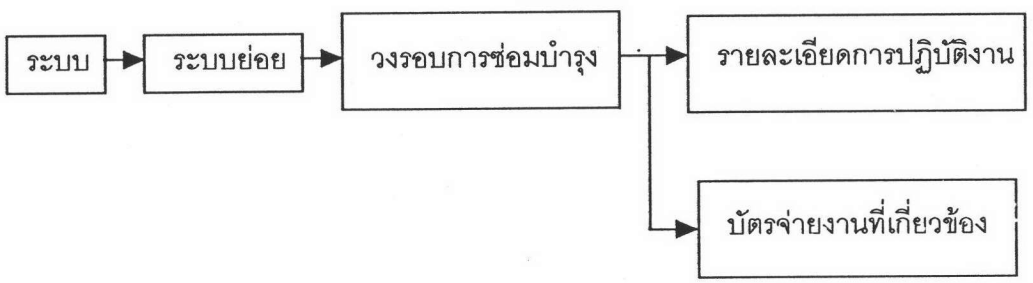
บทที่ 3

แนวคิดและทฤษฎี

ฐานข้อมูล (Database)

ฐานข้อมูล หมายถึง การเก็บรวบรวมกลุ่มของข้อมูลไว้ด้วยกัน เพื่อให้บุคคลและงานประยุกต์ต่างๆ สามารถดำเนินการอย่างใดอย่างหนึ่ง เช่น การเพิ่มข้อมูล ลบข้อมูล แก้ไขข้อมูล และการดึงข้อมูลมาใช้งาน เป็นต้น ฐานข้อมูลแบ่งตามลักษณะการใช้งานได้ 3 รูปแบบคือ (8)

1. ฐานข้อมูลแบบแตกสาขา(Hierarchical Database)(21) เป็นฐานข้อมูล ในรุ่นแรกๆ ปัจจุบันไม่นิยมใช้ เนื่องจากมีความยุ่งยากในการสร้างและใช้งานโดยหลักการข้อมูล จากสมาชิกระดับพ่อแม่(parent) ชี้ไปยังข้อมูลในระดับลูก(child) โดยตัวชี้(pointer) ของการเข้าถึงข้อมูล จะต้องผ่านสมาชิกระดับพ่อแม่(parent) ลงไปจนถึงสมาชิกระดับลูก(child) ผู้ออกแบบจะต้องกำหนดความเกี่ยวข้องระหว่างสมาชิกระดับพ่อแม่(parent)และสมาชิกระดับลูก(child) ตั้งแต่ขั้นตอนการสร้าง ข้อเสียของฐานข้อมูลชนิดนี้คือ ใช้งานยากและมีข้อจำกัดที่ว่าสมาชิกระดับลูก(child) จะเกี่ยวข้องกับสมาชิกระดับพ่อแม่(parent) ได้เพียง 1 ความสัมพันธ์เท่านั้น ดังรูปข้างล่าง

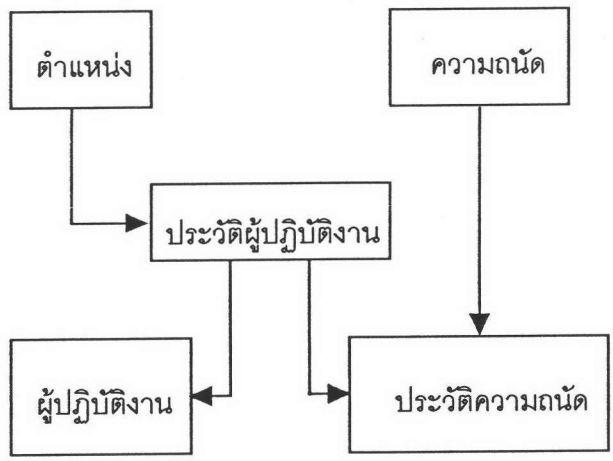


รูปที่ 3.1 แสดงโครงสร้างข้อมูลแบบแตกสาขา

จากรูป การที่จะทราบรายละเอียดการปฏิบัติงาน และบัตรจ่ายงานที่เกี่ยวข้องได้นั้น#จะต้องทราบระบบ ระบบว่อย วงรอบการซ่อมบำรุงก่อน จะเห็นได้ว่าการเข้าถึงสมาชิกระดับลูกจะต้องผ่านสมาชิกระดับแม่ก่อนและสมาชิกระดับลูก จะมีความสัมพันธ์กับสมาชิกระดับแม่ได้เพียงหนึ่งความสัมพันธ์เท่านั้น

2. ฐานข้อมูลแบบเครือข่าย (Network Database) ข้อเสียของฐานข้อมูลแบบ แตกสาขา ถูกแก้ด้วยระบบเครือข่ายคือ สมาชิกระดับลูก(child) สามารถเกี่ยวข้องกับสมาชิกระดับพ่อแม่ (parent)

ได้มากกว่า 1 แต่ก็ยังต้องกำหนดความสัมพันธ์ตั้งแต่เริ่มสร้างฐานข้อมูล และไม่สามารถเปลี่ยนแปลงในระหว่างการทำงานได้ ซึ่งแสดงได้ดังรูป



รูปที่ 3.2 แสดงโครงสร้างฐานข้อมูลแบบเครือข่าย

จากรูป จะเห็นว่าประวัติความถนัดซึ่งเป็นสมาชิกระดับลูก มีความสัมพันธ์กับสมาชิกระดับแม่ได้มากกว่าหนึ่งความสัมพันธ์ ได้แก่ ประวัติผู้ปฏิบัติงาน และรายละเอียดความถนัดโดยที่ผู้ปฏิบัติงานหนึ่งคนสามารถมีความถนัดได้มากกว่าหนึ่งรายการ และสามารถปฏิบัติงานได้มากกว่าหนึ่งงาน การค้นหาข้อมูลอาจวิ่งตามหัวลูกศรหรือทวนหัวลูกศรก็ได้ เช่น เมื่อต้องการทราบความถนัดของผู้ปฏิบัติงานแต่ละคน ก็ทำได้โดยการทราบรหัสและวิ่งตามลูกศรไปที่ประวัติความถนัดหรือต้องการทราบชื่อผู้ปฏิบัติงานอาจวิ่งทวนหัวลูกศรจากตาราง(table) ประวัติความถนัดไปยังตาราง(table) ประวัติผู้ปฏิบัติงานโดยต้องทราบรหัสผู้ปฏิบัติงานก่อนเป็นต้น

