วมในความสัมพันธ์แบบการมีส่วนร่วมทั้งหมด และเส้นเดี่ยวที่คงไว้ดังเดิมเป็นการมีส่วนร่วมแบบบางส่วน



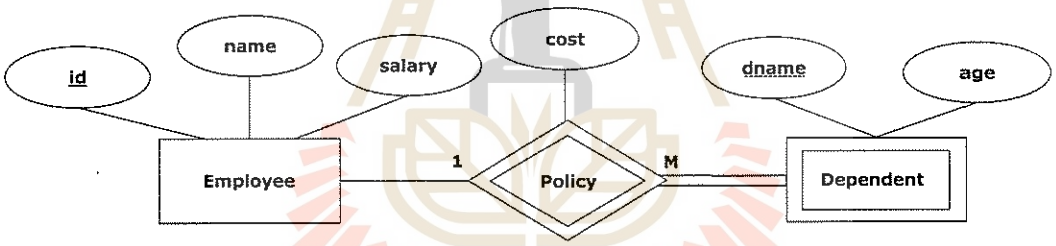
3.4.3 เอนทิตีอ่อนแอ (Weak Entities)

เอนทิตีที่กล่าวมาแล้วทั้งหมดล้วนเป็นเอนทิตีที่มีคีย์ ซึ่งในบางกรณีเอนทิตีไม่จำเป็นต้องมีคีย์ภายในเอนทิตีก็ได้ ยกตัวอย่างเช่น พนักงานสามารถซื้อกรมธรรม์สุขภาพสำหรับบุตร/ธิดา ฝ่ายบุคคลต้องการจัดเก็บข้อมูลเหล่านี้ โดยที่ข้อมูลของบุตร/ธิดานั้นไม่จำเป็นต้องเก็บไว้หากพนักงานคนนั้นๆ ลาออกไป เราสามารถเลือกที่จะระบุบุตร/ธิดาคนหนึ่งคนใดโดยใช้ชื่อเท่านั้นในการระบุ อย่างไรก็ตาม ชื่อบุตร/ธิดา ของพนักงานคนละคนอาจมีชื่อซ้ำกันได้ บุตรและธิดาของพนักงานคนเดียวกันย่อมมีชื่อไม่ซ้ำกัน การที่จะระบุบุตร/ธิดาให้เฉพาะเจาะจงสามารถกระทำได้โดยอ้างอิงถึงผู้ปกครองของบุตร/ธิดารวมกับชื่อของบุตร/ธิดาคนนั้น การที่เอนทิตีใดต้องอาศัยแอทริบิวต์ที่เป็นคีย์จากเอนทิตีอื่นๆ รวมกับแอนทริบิวต์บางแอทริบิวต์ในตัวเองเพื่อสร้างคีย์สำหรับเอนทิตีนั้นๆ เราเรียกเอนทิตีประเภทนี้ว่า **เอนทิตีอ่อนแอ (weak entity)**

กรณีตัวอย่างเพื่อแสดงการซื้อกรมธรรม์ของพนักงานให้กับบุตร/ธิดา สามารถยกตัวอย่างได้ดังนี้ สมชายและสมศรีเป็นพนักงานของบริษัท โดยสมชายมีบุตร/ธิดารวม 2 คนได้แก่คริสตินา และสรพงษ์ สมศรีมีบุตร 1 คนได้แก่ สรพงษ์ พนักงานทั้ง 2 คนนี้ซื้อกรรมธรรม์สุขภาพให้กับบุตร/ธิดาของตนเอง จากกรณีตัวอย่างข้อมูลพบว่าชื่อบุตร/ธิดาเท่านั้นไม่สามารถใช้เป็นคีย์ได้ เพราะเราไม่สามารถระบุถึงสรพงษ์ให้เฉพาะเจาะจงได้ เนื่องจากมีสรพงษ์ 2 คน เราต้องอ้างอิงรหัสพนักงานของของผู้ที่เป็นผู้ปกครองในความสัมพันธ์ซื้อกรมธรรม์ประกอบกับชื่อของบุตร/ธิดาด้วย เพื่อระบุว่าเป็นสรพงษ์ใด

พิจารณาลักษณะของเอนทิตีและความสัมพันธ์ที่เกี่ยวข้องกับเอนทิตีอ่อนแอ เราเรียกเอนทิตีที่เป็นเอนทิตีหลักที่เอนทิตีอ่อนแออ้างถึงว่าเอนทิตี**ผู้ระบุ (identifying owner)** และเรียกความสัมพันธ์ระหว่างเอนทิตีหลักกับเอนทิตีอ่อนแอว่า**ความสัมพันธ์แบบระบุ (indentifying relationship)** ของเอนทิตีอ่อนแอ เอนทิตีอ่อนแอนี้จะไม่คงอยู่ได้ถ้าเราลบข้อมูลในเอนทิตีหลัก ดังตัวอย่างทั้งข้อมูลของคริสตินาและสรพงษ์ที่เป็นธิดาและบุตรของนายสมชายจะถูกลบทิ้งไปเช่นเดียวกับข้อมูลของนายสมชายถ้านายสมชายลาออกจากบริษัท

หากวิเคราะห์ความสัมพันธ์ของเอนทิตี้ที่เกี่ยวข้องกับเอนทิตีอ่อนแอนี้จะพบว่า เอนทิตีหลักจะมีส่วนร่วมในความสัมพันธ์แบบระบุในลักษณะหนึ่งต่อกลุ่ม และลักษณะการมีส่วนร่วมของเอนทิตีอ่อนแอในความสัมพันธ์แบบระบุเป็นแบบการมีส่วนร่วมทั้งหมด ซึ่งสามารถแสดงได้ด้วยตัวอย่าง ERD ดังนี้



จากภาพ พนักงานซื้อกรมธรรม์คุ้มครองให้กับบุตร/ธิดา พนักงานจึงเป็นเอนทิตีผู้ระบุ ความสัมพันธ์กรมธรรม์คุ้มครอง (Policy) คือความสัมพันธ์แบบระบุ ซึ่งใช้เส้นคู่เป็นเส้นขอบของความสัมพันธ์ต่างจากการใช้เส้นเดี่ยวที่แสดงความสัมพันธ์ทั่วไป Policy บันทึกข้อมูลมูลค่า (cost) ของกรมธรรม์ เอนทิตีบุตร/ธิดา (Dependent) คือเอนทิตีอ่อนแอ จัดเก็บข้อมูลชื่อ (dname) และอายุ (age) ของบุตรธิดาแต่ละคน เราอาศัยรหัสพนักงานในความสัมพันธ์ Policy ซึ่งได้มาจากคีย์หลักของเอนทิตีผู้ระบุและ dname จากเอนทิตี Dependent ซึ่งเป็นเอนทิตีอ่อนแอเพื่อเป็นคีย์ที่ใช้ในการระบุตัวบุคคลที่ได้รับความคุ้มครองจากกรมธรรม์

3.4.4 ลำดับชั้นของคลาส (Class Hierarchies)

บางครั้งเราสามารถจำแนกเอนทิตีออกเป็นชนิดย่อยๆ ของเอนทิตีได้ เช่น พนักงานอาจจะมีพนักงานสัญญาจ้าง(Contract_Emps) กับพนักงานรายชั่วโมง (Hourly_Emps) ซึ่งเหตุผลที่เราแยกพนักงานออกเป็น 2 ประเภทนี้เนื่องจากมีข้อมูลบางส่วนนอกจากข้อมูลพนักงานที่แตกต่างกันเช่น พนักงานรายชั่วโมงมีการบันทึกจำนวนชั่วโมงที่ได้

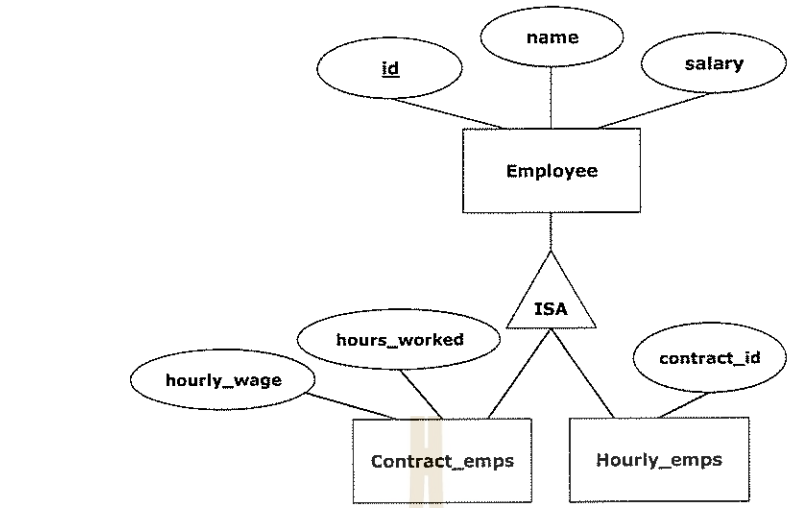
ทำไปแล้ว (hours_worked) และอัตราค่าจ้างต่อชั่วโมง (hours_wage) และมีการบันทึกหมายเลขสัญญาจ้าง (contract_id) สำหรับพนักงานสัญญาจ้าง

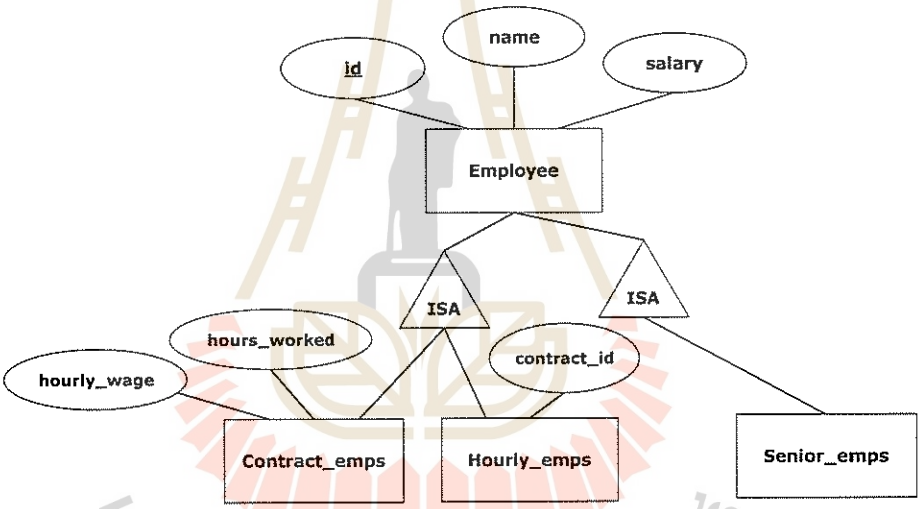
การแบ่งแยกชนิดของเอนทิตีนั้นเราอาศัยแนวคิดเชิงวัตถุที่เรียกว่าคลาส คลาสคือชนิดของเอนทิตีซึ่งอนุญาตให้คลาสย่อย (subclass) สามารถสืบทอด (inherited) คุณสมบัติจากคลาสหลัก (superclass) ได้ คลาสย่อยจะมีคุณสมบัติหรือแอทริบิวต์เช่นเดียวกับคลาสหลักและสามารถกำหนดคุณสมบัติเพิ่มเติมได้ จากตัวอย่าง Hourly_emps และ Contract_emps สืบทอดมาจากคลาส Employee เราให้ชื่อความสัมพันธ์ระหว่างคลาสหลักกับคลาสย่อยนี้ว่า ความสัมพันธ์ **เป็น (is a—ISA)** การสืบทอดจากคลาสหลักนั้นคลาสย่อยจะมีความเฉพาะเจาะจงลงไปอีกเช่นจากพนักงานทั่วไป ทำการสืบทอดกลายเป็นพนักงานสัญญาจ้าง ซึ่งมีความเฉพาะเจาะจง การสืบทอดจากคลาสหลักนี้มีลักษณะโดยทั่วไปเรียกว่าเอนทิตีหลักนั้น**ถูกทำให้มีลักษณะเฉพาะ (specialized)** ในขณะที่เราอาจมองกลับกันโดยมองว่าเอนทิตีย่อยนั้น**ถูกทำให้มีลักษณะทั่วไป (generalized)** มากขึ้น

การสืบทอดคลาสยังสามารถแบ่งสืบทอดได้หลายลำดับเช่นเราอาจสืบทอดเอนทิตี Employee เป็นพนักงานอาวุโส (Senior_emps) ซึ่งสามารถสืบทอดต่อด้านล่างของเอนทิตี Employee และ Senior_emps นี้สามารถถูกสืบทอดได้อีก

การออกแบบ ER model โดยใช้ประโยชน์จากแนวคิดของลำดับชั้นนี้ เราสามารถกำหนดเงื่อนไขบังคับบางอย่างในความสัมพันธ์ระหว่างเอนทิตีหลักกับเอนทิตีที่สืบทอดนี้ได้ ได้แก่

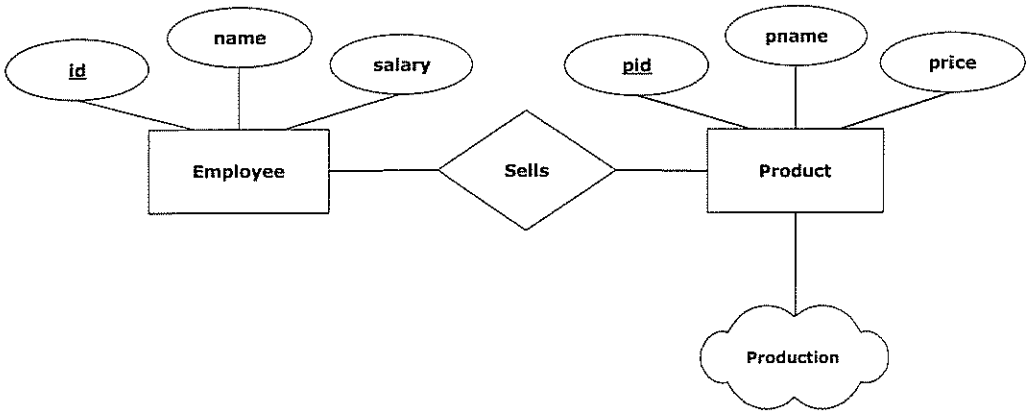
- **เงื่อนไขบังคับการซ้อนทับ (Overlap Constraints)** เป็นการระบุว่าเราอนุญาตให้เอนทิตีหนึ่งเอนทิตีใดสามารถอยู่ในกลุ่มย่อยหรือคลาสย่อยได้พร้อมๆ กันมากกว่า 1 คลาสหรือไม่ เช่น เราไม่อนุญาตให้พนักงานคนหนึ่งเป็นได้ทั้ง Hourly_emps และ Contract_emps เราอาจอนุญาตให้ Contract_emps เป็นพนักงานแบบ Senior_emps ได้ เงื่อนไขบังคับในกรณีที่เราอนุญาตให้คลาสมีการซ้อนทับกันคือ OVERLAPS โดยเขียนเงื่อนไขบังคับได้ดังตัวอย่าง "Contract_emps OVERLAPS Senoir_emps" ในกรณีที่เราไม่ได้กำหนดเงื่อนไขบังคับ OVERLAP นี้หมายถึงเราไม่อนุญาตให้คลาสซ้อนทับกัน
- **เงื่อนไขบังคับการครอบคลุม (Covering Constraints)** เป็นการระบุว่าเอนทิตีทั้งหมดที่อยู่ในคลาสหลักจะต้องจำแนกได้เป็นหรืออยู่ในคลาสย่อยด้วยหรือไม่ เช่น Employee ทุกคนจะต้องเป็นพนักงาน Contract_emps หรือ Hourly_emps ด้วยหรือไม่ ซึ่งไม่จำเป็นอาจมีพนักงานประจำที่ไม่ได้เซ็นสัญญาเป็นงวดๆ หรือไม่ได้เป็นพนักงานรายชั่วโมง แต่ถ้าเราจำแนกพนักงานออกเป็น 2 กลุ่มย่อยได้แก่ พนักงานชาย (Male_emps) และพนักงานหญิง (Female_emps) พนักงานที่อยู่ในกลุ่มย่อยทั้ง 2 กลุ่มทุกคนล้วนอยู่ในกลุ่ม Employee ทั้งสิ้น ซึ่งเราเขียนเงื่อนไขบังคับได้ดังตัวอย่าง "Male_emps AND Female_emps COVER Employee" หากเราไม่ระบุเงื่อนไขบังคับการครอบคลุมนี้ หมายความว่าเราอนุญาตที่จะให้มีพนักงานที่ไม่ได้เป็นทั้งชายหรือหญิง!





3.4.5 การรวมกลุ่ม (Aggregation)

ในบางกรณีที่ ERD มีขนาดใหญ่มากๆ เราสามารถรวมกลุ่มของ ERD บางส่วนเข้าเป็นกลุ่ม (aggregation) ได้ เพื่อสะดวกในการแสดง ERD เช่น เราอาจกำลังออกแบบ ERD ของฝ่ายขายอาจประกอบด้วยลูกค้า การสั่งซื้อและสินค้า แต่ในขณะที่ ERD ของฝ่ายผลิตสินค้า (Production) มีเอนทิตีในอีกลักษณะหนึ่งเช่น วัตถุดิบ แหล่งวัตถุดิบ พนักงานผู้ผลิต เครื่องจักรที่เกี่ยวข้อง ฯลฯ ทั้งนี้เราสามารถรวมกลุ่มของเอนทิตีในฝ่ายผลิตไว้เป็น aggregation ก่อนเป็นต้น ในที่นี้เป็นการใช้ aggregation ต่างตากรูปแบบของ Elmarsi และ Navathe เราไม่ค่อยใช้ aggregation บ่อยนัก รูปต่อไปนี้แสดงตัวอย่างของการใช้ aggregation สำหรับกลุ่มของเอนทิตีและความสัมพันธ์ Production

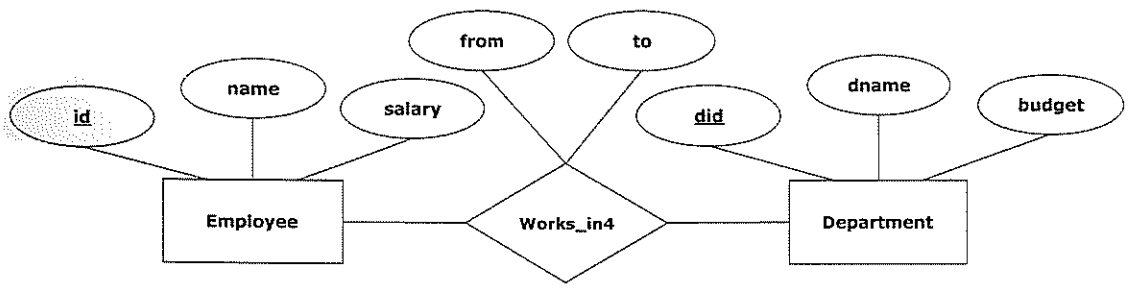


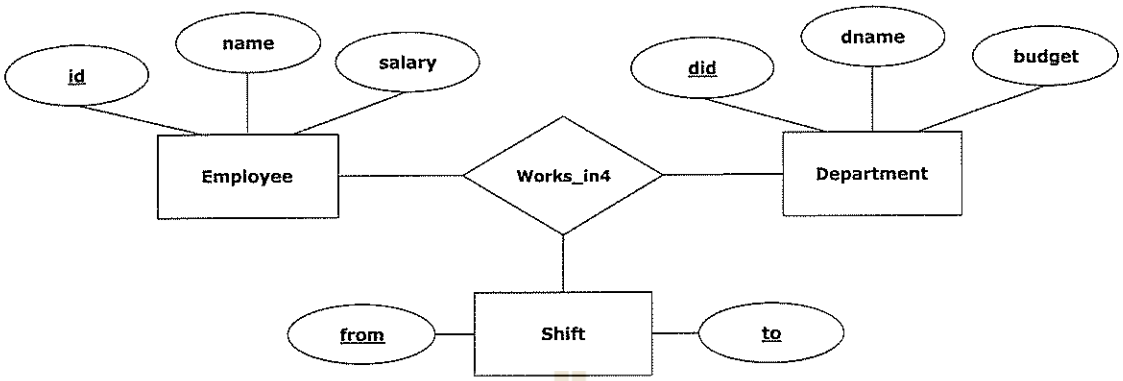
3.5 ข้อควรคำนึงถึงในการออกแบบแบบจำลองเอนทิตีและความสัมพันธ์

3.5.1 การกำหนดให้เป็นเอนทิตีหรือแอทริบิวต์

ในบางครั้งของการออกแบบ ER model เราอาจเกิดความไม่แน่ใจว่าจะกำหนดให้ข้อมูลนั้นๆ เป็นเอนทิตีหรือเป็นแอทริบิวต์ เช่น แอทริบิวต์ที่อยู่ (address) ของเอนทิตีพนักงาน เราสามารถออกแบบให้ address นั้นเป็นได้ทั้งแอทริบิวต์และเอนทิตีขึ้นอยู่กับลักษณะการใช้งานดังนี้ หากเรากำหนดให้ address เป็นแอทริบิวต์หนึ่งของเอนทิตีพนักงาน พนักงานคนหนึ่งจะมีที่อยู่ได้เพียงที่อยู่เดียว หรืออาจมีมากกว่า 1 แต่ต้องกำหนดจำนวนแอทริบวต์เป็น address1 address2 จำกัดตามจำนวนที่อยู่ซึ่งไม่สมเหตุสมผล ในกรณีที่เราต้องการให้พนักงานมีข้อมูล address ได้มากกว่า 1 เราควรแยก address ออกมาเป็น entity แล้วกำหนดความสัมพันธ์ระหว่างเอนทิตีพนักงานกับ address นอกจากนี้เรายังสามารถกระจายเอนทิตี address ที่จัดเก็บในลักษณะของข้อความตัวอักษรจัดเก็บข้อมูลที่อยู่ของพนักงานให้เป็นแอทริบิวต์ เลขที่ ถนน แขวง/ตำบล เขต/อำเภอ จังหวัด และรหัสไปรษณีย์ได้ ซึ่งการกระจายข้อมูลเป็นแอทริบิวต์ดังกล่าวจะสามารถทำให้เราค้นหาข้อมูลพนักงานด้วยข้อคำถามเช่น "พนักงานคนใดบ้างที่อาศัยอยู่ในเขตพระโขนง" เป็นต้น

นอกจากนี้พิจารณากรณีที่พนักงานคนหนึ่งสามารถทำงานให้กับแผนกๆ หนึ่งแบ่งเป็นกะให้กับแผนกต่างๆ รูปที่ 1** แสดงการออกแบบ ERD แทนการทำงานในลักษณะเป็นกะของพนักงาน โดยบันทึกข้อมูลเวลาเริ่มต้นและเลวสิ้นสุดของกะทำงาน เราพบว่าการทำงานของพนักงานมีเพียง 3 กะเท่านั้นได้แก่ 8:00-16:00 น. 16:00-24:00 น. และ 24:00-8:00 น. การออกแบบในลักษณะดังกล่าวจะทำให้เกิดความซ้ำซ้อนของข้อมูล เราสามารถแยกเวลาเริ่มต้นและสิ้นสุดของแต่ละกะเป็นเอนทิตีกะ (Shift) การทำงานได้ ซึ่งในกรณีนี้จะมีข้อมูลอยู่ที่ 3 ระเบียนในเบื้องต้น

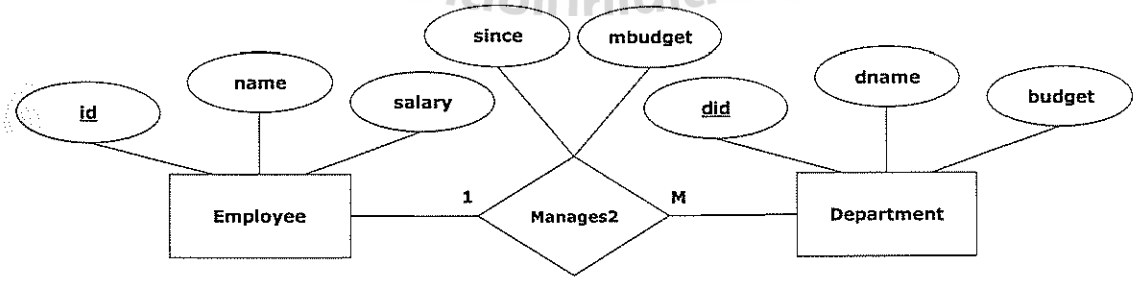


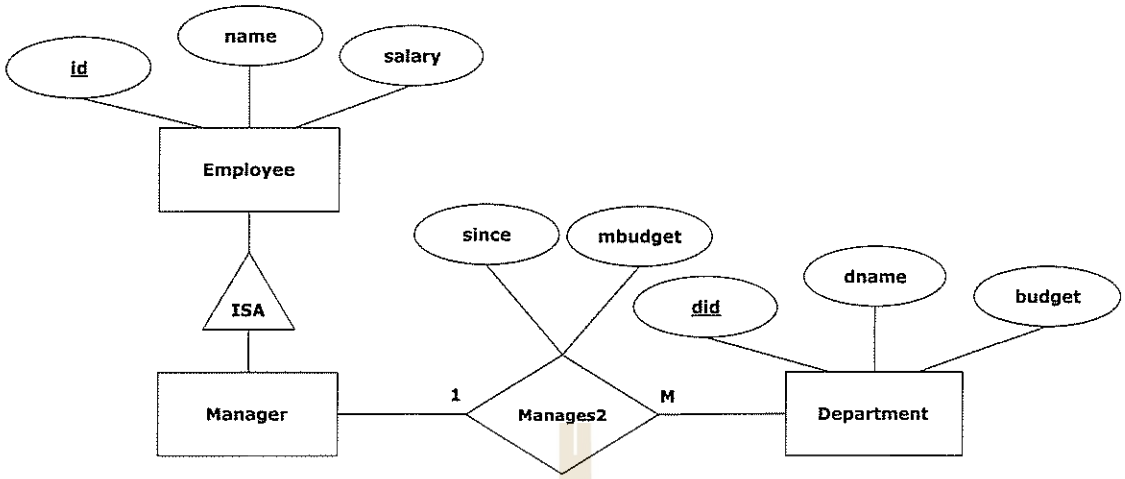


สรุปได้ว่าการออกแบบโดยเลือกใช้เอนทิตีแทนแอทริบิวต์ในกรณีที่มีความไม่แน่ใจในการออกแบบอาจเป็นกรณีของการที่ข้อมูลนั้นๆ มีความสัมพันธ์กับเอนทิตีเดียวกันในกลุ่มของเอนทิตีมากกว่า 1 ครั้ง หรือการจัดเก็บข้อมูลที่มีอาจมีความซ้ำซ้อนกันก็ได้ นอกจากนี้อาจมีกรณีอื่นๆ อีกซึ่งต้องเลือกใช้ให้ถูกต้องตามลักษณะการใช้งานจริง

3.5.2 การกำหนดให้เป็นเอนทิตีหรือความสัมพันธ์

ในกรณีที่ความสัมพันธ์ไม่สามารถรองรับการจัดเก็บข้อมูลตามความต้องการของผู้ใช้ได้ถูกต้อง เราอาจต้องพิจารณาปรับเปลี่ยนความสัมพันธ์โดยการเพิ่มเอนทิตีเข้าไปใน ER model เช่น จากตัวอย่างความสัมพันธ์ Manages2 นี้ ผู้จัดการแต่ละคนจะได้รับงบประมาณค่าใช้จ่ายส่วนตัวสำหรับแต่ละแผนก แต่ถ้าเราเพิ่มข้อกำหนดว่าผู้จัดการแต่ละคนจะได้รับงบประมาณค่าใช้จ่ายส่วนตัว (mbudget) ต่อคนเป็นจำนวนหนึ่งเท่านั้นสำหรับใช้จ่ายในการเป็นผู้จัดการของทุกแผนกที่ตนเองเป็นผู้จัดการ เราไม่สามารถใช้ ERD ในรูปที่ 1** ได้อีกต่อไป เนื่องจากการแอทริบิวต์ mbudget นั้นแสดงว่าผู้จัดการได้เงินใช้จ่ายส่วนตัวแยกแต่ละแผนก ซึ่งไม่ตรงตามข้อกำหนด เราสามารถสร้าง ERD ให้เป็นไปตามข้อกำหนดได้ดังแสดงในรูปที่ 1** ที่งบประมาณค่าใช้จ่ายส่วนตัวนั้นผูกติดอยู่กับผู้จัดการแต่ละคนซึ่งจะต้องใช้เงินจำนวนนี้เท่านั้นไม่ว่าจะเป็นผู้จัดการแผนกหลายแผนกก็ตาม



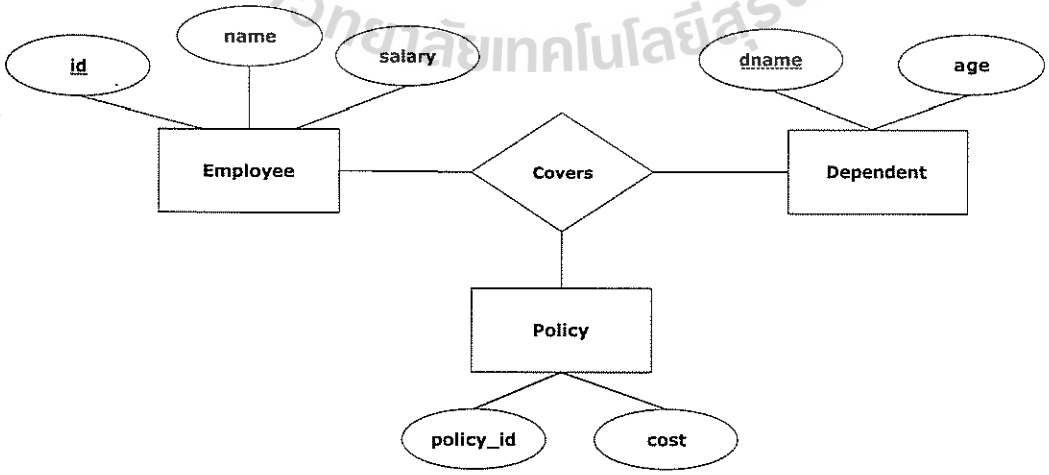


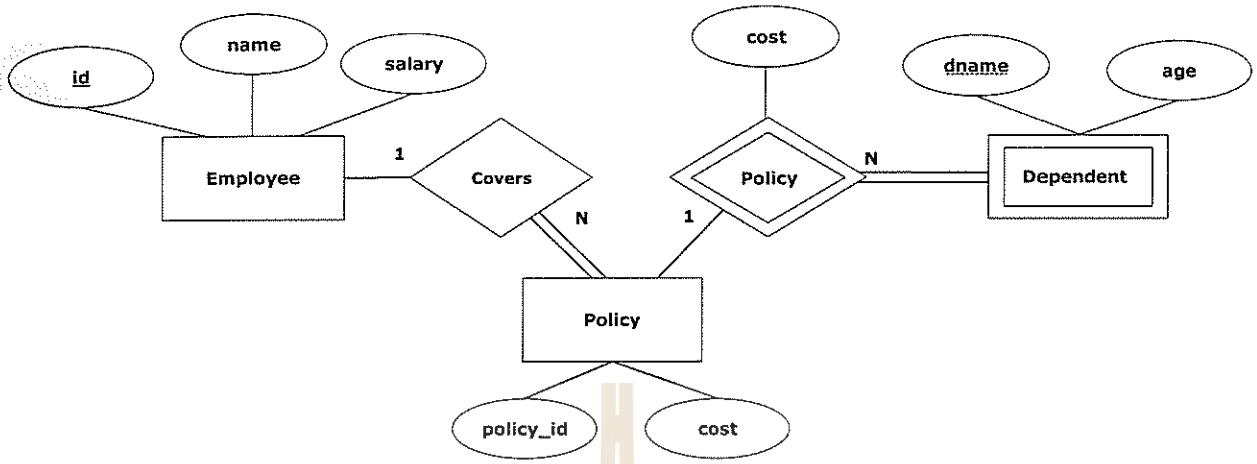
จากตัวอย่างสามารถสรุปได้ว่าหากความสัมพันธ์ไม่สามารถรองรับข้อมูลที่จะจัดเก็บตามความต้องการของผู้ใช้ได้ เรามีความจำเป็นต้องปรับเปลี่ยนการออกแบบซึ่งอาจอยู่ในรูปของการเพิ่มเอนทิตีและความสัมพันธ์ได้

3.5.3 การกำหนดให้เป็นความสัมพันธ์แบบทวิภาคหรือไตรภาค

รูปที่ 1** เป็นแบบจำลองของข้อมูลที่แสดงว่าพนักงานคนหนึ่งสามารถเป็นเจ้าของกรมธรรม์ประกันสุขภาพได้หลายกรมธรรม์ กรมธรรม์ๆ หนึ่งมีเจ้าของได้หลายคน และบุตร/ธิดาผู้ได้รับความคุ้มครอง (Dependents) สามารถได้รับความคุ้มครองจากกรมธรรม์หลายๆ กรมธรรม์

ในกรณีที่เราเพิ่มข้อกำหนดเกี่ยวกับกรมธรรม์ดังกล่าวโดยระบุว่ากรมธรรม์ๆ หนึ่ง ไม่สามารถมีเจ้าของร่วมได้ กรณีนี้เราจะไม่สามารถกำหนดเงื่อนไขบังคับคีย์ได้เพราะจะมีผลทำให้กรรมธรรม์หนึ่งคุ้มครองบุตร/ธิดาเพียงคนเดียวเท่านั้น เราสามารถปรับ ERD ได้รับรูปที่ 1** ซึ่งจะพบว่าเราอนุญาตให้กรรมธรรม์หนึ่งกรมธรรม์คุ้มครองบุตร/ธิดาได้หลายคน แต่เจ้าของผู้ซื้อกรมธรรม์มีได้เพียงคนเดียว





