# บทที่ 3

# แนวคิด และ ทฤษฎี

## คุณลักษณะของฐานข้อมูล (Characteristics of Database Systems)

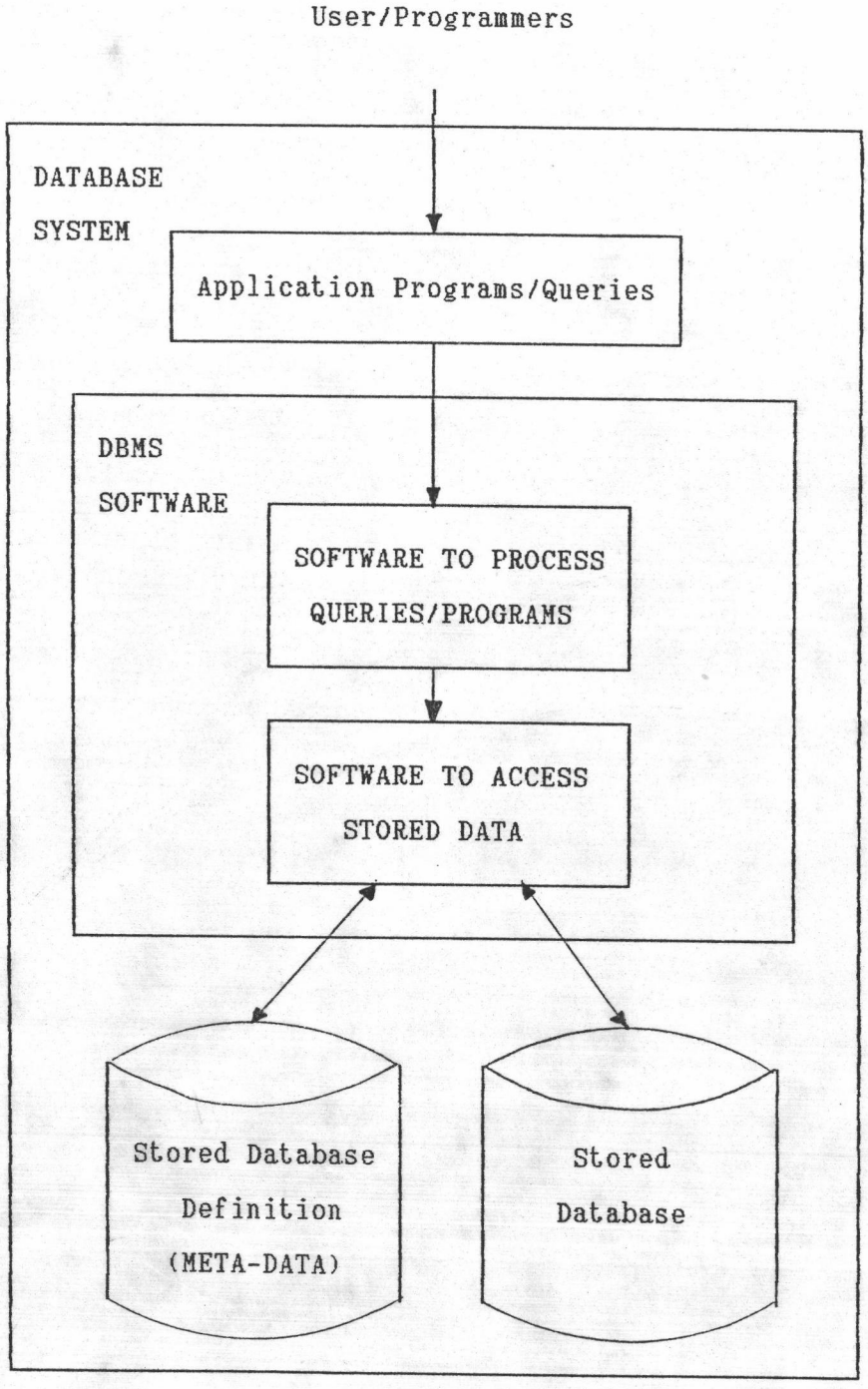
องค์ประกอบที่สำคัญของระบบสารสนเทศเพื่อการจัดการ คือ ฐานข้อมูล ฐานข้อมูลหมายถึง การรวบรวมข้อมูลที่ถูกจัดเป็นโครงสร้าง ที่สามารถใช้ร่วมกันในส่วนต่างๆ ของระบบสารสนเทศขององค์กร ฐานข้อมูลมีคุณลักษณะที่สำคัญดังนี้

1. โปรแกรม-ข้อมูล เป็นอิสระต่อกัน (Program-data independence) ในการประมวลผลแฟ้มข้อมูลแบบเก่านั้น โครงสร้างของแฟ้มข้อมูลจะแทรกอยู่ในตัวโปรแกรม ดังนั้นการเปลี่ยนแปลงใดๆ ของโครงสร้างของแฟ้มข้อมูล อาจจะทำให้ต้องมีการเปลี่ยนแปลงโปรแกรมทั้งหมดที่เรียกใช้แฟ้มข้อมูลนั้น ในทางตรงข้าม ระบบจัดการฐานข้อมูลจะเรียกใช้โปรแกรมที่ถูกเขียนขึ้นอย่างเป็นอิสระจากแฟ้มข้อมูล โครงสร้างของแฟ้มข้อมูลจะถูกเก็บแยกจากโปรแกรมที่เรียกใช้

2. ข้อมูลมีความเป็นอันหนึ่งอันเดียวกัน (Data integration) หมายความว่าฐานข้อมูลเป็นการรวบรวมข้อมูลซึ่งไม่ควรมีข้อมูลที่ซ้ำซ้อน ไม่จำเป็น หรือไม่ได้ใช้ เป็นต้น

3. มีความเป็นบูรณภาพของข้อมูล (Data integrity) หมายความว่าข้อมูลมีความน่าเชื่อถือ ไม่ผิดพลาด มีกฎเกณฑ์ในการควบคุมค่าของข้อมูล และในการดูแลรักษาระบบการทำงาน เราต้องแน่ใจว่าข้อมูลมีความสอดคล้อง และไม่มีความลักหลั่น (inconsistencies) เกิดขึ้นในฐานข้อมูล

4. แยกระหว่างมุมมองของข้อมูลเชิงตรรก กับเชิงกายภาพ (Seperate logical and physical views of data) ในการประมวลผลแฟ้มข้อมูล แต่ละแฟ้มข้อมูลอาจจะมีการกำหนดความยาวของเรคคอร์ด จำนวนไบท์ในแต่ละเรคคอร์ด และแต่ละเขตข้อมูลอาจถูกกำหนดโดยไบท์เริ่มต้นภายในเรคคอร์ดกับความยาวของเขตข้อมูลนั้นๆ แต่ในลักษณะของระบบฐานข้อมูล ผู้ใช้อาจไม่จำเป็นต้องสนใจว่าแต่ละเขตข้อมูลอยู่ตำแหน่งไหน เพียงแต่อ้างชื่อถึงเขตข้อมูลนั้นๆ ก็จะได้ค่าที่ต้องการ นั่นคือ ภายใต้แนวความคิดฐานข้อมูล จะพยายามแยกวิวของข้อมูลในเชิงธุรกิจออกจากรายละเอียดเกี่ยวกับลักษณะทางกายภาพ




รูปที่ 3.1 แสดงสภาพแวดล้อมระบบการจัดการฐานข้อมูลอย่างง่าย

## ระบบจัดการฐานข้อมูล (Database Management System)

ระบบจัดการฐานข้อมูล หรือ ดีบีเอ็มเอส (DBMS) เป็นชุดของโปรแกรม ซึ่งทำให้ผู้ใช้สามารถสร้างและดูแลฐานข้อมูล ตามรูปที่ 3.1 จะเห็นว่าระบบฐานข้อมูลจะประกอบด้วยฐานข้อมูลกับซอฟท์แวร์ที่ใช้ในการจัดการข้อมูล หรือดีบีเอ็มเอส นั่นเอง ดังนั้นผู้ใช้ และโปรแกรมจะเรียกดึงข้อมูล หรือเก็บข้อมูล โดยการโต้ตอบกับดีบีเอ็มเอส กล่าวโดยสรุปได้ว่า ดีบีเอ็มเอส อำนวยความสะดวกในการทำงานต่างๆ ดังนี้

- การสร้างและแก้ไขโครงสร้างของฐานข้อมูลรวมทั้งบรรจุข้อมูลในการทำงาน
- การเข้าถึงเนื้อหาในฐานข้อมูล เพื่อการแก้ไข หรือเรียกดูข้อมูล ได้พร้อมกัน โดยปราศจากข้อขัดแย้ง
- กำหนดคำจำกัดความ และข้อบังคับให้กระทำ เกี่ยวกับความต้องการในการรักษาความปลอดภัย การให้สิทธิ์ผู้ใช้ และการป้องกันความเสียหาย ตลอดจนการทำสำรองข้อมูล
- รวบรวมสถิติ เฝ้าคุมระบบ (system monitoring) และสามารถเปลี่ยนแปลงแก้ไขระบบเพื่อให้ทำงานได้ดีขึ้น

ทุกๆ ดีบีเอ็มเอสจะมีพจนานุกรมข้อมูล (Data Dictionary or Catalog) ซึ่งใช้เก็บข้อมูลเกี่ยวกับฐานข้อมูล เช่น โครงสร้าง ชนิด รูปแบบและข้อจำกัดของข้อมูล นั่นคือพจนานุกรมข้อมูลเองก็เป็นฐานข้อมูล ซึ่งเป็นที่เก็บข้อมูลของข้อมูล (meta data)

## โมเดลข้อมูล (Data Model)

โมเดลข้อมูล เป็นแนวความคิดซึ่งใช้อธิบายโครงสร้างของฐานข้อมูล โดยโครงสร้างของฐานข้อมูล หมายความถึง ชนิดข้อมูล ความสัมพันธ์ และข้อจำกัด ซึ่งใช้จัดการกับข้อมูล นอกจากนี้ โมเดลข้อมูล จะรวมถึงการปฏิบัติการในการเรียกใช้ ค้นหา และแก้ไขบนฐานข้อมูลด้วย เราสามารถแบ่งโมเดลของข้อมูลตามชนิดของแนวความคิดที่ใช้ในการอธิบายโครงสร้างของฐานข้อมูล ได้ดังนี้

1. โมเดลข้อมูลเชิงมโนภาพ (Conceptual data models) หรือโมเดลระดับบน (High-level) แนวคิดจะเป็นในลักษณะที่ผู้ใช้รับรู้ และเข้าใจถึงข้อมูล โครงสร้างของข้อมูลเชิงมโนภาพจะสะท้อนลักษณะข้อมูลของงานต่างๆ ในลักษณะที่เป็นกลาง (ไม่ขึ้นกับฮาร์ดแวร์และซอฟท์แวร์) โมเดลระดับบนนี้ จะใช้แนวคิดเกี่ยวกับเอนติตี แอตตริบิวต์ และรีเลชันชิป ซึ่งจะกล่าวในรายละเอียดต่อไป

2. โมเดลข้อมูลเชิงกายภาพ (Physical data models) หรือโมเดลระดับล่าง (Low-level) ใช้อธิบายรายละเอียดว่าข้อมูลถูกเก็บในคอมพิวเตอร์อย่างไร ซึ่งจะเป็นที่

เข้าใจสำหรับผู้เชี่ยวชาญด้านคอมพิวเตอร์ ไม่ใช่สำหรับผู้ใช้ทั่วไป โดยจะแสดงถึงรูปแบบของเรคคอร์ด การเรียงลำดับของเรคคอร์ด รวมทั้งวิธีการเข้าถึง หรือโครงสร้างที่ทำให้การค้นหาเรคคอร์ดที่ต้องการในฐานข้อมูลได้เร็วขึ้น

3. โมเดลข้อมูลระดับการติดตั้ง (Implementation data models) จะแสดงถึงข้อมูลโดยโครงสร้างของเรคคอร์ด ซึ่งโมเดลนี้จะถูกใช้ในดีบีเอ็มเอส โมเดลในระดับนี้ที่แพร่หลาย ได้แก่ โมเดลข้อมูลแบบลำดับชั้น (hierarchical model) โมเดลข้อมูลแบบเครือข่าย (network model) และโมเดลข้อมูลแบบรีเลชันนัล (relation model)

ภาษาฐานข้อมูล (Database language)

ภาษาฐานข้อมูล เกี่ยวข้องกับ คำจำกัดความขององค์ประกอบฐานข้อมูล (database object) และการจัดการข้อมูล ซึ่งมีองค์ประกอบที่สำคัญดังนี้