3. ระบบฐานข้อมูลแบบสัมพันธ์ (Relation Database)(9) เป็นฐานข้อมูลที่ได้รับการพัฒนาโดยที่ความสัมพันธ์ของข้อมูล จะถูกแทนในลักษณะของตาราง และแถวต่าง ๆ ในตารางจะแสดงค่าของข้อมูลที่มีความสัมพันธ์กัน ชื่อของตารางและชื่อของสดมภ์จะใช้ในการแปลความหมายของค่าในแต่ละแถวของตารางในฐานข้อมูล ดังรูปข้างล่างนี้

ประวัติผู้ปฏิบัติงาน

รหัสผู้ปฏิบัติงาน	ยศ	รหัสตำแหน่ง	ชื่อ-นามสกุล	ที่อยู่	วดป เกิด	รหัสหน่วยที่สังกัด	รหัสแผนก

ตำแหน่ง

รหัสตำแหน่ง	ชื่อตำแหน่ง

ความถนัด

รหัสความถนัด	ชื่อความถนัด

ประวัติความถนัด

รหัสผู้ปฏิบัติ	รหัสความถนัด

ผู้ปฏิบัติงาน

รหัสผู้ปฏิบัติ	รหัสงาน

รูปที่ 3.3 แสดงโครงสร้างฐานข้อมูลแบบสัมพันธ์

จากรูป จะเห็นว่าเราสามารถทราบงานของผู้ปฏิบัติงานได้ โดยการค้นหารหัสผู้ปฏิบัติงานที่ต้องการจากตาราง(table)ผู้ปฏิบัติงาน ในทางกลับกันถ้าต้องการทราบว่า รหัสงานนี้มีใครเป็นผู้ปฏิบัติบ้าง ก็เริ่มจากตารางผู้ปฏิบัติงานโดยค้นหารหัสงานที่ต้องการจาก ตาราง(table) ผู้ปฏิบัติงานซึ่งจะทำให้ได้รหัสผู้ปฏิบัติงาน ซึ่งจะทำให้ทราบว่าใครเป็นผู้ปฏิบัติงานนี้บ้าง

คุณสมบัติของระบบฐานข้อมูลแบบสัมพันธ์

1. รูปแปบฐานข้อมูลในระดับเชิงแนวคิด จะแทนเป็นตาราง(table) ที่สามารถเข้าใจได้ง่าย โดยในตารางจะแบ่งเป็นสดมภ์และเป็นแถว จุดตัดของแถวและสดมภ์จะเรียกว่า เขตข้อมูล(field) และภายในเขตข้อมูลจะเก็บค่า(value) ผู้ใช้สามารถเรียกหรือค้นข้อมูลที่ต้องการได้ทุก ๆ ค่า และหากเขตข้อมูลใดไม่มีค่าจะให้มีค่าว่าง(null) ไว้

2. ระบบจะต้องมีภาษาในระดับสูงเพื่อมาจัดการข้อมูลที่เก็บอยู่ในตารางโดยผู้ใช้งานเพียงระบุว่าต้องการข้อมูลอะไร ไม่ต้องบอกวิธีการประมวลผล ระบบจะจัดการเองภาษาในระดับสูงนี้อาจเรียกว่าภาษาในยุคที่ 4

3. ภาษาฐานข้อมูล เป็นภาษาที่มีการแบ่งประโยคคำสั่งแยกตามลักษณะงานได้ 3 กลุ่มดังนี้

3.1 ภาษานิยามข้อมูล (Data Definition Language:DDL) จะใช้เพื่อกำหนดโครงสร้างข้อมูล กำหนดตารางข้อมูลเทียบตามทัศนะของผู้ใช้ ได้แก่ คำสั่งในการสร้างตาราง เช่น คำสั่ง create table เป็นต้น คำสั่งในการสร้างวิว เช่น คำสั่ง create view เป็นต้น คำสั่งในการสร้างเลขลำดับ เช่น คำสั่ง create sequence เป็นต้น คำสั่งในการสร้างดัชนี เช่น คำสั่ง create index เป็นต้น คำสั่งในการสร้างคำเหมือน เช่น คำสั่ง create synonym เป็นต้น คำสั่งในการแก้ไขโครงสร้างของตาราง เช่น คำสั่ง alter table เป็นต้น และคำสั่งในการยกเลิก ตาราง วิว หรือดัชนี เช่น คำสั่ง drop index/ table/view เป็นต้น

3.2 ภาษาจัดการข้อมูล (Data Munipulation Language:DML) จะใช้เพื่อการดัดแปลงแก้ไขและการเรียกค้นข้อมูล ได้แก่ คำสั่งในการเพิ่มระเบียน(record) ใหม่ของข้อมูลในตาราง เช่น คำสั่ง insert เป็นต้น คำสั่งในการแก้ไขค่าในสดมภ์(column) เช่น คำสั่ง update เป็นต้น คำสั่งในการลบระเบียน(record) ของข้อมูลในตาราง เช่น คำสั่ง delete เป็นต้น

3.3 ภาษาควบคุมข้อมูล(Data Control Language:DCL) ใช้สำหรับควบคุมการเข้าถึงข้อมูลและฐานข้อมูล ตัวอย่างเช่น คำสั่งควบคุมการเข้าถึงฐานข้อมูล ได้แก่ คำสั่ง grant และคำสั่งยกเลิกการเข้าถึงฐานข้อมูลได้แก่ คำสั่ง revoke เป็นต้น

4. การเชื่อมต่อระหว่างตารางต่าง ๆ ภายในฐานข้อมูล ผู้ใช้จะไม่มีโอกาสได้รู้เห็นเพราะ ระบบจะปฏิบัติการให้อัตโนมัติ โดยระบบจะเลือกกลยุทธ์ที่ดีและได้ผลที่สุด