เราสามารถตั้งข้อสังเกตได้ว่าการกำหนดเงื่อนไขการมีส่วนร่วมบางประการสำหรับความสัมพันธ์แบบไตรภาค นั้นเราอาจต้องแยกความสัมพันธ์ออกเป็นเอนทิตีและใช้ความสัมพันธ์แบบทวิภาคแทน

3.6 ขั้นตอนการออกแบบ ER model

การออกแบบ ERD นั้นมีระเบียบวิธีที่สามารถเป็นแนวปฏิบัติเพื่อออกแบบ ERD ได้อย่างมีประสิทธิภาพดังนี้

1. ระบุเอนทิตีที่เกี่ยวข้อง
2. ระบุความสัมพันธ์ของเอนทิตี
3. ระบุข้อมูลหน่วยย่อยหรือที่เรียกว่าแอทริบิวต์ที่สัมพันธ์กับเอนทิตีและความสัมพันธ์ของเอนทิตี
4. ระบุขอบเขตของแอทริบิวต์
5. ระบุคีย์หลักที่ใช้แทนข้อมูลแต่ละระเบียนในเอนทิตีและความสัมพันธ์
6. กำหนดข้อกำหนดหรือเงื่อนไขของข้อมูลโดยการใช้คุณสมบัติที่แบบจำลอง ER รองรับ ขั้นตอนนี้เป็นขั้นตอนเสริมเท่านั้นซึ่งไม่จำเป็นต้องปฏิบัติ
7. ตรวจสอบแบบจำลองที่ได้มาเพื่อกำจัดความซ้ำซ้อนของข้อมูล
8. ตรวจสอบแบบจำลองระดับแนวคิดที่ออกแบบว่าสามารถรองรับธุรกรรมของผู้ใช้ได้ทั้งหมด
9. ยืนยันกับผู้ใช้

ระเบียบวิธีนี้ได้นำมาใช้กับกรณีศึกษาอย่างย่อๆ ในหัวข้อถัดไปเพื่อให้ได้มาซึ่ง ERD อย่างไรก็ตามการแสดงการออกแบบ ERD โดยละเอียดตามระเบียบวิธีการออกแบบได้แสดงไว้ในภาคผนวก

3.7 กรณีศึกษา

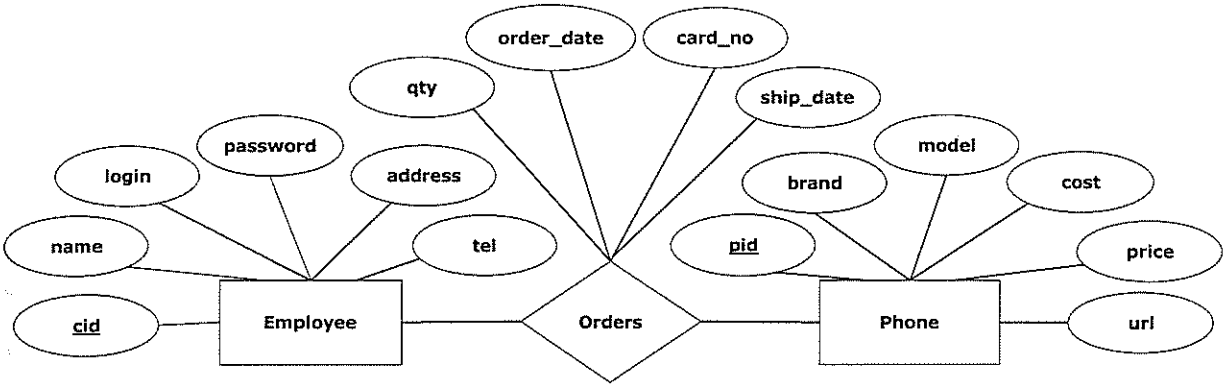
ร้านมือถือออนไลน์ 7-Elephant

ตัวอย่างร้าน 7-Elephant นี้แสดงการออกแบบ ERD สำหรับร้านมือถือออนไลน์ ซึ่งแสดงเฉพาะ ERD ที่ได้มาจากความต้องการของผู้ใช้ ในภาคผนวก เราได้ใช้ตัวอย่างร้าน 7-Elephant นี้เป็นกรณีศึกษาในการออกพัฒนาระบบฐานข้อมูล โดยแสดงขั้นตอนโดยละเอียดทุกขั้นตอน

เจ้าของร้าน 7-Elephant เล่าถึงขั้นตอนการทำงานของระบบที่ต้องการได้โดยละเอียด (ซึ่งโดยทั่วไปแล้วการระบุความต้องการของผู้ใช้นั้นต้องอาศัยทักษะของผู้รวบรวมและวิเคราะห์ความต้องการของผู้ใช้ในการรวบรวมข้อมูลและซักถามความต้องการจากผู้ใช้ซึ่งใช้เวลาพอสมควร) ดังนี้

"7-Elephant ต้องการพัฒนาระบบร้านค้าออนไลน์เพื่อให้ลูกค้าแสดงแคตตาล็อกมือถือและสั่งซื้อมือถือบนอินเตอร์เน็ตได้ ปัจจุบันลูกค้าสั่งซื้อสินค้าทางโทรศัพท์ ลูกค้าส่วนใหญ่เป็นลูกค้าประจำของร้าน เป็นลูกค้าระดับบริษัทขนาดกลางและใหญ่ ทางร้านจัดเก็บข้อมูลของลูกค้าไว้ได้แก่ ชื่อลูกค้า (name) ที่อยู่ (address) และหมายเลขโทรศัพท์ (tel) จะทำการสั่งซื้อสินค้าโดยบอกยี่ห้อและรุ่นของโทรศัพท์ ตลอดจนจำนวนที่ต้องการสั่งซื้อ โดยส่วนใหญ่เอาไปให้พนักงานในบริษัทใช้ในกรณีออกติดต่อกับลูกค้านอกสำนักงาน ลูกค้ามักชำระค่าสินค้าด้วยบัตรเครดิต จากนั้นร้าน 7-Elephant จะทำการเตรียมสินค้าให้ครบตามจำนวนที่ผู้ใช้สั่งซื้อเพื่อจัดส่งสินค้า ในกรณีที่สินค้ามีไม่พอกับยอดที่สั่ง ทางร้านจะสั่งสินค้ามาเพิ่มจนกว่าจะได้ครบตามกำหนดแล้วค่อยจัดส่งสินค้าไปในคราวเดียวกันสำหรับการสั่งซื้อแต่ละครั้ง ในแคตตาล็อกจะมีข้อมูลของมือถือทั้งหมดที่ร้านจำหน่าย โดยข้อมูลของมือถือ (Phone) ได้แก่ ยี่ห้อ (brand) รุ่น (model) ราคาทุน (cost) ราคาขาย (price) และ URL ของเว็บไซต์อย่างเป็นทางการของมือถือแต่ละรุ่น (url) เพื่อให้ลูกค้าสามารถดูรายละเอียดข้อกำหนดและความสามารถของมือถือแต่ละรุ่น เพื่อความสะดวกในการจัดการสินค้า ร้านค้ายังกำหนดรหัสประจำมือถือแต่ละรุ่น (pid) เนื่องจากรุ่นของมือถือแต่ละยี่ห้ออาจตั้งชื่อมาซ้ำกันได้ ลูกค้าที่จะสั่งซื้อสินค้าทางอินเตอร์เน็ตต้องมีชื่อผู้ใช้ (login) และรหัสผ่าน (password) ที่สัมพันธ์กับลูกค้า ลูกค้าที่ยังไม่มีข้อมูลกับทางร้านจะต้องโทรศัพท์มาเพื่อบันทึกข้อมูลลูกค้าและสร้างบัญชีผู้ใช้ก่อน ทางร้านจะกำหนดให้ลูกค้าแต่ละคนมีหมายเลขประจำลูกค้า (cid) สำหรับจัดเก็บข้อมูลการสั่งซื้อ หลังจากที่ลูกค้าเข้าใช้ระบบ แสดงสินค้าจากแคตตาล็อกและสั่งซื้อสินค้า"

ERD เบื้องต้นของระบบร้านค้าออนไลน์สามารถวิเคราะห์และออกแบบได้ดังนี้



** Employee

แม้ว่าระบบจะรองรับการเข้าใช้เว็บไซต์ ตลอดจนสั่งซื้อและจัดส่งมือถือได้ อย่างไรก็ตามหากพิจารณาเงื่อนไขที่เจ้าของร้านยังไม่ได้กล่าวถึง เช่น การสั่งซื้อมือถือรุ่นเดิมอีกครั้งในวันหลังจะไม่สามารถกระทำได้ ซึ่งจะต้องสอบถามถึงประเด็นดังกล่าวกับผู้ใช้งานระบบอีกครั้งหนึ่งเพื่อปรับการออกแบบ ERD

มหาวิทยาลัยเทคโนโลยีสุรนารี

มหาวิทยาลัยเทคโนโลยีสุรนารีเป็นมหาวิทยาลัยที่ตั้งอยู่บนที่ราบสูงแห่งหนึ่ง มีความต้องการพัฒนาระบบฐานข้อมูลเพื่ออำนวยความสะดวกในการจัดการเรียนการสอน ได้แก่การจัดการรายวิชา การลงทะเบียน การบันทึกผลการศึกษา ตลอดจนบันทึกข้อมูลการบริหารงานของสำนักวิชาและสาขาวิชาของผู้บริหาร ดังนี้

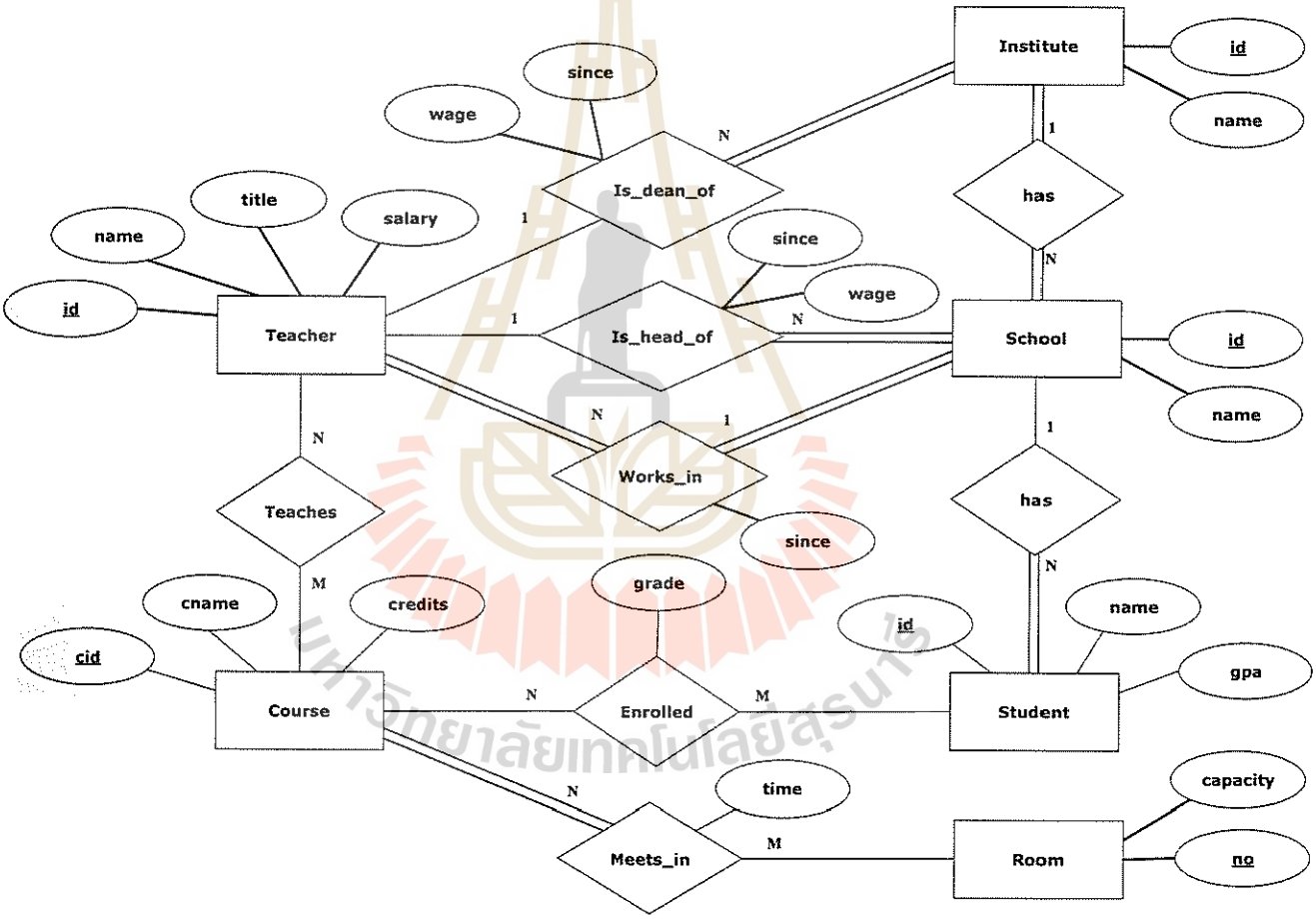
"มหาวิทยาลัยแบ่งการศึกษาออกเป็นสำนักวิชา 5 สำนักวิชามีรูปแบบเทียบเท่าคณะในมหาวิทยาลัยอื่นๆ ในแต่ละสำนักวิชายังแบ่งออกเป็นสาขาวิชาซึ่งมีรูปแบบเหมือนภาควิชานั่นเอง สาขาวิชาหนึ่งสาขาวิชาจะอยู่ภายใต้สำนักวิชาเพียงหนึ่งสำนักวิชาเท่านั้น คณาจารย์และนักศึกษา จะต้องสังกัดสาขาวิชาใดๆ สาขาวิชาหนึ่ง เพื่อประโยชน์การดูแลนักศึกษาและการบริหาร

ข้อมูลในการเรียนการสอนที่จะต้องจัดเก็บได้แก่การสอนรายวิชาของอาจารย์ว่าสอนในรายวิชาใด อาจารย์แต่ละคนสามารถสอนได้หลายรายวิชา และแต่ละรายวิชามีอาจารย์สอนได้หลายท่าน นักศึกษาสามารถลงทะเบียนได้หลายรายวิชา และรายวิชาแต่ละรายวิชาสามารถมีนักศึกษาลงทะเบียนได้หลายคนเช่นปกติทั่วไป ห้องที่ใช้ในการเรียนสำหรับแต่ละรายวิชาสามารถมีได้หลายห้องตามตารางสอนของแต่ละวัน ข้อมูลตารางการเรียนการสอน การใช้ห้องจะบันทึกไว้เฉพาะในภาคการศึกษาปัจจุบันเท่านั้น

ข้อมูลที่จัดเก็บเกี่ยวกับการบริหารด้านวิชาการได้แก่การจัดเก็บข้อมูลการบริหารสาขาวิชาและสำนักวิชา สำนักวิชาแต่ละสำนักวิชาจะต้องมีคณบดีซึ่งเป็นอาจารย์เป็นผู้จัดการสำนักวิชานั้น 1 คน โดยอาจารย์ 1 คนสามารถเป็นคณบดีได้หลายสำนักวิชา สาขาวิชาแต่ละสาขาวิชาจะต้องมีหัวหน้าสาขาวิชาซึ่งเป็นอาจารย์สาขาละ 1 คน อาจารย์ 1 คนสามารถเป็นหัวหน้าสาขาได้หลายสาขา โดยจะต้องทำการจัดเก็บอัตราเงินประจำตำแหน่ง และวันที่เริ่มวาระของผู้บริหารแต่ละคน ผู้บริหารแต่ละคนมีวาระ 4 ปีในแต่ละวาระ

ระบบดังกล่าวจะมีเจ้าหน้าที่ผู้บันทึกข้อมูลทั้งหมดไม่ว่าจะเป็นการเพิ่มข้อมูลนักศึกษา คณาจารย์ การบันทึกรายละเอียดรายวิชา ตารางการใช้ห้อง และเกรดของนักศึกษา ข้อมูลของคณาจารย์ นักศึกษา หน่วยงาน ฯลฯ ให้เก็บตามความเหมาะสม"

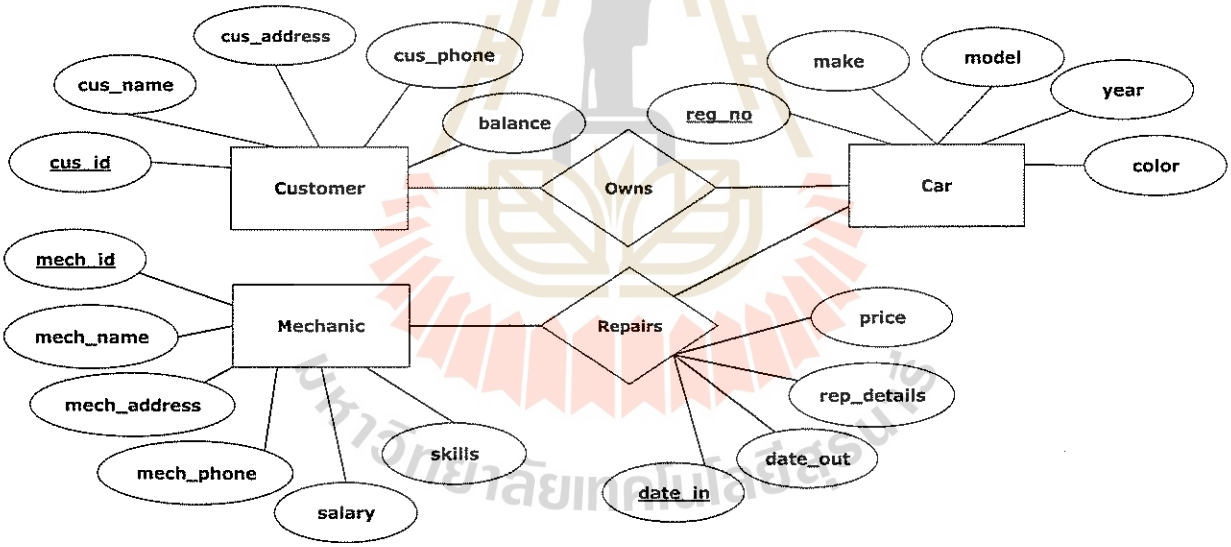
เราสามารถนำข้อกำหนดเหล่านี้มาออกแบบ ERD เบื้องต้นได้ดังนี้ ทั้งนี้จะพบว่ามีการกำหนดชื่อของเอนทิตีและแอทริบิวต์เป็นภาษาอังกฤษตามความนิยมในการออกแบบ ERD ทำให้พบว่าเรามีความจำเป็นต้องมีการอธิบายความหมายของคำแต่ละคำด้วย ซึ่งเราสามารถใช้พจนานุกรมข้อมูลในการอธิบายความหมายของคำแต่ละคำใน ERD ได้ ซึ่งพจนานุกรมนี้ได้กล่าวถึงในบทที่ 4 แบบจำลองเชิงสัมพันธ์ ตัวอย่างนี้มีการกล่าวถึงเงื่อนไขบังคับการมีส่วนร่วม และคาร์ดินัลลิตีของความสัมพันธ์ จึงได้ระบุไว้ใน ERD ด้วย



อู่เฉิดฉาย

เจ๊เกี๊ยวกุ้งเจ้าของอู่ซ่อมรถเฉิดฉายต้องการระบบจัดการอู่ซ่อมรถซึ่งมีความสามารถดังนี้

"ระบบอู่ซ่อมรถที่จะพัฒนาขึ้นควรที่จะสามารถจัดเก็บประวัติของลูกค้า (customer) ได้แก่รหัสลูกค้า (cus_id) ชื่อ (cus_name) ที่อยู่ (cus_address) หมายเลขโทรศัพท์ (cus_phone) และยอดค่าใช้บริการที่ยังไม่ได้ชำระ (balance) ตลอดจนข้อมูลของรถยนต์ (car) ได้แก่ทะเบียนรถยนต์ (reg_no) ยี่ห้อ (make) รุ่น (model) สี (color) และปีที่ผลิต (year) ลูกค้าสามารถเป็นเจ้าของรถได้หลายคนแต่รถยนต์หนึ่งคันมีเจ้าของได้เพียงคนเดียว ระบบจะต้องจัดเก็บประวัติการซ่อมรถได้แก่รถยนต์ที่ซ่อม (reg_no) ช่างผู้ซ่อม (mech) วันที่นำมาซ่อม (date_in) วันที่ซ่อมเสร็จ (date_out) รายละเอียดการซ่อม (rep_details) ซึ่งเป็นคำอธิบายรายละเอียดการซ่อม และค่าใช้จ่ายที่เรียกเก็บลูกค้า (price) ระบบต้องจัดเก็บข้อมูลช่างซ่อม (mech) ด้วย โดยจัดเก็บรหัสช่างซ่อม (mech_id) ชื่อ (mech_name) ที่อยู่ (mech_address) หมายเลขโทรศัพท์ (mech_phone) เงินเดือน (salary) และยี่ห้อของรถที่มีความชำนาญ (skills) ในรูปแบบของสตริงแสดงรายการยี่ห้อรถยนต์ เพื่อใช้ประโยชน์ในการจัดสรรงาน ระบบดังกล่าวจะใช้ในการจัดสรรช่างสำหรับซ่อมรถแสดงรายการของรถที่อยู่ในระหว่างซ่อม และออกใบเสร็จ"



3.8 แบบฝึกหัดท้ายบท

1. ให้นักศึกษาใช้วัตถุประสงค์ของบทเป็นแบบฝึกหัดท้ายบท**
2. ออกแบบ ER Diagram ของโครงงาน เช่า-ฉัน-สิ ในภาคผนวก

บทที่ 4 แบบจำลองเชิงสัมพันธ์ (Relational Model)

วัตถุประสงค์

- สามารถอธิบายถึงวิวัฒนาการของแบบจำลองเชิงสัมพันธ์
- สามารถอธิบายแนวคิดของแบบจำลองเชิงสัมพันธ์ได้
- สามารถอธิบายการแทนข้อมูลในแบบจำลองเชิงสัมพันธ์ได้
- สามารถอธิบายการกำหนดเงื่อนไขบังคับสำหรับบูรณภาพของข้อมูลแบบต่างๆ ได้
- สามารถสร้างแบบจำลองเชิงสัมพันธ์จาก ER model ได้

คำสำคัญ: กฎ 12 ข้อของ E. F. Codd (E. F. Codd's 12 rules); รีเลชัน (relation); สคีมา (schema); กรณีตัวอย่าง (instance); ทูเพิล (tuple); ฟิลด์ (field); โดเมน (domain); ดีกรี (degree); คาร์ดินัลลิตี (cardinality); เงื่อนไขบังคับบูรณภาพ (integrity constraints); เงื่อนไขบังคับโดเมน (domain constraints); เงื่อนไขบังคับคีย์ (key constraints); คีย์ (key); คีย์หลัก (primary key); คีย์คู่แข่ง (candidate key); คีย์ประกอบ (composite key); คีย์ภายนอก (foreign key); เงื่อนไขบังคับคีย์ภายนอก (foreign key constraints); บูรณภาพเชิงอ้างอิง (referential integrity); เงื่อนไขบังคับทั่วไป (general constraints); การสร้างแบบจำลองเชิงสัมพันธ์จาก ER model (ER model to relational model translation); วิว (view)

4.1 บทนำ

แบบจำลองเชิงสัมพันธ์เป็นพื้นฐานของระบบจัดการฐานข้อมูลเชิงสัมพันธ์ ในขั้นตอนการออกแบบฐานข้อมูลเพื่อนำมาใช้ประโยชน์นั้น หลังจากการออกแบบระดับแนวคิดด้วย ER model แล้ว เราจำเป็นต้องเลือกแบบจำลองข้อมูลเพื่อแทนข้อมูลที่จะนำมาบริหารจัดการ ในปัจจุบันนั้นแบบจำลองเชิงข้อมูลเชิงสัมพันธ์ได้รับความนิยมสูงสุด ระบบจัดการฐานข้อมูลเชิงสัมพันธ์มีการนำมาใช้ในเชิงธุรกิจและในแวดวงอื่นๆ ซึ่งก่อให้เกิดประโยชน์อย่างมหาศาล ในบทนี้เราจะได้กล่าวถึงแบบจำลองข้อมูลเชิงสัมพันธ์และการออกแบบฐานข้อมูลระดับตรรกะโดยใช้แบบจำลองเชิงสัมพันธ์

แนวคิดทั่วไปของแบบจำลองเชิงสัมพันธ์ได้แต่วิวัฒนาการและกฎที่ระบบฐานข้อมูลเชิงสัมพันธ์ต้องรองรับได้อธิบายในหัวข้อที่ 4.2.1 และ 4.2.2 ตามลำดับ ในหัวข้อที่ 4.2.3 อธิบายคำศัพท์พื้นฐานของแบบจำลองเชิงสัมพันธ์ หัวข้อที่ 4.3 อธิบายถึงการกำหนดเงื่อนไขบังคับต่างๆ ในแบบจำลองเชิงสัมพันธ์เพื่อบูรณภาพของข้อมูล จากนั้นในหัวข้อที่ 4.4 ได้อธิบายถึงวิวและความเกี่ยวข้องกับฐานข้อมูลระดับภายนอก

แบบจำลองเชิงสัมพันธ์นี้เป็นแบบจำลองข้อมูลระดับเชิงตรรกะของฐานข้อมูล เราสามารถออกแบบฐานข้อมูลโดยใช้แบบจำลองเชิงความหมาย (semantic model) ด้วย ER model ซึ่งไม่ยึดติดกับแบบจำลองข้อมูลใดๆ ER model ที่ได้มานี้หาจะนำมาพัฒนาเป็นระบบฐานข้อมูลจะต้องได้รับการแปลงให้เป็นฐานข้อมูลระดับตรรกะเสียก่อน ในหัวข้อ 4.5 เราได้อธิบายวิธีการในการแปลง ER model ให้อยู่ในรูปของแบบจำลองเชิงสัมพันธ์ จากนั้นหัวข้อที่ 4.6 ได้ยกตัวอย่างกรณีศึกษาของการสร้างแบบจำลองเชิงสัมพันธ์จาก ER model ที่ได้ออกแบบไว้จากความต้องการของผู้ใช้ในบทที่แล้ว

4.2 แนวคิดทั่วไปเกี่ยวกับแบบจำลองเชิงสัมพันธ์

4.2.1 วิวัฒนาการของแบบจำลองเชิงสัมพันธ์

E. F. Codd เป็นผู้นำเสนอแบบจำลองข้อมูลเชิงสัมพันธ์เป็นคนแรกในปี ค.ศ. 1970 ในขณะนั้นระบบจัดการฐานข้อมูลส่วนใหญ่อยู่บนพื้นฐานข้องแบบจำลองข้อมูลเชิงลำดับชั้นและแบบจำลองข้อมูลเชิงเครือข่าย การเสนอรูปแบบของแบบจำลองเชิงสัมพันธ์นั้นนำมาซึ่งการปฏิรูปครั้งใหญ่ในวงการฐานข้อมูล รูปแบบของแบบจำลองข้อมูลที่ชัดเจน สอดคล้องกับการใช้งาน และมีทฤษฎีมารองรับนั้นก่อให้เกิดการศึกษา พัฒนาและใช้งานระบบจัดการฐานข้อมูลอย่างแพร่หลาย

ในทศวรรษที่ 1970 ห้องปฏิบัติการวิจัยของบริษัท IBM ในเมือง San José มลรัฐแคลิฟอร์เนีย ได้พัฒนาระบบจัดการฐานข้อมูลบนพื้นฐานของแบบจำลองข้อมูลเชิงสัมพันธ์ โดยให้ชื่อระบบว่า IBM System R ซึ่งระบบดังกล่าวยังเป็นต้นแบบของการประยุกต์ระบบที่รองรับข้อกำหนดที่จำเป็นอื่นๆ ของระบบจัดการฐานข้อมูล เช่น การจัดการธุรกรรม การรองรับการทำงานพร้อมกัน การใช้งานข้อคำถามที่มีโครงสร้าง การปรับปรุงประสิทธิภาพของการแสดงข้อมูลจากข้อคำถาม ความมั่นคงปลอดภัยของข้อมูล บูรณภาพของข้อมูลหรือความถูกต้องของข้อมูล และส่วนติดต่อ

กับผู้ใช้ เป็นต้น System R นี้ยังเป็นโครงการที่เป็นต้นกำเนิดของภาษาที่ใช้ในการจัดการฐานข้อมูลเชิงสัมพันธ์ SQL ที่เป็นมาตรฐานในปัจจุบัน ตลอดจนเป็นต้นแบบของระบบจัดการฐานข้อมูลเชิงพาณิชย์ เช่น IBM DB/2 และ Oracle เป็นต้น

INGRES (Interactive Graphics Retrieval System) เป็นโครงการพัฒนาต้นแบบระบบจัดการฐานข้อมูลสำหรับฐานข้อมูลเชิงสัมพันธ์ที่พัฒนาโดย University of California at Berkeley โครงการดังกล่าวเกิดขึ้นในเวลาไล่เลี่ยกันกับ IBM System R ทั้งสองโครงการมีวัตถุประสงค์คล้ายคลึงกันในการพัฒนาระบบจัดการฐานข้อมูลเชิงสัมพันธ์และการเพิ่มความสามารถที่ทำให้ระบบทำงานได้อย่างมีประสิทธิภาพ จากความสำเร็จของต้นแบบของระบบจัดการฐานข้อมูลเชิงสัมพันธ์ต่างๆ จึงมีการสร้างระบบฐานข้อมูลเชิงสัมพันธ์เชิงพาณิชย์ขึ้นในช่วงปลายทศวรรษที่ 1970 เป็นต้นมา และยังคงมีใช้แพร่หลายในปัจจุบัน ระบบจัดการฐานข้อมูลที่ได้รับความนิยมได้แก่ Microsoft Access, Microsoft SQL Server, Microsoft Visual FoxPro, Oracle, IBM DB/2, MySQL ฯลฯ

4.2.2 กฎ 12 ข้อของ E. F. Codd (E. F. Codd' 12 Rules)

กฎ 12 ข้อของ E. F. Codd คือ กฎที่บัญญัติไว้สำหรับกำหนดให้ระบบจัดการฐานข้อมูลเชิงสัมพันธ์รองรับ

ในปี ค.ศ. 1985 Edgar Frank Codd ผู้ให้กำเนิดแบบจำลองข้อมูลเชิงสัมพันธ์ได้วางกฎไว้สำหรับเป็นบรรทัดฐานของการทำงานของระบบจัดการฐานข้อมูลเชิงสัมพันธ์ต่างๆ เพื่อให้เป็นระบบจัดการฐานข้อมูลที่รองรับแบบจำลองเชิงสัมพันธ์โดยสมบูรณ์ โดยกำหนดเป็นกฎ 12 ข้อ (E. F. Codd's 12 rules) ซึ่งโดยแท้จริงแล้วมี 13 ข้อ ดังนี้

กฎข้อที่ 0 กฎพื้นฐาน (Foundation rule)

ระบบฐานข้อมูลเชิงสัมพันธ์ต้องมีความสามารถในจัดการฐานข้อมูลทั้งหมดโดยใช้ความสามารถเชิงสัมพันธ์ กล่าวคือการจัดการข้อมูลใดๆ จะอ้างอิงถึงทฤษฎีเชิงสัมพันธ์เท่านั้น

กฎข้อที่ 1 กฎสารสนเทศ (Information rule)

ข้อมูลทั้งหมดในฐานข้อมูลเชิงสัมพันธ์จะต้องปรากฏอยู่อย่างชัดแจ้ง เช่นข้อมูลในฐานข้อมูลแต่ละระเบียนที่จะต้องปรากฏอยู่ในตาราง ตลอดจนระบุด้วยชื่อของตาราง และชื่อของคอลัมน์ และต้องจัดเก็บโดยแทนค่าแบบใดแบบหนึ่งเท่านั้น

กฎข้อที่ 2 กฎการรับประกันการเข้าถึงข้อมูล (Guaranteed access rule)

ข้อมูลทุกๆ ค่าที่เก็บไว้ในฐานข้อมูลจะต้องสามารถเข้าถึงได้โดยการระบุชื่อตาราง คีย์หลัก และชื่อคอลัมน์

กฎข้อที่ 3 การรองรับค่าว่าง (Systematic null value support)

ระบบจัดการฐานข้อมูลต้องรองรับค่าว่างหรือ null value โดยค่าดังกล่าวต้องแสดงว่าข้อมูลยังคงว่างเปล่าอยู่ ต่างจากค่าโดยปริยาย เช่น ต่างจากค่า 0 และไม่ขึ้นกับโดเมนของแอทริบิวต์ใดๆ

กฎข้อที่ 4 โครงสร้างของรีเลชันสามารถเรียกดูได้ (Dynamic On-line Catalog Based on the Relational Model)

โครงสร้างต่างๆ ของฐานข้อมูลที่นิยามหรือกำหนดไว้เช่น รายการตารางต่างๆ ในฐานข้อมูล รายการคอลัมน์ในตาราง จะต้องสามารถเรียกแสดงโครงสร้างและจัดการแก้ไขได้โดยภาษาที่ใดๆ ที่มีโครงสร้าง ซึ่งโครงสร้างเหล่านี้จะถูกอ้างอิงโดยข้อมูลฐานข้อมูล ตลอดจนผู้ใช้งานที่ใช้งานข้อมูลจะอาศัยโครงสร้างเดียวกันนี้ในการใช้งานฐานข้อมูล

