บทที่ 2
หลักการและทฤษฎี

การพัฒนาระบบงานฐานข้อมูลให้มีประสิทธิภาพนั้นเราจะต้องเข้าใจพื้นฐานการทำงานของระบบฐานข้อมูล องค์ประกอบ และคำศัพท์ต่าง ๆ ที่ใช้ในระบบฐานข้อมูล อีกทั้งยังต้องเข้าใจ ขั้นตอนต่างๆ ที่ใช้ในการพัฒนาระบบงาน เพื่อจะได้มองภาพรวมของระบบได้อย่างชัดเจนและส่งผลให้พัฒนาโปรแกรมได้อย่างมีประสิทธิภาพต่อไป โดยเนื้อหาที่เกี่ยวข้องมีดังต่อไปนี้

2.1 ระบบฐานข้อมูล [1]

2.1.1 ระบบแฟ้มข้อมูล

ในอดีตองค์กรต่าง ๆ มักจัดเก็บเอกสารไว้ในแฟ้มเอกสารต่างๆ ซึ่งมีความเกี่ยวข้องกันทางด้านข้อมูลน้อยหรืออาจไม่มีเลย เช่น ประวัติการรักษาพยาบาล มักแยกเก็บในแฟ้มเอกสารเฉพาะคนไข้แต่ละคน และเมื่อองค์กรมีขนาดใหญ่มากขึ้นการค้นหาเอกสารต้องใช้เวลามากขึ้น และมีความยากลำบากมากขึ้น การจัดเก็บข้อมูลในคอมพิวเตอร์จึงถูกนำมาใช้ในองค์กรแทนการจัดเก็บในรูปแบบเดิม โดยรูปแบบการเก็บข้อมูลในคอมพิวเตอร์นี้อยู่ในรูปแฟ้มข้อมูลแฟ้มข้อมูลที่เก็บรวมอยู่ด้วยกันนี้เรียกว่า “ระบบแฟ้มข้อมูล”

การใช้งานระบบแฟ้มข้อมูล จะต้องอาศัยโปรแกรมเมอร์พัฒนาโปรแกรม เพื่ออ่านข้อมูลจากแฟ้มข้อมูลต่าง ๆ ขึ้นมาประมวลผล เช่น ภาษา COBOL, FORTRAN, BASIC เป็นต้น ทำให้มีข้อจำกัดในการเรียกใช้ข้อมูล ความซับซ้อนในการพัฒนาโปรแกรม และแต่ละโปรแกรมถูกผูกติดกับแฟ้มข้อมูลต่างๆ เช่น เมื่อต้องการเปลี่ยนแปลงโครงสร้างแฟ้มข้อมูลจึงจำเป็นต้องเขียนโปรแกรมขึ้นมาใหม่

2.1.2 ปัญหาของระบบแฟ้มข้อมูล

ด้วยการจัดเก็บแฟ้มข้อมูลของระบบแฟ้มข้อมูลเป็นอย่างเอกเทศและกระจัดกระจาย เช่น ในแต่ละหน่วยงานในองค์กรมีการสร้างระบบแฟ้มข้อมูลขึ้นมาใช้งานภายในหน่วยงานของตนเองทำให้เกิดปัญหาต่าง ๆ เช่น การจัดเก็บข้อมูลที่ซ้ำซ้อนกัน การขัดแย้งกันของข้อมูล

2.1.3 ระบบฐานข้อมูล

จากปัญหาต่าง ๆ ของระบบแฟ้มข้อมูล ทำให้เกิดการจัดเก็บข้อมูลในรูปแบบใหม่ที่เรียกว่า “ฐานข้อมูล (Database)” ฐานข้อมูลคือเป็นการนำเอาข้อมูลต่างๆ ที่มีความสัมพันธ์กันมาเก็บไว้ในที่เดียวกันรวมกันเป็นฐานข้อมูลรวมทำให้เกิดการใช้ข้อมูลร่วมกัน

2.1.4 องค์ประกอบของระบบฐานข้อมูล

- ข้อมูล
- ฮาร์ดแวร์ (Hardware)
- ซอฟแวร์ (Software)
- ผู้ใช้ระบบฐานข้อมูล

2.1.5 ประโยชน์ของฐานข้อมูล

- สามารถลดความซ้ำซ้อนของการเก็บข้อมูลได้
- สามารถหลีกเลี่ยงความขัดแย้งกันของข้อมูลได้
- ในแต่ละองค์กรสามารถใช้ข้อมูลร่วมกันได้
- สามารถกำหนดระบบความปลอดภัยให้กับข้อมูลได้ โดยกำหนดระดับความสามารถในการเรียกใช้ข้อมูลของผู้ใช้แต่ละคน ให้แตกต่างกันตามความรับผิดชอบได้
- สามารถรักษาความถูกต้องให้กับข้อมูลได้โดยการระบุกฎเกณฑ์ในการควบคุมความผิดพลาดในการป้อนข้อมูลผิด
- ทำให้ข้อมูลเป็นอิสระจากโปรแกรมที่ใช้งานข้อมูลนั้นทำให้ผู้พัฒนาโปรแกรมทำการแก้ไขโครงสร้างของข้อมูลได้โดยไม่มีผลกระทบกับโปรแกรมที่เรียกใช้ข้อมูลนั้น

2.2 สถาปัตยกรรมของระบบฐานข้อมูล

สถาปัตยกรรมของฐานข้อมูลเป็นการอธิบายถึงรูปแบบและโครงสร้างของข้อมูลภายในระบบฐานข้อมูลโดยทั่วไปในระดับแนวความคิดไม่ขึ้นอยู่กับโครงสร้างของระบบฐานข้อมูลนั้น ๆ สำหรับสถาปัตยกรรมของระบบฐานข้อมูลที่นิยมใช้ได้แก่ สถาปัตยกรรม ANSI/SPARC ได้แบ่งออกเป็น 3 ระดับดังนี้

- ระดับ **Internal** เป็นสถาปัตยกรรมในระดับที่เกี่ยวข้องกับโครงสร้างทางกายภาพในการจัดเก็บ ข้อมูลในฐานข้อมูลมากที่สุดเนื่องจากเป็นระดับที่กล่าวถึงการจัดเก็บข้อมูล
- ระดับ **External** เป็นระดับที่เกี่ยวข้องกับผู้ใช้มากที่สุด เนื่องจากเป็นระดับที่กล่าวถึงมุมมองที่มีต่อข้อมูลของผู้ใช้แต่ละคน
- ระดับ **Conceptual** เป็นสถาปัตยกรรมที่กล่าวถึงโครงสร้างของข้อมูลในระดับแนวความคิด **ซึ่งเป็นภาพของโครงสร้างข้อมูลที่ใช้แทนโครงสร้างทางกายภาพของข้อมูลที่แท้จริงที่เก็บอยู่ในฐานข้อมูล**

2.3 แบบจำลองของฐานข้อมูลเชิงสัมพันธ์ (Relational Database Model)

ฐานข้อมูลที่มีโครงสร้างข้อมูลในแบบจำลองของฐานข้อมูลเชิงสัมพันธ์ได้รับการพัฒนาขึ้นจากแบบจำลองที่กล่าวถึงความสัมพันธ์ระหว่างข้อมูลที่มีชื่อว่า Relation Model ข้อมูลที่จัดเก็บอยู่ในฐานข้อมูลที่มีโครงสร้าง ข้อมูลในแบบจำลองของฐานข้อมูลเชิงสัมพันธ์ จะถูกแยกจัดเก็บออกเป็นหน่วยย่อย ๆ ที่เรียกว่า Relation หรือ ที่เรียกว่า Table ที่อยู่ในรูปของตารางที่ประกอบด้วยชุดของแถวและชุดของสดมภ์ ข้อมูลที่จัดเก็บอยู่ในแต่ละ ตารางจะเป็นข้อมูลที่แยกเป็นเอกเทศ แต่สามารถนำมาสร้างความสัมพันธ์ร่วมกันได้โดยความสัมพันธ์ที่สร้างขึ้นนี้จะอยู่ในรูปทางแนวความคิดมากกว่าทางกายภาพ

2.4 โครงสร้างข้อมูลของฐานข้อมูลแบบ Relational

ใน Relation Model ได้นิยามต่าง ๆ คือ Relation, Tuple, Attribute, Domain เพื่อใช้อธิบายถึงข้อมูลใน Relation Model นี้

2.4.1 Relation

เป็นหน่วยที่ใช้จัดเก็บข้อมูลที่อยู่ในรูปของตารางประกอบด้วยแถว(Row)และสดมภ์ (Column) สำหรับชื่อของแต่ละแถวของ Relation เรียกว่า "Tuple" ส่วนชื่อของแต่ละสดมภ์เรียกว่า "Attribute" ตัวอย่างเช่น

ตารางที่ 2.1 ตัวอย่าง Relation

EmpID	Name	Surname	Sex	Salaly	DepID
001	สมบูรณ์	สุขมาก	M	10000	01
002	จันจิรา	แจ้งเกิด	F	12,000	03

2.4.2 Domain

เป็นการนิยามของขอบของค่าที่เป็นไปได้ให้กับข้อมูลในแต่ละ Attribute เช่นการกำหนดให้ค่าของเงินเดือนพนักงาน จะต้องมีค่ามากกว่า 0 หรือ การกำหนดเพศของพนักงานจะต้องมีค่าเป็น ชาย (M) หรือหญิง (F)

2.4.3 Key

Key คือ Attribute หรือชุดของ Attribute ที่ทำให้ข้อมูลแต่ละ Tuple ใน Relation มีค่าข้อมูลที่ไม่ซ้ำกัน เช่น Attribute EmpID ในตัวอย่างในข้อ 2.4.1 key นั้นแบ่งออกเป็น ดังนี้

- **Primary key** จะถูกใช้เป็น key หลักสำหรับตรวจสอบการซ้ำกันของข้อมูลระหว่างที่ทำการป้อนข้อมูลหรือกำหนดข้อมูลใหม่ให้กับ Relation เช่น Attribute EmpID

ตารางที่ 2.2 ตัวอย่าง Primary Key

EmpID	Name	Surname	Sex	Salaly	DepID
001	สมบูรณ์	สุขมาก	M	10000	01
002	จันจิรา	แจ้งเกิด	F	12,000	03

- **Foreign Key** จะเป็น Attribute ใด Attribute หนึ่งใน Relation1 ที่ใช้อ้างอิงไปยัง Attribute ที่ทำหน้าที่เป็น Primary key ของอีกตารางหนึ่งใน "DepID" ของตารางแรกจะเป็น Foreign Ket

ตารางที่ 2.3 ตัวอย่าง Foreign Key

EmpID	Name	Surname	Sex	Salaly	DepID
001	สมบูรณ์	สุขมาก	M	10000	01
002	จันจิรา	แจ้งเกิด	F	12,000	03

DeptID	Depname
01	ธุรการ
03	บริการ

- **Candidate Key** เป็น key ที่สามารถระบุตำแหน่งของแต่ละ Tuple ใน Relation ได้ เช่น ในตารางตัวอย่างเรากำหนดให้ Attribute "Name" เป็น Candidate key ด้วยเพื่อให้ค่าของข้อมูลใน Tuple นั้นมีค่าไม่ซ้ำกัน

ตารางที่ 2.4 ตัวอย่าง Candidate Key

EmpID	Name	Surname	Sex	Salaly	DepID
001	สมบูรณ์	สุขมาก	M	10000	01
002	จันจิรา	แจ้งเกิด	F	12,000	03

- **Null** คือค่าของข้อมูลที่ไม่สามารถระบุค่าได้ ซึ่งอาจเกิดจากการกรอกข้อมูลไม่ครบถ้วน เช่น

ตารางที่ 2.5 ตัวอย่าง Null value

EmpID	Name	Surname	Sex	Salaly	DepID
001	สมบูรณ์	สุขมาก	M	10000	01
002	จันจิรา	แจ้งเกิด	F	12,000	03

2.4.4 กฎ Referentail Intrigrity

เป็นกฎที่กำหนดไว้ว่า "ฐานข้อมูลใด ๆ จะต้องไม่ปรากฏ Relation ที่มีข้อมูลที่มีค่าของ Foreign key ที่ไม่ขึ้นกับ Primary key ของ Relation อื่น

2.4.5 กฎ Entity Intrigrity

เป็นกฎที่กำหนดว่า Primary key ของ Relation ใด ๆ ห้ามมีค่าเป็น Null

2.5 การออกแบบฐานข้อมูล [1,2]

ฐานข้อมูลนับเป็นส่วนสำคัญสำหรับงานสารสนเทศที่ใช้คอมพิวเตอร์ในการประมวลผลเนื่องจากเป็นการเป็นส่นที่ใช้จัดเก็บข้อมูลต่าง ๆ ซึ่งใช้เป็นอินพุทของระบบสารสนเทศจึงต้องให้ความสำคัญกับการออกแบบฐานข้อมูลเช่นเดียวกับการออกแบบส่วนประมวลผล

2.5.1 วงจรชีวิตของการพัฒนาระบบฐานข้อมูล (Database Life Cycle)

วงจรชีวิตของการพัฒนาระบบฐานข้อมูล เป็นขั้นตอนที่กำหนดขึ้น เพื่อใช้เป็นแนวทางในการพัฒนาระบบฐานข้อมูลขึ้นใช้งาน ประกอบด้วยขั้นตอนต่าง ๆ ดังนี้