5. การประมวลผลฐานข้อมูลแบบสัมพันธ์ จะเป็นแบบกลุ่มของข้อมูล (set oriented) กล่าวคือ เมื่อมีการถามหรือเรียกค้น จะได้คำตอบเป็นชุด หรือเป็นเซตหลาย ๆ ระเบียนกลับมาซึ่งต่างกับการจัดการฐานข้อมูล (DBMS) สมัยก่อนจะให้คำตอบครั้งละระเบียน แล้วมีการวนรอบเรียกระเบียนต่อไป ในฐานข้อมูลแบบสัมพันธ์จะไม่มีการวนรอบ แต่จะมีการใช้งานผ่านตัวกระทำสัมพันธ์พื้นฐาน เช่น SELECT,PROJECT,JOIN และ DIVIDE กับตัวกระทำทางเซต เช่น การผนวก(UNION) การตัดกัน (INTERSECT) การทำผลต่าง(DIFFERENCE) และการทำผลคูณ (PRODUCT)

6. ข้อมูลเกี่ยวกับระบบตั้งแต่โครงสร้างฐานข้อมูล สิทธิของการใช้ ฯลฯ จะต้องเก็บบันทึกในตารางของระบบ เมื่อฐานข้อมูลมีการเปลี่ยนแปลงโครงสร้าง ตารางของระบบนี้จะมีการเปลี่ยนแปลงแก้ไขโดยอัตโนมัติด้วย ซึ่งเรียกตารางนี้ว่า พจนานุกรมข้อมูล (data dictionary) การเรียกค้นหรือการขอดูข้อมูลในพจนานุกรมข้อมูล จะต้องสามารถสั่งการด้วยภาษาฐานข้อมูลชุดเดิมได้

7. ฐานข้อมูลแบบสัมพันธ์และโครงสร้าง สามารถทำการเปลี่ยนแปลง แก้ไขได้โดยอาจเปลี่ยนแปลงที่เค้าร่างย่อยของส่วนการมองของผู้ใช้หรือเปลี่ยนค่าพร้อมกันในหลายแถวหรือหลายสดมภ์

8. การเปลี่ยนแปลงระบบฐานข้อมูลในระดับต่ำ จะไม่มีผลกระทบต่อส่วนที่อยู่ในระดับสูงกว่าหรือชุดคำสั่งที่ใช้งาน ไม่ว่าจะเป็นการเปลี่ยนแปลงในระดับภายใน หรือในระดับเชิงแนวคิด

9. ระบบฐานข้อมูลจะต้องเป็นอิสระ ไม่ขึ้นกับการจัดเก็บตามข้อกำหนดความถูกต้องของข้อมูล ไม่ขึ้นกับการควบคุมของภาษาระดับต่ำ เพื่อมาทำลายความถูกต้องของระบบและไม่ขึ้นกับการกระจายของข้อมูลที่อาจจะแยกกันอยู่หลาย ๆ แห่ง

จากคุณสมบัติที่ดีเหล่านี้งานวิจัยนี้ จึงเลือกใช้ระบบฐานข้อมูลแบบสัมพันธ์ ในการสร้างฐานข้อมูลเพื่อลดความซ้ำซ้อนของข้อมูล และสามารถหลีกเลี่ยงความขัดแย้งของข้อมูล และสามารถปรับปรุง และพัฒนาระบบฐานข้อมูลได้ต่อไปในอนาคต โดยไม่ส่งผลกระทบต่อคำสั่งที่ใช้งาน อีกทั้งยังมีระบบจัดการด้านความปลอดภัยของข้อมูล เพื่อให้ผู้ใช้สามารถดำเนินการกับข้อมูลได้ตามสิทธิของผู้ใช้แต่ละคน

จุดประสงค์ของวิธีการฐานข้อมูล (21)

1. ป้องกันคุณค่าของข้อมูล เพราะมีการควบคุมข้อมูลให้มีบูรณภาพ

2. มีการประมวลผลข้อมูลให้ตอบสนองต่อ ความต้องการสารสนเทศใหม่ ๆ ที่องค์กรต้องการ การจัดการฐานข้อมูลพางระบบ ได้รวมเทคนิคของโครงสร้างข้อมูล(Data Structure) ที่ยืดหยุ่นเพื่อสามารถทำให้จัดการกับสิ่งที่เปลี่ยนแปลงได้อย่างสะดวก ดังนั้น ผู้ออกแบบจะมีอิสระจากความจำเป็นที่จะต้องรู้รายละเอียดของโครงสร้างต่าง ๆ หรือสื่อที่ใช้เก็บข้อมูล วิธีการฐานข้อมูลใช้เทคนิคของแบบจำลองข้อมูลเชิงตรรกที่อนุญาตให้แหล่งข้อมูล ถูกกำหนดเป็นอิสระจากการติดตั้ง(implement) จริง ๆ ของแหล่งข้อมูลนั้น วิธีการของฐานข้อมูลจะเน้นถึงการจัดการความหมายของข้อมูลมากกว่า เพื่อให้องค์กรมีระบบการจัดการ การตัดสินใจที่ดีขึ้น และเพื่อช่วยลดค่าใช้จ่ายที่เกี่ยวกับการปรับปรุงประสิทธิภาพและการเปลี่ยนแปลงของการพัฒนาเทคโนโลยีทางคอมพิวเตอร์ ระบบจัดการฐานข้อมูลสามารถจัดการให้ผู้ใช้ได้รับอิสระจากรายละเอียดของสื่อในการจัดเก็บข้อมูลซึ่งสามารถถูกเปลี่ยนแปลงให้มีประสิทธิภาพเพิ่มขึ้นได้โดยปราศจากการเปลี่ยนแปลงแหล่งข้อมูลหรือวิธีการที่ใช้เข้าถึงข้อมูลนั้น ๆ

ประโยชน์ของฐานข้อมูล (9)

1. ความมีอยู่ของข้อมูล (Data Availability) นั่นคือ เมื่อใดที่ความต้องการของข้อมูลเกิดขึ้นหรือเมื่อผู้ใช้คนใดทำการปรับปรุงข้อมูลผู้ใช้คนอื่นจะต้องได้ข้อมูลที่ได้รับการปรับปรุงแล้ว และสามารถเรียกใช้ข้อมูลได้ตลอดเวลา