กฎข้อที่ 5 กฎการมีภาษาที่สนับสนุนอย่างเต็มรูปแบบ (Comprehensive data sublanguage rule)

จะต้องมีภาษาอย่างน้อยหนึ่งภาษาที่ออกแบบมาอย่างสมบูรณ์และสามารถจัดการฐานข้อมูลเชิงสัมพันธ์ได้เต็มความสามารถของแบบจำลอง (อาจไม่ใช่ SQL ก็ได้) โดยความสามารถที่ภาษาจะต้องรองรับได้แก่

1) การนิยามข้อมูล
2) การนิยามวิว
3) การแก้ไขข้อมูล ได้แก่การเพิ่ม ลบ และแก้ไข
4) การกำหนดกฎบูรณภาพหรือกฎเพื่อคงความถูกต้องของข้อมูล
5) การกำหนดสิทธิในการเข้าใช้ฐานข้อมูล
6) การจัดการธุรกรรม

กฎข้อที่ 6 กฎการแก้ไขข้อมูลผ่านทางวิว (View updating rule)

วิวทุกวิวที่อนุญาตให้แก้ไขข้อมูลได้ จะต้องสามารถแก้ไขข้อมูลในฐานข้อมูลผ่านทางวิวได้ กล่าวคือโดยปกติวิวจะเกิดจากรีเลชันหลักในฐานข้อมูล โอยอาจเป็นหลายๆ รีเลชันมารวมกัน การแก้ไขข้อมูลในวิวใดๆ ข้อมูลที่ได้รับการแก้ไข ข้อมูลในรีเลชันหลักที่สัมพันธ์กับข้อมูลในวิวจะต้องถูกปรับปรุงให้ตรงกัน

กฎข้อที่ 7 มีความสามารถในการเพิ่ม ลบ และแก้ไขข้อมูล (High-level insertion, update, and deletion)

ไม่ใช่แค่การแสดงข้อมูลเท่านั้นที่ระบบจัดการฐานข้อมูลรองรับ ระบบจัดการฐานข้อมูลต้องรองรับการเพิ่ม การแก้ไขหรือปรับปรุง และการลบข้อมูลได้ ซึ่งกฎข้อนี้กล่าวถึงความสามารถในการจัดการข้อมูลทีละหลายๆ กลุ่มของข้อมูลเช่น หลายๆ ระเบียน

กฎข้อที่ 8 มีความเป็นอิสระของข้อมูลระดับกายภาพ (Physical data independence)

โปรแกรมประยุกต์หรือโปรแกรมอื่นใดที่ใช้งานระบบจัดการฐานข้อมูลจะต้องไม่ได้รับผลกระทบกรณีมีการปรับเปลี่ยนข้อกำหนดด้านกายภาพ กล่าวคือโปรแกรมที่เรียกใช้งานระบบจัดการฐานข้อมูลจะต้องยังคงทำงานได้

ดังเดิมในกรณีที่โครงสร้างภายในของระบบจัดการฐานข้อมูลในระดับการจัดเก็บข้อมูลในอุปกรณ์หน่วยความจำสำรอง (ฮาร์ดดิสก์) เช่น อาจมีการเปลี่ยนลำดับของไฟล์ใหม่ มีการสร้างไฟล์ดัชนีใหม่ เป็นต้น

กฎข้อที่ 9 มีความเป็นอิสระของข้อมูลระดับตรรกะ (Logical data independence)

โปรแกรมประยุกต์หรือโปรแกรมอื่นใดที่ใช้งานระบบจัดการฐานข้อมูลจะต้องไม่ได้รับผลกระทบกรณีมีการปรับเปลี่ยนข้อกำหนดด้านตรรกะ กล่าวคือโปรแกรมที่เรียกใช้งานระบบจัดการฐานข้อมูลจะต้องยังคงทำงานได้ดังเดิม ในกรณีที่มีการเปลี่ยนโครงสร้างตารางข้อมูล อย่างไรก็ตาม หากโครงสร้างข้อมูลที่ปรับเปลี่ยนเกี่ยวข้องกับโปรแกรมประยุกต์นั้นๆ โดยตรงเช่นจะต้องมีการจัดเก็บข้อมูลเพิ่มอีก 1 คอลัมน์ ผ่านทางหน้าจอโปรแกรม เรายังคงต้องแก้ไขโปรแกรมประยุกต์ ให้รองรับการกรอกข้อมูลนั้น

กฎข้อที่ 10 มีความเป็นอิสระของบูรณภาพ (Integrity independence)

ภาษาฐานข้อมูลต้องรองรับการกำหนดข้อกำหนดหรือกฎต่างๆ ที่บังคับให้ข้อมูลมีความถูกต้องตามข้อกำหนดทางธุรกิจหรือที่เรียกว่าบูรณภาพของข้อมูล ข้อกำหนดเหล่านี้ต้องถูกจัดเก็บอยู่ในระบบจัดการฐานข้อมูลและไม่สามารถที่จะละเมิดข้อกำหนดนี้ได้

กฎข้อที่ 11 มีอิสระในการกระจาย (Distribution independence)

โปรแกรมประยุกต์หรือโปรแกรมอื่นใดที่ใช้งานระบบจัดการฐานข้อมูลจะต้องไม่ได้รับผลกระทบ สามารถใช้งานได้อย่างเป็นปกติ ในกรณีที่มีการกระจายของข้อมูล ซึ่งในที่นี้หมายถึงการที่ข้อมูลกระจายไปอยู่ในอุปกรณ์จัดเก็บหลายๆ แหล่ง หรือแม้แต่กลับมารวมกันในแหล่งเดียว

กฎข้อที่ 12 ไม่อนุญาตให้ภาษาในระดับต่ำกว่าเลี่ยงกฎบูรณภาพ (Nonsubversion rule)

ภาษาในระดับต่ำกว่า ในที่นี้หมายถึงภาษาที่จัดการข้อมูลครั้งละข้อมูล เช่น ครั้งละระเบียน ครั้งละคอลัมน์ในระเบียน จะต้องไม่สามารถเลี่ยงกฎต่างๆ ที่ตั้งไว้เพื่อบูรณภาพของข้อมูลได้

4.2.3 ข้อดีของระบบจัดการฐานข้อมูลเชิงสัมพันธ์

4.2.4 คำศัพท์ที่เกี่ยวข้องกับแบบจำลองเชิงสัมพันธ์

E. F. Codd ผู้เสนอแบบจำลองเชิงสัมพันธ์อาศัยทฤษฎีทางคณิตศาสตร์ในการนำเสนอแบบจำลองเชิงสัมพันธ์ ซึ่งรีเลชันนั้นเปรียบได้กับตารางข้อมูลนั่นเอง ในหัวข้อนี้อธิบายถึงคำศัพท์ต่างๆ ที่เกี่ยวข้องกับแบบจำลองเชิงสัมพันธ์

รีเลชัน (Relation)

รีเลชัน คือ ตารางข้อมูลที่ประกอบไปด้วยคอลัมน์/สดมภ์ (Columns) และแถว (Rows)

ในระบบจัดการฐานข้อมูลเชิงสัมพันธ์นั้น รีเลชันคือตารางข้อมูลนั่นเอง โดยแท้ที่จริงแล้วรีเลชันหมายถึงความสัมพันธ์ของกลุ่มข้อมูล ซึ่งในที่นี่หมายถึงแอทริบิวต์ ซึ่งเป็นกลุ่มข้อมูลที่อธิบายเอนทิตีหรือความสัมพันธ์ระหว่างเอนทิตี กลุ่มข้อมูลเหล่านี้มีความสัมพันธ์กันในกลุ่ม การนำแอทริบิวต์ที่เกี่ยวข้องมารวมกันตรงกับลักษณะการใช้งานในรูปแบบของตารางข้อมูล นอกจากรีเลชันซึ่งเป็นพื้นฐานหลักของระบบฐานข้อมูลเชิงสัมพันธ์แล้ว ยังมีส่วนประกอบอื่นๆ อีกดังต่อไปนี้

แอทริบิวต์ (Attribute)

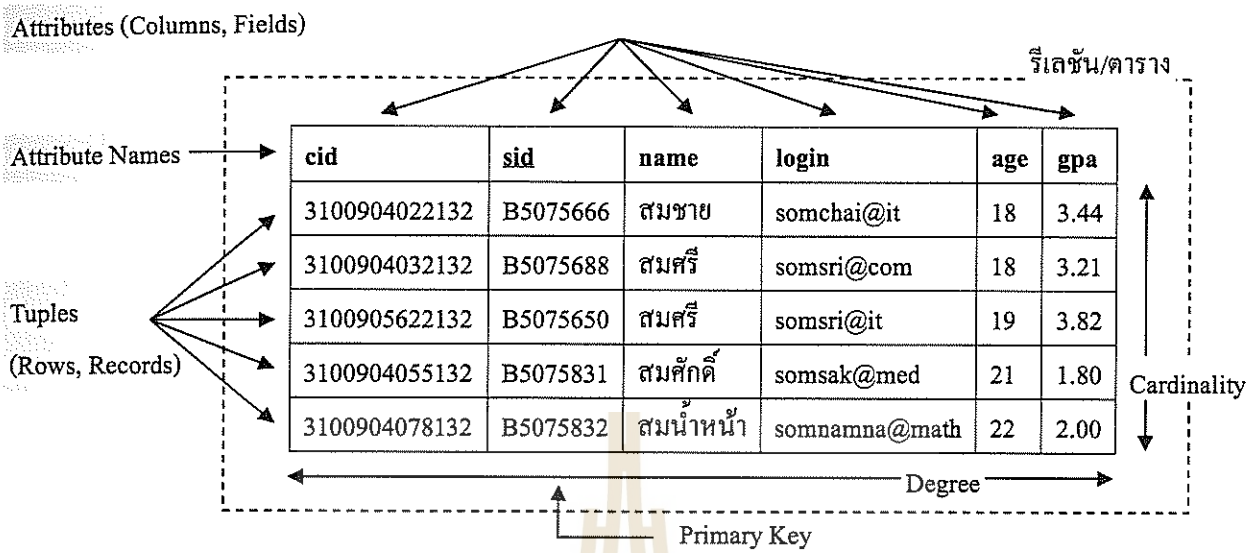
แอทริบิวต์ คือ คอลัมน์ของข้อมูลในตาราง

แอทริบิวต์ หรือ คอลัมน์ (column) หรือ ฟิลด์ (field) คือคุณสมบัติต่างๆ ที่อธิบายข้อมูลของเอนทิตีหรือความสัมพันธ์ระหว่างเอนทิตี สำหรับในระบบฐานข้อมูลเชิงสัมพันธ์ แอทริบิวต์ปรากฏอยู่ในรูปแบบของคอลัมน์ข้อมูลของตาราง ขอบเขตของค่าที่เป็นไปได้ของแอทริบิวต์เรียกว่า โดเมน **(domain)** การใช้งานฐานข้อมูลโดยปกติเรามีความจำเป็นต้องกำหนดขอบเขตของข้อมูลในแต่ละแอทริบิวต์เพื่อให้ข้อมูลมีความถูกต้อง เช่น เพศของคนแต่ละคนมีได้ 2 ค่าคือ ชายและหญิง เราสามารถกำหนดเงื่อนไขบังคับโดเมน **(domain constraint)** ในระบบจัดการฐานข้อมูลเพื่อบังคับให้ข้อมูลที่บรรจุอยู่ในฐานข้อมูลมีความถูกต้องได้

ทูเพิล (Tuple)

ทูเพิล คือ แถวของข้อมูลในตาราง

ทูเพิล หรือ แถว (row) หรือ เรคคอร์ด/ระเบียน (record) คือชุดของข้อมูล 1 ข้อมูลของรีเลชันนั้นๆ มีลักษณะเป็นข้อมูลที่อยู่ในแถวๆ หนึ่ง ซึ่งได้แก่ข้อมูลของเอนทิตีหรือความสัมพันธ์ระหว่างเอนทิตีนั่นเอง กลุ่มของแอทริบิวต์ทั้งหมดเป็นแม่แบบของข้อมูลแต่ละระเบียน เราจึงเรียกกลุ่มของแอทริบิวต์ที่เป็นตัวกำหนดโครงสร้างของรีเลชันว่า **สคีมา (schema)** และสำหรับข้อมูลที่เป็นไปตามโครงสร้างของสคีมาในรีเลชันเรียกว่า **กรณีตัวอย่าง (instance)** ของสคีมานั้นๆ ดังเช่นตัวอย่างรีเลชันของนักศึกษา (Student) ต่อไปนี้



cid	sid	name	login	age	gpa
3100904022132	B5075666	สมชาย	somchai@it	18	3.44
3100904032132	B5075688	สมศรี	somsri@com	18	3.21
3100905622132	B5075650	สมศรี	somsri@it	19	3.82
3100904055132	B5075831	สมศักดิ์	somsak@med	21	1.80
3100904078132	B5075832	สมน้ำหน้า	somnamna@math	22	2.00

ตัวอย่างของรีเลชันนักศึกษาประกอบไปด้วยแอทริบิวต์หรือคอลัมน์ 5 คอลัมน์ ได้แก่ sid, name, login, age และ gpa ซึ่งทั้ง 5 คอลัมน์นี้สามารถเขียนให้อยู่ในรูปแม่แบบ/โครงสร้าง หรือสคีมาของรีเลชันได้ดังนี้

Student (*id:* string, *sid:* string, *name:* string, *login:* string, *age:* integer, *gpa:* real)

แอทริบิวต์แต่ละแอทริบิวต์ในสคีมาถูกกำหนดไว้ด้วยโดเมน เช่น sid มีโดเมนเป็น string คือสายอักขระ และ age เป็น integer หมายถึงต้องเป็นจำนวนเต็ม เป็นต้น กรณีตัวอย่าง (instance) ของสคีมานักศึกษาคือข้อมูลในตาราง ทูเพิลของรีเลชันคือแถวแต่ละแถวในตาราง นอกจากนี้ยังมีศัพท์เฉพาะเพิ่มเติมสำหรับรีเลชันดังนี้

คีย์หลัก (Primary Key)

คีย์หลัก คือ แอทริบิวต์หรือกลุ่มของแอทริบิวต์ที่สามารถระบุทูเพิลได้อย่างเฉพาะเจาะจง

ข้อมูลแต่ละเรคคอร์ดจะต้องสามารถเข้าถึงได้อย่างเฉพาะเจาะจงโดยจากการระบุค่าแอทริบิวต์ใดแอทริบิวต์หนึ่งหรือกลุ่มแอทริบิวต์ก็ได้ จากตัวอย่างเรากำหดให้ sid ซึ่งเป็นรหัสนักศึกษาเป็นคีย์หลัก ในสคีมาเราใช้การขีดเส้นใต้เป็นสัญลักษณ์ของคีย์หลัก อย่างไรก็ตามเราอาจใช้ cid ซึ่งเป็นรหัสประจำตัวประชาชนเป็นคีย์หลักก็ได้ โดยรายละเอียดเกี่ยวกับคีย์ได้อธิบายไว้ในหัวข้อ 4.3.2

ดีกรี (Degree)

ดีกรี คือ จำนวนแอทริบิวต์ในรีเลชัน

จากตัวอย่างรีเลชันนักศึกษามีค่าของดีกรีเป็น 6

คาร์ดินัลลิตี (Cardinality)

คาร์ดินัลลิตี คือ จำนวนของแถวในรีเลชัน

จากตัวอย่างรีเลชันนักศึกษามีคาร์ดินัลลิตีเท่ากับ 5

สรุปศัพท์ที่เกี่ยวข้องกับแบบจำลองเชิงสัมพันธ์ได้ดังนี้

ศัพท์เฉพาะ	ศัพท์ทั่วไป
รีเลชัน (relation)	ตาราง (table)
ทูเพิล (tuple)	แถว (row) หรือ เรคคอร์ด/ระเบียน (record)
แอทริบิวท์ (attribute)	คอลัมน์/สดมภ์ (column) หรือ ฟิลด์ (field)
คาร์ดินัลลิตี (cardinality)	จำนวนแถว (number of rows)
ดีกรี (degree)	จำนวนแอทริบิวต์ (number of attributes)
คีย์หลัก (primary key)	ค่าเอกลักษณ์ (unique identifier)
โดเมน (domain)	ขอบเขตของข้อมูล (pool of legal values)

ฐานข้อมูลเชิงสัมพันธ์ (relational database) คือกลุ่มของรีเลชัน สคีมาของฐานข้อมูลเชิงสัมพันธ์ (relational database schema) คือกลุ่มของสคีมาของรีเลชันต่างๆ ในฐานข้อมูล นอกจากสคีมาที่กำหนดโครงสร้างและโดเมนของข้อมูลแล้ว การทำให้ข้อมูลมีความถูกต้องตรงตามข้อกำหนดทางธุรกิจยังสามารถกำหนดไว้ได้ในสคีมาของฐานข้อมูลเชิงสัมพันธ์ ดังจะได้กล่าวถึงในหัวข้อถัดไป

4.3 เงื่อนไขบังคับบูรณภาพ (Integrity Constraints)

เงื่อนไขบังคับบูรณภาพ คือ เงื่อนไขที่กำหนดไว้ในฐานข้อมูลเพื่อให้ข้อมูลในฐานข้อมูลมีความถูกต้องตรงตามลักษณะการใช้งาน

ฐานข้อมูลคือตัวข้อมูลที่เก็บไว้ในอุปกรณ์จัดเก็บเท่านั้น การเรียกใช้และจัดการข้อมูลจำเป็นต้องอาศัยระบบจัดการฐานข้อมูลในการเข้าถึงข้อมูล การจะกำหนดเงื่อนไขให้ข้อมูลที่จะบรรจุลงในฐานข้อมูลมีความถูกต้องเราจำเป็นต้องอาศัยระบบจัดการฐานข้อมูลเช่นเดียวกัน การกำหนดเงื่อนไขบังคับบูรณภาพ (integrity constraint) นั้นเป็นการกำหนดเงื่อนไขบางประการให้ข้อมูลที่จะบรรจุอยู่ในฐานข้อมูลมีความถูกต้องตรงตามข้อกำหนดการใช้งาน การกำหนดเงื่อนไขบังคับต่างๆ สามารถกระทำได้โดยการกำหนดเงื่อนไขบังคับไว้ในสคีมาของฐานข้อมูล ระบบจัดการฐานข้อมูลจะอาศัยเงื่อนไขบังคับที่ระบุไว้ในฐานข้อมูลนี้ในการตรวจสอบข้อมูลต่างๆ ก่อนทำการบรรจุข้อมูลลงในฐานข้อมูลว่าตรงตามเงื่อนไขหรือไม่ หากไม่ตรงจะไม่สามารถบรรจุข้อมูลลงไปในฐานข้อมูลได้และจำต้องมีการจัดการกับความผิดพลาดที่เกิดขึ้นนี้ต่อไป

เงื่อนไขบังคับเพื่อทำให้ข้อมูลมีบูรณภาพที่สามารถกำหนดได้ในฐานข้อมูลเชิงสัมพันธ์มีหลายชนิด เงื่อนไขบังคับโดเมน และเงื่อนไขบังคับคีย์เป็นตัวอย่างของเงื่อนไขบังคับบูรณภาพที่ได้กล่าวถึงไปแล้ว หัวข้อต่อไปนี้จะอธิบายถึงเงื่อนไขบังคับอื่นๆ ที่มีความจำเป็นโดยทั่วไปในการใช้งานระบบจัดการฐานข้อมูลเชิงสัมพันธ์

4.3.1 เงื่อนไขบังคับคีย์ (Key Constraints)

เงื่อนไขบังคับคีย์ คือ การกำหนดให้แอทริบิวต์หรือกลุ่มของแอทริบิวต์ใดเป็นคีย์ของรีเลชันสำหรับการระบุระเบียนใดระเบียนหนึ่งได้อย่างเฉพาะเจาะจง

คีย์คือค่าของแอทริบิวต์ หรือชุดของแอทริบิวต์ที่มีจำนวนน้อยที่สุดที่สามารถระบุข้อมูลระเบียนใดระเบียนหนึ่งได้ ยกตัวอย่างเช่น ในรีเลชันนักศึกษา รหัสนักศึกษาสามารถระบุระเบียนของนักศึกษาคนใดคนหนึ่งได้ ในขณะที่เราไม่สามารถใช้แอทริบิวต์ชื่อของนักศึกษา หรืออายุเป็นคีย์ได้เนื่องจากมีระเบียนที่ชื่อของนักศึกษาซ้ำกัน การระบุเพียงชื่อ "สมศรี" ไม่สามารถระบุได้ว่าเป็นสมศรีที่มีรหัสนักศึกษา "B5075688" หรือ "B5075650" ดังนั้นข้อมูลในแอทริบิวต์ดังกล่าวจึงห้ามซ้ำกัน เราสามารถกำหนดเงื่อนไขบังคับเพื่อไม่ให้แอทริบิวต์ใด หรือแอทริบิวต์กลุ่มใดมีค่าไม่ซ้ำกัน และประกาศว่าแอทริบิวต์ดังกล่าวเป็นคีย์หลักของรีเลชันได้โดยการกำหนดเงื่อนไขบังคับคีย์ (key constraint) ไว้ในสคีมาของฐานข้อมูล คีย์สามารถแบ่งออกได้เป็นหลายประเภทตามลักษณะต่างๆ ของคีย์ดังนี้

1) **Simple key** หรือ **คีย์อย่างง่าย** หมายถึงคีย์ที่ประกอบด้วยแอทริบิวต์เดียว จากตัวอย่างรีเลชันนักศึกษา รหัสนักศึกษาเป็นคีย์ของรีเลชันที่เป็นแอทริบิวต์เดียวเท่านั้น โดยทั่วไปแล้วคีย์หลักของตารางมีลักษณะเป็น simple key

2) **Composite key (compound key หรือ concatenated key)** หรือ **คีย์ประกอบ** หมายถึงคีย์ที่ประกอบด้วยแอทริบิวต์ มากกว่า 1 แอทริบิวต์ เรากล่าวว่านอกจากคีย์จะหมายถึงแอทริบิวต์หนึ่งแอทริบิวต์ที่สามารถระบุระเบียนได้อย่างเฉพาะเจาะจงแล้ว คีย์ยังสามารถเป็นชุดของแอทริบิวต์ได้ด้วย โดยเป็นชุดที่มีจำนวนแอทริบิวต์ที่น้อยที่สุดที่สามารถระบุระเบียนได้ เช่น ในกรณีตัวอย่างข้อมูลของรีเลชันนักศึกษา เราอาจกำหนดให้ชื่อและอายุ {*name, age*} เป็นชุดหรือเซตของแอทริบิวต์ที่ใช้เป็นคีย์สำหรับรีเลชันได้ ในกรณีนี้ {*name, age*} เป็นคีย์ชนิดคีย์ประกอบ พิจารณาข้อมูลในตารางนักศึกษาพบว่า ถึงแม้จะมีนักศึกษาที่ชื่อ (*name*) สมศรี อยู่ถึง 2 คน แต่หากเราระบุอายุไปด้วย เราจะสามารถระบุได้ว่าเป็นสมศรีคนใดเนื่องจากสมศรีคนหนึ่งมีอายุ 18 และอีกคนหนึ่งมีอายุ 19

3) **Candidate key** หรือ **คีย์คู่แข่ง** หมายถึงคีย์ที่สามารถจะเป็นคีย์หลักของของรีเลชันได้ เช่น ในรีเลชันนักศึกษา เราอาจเลือกใช้รหัสประจำตัวประชาชน (*cid*) เป็นคีย์หลักของรีเลชันได้ เนื่องจากรหัสประจำตัวประชาชนนั้นไม่ซ้ำกัน นอกจากนี้แอทริบิวต์หรือเซตของแอทริบิวต์ที่สามารถเป็นคีย์ของรีเลชันอาจได้แก่ {*login*} หรือ {*name, age*} หรือ {gpa} ฯลฯ ซึ่งเป็นคีย์ที่สามารถระบุระเบียนใดๆ ได้อย่างเฉพาะเจาะจง

4) **Primary key** หรือ **คีย์หลัก** หมายถึงคีย์คู่แข่งตัวหนึ่งที่ถูกเลือกขึ้นมาเป็นคีย์หลัก เราจะใช้สัญลักษณ์<u>ขีดเส้นใต้</u>กำกับไว้ใต้แอทริบิวต์ตัวนั้น ในการเขียนสคีมาของรีเลชัน จากตัวอย่างของคีย์คู่แข่ง เซตของแอทริบิวต์ที่สามารถเป็นคีย์หลักได้แก่ {*sid*}, {*cid*}, {*login*}, {*name, age*} หรือ {gpa} ฯลฯ อย่างไรก็ตามคีย์หลักนั้นควรเป็นคีย์ที่สามารถใช้งานได้อย่างสมเหตุสมผลและเหมาะสม พิจารณาคีย์ {*name, age*} นั้นถึงแม้จะเป็นคีย์ได้ แต่ก็เป็นได้เฉพาะในกรณีตัวอย่างของข้อมูลที่แสดง บ่อยครั้งที่นักศึกษาที่มีชื่อ

เหมือนกันจะมีอายุเท่ากัน หากเป็นเช่นนี้เราจะไม่สามารถใช้เซตแอทริบิวต์ {*name, age*} เป็นคีย์ได้อีกต่อไป เช่นเดียวกันกับ {*gpa*} ที่สามารถซ้ำกันได้ เราไม่นิยมนำข้อมูลที่อาจเปลี่ยนแปลงได้ง่ายมาใช้เป็นคีย์หลัก เช่น {*login*} สามารถเปลี่ยนแปลงได้ในกรณีที่นักศึกษาย้ายสาขาหรือเปลี่ยนชื่อ หรือแม้แต่เปลี่ยน login สำหรับรีเลชันนักศึกษานั้น คีย์ที่เหมาะสมสำหรับที่จะเป็นคีย์หลักได้แก่รหัสนักศึกษา {*sid*} รหัสประจำตัวประชาชนมีความยาวเกินไปในที่นี้ในการนำมาเป็นคีย์หลัก

5) **Secondary key (alternate key)** หรือ **คีย์รอง** หมายถึงคีย์คู่แข่ง ที่ไม่ได้ถูกเลือกให้เป็นคีย์หลัก
6) **Superkey** คือเซ็ตของแอทริบิวต์ที่มีคีย์เป็นสมาชิก เช่น {*sid, name*} เซ็ตของแอทริบิวต์ดังกล่าวสามารถระบุระเบียนได้อย่างเฉพาะเจาะจงเนื่องจากมีคีย์ sid อยู่แล้ว อย่างไรก็ตาม name เป็นแอทริบิวต์ที่ไม่มีความจำเป็นในการกำหนดเป็นคีย์ เราไม่ค่อยได้ใช้ประโยชน์ของ Superkey
7) **Foreign key** หรือ **คีย์ภายนอก** หมายถึงแอทริบิวต์ในรีเลชันหนึ่งที่อ้างอิงมาจากคีย์หลักของรีเลชันอื่น ซึ่งจะได้อธิบายในหัวข้อถัดไป

4.3.2 เงื่อนไขบังคับคีย์ภายนอก (Foreign Key Constraints)

เงื่อนไขบังคับคีย์บังคับภายนอก คือ ข้อกำหนดของข้อมูลที่อ้างอิงกันระหว่างรีเลชันเพื่อให้ข้อมูลมีความสอดคล้องกัน

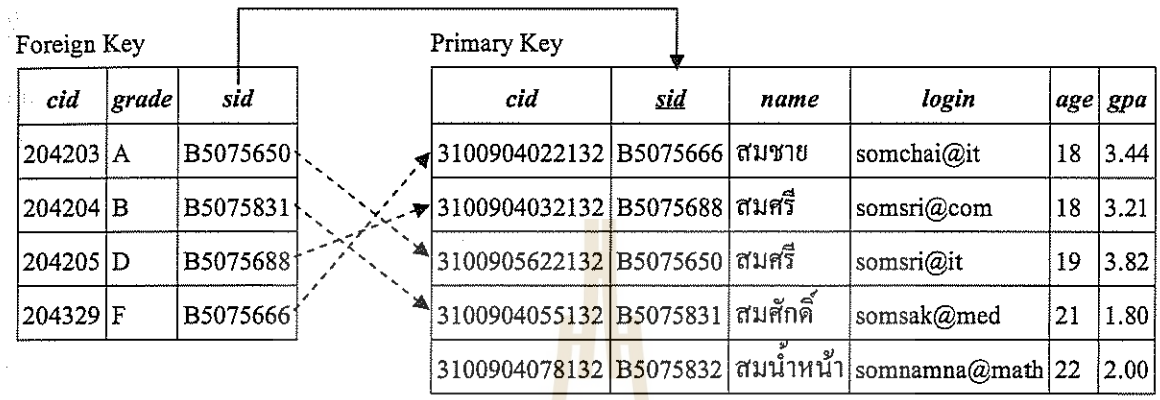
ข้อมูลที่บรรจุอยู่ในรีเลชันหนึ่งมักอ้างถึงข้อมูลที่บรรจุอยู่ในอีกรีเลชันหนึ่ง หากข้อมูลที่บรรจุอยู่ในรีเลชันหนึ่ง หากข้อมูลนั้นๆ ในรีเลชันหนึ่งจะได้รับการแก้ไข เรามีความจำเป็นต้องตรวจสอบข้อมูลในอีกรีเลชันหนึ่งที่อ้างอิงถึงกันซึ่งต้องได้รับการแก้ไขหรือดำเนินการอย่างหนึ่งอย่างใดให้ข้อมูลที่อ้างถึงกันนี้มีความสอดคล้องกัน เงื่อนไขบังคับตรวจสอบความสอดคล้องกันของรีเลชัน 2 รีเลชันที่ใช้กันโดยทั่วไปได้แก่**เงื่อนไขบังคับภายนอก** เช่น การกำหนดให้พนักงานทำงานประจำในแผนกใดๆ เราต้องสร้างความสัมพันธ์ระหว่างรหัสพนักงานกับรหัสของแผนก ซึ่งในบางครั้งผู้ป้อนข้อมูลอาจป้อนข้อมูลรหัสพนักงานผิดพลาดโดยไม่ปรากฏว่ามีรหัสพนักงานตามที่ป้อนในบริษัท เราจึงมีความจำเป็นต้องตรวจสอบก่อนว่าการกำหนดความสัมพันธ์ระหว่างพนักงานกับแผนกโดยรหัสพนักงานและรหัสแผนกนั้น รหัสพนักงานที่ป้อนเข้าไปในฐานข้อมูลจำเป็นต้องมีอยู่ในตารางข้อมูลพนักงานแล้ว เช่นเดียวกันกับรหัสแผนกที่จะต้องมีอยู่ในตารางของแผนก เราสามารถยกตัวอย่างอื่นๆ ประกอบกรณีตัวอย่างข้อมูลให้เห็นภาพได้ชัดเจนขึ้นดังนี้

จากสคีมาระบบฐานข้อมูลของมหาวิทยาลัยในบทที่ 1 แสดงรีเลชันการลงทะเบียนเรียนของนักศึกษาได้ดังนี้

Enrolled (*sid:* string, *cid:* string, *grade:* string)

ในที่นี้ Enrolled คือรีเลชันการลงทะเบียน sid คือแอทริบิวต์รหัสนักศึกษา cid และ grade คือผลการเรียนในรายวิชานั้น ข้อมูลแต่ละระเบียนที่เกิดจากสคีมานี้หมายถึงนักศึกษาคนใด ลงทะเบียนเรียนรายวิชาอะไรไปแล้ว และได้เกรดเท่าใด เพื่อให้แน่ใจว่านักศึกษาที่ลงทะเบียนเรียนในรายวิชาต่างๆ มีอยู่จริง (เพื่อป้องกันข้อมูลผิดพลาดอันอาจจะเกิดจากการกรอกข้อมูล) รหัสนักศึกษาที่จะเพิ่มในรีเลชัน Enrolled ต้องปรากฏอยู่ในรีเลชันนักศึกษา เราเรียก

แอทริบิวต์ sid ในตาราง Enrolled นี้ว่าเป็นคีย์ภายนอก (foreign key) ที่อ้างถึง (refer) รหัสนักศึกษา sid ในตารางนักศึกษา (Student) แอทริวบิวต์ทั้งสองต้องมีชนิดของข้อมูลที่สอดคล้องกัน แต่ไม่จำเป็นต้องใช้ชื่อเดียวกัน แสดงได้ดังกรณีตัวอย่างข้อมูลต่อไปนี้



cid	*grade*	*sid*
204203	A	B5075650
204204	B	B5075831
204205	D	B5075688
204329	F	B5075666

cid	*sid*	*name*	*login*	*age*	*gpa*
3100904022132	B5075666	สมชาย	somchai@it	18	3.44
3100904032132	B5075688	สมศรี	somsri@com	18	3.21
3100905622132	B5075650	สมศรี	somsri@it	19	3.82
3100904055132	B5075831	สมศักดิ์	somsak@med	21	1.80
3100904078132	B5075832	สมน้ำหน้า	somnamna@math	22	2.00

Enrolled (Referencing relation) Student (Referenced relation)

บูรณภาพเชิงอ้างอิง (Referential Integrity)