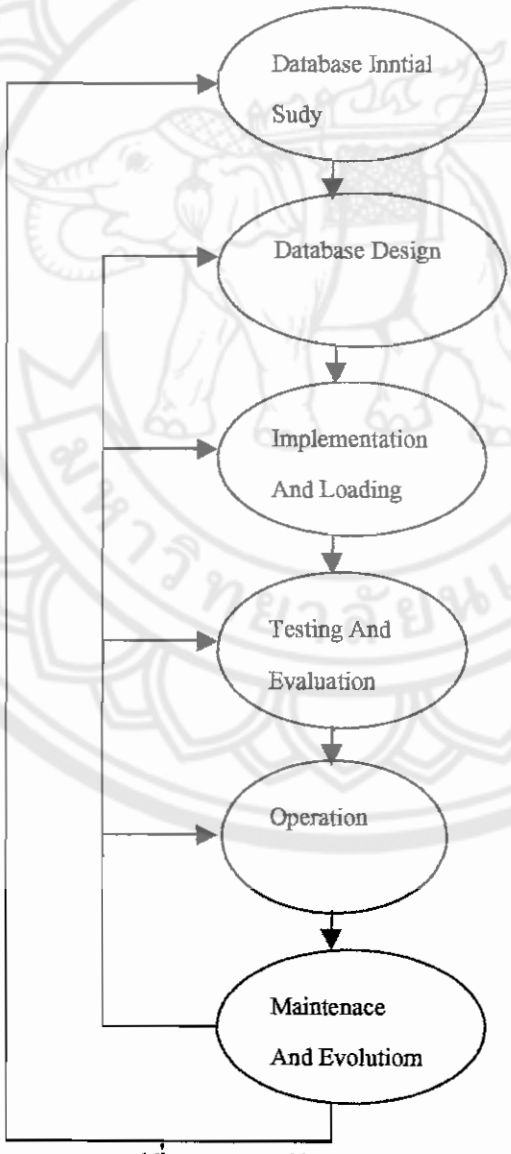
2.5.1.1 Database Design เป็นขั้นตอนในการเอารายละเอียดต่าง ๆ ที่ได้จากการวิเคราะห์ในขั้นตอนแรมากำหนดเป็นแนวทางในการออกแบบ ซึ่งแบ่งเป็น 3 ระดับ คือ การออกแบบฐานข้อมูลในระดับ Conceptual, Logical และ Physical

2.5.1.3 Implementation and Loading เป็นขั้นตอนที่นำเอาโครงร่างต่าง ๆ ของระบบฐานข้อมูลที่ได้จากการออกแบบในขั้นตอนการออกแบบ มาสร้างเป็นข้อมูลที่จะใช้เก็บจริง รวมทั้งแปลงข้อมูลของระบบงานเดิม ให้สามารถนำมาใช้งานในระบบฐานข้อมูลที่พัฒนาขึ้นมาใหม่

2.5.1.4 Testing and Evaluation เป็นขั้นตอนของการทดสอบระบบฐานข้อมูลที่พัฒนาข้อผิดพลาดต่าง ๆ รวมทั้งทำการประเมินความสามารถของระบบฐานข้อมูลเพื่อนำไปเป็นแนวทางในการปรับปรุงให้รองรับความต้องการความต้องการของผู้ใช้ในด้านต่าง ๆ ได้อย่างครบถ้วนและถูกต้อง

2.5.1.5 Operation เป็นขั้นตอนที่นำเอาระบบฐานข้อมูลขึ้นเรียบร้อยแล้วไปใช้งานจริง

2.5.1.6 Maintainance and Evolution เป็นขั้นตอนที่เกิดขึ้นระหว่างการใช้ฐานข้อมูลจริง เพื่อบำรุงรักษาให้ระบบฐานข้อมูลทำงานได้อย่างมีประสิทธิภาพ รวมทั้งขั้นตอนการแก้ไขและปรับปรุงระบบฐานข้อมูลในกรณีที่มาการเพิ่มหรือเปลี่ยนแปลงความต้องการของผู้ใช้ สามารถแสดงได้ดังนี้



รูปที่ 2.1 วงจรชีวิตของการพัฒนาฐานข้อมูล

รายละเอียดที่ได้จากแต่ละขั้นตอนของการพัฒนาระบบฐานข้อมูลจะสามารถสะท้อนกลับไปยังการทำงานในขั้นตอนก่อนหน้า ซึ่งจะช่วยปรับปรุง และแก้ไขข้อผิดพลาดในการออกแบบของขั้นตอนการออกแบบที่ผานมาได้เป็นอย่างดี

2.5.2 ขั้นตอนการออกแบบฐานข้อมูล

การออกแบบฐานข้อมูลแบ่งออกได้เป็น 3 ขั้นตอนดังนี้

2.5.2.1 การออกแบบฐานข้อมูลในระดับ Conceptual การออกแบบฐานข้อมูลในระดับนี้ จะเป็นการกำหนดโครงร่างเริ่มต้น ที่มีจุดหมายเพื่ออธิบายโครงสร้างหลัก ๆ ของข้อมูลภายในระบบฐานข้อมูลโดยไม่คำนึงถึงฐานข้อมูลที่นำมาใช้ การออกแบบใน ระดับนี้มีวามสำคัญมากเนื่องจากโครงสร้างที่ได้จากการออกแบบในขั้นตอนนี้จะถูกนำไปใช้ขั้นตอนต่อไป โครงร่างหรือที่เรียกว่า Schema ที่ได้จากการออกแบบในขั้นตอนนี้เรียกว่า Conceptual Schema

2.5.2.2 การออกแบบในระดับ Logical การออกแบบในระดับนี้จะเป็นระดับที่ต่อเนื่องมาจากระดับ conceptual กล่าวคือ การออกแบบในระดับนี้จะอาศัยโครงร่างที่จากการออกแบบในระดับ conceptual มาปรับปรุงให้มีโครงสร้างที่เป็นไปตามโครงสร้างข้อมูลที่จะนำมาใช้งาน โดยยังไม่คำนึงผลิตภัณฑ์ทางด้านฐานข้อมูลที่จะนำมาใช้งาน การออกแบบในขั้นตอนนี้ต้องปรับปรุงโครงสร้างบางอย่างใน Conceptual Schema ให้สอดคล้องกับฐานข้อมูลที่จะนำมาใช้งาน การออกแบบในขั้นตอนนี้จึงต้องมีการตรวจสอบความถูกต้องของโครงร่างที่ออกแบบขึ้นกับส่วนประมวลผลต่าง ๆ ที่ออกแบบไว้รวมทั้งจะต้องแปลงโครงร่างต่าง ๆ ให้อยู่ในรูป Relation

2.5.2.3 การออกแบบฐานข้อมูลในระดับ Physical การออกแบบในระดับนี้ จะเป็นขั้นตอนสุดท้ายของการออกแบบฐานข้อมูลในขั้นตอนนี้ จะเป็นการปรับปรุงโครงสร้างของโครงร่างที่ออกแบบเช่นเดียวกัน แต่การปรับปรุงโครงสร้างของการออกแบบฐานข้อมูลในขั้นตอนนี้ จะเป็นการนำเอาโครงร่างที่ได้จากการออกแบบในระดับ Logical มาปรับปรุงโครงสร้างให้เป็นไปตามโครงสร้างของผลิตภัณฑ์ทางด้านฐานข้อมูลที่จะนำมาใช้งาน ผลลัพธ์ที่ได้จากการออกแบบในระดับนี้คือโครงสร้างของระบบฐานข้อมูล ที่สามารถนำไปใช้การสร้างตัวฐานข้อมูลจริง

2.6 รูปแบบการไหลของข้อมูล (Data flow) [3]

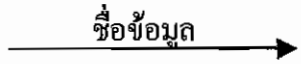
กรรมวิธีการวิเคราะห์ระบบอย่างมีโครงสร้างนั้น วิธีหนึ่งที่นิยมในทางปฏิบัติคือการ มองภาพรวมในรูปแบบการไหลของข้อมูล (Data flow) โดยที่วิธีนี้จะช่วยให้นักวิเคราะห์สามารถแบ่งระบบเป็นระบบย่อยได้ง่ายขึ้นและสามารถตรวจสอบได้สะดวก

การนำเสนอระบบแบบการไหลของข้อมูลนั้นจะใช้สัญลักษณ์แทนการบรรยายการทำงานของระบบ ซึ่งลักษณะที่จะใช้จะเป็นรูปวงกลม สี่เหลี่ยมจัตุรัส สี่เหลี่ยมผืนผ้าปลายเปิด เส้นโค้ง ลูกศร โดยนำสัญลักษณ์เหล่านี้มาเชื่อมต่อ การแสดงการต่อเนื่องของข้อมูลและการประมวลผล

2.6.1 สัญลักษณ์ Data Flow Diagram (DFD)

ในแผนภาพของ DFD จะประกอบด้วยสัญลักษณ์ต่าง ๆ ดังนี้

1. ลูกศร ใช้แทนการไหลของข้อมูลพร้อมกับชื่อของข้อมูลนั้น ๆ จะต้องกำกับไว้ด้วย



รูปที่ 2.2 การแทนกระแสข้อมูลเป็นลูกศร

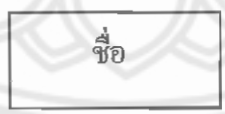
2. รูปวงกลม ใช้แทนกริยาการกระทำต่อข้อมูลที่ไหลเข้ามา โดยไม่คำนึงว่าจะเป็นการกระทำโดยคน หรือ คอมพิวเตอร์ก็ตาม จะได้มาซึ่งผลลัพธ์ที่จะไหลออกจากวงกลมภายในวงกลม จะระบุคำสั้น ๆ ที่จะใช้แทนการกระทำต่อข้อมูล

A = ข้อมูลเข้า

B = ข้อมูลออก

รูปที่ 2.3 การแทนกระแสข้อมูลเป็นลูกศร

3. รูปสี่เหลี่ยม ใช้แทนนามที่อยู่ภายนอกระบบซึ่งเป็นการกำเนิดของข้อมูลหรือสิ้นสุดของข้อมูล โดยมีชื่ออยู่ภายในสี่เหลี่ยม



รูปที่ 2.4 การแทนนามที่อยู่นอกระบบ

4. รูปสี่เหลี่ยมผืนผ้าปลายเปิด เป็นตัวแทนของแหล่งเก็บข้อมูลหรือแฟ้มข้อมูลเสมือนเป็นตัวพักหรือช่วงขาดของการไหลของข้อมูลเพื่อนำไปเก็บเท่านั้น การกำหนดชื่อของแหล่งเก็บข้อมูลต้องอยู่ในสี่เหลี่ยม

ชื่อ

รูปที่ 2.5 การแทนแหล่งเก็บข้อมูล

5. สัญลักษณ์เพิ่มเติม จะใช้เติมลงในสัญลักษณ์ที่กล่าวมาข้างต้นเพื่อแสดงความเป็นสิ่งเดียวกัน แต่จะถูกกล่าวหลาย ๆ ครั้งในแผนภาพ

รูปที่ 2.6 การแทนสัญลักษณ์เพิ่มเติม

2.6.2 ลำดับชั้นใน Data Flow Diagram

ในการเขียน DFD นักวิเคราะห์ระบบจะต้องมองระบบจากภาพรวมก่อนจากนั้นมองลึกเข้าสู่รายละเอียดข้างในของระบบยิ่งมองลึกมากเท่าใด ก็จะยิ่งเห็นรายละเอียดของระบบย่อยได้มากขึ้นเท่านั้น

DFD ระดับที่ 0

ให้คิดว่าระบบทั้งระบบเป็น PROCESS หรือวงกลมหนึ่งวง มีลูกศรแทน INPUT และ OUTPUT ตามที่จำเป็น

DFD ระดับที่ 1

ให้แตกวงกลมที่ลำดับ 0 ออกเป็นวงกลมย่อย 2-5 วงตามความเหมาะสม

DFD ระดับที่ 2

ให้แตกวงกลมที่ลำดับ 1 ออกเป็นวงกลมย่อยลงไปอีกเท่าที่จะทำได้

DFD ระดับที่ 3

ถ้าจำเป็นก็ต้องตรวจดูว่า วงกลมใดในภาพลำดับ 2 ยังมีความซับซ้อนที่จำเป็นต้องแตกย่อยก็ต้องแตกย่อย ก็ต้องสร้างแผนภาพประกอบด้วยวงกลมย่อยแทนวงกลมนั้นให้ได้รายละเอียดสุดท้าย

2.6.3 เหตุผลในการใช้ Data Flow Diagram

เหตุผลที่ต้องใช้ DFD เป็นแผนภาพของระบบก็เพราะ

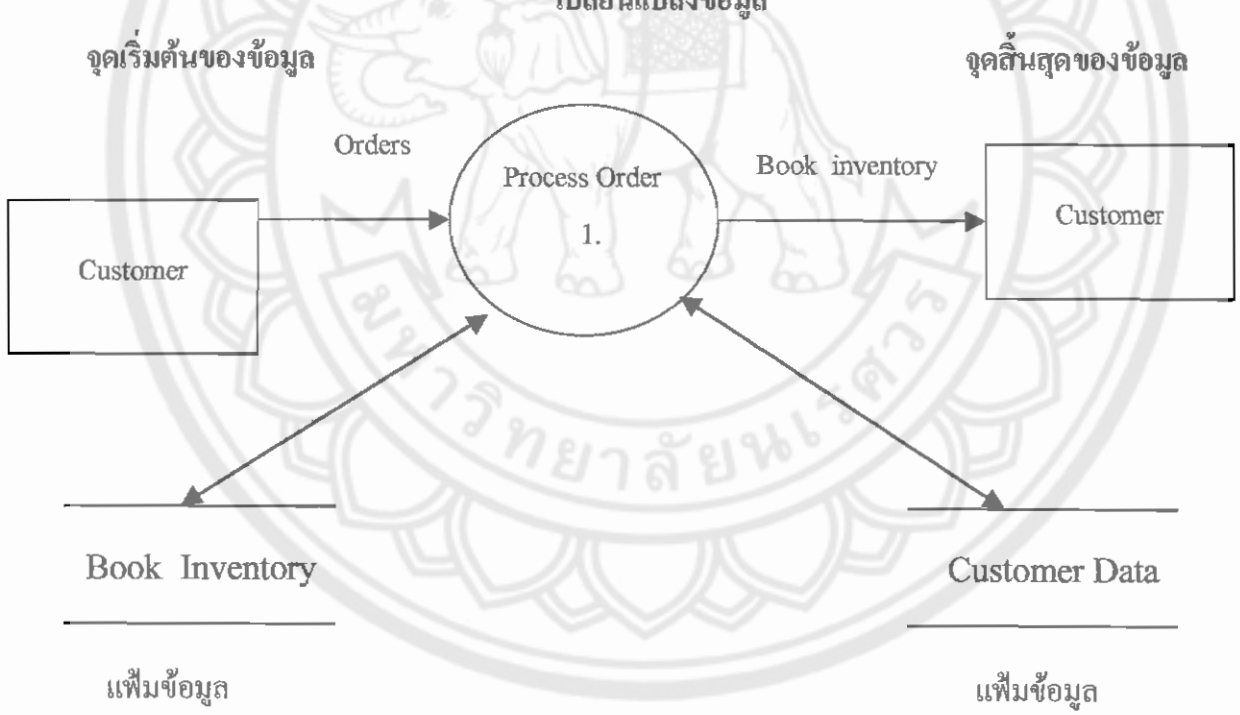
2.6.3.1 DFD ช่วยให้นักวิเคราะห์ระบบสามารถ