2. ลดความซ้ำซ้อนของข้อมูล(Data Redundancy) โดยข้อมูลตัวเดียวกันจะถูกเก็บอยู่เพียงที่เดียวเท่านั้น

3. ความสอดคล้องของข้อมูล (Data Consistency) ค่าของข้อมูลนั้นไม่ว่าจะเรียกใช้โดยวิธีใดๆ ก็ตามจะต้องให้ค่าเดิมเสมอเนื่องจากข้อมูลจะถูกเก็บอยู่ที่เดียวกัน ดังนั้นไม่ว่าจะทำการแก้ไข ปรับปรุงข้อมูล หรือสืบค้นข้อมูล ค่าของข้อมูลก็จะเป็นข้อมูลเดียวกัน

4. ความเชื่อถือได้ของข้อมูล (Data Reliability)

5. ข้อมูลมีความเป็นปัจจุบัน (Data Currency) หากเมื่อผู้ใช้คนหนึ่งทำการปรับปรุงข้อมูลผู้ใช้คนอื่นจะต้องได้ข้อมูลที่ได้รับการปรับปรุงแล้วเสมอ

6. ความยืดหยุ่นของข้อมูล (Data Flexibility) เราสามารถเปลี่ยนโครงสร้างฐานข้อมูลตามความต้องการที่เปลี่ยนแปลงไป โดยไม่กระทบกับระบบงานเดิม

7. ลดค่าใช้จ่าย (Low Cost) เนื่องจากฐานข้อมูลสามารถลดความซ้ำซ้อนของข้อมูลทำให้สามารถลดค่าใช้จ่ายด้านการทำงาน และการบริหารลงได้

8. เพิ่มประสิทธิภาพของข้อมูล (Data Efficiency)

แบบจำลองข้อมูล (Data Model)

แบบจำลองข้อมูล เป็นแนวความคิดที่รวบรวมโครงสร้างฐานข้อมูล ซึ่งได้แก่ ชนิดของข้อมูล ความสัมพันธ์แบบต่างๆ ระหว่างข้อมูล รวมทั้งปฏิบัติการต่าง ๆ ในการจัดการฐานข้อมูล ทั้งการค้นหา แก้ไข ปรับปรุงข้อมูลในฐานข้อมูลด้วย ซึ่งแบบจำลองข้อมูลถูกจำแนกตามชนิดของแนวความคิดที่ใช้อธิบายโครงสร้างฐานข้อมูล ได้ดังนี้

1. แบบจำลองข้อมูลเชิงมโนภาพ(Conceptual data model) หรือแบบจำลองข้อมูลระดับสูง (High level data model) เป็นการแสดงให้เห็นถึงโครงร่างของข้อมูลในส่วนมุมมองของผู้ใช้กลุ่มต่างๆ (Skeletal user view) โดยไม่คำนึงถึงรายละเอียดในการติดตั้ง (Implement)

2. แบบจำลองข้อมูลเชิงกายภาพ (Physical data model) หรือแบบจำลองข้อมูลระดับต่ำ (Low level data model) ใช้อธิบายรายละเอียดว่า ข้อมูลถูกเก็บลงคอมพิวเตอร์อย่างไร ซึ่งจะสามารถเข้าใจได้

โดยผู้ชำนาญงานคอมพิวเตอร์ ไม่ใช่สำหรับผู้ใช้ทั่วไปโดยจะแสดงถึงรูปแบบของข้อมูล การเรียงลำดับข้อมูลและวิถีทางเข้าถึงข้อมูล (Access path) ได้อย่างมีประสิทธิภาพ

การออกแบบระบบฐานข้อมูล (11) มีขั้นตอนดังนี้

1. การวิเคราะห์ความต้องการ (Requirements Analysis) เป็นการวิเคราะห์ความต้องการของผู้ใช้ในทุก ๆ ด้าน เช่น ความต้องการด้านสารสนเทศ ความต้องการด้านการประมวลผล ความต้องการการรายงานผล เป็นต้น การทำงานขั้นตอนนี้ประกอบด้วย การวางแผนกลยุทธ์ แบบจำลองทางธุรกิจ แบบจำลองกระแสข้อมูล ซึ่งได้ข้อมูลมาจาก ผู้ใช้โดยตรงหรือจากเอกสารที่เกี่ยวข้อง

2. การออกแบบโครงสร้างเชิงแนวความคิด (Design of Conceptual Structure) คือ การนำผลการวิเคราะห์ความต้องการ ที่ได้จากขั้นแรกมาสร้างความสัมพันธ์ของข้อมูล โดยอาศัยการสร้างแบบจำลองข้อมูลเชิงตรรก (Logical Data Model) ในการออกแบบฐานข้อมูล สิ่งที่เกิดขึ้นในขั้นตอนนี้ คือ แบบจำลองเอนทิตี แบบจำลองความสัมพันธ์ระหว่างเอนทิตี และภาพรวมเบ็ดเสร็จของระบบ

แบบจำลองข้อมูลเชิงตรรก ใช้แทนโครงสร้างข้อมูล สำหรับการสร้างฐานข้อมูลเชิงสัมพันธ์เพื่อให้ผู้ใช้สามารถใช้ข้อมูลต่าง ๆ ร่วมกันได้ และสามารถปรับปรุงแก้ไขข้อมูลได้ง่าย

3. การออกแบบเค้าร่างเชิงตรรกภาพ#(Design of logical Schema) เป็นขั้นตอนการเปลี่ยนแปลงแผนผัง เอนทิตีที่ได้เป็นเค้าร่างความสัมพันธ์ และดำเนินการปรุงแต่งเค้าร่างความสัมพันธ์ให้ดีขึ้นด้วยการทำให้เป็นบรรทัดฐาน ซึ่งเป็นวิธีการที่จะทำให้ได้แบบจำลองข้อมูลที่เหมาะสม คือ ทำให้ข้อมูลที่ได้นั้นมีความถูกต้อง ไม่มีความขัดแย้งกัน ไม่มีความซ้ำซ้อนของข้อมูล ประหยัดเนื้อที่ที่ใช้เก็บข้อมูล ดังนั้นเอนทิตีหลักที่ได้สามารถนำมาทำให้เป็นบรรทัดฐานโดยการสร้างเอนทิตีใหม่ และสามารถแสดงความสัมพันธ์ระหว่างเอนทิตีต่าง ๆ ได้