จากตัวอย่าง รหัสนักศึกษาทั้งหมดในรีเลชัน Enrolled ต้องอ้างอิงมาจากรีเลชัน Student แต่รหัสนักศึกษาไม่จำเป็นต้องอ้างอิงมาจากรีเลชัน Enrolled มีนักศึกษาบางคนที่ยังไม่ได้ลงทะเบียนเรียน เช่น "สมน้ำหน้า" ที่มีรหัสนักศึกษาคือ "B5075832" และไม่ปรากฏในรีเลชัน Enrolled

ถ้าเราทำการเพิ่มระเบียนใหม่ในรีเลชัน Enrolled ด้วยข้อมูล <"B5075833", 204203, C> การกำหนดเงื่อนไขบังคับคีย์ภายนอก ทำให้เราไม่สามารถเพิ่มข้อมูลนี้ได้เนื่องจากไม่มีนักศึกษารหัส B5075833 ปรากฏอยู่ รหัสนักศึกษานี้คล้ายคลึงกับ B5075832 ซึ่งอาจเป็นการพิมพ์ข้อมูลที่ผิดพลาด ทำให้ผู้ใช้สามารถจัดการแก้ไขข้อมูลให้ถูกต้องได้ นอกจากนี้หากมีการลบนักระเบียนนักศึกษา B5075831 ออก ระบบจัดการฐานข้อมูลจะไม่อนุญาตให้ดำเนินการเนื่องจากมีการอ้างถึงนักศึกษาดังกล่าวจากตาราง Enrolled ซึ่งลงทะเบียนเรียนไว้ในรายวิชา 204204 อย่างไรก็ตามเราอาจกำหนดให้ระบบจัดการฐานข้อมูลดำเนินการอย่างหนึ่งอย่างใดนอกเหนือจากการไม่อนุญาตให้ระเบียนนักศึกษาไม่สามารถลบได้ เช่น อาจกำหนดให้ทำการลบข้อมูลทุกระเบียนที่อ้างถึงรหัสนักศึกษาคนนั้นออกจากตาราง Enrolled ด้วยก็ได้ ซึ่งขึ้นอยู่กับข้อกำหนดและลักษณะการใช้งานจริง

เงื่อนไขบังคับภายนอกนี้ทำให้ข้อมูลมีความถูกต้องในการอ้างอิงข้อมูลซึ่งเราเรียกลักษณะความมีบูรณภาพของข้อมูลในลักษณะนี้ว่า **บูรณภาพเชิงอ้างอิง (referential integrity)** และการกำหนดเงื่อนไขบังคับนี้เป็นการ**บังคับให้ข้อมูลมีบูรณภาพเชิงอ้างอิง (enforce referenctial integrity)** แม้ว่าเงื่อนไขบังคับคีย์ภายนอกนี้จะใช้ชื่อดังกล่าวแต่เราสามารถอ้างอิงแอทริบิวต์ภายในรีเลชันเดียวกันก็ได้

4.3.3 เงื่อนไขบังคับทั่วไป (General Constraints)

เงื่อนไขบังคับทั่วไป คือ ข้อกำหนดของข้อมูลอื่นๆ ซึ่งอาจเกี่ยวข้องกับความหมายของข้อมูลหรือข้อกำหนดทางธุรกิจ

เงื่อนไขบังคับโดเมน คีย์ และคีย์ภายนอก เป็นเงื่อนไขบังคับโดยทั่วไปที่ระบบจัดการฐานข้อมูลรองรับ นอกจากนี้ระบบจัดการฐานข้อมูลยังมีความสามารถในการรองรับเงื่อนไขบังคับอื่นๆ อีกที่นอกเหนือจากเงื่อนไขบังคับที่ได้กล่าวถึง แต่มีความจำเป็นในการกำหนดเพื่อให้รองรับกับการใช้งานจริงหรือข้อกำหนดทางธุรกิจ เช่น นักศึกษาที่มีอายุตั้งแต่ 18 ปีขึ้นไป ต้องมีเกรดเฉลี่ยมากกว่า 2.75 เป็นต้น หากมีการเพิ่มหรือแก้ไขข้อมูลนักศึกษาที่ไม่ตรงตามเงื่อนไขจะไม่สามารถกระทำได้และจะต้องมีการจัดการความผิดพลาดนี้ นอกจากนี้ยังรวมถึงเงื่อนไขบังคับที่ไม่ซับซ้อนนักเช่นคอลัมน์นั้นๆ จะต้องปรากฏค่าอยู่ในกรณีที่มีการแทรกข้อมูลระเบียนใหม่ เป็นต้น

4.4 วิว (View)

วิว คือ มุมมองข้อมูลของผู้ใช้ซึ่งอยู่ในรูปแบบของตารางที่อาจเกิดจากรีเลชันหลายๆ รีเลชัน

ตารางข้อมูลในมุมมองของผู้ใช้นั้นอาจเป็นข้อมูลที่เกิดจากรีเลชันหลายๆ รีเลชันก็ได้ เราเรียกตารางข้อมูลตามมุมมองของผู้ใช้ว่าวิว ซึ่งวิวนี้ระบบจัดการฐานข้อมูลเชิงสัมพันธ์จะจัดเก็บไว้เพียงสคีมาของวิว กรณีตัวอย่างข้อมูลจริงจะเกิดจากการนำของมูลจากรีเลชันที่เกี่ยวข้องมาเชื่อมโยงกันและแสดงผลที่กำหนดโดยสคีมา

ตัวอย่าง

สคีมาของฐานข้อมูลระบบลงทะเบียนในมหาวิทยาลัยจากบทที่ 1 ประกอบไปด้วยรีเลชันดังนี้

Students(*cid:* string, *sid:* string, *name:* string, *login:* string, *age:* integer, *gpa:* real)

Faculty(*fid:* string, *fname:* string, *sal:* real)

Courses(*cid:* string, *cname:* string, *credits:* integer)

Rooms(*rno:* integer, *address:* string, *capacity:* integer)

Enrolled(*sid:* string, *cid:* string, *grade:* string)

Teaches(*fid:* string, *cid:* string)

Meets_In(*cid:* string, *rno*: integer, *time:* string)

พิจารณาตัวอย่างรีเลชัน Courseinfo ซึ่งเป็นวิวที่สร้างขึ้นเพื่อแสดงข้อมูลรายวิชา ประกอบด้วยรหัสวิชา cid ชื่ออาจารย์ผู้สอน fname และจำนวนผู้ลงทะเบียน enrollment

Courseinfo(*cid:* string, *fname:* string, *enrollment:* integer)

รีเลชัน Courseinfo นี้ไม่ได้มีตัวข้อมูลจริงเก็บไว้ในฐานข้อมูลแต่มีสร้างขึ้นเมื่อผู้ใช้เรียกดูข้อมูลผ่านทางวิวนี้ ซึ่งตารางข้อมูลที่ถูกสร้างขึ้นจะสร้างมาจากข้อมูลในตารางอื่น ได้แก่ cid จากตารางรายวิชา (Courses) fname ได้มา

จากตารางการสอน (Teaches) รวมกับตารางอาจารย์ (Faculty) และ enrollment ได้มาจากการคำนวณโดยนับจำนวนเรคคอร์ดที่นักศึกษาลงทะเบียนรายวิชานั้นๆ จากตารางลงทะเบียน (Enrolled) กรณีตัวอย่างของรีเลชัน Courseinfo และข้อมูลในตารางที่เกี่ยวข้องแสดงได้ดังนี้

ข้อมูลในตาราง

Enrolled

cid	*grade*	*sid*
204203	A	B5075650
204204	B	B5075831
204205	D	B5075688
204329	F	B5075666
204204	A	B5075688

Students

cid	*sid*	*name*	*login*	*age*	*gpa*
3100904022132	B5075666	สมชาย	somchai@it	18	3.44
3100904032132	B5075688	สมศรี	somsri@com	18	3.21
3100905622132	B5075650	สมศรี	somsri@it	19	3.82
3100904055132	B5075831	สมศักดิ์	somsak@med	21	1.80
3100904078132	B5075832	สมน้ำหน้า	somnamna@math	22	2.00
3100904032233	B5075899	สมแล้ว	somlaew@math	22	2.00

Teaches

fid	*cid*
146009	204203
146009	204204
146009	204205
146009	204329
147008	204205

Faculty

fid	*fname*	*salary*
146009	สถิตย์	50000
147008	พอลล่า	50000

ข้อมูลที่เกิดขึ้นจากวิว Courseinfo โดยระบุรหัสวิชาเป็น 204204

Courseinfo

cid	*fname*	*enrollment*
204204	สถิตย์	2

cid: จากตาราง Courses

fname: จากตาราง Faculty และ Teaches ในกรณีนี้ในตาราง Teaches อาจารย์ผู้สอนมีรหัสประจำตัวอาจารย์คือ 146009 ซึ่งเมื่ออ้างอิงถึงชื่ออาจารย์ในตาราง Faculty ได้แก่อาจารย์สถิตย์

enrollment: ได้จากการคำนวนโดยการนับจำนวนผู้เรียนรายวิชา 204204 ในตาราง Enrolled ซึ่งมีจำนวน 2 คนได้แก่ นักศึกษารหัส B5075831 และ B5075688

การสร้างวิวเป็นการสร้างมุมมองของผู้ใช้ซึ่งเป็นการรองรับฐานข้อมูลในระดับภายนอก ทำให้เกิดประโยชน์ดังนี้

- ความเป็นอิสระของข้อมูล กล่าวคือหากมีการเปลี่ยนแปลงโครงสร้างตารางใดๆ เช่นมีการเพิ่มแอทริบิวต์ตำแหน่งทางวิชาการของตารางอาจารย์แล้ว วิว Courseinfo ไม่ได้รับผลกระทบใดๆ
- ความสะดวกจากการเรียกดูและจัดการข้อมูลที่ซับซ้อน จากตัวอย่างพบว่าข้อมูลที่กว่าจะได้มาเป็นตาราง Courseinfo นี้ต้องทำการเชื่อมโยงตารางหลายๆ ตารางเข้าด้วยกัน การสร้างวิวและเรียกใช้งานผ่านทางวิวจะทำให้เรากำหนดการเชื่อมโยงต่างๆ เพียงครั้งเดียว หลังจากนั้นสามารถเรียกดูและจัดการข้อมูลผ่านทางวิวซึ่งเปรียบเสมือนตารางเพียงตารางเดียวได้โดยสะดวก
- ขจัดปัญหาความซ้ำซ้อนและผิดพลาดของข้อมูล เช่นค่าของคอลัมน์ enrollment ในวิว Courseinfo เกิดจากการคำนวณและเราไม่ต้องจัดเก็บข้อมูลนี้ไว้เพราะจำนวนผู้ลงทะเบียนสามารถเปลี่ยนแปลงได้ตลอดเวลา
- เพิ่มความปลอดภัยของข้อมูล โดยการซ่อนข้อมูลจริงไว้เบื้องหลังวิว จากตัวอย่าง DBA สามารถกำหนดไม่ให้สิทธิผู้ใช้ทั่วไปเข้าถึงข้อมูลเงินเดือนของอาจารย์ได้ สามารถเรียกดูข้อมูลผ่านทางวิวเท่านั้น

ซึ่งทั้งหมดที่กล่าวมานี้เป็นเพียงข้อดีส่วนหนึ่งของวิวเท่านั้น

รีเลชันที่กำหนดขึ้นโดยไม่ได้มีข้อมูลบรรจุอยู่ในตัวรีเลชันเราเรียกว่าวิว (View) ในขณะที่รีเลชันที่บรรจุข้อมูลจริงในฐานข้อมูลซึ่งได้แก่รีเลชัน Students, Faculty, Teaces, Enrolled ในตัวอย่างนี้เราเรียกว่า**รีเลชันฐาน (base relation)** การสร้างวิวด้วย SQL อธิบายไว้ในบทที่ 7

4.5 การแปลง ER Model ให้เป็นแบบจำลองข้อมูลเชิงสัมพันธ์

หลังจากได้ทราบถึงแนวคิดของแบบจำลองเชิงสัมพันธ์และฐานข้อมูลเชิงสัมพันธ์แล้ว ในหัวข้อต่อไปนี้เป็นการอธิบายวิธีการแปลงแบบจำลองฐานข้อมูลที่ได้ออกแบบไว้ในระดับแนวคิดให้เป็นฐานข้อมูลในระดับตรรกะก่อนที่จะออกแบบฐานข้อมูลในระดับกายภาพในขั้นสุดท้ายต่อไป ER model ซึ่งเป็นแบบจำลองข้อมูลในระดับแนวคิดนั้นเป็นการออกแบบในระดับสูง ไม่ยึดติดกับระบบจัดการฐานข้อมูลรูปแบบใด แต่การจะจัดการฐานข้อมูลให้มีประสิทธิภาพเราอาศัยระบบจัดการฐานข้อมูลเชิงสัมพันธ์ในที่นี้ เราต้องทำการแปลง ER model ให้อยู่ในรูปของแบบจำลองเชิงสัมพันธ์เพื่อนำไปใช้งานในระบบจัดการฐานข้อมูลเชิงสัมพันธ์ การแปลงให้อยู่ในรูปของฐานข้อมูลเชิงสัมพันธ์นั้นเราอาศัย SQL ในส่วนของการนิยามข้อมูล ใช้นิยามโครงสร้างของรีเลชันและกำหนดเงื่อนไขบังคับต่างๆ อย่างไรก็ดีแบบจำลองเชิงสัมพันธ์นั้นสามารถแทนได้ด้วยแผนภาพแบบจำลองฐานข้อมูลเชิงสัมพันธ์ (relational database model diagram) ซึ่งใช้ในการอธิบายการแปลงจาก ER model ให้เป็นแบบจำลองเชิงสัมพันธ์

แผนภาพแบบจำลองฐานข้อมูลเชิงสัมพันธ์นี้มีความคล้ายคลึงกับ ER model แต่มีการกำหนดรายละเอียดอื่นๆ ของแบบจำลองเชิงสัมพันธ์ เราไม่จำเป็นต้องใช้แผนภาพแบบจำลองฐานข้อมูลเชิงสัมพันธ์ก็ได้ แผนภาพแบบจำลองฐานข้อมูลเชิงสัมพันธ์ยังอาจถูกเรียกว่าเป็น ER model อีกรูปแบบหนึ่ง และในบางครั้งผู้ออกแบบฐานข้อมูลอาจใช้แผนภาพแบบจำลองฐานข้อมูลเชิงสัมพันธ์ในการออกแบบฐานข้อมูลในระดับแนวคิดเลยก็ได้เนื่องจากมีความคล้ายคลึงกันกับ ER model ทั้งนี้แล้วแต่ความเหมาะสมของระบบและความถนัดของผู้ออกแบบ โดยเราสามารถสร้างรีเลชันและกำหนดเงื่อนไขด้วย SQL ได้เลย ซึ่ง SQL นั้นได้อธิบายไว้อย่างละเอียดในบทที่ 7 ในบทนี้จะอธิบายถึงการเชื่อมโยงและความสัมพันธ์ระหว่าง ER model และแบบจำลองเชิงสัมพันธ์ จึงสรุปได้ว่าเราอาจเลือกวิธีการออกแบบฐานข้อมูลเชิงสัมพันธ์ได้ 2 แบบใหญ่ๆ ดังนี้

การใช้แผนภาพแบบจำลอง 2 ระดับ	**การใช้แผนภาพแบบจำลองข้อมูล 1 ระดับ**
1. เขียน ERD 2. แปลง ERD ให้เป็นโมเดลเชิงสัมพันธ์ด้วย • แผนภาพแบบจำลองฐานข้อมูลเชิงสัมพันธ์ หรือ • เขียนสคีมาของตารางโดยไม่ใช้แผนภาพ 3. สร้างคำสั่ง SQL จากการออกแบบ	1. เขียน ERD หรือ แผนภาพแบบจำลองฐานข้อมูลเชิงสัมพันธ์อย่างใดอย่างหนึ่ง 2. สร้างคำสั่ง SQL จากการออกแบบ

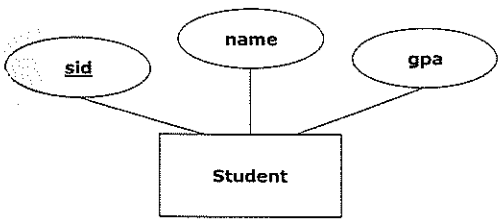
สำหรับหัวข้อต่อไปนี้จะเป็นการแสดงตัวอย่างการสร้างแผนภาพแบบจำลองฐานข้อมูลเชิงสัมพันธ์ซึ่งสามารถสร้างจากการวิเคราะห์ความต้องการของผู้ใช้ได้โดยตรง หรือแปลงจาก ER model ได้ดังนี้ สำหรับตัวอย่างการแปลง ER model ให้เป็นแบบจำลองเชิงสัมพันธ์ต่อไปนี้ใช้ตัวอย่างจากเรื่อง ER model ในบทที่ 3

4.5.1 การแปลงเอนทิตีให้เป็นตาราง

เอนทิตี (entity types) ในแบบจำลองเชิงสัมพันธ์แทนได้อย่างตรงไปตรงมาด้วยตารางรีเลชันหรือตารางข้อมูล และแอทริบิวต์ของเอนทิตีคือแอทริบิวต์ของรีเลชัน

ตัวอย่าง

ตัวอย่างเอนทิตีในมหาวิทยาลัย ERD ของเอนทิตีนักศึกษา (Student) ประกอบด้วยแอทริบิวต์ รหัสนักศึกษา (sid) ชื่อ-นามสกุล (name) เกรดเฉลี่ย (gpa) ดังนี้



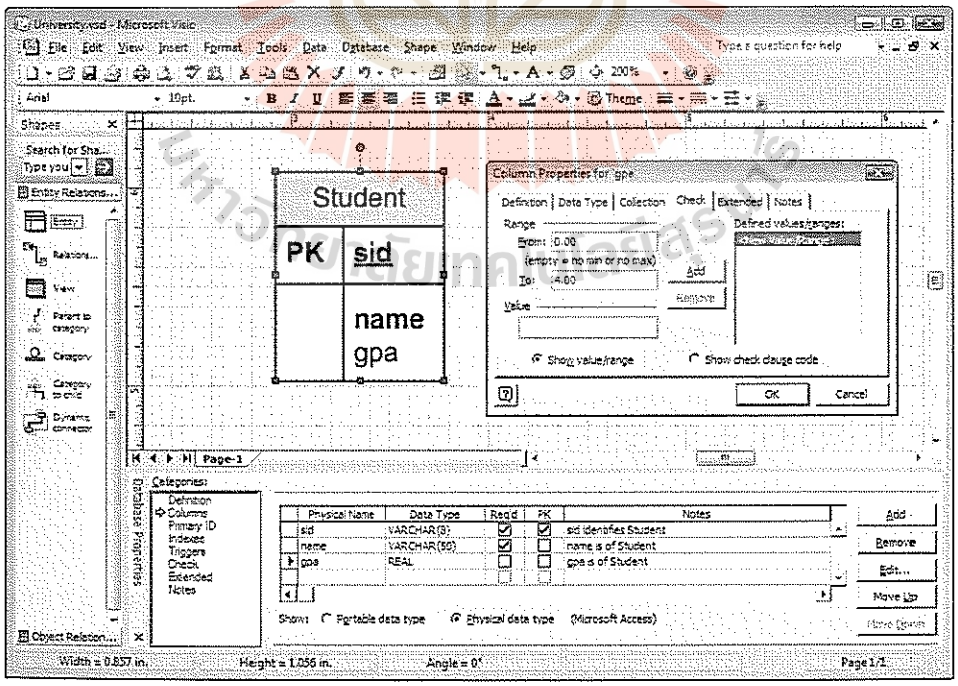
Student	
PK	sid
	name gpa

การแปลงจากเอนทิตีสามารถแปลงไปเป็นสคีมาของแบบจำลองเชิงสัมพันธ์ได้อย่างตรงไปตรงมา ทางด้านขวามือเป็นสคีมาของรีเลชัน Student โดยสัญลักษณ์ PK หมายถึงคีย์หลักซึ่งรวมถึงการขีดเส้นใต้แอทริบิวต์ด้วย ตัวหนาหมายถึงแอทริบิวต์นั้นจำเป็นต้องมีข้อมูลและเป็นค่าว่าง (null) ไม่ได้ กรณีตัวอย่างของข้อมูลในรีเลชันสามารถแสดงได้ดังนี้

Student

sid	**name**	**gpa**
B5075666	สมชาย	3.44
B5075688	สมศรี	3.21
B5075650	สมศรี	3.82
B5075831	สมศักดิ์	1.80
B5075832	สมน้ำหน้า	2.00

การสร้างแบบจำลองใดๆ ไม่ว่าจะเป็น ER model หรือแบบจำลองเชิงสัมพันธ์สามารถสร้างได้ด้วย CASE tools ที่ได้กล่าวถึงในบทที่ 1 CASE tool นี้ สามารถใช้เพื่อสร้างแผนภาพแบบจำลองข้อมูลใดๆ กำหนดข้อกำหนดต่างๆ ไว้ในแผนภาพ และสามารถแปลงเป็นคำสั่ง SQL ได้โดยอัตโนมัติ ตัวอย่างต่อไปนี้เป็นการใช้โปรแกรม Microsoft Office Visio ในการวาดแผนภาพแบบจำลองฐานข้อมูลเชิงสัมพันธ์ และกำหนดเงื่อนไขบังคับโดเมนให้กับแอทริบิวต์ gpa ให้มีข้อมูลชนิด REAL และมีค่าอยู่ระหว่าง 0.00 ถึง 4.00 เท่านั้น



การมีภาษาที่มีโครงสร้างสำหรับจัดการระบบฐานข้อมูลเชิงสัมพันธ์นั้นเป็นกฎข้อหนึ่งที่ระบบจัดการฐานข้อมูลเชิงสัมพันธ์ต้องมี เราสามารถใช้ SQL ซึ่งเป็นภาษาที่นิยมในการจัดการระบบจัดการฐานข้อมูลในการสร้างรีเลชันจาก ER model ดังกล่าวข้างต้นได้เช่นเดียวกัน คำสั่ง SQL ที่เกี่ยวข้องในการสร้างรีเลชันและกำหนดเงื่อนไขบังคับต่างๆ ยังได้แสดงไว้ในหัวข้อนี้ แต่จะกล่าวถึงเฉพาะจุดสำคัญที่เกี่ยวข้องกับการแปลงการออกแบบฐานข้อมูลระดับแนวคิดเป็นระดับตรรกะเท่านั้น สำหรับรายละเอียดคำสั่ง SQL อธิบายไว้ในบทที่ 7

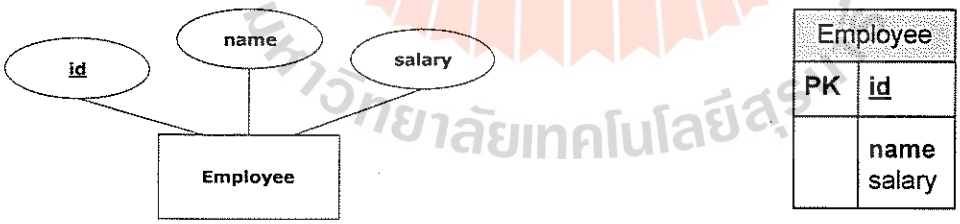
เอนทิตีนักศึกษาสามารถสร้างเป็นรีเลชันด้วยคำสั่ง SQL ดังนี้

```
CREATE TABLE student( sid     VARCHAR(8) NOT NULL,
                      name    VARCHAR(50) NOT NULL,
                      gpa     REAL,
                      PRIMARY KEY (sid))
```

ในที่นี้ student เป็นชื่อรีเลชัน sid, name และ gpa เป็นชื่อแอทริบิวต์ VARCHAR(8), VARCHAR(50) และ REAL เป็นการกำหนดเงื่อนไขบังคับโดเมนของแอทริบิวต์ให้มีชนิดเป็นตัวอักษรขนาดความยาวไม่เกิน 8, 50 และข้อมูลชนิดจำนวนจริงตามลำดับ PRIMARY KEY(sid) เป็นการกำหนดเงื่อนไขบังคับคีย์ ให้แอทริบิวต์ sid เป็นคีย์หลัก สำหรับคำสั่ง NOT NULL เป็นการระบุว่าแอทริบิวต์นั้นๆ ไม่สามารถเป็นค่าว่างได้

ตัวอย่าง

ตัวอย่างการแปลงจาก ERD เป็นแบบจำลองเชิงสัมพันธ์เพิ่มเติม เอนทิตีในบริษัททั่วไป ERD ของเอนทิตีพนักงาน (Employee) ประกอบด้วยแอทริบิวต์ รหัสประจำตัวประชาชน (id) ชื่อ-นามสกุล (name) เงินเดือน (salary) ดังนี้



แบบจำลองเชิงสัมพันธ์สามารถแสดงได้ดังแผนภาพแบบจำลองฐานข้อมูลเชิงสัมพันธ์ดังรูปด้านขวา และกำหนดเป็นคำสั่ง SQL ได้ดังนี้

```
CREATE TABLE employee( id      VARCHAR(11) NOT NULL,
                       name    VARCHAR(50) NOT NULL,
```

salary CURRENCY,

PRIMARY KEY (id))

4.5.2 การแปลงความสัมพันธ์ให้เป็นตาราง

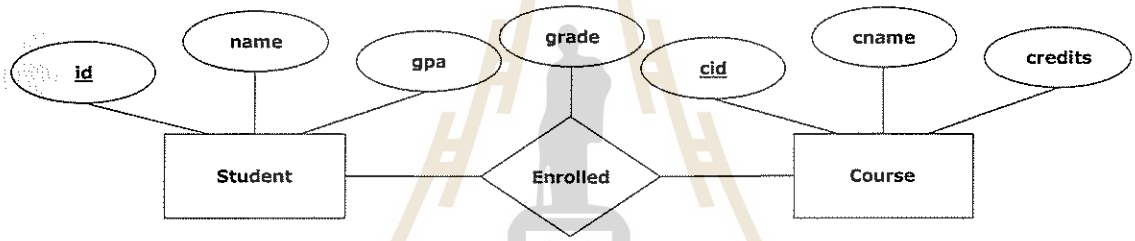
4.5.2.1 ความสัมพันธ์ที่ไม่มีเงื่อนไข

ความสัมพันธ์ที่ไม่มีเงื่อนไขนั้นโดยทั่วไปคือความสัมพันธ์ที่มีคาร์ดินัลลิตีแบบ many-to-many และยังไม่ได้กำหนดเงื่อนไขการมีส่วนร่วม

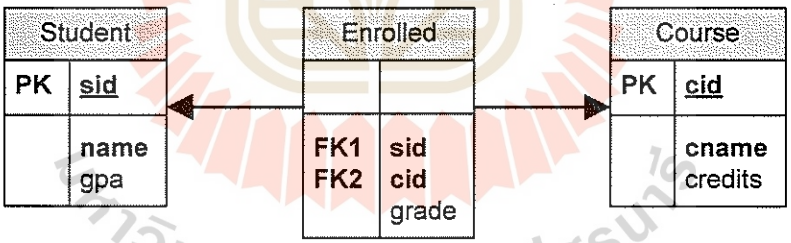
ตัวอย่าง

ตัวอย่างความสัมพันธ์ในมหาวิทยาลัย ความสัมพันธ์ของนักศึกษา (Student) **ลงทะเบียนเรียน (Enrolled)** รายวิชา (Course) ซึ่งความสัมพันธ์แต่ละความสัมพันธ์ในที่นี้นักศึกษาคนหนึ่งลงทะเบียนรายวิชาหนึ่ง ดังนี้

ER model



Relational database model



SQL

```
CREATE TABLE enrolled ( sid      VARCHAR(8)  NOT NULL,
                        cid      VARCHAR(6) NOT NULL,
                        grade    REAL,
                        FOREIGN KEY (sid) REFERENCES student,
                        FOREIGN KEY (cid) REFERENCES course (cid) )
```

ความสัมพันธ์แบบไม่มีเงื่อนไขนั้นในแบบจำลองเชิงสัมพันธ์แทนด้วยรีเลชันเช่นเดียวกันกับเอนทิตี และแอทริบิวต์ประจำความสัมพันธ์นั้นๆ แทนด้วยแอทริบิวต์ในแบบจำลองเชิงสัมพันธ์เช่นกัน นอกจากนี้ในแบบจำลองเชิงสัมพันธ์ต้องมีการระบุแอทริบิวต์ที่เป็นคีย์หลักจากเอนทิตีที่มีส่วนในความสัมพันธ์นั้นให้เป็นสมาชิกในรีเลชันด้วย

ดังนั้นการเพิ่มหรือแก้ไขข้อมูลใดๆ ในรีเลชันของความสัมพันธ์ที่อ้างอิงจากค่าของตารางหรือรีเลชันอื่น ค่านั้นๆ ต้องปรากฏอยู่ในรีเลชันที่อ้างถึงด้วย นำมาซึ่งหลักสำคัญอีกอย่างหนึ่งของการแทนแบบจำลองเชิงสัมพันธ์สำหรับความสัมพันธ์คือการกำหนดเงื่อนไขบังคับภายนอก ซึ่งในแบบจำลองฐานข้อมูลเชิงสัมพันธ์นี้ใช้สัญลักษณ์ FK โดย FK1 เป็นการกำหนดให้ sid ในรีเลชัน Enrolled อ้างอิงถึง sid ในรีเลชัน Student และ cid ในรีเลชัน Enrolled อ้างอิงถึง cid ของรีเลชัน Course หัวลูกศรของความสัมพันธ์ระหว่างรีเลชันชี้ไปยังตารางที่เป็นเจ้าของคีย์หลัก

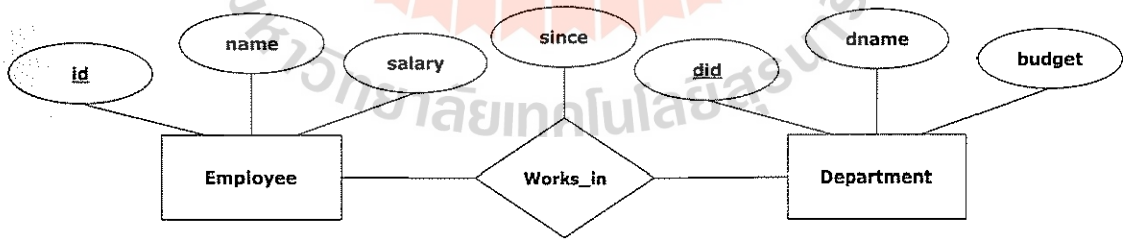
การสร้างรีเลชันจากความสัมพันธ์ Enrolled สอดคล้องกับคำสั่ง SQL ที่ได้แสดงไว้ โดยเป็นการสร้างรีเลชันที่ประกอบไปด้วยแอทริบิวต์ sid, cid และ grade แทนรหัสนักศึกษา รหัสรายวิชาที่นักศึกษาลงทะเบียนเรียน และเกรดที่ได้จากรายวิชานั้นตามลำดับ โดเมนของข้อมูลแอทริบิวต์ในรีเลชันที่เกิดจากความสัมพันธ์นี้ต้องสอดคล้องกับโดเมนของแอทริบิวต์ในเอนทิตีที่อ้างถึง

FOREIGN KEY (sid) REFERENCES student เป็นคำสั่งที่ใช้ในการกำหนดเงื่อนไขบังคับภายนอก หมายความว่าข้อมูลที่จะเพิ่มหรือแก้ไขในแอทริบวิต์ sid ต้องมีปรากฏอยู่ในแอทริบิวต์ชื่อเดียวกันนี้ในรีเลชัน Student สำหรับ FOREIGN KEY (cid) REFERENCES course (cid) มีข้อแตกต่างจากคำสั่งกำหนดเงื่อนไขบังคับแรกตรงที่มีการกำหนดชื่อของแอทริบิวต์ที่อ้างถึงด้วยในกรณีที่แอทริบิวต์ที่อ้างถึงอาจเป็นคนละชื่อกันกับแทริบิวต์ในรีเลชันที่สร้าง เช่น ในกรณีที่เราอาจใช้ชื่อแอทริบิวต์ course_id แทนรหัสวิชาที่นักศึกษาลงทะเบียน (ไม่ใช่ cid) รหัสวิชานี้อ้างอิงถึงรหัสวิชา cid ในรีเลชัน Course เราสามารถเขียนคำสั่ง SQL ได้เป็น FOREGN KEY (course_id) REFERENCES course (cid) ได้

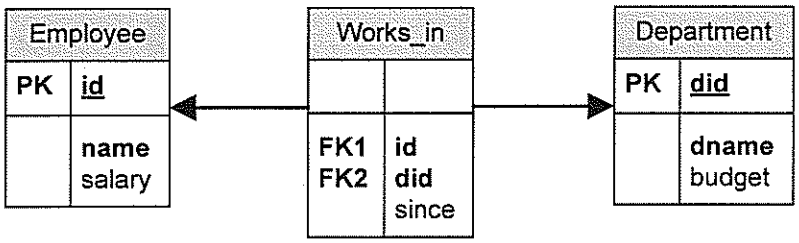
ตัวอย่าง

ตัวอย่างความสัมพันธ์ในบริษัทั่วไป ความสัมพันธ์ของพนักงาน (Employee) ทำงานใน **(Works_in)** แผนก (Department) ซึ่งความสัมพันธ์แต่ละความสัมพันธ์ในที่นี้คือพนักงานคนหนึ่งกับการทำงานให้แผนกๆ หนึ่ง ดังนี้

ER model



Relational Database Model



SQL

```
CREATE TABLE works_in (id       VARCHAR(11)  NOT NULL,
                       did      INTEGER NOT NULL,
                       since    DATE,
                       FOREIGN KEY (id) REFERENCES emplyoee,
                       FOREIGN KEY (did) REFERENCES department)
```