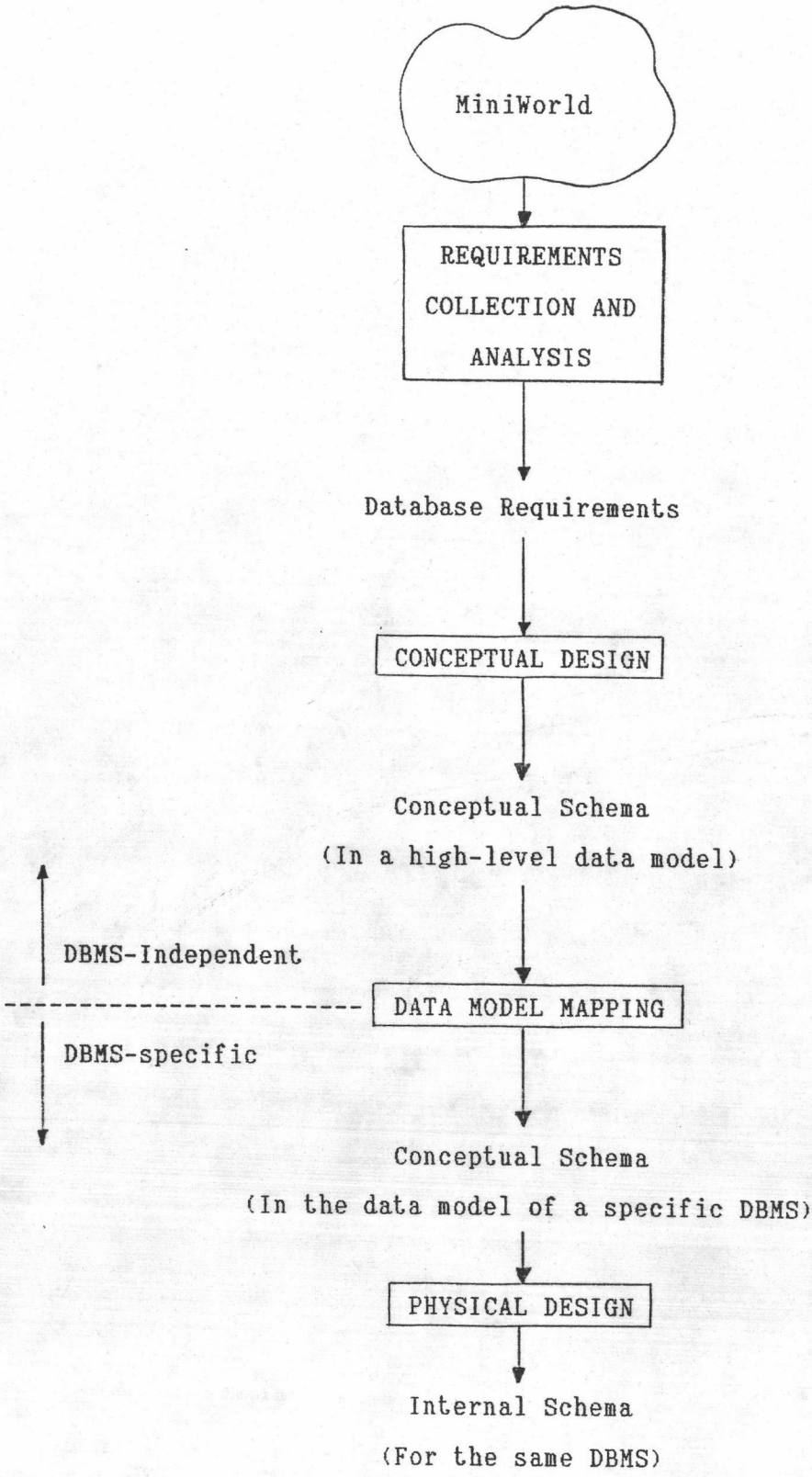
1. ดีดีแอล หรือภาษาจำกัดความข้อมูล (DDL : Data Definition Language) สำหรับให้ผู้ใช้กำหนดโครงสร้างฐานข้อมูลเชิงตรรก กล่าวคือ สามารถสร้าง แก้ไข และลบฐานข้อมูล และองค์ประกอบฐานข้อมูล ซึ่งได้แก่ ตาราง วิว ดัชนี และกฎต่างๆ เป็นต้น
2. ดีเอ็มแอล หรือภาษาจัดการข้อมูล (DML : Data Manipulation Language) สำหรับให้ผู้ใช้ลบ และแก้ไขข้อมูลตามจริงที่อยู่ในฐานข้อมูลส่วนต่างๆ ซึ่งดีเอ็มแอลอาจเป็นแบบกระบวน (procedural) หรือไม่ใช่ (nonprocedural) ตัวอย่างเช่น ในฐานข้อมูลแบบรีเลชันนัล ตารางเป็นส่วนพื้นฐานในการเก็บข้อมูล การจัดการข้อมูลจะหมายถึงการเพิ่ม หรือลบ แถวจากตาราง หรือการแก้ข้อมูลในคอลัมน์ภายในแต่ละแถว เป็นต้น

นอกจากนี้ยังมีดีซีแอล หรือภาษาควบคุมข้อมูล (DCL or Data Control Language) สำหรับใช้ในการควบคุมการเข้าถึงฐานข้อมูล และข้อมูลในฐานข้อมูล

การออกแบบระบบฐานข้อมูล

รูปที่ 3.2 แสดงให้เห็นกระบวนการในการออกแบบฐานข้อมูล ซึ่งการพัฒนาระบบฐานข้อมูลเป็นส่วนหนึ่งในการพัฒนาระบบสารสนเทศ ในการออกแบบฐานข้อมูลมีขั้นตอนดังนี้

1. ศึกษาและวิเคราะห์ความต้องการของผู้ใช้ (requirement collection and analysis) ในการวิเคราะห์ก็เพื่อที่จะสามารถกำหนดโครงสร้าง และความต้องการสำหรับระบบสารสนเทศที่ต้องการทำขึ้น เพื่อที่จะได้ทราบรายละเอียด จุดประสงค์ในการใช้งาน โดยพิจารณาว่าระบบจะต้องทำอะไรบ้าง ซึ่งจะเกี่ยวข้องกับการพิจารณาระบบที่เป็นอยู่ เป้าหมาย




รูปที่ 3.2 แสดงขั้นตอนในการออกแบบฐานข้อมูล

ในอนาคตและปัญหาต่างๆ เพื่อให้สามารถสนับสนุนกิจกรรมต่างๆ ในการทำงาน และสามารถสนับสนุนการวางแผนและช่วยในการตัดสินใจ ดังนั้นเพื่อให้บรรลุวัตถุประสงค์ดังกล่าว ต้องมีการสอบถาม สัมภาษณ์ และ รวบรวมเอกสาร รายงาน ผังองค์กร นโยบายต่างๆ ที่เกี่ยวข้อง นอกจากนี้ ต้องสังเกตและศึกษาการดำเนินการ ตลอดจนกระบวนการต่างๆ ได้แก่ การวิเคราะห์ชนิดรายการเปลี่ยนแปลง (transaction) และความถี่ที่เกิดขึ้น รวมทั้งกระแสการไหลของข้อมูล เป็นต้น

ในการรวบรวมความต้องการดังกล่าวอาจจะได้มาในรูปแบบที่ยังไม่ดีนัก รวมทั้งในการรวมภาพของระบบย่อย เข้ามาเป็นโครงสร้างทั้งหมด อาจเกิดการซ้ำซ้อนหรือมีข้อขัดแย้งกันได้ ดังนั้นอาจใช้เทคนิคต่างๆ เพื่อแปลงให้อยู่ในรูปแบบที่เข้าใจขึ้น เช่น HIPO SADT DFDs Orr-Warrnier diagrams และ Nassi-Schneiderman diagrams เป็นต้น ซึ่งอาจมีเอกสารข้อความ ตาราง ฯลฯ ประกอบกันไปกับไดอะแกรมด้วย

2. การออกแบบฐานข้อมูลเชิงมโนภาพ (Conceptual Database Design) ได้แก่ การออกแบบเค้าร่างเชิงมโนภาพ (Conceptual Schema Design) เค้าร่างเชิงมโนภาพเป็นการอธิบายความต้องการของผู้ใช้ รวมทั้งรายละเอียดของชนิดของข้อมูล ความสัมพันธ์ และ ข้อกำหนดต่างๆ ซึ่งแสดงด้วยโมเดลระดับบน อินพุทของขั้นตอนนี้คือ ผลจากการวิเคราะห์ระบบรวมทั้งความต้องการในการใช้งานและปัญหาต่างๆ ข้อกำหนดที่ได้จะสามารถใช้ในการติดตั้งระบบฐานข้อมูลแบบรีเลชันนัล เครือข่าย หรืออื่นๆ ก็ได้ โมเดลระดับบน หรือโมเดลข้อมูลเชิงมโนภาพที่ใช้นั้นควรมีลักษณะดังนี้

- แสดงความหมายแจ่มชัด สามารถชี้ให้เห็นข้อมูล ความสัมพันธ์ และข้อจำกัดต่างๆ อย่างเพียงพอ
- อยู่ในลักษณะที่ง่าย ผู้ใช้โดยทั่วไปสามารถเข้าใจ และ ใช้ง่าย
- ไม่ซับซ้อนยุ่งยาก และความหมายไม่กำกวม
- แสดงในลักษณะเป็นแผนภาพ (Diagrammatic representation)
- มีรูปแบบที่เป็นทางการ ระบุถึงข้อกำหนดของฐานข้อมูลที่ต้องการคืออะไร

ในการออกแบบเค้าร่างเชิงมโนภาพนี้ มี 2 ลักษณะ คือ

ก. วิธีออกแบบเค้าร่างแบบส่วนกลาง (Centralized schema design approach) กล่าวคือ ความต้องการของงานต่างๆ และผู้ใช้กลุ่มต่างๆ จากขั้นตอนที่ 1 จะถูกรวมเข้าเป็นชุดเดียวกันก่อนที่จะทำการออกแบบเค้าร่าง ซึ่งการรวมนี้ค่อนข้างยากและใช้เวลา

ข. วิธีรวมมุมมองเข้าด้วยกัน (View integration approach) วิธีนี้เค้าร่างหรือวิวจะถูกออกแบบสำหรับผู้ใช้แต่ละกลุ่ม หรืองานต่างๆ ตามแต่ละความต้องการ ดังนั้นจึงได้เค้าร่างในระดับบน หรือวิวสำหรับแต่ละกลุ่มผู้ใช้ จากนั้นค่อยรวมเค้าร่างต่างๆ เข้าด้วยกัน (integrate) การรวมนี้จะได้เป็นเค้าร่างเชิงมโนภาพครอบคลุม (global conceptual schema) สำหรับฐานข้อมูลทั้งหมด สำหรับฐานข้อมูลที่มีขนาดใหญ่ วิธีนี้อาจเหมาะสมและง่าย

อย่างไรก็ดี การรวมนี้ควรระวังและพิจารณาข้อขัดแย้งต่างๆ ที่เกิดขึ้นด้วย

3. การแปลงส่งโมเดลข้อมูล (Data model mapping) เป็นการติดตั้งฐานข้อมูลโดยการใช้ ดีบีเอ็มเอสที่ต้องการ ดังนั้นโมเดลระดับบน หรือ โมเดลข้อมูลเชิงมโนภาพจะถูกแปลงในรูปโมเดลข้อมูลระดับการติดตั้งของดีบีเอ็มเอสนั้นๆ ผลของขั้นตอนนี้คือ ประโยคดีดีแอล (DDL statement) ในภาษาของดีบีเอ็มเอสที่ได้เลือกแล้ว ซึ่งเป็นการกำหนดโครงสร้างเชิงตรรกของฐานข้อมูล การสร้าง เปลี่ยนแปลง และลบ ส่วนต่างๆ เป็นต้น