4. ออกแบบเค้าร่างเชิงกายภาพ (Design of Physical Schema) เป็นขั้นตอนการจัดทำแผนผังการเข้าถึงกลยุทธ์การกำหนดตำแหน่ง การทำดีชนี การศึกษาหน่วยเก็บข้อมูลที่เหมาะสม เพื่อนำไปใช้ในขั้นตอน การติดตั้งระบบงานต่อไป

การออกแบบฐานข้อมูลแบบความสัมพันธ์ (Relational Database Design) (10)

จากการวางแผนและวิเคราะห์ เมื่อได้แบบจำลองข้อมูลซึ่งจะได้แผนภาพแสดงความสัมพันธ์

ระหว่างเอนทิตีแล้ว ขั้นต่อมาเป็นการออกแบบฐานข้อมูลแบบความสัมพันธ์ กล่าวคือเป็นกระบวนการเปลี่ยนแบบจำลองข้อมูลเชิงตรรกให้เป็นฐานข้อมูลแบบความสัมพันธ์ ซึ่งมีขั้นตอนดังนี้

1. แปลงโครงร่างแบบตรรกภาพ ให้เป็นฐานข้อมูลแบบความสัมพันธ์เบื้องต้น (Translate the Logical Data Structure) จะพิจารณาความเป็นไปได้และองค์ประกอบต่าง ๆ ว่าครบถ้วนหรือไม่ซึ่งมีขั้นตอนดังนี้

1.1 กำหนดตาราง(identify table) สำหรับแต่ละเอนทิตี

1.2 กำหนดสดมภ์(identify column) ในแต่ละตารางสำหรับแต่ละเอนทิตีจะมีแอตตริบิวใช้สำหรับกำหนดคุณสมบัติของสดมภ์

จะเห็นได้ว่า ในเบื้องต้นนี้แต่ละตารางจะมีสดมภ์ ซึ่งประกอบด้วยกุญแจหลัก (primary key) กุญแจรอง(secondary key) กุญแจนอก(foreign key) รวมทั้งสดมภ์(column)ที่ไม่ใช่กุญแจ(key) ซึ่งการกำหนดสดมภ์(column)ของกุญแจนอก(foreign key) นั้นมาจากการแปลงความสัมพันธ์ที่ได้จากแบบจำลองข้อมูลเชิงตรรกภาพนั่นเอง

1.3 ปรับโครงสร้างข้อมูลให้เข้ากับสภาพแวดล้อมที่นำมาใช้ ทุก ๆ ระบบฐานข้อมูลแบบความสัมพันธ์ (RDBMS) จะมีการให้คำจำกัดความของตารางผ่านทางภาษา สำหรับนิยามข้อมูล (DDL:Data Definition Language) ซึ่งไวยากรณ์(Syntax) อาจต่างกันในแต่ละระบบ เราจะต้องปรับให้เข้ากับระบบที่เราต้องการ เช่น พิจารณาการสร้างตารางเก็บระบบฐานข้อมูล การเรียงลำดับสดมภ์ในตาราง เป็นต้น ในการกำหนดตารางให้กับระบบจัดการฐานข้อมูล(DBMS) ยังต้องพิจารณาถึงพารามิเตอร์ที่มีผลกระทบกับการจัดสรร(allocate) และจัดการของหน่วยเก็บรวมทั้งเนื้อที่ที่ว่างด้วย ซึ่งเนื้อที่ต้องมากพอ

2. การแปลความเป็นบูรณภาพของข้อมูลเชิงตรรก(Translate the Logical Data Integrity) การแปลในที่นี้คือ การบังคับให้เป็นไปตามกฎธุรกิจต่างๆ ซึ่งบางครั้งระบบจัดการข้อมูลที่นำมาใช้ อาจจะไม่สนับสนุนการทำงานบางอย่างก็อาจจะใช้กลไกอื่นๆ ช่วยสำหรับขั้นตอนต่าง ๆ ในขั้นนี้ได้แก่

2.1 บังคับคุณสมบัติเชิงตรรกของกุญแจหลักของเอนทิตี (Design for Business Rules about Entities) ในการนำมาใช้งาน เช่น ความเป็นเอกลักษณ์ การไม่ยอมให้มีค่าว่าง เป็นต้น โดยอาจกำหนดคุณสมบัติเหล่านี้ โดยใช้ภาษาสำหรับนิยามข้อมูล(DDL) หรือกำหนดโดยผ่านเทคนิคการใช้ข้อกำหนดของโดเมน

2.2 บังคับคุณสมบัติเชิงตรรกของความสัมพันธ์ (Design for Business Rules about Relationship) เกี่ยวกับข้อบังคับการเพิ่ม ลบ แก้ไขข้อมูล โดยอาจกำหนดโดยใช้ภาษาสำหรับนิยามข้อมูล

(DDL) หรือใช้เทคนิคการทำทริกเกอร์ (Trigger) มิฉะนั้นอาจเป็นหน้าที่ หรือความรับผิดชอบของผู้ใช้ที่จะต้องปฏิบัติตามข้อบังคับ

2.3 การออกแบบกฎธุรกิจของแอตตริบิว (Design for addition Business Rules about Attributes) อาจทำได้โดยการใช้ทริกเกอร์ ภาษานิยามข้อมูล(DDL) หรือใช้ข้อกำหนดของโดเมน และโดยปกติไม่ควรยอมให้สดมภ์บางสดมภ์เป็นค่าว่าง ได้แก่ สดมภ์ที่เป็นกุญแจหลัก กุญแจนอก(foreign key) ซึ่งมักถูกใช้ในการเปรียบเทียบและเชื่อมโยง(join) กับสดมภ์ที่ใช้ในเอสคิวแอล (SQL:Structure Query Language) ที่มีการใช้ where และสดมภ์ที่เป็นตัวเลขเพราะจะเป็นปัญหาในการคำนวณ

3. การปรับแต่งการออกแบบ(Tuning) เราสามารถใช้วิธีหลายอย่างเพื่อช่วยในการปรับ เพื่อเพิ่มประสิทธิภาพในการทำงาน เช่น