- สรุปข้อมูลที่เกี่ยวกับระบบ
- เข้าใจถึงปัญหาสำคัญของระบบและระบุส่วนของการทำงานที่ซ้ำซ้อน
- เข้าใจถึงความสัมพันธ์ระหว่างส่วนต่าง ๆ ของระบบการประกอบกันเป็นระบบ
- พัฒนาระบบได้อย่างมีประสิทธิภาพ

2.6.3.2 DFD เป็นเอกสารร่วมที่ช่วยให้นักวิเคราะห์ระบบ และผู้ใช้ระบบสามารถเข้าใจระบบและตรวจสอบความถูกต้องได้สองฝ่าย

2.6.3.3 ในการตรวจสอบเรื่องเวลาที่ใช้ในแต่ละขบวนการนั้น นักวิเคราะห์สามารถใช้ DFD เป็นเครื่องมือที่ช่วยให้ทราบถึงขอบเขตในการพัฒนารูปแบบของระบบว่ามีทางที่จะเป็นไปได้อย่างไรบ้าง

2.6.4 ตัวอย่างการใช้ Data Flow Diagram



รูปที่ 2.7 การปฏิบัติงานของบริษัท หนังสือไทย

เป็นแผนภาพแสดงการสั่งซื้อหนังสือจากร้านหนังสือไทยลูกค้าในช่องสี่เหลี่ยมซ้ายมือสั่งสินค้ามายังวงกลม สิ่งที่จะต้องทำก็คือ ตัดสต็อกของหนังสือที่ถูกสั่งออกไปพร้อมกับใบเก็บเงินไปยังลูกค้าด้านขวามือ โดยรูปแบบของสี่เหลี่ยมจัตุรัส ของลูกค้ามายังวงกลม สิ่งที่จะต้องทำก็คือ ตัดสต็อก

ของหนังสือที่ถูกสั่งออกไปพร้อมกับใบเก็บเงินไปยังลูกค้าขวามือ โดยรูปแบบของสี่เหลี่ยมจัตุรัส ของลูกค้าทั้งด้านซ้ายและขวามีลักษณะเหมือนกัน จะหมายถึงสิ่งเดียวกัน

2.7 Entity -Relationship Model [1]

การออกแบบฐานข้อมูลใช้งานในระบบสารสนเทศใด ๆ ต้องอาศัยแบบจำลองของข้อมูล เพื่อนำเสนอรายละเอียดต่าง ๆ ที่เกี่ยวข้องกับข้อมูลในฐานข้อมูลที่ออกแบบ แบบจำลองของข้อมูลที่นิยมใช้ ได้แก่ Entity-Relationship Model

2.7.1 Semantic Model

แบบจำลองของข้อมูลในยุคแรก ๆ มีข้อจำกัดในการนำเสนอรายละเอียดที่เกี่ยวข้องกับข้อมูลในฐานข้อมูล กล่าวคือ จะมีการนำเสนอเฉพาะรายละเอียดทางด้านโครงสร้างของฐานข้อมูล เช่น คุณสมบัติ Atomicity ของข้อมูล กฎระเบียบต่าง ๆ ที่ใช้สำหรับควบคุมความถูกต้องของ ข้อมูล ฯลฯ เป็นต้น แต่ยังไม่มีการนำเสนอรายละเอียดทางด้านความหมาย (Semantic) ของข้อมูลภายในฐานข้อมูลนั้น ๆ เช่น จำนวน และน้ำหนักของสินค้าว่ามีความสัมพันธ์กันอย่างไร หรือ Domain ของ ข้อมูลต่าง ๆ สามารถมีค่าเป็นอะไรได้บ้าง หรือข้อมูลใดที่ทำหน้าที่เป็น Candidate Key หรือ Foreign Key ฯลฯ ดังนั้น จึงมีการคิดค้นแบบจำลองของข้อมูลใหม่ขึ้นที่เรียกว่า "Semantic Model"

2.7.2 คำศัพท์ที่ใช้ใน Semantic Model

ใน Semantic Model ได้มีการนิยามคำขึ้นแทนข้อมูลในความหมายต่าง ๆ ที่เรียกว่า Concept ดังนี้

2.7.2.1 Entity

Entity เปรียบเสมือนกับคำนามที่สามารถระบุได้ในความเป็นจริง อาจเป็นสิ่งที่จับต้องได้ เช่น นายไมเคิล หนังชื่อที่ชื่อ "ระบบฐานข้อมูล" หรือเป็นสิ่งที่อยู่ในรูปนามธรรมที่ไม่สามารถจับต้องได้ เช่น จำนวนวันลาพักร้อนของพนักงาน ชื่อเมืองนำแต่ละ Entity มารวมกันภายใต้คุณลักษณะใดลักษณะหนึ่งที่เหมือนกันแล้ว Entity เหล่านั้น จะถูกเรียกว่า "Entity Set" ดังตัวอย่าง

ตารางที่ 2.6 ตัวอย่าง Entity

Entity ของ นาย ก.

นาย ก.	ไทย

Entity ของ นาย ข.

นาย ข.	ไทย

Entity Set "ประชากรสัญชาติไทย"

นาย ก.	ไทย
นาย ข.	ไทย

2.7.2.2 Property หรือ Attribute

Property หรือ Attribute คือ ข้อมูลที่แสดงลักษณะ และคุณสมบัติของ Entity เช่น Property ของ Entity Set ที่ชื่อ "รถยนต์" ที่ประกอบด้วย หมายเลขทะเบียนรถยนต์ หมายเลขตัวถัง หมายเลขเครื่องยนต์ ยี่ห้อ รุ่น สี ดังนี้

ตารางที่ 2.7 ตัวอย่าง Property

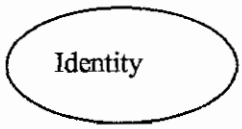
Property

ทะเบียน	หมายเลขตัวถัง	หมายเลขเครื่องยนต์	ยี่ห้อ	รุ่น	สี
4ษ-2100	2Azoo12523	EE859222	TOYOTA	CORONA	ฟ้า
1ศ-1700	1WE9944560	AS985112	HONDA	CIVIC	ม่วง
8ฮ-2545	6AA4456455	EZ123444	NISSAN	SUNNY	แดง

2.7.2.3 Identity

แต่ละ Entity ภายใต้ Entity Set เดียวกัน ถึงแม้ว่าจะต้องมี Property ที่เหมือนกัน แต่อย่างไรก็ตาม จะต้องมี Property ใน Property หนึ่ง ซึ่งเป็นเอกลักษณ์เฉพาะของ Entity นั้น เช่น Property "หมายเลขประชาชน" ของแต่ละ Entity ใน Entity Set "ประชาชน" ซึ่งจะไม่มีหมายเลขใดที่ซ้ำกัน ดังตารางข้างล่าง

ตารางที่ 2.8 ตัวอย่าง Identity



หมายเลขบัตรประชาชน	ชื่อ	นามสกุล	เพศ	สัญชาติ	วันเดือนปีเกิน
123456789	แดง	สด	ชาย	ไทย	12/12/22
987654321	ดำ	ดี	หญิง	ไทย	11/11/22

สำหรับ Property ที่สามารถนำมากำหนดเป็นเอกลักษณ์เฉพาะให้กับแต่ละ Entity นี้จะเรียกว่า " Identity"

2.7.2.4 Relationship

ได้แก่ Entity Set ที่สร้างขึ้นจาก 2 Entity Set เดิมหรือมากกว่า เพื่อใช้แสดงความสัมพันธ์ระหว่างแต่ละ Entity ใน Entity Set เดิมเหล่านั้น ในการสร้าง Relationship อาจสร้างด้วยการนำเอาแต่ละ Entity ใน Entity Set เดิมเหล่านั้นมาเชื่อมโยงข้อมูลกันภายใต้ค่าของ Property ที่เหมือนกัน ซึ่งการสร้างความสัมพันธ์ในลักษณะนี้ Property ของ Relationship จะเกิดจากการนำเอา Property ของแต่ละ Entity Set มารวมกัน เช่น Relationship จะเกิดจากการนำเอา Property ของแต่ละ Entity Set มารวมกัน เช่น Relationship ที่ชื่อ "สังกัดคณะ" ซึ่งเกิดจากการที่ Entity Set "นักศึกษา" และ "คณะ" มี Property "รหัสคณะ" ที่เหมือนกัน ดังนี้

ตารางที่ 2.9 ตัวอย่าง Relationship

Entity Set "นักศึกษา"

รหัสนักศึกษา	ชื่อ-สกุล	เพศ	รหัสคณะ
380012	เอก โท	ชาย	02
381202	ตรี จัตวา	หญิง	01
380052	เอก ตรี	ชาย	03

Entity Set "คณะ"

รหัสคณะ	คณะ
01	วิศวกรรมศาสตร์
02	บริหารธุรกิจ
03	วิทยาศาสตร์

Relationship "สังกัดคณะ"

รหัสนักศึกษา	ชื่อ-สกุล	เพศ	รหัสคณะ	คณะ
380012	เอก โท	ชาย	02	บริหารธุรกิจ
381202	ตรี จัตวา	หญิง	01	วิศวกรรมศาสตร์
380052	เอก ตรี	ชาย	03	วิทยาศาสตร์

2.7.2.5 Subtype และ Super

เป็น Entity Set ที่สามารถแบ่งสมาชิกของ Entity Set นั้น ออกเป็น Entity Set ย่อย ซึ่งสมาชิกของแต่ละ Entity Set ย่อยนั้น นอกเหนือจากจะต้องมี Property ที่เหมือนกับ Entity Set หลักแล้ว Entity Set ย่อยนั้น นอกเหนือจากจะต้องมี Property ที่เหมือนกับ Entity Set หลักแล้ว Entity Set ย่อยเหล่านั้น จะต้องมี Property ที่มีค่าซึ่งบ่งบอกถึงความแตกต่างของสมาชิกในแต่ละ Entity Set ย่อยเหล่านั้น สำหรับ Entity Set เหล่านี้ จะถูกเรียกว่า "Subtype Entity" ส่วน Entity Set หลัก จะถูกเรียกว่า "Supertype Entity" เช่น Entity Set "EMPLOYEE" ที่มี Property "MILITARY_STATUS" ที่ใช้กำหนดสถานภาพการเกณฑ์ของพนักงาน ซึ่งจะปรากฏเฉพาะพนักงานชายเท่านั้น ดังตัวอย่างข้อมูลต่อไปนี้

ตารางที่ 2.10 ตัวอย่าง Supertype

Entity Set "EMPLOYEE"

EmpID	NAME	SEX	MILITARY_STATUS	SALARY
001	หนึ่ง	M	1	8500
002	สอง	F	-	12000
003	สาม	M	0	15000
004	สี่	F	-	22222
005	ห้า	M	2	15250

จากตัวอย่างข้อมูล จะสังเกตเห็นว่า ข้อมูลใน Property "MILITARY_STATUS" จะปรากฏเฉพาะพนักงานชายเท่านั้น ดังนั้น จึงสามารถแบ่งข้อมูลใน Entity Set นี้ออกเป็น 2 Entity Set ย่อย ซึ่งได้แก่ Entity Set "MALE" สำหรับเก็บข้อมูลของพนักงานชาย และ Entity Set "FEMALE" สำหรับเก็บข้อมูลของพนักงานหญิงดังนี้

ตารางที่ 2.11 ตัวอย่าง Subtype

Entity Set "FEMALE"

EmpID	NAME	SEX	SALARY
002	สอง	F	12000
004	สี่	F	22222

Entity Set "MALE"

EmpID	NAME	SEX	MILITARY_STATUS	SALARY
001	หนึ่ง	M	1	8500
003	สาม	M	0	15000
005	ห้า	M	2	15250

ข้อสังเกต ถ้าสังเกตดู จะพบว่า Property คือ แต่ละ Attribute ส่วน Entity คือแต่ละ Tuple ต่าง ๆ ใน Relationship Model แต่ที่ในที่นี้ไม่เรียก Property ว่า Attribute และ Entity ว่า Tuple เนื่องจาก Semantic Model สามารถนำไปใช้ในการออกแบบฐานข้อมูลได้หลายระดับ ส่วนที่กล่าวถึงในตอนนี้ เป็นการออกแบบในระดับแนวความคิด ที่มีจุดหมายเพื่อนำเสนอข้อเท็จจริงต่าง ๆ ที่เกี่ยวข้องกับข้อมูล โดยยังไม่ได้ระบุโครงสร้างฐานข้อมูลที่ใช้ ดังนั้น จึงเรียก Property แทนคำว่า Attribute และ Entity แทนคำว่า Tuple เนื่องจากทั้ง 2 คำ จะใช้ฐานข้อมูลที่มีโครงสร้างแบบ Relation