4. การออกแบบฐานข้อมูลเชิงกายภาพ (Physical Database Design) ขั้นตอนนี้เป็นการระบุถึงโครงสร้างหน่วยเก็บภายใน, แผนในการเข้าถึงข้อมูล และ การจัดระเบียบแฟ้มข้อมูลสำหรับฐานข้อมูล ซึ่งจุดประสงค์ในขั้นตอนนี้ คือ เพื่อให้ได้ประสิทธิภาพในการทำงานที่ดี สามารถเรียกดูหรือแก้ไขข้อมูลได้อย่างรวดเร็ว ดังนั้นในการที่จะบรรลุวัตถุประสงค์นี้สำหรับฐานข้อมูล จะต้องพิจารณาภาวะพร้อมกัน (concurrency) และการประมวลผลที่มีประสิทธิภาพโดยที่ไม่ควรกระทบการออกแบบข้อมูลเชิงตรรก ซึ่งโครงสร้างทางกายภาพอาจถูกปรับโดยการจัดกลุ่ม (clustering, partitioning) เลือกวิธีการเข้าถึงที่ดี เป็นต้น ข้อกำหนดทางกายภาพสามารถตรวจสอบความถูกต้องผ่านการทำต้นแบบและวิเคราะห์ตรวจสอบซึ่งจะช่วยในการตัดสินใจเกี่ยวเนื่องกับโครงแบบ (configuration) ของระบบเป้าหมายต่อไป

## การพัฒนาโมเดลข้อมูลเชิงตรรก

การพัฒนาโมเดลข้อมูลเชิงตรรก ซึ่งใช้ในขั้นตอน การออกแบบฐานข้อมูลเชิงมโนภาพ ช่วยให้สามารถเข้าใจและชี้ให้เห็นโครงสร้างของข้อมูล กฎ และ ความสัมพันธ์ต่างๆ ชัดเจน ซึ่งจะเป็นเสมือนข้อมูลเข้าให้แก่กระบวนการออกแบบฐานข้อมูล โมเดลข้อมูลเชิงตรรก บางครั้งใช้คำว่า โมเดลข้อมูลเชิงอรรถ (Semantic data model) เพราะโมเดลนี้จะเน้นความสำคัญที่ความหมายของข้อมูลซึ่งจะแสดงความหมายของข้อมูลในขอบเขตที่สนใจ

โดยทั่วไป โมเดลข้อมูลเชิงตรรก เกี่ยวข้องกับ การระบุถึงสิ่งสำคัญในองค์กร (Entity), คุณสมบัติของสิ่งเหล่านั้น (Attribute) และ ความสัมพันธ์ระหว่างกัน (Relationship) ในการพัฒนาโมเดลนี้ จะพิจารณาทีละกิจกรรม ซึ่งโมเดลที่แสดงให้เห็นสารสนเทศที่ต้องการสำหรับกิจกรรมหนึ่งๆ นั้น เรียกว่า มุมมองของผู้ใช้ (user view) และในที่สุด มุมมองของผู้ใช้จะถูกรวบรวมเข้าเป็น โมเดลของข้อมูลเชิงตรรก (Logical Data Model) สำหรับกระบวนการหลักในการพัฒนานั้นจะดำเนินการตามขั้นตอนดังต่อไปนี้

1. สร้างโครงร่างมุมมองของผู้ใช้
   1.1 กำหนดเอนติตีหลัก
   1.2 พิจารณาหาความสัมพันธ์ หรือ รีเลชันชิป (Relationship)

1.3 กำหนดคีย์แอตตริบิวต์ (Key Attribute)

1.4 ระบุกฎธุรกิจของคีย์ (Key Business Rules)

2. เพิ่มรายละเอียดให้กับมุมมองของผู้ใช้

2.1 เพิ่มแอตตริบิวต์ที่ไม่ใช่คีย์ (Non-key Attribute)

3. การตรวจสอบความถูกต้องผ่านการนอร์มัลไลเซชัน (Normalization)

4. พิจารณากฎธุรกิจของแอตตริบิวต์เพิ่มเติม

4.1 พิจารณาโดเมน (Domain)

4.2 พิจารณาทริกเกอร์ดำเนินการ (Triggering Operations)

5. รวมมุมมองของผู้ใช้เข้าด้วยกัน

ซึ่งแต่ละขั้นตอนมีรายละเอียดดังต่อไปนี้

1. สร้างโครงร่างมุมมองของผู้ใช้

1.1 กำหนดเอนติตีหลัก

ขั้นตอนแรกของการพัฒนาโครงร่างมุมมองของผู้ใช้ คือ การกำหนดสิ่งที่เป็นสิ่งสำคัญที่ผู้ใช้สนใจ ซึ่งเราเรียกสิ่งสำคัญนั้นว่า เอนติตี อาจเป็นสิ่งที่มีอยู่จริง สัมผัสได้ เช่น ครู ตึก สินค้า เป็นต้น หรืออาจเป็นสิ่งที่อยู่ในความคิด จินตนาการ นามธรรม เช่น การจ้างงาน วิชา คาบเวลา สี เป็นต้น โดยที่การที่จะกำหนดสิ่งใดเป็นเอนติตีย่อมขึ้นอยู่กับการให้ความสำคัญ รวมทั้งความต้องการที่ผู้ใช้จำเป็นต้องรู้ หรือต้องเก็บรวบรวมข้อมูลเหล่านั้นเพื่อให้บรรลุวัตถุประสงค์ของงาน นั่นคือผู้ออกแบบจำเป็นจะต้องเลือกว่าสิ่งใดเป็นเอนติตีที่เหมาะกับองค์กรของตน ตัวอย่างเช่น ในระบบการตลาด ข้อมูลที่จำเป็นและเกี่ยวข้องได้แก่ สินค้า ลูกค้า ใบสั่งสินค้า การส่งของ ใบกำกับรายการสินค้า การจ่ายเงิน ดังนั้นจึงสามารถกำหนดสิ่งเหล่านี้ให้เป็นเอนติตี

ในการพัฒนาโมเดล จะพิจารณาเอนติตีในลักษณะเอนติตีเซ็ท นั่นคือ ทุกๆ สมาชิก (Occurrence) ของเอนติตีเซ็ทเดียวกันจะมีความหมายและคุณสมบัติเช่นเดียวกัน เช่น ทุกสมาชิกภายในเอนติตีเซ็ทพนักงาน ก็จะหมายถึงพนักงานทั้งหมด โดยที่สมาชิกหนึ่งๆ ในเซ็ทพนักงานก็จะหมายถึงพนักงานแต่ละคน ซึ่งจะพิจารณาคุณสมบัติ หรือ หัวข้อในรายละเอียดที่สนใจอย่างเดียวกัน เช่น ชื่อ อายุ น้ำหนัก ของพนักงาน เป็นต้น ดังนั้นในที่นี้เมื่อเอ่ยถึงเอนติตีจะหมายความถึงเอนติตีเซ็ท นอกจากนี้สามารถแบ่งบางเอนติตีออกเป็นซับไทป์ หรือ เอนติตีย่อย ซึ่งมีคำจำกัดความ และคุณสมบัติบางอย่างที่เฉพาะเจาะจงมากขึ้นจากเอนติตีเดิม (หรือเรียกว่า ซุเปอร์ไทป์) เช่น แบ่งเอนติตีพนักงาน ออกเป็นพนักงานประจำ และพนักงานชั่วคราว

- การแทนสัญลักษณ์ในแผนภาพ

ในแผนภาพ ใช้รูปสี่เหลี่ยมแทนเอนติตี โดยมีชื่อเอนติตีปรากฏที่มุมบนซ้าย ชื่อเอนติตีควรสั้นและง่ายต่อการเข้าใจ

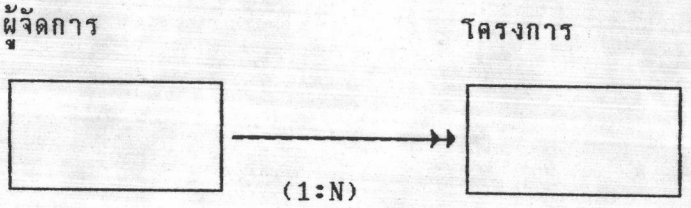
1.2 การกำหนดความสัมพันธ์ หรือรีเลชันชิป ระหว่างเอนติตี

ขั้นตอนนี้ เป็นการกำหนดความสัมพันธ์ระหว่าง ต่างเอนติตี อย่างน้อย 2 เอนติตี หรือความสัมพันธ์ ระหว่างเอนติตีเดียวกัน ตัวอย่าง เช่น ความสัมพันธ์ "การสอน" เกี่ยวพันระหว่าง 2 เอนติตี ได้แก่ เอนติตี ครู กับ เอนติตี นักเรียน

ความสัมพันธ์หนึ่งๆ จะต้องมีทิศทางซึ่งระบุว่า ความสัมพันธ์นั้นมีเอนติตีใดเป็นเอนติตีแม่ (Parent) และ เอนติตีใด เป็นเอนติตีลูก (Child) หมายความว่า เป็นความสัมพันธ์จากเอนติตีหนึ่ง (เอนติตีแม่) ไปยังอีกเอนติตีหนึ่ง (เอนติตีลูก) และจะมีสัดส่วนเป็นตัวเลขเพื่อคาดว่า จำนวนสมาชิกของเอนติตีที่มีความสัมพันธ์ต่อกันนั้นอยู่ในสัดส่วนเท่าใด เช่น อัตราส่วนระหว่างใบสั่งสินค้ากับรายการในใบสั่งสินค้า เท่ากับ 1:2 หมายความว่า ใบสั่งสินค้าหนึ่งๆ จะมี 2 รายการ เป็นต้น โดยปกติ ถ้าแยกชนิดของความสัมพันธ์ตามสัดส่วน สามารถแบ่งออกได้เป็น 3 ชนิด ดังนี้

ก. ความสัมพันธ์แบบ 1:1 หมายความว่า แต่ละสมาชิกของเอนติตีหนึ่งจะมีความสัมพันธ์ได้กับหนึ่งสมาชิกของอีกเอนติตีหนึ่งเท่านั้น เช่น ความสัมพันธ์ระหว่าง ตั๋ว กับ ผู้โดยสาร โดยที่ตั๋วแต่ละใบสำหรับผู้โดยสารหนึ่งคนเท่านั้น

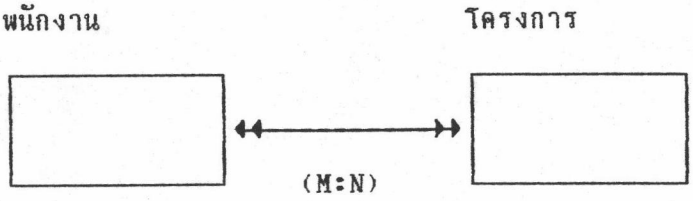
ข. ความสัมพันธ์แบบ 1:N (1:หลาย) หมายความว่า แต่ละสมาชิกของเอนติตีจะมีความสัมพันธ์กับสมาชิกของอีกเอนติตีได้มากกว่า หรือเท่ากับหนึ่งสมาชิกหรือ ไม่มีเลยก็ได้ เช่น ความสัมพันธ์ระหว่างโครงการ กับ ผู้จัดการ โดยที่โครงการแต่ละโครงการมีผู้จัดการหนึ่งคนเท่านั้น แต่ผู้จัดการคนหนึ่งๆ อาจเป็นผู้จัดการได้หลายโครงการ ตามตัวอย่างนี้ เรียกเอนติตี ผู้จัดการ ว่าเป็นเอนติตีแม่ และ เอนติตี โครงการ เป็นเอนติตี ลูก




รูปที่ 3.3 แสดงความสัมพันธ์ระหว่างผู้จัดการกับโครงการแบบ 1:N

ค. ความสัมพันธ์แบบ M:N (หลาย:หลาย) หมายความว่า แต่ละสมาชิกของเอนติตี จะมีความสัมพันธ์กับสมาชิกของอีกเอนติตี ได้มากกว่า หรือเท่ากับหนึ่งสมาชิก หรือไม่มีเลยก็ได้ รวมทั้งในทางกลับกันเช่นกัน เช่น ความสัมพันธ์ระหว่าง โครงการ กับ พนักงาน

โดยที่โครงการแต่ละโครงการ มีพนักงานได้หลายคน และพนักงานคนหนึ่งๆ อาจเป็นทำงานได้หลายโครงการ