ตัวอย่าง

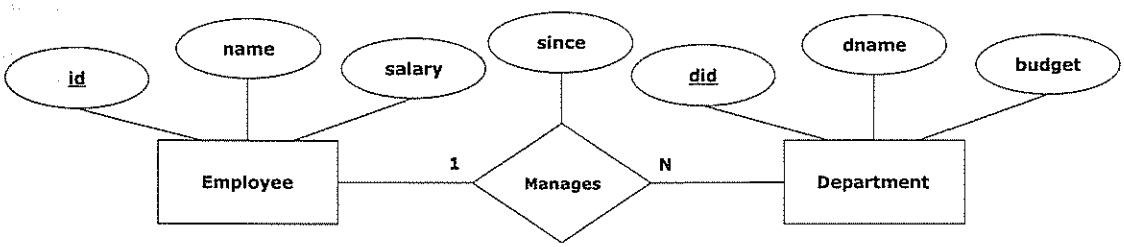
ตัวอย่างความสัมพันธ์ในบริษัทั่วไป ความสัมพันธ์ของพนักงาน (Employee) **ทำงานใน (Works_in)** แผนก (Department) ซึ่งความสัมพันธ์แต่ละความสัมพันธ์ในที่นี้คือพนักงานคนหนึ่งกับการทำงานให้แผนกๆ หนึ่ง ดังนี้ ไตรภาค

**

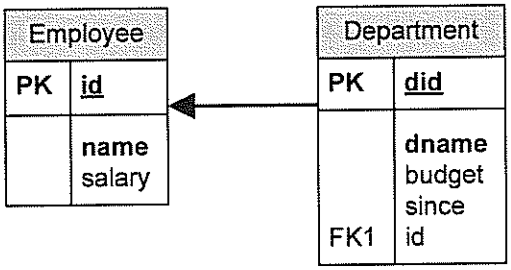
4.5.2.2 ความสัมพันธ์ที่มี Key Constraints หรือ มีคาร์ดินัลลิตีแบบ one-to-many

เงื่อนไขบังคับคีย์ในความสัมพันธ์หมายถึงการที่คีย์หลักของเอนทิตีหนึ่งจะมีค่าอยู่ในความสัมพันธ์ได้เพียงครั้งเดียว เช่นแผนกหนึ่งแผนกจะมีผู้จัดการได้เพียงคนเดียว รหัสแผนก 1 แผนกจะปรากฏอยู่ในความสัมพันธ์การจัดการได้ 1 ครั้งเท่านั้น ในกรณีที่พนักงาน 1 คนสามารถเป็นผู้จัดการได้หลายแผนกเราจะพบว่าเงื่อนไขบังคับคีย์ใช้สำหรับความสัมพันธ์ในรูปแบบ one-to-many นั่นเอง แบบจำลองเชิงสัมพันธ์จาก ER model ของการจัดการแผนกสามารถแสดงได้ดังต่อไปนี้

ER model



Relational Database Model



SQL

```
CREATE TABLE department (did    INTEGER NOT NULL,
                         dname  VARCHAR(50),
                         budget CURRENCY,
                         since  DATE,
                         id     VARCHAR(11),
                         PRIMARY KEY (did),
                         FOREIGN KEY (id) REFERENCES employee (id) )
```

ความสัมพันธ์แบบ one-to-many สามารถแทนได้ด้วยรีเลชันเช่นเดียวกับความสัมพันธ์แบบไม่มีเงื่อนไข แต่เราสามารถลดรีเลชันได้อีก โดยไม่แทนความสัมพันธ์ด้วยรีเลชัน ด้วยการยุบความสัมพันธ์นั้นแล้วนำแอทริบิวต์ของความสัมพันธ์ไปรวมเข้ากับเอนทิตีในฝั่ง *N* พิจารณาจากข้อกำหนดของความสัมพันธ์จัดการ (Manges) แผนกทุกแผนกมีผู้จัดการได้ 1 คนอยู่แล้ว ดังนั้นเราสามารถเพิ่มแอทริบิวต์ผู้จัดการในรีเลชันแผนกได้เลย ในแผนภาพคือ id ซึ่งอ้างถึงรหัสพนักงานที่จะเป็นผู้จัดการในแผนกนั้น และนำแอทริบิวต์ since มารวมในรีเลชันแผนกเช่นเดียวกัน การรวมรีเลชันเช่นนี้จะลดความยุ่งยากและทำให้การใช้งานรวดเร็วขึ้นในการแสดงข้อมูลการจัดการของแผนก (ในบางกรณีทำให้ประหยัดเวลาประมวลผลเป็นอย่างมาก) โดยเราไม่จำเป็นต้องเชื่อมรีเลชันถึง 3 รีเลชันเข้าด้วยกัน อย่างไรก็ตามการลดรีเลชันสำหรับความสัมพันธ์แบบ one-to-many อาจมีผลเสีย กล่าวคือหากข้อกำหนดทางธุรกิจไม่ได้ระบุว่าแผนกทุกแผนกต้องมีผู้จัดการ ข้อมูลผู้จัดการในตารางแผนกจะว่างอยู่ (ซึ่งต้องแทนด้วย NULL) หากเป็นข้อมูลอื่นที่มีจำนวนมากกว่านี้ การยุบรวมรีเลชันจะทำให้เกิดค่าว่างมากมายและเสียเนื้อที่จัดเก็บข้อมูล

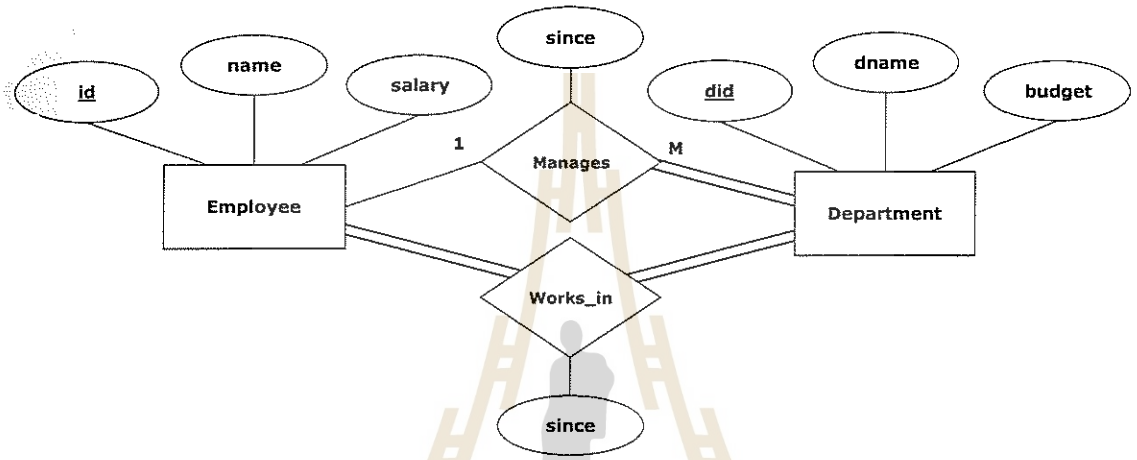
4.5.2.3 ความสัมพันธ์ที่มี Participation Constraints

การกำหนดการมีส่วนร่วมแบบทั้งหมด (total participation) สามารถกระทำได้โดยการบังคับให้แอทริบิวต์ที่มีส่วนร่วมในความสัมพันธ์ไม่สามารถเป็นค่าว่างได้

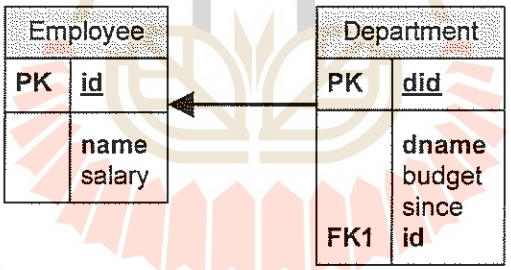
ตัวอย่าง

ความสัมพันธ์จัดการ (Manages) ที่ระบุเงื่อนไขเพิ่มเติมว่านอกจากแต่ละแผนกจะมีผู้จัดการได้เพียง 1 คนแล้ว ทุกแผนกจะต้องมีผู้จัดการ ดังนั้นแต่ละทูเพิลของแผนก จะต้องระบุรหัสผู้จัดการด้วย

ER model



Relational database model

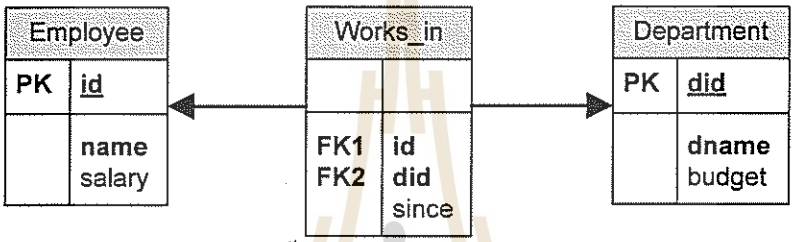


SQL

```
CREATE TABLE department (did    INTEGER NOT NULL,
                         dname  VARCHAR(50),
                         budget CURRENCY,
                         since  DATE,
                         id     VARCHAR(11) NOT NULL,
                         PRIMARY KEY (did),
                         FOREIGN KEY (id) REFERENCES employee (id) )
```

แผนกทุกแผนกจำเป็นต้องมีผู้จัดการ สามารถรองรับได้ด้วยเงื่อนไขบังคับการมีส่วนร่วมทั้งหมด การบังคับให้ทุกๆ ทูเพิลของแผนกมีรหัสพนักงานที่เป็นผู้จัดการกำหนดได้โดยการใช้ตัวหน้าในแบบจำลองฐานข้อมูลเชิงสัมพันธ์และโดยการใช้คำสั่ง NOT NULL สำหรับแอทริบิวต์ id ในตาราง department ด้วย SQL ดังที่แสดง

สำหรับความสัมพันธ์ที่แทนด้วยรีเลชันโดยไม่ได้ทำการยุบรีเลชัน (ที่กล่าวถึงในหัวข้อ 4.5.1) หรือรีเลชันที่สร้างจากความสัมพันธ์ที่ไม่มีเงื่อนไข (many-to-many) นั้นไม่สามารถที่จะกำหนดเงื่อนไขบังคับได้จากคำสั่งพื้นฐานในการสร้างรีเลชันด้วย SQL เราต้องใช้ SQL ชนิดที่มีความซับซ้อนกว่านี้ในรูปแบบที่เรียกว่า assertion ซึ่งจะได้กล่าวถึงในบทที่ 7 ซึ่งรูปต่อไปนี้แสดงถึงความสัมพันธ์ที่ต้องอาศัย assertion ดังกล่าว



เราสามารถระบุได้ว่าทุกๆ ทูเพิลที่เกิดขึ้นในความสัมพันธ์พนักงานทำงานในแผนก (Works_in) จำเป็นต้องอ้างถึงรีเลชันพนักงาน และแผนก ตลอดจนมีค่าว่างไม่ได้ แต่ไม่สามารถบังคับได้ว่ารหัสพนักงานทุกคนต้องปรากฏในความสัมพันธ์ Works_in ด้วยในกรณีที่มีการกำหนดว่าพนักงานทุกคนต้องทำงานให้แผนกใดแผนกหนึ่ง

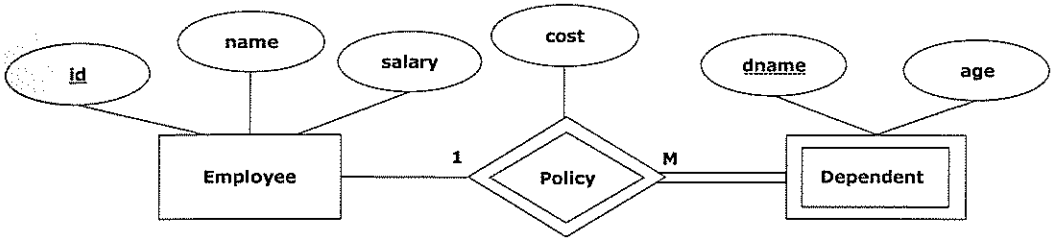
4.5.3 การแปลง Weak Entity

เอนทิตีอ่อนแอนั้นจะอยู่ในรูปแบบของความสัมพันธ์แบบ one-to-many และการมีส่วนร่วมทั้งหมดตลอด ดังนั้นจึงมีการสร้างรีเลชันเช่นเดียวกับในตัวอย่างหัวข้อ 4.5.2.3 แต่กำหนดการจัดการเพิ่มเติมกรณีที่ข้อมูลในตารางหลักที่ถูกอ้างอิงถูกลบทิ้งได้

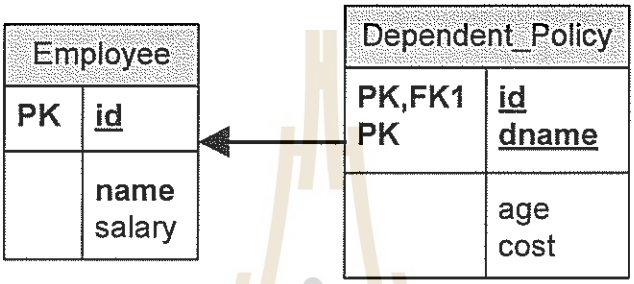
ตัวอย่าง

พนักงานซื้อกรมธรรม์สุขภาพให้กับบุตร/ธิดา บุตร/ธิดาแต่ละคนต้องมีรหัสพนักงานกำกับไว้ว่าเป็นผู้ปกครองเสมอ ดังนั้นเราสามารถยุบความสัมพันธ์ Policy เข้ากับเอนทิตี Dependent ได้ รวมถึงแอทริบิวต์ cost ที่นำมาบรรจุไว้ในเอนทิตี Dependent มีลักษณะเดียวกันกับการกำหนดเงื่อนไขบังคับการมีส่วนร่วมทั้งหมด แต่ทูเพิลในเอนทิตีอ่อนแอจะไม่จำเป็นต้องถูกเก็บไว้หากข้อมูลที่อ้างถึงในเอนทิตีหลักถูกลบทิ้ง เช่นเมื่อพนักงานคนใดลาออก ข้อมูลของบุตร/ธิดาที่อยู่ในเอนทิตีอ่อนแอจะถูกลบทิ้งด้วย ซึ่งแทนด้วยแบบจำลองเชิงสัมพันธ์ และ SQL ดังนี้

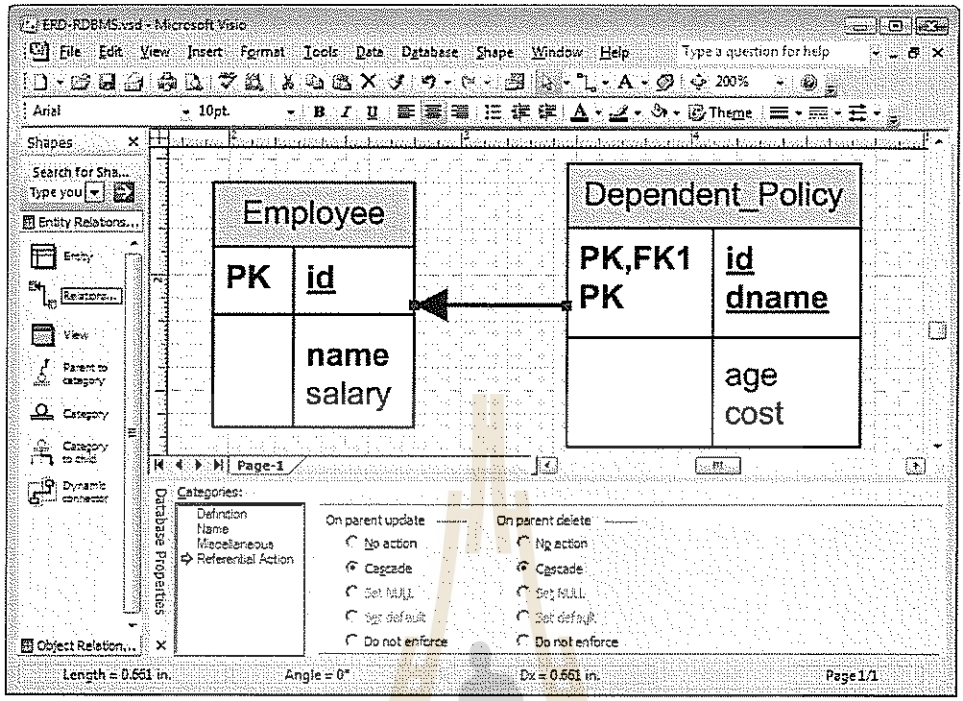
ER model



Relational database model



เอนทิตีอ่อนแอ คือเอนทิตีต้องอาศัยคีย์หลักจากตารางอ้างอิงรวมกับแอทริบิวต์ใดแอทริบิวต์หนึ่งในเอนทิตีอ่อนแอเพื่อกำหนดเป็นคียหลัก ในที่นี้ Dependent_Policy เป็นรีเลชันที่เกิดจากเอนทิตีอ่อนแออ้างอิงรหัสพนักงาน (id) ยังตาราง Employee รวมกับแอทริบิวต์ dname เราจึงกำหนดให้ทั้ง id และ dname เป็น PK สำหรับ id เป็น FK อ้างอิงจากตาราง Employee สำหรับการกำหนดให้ข้อมูลในตาราง Dependent_Policy สามารถลบทิ้งหรือปรับปรุงตามข้อมูลในตารางหลักเรากำหนดได้ด้วย CASE tool ดังรูปต่อไปนี้ ในที่นี้ระบุว่าหากมีการปรับปรุงหรือลบข้อมูลในตารางหลัก Employee (id) ข้อมูลในเอนทิตีอ่อนแอ Dependent_Policy ซึ่งได้แก่รหัสพนักงานต้องถูกปรับปรุงตามหรือลบทิ้งทั้งทูเพิล โดยใช้การวิธี Cascade



SQL

```
CREATE TABLE dependent_policy (id      VARCHAR(11)  NOT NULL,
                               dname   VARCHAR(50)  NOT NULL,
                               age     INTEGER,
                               cost    CURRENCY,
                               PRIMARY KEY (id, dname),
                               FOREIGN KEY (id) REFERENCES employee (id)
                                    ON DELETE CASCADE)
```

การสร้างคอลัมน์ด้วย SQL มีลักษณะเหมือนการสร้างรีเลชันทั่วไป เนื่องจากตารางนี้มีคีย์หลักแบบคีย์ประกอบซึ่งประกอบด้วย id และ dname จึงใช้คำสั่ง PRIMARY KEY (id, dname) สำหรับการกำหนดให้เรคอร์ดที่มีค่าขึ้นอยู่กับข้อมูลในตารางหลัก เราใช้คำสั่ง ON DELETE CASCADE สำหรับกำหนดให้ข้อมูลลบทิ้งตามตารางหลัก

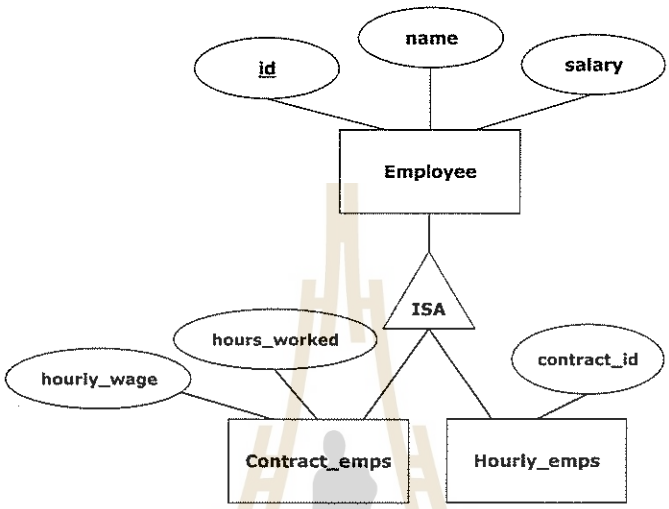
4.5.4 การแปลง Class Hierarchies

การแปลงลำดับชั้นของคลาสให้อยู่ในรูปแบบของแบบจำลองเชิงสัมพันธ์โดยทั่วไปทำได้ 2 วิธีคือการสร้างรีเลชันของคลาสหลัก จากนั้นทำการสร้างรีเลชันของคลาสที่สืบทอดและอ้างอิงไปยังรีเลชันของคลาสหลัก รีเลชันที่สร้างจากคลาสสืบทอดสามารถบรรจุแอทริบิวต์เฉพาะของคลาสตนเองเพิ่มได้นอกจากแอทริบิวต์ที่คงอยู่ในคลาสหลักแล้ว อีกวิธีหนึ่งคือการสร้างรีเลชันที่เกิดจากคลาสที่สืบทอดทั้งหมดโดยไม่ต้องสร้างคลาสหลัก แอทริบิวต์ที่มีอยู่ในคลาสหลักก็จะมีบรรจุอยู่ในทั้งรีเลชันของคลาสย่อยทั้งหมด

ตัวอย่าง

พิจารณาลำดับชั้นของคลาสพนักงานแบบเซ็นสัญญาทำงานและพนักงานรายชั่วโมงจากตัวอย่างในบทที่ 3 ดังนี้

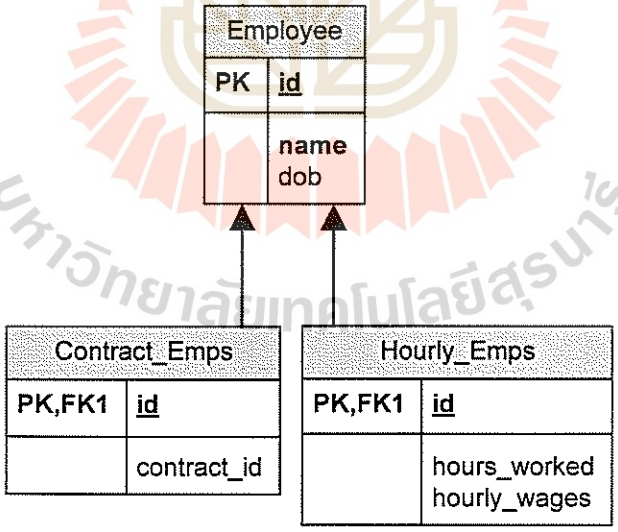
ER model



Relational database model

แบบจำลองเชิงสัมพันธ์ของลำดับชั้นของคลาสสามารถสร้างได้ 2 รูปแบบดังนี้

แบบที่ 1



แบบที่ 2

Contract_Emps	
PK	id
	name dob contract_id

Hourly_Emps	
PK	id
	name dob hourly_wages hours_worked

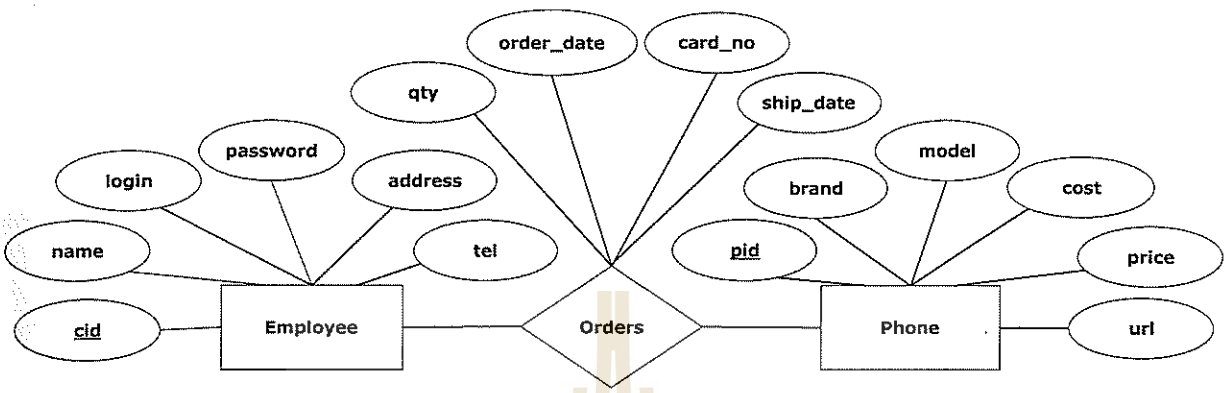
แบบที่ 1 คือการสร้างรีเลชันจากคลาสหลักและคลาสที่สืบทอดทั้งหมด คลาสหลักบรรจุแอทริบิวต์ที่คลาสย่อยมีเช่นเดียวกัน ทำให้ข้อมูลที่จัดเก็บไม่ซ้ำซ้อน นอกจากนี้แอทริบิวต์พิเศษของคลาสย่อยแต่ละคลาสที่ไม่ปรากฏในอีกคลาสหนึ่งก็สามารถแยกเก็บไว้คนละรีเลชันได้ โดยทั่วไปการสร้างรีเลชันในลักษณะนี้สามารถกระทำได้และความเหมาะสม แต่มีความซับซ้อนในการจัดการระดับหนึ่งเนื่องจากต้องมีการรวมตารางหลักและตารางจากคลาสสืบทอดเพื่อให้ได้ข้อมูลของพนักงานคนใดๆ ให้ครบถ้วน ในขณะที่แบบที่ 2 เป็นการจัดเก็บแอทริบิวต์ที่เหมือนกันในทั้ง 2 ตาราง ได้แก่มีทั้งรหัสพนักงาน (id) ชื่อ (name) และวันเดือนปีเกิด (dob) และเพิ่มแอทริบิวต์ประจำสำหรับรีเลชันที่แทนพนักงานแต่ละประเภท การประยุกต์ในลักษณะนี้มีความสะดวกมากกว่า แต่อาจไม่รองรับการทำงานบางประเภท เช่นในกรณีที่พนักงาน 1 คนเป็นพนักงานที่ทำงานทั้งแบบเซ็นสัญญาและแบบรายชั่วโมง การจัดเก็บข้อมูลจะมีความซ้ำซ้อน หรือในกรณีที่พนักงานไม่ได้เป็นประเภทใดประเภทหนึ่งเลยก็จะไม่สามารถจัดเก็บข้อมูลได้ การสร้างรีเลชันด้วย SQL ใช้คำสั่งเดียวกันกับการสร้างรีเลชัน และการใช้เงื่อนไขบังคับอื่นๆ เพียงแต่ต้องสร้างรีเลชันให้สอดคล้องกับคลาสเท่านั้น

4.7 กรณีศึกษา

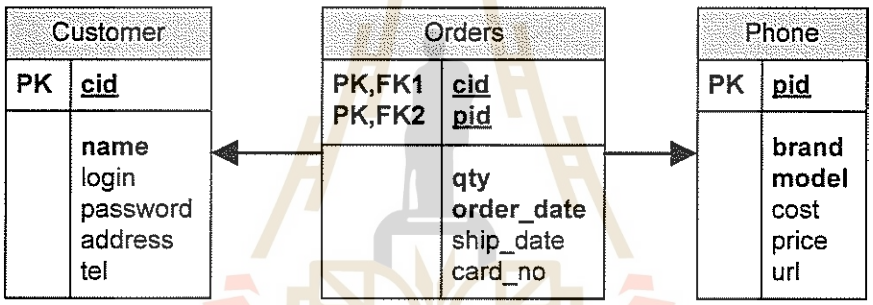
ตัวอย่างกรณีศึกษาในบทที่ 3 สามารถนำมาสร้างเป็นแบบจำลองเชิงสัมพันธ์โดยใช้แผนภาพแบบจำลองฐานข้อมูลเชิงสัมพันธ์ที่สัมพันธ์กับ ER model โดยใช้หลักการแปลงจาก ER model เป็นแบบจำลองเชิงสัมพันธ์ดังได้อธิบายในบทนี้ การสร้างแบบจำลองเชิงสัมพันธ์ที่ซับซ้อนต้องเริ่มจากการค่อยๆ แปลงส่วนใดส่วนหนึ่งของ ER model จากนั้นค่อยๆ ขยายออกไปยังรีเลชันที่เกี่ยวข้องจนเสร็จสมบูรณ์ ซึ่งแสดงผลลัพธ์ของการแปลงได้ดังต่อไปนี้