2.7.3 Entity -Relationship Model

Semantic Model ที่นิยมใช้มากที่สุด ได้แก่ Entity –Relationship Model หรือที่นิยมเรียกกันสั้น ๆ ว่า "E-R Model" E-R Model นับเป็นแบบจำลองที่ครอบคลุมนิยามต่าง ๆ ที่กำหนดไว้ใน Semantic Model เนื่องจากมีรูปแบบที่ใช้แทนทุก ๆ แนวความคิดที่กำหนดไว้ Semantic Model ซึ่งได้แก่ Entity, Property, Relation และ Subtype สำหรับแผนภาพที่สร้างขึ้นโดยใช้รูปภาพต่าง ๆ ภายใน E-R Model เพื่อแสดงความเป็นจริงต่าง ๆ ของข้อมูลในฐานข้อมูล จะเรียกว่า Entity-Relationship Diagram หรือที่นิยมเรียกกันสั้น ๆ ว่า แผนภาพ E-R (E-R Diagram)

2.7.3.1 Entity

ได้แก่ Entity Set ต่าง ๆ ที่นิยามไว้ใน Semantic Model แต่ใน E-R Model จะเรียก Entity Set ว่า Entity แทน สำหรับ Entity ใน E-R Model จะแบ่งออกเป็น 3 ประเภทดังนี้

1. Regular Entity หรือบางครั้งเรียกว่า Strong Entity ได้แก่ Entity ส่วนใหญ่ที่ปรากฏอยู่ในระบบฐานข้อมูล ซึ่งสมาชิกภายใน Entity ประเภทนี้ สามารถมีคุณสมบัติ Identity

ได้ด้วยตัวมันเอง เช่น Entity "EMPLOYEE" ซึ่งสมาชิกภายใน Entity นี้ได้แก่ ข้อมูลของพนักงานแต่ละคนในบริษัท ซึ่งประกอบด้วย Property ต่าง ๆ ดังนี้

ตารางที่ 2.12 ตัวอย่าง Regular Entity

Entity "EMPLOYEE"

EmpID	NAME	SEX	SALARY
0001	สมชาย นิลกลัด	M	8500
0002	สมถวิล กลั่นเจริญ	F	9000
0003	เจริญ ก้าวหน้า	M	12000

จากตัวอย่างข้อมูลของ Entity "EMPLOYEE" จะสังเกตเห็นว่า สมาชิก ซึ่งได้แก่ ข้อมูลในแต่ละแถวของตาราง สามารถมีคุณสมบัติ Identity ได้โดยอาศัยค่าของ Property " EmpID" เนื่องจากพนักงานแต่ละคน จะมีหมายเลขประจำตัวพนักงานที่ไม่ซ้ำกัน สำหรับรูปภาพที่ใช้แทน Entity ประเภทนี้ ได้แก่ รูปสี่เหลี่ยมผืนผ้า โดยมีชื่อเสียงของ Entity นั้นอยู่ภายใน ดังรูป

รูปที่ 2.8 Regular Entity

2. Weak Entity เป็น Entity ที่มีลักษณะตรงข้ามกับ Regular Entity กล่าวคือ สมาชิกของ Entity ประเภทนี้จะสามารถมีคุณสมบัติ Identity ได้ จะต้องอาศัย Propertyใด Property หนึ่งของ Regular Entity มาประกอบกับ Property ของตัวมันเอง เช่น Entity "TIME_IN_OUT" ซึ่งสมาชิกของ Entity นี้ได้แก่ เวลาเข้าออกของพนักงานแต่ละคนในแต่ละวัน ซึ่งประกอบไปด้วย Property ต่าง ๆ ดังนี้

ตารางที่ 2.13 ตัวอย่าง Weak Entity

Entity " TIME_IN_OUT

EmpID	Date	Time_In	Time_Out
0001	15/9/41	7.30	17.13
0002	15/9/41	8.00	18.00
0003	15/9/41	7.45	17.49

0001	16/9/41	8.00	18.00
0002	16/9/41	7.45	17.49
0003	16/9/41	7.30	17.13

จากตัวอย่างข้อมูลของ Entity "TIME_IN_OUT" จะสังเกตเห็นว่า แต่ละสมาชิกของ Entity "TIME _ IN_OUT" จะสามารถมีคุณสมบัติ Identity ได้จะต้องอาศัยค่าของ Property "EmpID" ซึ่งเป็น Property ของ Entity "TIME_IN_OUT" มาประกอบกับ Property "Date" ซึ่งเป็น Property ของ ตนเอง จึงทำให้ แต่ละสมาชิกของ Entity "TIME_IN_OUT" มีคุณสมบัติ Identity สำหรับรูปภาพที่ใช้แทน Entity ประเภทนี้ ได้แก่ รูปสี่เหลี่ยมผืนผ้า 2 รูปซ้อนกัน โดยมีชื่อของ Entity นั้นอยู่ภายใน ดังรูป

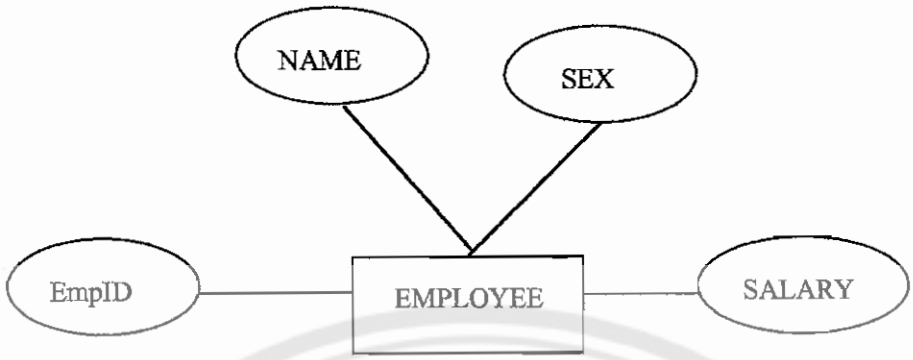
รูปที่ 2.9 Weak Entity

ข้อสังเกตอย่างหนึ่งสำหรับ Weak Entity คือ ถ้ามีการลบสมาชิกใดสมาชิกหนึ่งใน Regular Entity ที่มีความสัมพันธ์กับ Weak Entity ที่ใช้ค่าของ Property ของสมาชิกใน Regular Entity ที่ถูกลบจะต้องถูกลบทิ้งตามไปด้วย เพื่อรักษาคุณสมบัติ Identity ไว้

2.7.3.2 Property

ได้แก่ Property ต่าง ๆ ของ Entity หรือ Relationship ที่นิยามไว้ใน Semantic Model เช่น property "EmpID", "NAME", "SEX" และ "SALARY" ของ Entity "EMPLOYEE" เป็นต้นสำหรับ Property ใน E-R Model จะสามารถแบ่งย่อยได้ดังนี้

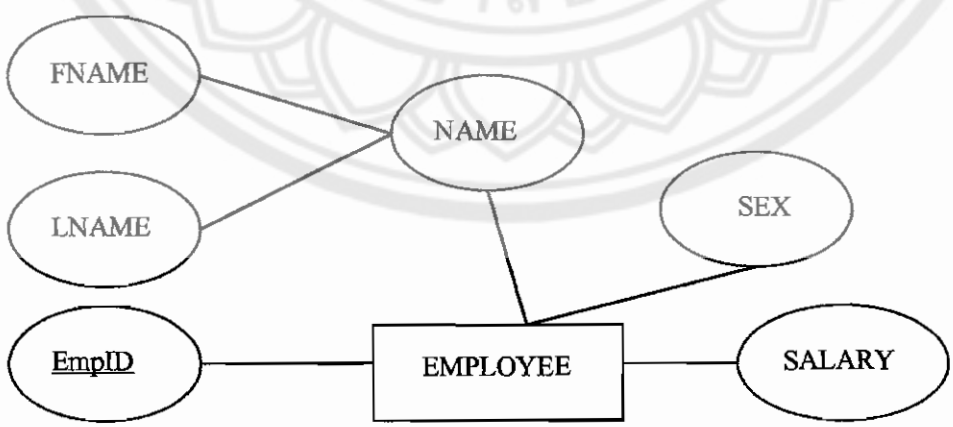
1) Simple Property ได้แก่ Property ที่ค่าภายใน Property นั้น ไม่สามารถแบ่งย่อยได้อีก ซึ่งได้แก่ Property ของ Entity โดยทั่วไป เช่น Property " SEX" ของ Entity "EMPLOYEE" ที่ใช้เก็บเพศของพนักงานแต่ละคน หรือ Property "SALARY" ของ Entity "EMPLOYEE" ที่ใช้เก็บเงินเดือนพนักงานแต่ละคน ซึ่งค่าของทั้ง 2 Property นี้ ไม่สามารถแบ่งออกเป็นค่าย่อยได้อีกสำหรับรูปภาพที่ใช้แทน Property ประเภทนี้ วงรีที่มีเส้นเชื่อมต่อไปยัง Entity ที่เป็นเจ้าของ Property นั้น โดยมีชื่อของ Property นั้นอยู่ภายใน เช่น Property ต่าง ๆ ของ Entity "ENPLOYEE" ดังรูป



รูปที่ 2.10 Weak Entity

2) Composite Property เป็น Property ที่มีลักษณะตรงข้ามกับ Simple Property กล่าวคือ จะเป็น Property ที่ค่าภายใน Property นั้น ยังสามารถแยกเป็น Property ย่อยได้อีก เช่น Property "NAME" ของ Entity "EMPLOYEE" ที่สามารถแบ่งย่อยออกเป็น Property "FNAME" ที่ใช้เก็บชื่อและ Property " SNAME" ที่ใช้เก็บนามสกุล เป็นต้น สำหรับรูปภาพที่ใช้แทน Property ประเภทนี้จะใช้วงรีเดียวกับ Simple Property แต่จะเป็นวงรีที่เชื่อมกับวงรีของ Simple Property ที่เป็นเจ้าของ Composite Property นั้น เช่น Property "FNAME" และ "SNAME" ของ Entity "EMPLOYEE"

3) Key เป็น Property หรือกลุ่มของ Property ที่มีค่าในแต่ละสมาชิกของ Entity ไม่ซ้ำกันซึ่งถูกนำมาใช้กำหนดคุณสมบัติ Identity ให้กับ Entity เช่น Property "EmpID" ของ Entity "EMPLOYEE" ซึ่งใช้แทนรหัสประจำตัวพนักงาน สำหรับรูปภาพที่ใช้แทน Key ของ Entityจำใช้รูปวงรีเช่นเดียวกับ Property แต่จะมีเส้นขีดอยู่ใต้ Property ที่เป็น Key ดังรูป



รูปที่ 2.11 Key

4) Single-valued Property เป็น Property ที่มีค่าของข้อมูลภายใต้ค่าของ Property ใด Property เพียงค่าเดียว เช่น Property "SALARY" ที่ใช้เก็บเงินเดือนของพนักงาน ซึ่งพนักงานแต่ละคนสามารถมีเงินเดือนได้เพียงค่าเดียว ดังตัวอย่างข้อมูลต่อไปนี้

ตารางที่ 2.14 ตัวอย่าง Single-valued Property

EmpID	NAME	SEX	SALARY
0001	สมชาย นิลกลัด	M	8500
0002	สมถวิล กลั่นเจริญ	F	9000
0003	เจริญ ก้าวหน้า	M	12000
0004	ชุติมา สกุลดี	F	10000
0005	นิวัติ เหล่าสุวรรณ	M	25000

5) Multi-valued Property เป็น Property ที่มีลักษณะตรงข้ามกับ Property แบบ Single valued กล่าวคือ เป็น Property ที่มีค่าของข้อมูลได้หลายค่าภายใต้ค่าของ Property ใด Property หนึ่ง เช่น Property "DEGREE" ที่ใช้ระบุระดับการศึกษาของพนักงานแต่ละคน ซึ่งพนักงานแต่ละคน จะมีระดับการศึกษาได้หลายระดับ ดังตัวอย่างข้อมูลต่อไปนี้

ตารางที่ 2.15 ตัวอย่าง Multi-valued Property

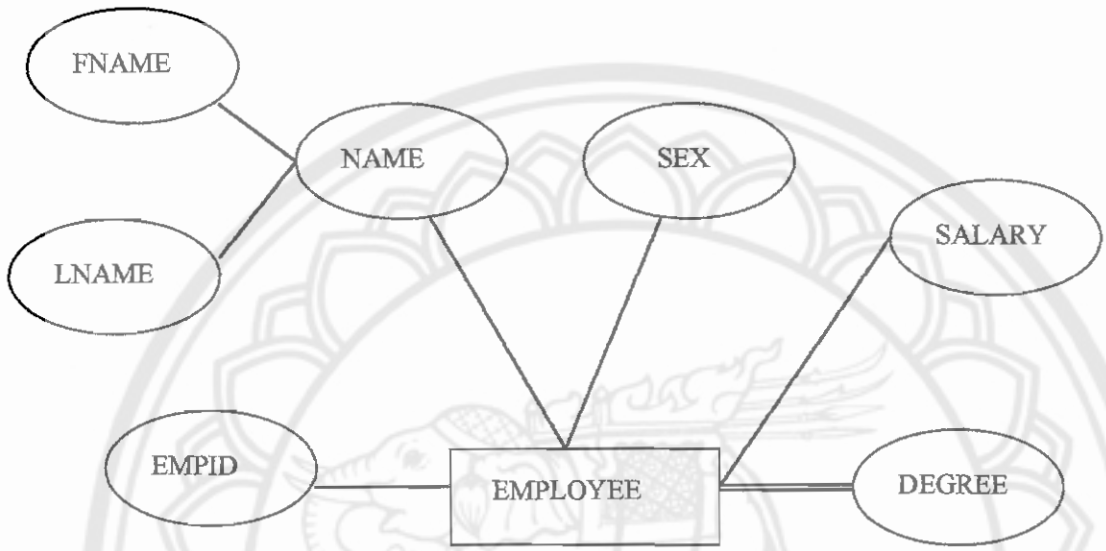
Entity "EMPLOYEE"