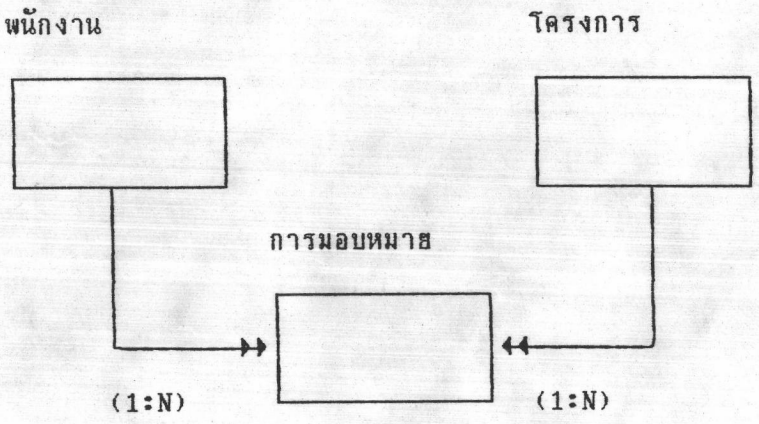



รูปที่ 3.4 แสดงความสัมพันธ์ระหว่างพนักงานกับโครงการแบบ M:N

ในการทำโมเดลข้อมูลนี้ การแสดงความสัมพันธ์ แบบ M:N นั้น เป็นสิ่งที่ซับซ้อน ในหลายระบบฐานข้อมูลไม่สนับสนุนความสัมพันธ์แบบ M:N ดังนั้น ในการกำหนดความสัมพันธ์แบบ M:N ควรมีการทำให้ง่ายขึ้น โดยการแปลงความสัมพันธ์นั้นให้เป็นเอนติตีใหม่ซึ่งสัมพันธ์กับเอนติตีเดิม 2 เอนติตีแบบ 1:N

ตัวอย่าง เช่น พนักงานคนหนึ่ง ถูกมอบหมายให้ทำ หลายโครงการ
โครงการหนึ่ง มอบหมายให้ทำโดย หลายพนักงาน

ความสัมพันธ์ระหว่างพนักงานกับโครงการเป็นแบบ M:N ดังนั้นจึงกำหนดเอนติตีใหม่ขึ้นมาชื่อ "การมอบหมาย" ซึ่งเอนติตีใหม่นี้มีความสัมพันธ์กับ เอนติตี พนักงาน และโครงการ เป็นในลักษณะ 1:N




รูปที่ 3.5 แสดงการแปลงความสัมพันธ์เป็นเอนติตีใหม่

นอกจากนี้ ยังมีชนิดของความสัมพันธ์แบบอื่นๆ อีก ได้แก่

- ความสัมพันธ์แบบซับซ้อน กล่าวคือ เป็นความสัมพันธ์ระหว่างเอนติตีที่

มากกว่า 2 เอนติตี ซึ่งจำเป็นต้องมีการปรากฏของเอนติตีทั้งหมด (มากกว่า 2) จึงจะเกิดความสัมพันธ์ขึ้นมา เพื่อให้ง่ายเข้า อาจแปลงความสัมพันธ์ให้เป็นในลักษณะไบนารีรีเลชันชิป โดยกำหนดความสัมพันธ์นั้น ขึ้นเป็นเอนติตีใหม่ แล้วสัมพันธ์กับเอนติตีเดิม แบบ 1:N

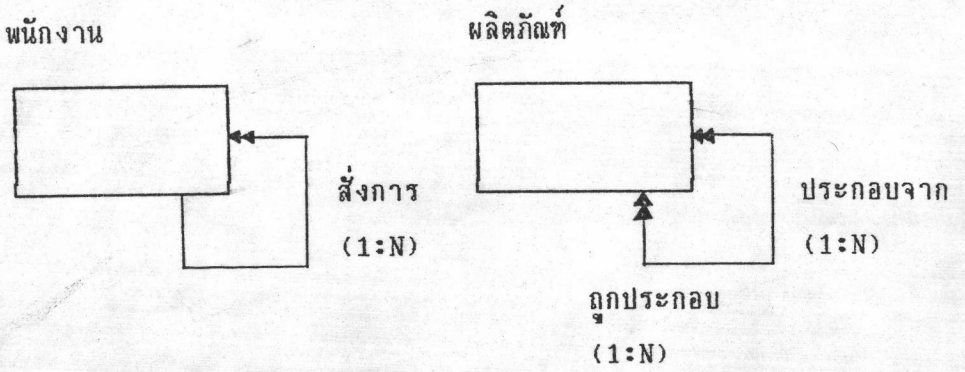
- ความสัมพันธ์แบบแบ่งจำพวก (Category) เป็นความสัมพันธ์ของเอนติตีชนิดเดียวกัน ซึ่งเป็นลักษณะ 1:1 ระหว่างซับไทป์ และ ซูเปอร์ไทป์

- ความสัมพันธ์แบบเข้าหาตัวเอง (Bill-of-materials) เป็นความสัมพันธ์ระหว่างสมาชิกของเอนติตีชนิดเดียวกัน ซึ่งอาจเป็นแบบ 1:N หรือ M:N ตัวอย่างเช่น

แบบ 1:N เช่นพนักงานคนหนึ่ง สั่งการกับ พนักงานหลายคน
พนักงานคนหนึ่ง ถูกสั่งการโดย พนักงานคนหนึ่ง

M:N เช่นผลิตภัณฑ์ชนิดหนึ่ง ประกอบมาจาก ผลิตภัณฑ์หลายชนิด
ผลิตภัณฑ์ชนิดหนึ่ง ถูกนำไปประกอบ ผลิตภัณฑ์หลายชนิด

ซึ่งในกรณี M:N จะมีการสร้างเอนติตีใหม่ขึ้นมา แล้วสัมพันธ์กับเอนติตีเดิม ผ่านความสัมพันธ์ 2 ความสัมพันธ์แบบ 1:N ดังที่เคยอธิบายมาแล้ว




รูปที่ 3.6 แสดงความสัมพันธ์แบบเข้าหาตัวเอง

- การแทนสัญลักษณ์ในแผนภาพ

ในแผนภาพ จะใช้สัญลักษณ์ของความสัมพันธ์ โดยการลากเส้นโยงระหว่างเอนติตีที่มีความสัมพันธ์กัน โดยมีหัวลูกศรตามทิศทางที่ปลายของเส้น ซึ่งจะเพิ่มหัวลูกศรเป็นลูกศรคู่ กรณีที่สมาชิกในเอนติตีมีความสัมพันธ์กับสมาชิกในอีกเอนติตีมากกว่า หนึ่งสมาชิก

1.3 การกำหนด คีย์แอตตริบิวต์

แอตตริบิวต์หมายถึงรายละเอียดซึ่งให้เห็น หรือบ่งชี้คุณลักษณะ ประเภท คุณสมบัติ หรือแสดงสถานะของเอนติตี อาจเป็นข้อความ ตัวเลข ความรู้สึก ฯลฯ เช่น ประโยคที่ว่า "อายุของพนักงาน ก คือ 24" ในที่นี้ อายุเป็นแอตตริบิวต์ของเอนติตีพนักงาน และ 24 เป็นค่าของแอตตริบิวต์อายุ นอกจากนี้ เพื่อจะให้เข้าใจ และรู้ถึงรายละเอียดของ

เอนติตีพนักงาน อาจถูกอธิบายด้วยแอตตริบิวต์อื่นๆ อีก ได้แก่ ชื่อ ที่อยู่ เงินเดือนของพนักงาน เป็นต้น

- การแทนสัญลักษณ์ในแผนภาพ

การแทนแอตตริบิวต์ในแผนภาพ จะเขียนชื่อของแอตตริบิวต์ภายในกรอบสี่เหลี่ยมของเอนติตีนั้น

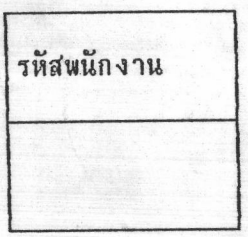
ในการกำหนดแอตตริบิวต์ให้แก่เอนติตี สามารถพิจารณาแยกความแตกต่างระหว่างสมาชิกหนึ่งของเอนติตีจากสมาชิกอื่นของเอนติตีได้ นั่นคือใช้แนวความคิดเกี่ยวกับ คีย์ ซึ่งหมายถึงแอตตริบิวต์ที่ใช้ในการกำหนดเอนติตี สำหรับขั้นตอนต่อไปนี้จะเป็นการกำหนดแอตตริบิวต์ประเภทคีย์ ดังนี้

1.3.1 กำหนด คีย์หลัก (Primary Key)

แต่ละสมาชิกในเอนติตีจะต้องถูกกำหนด หรือ ระบุความเป็นเอกลักษณ์ เช่น โดยทั่วไป จะตั้งรหัสพนักงานสำหรับพนักงานแต่ละราย ซึ่งหมายเลขในการระบุถึงพนักงานแต่ละคนย่อมเป็นเอกลักษณ์ ดังนั้นเราเรียกแอตตริบิวต์รหัสพนักงานว่าเป็นคีย์หลัก คีย์หลักอาจประกอบด้วยแอตตริบิวต์มากกว่า 1 แอตตริบิวต์ จึงเพียงพอที่จะสามารถระบุความเป็นเอกลักษณ์ก็ได้

- การแทนสัญลักษณ์ในแผนภาพ

ในแผนภาพ จะใส่แอตตริบิวต์ที่เป็นคีย์หลักไว้เหนือเส้นในกรอบ(ช่องบน) และแอตตริบิวต์ที่ไม่ใช่คีย์หลักจะอยู่ใต้เส้นในกรอบ (ช่องล่าง)




รูปที่ 3.7 แสดงเอนติตีพนักงานโดยมีรหัสพนักงานเป็นคีย์หลัก

1.3.2 กำหนด คีย์รอง (Alternate Key)

แอตตริบิวต์อื่น หรือกลุ่มของแอตตริบิวต์ อาจจะระบุความเป็นเอกลักษณ์ในเอนติตีได้เช่นกัน เช่น ถ้าในกรณีชื่อลูกค้าแต่ละคนไม่ซ้ำกัน ก็อาจเลือกแอตตริบิวต์นี้เป็นคีย์หลักได้ ดังนั้นทั้งชื่อลูกค้า และหมายเลขลูกค้า เรียกว่า เป็นแคนดิเดทคีย์ (Candidate Key) นั่นคือ สามารถเลือกแอตตริบิวต์ใดเป็นคีย์หลักก็ได้ แคนดิเดทคีย์ที่ไม่ได้ถูกเลือกเป็นคีย์หลักเรียกว่าเป็นคีย์รอง ถ้าแคนดิเดทคีย์ประกอบด้วยแอตตริบิวต์มากกว่า 1 แอตตริบิวต์ เราเรียกว่า

เป็นคีย์ประกอบ (Composite Key)

โดยทั่วไป คีย์หลักของเอนติตีซึ่งเป็นซับไทป์ของอีกเอนติตีหนึ่ง เราจะเลือกให้เหมือนกับคีย์หลักของเอนติตีที่เป็นซูเปอร์ไทป์

- การแทนสัญลักษณ์ในแผนภาพ

ในแผนภาพ จะใส่แอตตริบิวต์ที่เป็นคีย์รองไว้ในกรอบ โดยมีการระบุด้วยสัญลักษณ์ akn โดยที่ n เป็นเลขจำนวนเต็มมีค่า ตั้งแต่ 1...n หมายถึงคีย์รองตัวที่ n กรณีที่มีหลายแอตตริบิวต์ประกอบกันขึ้นเป็นหนึ่งคีย์รองแล้ว ตัวเลขต่อท้าย ak จะเป็นตัวเดียวกัน

1.3.3 กำหนด ฟอร์เรนจ์คีย์ (Foreign Key)

ฟอร์เรนจ์คีย์เป็นแอตตริบิวต์ในเอนติตีหนึ่ง ซึ่งเป็นคีย์หลักของอีกเอนติตีหนึ่ง ในกรณีที่ 2 เอนติตีมีความสัมพันธ์กันแบบ 1:N แล้ว คีย์หลักของเอนติตีแม่ย่อมเข้ามาปรากฏในเอนติตีลูก แอตตริบิวต์ที่ปรากฏในเอนติตีลูกนั้นเรียกว่า ฟอร์เรนจ์คีย์ ตัวอย่างเช่น ความสัมพันธ์ระหว่างเอนติตีลูกค้ากับเอนติตีใบสั่งสินค้า ลูกค้าคนหนึ่งอาจวางใบสั่งสินค้าได้หลายใบ ดังนั้นในเอนติตีใบสั่งสินค้า ย่อมมีฟอร์เรนจ์คีย์คือหมายเลขลูกค้า ซึ่งเป็นคีย์หลักของเอนติตีลูกค้า ดังนั้น แทนที่จะต้องใส่รายละเอียดของลูกค้าซ้ำๆ กันในใบสั่งสินค้า เพียงแต่ รู้หมายเลขลูกค้า ก็สามารถรู้รายละเอียดของลูกค้า โดยเทียบกับหมายเลขลูกค้าในเอนติตีลูกค้าให้ตรงกัน ดังนั้น จึงสามารถใช้ฟอร์เรนจ์คีย์ เพื่อหารายละเอียดจากเอนติตีที่มีค่าคีย์หลักตรงกันกับค่าฟอร์เรนจ์คีย์ได้