7-Elephant

ER Model

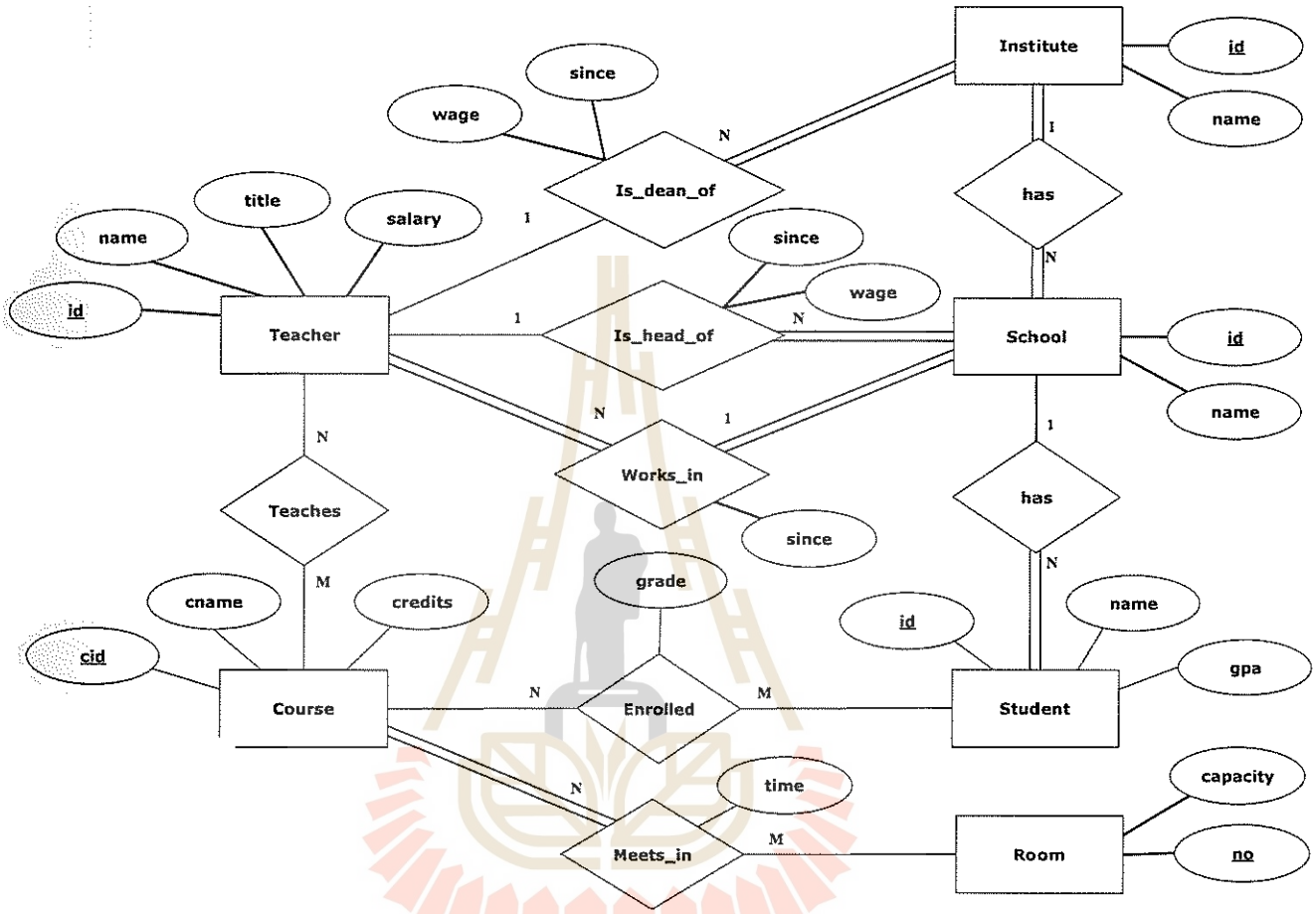


Relational database model

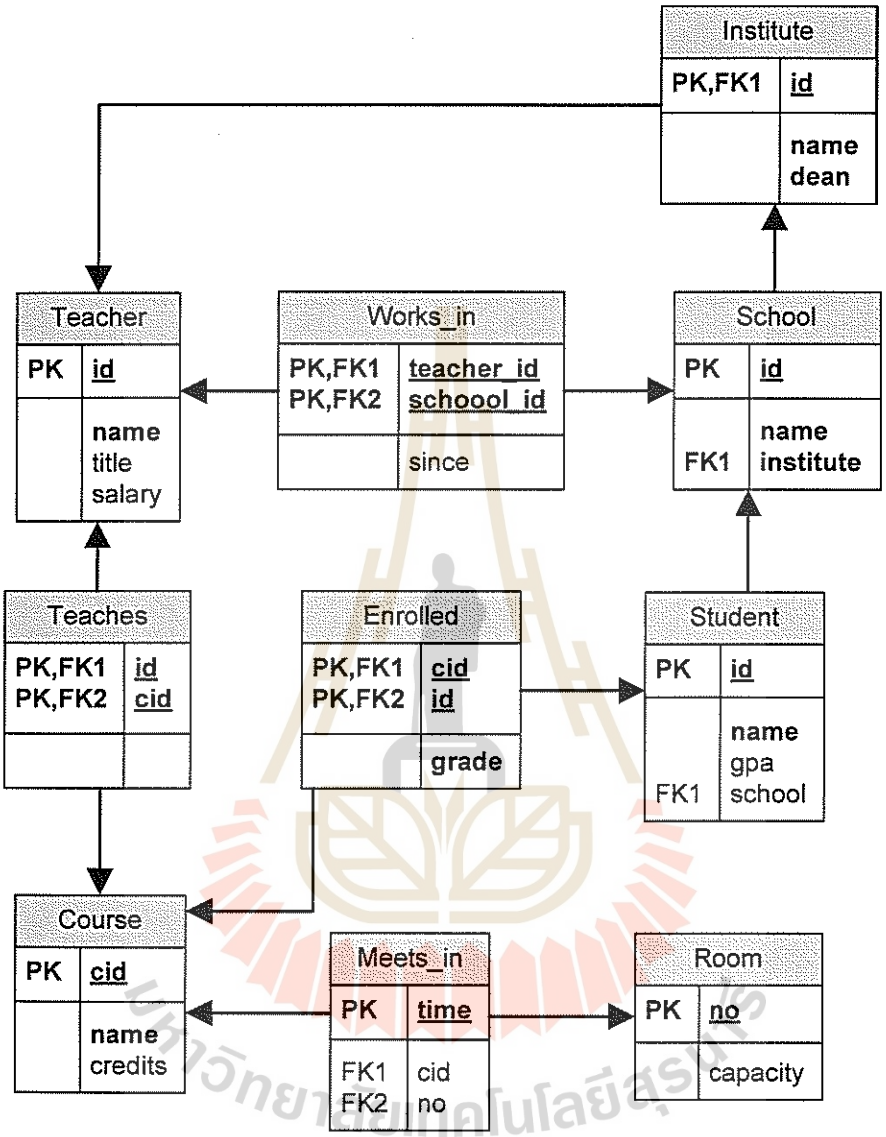


มหาวิทยาลัยเทคโนโลยีสุรบุรุษ

ER model

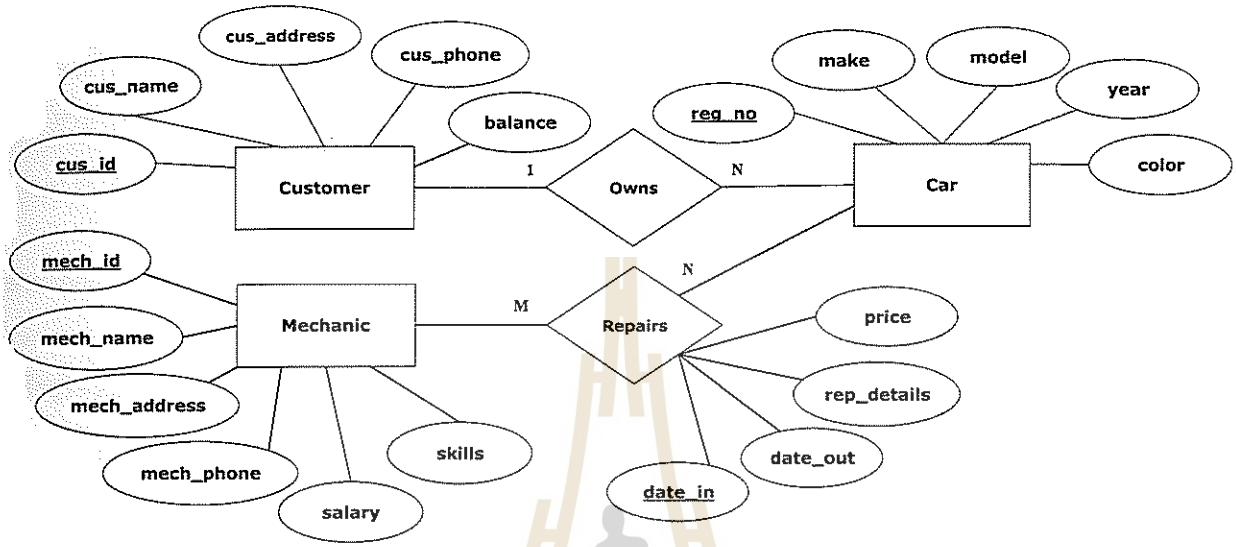


Relaitonal database model

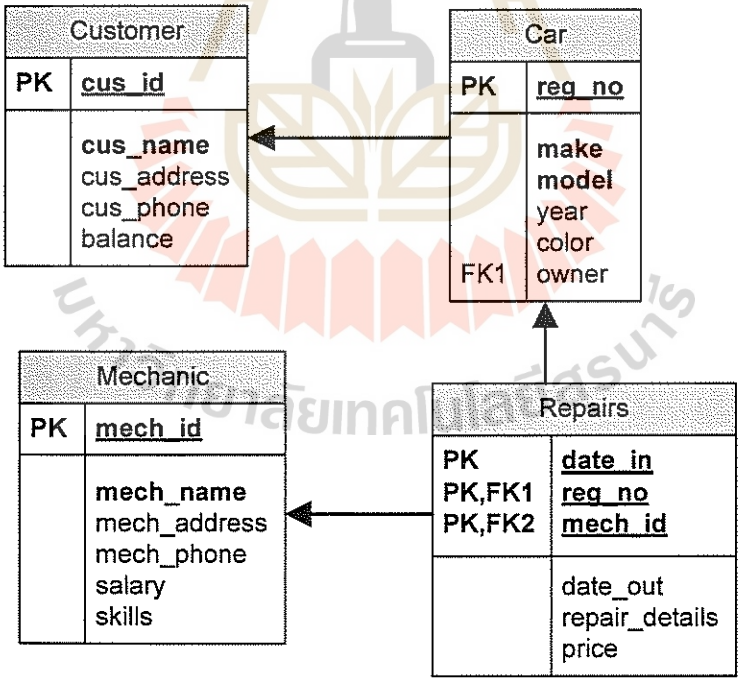


อู่เฉิดฉาย

ER model



Relational database model



4.8 แบบฝึกหัดท้ายบท

1. ให้นักศึกษาใช้วัตถุประสงค์ของบทเป็นแบบฝึกหัดท้ายบท**
2. ออกแบบ Logical Model ของโครงงาน เช่า-ฉัน-สิ ในภาคผนวก

บทที่ 5 การทำให้เป็นรูปแบบบรรทัดฐาน (Normalization)

วัตถุประสงค์

- สามารถอธิบายถึงความจำเป็นในการทำให้เป็นรูปแบบบรรทัดฐานได้
- สามารถอธิบายเกี่ยวกับภาวะพึ่งพิงทางฟังก์ชันได้
- สามารถอธิบายเกี่ยวกับรูปแบบบรรทัดฐานขั้นที่ 1-3 BCNF และ ขั้นที่ 4-5 ได้
- สามารถทำให้ตารางข้อมูลอยู่ในรูปแบบบรรทัดฐานขั้นที่ 1-3 BCNF และ ขั้นที่ 4-5 ได้

คำสำคัญ: ภาวะพึ่งพิงทางฟังก์ชัน (functional depency); determinant; dependent; full dependency; partial dependency; transitive dependency; multi-value dependency; trivial dependency; non-tirvial dependency; join dependency; การทำให้อยู่ในรูปแบบบรรทัดฐาน (normalization); รูปแบบบรรทัดฐาน (normal form); first normal form; second normal form; third normal form; Boyce-Codd normal form; forth normal form; fifth normal form

5.1 ความหมายของคำที่เกี่ยวข้อง

5.1.1 FD (Functionally Dependent หรือ Function Dependency)

เป็นความสัมพันธ์ระหว่าง attribute แบบ m:1 หรือ 1:1 โดย attribute ทางขวามีฟังก์ชันขึ้นกับ attribute ทางซ้าย

กรณี 1:1 ค่าของ attribute ทางซ้าย 1 ค่า จะสัมพันธ์กับค่าของ attribute ทางขวา 1 ค่า

กรณี m:1 ค่าของ attribute ทางซ้ายมากกว่า 1 ค่าจะสัมพันธ์กับค่าของ attributeทางขวา 1 ค่า

นิยาม

"กำหนดให้ x และ y เป็น attribute ของ relation R จะได้ว่า y มีฟังก์ชันขึ้นกับ x(y เป็น FD กับ x) ก็ต่อเมื่อถ้า 2 tuples ใน R มีค่าของ x ตรงกันแล้ว ทั้ง 2 tuples นี้ จะต้องมีค่าของ y ตรงกันด้วย"

ถ้า relation R มี attribute x, y จะได้ว่า

R.x → R.y

อ่านว่า x determine y

หรือ y depends on x

หมายความว่า x จะเป็นตัวกำหนดค่า (determine) ของ y หรือค่าของ y จะขึ้นอยู่กับค่าของ x (y depends on x หรือ y เป็น FD กับ x) เราจะเรียก x ว่า determinant

หลักในการเขียน Functional Dependency

FDs : determinant-attribute → dependency-attribute

FDs : PERSON_ID → PERSON_NAME

ประเภทของ Functional Dependency

- FD ที่เกิดจากความสัมพันธ์ระหว่าง Determinant และ Dependency อย่างละ 1 ค่า

  PERSON_ID → PERSON_NAME
- FD ที่เกิดจากความสัมพันธ์ระหว่าง Determinant 1 ค่า กับ Dependency หลายค่า

  PERSON_ID → FNAME, LNAME, ADDRESS, BIRTH_DATE, ISSUE_DATE
- FD ที่มีความสัมพันธ์ 2 ทาง ที่ทั้ง Determinant และ Dependency ต่างสามารถทำหน้าที่ของอีกฝ่ายหนึ่งได้

  PROJECT_NO → MANAGER

  MANAGER → PROJECT_NO

  PROJECT_NO ← → MANAGER
- FD ที่ต้องใช้ Determinant มากกว่า 1 ค่าเพื่ออ้างถึง Dependency

  PRODUCT_LINE, ITEM_NO → USED_QTY

5.1.2 Full FD (Full Functional Dependence)

นิยาม

"attribute y ของ relation R จะเป็น Full FD บน attribute x ของ relation R ถ้า y เป็น FD กับ x และไม่เป็น FD กับ subset ใด ๆ ของ x"

หรืออาจกล่าวได้ว่าเป็น FD ที่มี Determinant ที่มีขนาดเล็กที่สุดและสามารถระบุถึง Dependency ได้ ดังตัวอย่างต่อไปนี้

- D1 : PERSON_ID → ADDRESS

  D2 : PERSON_ID, PERSON_NAME → ADDRESS

- D3 : PRODUCT_LINE, ITEM_NO → USE_QTY

  D4 : PRODUCT_LINE, ITEM_NO, MANAGER → USE_QTY

ดังนั้น จากตัวอย่างสามารถสรุปได้ว่า D1 กับ D3 ถือเป็น Full FD

5.1.3 Partial Dependency

ความสัมพันธ์แบบนี้จะเกิดขึ้นได้ก็ต่อเมื่อ relation หนึ่ง ๆ มีคีย์หลักเป็นคีย์ผสม (composite key) นั่นคือ คีย์หลักของ relation นั้น ๆ ประกอบด้วย attribute หลาย attribute ความสัมพันธ์ระหว่างค่าของ attribute แบบบางส่วนเกิดขึ้นเมื่อ attribute บางส่วนของคีย์หลักสามารถระบุค่าของ attribute อื่น ๆ ที่ไม่ใช่คีย์หลักของ relation ได้ (Non-key attribute) ซึ่งความสัมพันธ์แบบนี้จะทำให้เกิดปัญหาในเรื่องของความซ้ำซ้อน และการปรับปรุงข้อมูล

5.1.4 Transitive Dependency

attribute ที่มีคุณสมบัติเป็นคีย์หลักจะสามารถระบุค่าของทุก attribute ในแต่ละ tuples ได้ อย่างไรก็ตาม ในบาง relation อาจจะมีกรณี attribute อื่น (Nonkey attribute) ที่สามารถระบุค่าของ attribute อื่น ๆ ใน tuples ได้ ลักษณะของความสัมพันธ์ในการระบุค่า attribute แบบนี้ เรียกว่า ความสัมพันธ์ระหว่างค่าของ attribute แบบนี้ว่า Transitive Dependency ดังตัวอย่างเช่น

รหัสนักศึกษา → รหัสสาขาวิชา

รหัสสาขาวิชา → ชื่อสาขาวิชา

จะเห็นได้ว่า รหัสสาขาวิชา → ชื่อสาขาวิชา เป็น Transitive Dependency ซึ่งถือว่าเป็นส่วนที่เกินมาโดยไม่จำเป็น เนื่องจาก รหัสนักศึกษา → รหัสสาขาวิชา ก็สามารถดึงข้อมูลในส่วนของชื่อสาขาวิชาได้ ดังนั้น เราจะไม่เก็บชื่อสาขาวิชาไว้ในตารางนักศึกษา

5.1.5 Multivalued Dependency

การขึ้นต่อกันหลายค่า (Multivalued Dependency) ในความสัมพันธ์ R มี attribute A, B, C เราจะกล่าวว่า attribute B เป็นการขึ้นต่อกันหลายค่าบน A ก็ต่อเมื่อเซทของค่าของ attribute B ในความสัมพันธ์ R ที่มีความสัมพันธ์ตรงกับคู่ของ <A, C> นั้น จะไม่ขึ้นกับค่าของ C เขียนแทนด้วย

A ----->> B และจะมีลักษณะที่เห็นได้ดังนี้

1 attribute A ค่าหนึ่ง จะเป็นตัวกำหนดกลุ่มของค่า attribute B คือ เมื่อ 2 tuples ในความสัมพันธ์ R มีค่า A เดียวกัน ไม่จำเป็นต้องมีค่า B เหมือนกัน แต่ค่าของ B จะต้องอยู่ในกลุ่มของค่า B ที่ถูกกำหนดโดย A

2 การเปลี่ยนแปลงค่าใดใน attribute C จะไม่มีผลกระทบต่อค่า B

3 สอง tuples ในความสัมพันธ์ R ที่มีค่า B เหมือนกันไม่จำเป็นต้องมีค่า A เดียวกัน

4 ค่าของ attribute C สองค่าที่มีความสัมพันธ์กับค่า A เดียวกันจะต้องสัมพันธ์กับค่าของ B ในกลุ่มเดียวกันและเป็นกลุ่มที่ถูกกำหนดโดยค่า A นั้น ๆ

หรืออาจกล่าวได้ว่า ค่าของ Determinant 1 ค่าสามารถระบุค่าของ attribute ที่ทำหน้าที่เป็น Dependency ได้ตั้งแต่ 2 attribute ขึ้นไป ซึ่งอยู่ในรูปของชุดข้อมูล

EMPLOYEE# → → DEPARTMENT#, PROJECT#

5.1.6 Trivial FD's

FD (Function Dependancy) เป็น Trivial FD's ก็ต่อเมื่อ attribute ทางด้านขวาอยู่ใน attribute ทางด้านซ้าย

ตัวอย่างเช่น

SUPPLIER(SNAME,ADDRESS,ITEM,PRICE)

SNAME → ADDRESS

SNAME,ITEM → PRICE

5.2 การนอร์มัลไลซ์ (Normalization)

หลังจากที่ผู้ออกแบบได้ขอบเขตข้อมูลทั้งหมดที่ต้องการเก็บแล้ว ซึ่งโดยมากเกิดจากรูปแบบ รายงานบ้าง รูปแบบใบเสร็จรับเงินบ้าง รูปแบบใบส่งสินค้าบ้าง โดยมากมักจะเหมารวมเอาว่านั้นคือ รูปแบบของตารางที่ต้องการเก็บข้อมูล ซึ่งนำมาซึ่งความซ้ำซ้อนของข้อมูลในรายกรณี และทำให้มีขนาดใหญ่เกินความจำเป็น ส่วนที่ซ้ำซ้อนปัญหาของรีเลชั่นที่เกิดขึ้นเหล่านี้ สามารถขจัดได้ด้วย " ขบวนการ Normalization" ซึ่งแนวคิดนี้ถูกคิดค้นโดย

E.F.Codd ซึ่งเป็นกระบวนการที่นำเค้าร่างของ relation มาทำให้อยู่ในรูปแบบที่เป็นบรรทัดฐาน (Normal Form) เพื่อให้แน่ใจว่าการออกแบบเค้าร่างของ relation เป็นการออกแบบที่เหมาะสม

ให้ทำการตรวจสอบเอนทิตีต่าง ๆ ให้อยู่ในกฎนอร์มัลไลเซชันซึ่งประกอบด้วย 1NF, 2NF, 3NF, BCNF, 4NF, 5NF ซึ่งจะได้กล่าวในหัวข้อต่อไป ประโยชน์ก็คือ

1. ลดที่ว่างที่ต้องใช้ในการเก็บข้อมูล
2. ลดความผิดพลาด ความไม่ตรงกันของข้อมูลในฐานข้อมูล
3. ลดการเกิดอะนอร์มัลไลของการลบและแก้ไขข้อมูล
4. เพิ่มความคงทนแก่โครงสร้างฐานข้อมูล

ในทางปฏิบัติการทำนอร์มอลไลซ์จะเริ่มจาก E-R Model ก่อน แล้วจึงทำการ map จาก E-R Model เป็น relation แบบ 1NF ก่อน โดยให้ attributes ที่เกี่ยวข้องกันจะอยู่ในตารางเดียวกัน สำหรับ application ใหญ่ ๆ มี attributes ประมาณ 500 attributes ใช้ E-R Model จะได้ 1NF ประมาณ 80 ตาราง เมื่อทำถึง 5NF จะได้ไม่เกิน 100 ตาราง ในกรณีได้ตารางเป็นนอมัลไลเซชันที่สมบูรณ์แล้ว สิ่งที่ต้องระวังคือไม่แตกตารางนั้นย่อยลงไปอีก

ระดับนอร์มัลไลเซชัน

นอร์มัลไลเซชัน เป็นกระบวนการเพื่อพัฒนาการ เชื่อมต่อของข้อมูลเพื่อแก้ปัญหาของรีเลชั่น ที่ว่าการออกแบบฐานข้อมูลทั้งทางตรรกะ และทางกายภาพที่ได้ออกมาใช้ได้หรือยัง การนอร์มัลไลเซชันแบ่งออกได้เป็นหลายระดับ ได้แก่

5.2.1 First Normal Form (1NF)

ทุก ๆ field ในแต่ละ record จะเป็น single value นั่นคือ ในตารางหนึ่ง ๆ จะไม่มี ค่าของกลุ่ม ข้อมูลที่ซ้ำกัน

(Repeating Group) ตัวอย่างเช่น ตารางดังต่อไปนี้

ตารางที่มีลักษณะข้อมูลเป็น Repeating group

รหัสนักศึกษา	ชื่อ	นามสกุล	รหัสวิชาที่ลงทะเบียน
001	สมชาย	สมใจนึก	204-101 204-204 204-205
002	ธีรชาย	บุญมาศ	204-102 204-204

เราสามารถทำให้อยู่ในรูป 1NF ได้ดังนี้

รหัสนักศึกษา	ชื่อ	นามสกุล	รหัสวิชาที่ลงทะเบียน
001	สมชาย	สมใจนึก	204-100
001	สมชาย	สมใจนึก	204-204
001	สมชาย	สมใจนึก	204-125
002	ธีรชาย	บุญมาศ	204-102
002	ธีรชาย	บุญมาศ	204-204

จะเห็นว่าการเก็บข้อมูลแบบนี้เป็นการสิ้นเปลืองโดยใช่เหตุ เพราะมีค่าของกลุ่มข้อมูลที่ซ้ำกันมากมาย เพราะนักศึกษาคนหนึ่ง สามารถลงทะเบียนได้มากกว่าหนึ่งวิชา

สรุปก็คือ นอร์มัลไลเซชันระดับที่ 1 (First normal form : 1NF) เป็นการขจัดแอตตริบิว หรือกลุ่มแอตตริบิวที่ซ้ำกันไปอยู่ในเอนทิตีลูก เพื่อแต่ละรายการในเอนทิตี ไม่มีค่าของแอตตริบิวหรือค่าของกลุ่มแอตตริบิวที่ซ้ำกัน

สำหรับ 1NF จะมีข้อเสียงในการแก้ไข การลบ และการเพิ่มข้อมูล ดังนี้

1) การแก้ไขข้อมูล (Update) เนื่องจากมีข้อมูลอยู่หลาย tuples จะต้องแก้ไขทุก tuples นั่นคือต้องมีการแก้ไขข้อมูลมากกว่าหนึ่งแห่ง
2) การลบข้อมูล (Delete) ถ้าต้องการลบข้อมูลบางส่วนออกไป จะทำให้ลบข้อมูลอื่นออกไปด้วยโดยไม่ตั้งใจ
3) การเพิ่มข้อมูล (Insert) อาจจะทำให้ไม่สามารถเพิ่มข้อมูลบางอย่างไม่ได้ หรือเพิ่มแล้วขัดแย้งกับข้อมูลเดิม

5.2.2 Second Normal Form (2NF)

ต้องเป็น First Normal Form (1NF) และต้องมี key (บางตำรา อาจจะเรียกว่า index) ที่ทุก Non-key จะต้องขึ้นอยู่

(depends on) กับ key นี้ และมีเพียง key เดียวในหนึ่งตาราง ซึ่งเรียกว่า Primary Key การที่ทุกตาราง (Table) ต้องมี

Key ก็เพราะเราต้องการให้แน่ใจว่าทุกข้อมูลใน record ต่าง ๆ สามารถค้นหาได้โดยใช้ key

สรุปก็คือ นอร์มัลไลเซชันระดับที่ 2 (Second normal form : 2NF) เป็นการขจัดแอตตริบิวที่ไม่ขึ้นกับทั้งส่วนของ

คีย์หลักออกไป เพื่อให้แอตตริบิวอื่นทั้งหมดขึ้นตรงกับส่วนที่เป็นคีย์หลักทั้งหมดเท่านั้น

นิยาม

เป็น First Normal Form (1NF) และทุก Non-key จะต้องขึ้นอยู่ (depends on) กับ key อย่างสมบูรณ์ (Full FD) หรืออาจกล่าวได้ว่าไม่มี Non-key ที่สามารถ imply ถึง Non-key ตัวอื่นได้ เช่น A → (B , C) and B → C รวมไปถึงการที่ Non-Key บางตัวที่ขึ้นกับ บางส่วนของ Key

ตัวอย่างเช่น

ABC (*ชิ้นส่วน , ชื่อโกดัง* , จำนวน , ที่อยู่โกดัง)

FD = { ชิ้นส่วน และ ชื่อโกดัง → จำนวน , ชื่อโกดัง → ที่อยู่โกดัง }

เนื่องจาก (ชิ้นส่วน และชื่อโกดัง) เป็น key แต่ (ที่อยู่โกดัง) ไม่ได้ขึ้นตรงกับ key (fully-depended on key) ดังนั้น จึงไม่ใช่ 2NF ดังนั้น จะต้องทำการแตกรีเลชั่น เพื่อลดปัญหาความซ้ำซ้อนของข้อมูลเป็นดังนี้

สินค้า (*ชิ้นส่วน, ชื่อโกดัง*, จำนวน) โดยให้ ชิ้นส่วนและชื่อโกดัง เป็นคีย์หลัก

โกดัง (*ชื่อโกดัง*, ที่อยู่โกดัง) โดยให้ ชื่อโกดัง เป็นคีย์หลัก

5.2.3 Third Normal Form (3NF)

นอร์มัลไลเซชันระดับที่ 3 (Third normal form : 3NF) คือ ขบวนการที่พยายามขจัดสภาพของ Transitive Dependency ออกไป

นิยาม นอร์มัลไลเซชันระดับที่ 3 (Third normal form : 3NF)

ต้องเป็น Second Normal Form (2NF) และ ไม่มี Transitive dependence หรือ เป็นการขจัดแอตตริบิวที่ไม่เป็นคีย์ที่ขึ้น (Transitive dependent) ตรงกับแอตตริบิวอื่นที่ไม่ใช่คีย์หลักออกไป เพื่อให้แอตตริบิวที่ไม่ใช่คีย์หลักต้องขึ้นตรงกับทั้งส่วนที่เป็นคีย์หลัก และไม่ขึ้นกับแอตตริบิวอื่นที่ไม่ใช่คีย์หลัก

นิยาม ของ Transitive dependency

การไม่ขึ้นตรงกับคีย์หลัก (Transitively Dependency) ถ้าในความสัมพันธ์ R มีคีย์หลักคือ K และแอตตริบิว A และ B จะกล่าวว่าแอตตริบิว B ไม่ขึ้นตรงกับคีย์หลัก

เมื่อ K ----------> A และ A ----------> B และ A ---/--> K

ตัวอย่างการทำตารางให้เป็น 3NF

ผู้บริหาร (<u>เลขประจำตัว</u> , ชื่อนามสกุล , ที่อยู่ , ตำแหน่ง, ยี่ห้อรถประจำตำแหน่ง)

FD = { เลขประจำตัว → ชื่อนามสกุล , ที่อยู่ , ตำแหน่ง

ตำแหน่ง → ยี่ห้อรถประจำตำแหน่ง }

ในตัวอย่างจะเห็นได้ว่า set ของ ผู้บริหาร (<u>เลขประจำตัว</u> , ชื่อนามสกุล , ที่อยู่, ตำแหน่ง, ยี่ห้อรถประจำตำแหน่ง) นี้ ยังไม่ใช่ 3NF เพราะ เลขประจำตัว→ตำแหน่ง ตำแหน่ง→ ยี่ห้อรถประจำตำแหน่ง ดังนั้น ควรจะแยกเซ็ทผู้บริหาร ออกเป็น 2 เซ็ท คือ

<u>3NF</u>: ผู้บริหาร (<u>เลขประจำตัว</u> , ชื่อนามสกุล , ที่อยู่, ตำแหน่ง)

ตำแหน่งบริหาร (<u>ตำแหน่ง</u> , ยี่ห้อรถประจำตำแหน่ง)

5.2.4 BCNF (Boyce/Codd Normal Form)

นิยาม

"ต้องเป็น 3NF และไม่มี attribute อื่นในรีเลชันที่สามารถระบุค่าของ attribute ที่เป็นคีย์หลัก หรือส่วนหนึ่งส่วนใดของคีย์หลักในกรณีที่คีย์หลักเป็นคีย์ผสม"

โดยทั่วไปรูปแบบ BCNF จะอยู่ในรูปแบบ 3NF แต่ไม่จำเป็นเสมอไปที่รูปแบบ 3NF จะอยู่ในรูปแบบ BCNF ทั้งนี้เนื่องจากรูปแบบนี้เป็นการขยายขอบเขตของรูปแบบ 3NF ให้เหมาะสมยิ่งขึ้น โดยรูปแบบที่ต้องทำให้เป็น BCNF มักจะมีคุณสมบัติ ดังนี้

เป็นรีเลชันที่มีคีย์คู่แข่งหลายคีย์ (Multiple Candidate Key) โดยที่

คีย์คู่แข่งเป็นคีย์ผสม (Composite Key) และ

คีย์คู่แข่งนั้นมีบางส่วนซ้ำซ้อนกัน (Overlapped) ๖มี attribute บางตัวร่วมกันอยู่)

จากตัวอย่างตารางต่อไปนี้

รหัสนักศึกษา	ชื่อนักศึกษา	รหัสวิชา	เกรด
001	Jane	C01	A

001	Jane	C02	B
002	Timmy	C01	C
002	Timmy	C02	B
002	Timmy	C03	D
003	Nan	C04	D

จากตารางนี้จะได้ว่า

รหัสนักศึกษา, รหัสวิชา → เกรด

ชื่อนักศึกษา, รหัสวิชา → เกรด

รหัสนักศึกษา → ชื่อนักศึกษา

รหัสพนักงาน → ชื่อพนักงาน

ตารางนี้มี 4 determinants คือ (รหัสนักศึกษา, รหัสวิชา) (ชื่อนักศึกษา, รหัสวิชา) (รหัสพนักงาน) และ (ชื่อพนักงาน) แต่ตารางนี้มีเพียง 2 candidate keys คือ (รหัสนักศึกษา, รหัสวิชา) และ (ชื่อนักศึกษา, รหัสวิชา) สำหรับรหัสนักศึกษา และชื่อนักศึกษาไม่ใช่ candidate key เราจะพบว่าตารางนี้เป็น 3NF ที่ไม่ดีพอเนื่องจากตารางนี้มี candidate key 2 keys ด้วยกัน เราสามารถเลือกอันใดอันหนึ่งเป็น primary key ได้ นอกจากนี้ candidate key ทั้งสองยังเป็น composite key และ overlap กันอยู่ เพราะมีรหัสวิชาร่วมกัน ทำให้มีข้อมูลหนึ่งชุดปรากฏหลายหนถึงแม้จะเป็น 3NF แล้วก็ตาม จึงควรปรับต่อไปเพื่อลดความซ้ำซ้อนนี้ลง และเพื่อแก้ปัญหาการ insert delete และ update ข้อมูลในตาราง โดยใช้วิธีการของ BCNF ดังนี้

1. ถ้ามี Transitive FD ให้ขจัด Transitive FD ทิ้งไป
2. attribute ตัวใดที่เป็น determinant ทำให้เป็น candidate key

จากตารางข้างต้นเราสามารถแยกออกเป็นตารางใหม่ ได้ 2 ตาราง ดังนี้

นักศึกษา (รหัสนักศึกษา, ชื่อนักศึกษา)

เกรด (รหัสนักศึกษา, รหัสวิชา, เกรด)

หรือ

นักศึกษา (รหัสนักศึกษา, ชื่อนักศึกษา)

เกรด (ชื่อนักศึกษา, รหัสวิชา, เกรด)

5.2.5 4NF (Forth Normal Form)

นิยาม

"ต้องอยู่ในรูปแบบ BCNF และเป็นรีเลชันที่ไม่มีความสัมพันธ์ในการระบุค่าของ attribute แบบหลายค่า โดยที่ attribute ที่ถูกระบุค่าเหล่านี้ไม่มีความสัมพันธ์กัน (Independently Multivalued Dependency)"

จากตัวอย่างข้อมูลต่อไปนี้

Project

Person	Project	Part	QtyUsed	HrsSpent
John	P1	Nut	11	7
Emmy	P1	Bolt	7	17
John	P1	Bolt	7	7
Emmy	P1	Nut	11	17
John	P2	Bolt	7	32
Jim	P2	Screw	9	45
John	P2	Screw	9	32
Jim	P2	Bolt	7	45

สามารถเขียน FDs ได้ดังต่อไปนี้

Project →→ Part, QtyUsed

Project, Person → HrsSpent

Project, Part → QtyUsed

จะเห็นได้ว่าตารางนี้มีขึ้นต่อกันแบบหลายค่า (Multivalued Dependency) ซึ่งได้แก่ Part และ QtyUsed ที่ข้อมูลจะมีลักษณะเป็นคู่ ๆ ตาม Project ดังนั้น ตารางนี้จึงยังไม่อยู่ในรูปแบบ BCNF จะต้องทำการแตกตารางออกเป็น 2 ตาราง ดังนี้

Person-Works-On-Project

Project	Person	HrsSpent
P1	John	7
P1	Emmy	17
P2	John	32

P2	Jim	45

FDs : Project, Person → HrsSpent

Part-Used-In-Project

Project	Part	QtyUsed
P1	Nut	11
P1	Bolt	7
P2	Bolt	7
P2	Screw	9

FDs : Project, Part → QtyUsed

5.2.6 5NF (Fifth Normal Form)

5NF หรือเรียกว่า Project-Join Normal Form (PJ/NF)

นิยาม

"ต้องอยู่ในรูปแบบ 4NF และไม่มี Symmetric Constraint กล่าวคือ หากมีการแตกรีเลชันออกเป็นรีเลชันย่อย (Projection) และเมื่อทำการเชื่อมโยงรีเลชันย่อยทั้งหมด (Joint) จะไม่ก่อนให้เกิดข้อมูลใหม่ที่ไม่เหมือนรีเลชั่นเดิม (Spurious Tupes)

ในการแตกรีเลชันออกมาจากรูปแบบ 4NF นั้น ถ้าทำการเชื่อมโยงรีเลชันย่อยนั้นใหม่ หากไม่มีข้อมูลที่แตกต่างไปจากรีเลชันเดิม ก็จะสามารถแตกรีเลชันนั้นได้ แต่ถ้าหากแตกเป็นรีเลชันย่อยแล้วเกิดข้อมูลไม่เหมือนกับรีเลชันเดิม ก็ไม่ควรแตกรีเลชัน และให้ถือว่ารีเลชันเดิมอยู่ใน 5NF แล้ว

5.3 ประเด็นที่ควรคำนึงถึงในการทำให้อยู่ในรูปแบบ Normal Form

5.3.1 การแตกรีเลชันมากเกินไป (Overnormalization)

วัตถุประสงค์ในการทำให้เป็นรูปแบบนอร์มัลไลซ์ก็คือ เพื่อลดปัญหาความซ้ำซ้อนของข้อมูล และลดปัญหาในเรื่องการเพิ่ม การปรับปรุง หรือลบข้อมูล โดยทั่วไปแล้วการออกแบบในระแนวความคิด ผู้ออกแบบจะพยายามวิเคราะห์ให้อยู่ในรูปแบบ 3NF แต่ถ้ามีกรณีของปัญหาที่จำเป็นต้องทำต่อไปถึงรูปแบบของ BCNF หรือ 4NF หรือ 5NF (ซึ่งเกิดขึ้นน้อยมากในทางปฏิบัติ) แต่อย่าพยายามแตกรีเลชันให้มากเกิดความจำเป็น เพราะการแตกรีเลชันออกมากเกินไปนั้น จะมีผลต่อประสิทธิภาพในการทำงานของฐานข้อมูลนั้น ๆ เช่น ในการค้นหาข้อมูลจะใช้เวลามาก เป็นต้น