EmpID	NAMe	SEX	SALARY	DEGREE	SCHOLL/UNIVERSITY
0001	สมชาย นิลกลัด	M	8500	0	ร.ร.ขอนแก่นวิทยา
0002	สมถวิล กลั่นเจริญ	F	9000	0	ร.ร.อ่างทองวิทยา
0003	เจริญ ก้าวหน้า	M	12000	0	ร.ร. วัดสุทธิวราราม
				1	ม. กรุงเทพฯ
0004	ชุติมา สกุลดี	F	10000	0	ร.ร. นรีเวชวิทยา
0005	นิวัติ เหล่าสุวรรณ	M	25000	0	ร.ร. กรุงเทพคริสเตียน
				1	ม. ธรรมศาสตร์

จากตัวอย่างข้อมูล จะสังเกตเห็นว่า Property "DEGREE" ภายใต้พนักงานแต่ละคนสามารถมีค่าของข้อมูลได้มากกว่า 1 ค่า เช่น Property " DEGREE" ของพนักงานที่ชื่อ "เจริญ ก้าวหน้า" จะมีอยู่

ด้วยกัน 2 ค่า คือ 0 และ 1 เนื่องจากพนักงานคนนี้ จบการศึกษาระดับปริญญาตรี จึงมีข้อมูลทั้งระดับมัธยมศึกษา และปริญญาตรี หรือ Property "DEGREE" ของพนักงานที่ชื่อ "นิวัติ เหล่าสุวรรณ" จะมีอยู่ด้วยกัน 3 ค่า คือ 0,1 และ 2 เนื่องจากพนักงานคนนี้จบการศึกษาระดับปริญญาโท เป็นต้น

สำหรับรูปภาพที่ใช้แทน Property ประเภทนี้ จะใช้รูปภาพของ Property ประเภทนี้กับรูปภาพของ Entity หรือ Relation จะใช้เส้น 2 เส้นแทน ดังรูป



รูปที่ 2.12 Multi-valued Property

6) Derived Property เป็น Property ที่ค่าของข้อมูลได้มาจากการนำเอาค่าของ Property อื่นมาคำนวณ ซึ่งค่าของ Property ประเภทนี้ จะต้องเปลี่ยนแปลงทุกครั้ง เมื่อมีการเปลี่ยนแปลงค่าของ Property ที่ ถูกนำมาคำนวณ เช่น Property "TOT_SAL" ของ Entity "EMPLOYEE" ตามตัวอย่างข้อมูลดังนี้

ตารางที่ 2.16 ตัวอย่าง Derived Property

Entity "EMPLOYEE'

EmpID	NAME	SEX	SALARY	DEGREE	TOT_SAL
0001	สมชาย นิลกลัด	M	8500	0	25500
0002	สมถวิล กลั่นเจริญ	F	9000	0	27000
0003	เจริญ ก้าวหน้า	M	12000	1	36000
0004	ชุติมา สกุลดี	F	10000	0	30000
0005	นิวัติ เหล่าสุวรรณ	M	25000	2	75000

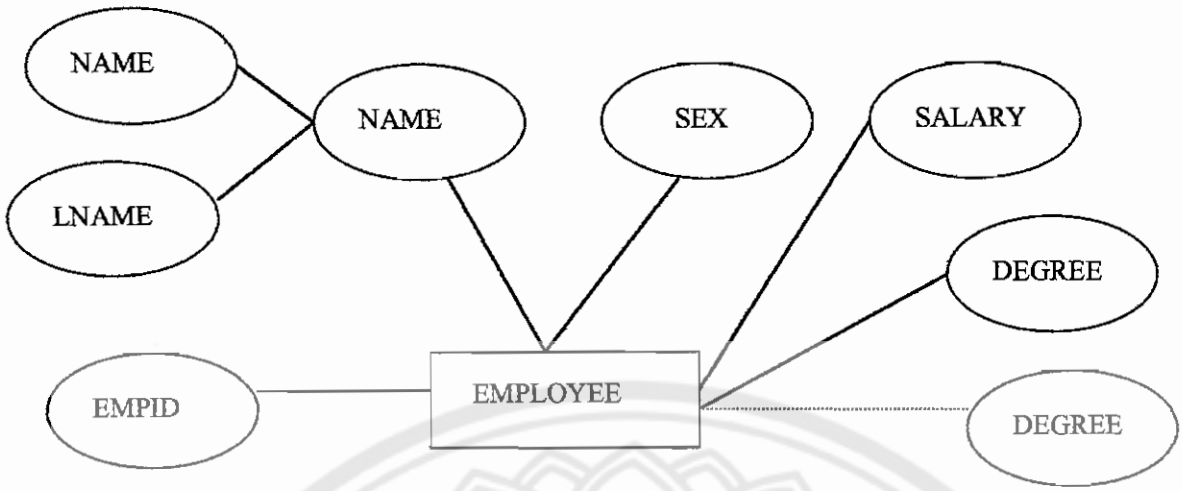
Entity "MTHLY_SALARY"

EmpID	MONTH	INCOME
0001	1	8500
0002	1	9000
0003	1	12000
0004	1	10000

EmpID	MONTH	INCOME
0005	1	25000
0001	2	8500
0002	2	9000
0003	2	12000
0004	2	10000
0005	2	25000
0001	3	8500
0002	3	9000
0003	3	12000
0004	3	10000
0005	3	25000

จากตัวอย่างข้อมูล จะเห็นว่า ค่าใน Property "TOT_SAL" ใน Entity "EMPLOYEE" ซึ่งใช้เก็บเงินเดือนสะสมของพนักงานแต่ละคนเพื่อนำไปคำนวณภาษี ได้มาจาก ผลรวมของค่าใน Property "INCOME" ของ Entity " MTHLY_SALARY" ซึ่งเป็นเงินเดือนที่พนักงานแต่ละคนได้รับในแต่ละเดือน

สำหรับรูปภาพที่ใช้แทน Property ประเภทนี้ จะใช้รูปภาพเดียวกัน Simple Property แต่เส้นที่ใช้เชื่อมระหว่างรูปภาพของ Property ประเภทนี้กับรูปภาพของ Entity หรือ Relationship จะใช้เส้นประแทน ดังรูป



รูปที่ 2.13 Derived Property

2.7.3.3 Relationship

ได้แก่ Relationship ที่นิยามไว้ใน Semantic Model สำหรับรูปภาพที่ใช้แทน Relationship ใน E-R Model จะได้แก่ รูปสี่เหลี่ยมข้ามหลามตัด ที่มีชื่อเสียงของ Relationship นั้นอยู่ภายใน สำหรับรูปภาพของ Relationship นี้ จะไม่สามารถปรากฏอยู่เดี่ยว ๆ ได้ แต่จะต้องปรากฏอยู่คู่กับ Entity เสมอ

Relationship โดยทั่วไปจะกำหนดขึ้นจาก Entity ที่มี Property ร่วมกัน เช่น Entity "EMPLOYEE" และ "Department" ซึ่งสามารถแสดงด้วยตัวอย่างข้อมูลได้ดังนี้

ตารางที่ 2.17 ตัวอย่าง Relationship

Entity "EMPLOYEE"

EmpID	NAME	SEX	SALARY	DEP_ID
0001	สมชาย นิลกลัด	M	8500	01
0002	สมถวิล กลั่นเจริญ	F	9000	03
0003	เจริญ ก้าวหน้า	M	12000	01
0004	ชุติมา สกุลดี	F	10000	02
0005	**นิวัติ เหล่าสุวรรณ**	**M**	**25000**	**02**

ตารางที่ 2.17 (ต่อ) ตัวอย่าง Relationship

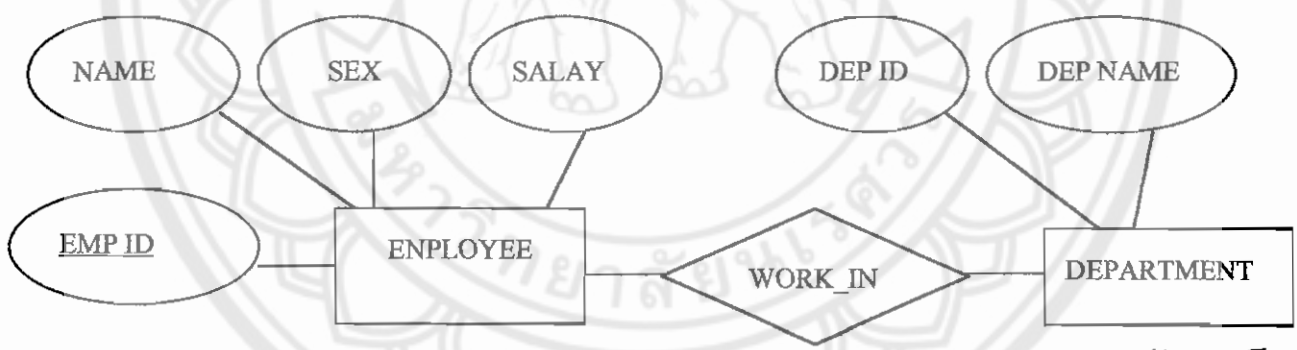
Entity "Department"

DEP_ID	DEP_NAME
01	ธุรการ
02	บุคคล
03	การเงิน

Relation "WORK_IN'

EmpID	NAME	SEX	SALARY	DEP_ID	DEP_NAME
0001	สมชาย นิลกลัด	M	8500	01	ธุรการ
0002	สมถวิล กลั่นเจริญ	F	9000	03	การเงิน
0003	เจริญ ก้าวหน้า	M	12000	01	ธุรการ
0004	ชุติมา สกุลดี	F	10000	02	บุคคล
0005	นิวัติ เหล่าสุวรรณ	M	25000	02	บุคคล

และสามารถแสดงด้วยรูปภาพได้ดังนี้



รูปที่ **2.14** Relationship

2.7.3.4 Cardinality Ratio

สมาชิกใน Entity ที่เกี่ยวข้องกับ Relation นั้นและจะถูกนำไปใช้กำหนดประเภทของ Relationship ที่เรียกว่า "Cardinality Ratio"

1. One-to-One Relationship เป็น Relationship ที่แต่ละ Participant ของ Entity หนึ่ง จะมีความสัมพันธ์กับอีก Participant ของอีก Entity หนึ่งเพียง Participant เดียว ดังตัวอย่างข้อมูลต่อไปนี้

ตารางที่ 2.18 ตัวอย่าง Relationaship แบบ One-to-On

Entity "CUSTOMER"

NAME	ADDERSS	ACCT_NO
แพง พลเมืองดี	111 บางพลัด กทม.	1111111111
จิราพร สมตน	222 บางซื่อ กทม.	2222222222
สุภาพร อุดมศิลป์	333 ปทุมวัน กทม.	3333333333
กิตติ มั่นคง	444 บางบอน กทม.	4444444444

Entity "ACCOUNT"

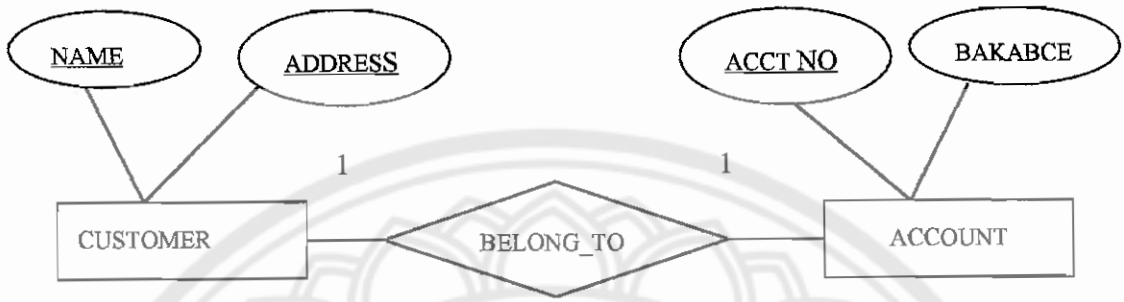
ACCT_NO	BALANCE
111111111	54000
222222222	12000
333333333	14000
444444444	100000

Relationship "BELONG"

NAME	ADDERSS	ACCT_NO	BALANCE
แพง พลเมืองดี	111 บางพลัด กทม.	111111111	54000
จิราพร สมตน	222 บางซื่อ กทม.	222222222	12000
สุภาพร อุดมศิลป์	333 ปทุมวัน กทม.	333333333	14000
กิตติ มั่นคง	444 บางบอน กทม.	444444444	100000

จากตัวอย่างข้อมูลของ Relationship "BELONG_TO" จะสังเกตเห็นว่า ลูกค้า ซึ่งได้แก่ แต่ละรายการใน "Entity "CUSTOMER" จะมีความสัมพันธ์กับรายการใน Entity "ACCOUNT" ได้เพียงรายการเดียว และเพียงรายการเดียว และในมุมกลับกัน แต่ละรายการใน Entity "ACCOUNT" ก็จะมีความสัมพันธ์กับรายการใน Entity "CUSTOMER" ได้เพียงรายการเดียวเช่นเดียวกัน