- การแทนสัญลักษณ์ในแผนภาพ

จะใส่ชื่อแอตตริบิวต์ ภายในกรอบเอนติตีที่เป็นเอนติตีลูก ต่อท้ายด้วย fkn โดยที่ n เป็นเลขจำนวนเต็มมีค่า ตั้งแต่ 1...n หมายถึงฟอร์เรนจ์คีย์ตัวที่ n กรณีที่มีหลายแอตตริบิวต์ประกอบกันขึ้นเป็นหนึ่งฟอร์เรนจ์คีย์แล้ว ตัวเลขต่อท้าย fk จะเป็นตัวเดียวกัน ฟอร์เรนจ์คีย์อาจเป็นส่วนหนึ่งของคีย์หลัก ในแผนภาพจะแสดงอยู่เหนือเส้นในช่องบน

1.4 การกำหนดกฎธุรกิจของคีย์

ในการทำโมเดลของข้อมูล จะต้องมีการกำหนดกฎเกณฑ์เพื่อแน่ใจว่าข้อมูลมีความถูกต้องครบถ้วนสมบูรณ์ และพิจารณาผลกระทบของการเพิ่ม ลบ แก้ไข ภายใต้ความสัมพันธ์ต่างๆ

1.4.1 กฎที่เกี่ยวข้องในระดับเอนติตี คือ การกำหนดคีย์หลักและคีย์รอง กล่าวคือ

- การกำหนดคีย์หลัก จะมีกฎ 2 ข้อ คือ

1. คีย์หลัก ต้องมีค่าสำหรับแต่ละสมาชิก (occurence) ของเอนติตี

2. คีย์หลัก ต้องเป็นเอกลักษณ์ สำหรับแต่ละสมาชิกของเอนติตี

- การกำหนดคีย์รอง จะมีกฎ 2 ข้อ คือ

1. ถ้าคีย์รองมีค่าสำหรับแต่ละสมาชิกของเอนติตี จะต้องมีความเป็นเอกลักษณ์

2. คีย์รอง อาจมีค่าหรือไม่ก็ได้ สำหรับแต่ละสมาชิกของเอนติตี

1.4.2 กฎที่เกี่ยวข้องกับความสัมพันธ์ คือการเพิ่ม ลบ ความสัมพันธ์นั้นต้องพิจารณาผลกระทบกรณีที่มีการเปลี่ยนแปลงคีย์หลัก หรือฟอร์เรนจ์คีย์ ซึ่งจะรับประกันว่าค่าของฟอร์เรนจ์คีย์นั้น จะสอดคล้องกันกับค่าคีย์หลัก โดยพิจารณาดังนี้

- การเพิ่ม เป็นการพิจารณาสถานการณ์ เงื่อนไขที่สมเหตุสมผล ถูกต้อง เมื่อเราจะเพิ่มเอนติตีลูก หรือแก้ไขฟอร์เรนจ์คีย์ในแต่ละสมาชิกในเอนติตี ซึ่งสามารถจำแนกเงื่อนไขได้เป็น 6 แบบ :

ก. โดยขึ้นต่อกัน (Dependent) กล่าวคือ จะยอมให้มีการเพิ่มสมาชิกของเอนติตีลูกเมื่อสมาชิกในเอนติตีแม่ที่ตรงกันมีอยู่

ข. โดยอัตโนมัติ (Automatic) กล่าวคือ ให้มีการเพิ่มสมาชิกของเอนติตีลูกได้เสมอ ถ้าสมาชิกในเอนติตีแม่ที่ตรงกันยังไม่มีก็สร้างขึ้นมา

ค. โดยให้ค่าว่าง (Nullify) กล่าวคือ ให้มีการเพิ่มสมาชิกของเอนติตีลูกได้เสมอ ถ้าสมาชิกในเอนติตีแม่ที่ตรงกันไม่มีอยู่ ให้ตั้งค่าฟอร์เรนจ์คีย์ในเอนติตีลูกเป็นค่าว่าง

ง. โดยให้ค่าปริยาย (Default) กล่าวคือ ให้มีการเพิ่มสมาชิกของเอนติตีลูกได้เสมอ ถ้าสมาชิกในเอนติตีแม่ที่ตรงกันไม่มีอยู่ ให้ตั้งค่าฟอร์เรนจ์คีย์ในเอนติตีลูกเป็นค่าที่กำหนดไว้ก่อน (ค่าปริยาย)

จ. โดยตามธรรมเนียม (Customized) กล่าวคือ ยอมให้มีการเพิ่มสมาชิกในเอนติตีลูกก็ต่อเมื่อตรวจสอบถูกต้องตรงกับสถานการณ์ที่กำหนด

ฉ. ไม่มีผลกระทบใดๆ (No Effect) กล่าวคือ ให้มีการเพิ่มสมาชิกของเอนติตีลูกได้เสมอ สมาชิกในเอนติตีแม่ที่ตรงกันไม่จำเป็นต้องมีอยู่ และไม่ต้องมีการตรวจสอบใดๆ

- การลบ เป็นการพิจารณาสถานการณ์ เงื่อนไขที่เป็นไปได้ที่จะสามารถลบเอนติตีแม่ได้ถูกต้อง หรือแก้ไขคีย์หลัก ที่ถูกอ้างถึงโดยฟอร์เรนจ์คีย์ ซึ่งสามารถจำแนกเงื่อนไขได้เป็น 6 แบบ :

ก. โดยจำกัด (Restrict) กล่าวคือ จะยอมให้มีการลบสมาชิกของเอนติตีแม่เมื่อไม่มีสมาชิกในเอนติตีลูกที่ตรงกันมีอยู่

ข. โดยต่อเนื่อง (Cascade) กล่าวคือให้มีการลบสมาชิกของเอนติตีแม่ได้เสมอและให้ลบสมาชิกในเอนติตีลูกที่ตรงกันทุกสมาชิก

ค. โดยให้ค่าว่าง (Nullify) กล่าวคือ ให้มีการลบสมาชิกของเอนติตีแม่ได้เสมอ ถ้ามีสมาชิกในเอนติตีลูกตรงกันมีอยู่ ให้ตั้งค่าฟอร์เรนจ์คีย์ในเอนติตีนั้นเป็นค่าว่าง

ง. โดยให้ค่าปริยาย (Default) กล่าวคือ ให้มีการลบสมาชิกของเอนติตีแม่ได้เสมอ ถ้ามีสมาชิกในเอนติตีลูกที่ตรงกันมีอยู่ ให้ตั้งค่าฟอร์เรนจ์คีย์ในเอนติตีเป็นค่าที่กำหนดไว้ก่อน

จ. โดยตามธรรมเนียม (Customized) กล่าวคือ ยอมให้มีการลบสมาชิกในเอนติตีแม่ก็ต่อเมื่อตรวจสอบตรงกับสถานการณ์ที่กำหนด

ฉ. ไม่มีผลกระทบใดๆ (No Effect) กล่าวคือ ให้มีการลบสมาชิกของเอนติตีแม่ได้เสมอ อาจมีหรือไม่มีสมาชิกในเอนติตีลูกที่ตรงกัน และไม่จำเป็นต้องมีการตรวจสอบใดๆ

ผู้ใช้ควรจะกำหนดกฎต่างๆ ให้ตรงกับภาพในธุรกิจ โดยทั่วไปควรหลีกเลี่ยงการให้ค่าว่างสำหรับฟอร์เรนจ์คีย์ เพราะบางครั้งจะไม่สมเหตุสมผลถ้าฟอร์เรนจ์คีย์นั้นเป็นส่วนหนึ่งของคีย์หลักของเอนติตีลูก สำหรับกฎในการเพิ่มความสัมพันธ์ แบบซูเปอร์ไทป์กับซับไทป์นั้น เงื่อนไขจะเป็นแบบขึ้นต่อกันหรือไม่ก็แบบอัตโนมัติ ส่วนการลบจะเป็นในลักษณะต่อเนื่อง

2. เพิ่มรายละเอียดให้กับมุมมองของผู้ใช้

2.1 การเพิ่มแอตตริบิวต์ที่ไม่ใช่คีย์ให้กับเอนติตี

นอกจากแอตตริบิวต์ทั้ง 3 ชนิดที่กล่าวมา (คีย์หลัก คีย์รอง ฟอร์เรนจ์คีย์) ยังมีแอตตริบิวต์ที่ช่วยให้รายละเอียดของเอนติตีนั้นๆ แจ่มชัดขึ้น คือแอตตริบิวต์ที่ไม่ใช่คีย์ (Non-key attribute) เช่นที่อยู่ เบอร์โทร วันเกิด เป็นแอตตริบิวต์ที่ไม่ใช่คีย์แอตตริบิวต์ในเอนติตีพนักงาน การเพิ่มแอตตริบิวต์เหล่านี้มีข้อแนะนำในการกระทำดังนี้

- หลีกเลี่ยงการใช้แอตตริบิวต์ ในรูปแบบที่เป็นการเข้ารหัส (เช่น 01=สีแดง, 02=สีฟ้า) ในกรณีที่ไม่สื่อความหมายต่อผู้ใช้ หรือไม่ใช่รหัสที่ผู้ใช้กำหนดขึ้น ถ้าจำเป็นควรกำหนดรหัสที่ไม่ควรรวมความหมายมากกว่า 1 อย่างในรหัสเดียวกัน
- ถ้ามีแอตตริบิวต์ซึ่งเป็นในลักษณะที่อธิบายความสัมพันธ์ ให้ยกขึ้นความสัมพันธ์เป็นเอนติตี และเชื่อมความสัมพันธ์ระหว่างเอนติตีใหม่เข้ากับเอนติตีเดิมทั้งสอง
- บางครั้งอาจรวมแอตตริบิวต์ที่มีความสำคัญ ซึ่งได้มาจากแอตตริบิวต์อื่นเข้าในโมเดลด้วย แอตตริบิวต์ที่มาจากแอตตริบิวต์อื่น เรียกว่า ดิไรฟ์วแอตตริบิวต์ (Derived attribute) เช่น แอตตริบิวต์ที่ได้มาจากการคำนวณจากแอตตริบิวต์อื่นๆ (ในแผนภาพจะใช้

สัญลักษณ์ d ต่อท้ายแอตตริบิวต์)

- แอตตริบิวต์ ที่มีลักษณะร่วม (Common) ควรอยู่ที่ ซูเปอร์ไทป์มากกว่าอยู่ที่ซับไทป์ ส่วนแอตตริบิวต์ที่ไม่ได้เป็นลักษณะร่วมแต่เป็นเอกลักษณ์ในแต่ละซับไทป์ก็ควรอยู่ในซับไทป์ที่เหมาะสม

- รวมเอนติตีเป็นเอนติตีเดียวกันเมื่อไม่ทำให้ความหมายเสียไป โดยทั่วไปจะรวม

ก. เอนติตีที่มีคีย์หลักเหมือนกัน ยกเว้นว่า เอนติตีนั้นมีความหมายคนละอย่าง

ข. ซับไทป์ที่มีแอตตริบิวต์และความสัมพันธ์เหมือนกัน

ค. ซับไทป์กับซูเปอร์ไทป์ ถ้าทุกสมาชิกของซูเปอร์ไทป์ จะมีสมาชิกหนึ่งของซับไทป์เสมอ (subtype spans supertype) ถ้ามิฉะนั้นรวมกันแล้วอาจทำให้บางแอตตริบิวต์เป็นค่าว่างได้

ง. เอนติตีแม่รวมกับเอนติตีลูก เมื่อเอนติตีแม่ไม่มีแอตตริบิวต์ที่ไม่ใช่คีย์

3. การตรวจสอบความถูกต้องผ่านการนอร์มัลไลเซชัน

จากขั้นตอนต่างๆ ที่ผ่านมาเป็นการรวมแอตตริบิวต์เข้ามาในเอนติตี ซึ่งการที่ตรวจว่าแอตตริบิวต์เหล่านั้นอยู่ในเอนติตีที่เหมาะสมหรือไม่ จะใช้เทคนิคการนอร์มัลไลเซชัน โดยวิเคราะห์แยกโครงสร้างของข้อมูล ซึ่งการทำนอร์มัลไลเซชันเป็นกระบวนการให้แน่ใจว่า โมเดลของข้อมูลอยู่ในรูปแบบที่เป็นมาตรฐาน ลดความซ้ำซ้อนของข้อมูล โดยที่มีการใส่แอตตริบิวต์ที่เหมาะสมให้กับเอนติตี หรือบางทีอาจมีการสร้างเอนติตีใหม่ขึ้นมา และ ช่วยให้มีความยืดหยุ่นต่อความต้องการในการใช้งาน และยังทำให้สามารถเข้ากับการออกแบบฐานข้อมูลได้กว้างขวางขึ้น ปัจจุบันมีการทำนอร์มัลไลเซชันในระดับต่างๆ 5 ระดับ ซึ่งโดยปกติโมเดลข้อมูลที่อยู่ในรูปแบบนอร์มัลระดับที่ 3 (3NF) นั้นเพียงพอแล้ว