3.1 การวิเคราะห์กลไกการเข้าถึง (Tuning Access Mechanism) โดยหาทางเลือกที่ให้ประโยชน์สูงสุด

3.2 การจัดรวบรวมข้อมูลให้เป็นหมู่ หรือกลุ่มเดียวกันไว้ด้วยกัน (Define Clustering Sequence)

3.3 การสร้างดัชนีเพิ่ม (Add Indexes) ซึ่งข้อแนะนำคือควรสร้างดัชนีในตารางที่มีขนาดกลางจนถึงขนาดใหญ่ หรือถ้าในการเรียกใช้หรือดึงข้อมูลในตารางนั้น ๆ แล้วได้ผลลัพธ์ประมาณ 20% ของจำนวนแถวและในการประเมินประสิทธิภาพของดัชนีนั้น จะต้องพิจารณาผลกระทบในการเพิ่ม ลบ แก้ไข ตลอดจนการประมวลผลต่าง ๆ และยังต้องพิจารณาค่าใช้จ่ายเนื่องจากความต้องการหน่วยเก็บอีกด้วย

3.4 การยอมให้ข้อมูลมีการซ้ำซ้อนโดยมีการควบคุม (Tune by introducting Redundancy) สิ่งสำคัญที่ต้องพิจารณาการยอมให้มีข้อมูลซ้ำ คืออาจมีเหตุผลที่ทำให้เอนทิตีหนึ่งๆ ที่วิเคราะห์ออกมาอาจจะไม่กลายเป็นตาราง 1 ตารางในฐานข้อมูลเสมอไป ซึ่งมีเหตุผล 2 ประการโดยทั่วไปก็คือ

3.4.1 เอนทิตีที่ถูกกำหนดขึ้นมา ไม่ได้เกี่ยวพันกับการใช้งานด้วยคอมพิวเตอร์ ขณะที่กำลังทำอยู่ซึ่งเราอาจจะตัดสินใจหรือไม่ก็ได้ ว่าจะยกเอนทิตีนี้ไปเป็นตารางต่อไป ถ้าหากมีการขยายการใช้งาน

3.4.2 อาจต้องมีการรวมเอนทิตี 2 เอนทิตีหรือมากกว่า เป็นเพียง 1 ตารางเพื่อเพิ่มประสิทธิภาพในการทำงาน เช่น การที่กำหนดเอนทิตีเป็นสิ่งสำคัญที่มองแยกออกไป แต่ใน

โครงสร้างของตารางจะรวมกันเข้า เพื่อเลี่ยงในการเสียเวลาในการเชื่อมข้อมูลต่อกัน และบางครั้งการทำให้อยู่ในรูปนอร์มัลอาจทำให้เกิดความซ้ำซ้อนเกินจำเป็น ดังนั้นอาจต้องมีการเปลี่ยนแปลงนั่นคือมีการทำให้อยู่ในรูปดีนอร์มัล(Denormalize) ซึ่งจะก่อให้เกิดความซับซ้อนของข้อมูลในฐานข้อมูลขึ้นใหม่ แต่ก่อนที่จะพิจารณาการดีนอร์มัล จะต้องผ่านกระบวนการทำนอร์มัลในระดับที่ 3 ก่อนและจะต้องตัดสินใจมากขึ้น เพื่อให้แน่ใจว่าข้อมูลที่ซ้ำกันไม่ได้ลักหลั่นกันซึ่งถ้าทำให้อยู่ในรูปดีนอร์มัลแล้ว อาจทำให้การพัฒนางานง่ายขึ้น และทำให้เพิ่มประสิทธิภาพการเรียกดู หรือค้นหาข้อมูลไม่ต้องสืบค้นหลายตารางหรือไม่เสียเวลาในการคำนวณ เป็นต้น แต่อย่างไรก็ตามต้องพิจารณาผลดีผลเสียที่เกิดขึ้นซึ่งมีผลกระทบกับหน่วยความจำ การดึงข้อมูลด้วยประโยคจัดการข้อมูล (DML:Data Manipulate Language) ที่ใช้ความเป็นบูรณภาพของข้อมูลตลอดจนประสิทธิภาพในการใช้งาน ดังนั้นสดมภ์ที่ซ้ำซ้อนอาจได้มาจากสาเหตุต่างๆ ดังนี้

3.4.2.1 สดมภ์ที่ลอกมาจากสดมภ์ที่มีอยู่แล้วในตารางอื่น เช่น ถ้ามีตารางเก็บประวัติการทำงาน แต่เราอาจเก็บการทำงานปัจจุบันในตารางพนักงานอีกด้วย ทำให้ข้อมูลปรากฏ 2 แห่ง เป็นต้น

3.4.2.2 สดมภ์ที่เป็นผลสรุปมาจากสดมภ์อื่น (derived column) ซึ่งมีความสำคัญต่อผู้ใช้ทำให้สามารถตอบคำถามในการสอบถามได้เร็วขึ้น แต่การยอมให้ผู้ใช้เปลี่ยนแปลงค่า ควรให้กระทำที่ต้นกำเนิด หรือที่มาของข้อมูลเท่านั้น

3.4.2.3 สดมภ์ที่เป็นกลุ่มซ้ำ เมื่อกลุ่มซ้ำนั้นมีจำนวนแน่นอนและกลุ่มซ้ำนั้นมักถูกเรียกใช้หรือแก้ไขพร้อมกัน เช่น ถ้าทราบว่าพนักงานแต่ละคนมีทักษะ 2 อย่าง เสมอเราอาจเก็บข้อมูลเป็นสองสดมภ์ในตาราง แต่ปัญหาก็คือถ้าภายหลังมีการเพิ่มทักษะประเภทที่ 3 จะจัดการอย่างไร ดังนั้นต้องพิจารณาผลดี ผลเสีย ในการจำกัดความยืดหยุ่น

3.4.2.4 สดมภ์อย่างย่อที่สร้างขึ้นมาเพื่อแทนสดมภ์ที่มีอยู่ เช่น ในกรณีที่กุญแจหลักประกอบด้วยหลายสดมภ์และยาว อาจสร้างเป็นสดมภ์เพื่อแทนกุญแจหลักนั้น ๆ โดยอาจใช้เป็นรหัสในการอ้างถึง ดังตาราง