สำหรับรูปภาพที่ใช้แทนความสัมพันธ์ในลักษณะนี้ จะใช้รูปภาพเดียวกัน Relationship โดยทั่วไป ดังนั้น รูปภาพของความสัมพันธ์ จึงมีลักษณะดังรูป



รูปที่ 2.15 Relationship แบบ One-to –One

2 One-to-Many Relationship เป็น Relationship ที่แต่ละ Participant ของ Entity หนึ่ง มีความสัมพันธ์กับอีก Participant ของ Entity หนึ่งมากกว่า 1 Participant เช่น กรณีลูกค้าสามารถมีเงินฝากได้มากกว่า 1 บัญชี และแต่ละบัญชีเงินฝากจะต้องมีเจ้าของบัญชีเพียงคนเดียว ดังตัวอย่างข้อมูลต่อไปนี้

ตารางที่ 2.19 ตัวอย่าง Relationship แบบ One-to –Many

Entity "CUSTOMER"

NAME	ADDERSS	ACCT_NO
แพง พลเมืองดี	111 บางพลัด กทม.	111111111
จิราพร สมตน	222 บางซื่อ กทม.	222222222
สุภาพร อุดมศิลป์	333 ปทุมวัน กทม.	333333333
กิตติ มั่นคง	444 บางบอน กทม.	444444444

Entity "ACCOUNT"

ACCT_NO	BALANCE
111111111	54000
222222222	12000
333333333	14000
444444444	4000

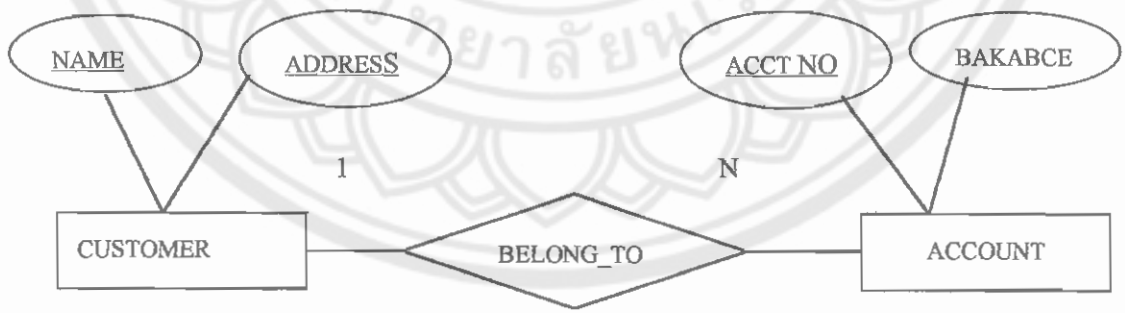
ตารางที่ 2.19 (ต่อ) ตัวอย่าง Relationship แบบ One-to-Many

Relationship "BELONG_TO"

NAME	ADDERSS	ACCT_NO	BALANCE
แพง พลเมืองดี	111 บางพลัด กทม.	111111111	54000
		111111112	4000
จิราพร สมตน	222 บางซื่อ กทม.	222222222	12000
สุภาพร อุดมศิลป์	333 ปทุมวัน กทม.	333333333	14000
กิตติ มั่นคง	444 บางบอน กทม.	444444444	100000

จากตัวอย่างข้อมูลของ Relationship "BELONG_To" จะสังเกตเห็นว่า ลูกค้าที่ชื่อ "แพง พลเมืองดี" เป็นเจ้าของบัญชีเงินฝาก 2 บัญชี คือ บัญชีเงินฝากเลขที่ "111111111" และ "111111112" แต่ในมุมกลับกัน แต่ละบัญชี จะมีเจ้าของได้เพียงคนเดียว

สำหรับสัญญลักษณ์ที่ใช้กับ Relationship ประเภทนี้ ได้แก่ ตัวเลข 1 และตัวอักษร M โดยตัวเลข 1 จะถูกกำหนดไว้ทางด้านของ Entity ที่มีจำนวน Participant ที่เกี่ยวข้องกับ Relationship เพียง Participant เดียว ส่วนตัวอักษร M จะถูกกำหนดไว้ทางด้านของ Entity ที่มีจำนวน Participant ที่เกี่ยวข้องกับ Relationship มากกว่า 1 Participant ซึ่งจากรูป ด้าน Participant เดียว ส่วนด้าน Entity "ACCOUNT" จะเป็น Entity ที่มีจำนวน Participant ที่เกี่ยวข้องกับ Relationship มากกว่า 1 Participant ดังนั้น จึงปรากฏตัวเลข 1 ไว้ทางด้าน Entity "CUSTOMER" และตัวอักษร M ทางด้าน Entity " ACCOUNT" ดังรูป



รูปที่ 2.16 Relationship แบบ One-to –Many

3) Many-to-Many Relationship เป็น Relationship ที่ Participant มากกว่าว 1 Participant ของ Entity หนึ่ง มีความสัมพันธ์กับอีก Participant ของอีก Entity หนึ่งมากกว่า 1

Participant เช่น กรณีลูกค้าสามารถมีเงินฝากได้มากกว่า 1 บัญชี และแต่ละบัญชีเงินฝากสามารถมีเจ้าของบัญชีได้มากกว่า 1 คน ดังตัวอย่างข้อมูลต่อไปนี้

ตารางที่ 2.20 ตัวอย่าง Relationship แบบ Many-to-Many

Entity "CUSTOMER"

NAME	ADDERSS	ACCT_NO
แพง พลเมืองดี	111 บางพลัด กทม.	111111111
แพง พลเมืองดี	111 บางพลัด กทม.	111111112
จิราพร สมตน	222 บางซื่อ กทม.	222222222
สุภาพร อุดมศิลป์	333 ปทุมวัน กทม.	333333333
เอกสิทธิ์ ทับทอง	444 คลองตัน กทม.	111111112

Entity "ACCOUNT"

ACCT_NO	BALANCE
111111111	54000
222222222	12000
333333333	14000
444444444	100000
111111112	4000

Relationship "BELONG_TO"

NAME	ADDERSS	ACCT_NO	BALANCE
แพง พลเมืองดี	111 บางพลัด กทม.	111111111	54000
		111111112	4000
จิราพร สมตน	222 บางซื่อ กทม.	222222222	12000
สุภาพร อุดมศิลป์	333 ปทุมวัน กทม.	333333333	14000
กิตติ มั่นคง	444 บางบอน กทม.	444444444	100000
เอกสิทธิ์ ทับทอง	444 คลองตัน กทม.	111111112	4000

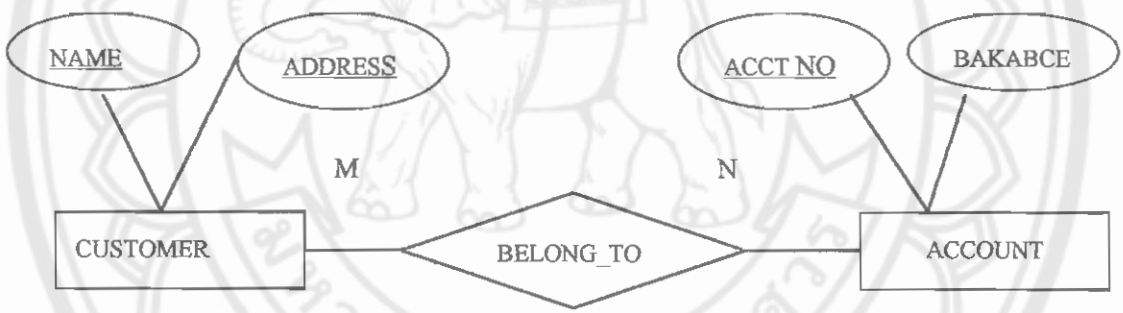
และในมุมกลับกัน เมื่อพิจารณาจากทางด้านบัญชีเงินฝาก จะสังเกตเห็นว่า บัญชีเลขที่ "111111112" มีเจ้าของบัญชี 2 คน คือ "แพง พลเมืองดี" และ "เอกสิทธิ์ ทับทอง" ซึ่งสามารถแสดงด้วย Relationship "BELONG_TO" ได้เช่นเดียวกัน ดังนี้

ตารางที่ 2.20 (ต่อ) ตัวอย่าง Relationship แบบ Many-to-Many

Relationship "BELONG_TO"

ACCT_NO	NAME	ADDERSS	BALANCE
111111112	แพง พลเมืองดี	111 บางพลัด กทม.	4000
	เอกสิทธิ์ ทับทอง	444 คลองตัน กทม.	4000
111111111	แพง พลเมืองดี	111 บางพลัด กทม.	54000
222222222	จิราพร สมตน	222 บางซื่อ กทม.	12000
333333333	สุภาพร อุดมศิลป์	333 ปทุมวัน กทม.	14000
444444444	กิตติ มั่นคง	444 บางบอน กทม.	100000

ดังนั้น ความสัมพันธ์ระหว่างลูกค้าและบัญชีเงินฝากนี้ จึงเป็น Many-to-Many Relationship สำหรับสัญลักษณ์ ที่ใช้กับ Relationship ประเภทนี้ ได้แก่ ตัวอักษร M โดยจะกำหนดไว้ทั้ง 2 ด้าน ของ Entity ดังรูป



รูปที่ 2.17 Relationship แบบ Many – TO Many

2.7.3.5 คุณสมบัติของแผนภาพ E-R ที่ดี

เนื่องจากแผนภาพ E-R ถูกใช้เป็นเครื่องมือในการนำเสนอความเป็นจริงที่เกี่ยวข้องกับข้อมูล ดังนั้นจึงควรที่จะต้องมีคุณสมบัติดังนี้

1 Expressiveness

แผนภาพ E-R ที่ดีจะต้องอธิบายโครงสร้างของข้อมูลได้เป็นอย่างดีและครบถ้วน

2 Simplicity

แผนภาพ E-R ที่ดี จะต้องมีรูปแบบที่ง่ายต่อการเข้าใจ

3 Minimality

แผนภาพ E-R ที่ดี จะต้องมีความชัดเจน และไม่สามารถตีความเป็นอย่างอื่น

4 Formality

แผนภาพ E-R ที่ดี จะต้องไม่ซ้ำซ้อน และมีรูปแบบที่เป็นมาตรฐาน

2.7.3.6 การค้นหา Entity

สิ่งที่จำเป็นสำหรับการวาดแผนภาพ E-R ได้แก่ การกำหนด Entity, Property, Relationship, และ Cardinality ที่ใช้แทนข้อมูลต่าง ๆ ภายในฐานข้อมูลที่ต้องการออกแบบนั้น ซึ่งมีขั้นตอนที่ค่อนข้างซับซ้อน แต่อย่างไรก็ตามผู้ออกแบบสามารถที่จะกำหนด Entity ขึ้นภายในระบบได้อย่างคร่าว ๆ โดยใช้วิธีการค้นหา Entity ด้วยวิธีของ Data Perspective ซึ่งกำหนดไว้ว่า Entity จะถูกกำหนดขึ้นจากสิ่งต่าง ๆ ที่ปรากฏอยู่ในความต้องการของผู้ใช้ดังต่อไปนี้

1. ขั้นตอนการทำงาน เอกสาร และรายงานต่าง ๆ ที่ปรากฏอยู่ในระบบ
2. อุปกรณ์ต่าง ๆ ที่ระบบใช้ในการติดต่อ
3. ระบบงานที่เกี่ยวข้องในการส่งผ่านข้อมูลระหว่างระบบกัน
4. เหตุการณ์ที่เกิดขึ้นในระบบจะต้องมีการบันทึกไว้เป็นหลักฐาน
5. บุคคลหรือสิ่งที่มีบทบาทต่อระบบงาน
6. สถานที่ที่เกี่ยวข้องกับระบบงาน
7. หน่วยงานต่าง ๆ ในองค์กรที่เกี่ยวข้องกับระบบงาน

2.8 การทำ Normalization [1]

การออกแบบฐานข้อมูลด้วย E-R Model มีจุดมุ่งหมายเพื่อนำเสนอข้อเท็จจริงต่าง ๆ ที่เกี่ยวข้องกับข้อมูล โดยไม่ได้คำนึงว่า ฐานข้อมูลที่ออกมานั้น จะมีปัญหาทางด้านความซ้ำซ้อนของข้อมูล ความถูกต้องของข้อมูล และความผิดพลาดในการเพิ่มลบ และแก้ไขข้อมูล หรือไม่ ดังนั้น จึงต้องมีวิธีการตรวจสอบ และแก้ไขปัญหาต่าง ๆ เหล่านี้ วิธีการดังกล่าว ได้แก่ "การทำ Normalization"

2.8.1 Normalization

เป็นวิธีการที่ใช้ในการตรวจสอบ และแก้ไขปัญหาทางด้านความซ้ำซ้อนของข้อมูล โดยการดำเนินการให้ข้อมูลในแต่ละ Relation อยู่ในรูปที่เป็นหน่วยเล็กที่สุดที่ไม่สามารถแตกออกเป็นหน่วยย่อย ๆ ได้อีก โดยยังคงความสัมพันธ์ระหว่างข้อมูลใน Relation ต่าง ๆ ไว้ตามหลักการที่กำหนดไว้ใน Ralation Medel

การทำ Normalization นี้ เป็นการดำเนินงานอย่างเป็นลำดับ ที่กำหนดไว้ด้วยกันเป็นขั้นตอนตามปัญหาที่เกิดขึ้นในขั้นตอนนั้น ๆ ซึ่งแต่ละขั้นตอนจะมีชื่อตามโครงสร้างข้อมูลที่กำหนดไว้ ดังนี้