ก่อนที่จะพิจารณาวิธีในการทำนอร์มัลไลเซชัน จะสมมติให้โมเดลของข้อมูลตามตัวอย่าง เป็นเอนติตีพนักงาน ประกอบด้วยแอตตริบิวต์ต่างๆ ตามตารางที่ 3.1

พนักงาน

| รหัสพนักงาน | ชื่อ | สกุล | ประเภททักษะ | ชื่อทักษะ | ระดับทักษะ | ตำแหน่ง | เงินเดือน |
|---|---|---|---|---|---|---|---|
| 001 | สมใจ | ดีงาม | CSC01 | โปรแกรม ซี | 01 | โปรแกรมเมอร์ 1 | 5500 |
| 002 | ประสิทธิ์ | หมั่นเพียร | MGMT | บริหาร | 01 | บริหาร | 8000 |
| 001 | สมใจ | ดีงาม | CSC08 | วิเคราะห์ระบบ | 02 | โปรแกรมเมอร์ 1 | 5500 |

ตารางที่ 3.1 แสดงข้อมูลของเอนติตีพนักงาน

วิธีการทำนอร์มัลไลเซชันในระดับต่างๆ ตามลำดับ มีดังนี้

- 1NF : การทำนอร์มัลในระดับที่ 1 (First Normal Form)

นำแอตตริบิวต์ที่ซ้ำหรือกลุ่มของแอตตริบิวต์ที่ซ้ำออก และยกขึ้นเป็นอีกเอนติตีหนึ่งซึ่งคีย์หลักของเอนติตีนี้ประกอบด้วยคีย์หลักของเอนติตีเดิมและคีย์ของกลุ่มที่ซ้ำกัน จากตัวอย่างจะเห็นว่าพนักงานคนหนึ่งอาจมีทักษะหลายอย่าง ดังนั้นจึงมีการแยกออกเป็นเอนติตีใหม่ ซึ่งแทนทักษะของแต่ละคน โดยประกอบด้วย รหัสพนักงาน ประเภททักษะ ชื่อทักษะ และคีย์หลักก็คือประเภททักษะ กับรหัสพนักงาน

พนักงาน

| รหัสพนักงาน | ชื่อ | สกุล | ตำแหน่ง | เงินเดือน |
|---|---|---|---|---|
| 001 | สมใจ | ดีงาม | โปรแกรมเมอร์ 1 | 5500 |
| 002 | ประสิทธิ์ | หมั่นเพียร | บริหาร | 8000 |

ความชำนาญแต่ละคน

| รหัสพนักงาน | ประเภททักษะ | ชื่อทักษะ | ระดับทักษะ |
|---|---|---|---|
| 001 | CSC01 | โปรแกรม ซี | 01 |
| 002 | MGMT | บริหาร | 01 |
| 001 | CSC08 | วิเคราะห์ระบบ | 02 |

ตารางที่ 3.2 แสดงเอนติตีพนักงานใน 1NF

- 2NF : การทำนอร์มัลในระดับที่ 2 (Second Normal Form)

เอนติตีนั้นต้องอยู่ใน 1NF และนำแอตตริบิวต์ที่ขึ้นอยู่กับบางส่วนของคีย์หลักออก กล่าวคือ ถ้ามีเอนติตีที่มีคีย์หลักซึ่งประกอบด้วยแอตตริบิวต์มากกว่า 1 แอตตริบิวต์ และถ้าแอตตริบิวต์อื่นๆ ขึ้นกับบางส่วนของคีย์หลักนั้นแล้ว ให้ยกเป็นเอนติตีใหม่ แต่ตามตัวอย่างเอนติตีพนักงานมีคีย์หลักเป็นแอตตริบิวต์เพียงแอตตริบิวต์เดียว ดังนั้นเอนติตีพนักงานจึงอยู่ในรูปแบบที่เป็น 2NF อยู่แล้ว และจะเห็นว่าเอนติตีประเภททักษะจะถูกสร้างขึ้น เพื่อแก้ปัญหาความสัมพันธ์แบบ 1:1 ระหว่างประเภททักษะกับชื่อทักษะ

พนักงาน

| รหัสประจำตัว | ชื่อ | สกุล | ตำแหน่ง | เงินเดือน |
|---|---|---|---|---|
| 001 | สมใจ | ดีงาม | โปรแกรมเมอร์ 1 | 5500 |
| 002 | ประสิทธิ์ | หมั่นเพียร | บริหาร | 8000 |

ทักษะแต่ละคน

| รหัสพนักงาน | ประเภททักษะ | ระดับ |
|---|---|---|
| 001 | CSC01 | 01 |
| 002 | MGMT | 01 |
| 001 | CSC08 | 02 |

ประเภททักษะ

| ประเภททักษะ | ชื่อทักษะ |
|---|---|
| CSC01 | โปรแกรม ซี |
| MGMT | บริหาร |
| CSC08 | วิเคราะห์ระบบ |

ตารางที่ 3.3 แสดงเอนติตีพนักงานใน 2NF

- 3NF : การทำนอร์มัลในระดับที่ 3 (Third Normal Form)

เอนติตีนั้นต้องอยู่ใน 2NF และนำแอตตริบิวต์ที่ขึ้นกับแอตตริบิวต์ที่ไม่ใช่คีย์หลักหรือไม่ใช่บางส่วนของคีย์หลักออก และยกขึ้นเป็นเอนติตีใหม่ ตามตัวอย่าง สังเกตว่าในเอนติตีพนักงาน เงินเดือนเป็นค่าที่แน่นอน ซึ่งขึ้นอยู่กับตำแหน่ง ดังนั้น จึงแยกออกมา เป็นเอนติตีใหม่ ซึ่งประกอบด้วย ตำแหน่งและ เงินเดือน

พนักงาน

| รหัสประจำตัว | ชื่อ | สกุล | ตำแหน่ง |
|---|---|---|---|
| 001 | สมใจ | ดีงาม | โปรแกรมเมอร์ 1 |
| 002 | ประสิทธิ์ | หมั่นเพียร | บริหาร |

งาน

| ตำแหน่ง | เงินเดือน |
|---|---|
| โปรแกรมเมอร์ 1 | 5500 |
| บริหาร | 8000 |

ทักษะแต่ละคน

| รหัสพนักงาน | ประเภททักษะ | ระดับ |
|---|---|---|
| 001 | CSC01 | 01 |
| 002 | MGMT | 01 |
| 001 | CSC08 | 02 |

ประเภททักษะ

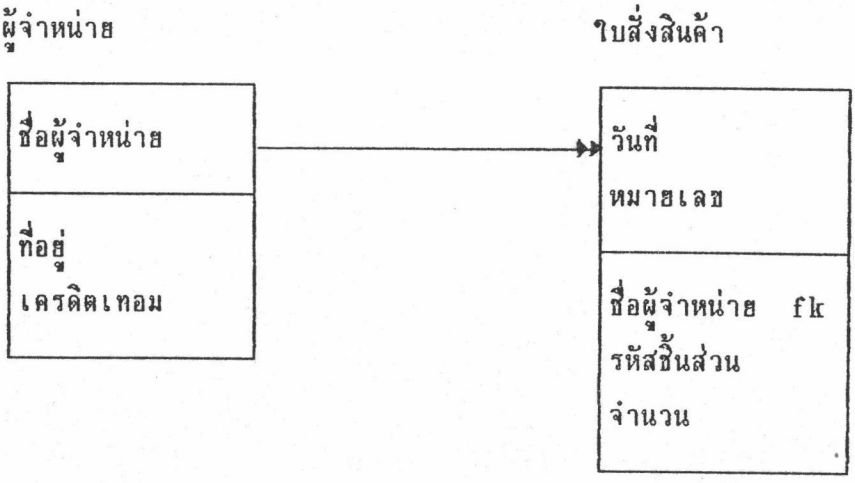
| ประเภททักษะ | ชื่อทักษะ |
|---|---|
| CSC01 | โปรแกรม ซี |
| MGMT | บริหาร |
| CSC08 | วิเคราะห์ระบบ |

ตารางที่ 3.4 แสดงเอนติตีพนักงานใน 3NF

ข้อควรระวังในการทำนอร์มัลไลเซชัน

ก. ไม่ควรแยกเอนติตีออกไปอีกเมื่อทำมาถึงรูปแบบที่เป็น 3NF แล้ว (Overnormalize) (ถ้าไม่ได้แยกเพื่อทำให้เป็นรูปแบบที่นอร์มัลในระดับที่ 4 หรือ 5) เพราะจะทำให้โมเดลยุ่งยากเกินความจำเป็น

ข. พิจารณาความจำเป็นที่ต้องเพิ่มแอตตริบิวต์ หรือ เอนติตีเกี่ยวกับการเก็บประวัติต่างๆ ตัวอย่าง เช่น ความสัมพันธ์ระหว่างผู้จำหน่าย กับใบสั่งสินค้า ตามรูปที่ 3.8




รูปที่ 3.8 แสดงความสัมพันธ์ระหว่างผู้จำหน่ายกับใบสั่งสินค้า

ในการลบ หรือแก้ไขในเอนติตีผู้จำหน่ายลองพิจารณาในแง่ของเวลาประกอบด้วยว่าจะมีผลอย่างไร เช่น การจัดการในกรณีที่ผู้จำหน่ายย้ายที่อยู่ จำเป็นหรือไม่ที่ต้องรู้ที่อยู่ของผู้จำหน่ายในใบสั่งสินค้าเก่า เป็นต้น คำตอบที่เป็นไปได้ในการดำเนินการ ได้แก่

- อาจจะยอมให้มีการสูญเสียข้อมูล เช่น ในกรณีนี้คือ ไม่สนใจที่อยู่เก่าของผู้จำหน่าย
- เพิ่มแอตตริบิวต์ที่ต้องการขึ้นมา ตามตัวอย่างนี้อาจเพิ่มแอตตริบิวต์ที่อยู่ในเอนติตีใบสั่งสินค้า ดังนั้นจะมีการเก็บข้อมูลที่อยู่ผู้จำหน่ายใน 2 เอนติตี
- สร้างซับไทป์ของผู้จำหน่าย เป็นผู้จำหน่าย(ประวัติ) และผู้จำหน่าย(ปัจจุบัน) ซึ่งต้องระบุกฎในการลบ เช่น ถ้าลบในเอนติตีผู้จำหน่าย(ปัจจุบัน) ก็ให้เพิ่มในเอนติตีผู้จำหน่าย(ประวัติ) เป็นต้น

อย่างไรก็ตาม การทำนอร์มัลไลเซชัน อาจจะยังไม่ใช่คำตอบที่ถูกต้องเสมอไป สำหรับปัญหาในการออกแบบฐานข้อมูล ยังมีปัจจัยอื่นที่ต้องพิจารณา ในการพัฒนาโมเดลเชิงตรรกอีกด้วย

## 4. การพิจารณากฎธุรกิจของแอตตริบิวต์เพิ่มเติม

### 4.1 การพิจารณาโดเมน (Domain)

โดเมน คือเซ็ทของค่าที่เป็นไปได้ในแอตตริบิวต์หนึ่งๆ หลายๆ แอตตริบิวต์อาจมีโดเมนเหมือนกัน เช่น แอตตริบิวต์วันเกิด วันทำสัญญา วันจ้างงาน แอตตริบิวต์เหล่านี้มีโดเมนเป็น "วัน" (Date) หรือ แอตตริบิวต์ราคาสินค้า เงินเดือน แอตตริบิวต์เหล่านี้มีโดเมนเป็น "เงิน" เป็นต้น โดเมนหนึ่งจะแสดงช่วงของค่าของแอตตริบิวต์ที่เป็นไปได้ และอาจมีการแทนค่าในหลายลักษณะเช่น ค่าในโดเมน "เงิน" อาจอยู่ในหน่วยของดอลลาร์ บาท เป็นต้น

หรือค่าในโดเมน "ความยาว" อาจอยู่ในหน่วยของเมตร กิโลเมตร เป็นต้น ซึ่งแต่ละแอตตริบิวควรกำหนดลักษณะต่างๆ ของโดเมนได้แก่