5.3.2 การดีนอร์มัลไลเซชัน (Denormalization)

ในกรณีที่บางรีเลชันถูกออกแบบโดยการไม่ทำให้อยู่ในรูปแบบบรรทัดฐานที่เป็นไปตามกฎเกณฑ์ที่กำหนดไว้ เช่น ควรปรับให้อยู่ในรูปแบบ 3NF แต่หยุดอยู่เพียงแค่ 2NF เป็นต้น ทั้งนี้ อาจเป็นเพราะเหตุผลในเรื่องของประสิทธิภาพในการเรียกดู หรือค้นหาข้อมูล และยอมให้เกิดความซ้ำซ้อนของข้อมูลได้

ด้วยเหตุที่การดีนอร์มัลไลเซชัน อาจทำให้เกิดปัญหาความซ้ำซ้อนของข้อมูล จึงควรมีการระบุสาเหตุและวิธีการในการปรับปรุงข้อมูลในโปรแกรมประยุกต์ใช้งาน เพื่อป้องกันไม่ให้เกิดปัญหาข้อมูลไม่ถูกต้อง อีกประเด็นที่ยอมให้มีการดีนอร์มัลไลซ์หรือไม่ ก็คือ ถ้าข้อมูลในรีเลชันนั้น ๆ ส่วนใหญ่ จะเป็นการเรียกดูข้อมูล (Select) มากกว่าการเพิ่ม ปรับปรุง หรือลบข้อมูล ก็อาจจะดีนอร์มัลไลเซชันได้ ถ้าคิดว่าการออกแบบลักษณะนี้จะเพิ่มประสิทธิภาพในการทำงานของฐานข้อมูล และไม่มีปัญหาด้านความไม่ถูกต้องของข้อมูลที่ซ้ำซ้อนกันได้

แบบฝึกหัด

1. ให้ปรับโครงสร้างตารางข้อมูลการลงทะเบียนเรียนของนักศึกษาต่อไปนี้ ให้อยู่ในรูป Normal Form จนถึง 5NF และเขียน FD ในแต่ละ Normal Form ด้วย (ให้กำหนด Primary Key ตามความเหมาะสม)

รหัสนักศึกษา	ชื่อ	สาขา	สำนัก	รหัสวิชา	ชื่อวิชา	หน่วยกิต	เกรด	อาจารย์ผู้สอน	ห้องพักอาจารย์
B4301	แสง	เคมี	วิทยาศาสตร์	C101	เคมี 1	3	B	สมชาย	4071
				C702	คณิตศาสตร์	3	A	สมศรี	4052
				C111	คอมพิวเตอร์	4	B	สมศรี	4052
B4303	ทอง	คณิตศาสตร์	วิทยาศาสตร์	C702	คณิตศาสตร์	3	B	สมศรี	4052
				C101	เคมี 1	3	A	สมชาย	4071
				C811	การสื่อสาร	2	A	ธนดล	4711
B4305	ดวง	เคมี	วิทยาศาสตร์	C101	เคมี 1	3	A	สมชาย	4071
B4307	เด่น	สารสนเทศ	เทคโนโลยีสังคม	C111	คอมพิวเตอร์	4	A	สมศรี	4052
				C702	คณิตศาสตร์	3	B	สมศรี	4052
				C811	การสื่อสาร	2	A	ธนดล	4711

Enterprise Rule : นักศึกษา 1 คน ลงทะเบียนได้หลายวิชา, แต่ละวิชามีอาจารย์ผู้สอนได้เพียงคนเดียว, อาจารย์แต่ละคนสามารถสอนได้มากกว่า 1 วิชา, อาจารย์แต่ละคนมีห้องพักเพียงห้องเดียว

2. จากตารางทักษะของพนักงานกำหนดให้ทุก attribute เป็นคีย์หลัก จงแสดงว่าสามารถแตกตารางต่อไปได้อีกหรือไม่

รหัสพนักงาน	ทักษะ	งานที่ทำ
P1	คอมพิวเตอร์	Prog1
P1	คอมพิวเตอร์	Auto1
P1	เครื่องกล	Auto1

P1	เครื่องกล	Auto2
P2	เครื่องกล	Auto1
P3	คอมพิวเตอร์	Auto1

บทที่ 6 พีชคณิตเชิงสัมพันธ์และแคลคูลัสเชิงสัมพันธ์ (Relational Algebra & Relational Calculus)

วัตถุประสงค์

- สามารถอธิบายการดำเนินการกับรีเลชันด้วยพีชคณิตเชิงสัมพันธ์และสร้างรีเลชันจากพีชคณิตเชิงสัมพันธ์ได้
- สามารถอธิบายการดำเนินการกับรีเลชันด้วยแคลคูลัสเชิงสัมพันธ์แบบทูเพิลและสร้างรีเลชันจากแคลคูลัสเชิงสัมพันธ์แบบทูเพิลได้
- สามารถอธิบายการดำเนินการกับรีเลชันด้วยแคลคูลัสเชิงสัมพันธ์แบบโดเมนและสร้างรีเลชันจากแคลคูลัสเชิงสัมพันธ์แบบโดเมนได้

คำสำคัญ: พีชคณิตเชิงสัมพันธ์ (relational algebra); แคลคูลัสเชิงสัมพันธ์ (relational calculus); ภาษาสอบถาม (query language); ภาษาเชิงกระบวนคำสั่ง (procedural language); ภาษาเชิงไร้กระบวนคำสั่ง (non-procedural language); บริบูรณ์เชิงสัมพันธ์ (relationally complete); ซีเลคชัน (selection); โปรเจคชัน (projection); ผลคูณคาร์ทีเซียน (Cartesian product); ยูเนียน (union); ผลต่างของเซต (set difference); การเชื่อม (join); อินเตอร์เซกชัน (intersection); การหาร (division); แคลคูลัสเชิงสัมพันธ์แบบทูเพิล (tuple relational calculus); แคลคูลัสเชิงสัมพันธ์แบบโดเมน (domain relational calculus)

6.1 บทนำ

การแสดงและแก้ไขข้อมูลในฐานข้อมูลเชิงสัมพันธ์จำเป็นต้องอาศัยภาษาที่มีโครงสร้างรองรับแบบจำลองข้อมูลเชิงสัมพันธ์ เราเรียกภาษานี่ว่าภาษาสอบถาม (query language) ซึ่งใช้สำหรับแสดงและแก้ไขข้อมูลในระบบจัดการฐานข้อมูลเชิงสัมพันธ์ ในบทนี้อธิบายถึงพื้นฐานของภาษาสอบถามอันได้แก่ พีชคณิตเชิงสัมพันธ์ (relational algebra) และแคลคูลัสเชิงสัมพันธ์ (relational calculus) ที่ Codd ใช้อ้างอิงในการจัดการกับข้อมูลในฐานข้อมูลเชิงสัมพันธ์ พีชคณิตเชิงสัมพันธ์จัดเป็นพื้นฐานของภาษาสอบถามแบบ สำหรับภาษาสอบถามที่อยู่บนพื้นฐานของพีชคณิตเชิงสัมพันธ์และแคลคูลัสเชิงสัมพันธ์ที่นิยมมากที่สุดได้แก่ SQL ซึ่งอธิบายในบทถัดไป ภาษาเชิงกระบวนคำสั่ง (procedural language) หมายถึงภาษาที่สามารถสั่งเป็นลำดับขั้นตอน ในขณะที่แคลคูลัสเชิงสัมพันธ์เป็นพื้นฐานของภาษาสอบถามแบบ ภาษาไร้กระบวนคำสั่ง (non-procedural language) หมายถึงภาษาที่ไม่จำเป็นต้องสั่งเป็นลำดับขั้นตอนแต่สามารถได้ผลลัพธ์เช่นเดียวกัน ภาษาสอบถามใดๆ จะต้องมีความบริบูรณ์เชิงสัมพันธ์ (relationally complete) กล่าวคือหากแคลคูลัสเชิงสัมพันธ์สามารถสร้างรีเลชันใดๆ ได้ ภาษาสอบถามที่สร้างขึ้นต้องสามารถสร้างรีเลชันนั้นๆ ได้เช่นเดียวกัน

หัวข้อที่ 6.2 อธิบายถึงพีชคณิตเชิงสัมพันธ์ หัวข้อที่ 6.3 อธิบายถึงแคลคูลัสเชิงสัมพันธ์ซึ่งมีสองรูปแบบได้แก่แคลคูลัสเชิงสัมพันธ์แบบทูเพิล และแคลคูลัสเชิงสัมพันธ์แบบโดเมน ซึ่งได้อธิบายไว้ในหัวข้อที่ 6.3.1 และ 6.3.2 ตามลำดับ

6.2 พีชคณิตเชิงสัมพันธ์ (Relational Algebra)

พีชคณิตเชิงสัมพันธ์เป็นภาษาเชิงทฤษฎีที่ใช้ตัวดำเนินการกับรีเลชันหนึ่งรีเลชันหรือหลายรีเลชัน ผลลัพธ์ที่ได้คือรีเลชันเช่นเดียวกัน ซึ่งผลลัพธ์นี้สามารถนำมากระทำด้วยตัวดำเนินการได้อีกในลักษณะของการซ้อนกันของตัวดำเนินการเช่นเดียวกับการใช้ตัวดำเนินการทางคณิตศาสตร์ทั่วๆ ไป เช่น A + B - C เป็นต้น การที่ทั้งรีเลชันที่ถูกดำเนินการและผลลัพธ์ต่างก็เป็นรีเลชันนี้เป็นสมบัติของพีชคณิตเชิงสัมพันธ์ที่เรียกว่าสมบัติการปิด (clusre property)

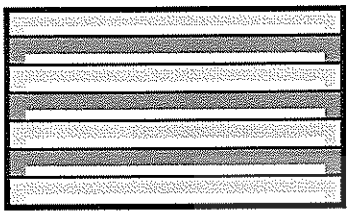
ตัวดำเนินการพีชคณิตเชิงสัมพันธ์ที่จะกล่าวถึงได้แก่ selection, projection, Cartesian product, union, set difference, join, intersection และ division ซึ่ง selection และ projection นั้นเป็นตัวดำเนินการแบบเอกภาค (unary operator) เนื่องจากกระทำบนรีเลชันเดียว ในขณะที่ตัวดำเนินการที่กระทำกับคู่ของรีเลชันจัดเป็นตัวดำเนินการแบบทวิภาค (binary operator) สำหรับ union, intersection, set difference และ Cartesian product เป็นตัวดำเนินการเซต

6.2.1 การดำเนินการเอกภาค (Unary Operations)

ตัวดำเนินการที่กระทำกับรีเลชันเพียงหนึ่งรีเลชัน ได้แก่ Selection และ Projection

6.2.1.1 ซีเลคชัน (Selection)

ซีเลคชัน หรือ เรสตริคชัน (Restriction) คือ การกระทำบนรีเลชันหนึ่งรีเลชันในการสร้างรีเลชันใหม่ที่มีทูเพิลตรงตามเงื่อนไขที่กำหนด



รูปแบบกระทำกับรีเลชัน

$\sigma_{predicate}(Relation)$

ชื่อตาราง WHERE <CONDITION>

สัญลักษณ์การดำเนินการซีเลคชัน (บน) และ รูปแบบ SQL (ล่าง) ที่สัมพันธ์กัน

σ คือตัวดำเนินการ selection และ predicate คือเงื่อนไขของการแสดงโดยกำหนดเงื่อนไขของค่าในคอลัมน์ใดๆ สัมพันธ์กับ SQL "WHERE" ที่จะกล่าวถึงในบทที่ 8

ตัวอย่าง

ให้แสดงข้อมูลของนักศึกษาทั้งหมดที่มีเกรดตั้งแต่ 2.00 ขึ้นไป

$\sigma_{gpa >= 2.00}(Student)$

Student

	cid	sid	name	login	age	gpa
⇨	3100904022132	B5075666	สมชาย	somchai@it	18	**3.44**
⇨	3100904032132	B5075688	สมศรี	somsri@com	18	**3.21**
⇨	3100905622132	B5075650	สมศรี	somsri@it	19	**3.82**
	3100904055132	B5075831	สมศักดิ์	somsak@med	21	**1.80**
⇨	3100904078132	B5075832	สมน้ำหน้า	somnamna@math	22	**2.00**

รีเลชันที่ถูกดำเนินการได้แก่รีเลชัน Student และ predicate คือ gpa >= 2.00 ตัวดำเนินการ selection สร้างรีเลชันที่บรรจุทูเพิลของนักศึกษาที่มีเกรดเฉลี่ยตั้งแต่ 2.00 ขึ้นไป ซึ่งเป็นนักศึกษาทั้งหมดยกเว้นสมศักดิ์ ผลลัพธ์แสดงได้ดังนี้

cid	sid	name	login	age	gpa
3100904022132	B5075666	สมชาย	somchai@it	18	3.44
3100904032132	B5075688	สมศรี	somsri@com	18	3.21
3100905622132	B5075650	สมศรี	somsri@it	19	3.82
3100904078132	B5075832	สมน้ำหน้า	somnamna@math	22	2.00

นอกจากนี้เงื่อนไขยังสามารถมีความซับซ้อนหรือกำหนดเงื่อนไขหลายๆ เงื่อนไขได้ด้วยตัวดำเนินการทางตรรกะ □ (และ) □ (หรือ) ~(นิเสธ)

ตัวอย่าง

ให้แสดงข้อมูลนักศึกษาทั้งหมดที่เกรดเฉลี่ยตั้งแต่ 2.00 ขึ้นไปและมีอายุมากกว่า 20 ปี

$\sigma_{gpa >= 2.00 \square age > 20}(Student)$

Student

cid	sid	name	login	age	gpa
3100904022132	B5075666	สมชาย	somchai@it	18	3.44
3100904032132	B5075688	สมศรี	somsri@com	18	3.21
3100905622132	B5075650	สมศรี	somsri@it	19	3.82
3100904055132	B5075831	สมศักดิ์	somsak@med	21	1.80
3100904078132	B5075832	สมน้ำหน้า	somnamna@math	22	2.00

พบว่านักศึกษาที่มีเกรดเฉลี่ยตั้งแต่ 2.00 ขึ้นไปคือนักศึกษาที่ชี้ด้วยเครื่องหมาย ⇨ ซึ่งมีจำนวน 4 คน ในขณะที่นักศึกษาที่มีอายุมากกว่า 20 ปี ชี้ด้วยเครื่องหมาย ➡ มีจำนวน 2 คน แต่นักศึกษาที่ทั้งมีเกรดเฉลี่ยตั้งแต่ 2.00 และมีอายุมากกว่า 20 ปีขึ้นไปมีเพียงคนเดียวได้แก่ สมน้ำหน้า จึงได้ผลลัพธ์เป็นรีเลชันดังนี้

cid	sid	name	login	age	gpa
3100904078132	B5075832	สมน้ำหน้า	somnamna@math	22	2.00

ตัวอย่าง

ให้แสดงข้อมูลนักศึกษาทั้งหมดที่เกรดเฉลี่ยตั้งแต่ 2.00 ขึ้นไปหรือนักศึกษาที่มีอายุมากกว่า 20 ปี

$\sigma_{gpa >= 2.00 \square age > 20}(Student)$

Student

cid	sid	name	login	age	gpa
3100904022132	B5075666	สมชาย	somchai@it	18	3.44
3100904032132	B5075688	สมศรี	somsri@com	18	3.21
3100905622132	B5075650	สมศรี	somsri@it	19	3.82
3100904055132	B5075831	สมศักดิ์	somsak@med	21	1.80
3100904078132	B5075832	สมน้ำหน้า	somnamna@math	22	2.00

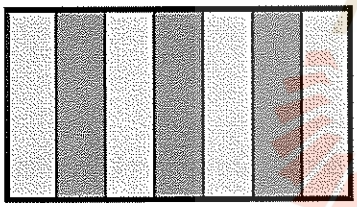
พบว่านักศึกษาที่มีเกรดเฉลี่ยตั้งแต่ 2.00 ขึ้นไปคือนักศึกษาที่ชี้ด้วยเครื่องหมาย ⇨ ซึ่งมีจำนวน 4 คน ในขณะที่นักศึกษาที่มีอายุมากกว่า 20 ปี ชี้ด้วยเครื่องหมาย ➡ มีจำนวน 2 คน ปรากฏว่านักศึกษาทั้งหมดที่มีคุณสมบัติเป็นไปตามเงื่อนไขข้อหนึ่งข้อใดหรือทั้งสองข้อ จึงได้ผลลัพธ์เป็นรีเลชันดังนี้

Student

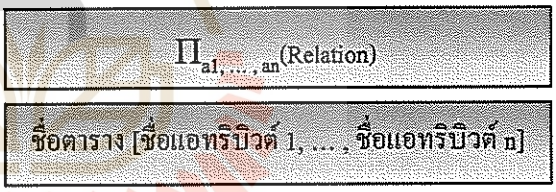
cid	sid	name	login	age	gpa
3100904022132	B5075666	สมชาย	somchai@it	18	3.44
3100904032132	B5075688	สมศรี	somsri@com	18	3.21
3100905622132	B5075650	สมศรี	somsri@it	19	3.82
3100904055132	B5075831	สมศักดิ์	somsak@med	21	1.80
3100904078132	B5075832	สมน้ำหน้า	somnamna@math	22	2.00

6.2.1.2 โปรเจคชัน (Projection)

โปรเจคชัน คือ การกระทำบนรีเลชันหนึ่งรีเลชันในการสร้างรีเลชันใหม่ที่มีเฉพาะซับเซตในแนวตั้งหรือคอลัมน์ตามรายการแอทริบิวต์ที่ระบุ



รูปแบบกระทำกับรีเลชัน



สัญลักษณ์การดำเนินการโปรเจคชัน (บน) และรูปแบบแบบคำสั่ง (ล่าง) ที่สัมพันธ์กัน

Π คือตัวดำเนินการ projection และ $a_1, \ldots, a_n$ รายการแอทริบิวต์ที่ต้องการแสดงจากรีเลชัน หรืออาจเขียนในรูปแบบชื่อตารางตามด้วยรายการแอทริบิวต์ที่ต้องการแสดง

ตัวอย่าง

ให้แสดงรหัสนักศึกษา ชื่อ และเกรดเฉลี่ยของนักศึกษาทั้งหมด

$\Pi_{sid, name, gpa}(Student)$

Student ⇩ ⇩ ⇩

cid	sid	name	login	age	gpa
3100904022132	B5075666	สมชาย	somchai@it	18	3.44
3100904032132	B5075688	สมศรี	somsri@com	18	3.21
3100905622132	B5075650	สมศรี	somsri@it	19	3.82
3100904055132	B5075831	สมศักดิ์	somsak@med	21	1.80
3100904078132	B5075832	สมน้ำหน้า	somnamna@math	22	2.00

Projection สร้างรีเลชันใหม่จากรีเลชัน Student ที่ประกอบด้วยคอลัมน์ที่ระบุไว้คือ sid, name และ gpa ได้ผลลัพธ์ดังนี้

sid	name	gpa
B5075666	สมชาย	3.44
B5075688	สมศรี	3.21
B5075650	สมศรี	3.82
B5075831	สมศักดิ์	1.80
B5075832	สมน้ำหน้า	2.00

ผลลัพธ์ที่ได้จากตัวดำเนินการเชิงสัมพันธ์ยังคงเป็นรีเลชันซึ่งสามารถนำไปกระทำโดยตัวดำเนินการอื่นๆ ได้ ในกรณีนี้ผลลัพธ์ที่ได้จากตัวดำเนินการ projection สามารถนำไปดำเนินการต่อด้วยตัวดำเนินการ selection ได้ หรือสามารถสลับลำดับก่อนหลังได้ ทั้งนี้ระบบจัดการฐานข้อมูลจะเป็นผู้ทำการเลือกลำดับของตัวดำเนินการเพื่อให้การแสดงผลข้อมูลเกิดความรวดเร็วที่สุด ซึ่งเป็นข้อดีอันหนึ่งของระบบจัดการฐานข้อมูล

$\sigma_{condition}(\Pi_{a1, \ldots, an}(Relation))$

ชื่อรีเลชั่น [ชื่อแอททริบิวต์ 1, ...] WHERE <CONDITION>

หรือ

$\Pi_{a1, \ldots, an}(\sigma_{condition}(Relation))$

ชื่อรีเลชั่น WHERE <CONDITION> [ชื่อแอททริบิวต์ 1, ...]

ตัวอย่าง

ให้แสดงรหัสนักศึกษา ชื่อ และเกรดเฉลี่ยของนักศึกษาที่มีเกรดเฉลี่ยตั้งแต่ 2.00 ขึ้นไป

$\sigma_{gpa >= 2.00}(\Pi_{sid, name, gpa}(Student))$

Student

	cid	sid	name	login	age	gpa
⇨	3100904022132	B5075666	สมชาย	somchai@it	18	**3.44**
⇨	3100904032132	B5075688	สมศรี	somsri@com	18	**3.21**
⇨	3100905622132	B5075650	สมศรี	somsri@it	19	**3.82**
	3100904055132	B5075831	สมศักดิ์	somsak@med	21	**1.80**
⇨	3100904078132	B5075832	สมน้ำหน้า	somnamna@math	22	**2.00**

Projection สร้างรีเลชันใหม่จากรีเลชัน Student ที่ประกอบด้วยคอลัมน์ที่ระบุไว้คือ sid, name และ gpa ได้ผลลัพธ์ดังนี้

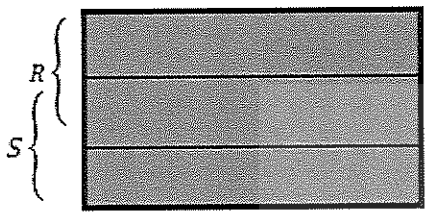
sid	name	gpa
B5075666	สมชาย	3.44
B5075688	สมศรี	3.21
B5075650	สมศรี	3.82
B5075832	สมน้ำหน้า	2.00

6.2.2 ตัวดำเนินการเซต (Set Operations)

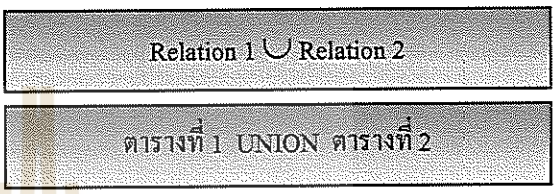
การดำเนินการ selection และ projection กระทำกับรีเลชันเพียง 1 รีเลชันเท่านั้น แต่การใช้งานฐานข้อมูลโดยทั่วไปเรามีความจำเป็นต้องสร้างข้อมูลจากรีเลชันมากกว่า 1 รีเลชัน ในหัวข้อนี้จะได้อธิบายถึงการดำเนินการของเซตเริ่มด้วย union, intersection, set difference, Cartesian product และ

6.2.2.1 ยูเนียน (Union)

ยูเนียน คือ การรวมรีเลชันเข้าด้วยกันโดยทูเพิลทั้งหมดในรีเลชันที่ถูกดำเนินการจะอยู่ในรีเลชันผลลัพธ์ ทูเพิลที่ซ้ำซ้อนกันจะถูกคงไว้เพียง 1 ทูเพิล และรีเลชันที่ยูเนียนกันต้องเข้ากันได้ (union-compatible)



รูปแบบกระทำกับรีเลชัน



สัญลักษณ์การดำเนินการยูเนียน (บน) และ
รูปแบบแบบคำสั่ง (ล่าง) ที่สัมพันธ์กัน

∪ คือตัวดำเนินการ union สำหรับ R และ S หรือ Relation 1 และ Relation 2 เป็นรีเลชันที่จะรวมเข้าด้วยกัน ทั้งสองรีเลชันนั้นต้องเข้ากันได้โดยมีจำนวนแอทริบิวต์เท่ากัน ลำดับของแอทริบิวต์ตรงกัน และโดเมนของแต่ละแอทริบิวต์สัมพันธ์กัน

ตัวอย่าง

R และ S เป็นรีเลชันที่มีข้อมูลของนักศึกษา R ∪ S แสดงได้ดังนี้

R

sid	name	gpa
B5075666	สมชาย	3.44
B5075831	สมศักดิ์	1.80
B5075832	สมน้ำหน้า	2.00

∪

S

sid	name	gpa
B5075666	สมชาย	3.44
B5075688	สมศรี	3.21

R

S

sid	name	gpa
B5075666	สมชาย	3.44
B5075831	สมศักดิ์	1.80
B5075832	สมน้ำหน้า	2.00
B5075666	สมชาย	3.44
B5075688	สมศรี	3.21

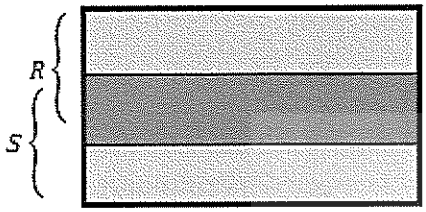
R ∪ S

sid	name	gpa
B5075666	สมชาย	3.44
B5075831	สมศักดิ์	1.80
B5075832	สมน้ำหน้า	2.00
B5075688	สมศรี	3.21

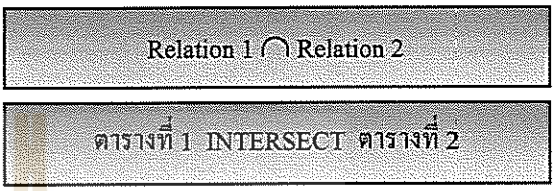
R มีนักศึกษา 3 คน และ S มีนักศึกษา 2 คน การ union กันนั้นนำทูเพิลทั้งหมดมารวมกัน แต่ทูเพิลที่ซ้ำซ้อนกันที่ได้ทำเครื่องหมาย ⇨ ไว้นั้นได้แก่ทูเพิลของสมชาย ซึ่งเป็นนักศึกษาคนเดียวกัน สังเกตได้จากข้อมูลที่เหมือนกันโดยเฉพาะรหัสนักศึกษา ให้คงไว้เพียงทูเพิลเดียว จึงได้ผลลัพธ์ R ∪ S ที่ประกอบ ด้วยทูเพิล 4 ทูเพิลดังแสดง

6.2.2.2 อินเตอร์เซกชัน (Intersection)

อินเตอร์เซกชัน คือ การรวมรีเลชันเข้าด้วยกัน โดยทูเพิลที่อยู่ในรีเลชันผลลัพธ์ทั้งหมดจะต้องปรากฏอยู่ในรีเลชันที่ถูกดำเนินการทุกรีเลชัน และรีเลชันที่อินเตอร์เซกกันต้องเข้ากันได้ (union-compatible)



รูปแบบกระทำกับรีเลชัน



สัญลักษณ์การดำเนินการอินเตอร์เซกชัน (บน) และรูปแบบแบบคำสั่ง (ล่าง) ที่สัมพันธ์กัน

∩ คือตัวดำเนินการ intersection สำหรับ R และ S หรือ Relation 1 และ Relation 2 เป็นรีเลชันที่จะรวมเข้าด้วยกัน ทั้งสองรีเลชันนั้นต้องเข้ากันได้โดยมีจำนวนแอทริบิวต์เท่ากัน ลำดับของแอทริบิวต์ตรงกัน และโดเมนของแต่ละแอทริบิวต์สัมพันธ์กัน ผลลัพธ์ที่ได้คือทูเพิลที่ปรากฏอยู่ในทั้ง R และ S

ตัวอย่าง

R และ S เป็นรีเลชันที่มีข้อมูลของนักศึกษา R ∩ S แสดงได้ดังนี้

R

sid	name	gpa
B5075666	สมชาย	3.44
B5075831	สมศักดิ์	1.80
B5075832	สมน้ำหน้า	2.00

∪

S

sid	name	gpa
B5075666	สมชาย	3.44
B5075688	สมศรี	3.21

⇨

R

sid	name	gpa
B5075666	สมชาย	3.44
B5075831	สมศักดิ์	1.80
B5075832	สมน้ำหน้า	2.00

S

sid	name	gpa
B5075666	สมชาย	3.44
B5075688	สมศรี	3.21

⇨

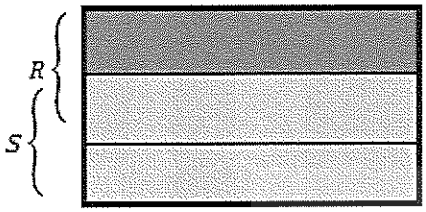
R ∩ S

sid	name	gpa
B5075666	สมชาย	3.44

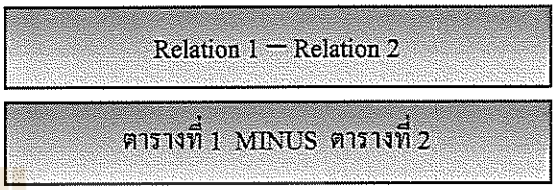
R มีนักศึกษา 3 คน และ S มีนักศึกษา 2 คน การ intersect กันนั้นให้ระบุทูเพิลที่ทั้ง 2 รีเลชันมีเหมือนกัน ในตัวอย่างมี 1 ทูเพิล ได้แก่ ทูเพิลของสมชายซึ่งเป็นนักศึกษาคนเดียวกันที่ได้ทำเครื่องหมาย ⇨ ไว้สังเกตได้จากข้อมูลที่เหมือนกันโดยเฉพาะรหัสนักศึกษา และให้คงไว้เพียงทูเพิลเดียว จึงได้ผลลัพธ์ R ∩ S ที่ประกอบ ด้วยทูเพิล 1 ทูเพิลดังแสดง

6.2.2.3 ผลต่างของเซต (Set Difference)

ผลต่างของเซต หรือ ผลลบ คือ การสร้างรีเลชันผลลัพธ์ที่ประกอบไปด้วยทูเพิลที่มีอยู่ในรีเลชันของตัวถูกดำเนินการแรก แต่ไม่มีอยู่ในตัวถูกดำเนินการตัวหลัง และรีเลชันที่ถูกกระทำต้องเข้ากันได้ (union-compatible)



รูปแบบกระทำกับรีเลชัน



สัญลักษณ์การดำเนินการหาผลต่างของเซต (บน) และรูปแบบแบบคำสั่ง (ล่าง) ที่สัมพันธ์กัน

– คือตัวดำเนินการ set difference สำหรับ R และ S หรือ Relation 1 และ Relation 2 เป็นรีเลชันที่จะถูกดำเนินการ โดย R เป็นตัวตั้งและ S เป็นตัวลบ ทั้งสองรีเลชันนั้นต้องเข้ากันได้โดยมีจำนวนแอทริบิวต์เท่ากัน ลำดับของแอทริบิวต์ตรงกัน และโดเมนของแต่ละแอทริบิวต์สัมพันธ์กัน ผลลัพธ์ที่ได้คือทูเพิลที่ปรากฏอยู่ในทั้ง R แต่ไม่ปรากฏอยู่ใน S

ตัวอย่าง

R และ S เป็นรีเลชันที่มีข้อมูลของนักศึกษา R – S แสดงได้ดังนี้

R

sid	name	gpa
B5075666	สมชาย	3.44
B5075831	สมศักดิ์	1.80
B5075832	สมน้ำหน้า	2.00

–

S

sid	name	gpa
B5075666	สมชาย	3.44
B5075688	สมศรี	3.21

⇨

R

sid	name	gpa
B5075666	สมชาย	3.44
B5075831	สมศักดิ์	1.80
B5075832	สมน้ำหน้า	2.00

S

sid	name	gpa
B5075666	สมชาย	3.44
B5075688	สมศรี	3.21

⇨

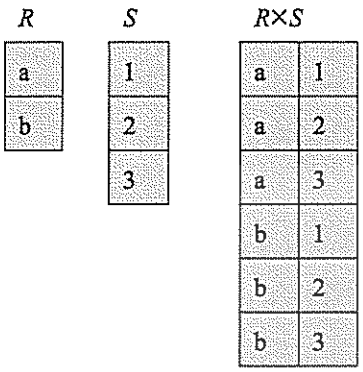
R – S

sid	name	gpa
B5075831	สมศักดิ์	1.80
B5075832	สมน้ำหน้า	2.00

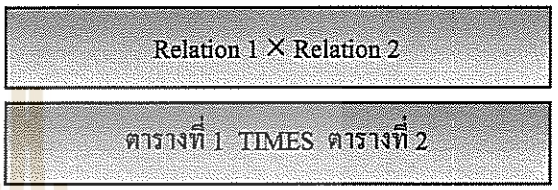
R มีนักศึกษา 3 คน และ S มีนักศึกษา 2 คน การลบกันนั้นให้ระบุทูเพิลที่มีใน R แต่ไม่ปรากฏใน S จากตัวอย่างพบว่า สมศักดิ์ และ สมน้ำหน้า ปรากฏใน R และไม่มีอยู่ใน S ในขณะที่สมชาย ถึงแม้จะมีปรากฏใน R แต่ว่าปรากฏอยู่ใน S ด้วย ตามที่ระบุด้วยเครื่องหมาย ⇨ สมชายจึงไม่เป็นทูเพิลในผลลัพธ์ ได้ผลลัพธ์ R – S ที่ประกอบด้วยทูเพิล 2 ทูเพิลดังแสดง

6.2.2.4 ผลคูณคาร์ทีเซียน (Cartesian Product)

ผลคูณคาร์ทีเซียน คือ การสร้างรีเลชันผลลัพธ์ที่ประกอบไปด้วยทูเพิลเป็นการจับคู่และต่อทูเพิลทุกทูเพิลในรีเลชันของตัวถูกดำเนินการตัวแรก เข้ากับทูเพิลทุกทูเพิลของตัวถูกดำเนินการตัวหลัง