วงรอบการซ่อมบำรุง

รหัสระบบ	รหัสระบบย่อย	รหัสส่วนของระบบย่อย	จำนวน/วงรอบปกติ	ประเภทของวงรอบ	จำนวนงาน
คจญ	001	00	1	ด	2
คจญ	001	01	1	ว	1
คจญ	001	05	6	ด	2
ยพน	405	02	1	ป	1
ยพน	405	03	2	ป	3

ตารางที่ 3.1 แสดงตัวอย่างตารางที่อยู่ในรูปแบบนอร์มัล(normal form)

จากตาราง จะเห็นว่าตารางที่ได้อยู่ในรูปของนอร์มัลแล้ว ซึ่งกุญแจหลักของตารางนี้ประกอบด้วย รหัสระบบ รหัสระบบย่อย รหัสส่วนของระบบย่อย จำนวน/วงรอบปกติ และประเภทของรอบ ซึ่งทำให้การใช้งานไม่สะดวกในการอ้างถึงกุญแจหลัก ดังนั้น เพื่อให้การทำงานสะดวกขึ้น จึงอาจทำให้อยู่ในรูปดีนอร์มัลได้ดังตารางข้างล่าง โดยการเพิ่มสดมภ์(column) รหัส ซึ่งประกอบด้วยสดมภ์ต่าง ๆ ที่นำมารวมกันเพื่อใช้เป็นกุญแจในการอ้างอิงได้สะดวกรวดเร็วขึ้นแม้ว่าข้อมูลจะซ้ำซ้อนกันก็ตาม

วงรอบการซ่อมบำรุง

รหัสระบบ	รหัสระบบย่อย	รหัสส่วนของระบบย่อย	รหัส	จำนวน/วงรอบปกติ	ประเภทของวงรอบ	จำนวนงาน
คจญ	001	00	คจญ001001ด	1	ด	2
คจญ	001	01	คจญ001011ว	1	ว	1
คจญ	001	05	คจญ001056ด	6	ด	2
ยพน	405	02	ยพน405021ป	1	ป	1
ยพน	405	03	ยพน405032ป	2	ป	3

ตารางที่ 3.2 แสดงตัวอย่างตารางที่อยู่ในรูปดีนอร์มัล(denormalize)

3.5 พิจารณาโครงสร้างของฐานข้อมูลใหม่ (Tune by Redefining the Relation database Structure) มีวิธีดำเนินการอยู่ 2 ลักษณะคือ

3.5.1 การกำหนดสดมภ์ใหม่ (Redefine Columns) เนื่องจากมีความจำเป็นต้องปรับโครงสร้างให้เข้ากับข้อจำกัดของระบบการจัดการฐานข้อมูล(DBMS) ที่นำมาใช้งาน

เช่น การกำหนดชนิดข้อมูลให้สดมภ์เป็นแบบความยาวไม่จำกัด (variable length) กระทำไม่ได้ เป็นต้น จึงหาวิธีแก้ไขโดยการสร้างตารางเพื่อรองรับขึ้นมา

3.5.2 การกำหนดตารางใหม่ (Redefine Tables) เพื่อให้การทำงานเร็วขึ้นซึ่งมีวิธีการต่าง ๆ เช่น การเพิ่มตาราง ได้แก่ ตารางที่สรุปมาจากตารางอื่น เป็นต้น การแบ่งตารางออกไป (Segmenting) ตามแนวนอนหรือแนวตั้ง ซึ่งบางครั้งทำให้ประโยคจัดการข้อมูล(DML)ซับซ้อนขึ้น อีกทั้งต้องระบุกฎเงื่อนไขต่าง ๆ เพิ่ม นอกจากนี้ ยังมีวิธีการรวมตารางเข้าด้วยกัน อาจเป็นลักษณะในแนวนอน หรือแนวตั้ง คือรวมตารางแม่กับลูก ซึ่งการควบคุมความเป็นบูรณภาพของข้อมูลยุ่งยากขึ้น เพราะไม่ได้อยู่ในรูปแบบนอร์มัล และสดมภ์ อาจจะมีค่าว่างเกิดขึ้น

การอนุญาตให้เข้าถึงข้อมูลโดยการผ่านวิว

วิวเป็นเสมือนหน้าต่าง บนฐานข้อมูลซึ่งแสดงลักษณะตาราง 2 มิติ (แถว/สดมภ์)โดยเป็นผลมาจากการเชื่อมโยง(join) หรือการปฏิบัติการแบบเชิงสัมพันธ์(relational operation) ที่มาจากตารางพื้นฐาน(base table)ในฐานข้อมูล หรืออาจได้มาจากวิวอีกทีก็ได้ แต่ค่าของข้อมูลจะเก็บในตารางจริงเท่านั้น การให้ผู้ใช้ดึงข้อมูลโดยใช้วิวมีประโยชน์ดังนี้

1. วิวสามารถแทนตาราง ให้กับผู้ใช้ในลักษณะใดก็ได้ โดยไม่มีผลกระทบต่อโครงสร้างข้อมูลจริง เช่น อาจทำให้ข้อมูลอยู่ในรูปแบบนอร์มัลดูเหมือนดีนอร์มัล เป็นต้น

2. ทำให้การเรียกดูข้อมูลง่ายขึ้น วิวสามารถถูกกำหนดให้มาจากการเชื่อมตารางหลายตาราง ทำการคำนวณต่าง ๆ หรือเลือกเฉพาะสดมภ์ หรือแถวที่ต้องการก็ได้

3. สามารถใช้วิวในการกำหนดคำเหมือน (synonym) สำหรับตารางรวมทั้งสำหรับสดมภ์และข้อมูลที่คำนวณได้ต่าง ๆ

4. ทำให้สามารถจำกัดอำนาจในการเข้าถึงข้อมูล

แต่อย่างไรก็ตามการสร้างวิว โดยไม่มีการจัดการที่ดี จะทำให้เกิดปัญหาต่าง ๆ ได้ เช่น เกิดวิวจำนวนมาก ยากแก่การดูแล ชื่อข้อมูลมีมาก อาจทำให้เกิดการสับสนและต้องมีการสร้างโปรแกรมเพิ่มขึ้นเพื่อจัดการกับวิวต่าง ๆ เป็นต้น