- ชนิดของข้อมูล
- ความยาวของข้อมูล
- ความหมาย
- ระบุว่าเป็นค่าว่างได้หรือไม่
- รูปแบบ
- ค่าที่เป็นไปได้ หรือ ข้อจำกัด
- ความเป็นเอกลักษณ์
- ค่าโดยปริยาย

นอกจากนี้ ถ้าแอตตริบิวต์นั้นเป็นแอตตริบิวต์ที่ได้มาจากแอตตริบิวต์อื่น (Derived attribute) ควรระบุอัลกอริทึมในการได้มานั้นด้วย

4.2 การพิจารณากฎธุรกิจของแอตตริบิวต์อื่นๆ (Triggering Operations)

ทริกเกอร์ดำเนินการ เป็นกฎซึ่งจัดการเกี่ยวกับความถูกต้องสมบูรณ์ของค่าของแอตตริบิวต์เมื่อมีการเพิ่ม ลบ แก้ไข รวมกระทั่งการดำเนินการเกี่ยวกับผลกระทบในการปฏิบัติการเหล่านั้นบนเอนติตีอื่น หรือบนแอตตริบิวต์อื่นในเอนติตีเดียวกัน ทริกเกอร์ดำเนินการจะมีองค์ประกอบ 2 อย่าง คือ ทริกเกอร์ หรือเหตุการณ์ หรือเงื่อนไข ซึ่งเป็นสาเหตุให้เกิดการปฏิบัติการต่างๆ ขึ้น และ องค์ประกอบอีกอย่าง ก็คือ การดำเนินการ หรือการปฏิบัติการ การกำหนดและระบุลอจิกสำหรับทริกเกอร์ดำเนินการจะช่วยให้การทำงานในขั้นตอนต่อไปสะดวกขึ้น ดังนั้นในการจัดทำเอกสารเพื่อการออกแบบหรือบันทึกลงในพจนานุกรมข้อมูลเกี่ยวกับทริกเกอร์ดำเนินการ จะระบุถึงส่วนต่างๆ ดังนี้

- เหตุการณ์ ซึ่งเป็นสาเหตุให้เกิดการปฏิบัติการต่างๆ ขึ้น
- องค์ประกอบของเหตุการณ์ ได้แก่ ชื่อเอนติตี และ/หรือ แอตตริบิวต์ที่เกี่ยวข้อง
- เงื่อนไข ภายใต้ทริกเกอร์ดำเนินการที่เกิดขึ้น
- การกระทำที่ตามมาสืบเนื่องจากเหตุการณ์ดังกล่าว

ตัวอย่าง สถานการณ์ที่จะควรกำหนดทริกเกอร์ดำเนินการ เช่น ในกรณีแก้ไขแอตตริบิวต์ที่เป็นที่มาของดิไรฟ์วแอตตริบิวต์ กรณีที่ชับไทป์ถูกลบและต้องลบซูเปอร์ไทป์ เป็นต้น และไม่ว่าเราจะอ้างกฎธุรกิจของคีย์ กฎโดเมน หรือ ทริกเกอร์ดำเนินการหรือไม่ ไม่ใช่สิ่งสำคัญ จุดมุ่งหมายที่สำคัญก็คือ การระบุถึงเงื่อนไขความเป็นบูรณภาพของข้อมูล ในโมเดลให้ครบถ้วนและถูกต้อง แต่การแยกพิจารณาเป็นกฎชนิดต่างๆ ก็เพื่อให้เห็นความเป็นไปได้ และเป็นสิ่งที่ต้องคำนึงถึงขณะที่สอบถามผู้ใช้ คำถามที่เฉพาะเจาะจงเกี่ยวกับความสัมพันธ์ แอตตริบิวต์และเอนติตีจะทำให้ได้อย่างน้อยสุด คือกฎธุรกิจของคีย์ และกฎโดเมน ส่วนทริกเกอร์ดำเนินการอาจได้มาจากคำวิจารณ์ของผู้ใช้ ข้อคิดเห็น หรือคำถามอื่นๆ ที่ลึกลงไป

5. เชื่อมโยงและรวบรวมมุมมองของผู้ใช้เข้าด้วยกัน

ขั้นตอนนี้ จะเป็นการรวบรวมมุมมองของผู้ใช้ที่ถูกกำหนดจากกลุ่มผู้ใช้ต่างๆ หรือถูกกำหนดมาจากกิจกรรมต่างๆ เข้าด้วยกัน เป็นโมเดลเดียว รวมสิ่งที่มีการคาบเกี่ยวกัน และแก้ปัญหาความไม่สอดคล้อง เพื่อให้ได้โมเดลข้อมูลเชิงตรรกที่ประกอบกันอย่างสมบูรณ์ (Composite logical data model) ซึ่งต้องมีการวิเคราะห์พิจารณาส่วนที่แตกต่าง และส่วนที่เหมือนกัน รวมทั้งการเชื่อมโยงระหว่างมุมมอง

การเชื่อมโยงและรวบรวมหลายๆ มุมมองของผู้ใช้เข้าด้วยกัน อาจกระทำดังนี้

- รวมเอนติตีที่หมายถึงสิ่งเดียวกัน มีคีย์หลักเดียวกัน และโดเมนของคีย์หลักเทียบเท่ากัน โดยที่รวมแอตตริบิวต์จากเอนติตีเหล่านั้นเข้าด้วยกัน และขจัดแอตตริบิวต์ที่ซ้ำซ้อนออก
- สร้างความสัมพันธ์แบบซับไทป์-ซูเปอร์ไทป์ขึ้นมา ให้กับเอนติตีที่เกี่ยวข้องที่มีคีย์หลักเช่นเดียวกัน และโดเมนของคีย์หลักเหมือนกัน ซึ่งข้อกำหนดของค่าที่เป็นไปได้อาจมีค่าคาบเกี่ยวกัน, เป็นสับเซ็ทกัน แต่ไม่เทียบเท่ากัน
- รวมความสัมพันธ์ระหว่างเอนติตีเมื่อมีความหมายในทางเดียวกัน
- ขจัดความสัมพันธ์ที่ซ้ำซ้อน และเติมความสัมพันธ์ที่ต้องมีระหว่างกัน (inter-view)
- ตรวจสอบผ่านการนอร์มัลไลเซชันอีกครั้ง
- เมื่อทำการเชื่อมโยงแล้ว ควรพิจารณาแอตตริบิวต์ ว่ากฎธุรกิจต่างๆ หรือคุณสมบัติต่างๆ ไปด้วยกันหรือถูกต้องหรือไม่

หลังจากนี้ นำโมเดลที่ได้มารวมเข้ากับโมเดลที่มีอาจมีอยู่เดิม เป็นเค้าร่างเชิงมโนภาพของธุรกิจ ซึ่งในที่สุดจะได้เป็นโมเดลข้อมูลที่จะแสดงความต้องการของธุรกิจทั้งหมด (โครงสร้าง ความหมาย ข้อบังคับ) ในรายละเอียด และในขั้นตอนสุดท้าย คือการวิเคราะห์และพิจารณาถึงการเปลี่ยนแปลงในอนาคตซึ่งอาจมีผลกระทบกับโมเดลที่ได้

<u>การออกแบบฐานข้อมูลแบบรีเลชันนัล</u> (Relational Database Design)

จากการวางแผนและวิเคราะห์ เมื่อได้โมเดลของข้อมูล ซึ่งจะได้แผนภาพแสดงความสัมพันธ์ระหว่างเอนติตีแล้ว ขั้นตอนต่อมาเป็นการออกแบบฐานข้อมูลแบบรีเลชันนัล กล่าวคือเป็นกระบวนการแปลงโมเดลข้อมูลเชิงตรรกให้เป็นฐานข้อมูลแบบรีเลชันนัล ซึ่งขั้นตอนเหล่านี้ยังช่วยสนับสนุนในการพัฒนาต้นแบบ โดยในขั้นตอนนี้สามารถแบ่งออกเป็นขั้นตอนต่างๆ ได้ดังนี้ คือ

1. แปลงจากโครงร่างข้อมูลแบบตรรกภาพ ให้เป็นฐานข้อมูลแบบรีเลชันนัลในเบื้องต้น ขั้นตอนนี้จะพิจารณาความเป็นไปได้และองค์ประกอบต่างๆ ว่าครบถ้วนหรือไม่ ซึ่งจะกระทำดังนี้

ก. กำหนดตาราง (Table) สำหรับแต่ละเอนติตี ตลอดจนเอนติตีที่เป็นซับไทป์และซูเปอร์ไทป์ด้วย

ข. กำหนดคอลัมน์ (Column) ในแต่ละตารางสำหรับแต่ละแอตตริบิวต์ของเอนติตี

จะเห็นว่าในเบื้องต้นนี้ แต่ละตารางจะมีคอลัมน์ ซึ่งประกอบด้วยคีย์หลัก คีย์รอง ฟอร์เรนจ์คีย์ รวมทั้งคอลัมน์ที่ไม่ใช่คีย์ ซึ่งการกำหนดคอลัมน์ของฟอร์เรนจ์คีย์นั้น มาจากการแปลงความสัมพันธ์ที่ได้จากโมเดลข้อมูลเชิงตรรกนั่นเอง และขณะนี้ตารางได้อยู่ในรูปแบบนอร์มัล เพราะโมเดลข้อมูลเชิงตรรกอยู่ในรูปแบบนอร์มัลอยู่แล้ว

ค. ปรับโครงสร้างข้อมูลให้เข้ากับสภาพแวดล้อมที่นำมาใช้ ทุกๆ ระบบฐานข้อมูลแบบรีเลชันนัล (RDBMS) จะมีการให้จำกัดความของตารางผ่านทางประโยคดีดีแอล (DDL = Data Definition Language) ซึ่งไวยากรณ์ (Syntax) อาจต่างกันในแต่ละระบบ เราจะต้องปรับให้เข้ากับระบบที่ต้องการ เช่น พิจารณาการสร้างตารางเก็บในระบบฐานข้อมูล การเรียงลำดับคอลัมน์ในตาราง เป็นต้น ในการกำหนดตารางให้กับดีบีเอ็มเอส ยังต้องพิจารณาถึงพารามิเตอร์ที่มีผลกระทบกับการจัดสรร (allocate) และจัดการของหน่วยเก็บ รวมทั้งเนื้อที่ที่ว่างด้วย ซึ่งเนื้อที่ต้องมากเพียงพอ

2. แปลความเป็นบูรณภาพของข้อมูลเชิงตรรก กระบวนการแปลในที่นี้คือการบังคับให้เป็นไปตามกฎธุรกิจต่างๆ ซึ่งบางครั้งระบบจัดการฐานข้อมูลที่นำมาใช้ อาจจะไม่สนับสนุนการทำงานบางอย่าง ก็อาจใช้กลไกอื่นๆ ช่วย สำหรับขั้นตอนต่างๆ ในขั้นนี้ได้แก่

ก. บังคับคุณสมบัติเชิงตรรกของคีย์หลักของเอนติตี ในการนำมาใช้งาน เช่น ความเป็นเอกลักษณ์ การไม่ยอมให้มีค่าว่าง ฯลฯ โดยอาจกำหนดคุณสมบัติเหล่านี้ ผ่านทางประโยคดีดีแอล หรือกำหนดโดยผ่านเทคนิคการใช้ข้อกำหนดของโดเมน

ข. บังคับคุณสมบัติเชิงตรรกของรีเลชันชิป เกี่ยวกับข้อบังคับการเพิ่ม ลบ แก้ไขคีย์แอตตริบิวต์ โดยอาจกำหนดผ่านทางประโยคดีดีแอล หรือใช้เทคนิคการทำทริกเกอร์ มิฉะนั้นอาจเป็นหน้าที่รับผิดชอบของผู้ใช้ในการที่ต้องปฏิบัติตามข้อบังคับ

ค. บังคับกฎธุรกิจของแอตตริบิวต์ ซึ่งอาจผ่านทางทริกเกอร์ ประโยคดีดีแอล หรือผ่านโดยข้อกำหนดโดเมน และโดยปกติไม่ควรยอมให้คอลัมน์บางคอลัมน์เป็นค่าว่าง ได้แก่คอลัมน์ที่เป็นคีย์หลัก ฟอร์เรนจ์คีย์ซึ่งมักถูกใช้ในการเปรียบเทียบและเชื่อมโยง (join) คอลัมน์ที่ใช้ในเอสคิวแอล WHERE syntax และคอลัมน์ที่เป็นตัวเลข เพราะจะเป็นปัญหาในการคำนวณ