รูปแบบกระทำกับรีเลชัน



สัญลักษณ์การดำเนินการคูณคาร์ทีเซียนชัน (บน) และรูปแบบแบบคำสั่ง (ล่าง) ที่สัมพันธ์กัน

× คือตัวดำเนินการ Cartesian product สำหรับ R และ S หรือ Relation 1 และ Relation 2 เป็นรีเลชันที่จะถูกดำเนินการ โดย R เป็นตัวตั้งและ S เป็นตัวคูณ ทั้งสองรีเลชันนั้นไม่จำเป็นต้องเข้ากันได้ ผลลัพธ์ที่ได้คือการต่อทูเพิลทั้งหมดใน R ด้วย ทูเพิลทั้งหมดใน S

ตัวอย่าง

แสดงข้อมูลการจับคู่การให้คำปรึกษาที่เป็นไปได้ทั้งหมดระหว่างอาจารย์และนักศึกษา โดย R เป็นรีเลชันข้อมูลอาจารย์ซึ่งมีเพียงคอลัมน์เดียว และ S เป็นรีเลชันที่มีข้อมูลของนักศึกษา R × S แสดงได้ดังนี้

R

fname
สถิตย์
พอลล่า

S

sid	name	gpa
B5075666	สมชาย	3.44
B5075831	สมศักดิ์	1.80
B5075832	สมน้ำหน้า	2.00

R × S

fname	sid	name	gpa
สถิตย์	B5075666	สมชาย	3.44
สถิตย์	B5075831	สมศักดิ์	1.80
สถิตย์	B5075832	สมน้ำหน้า	2.00
พอลล่า	B5075666	สมชาย	3.44
พอลล่า	B5075831	สมศักดิ์	1.80
พอลล่า	B5075832	สมน้ำหน้า	2.00

R มีอาจารย์ 2 คน และ S มีนักศึกษา 3 คน การคูรกันนั้นการต่อทูเพิลทั้งหมดใน R ด้วย ทูเพิลทั้งหมดใน S โดยเริ่มจากทูเพิลแรกในรีเลชัน R ได้แก่อาจารย์สถิตย์ ต่อท้ายด้วยข้อมูลนักศึกษาแต่ละคนได้ 3 ทูเพิลในผลลัพธ์ R × S

เมื่อทูเพิลแรกในรีเลชัน R ถูกต่อท้ายด้วยทุกทูเพิลในรีเลชัน S ดังนี้แล้ว ต่อไปเป็นการดำเนินการกับทูเพิลถัดมาได้แก่ทูเพิลอาจารย์พอลล่า นำมาต่อท้ายด้วยทูเพิลนักศึกษาใน S เช่นเดียวกันได้อีก 3 ทูเพิล รวมเป็น 6 ทูเพิลในผลลัพธ์ $R \times S$

6.2.3 (การดำเนินการเชื่อม) Join Operations

โดยปกติแล้วเราต้องการผลลัพธ์จากการเชื่อมตารางเข้าด้วยกันแบบ Cartesian product ที่เป็นไปตามเงื่อนไขเท่านั้น Cartesian product สร้างผลลัพธ์ที่มากเกินความจำเป็นและใช้เวลามาก เราจึงใช้การดำเนินการ join แทน การ join เป็นการสร้างรีเลชันที่เกิดจากการรวมรีเลชันเข้าด้วยกันซึ่งสำคัญเป็นอย่างมากในพีชคณิตเชิงสัมพันธ์ การ join เทียบได้กับการสร้าง Cartesian product จากนั้นทำการใช้การดำเนินการ selection เพื่อเลือกเฉพาะทูเพิลที่ตรงตามเงื่อนไข การ join นั้นมีหลายประเภทได้แก่ Theta join, equijoin, natural join, outer join, semijoin

6.2.3.1 Theta join (θ-join)

Theta join มีรูปแบบเป็น $R \bowtie_F S$ เป็นการสร้างรีเลชันที่เป็นไปตามเงื่อนไขใน predicate F ที่เกิดจาก Cartesian ของ R และ S ซึ่ง F คือเงื่อนไขในลักษณะ R.a θ S.b โดยที่ a และ b เป็นแอทริบิวต์ใน R และ S ตามลำดับ และ θ คือตัวดำเนินการเปรียบเทียบ ได้แก่ $<, \leq, >, \geq, =$ และ $\neq$

Theta join จึงมีรูปแบบที่สามารถเขียนได้อีกรูปแบบหนึ่งคือ $R \bowtie_F S = \sigma_F (R \times S)$

6.2.3.2 Equijoin

Equijoin คือ Theta join รูปแบบหนึ่ง ในกรณีที่ θ เป็นการเปรียบเทียบโดยเครื่องหมาย "=" เท่านั้น เราเรียกการ join ประเภทนี้ว่า **equijoin** ซึ่งเป็นการ join โดยทั่วไปของการใช้งานฐานข้อมูลเชิงสัมพันธ์

ตัวอย่าง

แสดงข้อมูลนักศึกษาและผลการเรียนสำหรับนักศึกษาที่ได้ลงทะเบียนเรียนในรายวิชาต่างๆ

Students $\bowtie_{\text{Students.sid = Enrolled.sid}}$ Enrolled

Students

sid	name	gpa
B5075666	สมชาย	3.44
B5075688	สมศรี	3.21
B5075650	สมศรี	3.82
B5075831	สมศักดิ์	1.80
B5075832	สมน้ำหน้า	2.00

Enrolled

cid	grade	sid
204203	A	**B5075650**
204204	B	**B5075831**
204205	D	**B5075688**
204329	F	**B5075666**
204204	A	**B5075688**

Students $\bowtie_{\text{Students.sid = Enrolled.sid}}$ Enrolled

Student.sid	name	gpa	cid	grade	Enrolled.sid
B5075666	สมชาย	3.44	204329	F	B5075666
B5075688	สมศรี	3.21	204205	D	B5075688
B5075688	สมศรี	3.21	204024	A	B5075688
B5075650	สมศรี	3.82	204203	A	B5075650
B5075831	สมศักดิ์	1.80	204204	B	B5075831

จากตัวอย่าง นอกจากการมองในลักษณะการ selection ที่ Students.sid = Enrolled.sid ของ Students × Enrolled แล้ว การดำเนินการ join สามารถมองได้ว่ามีการดำเนินการดังนี้ โดยเริ่มดำเนินการที่ทูเพิลแรกของตาราง Students ได้แก่นักศึกษารหัส B5075666 จากนั้นมาพิจารณาทูเพิลในตาราง Enrolled ว่าทูเพิลใดที่มี sid มีค่าเท่ากับ B5075666 พบว่ามี 1 ทูเพิลได้แก่รายวิชา 204329 ได้ผลลัพธ์เป็นทูเพิลแรกใน Students $\bowtie_{\text{Students.sid = Enrolled.sid}}$ Enrolled พิจารณาไม่มี

ทูเพิลใดใน Enrolled ที่ sid มีค่าเท่ากับ B5075666 อีกแล้ว จึงทำการดำเนินการ join ต่อกับทูเพิลถัดมาของตาราง Students ทูเพิลถัดมาได้แก่นักศึกษาที่มีรหัส B5075688 ซึ่งลงทะเบียนไว้ 2 รายวิชาคือ 204025 และ 204204 ได้เป็นผลลัพธ์ทูเพิลที่ 2 และ 3 ใน Students $\bowtie_{\text{Students.sid = Enrolled.sid}}$ Enrolled ดำเนินการเช่นนี้เรื่อยไป ในที่นี้ไม่ปรากฏว่าสมน้ำหน้า รหัส B5075832 ลงทะเบียนรายวิชาใด จึงไม่ปรากฏในตารางผลลัพธ์

6.2.3.3 Natural join

Natural join คือ equijoin แบบหนึ่ง มีรูปแบบเป็น $R \bowtie S$ ซึ่งกระทำบนแอทริบิวต์ที่เหมือนกันในรีเลชัน R และ S แอทริบิวต์ที่เหมือนกันจะคงไว้เพียง 1 แอทริบิวต์

Natural join เป็นการดำเนินการ equijoin กับรีเลชันที่ถูกกระทำโดย join ด้วยแอทริบิวต์ที่มีชื่อเหมือนกันในทั้งสองรีเลชัน โดยแอทริบิวต์ที่เป็นตัวเชื่อมจะคงไว้เพียงแอทริบิวต์เดียว

ตัวอย่าง

จากตัวอย่าง equijoin เป็นการ join โดยใช้แอทริบิวต์ sid จากทั้ง 2 รีเลชันอยู่แล้ว ดังนั้นเราสามารถเขียน equijion ให้อยู่ในรูปของ natural join และแสดงผลลัพธ์ของการดำเนินการได้ดังนี้

Students $\bowtie_{\text{Students.sid = Enrolled.sid}}$ Enrolled

Student.sid	name	gpa	cid	grade	Enrolled.sid
B5075666	สมชาย	3.44	204329	F	B5075666
B5075688	สมศรี	3.21	204205	D	B5075688
B5075688	สมศรี	3.21	204024	A	B5075688
B5075650	สมศรี	3.82	204203	A	B5075650
B5075831	สมศักดิ์	1.80	204204	B	B5075831

Students ⋈Enrolled

sid	name	gpa	cid	grade
B5075666	สมชาย	3.44	204329	F
B5075688	สมศรี	3.21	204205	D
B5075688	สมศรี	3.21	204024	A
B5075650	สมศรี	3.82	204203	A
B5075831	สมศักดิ์	1.80	204204	B

เนื่องจาก Students $\bowtie_{\text{Students.sid = Enrolled.sid}}$ Enrolled เป็นการ join โดยแอทริบิวต์ sid ทั้งคู่ จึงสามารถตัด predicate ทิ้งได้ เกิดเป็นการ join แบบ natural join ทั้งนี้แอทริบิวต์ Student.sid และ Enrolled.sid เป็นแอทริบิวต์ที่เหมือนกันและใช้ในการ join จึงคงไว้เพียง 1 แอทริบิวต์ ได้ผลลัพธ์เป็นรีเลชัน Students ⋈Enrolled ในรูปทางขวามือ

6.2.3.4 Outer join

Outer join มีรูปแบบเป็น $R \times S$ (left outer join) คือการ join ระหว่างรีเลชัน R และ S โดยที่ทูเพิลใน R ที่มีค่าของแอทริบิวต์ร่วมที่ใช้ในการ join ไม่สามารถจับคู่กับแอทริบิวต์ร่วมใน S ยังคงปรากฏอยู่ในผลลัพธ์ สำหรับค่าที่ขาดไปในตาราง S กำหนดให้เป็นค่า null

ตัวอย่าง

จากตัวอย่าง natural join สามารถดำเนินการแบบ outer join ได้ดังนี้ Students ⟕ Enrolled

Students

sid	name	gpa
B5075666	สมชาย	3.44
B5075688	สมศรี	3.21
B5075650	สมศรี	3.82
B5075831	สมศักดิ์	1.80
B5075832	สมน้ำหน้า	2.00

Enrolled

cid	grade	sid
204203	A	B5075650
204204	B	B5075831
204205	D	B5075688
204329	F	B5075666
204204	A	B5075688

Students ⟕ Enrolled

sid	name	gpa	cid	grade
B5075666	สมชาย	3.44	204329	F
B5075688	สมศรี	3.21	204205	D
B5075688	สมศรี	3.21	204024	A
B5075650	สมศรี	3.82	204203	A
B5075831	สมศักดิ์	1.80	204204	B
B5075832	สมน้ำหน้า	2.00	null	null

จากตัวอย่างพบว่า การดำเนินการ natural join ระหว่าง Students และ Enrolled จะไม่ปรากฏทูเพิลนักศึกษาที่ชื่อสมน้ำหน้าในผลลัพธ์เนื่องจากไม่สามารถจับคู่รหัสนักศึกษา (sid) ได้ในตาราง Enrolled ได้ แต่หากกระทำ outer join ทูเพิลของนักศึกษาสมน้ำหน้าจะยังคงปรากฏอยู่ในตารางผลลัพธ์ดังแสดง และค่าที่ว่างอยู่ในตารางจะมีค่าเป็นค่าว่างหรือ null

ตัวอย่างที่ได้กล่าวถึงคือการดำเนินการ left natural outer join คือการทำ natural join (ไม่ได้ระบุ predicate สำหรับการ join) ที่คงค่าทูเพิลของตัวถูกดำเนินการทางซ้ายไว้ทั้งหมด นอกจาก left outer join แล้ว เรายังสามารถดำเนินการ right outer join ได้ กล่าวคือเป็นการดำเนินการ outer join ที่คงค่าทูเพิลของตัวถูกดำเนินการทางขวาไว้ทั้งหมดแม้จะไม่มีค่าที่จับคู่กันได้สำหรับแอทริบิวต์ร่วมกับตัวถูกดำเนินการทางซ้าย รวมทั้งการดำเนินการแบบ full outer join ก็สามารถกระทำได้ โดยผลลัพธ์แสดงทูเพิลจากการ join กันของรีเลชันรวมกับทูเพิลของตัวถูกดำเนินการทั้งทางซ้ายและขวาของตัวดำเนินการที่ค่าในแอทริบิวต์ร่วมในการ join ไม่สามารถจับคู่กันได้

6.2.3.5 Semijoin

Semijoin มีรูปแบบเป็น $R \rhd_F S$ คือการสร้างรีเลชันที่ประกอบไปด้วยทูเพิลของ R ที่ปรากฏในการ join ระหว่างรีเลชัน R และ S

Semijoin มีรูปแบบที่สามารถเขียนได้อีกรูปแบบหนึ่งคือ $R \rhd_F S = \Pi_A (R \times S)$

โดยที่ A คือเซตของแอทริบิวต์ทั้งหมดใน R

Semijoin เป็นการดำเนินการ join ระหว่างรีเลชันและแสดงผลลัพธ์เฉพาะแอทริบิวต์ที่อยู่ในรีเลชันที่เป็นตัวถูกดำเนินการตัวหน้าเท่านั้น การดำเนินการ semijoin นี้มีข้อดีในระบบจัดการฐานข้อมูลเชิงสัมพันธ์ตรงที่เราสามารถลดจำนวนแอทริบิวต์ที่เกี่ยวข้องการดำเนินการ join ทำให้การดำเนินการมีความรวดเร็วขึ้น

ตัวอย่าง

จากตัวอย่าง equijion ให้แสดงข้อมูลนักศึกษาที่ได้ทำการลงทะเบียนเรียนแล้ว Students $\triangleright_{\text{Students.sid = Enrolled.sid}}$ Enrolled

Students $\bowtie_{\text{Students.sid = Enrolled.sid}}$ Enrolled

Student.sid	name	gpa	cid	grade	Enrolled.sid
B5075666	สมชาย	3.44	204329	F	B5075666
B5075688	สมศรี	3.21	204205	D	B5075688
B5075688	สมศรี	3.21	204024	A	B5075688
B5075650	สมศรี	3.82	204203	A	B5075650
B5075831	สมศักดิ์	1.80	204204	B	B5075831

Students $\triangleright_{\text{Students.sid = Enrolled.sid}}$ Enrolled

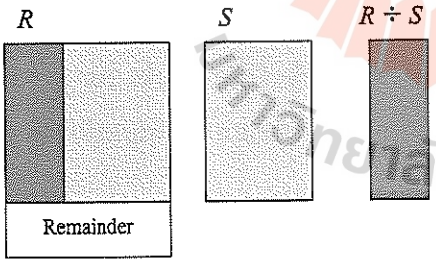
Student.sid	name	gpa
B5075666	สมชาย	3.44
B5075688	สมศรี	3.21
B5075688	สมศรี	3.21
B5075650	สมศรี	3.82
B5075831	สมศักดิ์	1.80

การแสดงข้อมูลตามตัวอย่างตรงกับการ join กันระหว่างรีเลชัน Students และ Enrolled แต่ผลลัพธ์ของ semijoin นั้นแสดงเฉพาะแอทริบิวต์ใน Students ดังแสดงในรูปทางด้านขวา

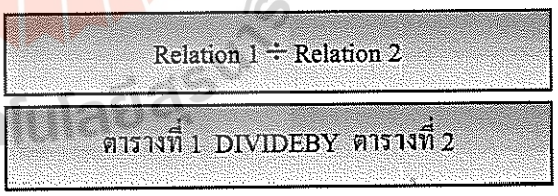
ในกรณีนี้สามารถเขียนในรูป Students ▷ Enrolled ก็ได้เนื่องจากเป็น natural semijoin

6.2.4 การดำเนินการหาร (Division Operations)

การหาร คือ การสร้างรีเลชันจากสองรีเลชั่น โดยที่รีเลชั่นทั้งสองมีแอททริบิวต์อย่างน้อยหนึ่งแอททริบิวต์ที่เหมือนกัน ผลลัพธ์ที่ได้ จะเป็นค่าของแอททริบิวต์จากรีเลชั่นที่มีจำนวนแอททริบิวต์มากกว่าซึ่งเป็นค่าของแอททริบิวต์หนึ่งที่มีค่าหนึ่งที่จับคู่ตรงกับทุกค่าของแอททริบิวต์ที่เหมือนกันนี้ในอีกรีเลชั่นที่มีแอททริบิวต์น้อยกว่า



รูปแบบกระทำกับรีเลชัน

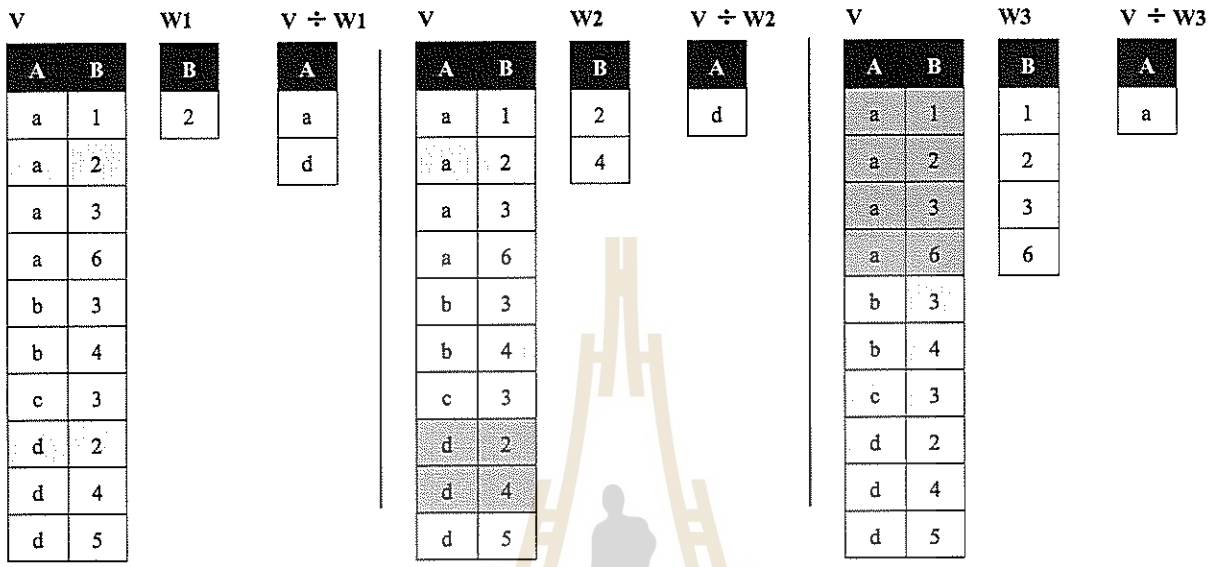


สัญลักษณ์การดำเนินการหารของเซต (บน) และ
รูปแบบแบบคำสั่ง (ล่าง) ที่สัมพันธ์กัน

÷ คือตัวดำเนินการ division สำหรับ R และ S หรือ Relation 1 และ Relation 2 เป็นรีเลชันที่จะถูกดำเนินการโดย R เป็นตัวตั้งและ S เป็นตัวหาร ทั้งสองรีเลชันนั้นไม่จำเป็นต้องเข้ากันได้ ผลลัพธ์ที่ได้คือการต่อทูเพิลทั้งหมดใน R ด้วย ทูเพิลทั้งหมดใน S

ตัวอย่าง

รีเลชัน V และ W ใช้เป็นตัวอย่างสำหรับ division ซึ่งแสดงไว้ 3 ตัวอย่าง ดังนี้



ในตัวอย่างซ้ายสุด V เป็นตัวตั้งและ W1 เป็นตัวหาร ทูเพิลใดบ้างที่แอทริบิวต์ B ใน V ที่มีค่าตรงกับใน W1 ซึ่งในที่นี้คือค่า 2 ผลปรากฏว่ามีอยู่ 2 ทูเพิล ผลลัพธ์ V ÷ W1 จำได้แก่ a และ d ซึ่งเป็นค่าแอทริบิวต์ในรีเลชันที่มีจำนวนแอทริบิวต์มากกว่าและเป็นตัวตั้ง

ในตัวอย่างกลางนั้น W2 ที่เป็นตัวหารมี 2 ทูเพิลได้แก่ค่า 2 และ 4 ซึ่งทูเพิลที่มีค่าแอทริบิวต์ B เท่ากับ 2 หรือ 4 ในรีเลชัน V นั้นมีอยู่ 4 ทูเพิลที่ได้แรงเงาไว้ อย่างไรก็ตามต้องพิจารณาว่าค่าในแอทริบิวต์ที่เป็นตัวตั้งต้องมีค่าเหมือนกันด้วย จึงมีเพียงทูเพิลที่มีค่าแอทริบิวต์ A ที่มีค่าเป็น d เท่านั้นที่เหมือนกันสำหรับทูเพิลที่แรงเงาไว้ จำเป็นคำตอบของ V ÷ W2 เช่นเดียวกับในตัวอย่างทางขวาสุดที่ทูเพิลใน V ที่มีค่าแอทริบิวต์ B ที่ 1, 2, 3 หรือ 6 สัมพันธ์กับตัวหาร W3 แต่มีเพียงทูเพิลที่มีค่าแอทริบิวต์ A เป็น a เท่านั้นที่มีค่าจับคู่ได้ทั้ง 4 ทูเพิล

ตัวอย่าง

การดำเนินการหารรีเลชันนั้นมักนำมาใช้งานในรูปแบบของข้อคำถามเกี่ยวกับกิจกรรมที่เอนทีตีหลายเอนทิตีกระทำเหมือนกันเช่นการแสดงรายวิชาที่นักศึกษากลุ่มหนึ่งลงทะเบียนเรียนเหมือนกันทั้งหมด ดังตัวอย่างต่อไปนี้

Enrolled1

cid	*sid*
204203	B5075650
204204	B5075831
204205	B5075688
204329	B5075666
204204	B5075688
204204	B5075666

Student1

sid
B5075688
B5075666

Enrolled1

cid	*sid*
204203	B5075650
204204	B5075831
204205	B5075688
204329	B5075666
204204	B5075688
204204	B5075666

Enrolled1 ÷ Student1

cid
204204

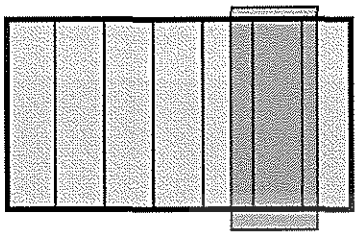
ทูเพิลที่มีแอทริบิวต์ตรงกันกับตัวหาร (Student1) มี 4 ทูเพิล ดังที่ได้แรเงาไว้ เป็นการแสดงว่านักศึกษา B5075688 และ B5075666 ลงทะเบียนรายวิชาใดบ้าง แต่เราต้องการทราบเพียงรายวิชาเรียนที่เหมือนกันซึ่งเป็นผลหารของรีเลชัน พบว่ามีเพียงรายวิชา 204204 เท่านั้นที่ทั้ง 2 คนลงทะเบียนเรียนทั้งคู่ซึ่งได้ทำเครื่องหมาย⇨ ระบุไว้ ผลลัพธ์ที่ได้เป็นแอทริบิวต์ที่เหมือนกันนี้ในรีเลชันตัวตั้ง (Enrolled) ได้ผลลัพธ์ Enrolled1 ÷ Student1 ดังแสดง

6.2.5 การดำเนินการแบบเป็นชุดและจัดกลุ่ม (Aggregation and Grouping Operations)

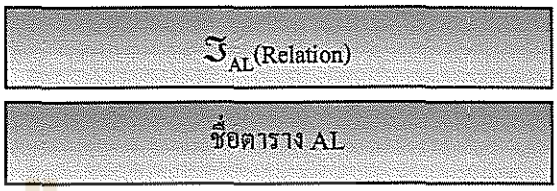
บ่อยครั้งที่เราต้องการดำเนินการกับข้อมูลในรูปแบบของการคำนวณใดๆ ในกลุ่มของข้อมูล (aggregation) เช่นการหาผลรวมยอดขายสินค้า และการดำเนินการนั้นยังอาจมีการจัดกลุ่มของข้อมูล (grouping) เช่นการหาผลรวมแยกตามสาขา ทั้งนี้พีชคณิตเชิงสัมพันธ์พื้นฐานไม่สามารถกระทำได้ ในหัวข้อนี้อธิบายการดำเนินการเพิ่มเติมของพีชคณิตเชิงสัมพันธ์ที่สามารถจัดการข้อมูลเป็นชุดๆ ได้

6.2.5.1 Aggregate Operations

Aggregate operation คือ การกระทำบนรีเลชันโดยการใช้ฟังก์ชันแบบ aggregate function กับแอทริบิวต์ที่กำหนด



รูปแบบกระทำกับรีเลชัน



สัญลักษณ์การดำเนินการโปรเจคชัน (บน) และ
รูปแบบแบบคำสั่ง (ล่าง) ที่สัมพันธ์กัน

ℑ คือตัวดำเนินการ aggregate operation และ AL คือรายการฟังก์ชันที่กำหนดให้กระทำกับรีเลชัน รายการรีเลชันแต่ละรายการประกอบด้วยฟังก์ชันและแอทริบิวต์ที่กำหนดสำหรับฟังก์ชันนั้นๆ aggregate function หลักๆ ได้แก่

COUNT—นับจำนวนของค่าในแอทริบิวต์ที่กำหนด AVG—หาค่าเฉลี่ยของค่าในแอทริบิวต์ที่กำหนด

SUM—หาผลรวมของค่าในแอทริบิวต์ที่กำหนด MIN/MAX—หาค่าต่ำสุด/สูงสุด

ตัวอย่าง

แสดงจำนวนนักศึกษาโดยการนับด้วยรหัสนักศึกษา พร้อมทั้งแสดงเกรดเฉลี่ย และอายุเฉลี่ยของนักศึกษา

$\rho_{sReports}$ (sCount, maxAge, avgGpa) $\Im_{COUNT\ sid,\ MAX\ age,\ AVG\ gpa}$ (Students)

Students

cid	sid (COUNT)	name	login	age (MAX)	gpa (AVG)
3100904022132	B5075666	สมชาย	somchai@it	18	3.44
3100904032132	B5075688	สมศรี	somsri@com	18	3.21
3100905622132	B5075650	สมศรี	somsri@it	19	3.82
3100904055132	B5075831	สมศักดิ์	somsak@med	21	1.80
3100904078132	B5075832	สมน้ำหน้า	somnamna@math	22	2.00

sReports

sCount	maxAge	avgGpa
5	22	2.85

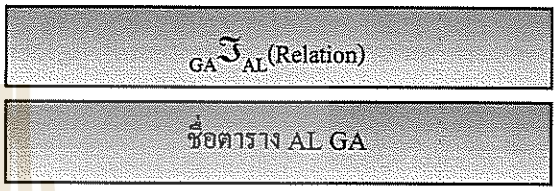
$\rho_{sReports}$ (sCount, maxAge, avgGpa) คือการตั้งชื่อของผลลัพธ์ที่ได้ในที่นี้ระบุชื่อรีเลชันเป็น sReports กำหนดแอทริบิวต์ให้มีชื่อว่า sCount, maxAge และ avgGpa ตามลำดับ ℑ เป็นเครื่องหมายการดำเนินการแบบ aggregate และรายการ COUNT sid, MAX age, AVG gpa คือ aggregarte list ซึ่งแต่ละรายการประกอบไปด้วย aggregate function และแอทริบิวต์ที่ฟังก์ชัน นั้นๆ กระทำ ซึ่งรายการแรกได้แก่ฟังก์ชัน COUNT ซึ่งเป็นการนับจำนวน

รหัสนักศึกษา (sid) รายการถัดมาได้แก่ฟังก์ชัน MAX ซึ่งเป็นการหาอายุ (age) ที่มากที่สุดของนักศึกษา และรายการสุดท้ายได้แก่ AVG ซึ่งเป็นฟังก์ชันหาค่าเฉลี่ยของเกรดเฉลี่ย (gpa) ของนักศึกษาทั้งหมดในรีเลชัน Students

6.2.5.2 Grouping Operations

Grouping operation คือ การกระทำบนรีเลชันโดยการใช้ฟังก์ชันแบบ aggregate function กับแอทริบิวต์ที่กำหนด และจัดกลุ่มตามแอทริบิวต์ที่ระบุ

รูปแบบกระทำกับรีเลชัน



สัญลักษณ์การดำเนินการหาผลต่างของเซต (บน) และ
รูปแบบแบบคำสั่ง (ล่าง) ที่สัมพันธ์กัน

ℑ คือตัวดำเนินการ aggregate operation และ AL คือรายการฟังก์ชันที่กำหนดให้กระทำกับรีเลชัน รายการรีเลชันแต่ละรายการประกอบด้วยฟังก์ชันและแอทริบิวต์ที่กำหนดสำหรับฟังก์ชันนั้นๆ เช่นเดียวกันกับ aggregate operation ในหัวข้อที่แล้ว แต่การดำเนินการนั้นกระทำเป็นกลุ่มๆ สำหรับทูเพิลที่มีค่าของแอทริบิวต์ที่ระบุไว้ใน grouping attributes (GA) ที่เหมือนกัน

ตัวอย่าง

แสดงจำนวนนักศึกษาโดยการนับด้วยรหัสนักศึกษา พร้อมทั้งแสดงเกรดเฉลี่ย และอายุเฉลี่ยของนักศึกษา

$\rho_{sReports}(age, avgGpa)_{age}\Im_{AVG\ gpa}(Students)$

Students

GROUP BY AVG

cid	sid	name	login	age	gpa
3100904022132	B5075666	สมชาย	somchai@it	18	3.44
3100904032132	B5075688	สมศรี	somsri@com	18	3.21
3100905622132	B5075650	สมศรี	somsri@it	19	3.82
3100904055132	B5075831	สมศักดิ์	somsak@med	21	1.80
3100904078132	B5075832	สมน้ำหน้า	somnamna@math	22	2.00

sReports

age	avgGpa
18	3.33
19	3.82
21	1.80
22	2.00

$\rho_{sReports}$ (age, avgGpa) คือการตั้งชื่อของผลลัพธ์ที่ได้ในที่นี้ระบุชื่อรีเลชันเป็น sReports กำหนดแอทริบิวต์ให้มีชื่อว่า age และ avgGpa ตามลำดับ ℑ เป็นเครื่องหมายการดำเนินการแบบ aggregate และรายการ AVG gpa คือ aggregarte list ซึ่งมีเพียงรายการเดียวได้แก่ AVG ซึ่งเป็นฟังก์ชันหาค่าเฉลี่ยของเกรดเฉลี่ย (gpa) ของนักศึกษาจำแนก

ตามอายุ (age) ซึ่งระบุไว้ในตำแหน่งของ GA หรือหน้าเครื่องหมาย $\Im$ ในตัวอย่างสามารถจัดกลุ่มนักศึกษาที่มีอายุเท่ากันได้เป็น 4 กลุ่ม จากนั้นหาค่าเฉลี่ยของเกรดเฉลี่ยของนักศึกษาในแต่ละกลุ่มได้เป็นผลลัพธ์ sReports ดังแสดง

สรุป**

6.3 แคลคูลัสเชิงสัมพันธ์ (Relational Calculus)

แคลคูลัสเชิงสัมพันธ์เป็นการดำเนินการบนโมเดลเชิงสัมพันธ์รูปแบบหนึ่ง ซึ่งต่างจากพีชคณิตเชิงสัมพันธ์ตรงที่แคลคูลัสเชิงสัมพันธ์นี้ เป็นการดำเนินการที่ไม่ได้ระบุขั้นตอนการได้มาซึ่งผลลัพธ์ หรือไม่ได้ระบุว่า ทำอย่างไร (how) เหมือนในพีชคณิตเชิงสัมพันธ์ แต่เป็นการระบุว่าอะไร (what) เป็นผลลัพธ์ที่ต้องการจากความสัมพันธ์ต่างๆ ในโมเดลเชิงสัมพันธ์ที่ระบุไว้ในข้อคำถามของแคลคูลัสเชิงสัมพันธ์ แคลคูลัสเชิงสัมพันธ์ไม่ได้เกี่ยวข้องกับ differential calculus หรือ integral calculus แต่ได้ชื่อมาจากศาสตร์ที่ใช้สัญลักษณ์ทางตรรกะที่เรียกว่า predicate calculus ในโมเดลเชิงสัมพันธ์ แคลคูลัสเชิงสัมพันธ์แบ่งเป็น 2 รูปแบบได้แก่ แคลคูลัสเชิงสัมพันธ์แบบทูเพิล (tuple relational calculus) และ แคลคูลัส domain relational calculus

6.4 แบบฝึกหัดท้ายบท

ให้นักศึกษาอธิบายความหมายและยกตัวอย่างการดำเนินการพีชคณิตเชิงสัมพันธ์ โดยยกตัวอย่างจากโครงงาน เช่า-ฉัน-สิ ในภาคผนวก