1. ขั้นตอนการทำ First Normal Form (1NF)
2. ขั้นตอนการทำ Second Normal Form (2NF)
3. ขั้นตอนการทำ Third Normal Form(3NF)
4. ขั้นตอนการทำ Boyce-Codd Normal Form (BCNF)

5. ขั้นตอนการทำ Fourth Normal Form (4NF)

ในแต่ละขั้นตอนของการทำ Normalization จะมีการระบุรูปแบบของโครงสร้างของข้อมูลที่ควรจะเป็นเรียกว่า Normal Form ไว้ ซึ่งโครงสร้างเหล่านี้จะสามารถแก้ไขปัญหาที่เกิดขึ้นในโครงสร้างที่ระบุนี้ จะสามารถแก้ไขปัญหาที่เกิดขึ้นในโครงสร้างข้อมูลของขั้นตอนก่อนหน้าได้หรือกล่าวอีกนัยหนึ่ง การทำ Normalization ในขั้นตอน จะต้องอาศัยผลที่ได้จากการทำ Normalization ในขั้นตอนก่อนหน้า มาปรับปรุงเพื่อให้มีโครงสร้างเป็นไปตามโครงสร้างที่กำหนดไว้ในขั้นนั้น ๆ แต่อย่างไรก็ตาม ในการทำ Normalizatio ไม่จำเป็นต้องเริ่มจากขั้นตอนการทำ First Normal Form และสิ้นสุดในขั้นตอนการทำ Fourth Normal Form เสมอไป กล่าวคือ การทำ Normalization จะพิจารณาโครงสร้างของข้อมูลที่นำมาทำ Normalization นั้น ว่าจัดอยู่ในโครงสร้างข้อมูลของขั้นตอนใด แล้วจึงเริ่มทำ Normalization จากขั้นตอนนั้นเป็นต้นไป และเช่นเดียวกัน ในการพิจารณาว่า สิ้นสุดที่ขั้นตอนใด จะขึ้นอยู่กับว่า โครงสร้างข้อมูลที่ได้นั้น มีความถูกต้องตามความหมายของข้อมูลที่กำหนดไว้แล้วหรือไม่ ถ้าผลที่ได้จากการทำ Normalization ในขั้นตอนใด ส่งผลให้โครงสร้างของข้อมูลมีความหมายตามที่กำหนดไว้ การทำ Normalization ก็จะสิ้นสุดที่ขั้นตอนนั้น

2.8.2 Functional Dependency (FDs)

การพิจารณาโครงสร้างของแต่ละ Relation ว่ามีโครงสร้างอยู่ใน Normal Form ระดับใด จะพิจารณาจาก Functional Dependency ซึ่งเป็นความสัมพันธ์ระหว่าง Attribute ต่าง ๆ ภายใน Relation กับ Attribute หรือกลุ่ม Attribute ที่ทำหน้าที่เป็น Key ของ Relation นั้น ซึ่งความ สัมพันธ์นี้ จะถูกนิยามด้วย รูปแบบทางคณิตศาสตร์ ที่เรียกว่า "Functional Dependency" มีรูปแบบดังนี้

FDs : determinant-attribute → dependency-attribute

โดยที่ determinant-attribute หมายถึง Attribute ที่เมื่อถูกระบุว่าด้วยค่าใดค่าหนึ่งแล้ว จะสามารถแสดงค่าของ dependency-attribute ซึ่งเป็น Attribute ที่มีความสัมพันธ์กับ determinant attribute นั้นออกมาเช่น ตารางแสดงความสัมพันธ์ระหว่างหมายเลขบัตรประชาชนและชื่อเจ้าของบัตรต่อไปนี้

ตารางที่ 2.21 ตัวอย่าง Functional Dependency

PERSON_ID	PERSON_NAME
13045678927	สมเกียรติ
13024893450	น้ำผึ้ง
12365498701	กาญจนา
12345678910	กาญจนา

จากตารางจะสังเกตเห็นว่า เมื่อมีการระบุหมายเลขบัตรประชาชน (Attribute "PERSON_ID") ด้วยหมายเลขใดหมายเลขหนึ่งจะสามารถทราบถึงชื่อของเจ้าของบัตรประชาชน(Attribute"PERSON_NAME") ตามหมายเลขที่ระบุนั้นได้ แต่ในทางกลับกัน เมื่อมีการระบุชื่อเจ้าของบัตรประชาชน กลับไม่สามารถทราบถึงหมายเลขบัตรประชาชนตามที่ระบุนั้นได้ทุกกรณี เนื่องจากมีชื่อเจ้าของบัตรประชาชนที่ซ้ำกัน จึงกล่าวได้ว่าหมายเลขบัตรประชาชน (Attribute " PERSON_ID") เป็น determinant-attribute และชื่อเจ้าของบัตรประชาชน (Attribute "PERSON_NAME") เป็น dependency-attribute จึงสามารถนำความสัมพันธ์นี้มาแสดงในรูป Functional Dependency ได้ดังนี้

FDs : PERSON_ID → PERSON_NAME

ซึ่งสามารถแสดงด้วยแผนภาพได้ดังนี้

รูปที่ 2.18 Relationship แบบ Many-to-Many

สำหรับ Attribute ที่ปรากฏในส่วน determinant-attribute ของ Functional Dependency นี้ จะเรียกว่า "Determinant" สำหรับ Attribute ที่ปรากฏในส่วน dependency-attribute จะเรียกว่า "Dependency"

2.8.3 ประเภทของ Functional Dependency

Functional Dependency สามารถแบ่งออกได้เป็น 4 ประเภทได้ดังนี้

1. Functional Dependency ที่เกิดจาก Determinant และ Dependency อย่างละ 1 ค่า เช่น ความสัมพันธ์ระหว่างหมายเลขบัตรประชาชน และชื่อเจ้าของบัตร ดังตารางต่อไปนี้

ตารางที่ 2.22 ตัวอย่าง Functional Dependency ประเภทที่ 1

PERSON_ID	PERSON_NAME
I3045678927	สมเกียรติ
13024893450	น้ำผึ้ง
12365498701	กาญจนา
12345678910	กาญจนา

จากตาราง จะสังเกตเห็นว่า

เมื่อมีการระบุหมายเลขบัตรประชาชน (Attribute "PERSON_ID") ด้วยหมายเลขใดหมายเลขหนึ่ง จะสามารถทราบถึงชื่อเจ้าของบัตรประชาชน (Attribute "PERSON_ NAME") ตามหมายเลขที่ระบุนั้นได้ ซึ่งสามารถแสดงในรูป Functional Dependency ได้ดังนี้

FDs : PERSON_ID ➔ PERSON_NAME

2. Functional Dependency ที่เกิดขึ้นจากความสัมพันธ์ระหว่าง Determinant 1 ค่ากับ Dependency หลายค่า เช่น ความสัมพันธ์ระหว่างหมายเลขบัตรประชาชน และข้อมูลที่ปรากฏอยู่บนบัตรประชาชน ดังตารางต่อไปนี้

ตารางที่ 2.23 ตัวอย่าง Functional Dependency ประเภทที่ 2

PERSON_ID	FNAME	LNAME	ADDRESS	BIRTY_DATE	ISSUE_DATE
13045678927	สมเกียรติ	ชื่นยศ	123 บ้านโป่ง	3/4/2511	10/4/2542
13024893450	น้ำผึ้ง	สู่สุข	39/8 กทม.	3/4/2512	11/12/2540
12365498701	กาญจนา	หิรัญรัตน์	456 พิจิตร	5/6/2510	9/10/2541
21222423624	เอก	ชื่นยศ	258 เยาวราช	5/6/2510	11/12/2540
12345678910	กาญจนา	ยิ้มเจริญ	39/8 กทม.	5/6/2509	10/10/2539

จากตาราง จะสังเกตเห็นว่า

- เมื่อมีการระบุค่าหมายเลขบัตรประชาชน ด้วยหมายเลขใดหมายเลขหนึ่ง จะสามารถทราบถึงชื่อ นามสกุล ที่อยู่ วันเกิด และวันที่ทำบัตร ของเข้าของบัตรตามหมายเลขที่ระบุนั้น
- เมื่อมีการระบุชื่อเจ้าของบัตรประชาชนด้วยชื่อใดชื่อหนึ่ง จะไม่สามารถระบุข้อมูลอื่นๆ ที่ไม่ปรากฏอยู่บนบัตรประชาชนตามชื่อของเจ้าของบัตรประชาชนที่ระบุนั้นได้ เนื่องจากมีเจ้าของบัตรประชาชนที่มีชื่อซ้ำกัน
- เมื่อมีการระบุนามสกุลของเจ้าของบัตรประชาชนด้วยนามสกุลใดนามสกุลหนึ่งจะไม่สามารถทราบถึงข้อมูลอื่นๆ ที่ปรากฏอยู่บนบัตรประชาชนตามนามสกุลที่ระบุนั้นได้ เนื่องจากมีเจ้าของบัตรประชาชนที่มีนามสกุลซ้ำกัน
- เมื่อมีการระบุที่อยู่ หรือวันเกิด หรือวันที่ทำบัตรของเจ้าของบัตรประชาชน จะได้ผลเช่นเดียวกับระบุด้วยชื่อ หรือนามสกุล กล่าวคือ มีโอกาสที่จะปรากฏข้อมูลที่ซ้ำกันเช่นเดียวกัน

จากที่กล่าวมาเหล่านี้ จึงกล่าวได้ว่า หมายเลขบัตรประชาชน (Attribute "PERSON_ID") เป็น Attribute ที่ทำหน้าที่เป็น Dependency ซึ่งสามารถแสดงความสัมพันธ์ได้ดังนี้

PERSON_ID → FNAME, LNAME,ADDRESS,BIRTH+DATE,ISSUE_DATE

3. Functional Dependency ที่มีความสัมพันธ์ 2 ทาง ซึ่งเป็น Functional Dependency ที่ทั้ง Determinant และ Dependency ต่างทำหน้าที่ของอีกฝ่ายหนึ่งได้ เช่น ความสัมพันธ์ระหว่างชื่อของผู้จัดการโครงการกับชื่อ โครงการ(Attribute"MANAGER")กับชื่อ โครงการ(Attribute"PROJECT_NO") ที่ซึ่งเมื่อเราทราบถึงชื่อผู้จัดการโครงการจะสามารถทราบถึงชื่อโครงการที่ผู้จัดการนั้นเป็นเจ้าของได้ ในขณะเดียวกัน เมื่อทราบถึงชื่อโครงการ ก็จะสามารถทราบถึงชื่อของผู้จัดการโครงการนั้นได้เช่นเดียวกัน ดังตารางต่อไปนี้

ตารางที่ 2.24 ตัวอย่าง Functional Dependency ประเภทที่ 3

PROJECT_NO	MANAGER
PJ001	EN001
PJ002	EM002
PJ003	EM003
PJ004	EM004

จากตาราง จะสังเกตเห็นว่า ข้อมูลใน Attribute "MANAGER" และ "PROJECT_NO" ต่างไม่มีข้อมูลที่ซ้ำกันเลย ดังนั้นทั้ง 2 Attribute จึงไม่สามารถทำหน้าที่เป็น Determinant ได้เช่นเดียวกัน สำหรับความสัมพันธ์ในลักษณะนี้สามารถแสดงด้วย ได้ดังนี้ Functional Dependency ได้ดังนี้

Project_no → Manager

Manager → Project_no

Project_no ← → Manager

4. Functional Dependency ที่ต้องใช้ Determinant มากกว่า 1 ค่าเพื่ออ้าง Dependency เช่น ความสัมพันธ์ระหว่างจำนวนสินค้าที่ผลิตได้ ของสินค้าแต่ละชนิดภายใต้สายการผลิตต่างๆ ดัง ตารางต่อไปนี้

ตารางที่ 2.25 ตัวอย่าง Functional Dependency ประเภทที่ 4

PRODUCT_LINE	ITERM_NO	USE_QTY
L001	I012	30
	I019	40
	I024	90
L004	I001	73
L005	I001	45
	I012	98
	I024	34
L006	I009	56
	I019	32
L010	I004	94

จากตางราง จะสังเกตเห็นว่า จะต้องทราบถึงรหัสสินค้า (Attribute "ITEM_NO") และหมายเลขของสายการผลิต (Attribute "PRODUCT_LINE") จึงจะสามารถระบุจำนวนที่ผลิต (Attribute "USE_QTY")ของสินค้านั้น เช่น เมื่อระบุรหัสสินค้าเป็น "I012" จะไม่สามารถทราบจำนวนทีผลิตได้ เนื่องจาก รหัสสินค้าที่กำหนด ปรากฏอยู่ใน 2 สายการผลิต ดังนั้น จึงต้องระบุหมายเลขของการผลิต ประกอบกับรหัสสินค้า จึงจะสามารถระบุจำนวนที่ผลิตได้ สำหรับความสัมพันธ์ใน ลักษณะนี้สามารถแสดงด้วย Functional Dependency ได้ดังนี้