3. <u>การปรับแต่งการออกแบบ</u> (Tuning) เราสามารถใช้วิธีการหลายอย่าง เพื่อช่วยในการปรับเพื่อเพิ่มประสิทธิภาพในการทำงาน เช่น การวิเคราะห์กลไกในการเข้าถึง (access) โดยหาทางเลือกที่ให้ประโยชน์สูงสุด การสร้างดัชนี ซึ่งข้อแนะนำคือ ควรสร้างดัชนีในตารางที่มีขนาดกลางจนถึงขนาดใหญ่ หรือถ้าในการเรียกใช้หรือดึงข้อมูลในตารางนั้นๆ แล้วได้ผลลัพธ์ประมาณ 20% ของจำนวนแถว และในการประเมินประสิทธิภาพของดัชนีนั้น จะต้องพิจารณาผลกระทบในการเพิ่ม ลบ แก้ไข ตลอดจนการประมวลผลต่างๆ แล้ว ยังต้องพิจารณาค่าใช้จ่ายเนื่องจากความต้องการหน่วยเก็บอีกด้วย

นอกจากนี้ถ้าวิธีการปรับที่กล่าวมาแล้วไม่เพียงพออาจมีการกระทำต่างๆ ดังนี้

ก. ยอมให้ข้อมูลมีการซ้ำซ้อนโดยมีการควบคุม

สิ่งสำคัญที่ต้องพิจารณาคือ อาจมีเหตุผลที่ทำให้ สำหรับเอนติตีหนึ่งๆ ที่วิเคราะห์ออกมาอาจไม่กลายเป็นตาราง 1 ตารางในฐานข้อมูลเสมอไป เหตุผล 2 ประการโดยทั่วไปก็คือ

1. เอนติตีที่ถูกกำหนดขึ้นมาไม่ได้เกี่ยวพันกับการใช้งานด้วยคอมพิวเตอร์ขณะที่กำลังทำอยู่ ซึ่งเราอาจจะตัดสินใจหรือไม่ก็ได้ ว่าจะยกเอนติตีนี้เป็นตารางต่อไป ถ้าหากมีการขยายการใช้งาน

2. อาจต้องการรวมเอนติตี 2 เอนติตี หรือมากกว่า เป็นเพียง 1 ตาราง เพื่อเพิ่มประสิทธิภาพในการทำงาน เช่น การที่กำหนดเอนติตีเป็นสิ่งสำคัญที่มองแยกออกไป แต่ในโครงสร้างของตารางจะรวมกันเข้าเพื่อเลี่ยงการเสียเวลาในการเชื่อมข้อมูลต่อกัน และบางครั้ง การทำให้อยู่ในรูปนอร์มัลอาจทำให้เกิดความซับซ้อนเกินจำเป็น ดังนั้น อาจต้องมีการเปลี่ยนแปลง นั่นคือ มีการทำให้อยู่ในรูปดีนอร์มัล (Denormalize) ซึ่งจะก่อให้เกิดความซ้ำซ้อนของข้อมูลในฐานข้อมูลขึ้นใหม่ แต่ก่อนที่จะพิจารณาการดีนอร์มัล จะต้องผ่านกระบวนการทำนอร์มัลในระดับที่ 3 ก่อน และจะต้องตัดสินใจมากขึ้น เพื่อให้แน่ใจว่าข้อมูลที่ซ้ำกันนั้นไม่ลักหลั่นกัน ซึ่งถ้าทำให้อยู่ในรูปดีนอร์มัลแล้ว อาจทำให้การพัฒนางานง่ายขึ้น และทำให้เพิ่มประสิทธิภาพการเรียกดูหรือค้นหาข้อมูล ไม่ต้องสืบค้นหลายตาราง หรือไม่เสียเวลาในการคำนวณ เป็นต้น แต่อย่างไรก็ตามต้องพิจารณา ผลดีผลเสียที่เกิดขึ้น ซึ่งจะมีผลกระทบกับหน่วยความจำ การดึงข้อมูล ประโยคดีเอ็มแอลที่ใช้ ความเป็นบูรณภาพของข้อมูล ตลอดจนประสิทธิภาพในการใช้งาน

คอลัมน์ที่ซ้ำซ้อนอาจได้มาจาก

- คอลัมน์ที่ลอกมาจากคอลัมน์ที่มีอยู่แล้วในตารางอื่น เช่น ถ้ามีตารางเก็บประวัติการทำงานแต่เราอาจเก็บการทำงานปัจจุบันในตารางพนักงานอีกด้วย ทำให้มีข้อมูลปรากฏ 2 แห่ง เป็นต้น

- คอลัมน์ที่เป็นผลสรุปมาจากคอลัมน์อื่นๆ (derived column) ซึ่งมีความสำคัญต่อผู้ใช้ ทำให้สามารถตอบคำถามในการสอบถามได้เร็วขึ้น แต่การยอมให้ผู้ใช้แก้ไขเปลี่ยนแปลงค่าควรกระทำที่คอลัมน์ที่เป็นตัวกำเนิด หรือที่มาเท่านั้น

- คอลัมน์ที่เป็นกลุ่มซ้ำ เมื่อกลุ่มซ้ำนั้นมีจำนวนแน่นอน และกลุ่มซ้ำนั้นมักถูก

เรียกใช้หรือแก้ไขไปพร้อมๆ กัน เช่นถ้าทราบว่า พนักงานแต่ละคนมีทักษะ 2 อย่างเสมอ เราอาจเก็บเป็น 2 คอลัมน์ในตาราง แต่ปัญหาก็คือ หากภายหลังต้องการเพิ่มประเภททักษะชนิดที่ 3 จะจัดการอย่างไร ดังนั้นต้องพิจารณาผลดีผลเสียในการจำกัดความยืดหยุ่นของโมเดล

- คอลัมน์อย่างย่อที่สร้างขึ้นเพื่อแทนคอลัมน์ที่มีอยู่ เช่นในกรณีที่คีย์หลักประกอบด้วยหลายคอลัมน์และยาว อาจสร้างเป็นคอลัมน์เพื่อแทนคีย์หลักนั้นๆ โดยอาจใช้เป็นรหัสในการอ้างถึง

ข. พิจารณาโครงสร้างของฐานข้อมูลใหม่ อาจมีสาเหตุต่างๆ ดังนี้

- เนื่องจากมีความจำเป็นต้องปรับให้เข้ากับข้อจำกัดของดีบีเอ็มเอสที่นำมาใช้ เช่น การกำหนดชนิดข้อมูลให้คอลัมน์เป็นแบบ long หรือมีความยาวแบบไม่จำกัด (variable-length) กระทำไม่ได้ เป็นต้น ต้องหาวิธีแก้ไขโดยการสร้างตารางเพื่อรองรับขึ้นมา

- เพื่อให้ทำงานได้เร็วขึ้น อาจใช้วิธีต่างๆ เช่น การเพิ่มตาราง ได้แก่ ตารางที่สรุปมาจากตารางอื่น เป็นต้น การแบ่งตารางออกไป (segmenting) ตามแนวนอนหรือแนวตั้ง ซึ่งบางครั้งทำให้การใช้ประโยคดีเอ็มแอลซับซ้อนขึ้น อีกทั้งต้องระบุกฎเงื่อนไขต่างๆ เพิ่ม นอกจากนี้ยังมีวิธีการรวมตารางเข้าด้วยกัน อาจเป็นในลักษณะแนวนอน คือ รวมตารางที่มีคีย์หลักเหมือนกัน หรือแนวตั้ง คือรวมตารางแม่กับลูก ซึ่งการที่จะควบคุมความเป็นบูรณภาพของข้อมูลจะยุ่งยากขึ้น เพราะไม่ได้อยู่ในรูปแบบนอร์มัล และคอลัมน์อาจจะมีค่าว่างเกิดขึ้น

4. <u>การอนุญาตให้เข้าถึงข้อมูลโดยการผ่านวิว</u> นอกจากหัวข้อในการออกแบบดังที่กล่าวมาข้างต้น ยังต้องมีการพิจารณาในส่วนอื่นๆ อีก เกี่ยวกับการอนุญาตให้เข้าถึงข้อมูลโดยผ่านทางวิว วิวเป็นเสมือนหน้าต่างบนฐานข้อมูล ซึ่งแสดงในลักษณะตารางความสัมพันธ์ 2 มิติ (แถว/คอลัมน์) โดยเป็นผลมาจาก การเชื่อมโยง (join) หรือ การปฏิบัติการแบบรีเลชันนัล (relational operation) ที่มาจากตารางจริง (base table) ในฐานข้อมูล หรืออาจได้มาจากวิวอีกทีก็ได้ แต่ค่าของข้อมูล จะเก็บในตารางจริงเท่านั้น การให้ผู้ใช้ดึงข้อมูลผ่านวิวมีประโยชน์ดังนี้

- วิวสามารถแทนตารางให้กับผู้ใช้ในลักษณะใดก็ได้ โดยปราศจากผลกระทบของโครงสร้างข้อมูลจริง เช่น อาจทำให้ข้อมูลที่อยู่ในรูปแบบนอร์มัล ดูเหมือน ดีนอร์มัล เป็นต้น

- ทำให้การเรียกดูข้อมูลง่ายขึ้น วิวสามารถถูกกำหนดให้มาจากการเชื่อมตารางหลายตาราง ทำการคำนวณต่างๆ หรือเลือกเฉพาะคอลัมน์ หรือแถวที่ต้องการก็ได้

- สามารถใช้วิวในการกำหนดคำเหมือน (synonym) สำหรับตาราง รวมทั้งสำหรับคอลัมน์และข้อมูลที่เป็นดิไรฟ์ต่างๆ

- ทำให้สามารถจำกัดอำนาจในการเข้าถึงข้อมูล

แต่อย่างไรก็ตาม การสร้างวิวโดยไม่มีการจัดการที่ดี จะทำให้เกิดปัญหาต่างๆ ได้ เช่น เกิดวิวจำนวนมาก ยากแก่การดูแล ชื่อข้อมูลมีมาก อาจทำให้เกิดการสับสน และต้องมี

การสร้างโปรแกรมเพิ่มขึ้นในการจัดการกับวิวต่างๆ เป็นต้น

โดยทั่วไป จะใช้วิว และคำสั่งดีซีแอลเพื่อป้องกันในการเข้าถึงข้อมูล โดยที่วิวจะระบุแถวหรือคอลัมน์ที่เฉพาะตามที่ต้องการ และคำสั่ง GRANT/REVOKE จะระบุฟังก์ชันสำหรับผู้ใช้ในการใช้วิวหรือตาราง ถ้าต้องการจำกัด หรือให้อำนาจในการใช้ตารางทั้งตาราง เราอาจอนุญาตผ่านวิวหลัก (master view) หมายถึงวิวซึ่งรวมทุกคอลัมน์ทุกแถว จาก 1 ตาราง และถ้าต้องการควบคุมมากขึ้นที่ ระดับแถว หรือคอลัมน์ ก็อาจกำหนดวิวในระดับถัดมา คือ วิวในระดับที่ 2 เหนือวิวหลักขึ้นไป กล่าวคือเป็นวิวที่ถูกกำหนดมาจากวิวอีกทีหนึ่ง ซึ่งจะจำกัดการใช้งานแล้วแต่ความเหมาะสม และนอกจากนี้ ปกติความต้องการความปลอดภัยจะเป็นการจำกัดการเข้าถึงแถวของตารางโดยขึ้นกับรหัสผู้ใช้ (user ID) ตัวอย่างเช่น ในการดึงข้อมูลจากตารางพนักงาน โดยจำกัดตามแต่ละหน่วยงานให้เรียกใช้ได้เฉพาะข้อมูลในส่วนหน่วยงานของตนเอง ซึ่งวิธีการดำเนินการ โดยการกำหนดวิวต่างๆ สำหรับแต่ละหน่วยงานนั้น ถ้ามีหน่วยงานมาก วิวก็ย่อมมาก ทำให้ยุ่งยาก และการออกแบบโปรแกรมจะซับซ้อนขึ้นด้วย ดังนั้นทางออกก็คือใช้วิธีการกำหนดตารางระบุสิทธิ์การใช้งานของผู้ใช้ ซึ่งอาจประกอบด้วยรหัสผู้ใช้ กับค่าของคอลัมน์ สำหรับแถวที่ผู้ใช้นั้นมีสิทธิ์ในการทำงาน จากนั้นจึงกำหนดวิวซึ่งมาจากการเชื่อม (join) จากตารางระบุสิทธิ์การใช้งานของผู้ใช้ กับตาราง หรือวิวหลัก ที่ต้องการใช้ ดังนั้นจากการเชื่อมย่อมจะได้เพียงแถวที่ผู้ใช้คนนั้นมีสิทธิ์ในการเรียกใช้เท่านั้น