โดยทั่วไป จะใช้วิวและคำสั่งควบคุมข้อมูล(DCL)เพื่อป้องกันในการเข้าถึงข้อมูลโดยที่วิวจะระบุแถวหรือสดมภ์ตามเงื่อนไขที่ต้องการและคำสั่ง GRANT/REVOKE จะระบุฟังก์ชัน สำหรับผู้ใช้ในการใช้วิวหรือตาราง ถ้าต้องการจำกัด หรือให้อำนาจในการใช้ตารางทั้งตาราง เราอาจอนุญาต

ผ่านวิวหลัก (master view) หมายถึง วิวซึ่งรวมทุกสดมภ์ทุกแถวจาก 1 ตาราง และถ้าต้องการควบคุมมากขึ้นที่ระดับแถว หรือสดมภ์ ก็อาจกำหนดวิวในระดับถัดมา คือ วิวในระดับที่ 2 เหนือวิวหลักขึ้นไป กล่าวคือเป็นวิวที่ถูกกำหนดมาจากวิวอีกทีหนึ่ง ซึ่งจะจำกัดการใช้งานแล้วแต่ความเหมาะสม และโดยทั่วไปความต้องการความปลอดภัยจะเป็นการจำกัดการเข้าถึงแถวของตารางโดยขึ้นกับรหัสผู้ใช้(User ID) ตัวอย่างเช่น ผู้ปฏิบัติงานแต่ละคนสามารถเข้าถึงข้อมูลประวัติผู้ปฏิบัติงาน ได้เฉพาะข้อมูลของตนเองเท่านั้น ไม่สามารถดูข้อมูลของคนอื่นได้ ดังเช่น ข้อมูลประวัติผู้ปฏิบัติงานจากตารางพื้นฐาน ซึ่งจะประกอบไปด้วยรายละเอียดต่าง ๆ ได้แก่ รหัส ยศ ตำแหน่ง ชื่อ-นามสกุล ที่อยู่ วัน-เดือน-ปี เกิด หน่วยและแผนกที่สังกัด สามารถแสดงได้ ดังตารางข้างล่าง

ประวัติผู้ปฏิบัติงาน (ข้อมูลจากตารางพื้นฐาน(base table))

รหัสผู้ปฏิบัติงาน	ยศ	รหัสตำแหน่ง	ชื่อ-นามสกุล	ที่อยู่	วดป.เกิด	รหัสหน่วยที่สังกัด	รหัสแผนก
001	พัน	02	สมชาย รักษ์	105	01/02/2500	! 03	02
002	จ่า	05	สมบูรณ์ ดำดี	105	15/11/2501	06	05
003	พล	04	สมศักดิ์ สุโข	105	01/12/2488	06	05

ตารางที่ 3.3 แสดงตัวอย่างข้อมูลจากตารางพื้นฐาน

จากตาราง จะเห็นว่าข้อมูลจากตารางประวัติผู้ปฏิบัติงาน จะมีข้อมูลต่าง ๆ ของทุกคนและเมื่อมีการเข้าถึงข้อมูล โดยการผ่านวิวก็จะได้เฉพาะข้อมูลตามสิทธิ์ของตนเองเท่านั้น เช่น การเข้าถึงข้อมูลโดยนายสมศักดิ์ สุโข จะได้ข้อมูลเฉพาะของนายสมศักดิ์ สุโข เท่านั้น เนื่องจาก นายสมศักดิ์ ได้รับสิทธิ์การเข้าถึงข้อมูลในระดับบุคคลทั่วไป ดังนั้นนายสมศักดิ์สามารถเข้าถึงข้อมูลได้เฉพาะข้อมูลของตนเองเท่านั้น ซึ่งจะเห็นว่า วิวสามารถจำกัดอำนาจในการเข้าถึงข้อมูลของผู้ใช้แต่ละคนได้ ดังแสดงตารางประวัติผู้ปฏิบัติงานในมุมมองของ นายสมศักดิ์ สุโข

ประวัติผู้ปฏิบัติงาน (ข้อมูลที่ได้มาโดยการผ่านวิว)

รหัสผู้ปฏิบัติงาน	ยศ	รหัสตำแหน่ง	ชื่อ-นามสกุล	ที่อยู่	วดป.เกิด	รหัสหน่วยที่สังกัด	รหัสแผนก
003	พล	04	สมศักดิ์ สุโข	105	01/12/2488	06	05

ตารางที่ 3.4 แสดงตัวอย่างการเข้าถึงข้อมูลของนายสมศักดิ์ สุโข โดยการผ่านวิว

นิยามศัพท์ที่เกี่ยวข้อง

1. ลักษณะประจำ(attribute) หมายถึง ลักษณะเฉพาะของข้อมูลที่ใช้แสดง ลักษณะ ประเภท สถานะ และคุณสมบัติของเอนทิตี เช่น เอนทิตี "ระบบ" ประกอบด้วย แอตตริบิว "รหัส" และ "ชื่อระบบ" เป็นต้น

2. ข้อมูลหลัก(entity) หมายถึง คน สถานที่ สิ่งของหรือความคิดที่ต้องการบันทึกจะต้องอยู่ได้โดยลำพังและสามารถบอกความแตกต่างระหว่างเอนทิตีได้

3. กุญแจนอก(foreign key) หมายถึง กุญแจที่ใช้เชื่อมความสัมพันธ์ระหว่างเอนทิตี ซึ่งเป็นกุญแจหลักในเอนทิตีอื่น

4. กุญแจหลัก(primary key) หมายถึง ลักษณะประจำที่ใช้เป็นตัวแทนในการสืบค้นรายละเอียดอื่นในเอนทิตีนั้น จะใช้ลักษณะประจำมากกว่า 1 ตัวก็ได้

5. ความสัมพันธ์(relationship) หมายถึง ความสัมพันธ์ระหว่างเอนทิตีตั้งแต่สองเอนทิตีขึ้นไป ซึ่งอาจเป็นแบบหนึ่งต่อหนึ่ง แบบหนึ่งต่อหลาย หรือแบบหลายต่อหลาย

6. วิว(view) หมายถึง ภาพของตาราง ในมุมมองของผู้ใช้แต่ละคน