PRODUCT_LINE, ITEM_NO → USE_QTY

2.8.4 Fully Functional Dependency

ได้แก่ Functional Dependency ที่มี Determinant ที่มีขนาดเล็กที่สุดและสามารถระบุถึง Dependency ได้ เช่น Functional Dependency ต่อไปนี้

a) d1 : PERSON_ID → ADDRESS

D2 : PERSON_ID, PERSON_NAME

จากตัวอย่าง เมื่อกำหนดให้ Attribute " PERSON_ID" ทำหน้าที่เป็น Key ของ Relation จะสังเกตเห็นว่า d1 จัดเป็น Fully Functional Dependency เนื่องจาก d1 มี Determinant ที่มีขนาดเล็กที่สุดและสามารถระบุDependencyซึ่งได้แก่Attribute"ADDRESS"ได้ส่วนd2Attribute'PERSON_NAME " ซึ่งไม่มีความจำเป็นแต่อย่างใดมาประกอบ จึงไม่จัดเป็น Determinant ที่มีขนาดเล็กที่สุด

b) d1 : PRODUCT_LINE, ITERM_ NO ➔ USE_QTY

d2 : PRODUCT_LINE, ITERM_NO, MANAGER ➔ USE_QTY

จากตัวอย่าง เมื่อกำหนดให้ Attribute "PRODUCY_ LINE" และ "ITEM_NO" ทำหน้าที่เป็น Key ของ Relation จะสังเกตเห็นว่า d1 จัดเป็น Fully Functional Dependency เนื่องจากไม่จำเป็นต้องมี Attribute "MANAGER" มาประกอบ ก็สามารถระบุค่าของ Attribute " USE_QTY" ได้

2.8.5 Multi-Value Dependency (MVDs)

เป็นความสัมพันธ์ในลักษณะที่ค่าของ Determinant 1 ค่าสามารถระบุค่าของ Attribute ที่ทำหน้าที่เป็น Dependency ได้ตั้งแต่ 2 Attribute ขึ้นไป โดยค่าของข้อมูลใน Attribute ที่เป็น Dependency เหล่านี้ จะอยู่ในรูปของข้อมูลแทน ซึ่งแตกต่างจากความสัมพันธ์ระหว่าง Determinant และDependency โดยทั่วไป ที่เมื่อระบุค่าให้กับ Determinant จะปรากฏค่าของแต่ละ Dependency ที่มีความสัมพันธ์กับ Determinant ที่ระบุเพียงค่าเดียว เช่น กรณีที่พนักงานสามารถสังกัดฝ่ายได้มากกว่า 1 ฝ่าย และสามารถทำงานในโครงการได้มากกว่า 1 โครงการ ดังตัวอย่างข้อมูลต่อไปนี้

ตารางที่ 2.26 ตัวอย่าง Multi –Value Dependency

EMPLOYEE#	DEPARTMENT#	PROJECT#
11001	01	P1
11001	02	P3
11001	01	P2
12003	03	P7
12114	01	P1
12114	02	P3

จากตารางจะสังเกตเห็นว่า เมื่อระบุรหัสของพนักงานให้กับ Attribute "EMPLOYEE#" จะปรากฏค่าใน Attribute "DEPARTMENT# ซึ่งเก็บรหัสพนักงานสังกัดมากกว่า 1 ค่า และค่าใน Attribute " PROJECT# " ซึ่งเก็บรหัสโครงการที่พนักงานทำงานอยู่มากกว่า 1 ค่าเช่นเดียวกัน เช่น เมื่อระบุรหัสพนักงานเป็น "11001" จะปรากฏฝ่ายที่สังกัด 2 ฝ่าย คือ ฝ่ายที่มีรหัส "01" และ "02" และโครงการที่พนักงานทำงานอยู่ 3 โครงการ คือ โครงการ "P1", "P2", "P3"

สำหรับ Functional Dependency ในลักษณะนี้จะใช้ลูกศรจำนวน 2 อันประกอบกัน ดังนั้น จากตัวอย่างสามารถแสดงด้วย FDs ได้ดังนี้

MVDs Statement: EMPLOYEE# ➔➔
DEPARTMENT# , PROJECT#

2.8.6 Relation Key

ได้แก่ Attribute หรือกลุ่ม Attribute ที่ทำหน้าที่เป็น Key ให้กับ Relation ซึ่งจะต้องมีคุณสมบัติดังนี้

1. จะต้องเป็นตัวที่สามารถบ่งชี้ค่าของ Attribute หรือกลุ่มของ Attribute อื่นๆ ในแต่ละ Tuple ของ Relation ได้

2. จะต้องเป็น Attribute หรือกลุ่มของ Attribute ที่มีขนาดเล็กที่สุดเช่นเดียวกับคุณสมบัติของ Key โดยทั่วไป

3. จะต้องไม่มีค่าที่เป็น null

สำหรับ Attribute หรือกลุ่มของ Attribute ที่กำหนดให้เป็น Relation Key จะแสดงด้วยการขีดเส้นใต้ Attribute นั้น เช่น Attribute "EMPLOYEE" ที่มีตัวอย่างข้อมูลต่อไปนี้

ตารางที่ 2.27 ตัวอย่าง Relation Key

EMPLOYEE

EMP_ID	NAME	BIRTHDAY	CITY_OF_BIRTH
E001	ANN	11/12/1959	กรุงเทพฯ
E002	MAM	12/12/1959	เชียงใหม่
E003	JOE	13/12/1959	ระยอง
E00	ANN	14/12/1959	พิจิตร
E005	MAY	09/03/1955	ระยอง

จะสังเกตเห็นว่า Attribute "EMP_ID" ถูกกำหนดให้เป็น Relation Key ของ Relation เนื่องจากค่าของ Attribute นี้ สามารถอ้างถึงค่าของ Attribute อื่นๆ ในแต่ละ Tuple ของ Relation ได้ ส่วน Attributer "BIRTHDAY" และ "CITY_OF_BIRTH" นั้น แม้จะไม่มีค่าที่ซ้ำกันในแต่ละ Tuple เช่นเดียวกับ Attribute "EMP_ID" แต่สาเหตุที่ไม่ถูกเลือกให้เป็น Relation Key เนื่องจากมีโอกาสที่จะมีคน

เกิดวันที่ และสถานที่เกิดเดียวกันเกิดขึ้นได้ สำหรับ Attribute "NAME" ที่ไม่ถูกกำหนดให้เป็น Relation Key เนื่องจากมีพนักงานมากกว่าหนึ่งคนที่มีชื่อเดียวกัน

2.8.7 First Normal Form (1NF)

เป็นขั้นตอนสำหรับปรับโครงสร้างข้อมูลของ Relation เพื่อให้ ทุก Attribute ของ Relation มีคุณสมบัติ Atomicity กล่าวคือ โครงสร้างข้อมูลของ Relation ในแบบ 1NF นี้จะต้องประกอบด้วย Attribute ที่ไม่อยู่ในรูป Repeating Group เช่น ตัวอย่างข้อมูลต่อไปนี้

ตารางที่ 2.28 ตัวอย่าง First Normal Form

ORDER1

CUST_NO	CUST_NAME	CITY	ZONE_SALE	ORDER_CONTERNT	
				PRODUCT_ID	ORDER_QTY
C001	นารี เกิดสว่าง	อยุธยา	001	P001	24
				P003	30
				P004	50
C002	สลักจิต สว่างภพ	ศรีสะเกษ	002	P001	29
				P002	40
				P004	30
C003	สุทิศา เจ็กสกลุ	เชียงใหม่	004	P005	60
C004	ฟ้า เพิ่มพร	ศรีสะเกษ	002	P002	50
				P003	38
C009	ต้นสาย ตันเจริฯ	เชียงใหม่	004	P003	40

จากตารางจะสังเกตเห็นว่า (Attribute "CUST_NO') 1 คน สามารถมีรายการสินค้าที่สั่งซื้อ (Attribute "CUST_NAME") ได้มากกว่า 1 รายการ ดังนั้น จึงกล่าวได้ว่า Attribute "CUST_NO" นี้มีความสัมพันธ์กับ Attribute "ORDER_CONTENT" ในแบบ Repeating Group ส่งผลให้ Relation นี้มีโครงสร้างที่ไม่สอดคล้องกับ 1NF ดังนั้น จึงต้องทำการ Normalization โดยการแปลง Attribute ที่อยู่ในรูป Repeating Group ให้มีคุณสมบัติ Atomicity พร้อมกับกำหนดให้ Attribute ดังกล่าวเป็น Reletion Key ของ Reletion ดังนั้น จากตัวอย่างข้างต้น จึงถูกแปลงให้อยู่ในรูปดังนี้

ตารางที่ 2.28 (ต่อ) ตัวอย่าง First Normal Form

ORDER1

CUST_NO	CUST_NAME	CITY	ZONE_SALE	PRODUCT_ID	ORDER_QTY
C001	นารี เกิดสว่าง	อยุธยา	001	P001	24
C001	นารี เกิดสว่าง	อยุธยา	001	P003	30
C001	นารี เกิดสว่าง	อยุธยา	001	P004	50
C002	สลักจิต สว่างภพ	ศรีสะเกษ	002	P001	29
C002	สลักจิต สว่างภพ	ศรีสะเกษ	002	P002	40
C002	สลักจิต สว่างภพ	ศรีสะเกษ	002	P004	30
C003	สุทิศา เจ็กสกุล	เชียงใหม่	004	P005	60
C004	ฟ้า เพิ่มพร	ศรีสะเกษ	002	P002	50
C004	ฟ้า เพิ่มพร	ศรีสะเกษ	002	P003	38
C009	ต้นสาย ต้นเจริญ	เชียงใหม่	004	0003	40

จะสังเกตเห็นว่า แต่ละค่าที่เป็น Repeating Group ของ Attribute "PRODUCT_ID" และ "ORDER_QTY" จะถูกแยกออกมา Tuple ใหม่ พร้อมกับกำหนดให้ Attribute "PRODUCT_ID" ทำหน้าที่ Relation Key เป็นร่วมกับ Attribute " CuST_ NO"

อย่างไรก็ตาม Relation ที่อยู่ในรูป 1NF ถึงแม้จะทำให้ทุก Attribute มีคุณสมบัติ Atomicity แต่กลับเกิดปัญหาความซ้ำซ้อนกันของข้อมูล(Redundacy)ขึ้นในAttribute"CUST_NO","CUDT_NAME","CITY" และ "ZONE_SALE" ก่อให้เกิดปัญหาทางด้าน Anomaly ดามดังนี้

1. Insert Anomaly

เมื่อพิจารณาจากตัวอย่าง จะสังเกตเห็นว่าการเพิ่มข้อมูลลูกค่า จะทำได้ก็ต่อเมื่อลูกค้านั้นมีรายการสั่งซื้อสินค้าแล้วเท่านั้น หมายถึงว่า Relation นี้ จะไม่สามารถจัดเก็บข้อมูลของลูกค้าที่ยังไม่มีการสั่งซื้อได้ เช่น เมื่อต้องการเพิ่มข้อมูลลูกค้ารหัส "C006" โดยที่ยังไม่มีการกำหนดการสั่งซื้อสินค้าใน Attribute "PRODUCT_ID" จะไม่สามารถกระทำได้ เนื่องจาก Attribute "PRODUCT_ID" นี้ถูกกำหนดให้เป็น Relation Key จึงไม่สอดคล้องตามกฎของ Entity Integrity Rule ในส่วนที่ว่า **Attribute หรือกลุ่ม Attribute ที่เป็น Relation Key จะมีค่าเป็น Null ไม่ได้**

2. Delete Anomaly

เมื่อพิจารณาจากตัวอย่าง จะสังเกตเห็นว่า การลบข้อมูลรายการสั่งซื้อ (Attribute "PRODUCT_ID" และ "ORDER_QTY") บาง Tuple ใน Relation นี้ จะทำให้ข้อมูลลูกค้าบางคนสูญหายไป เช่น เมื่อลบข้อมูลรายการสั่งซื้อของสินค้ารหัส "P005" ของลูกค้ารหัส "C003" จะทำให้ข้อมูล ลูกค้ารหัส "C003" ถูกลบตามไปด้วย

3. Update Anomaly

เมื่อพิจารณาจากตัวอย่าง จะสังเกตเห็นว่า การปรับปรุงข้อมูลใน Tuple ที่มีค่าของข้อมูลที่ซ้ำซ้อนกันไม่ครบถ้วน อาจก่อให้เกิดความขัดแย้ง ของข้อมูลที่ซ้ำซ้อนกันนั้นได้ เช่น การเปลี่ยนชื่อลูกค้าจาก "สลักจิต สว่างภพ" เป็น "รินลณี สว่างภพ" ของลูกรหัส "C002" ซึ่งถ้าแก้ไขไม่ครบแล้ว จะทำให้ลูกค้ารหัส "C002" มีชื่อทั้ง "สลักจิต สว่างภพ" และ "รินลณี สว่างภพ"

2.8.8 Second Normal Form (2NF)

ในการทำ Normalization ในขั้นตอน Second Normal Form จำเป็นที่จะต้องรู้จักถึง Prime Attribute และ Nonprime Attribute เนื่องจาก Attribute ทั้ง 2 ประเภทนี้ จะมีความสำคัญต่อการทำ Normalization แบบ Second Normal Form

Prime Attribute ได้แก่ ทุก Attribute ที่ทำหน้าที่เป็น Relation Key ของ Relation ส่วน Nonprime Attribute ได้แก่ Attribute ที่ไม่ได้เป็นส่วนหนึ่งของ Relation Key

คุณสมบัติ Relation ที่มีโครงสร้างในแบบ 2NF จะต้องมีคุณสมบัติดังนี้

1. ต้องมีโครงสร้างเป็นไปตามโครงสร้างของ 1NF
2. ทุก Nonprime Attribute จะต้องไม่ขึ้นกับ Relation Key ที่อยู่ในรูป Subset

ตัวอย่างเช่น Relation "ORDER1" ในตัวอย่างที่ผ่านมา ซึ่งเป็น Relation ที่มีคุณสมบัติของ 1NF จะสังเกตเห็นว่า Attribute (Cust No, Product ID) เป็น Attribute ที่ทำให้ข้อมูลในแต่ละ Tuple มีค่าไม่ซ้ำกัน ดังนั้น Attribute ทั้ง 2 จึงทำหน้าที่เป็น Relation Key ซึ่งสามารถเขียนได้ด้วย Fuctional Dependency ได้ดังนี้

FD: CUST_NO,PORODUCT_ID ➔
CUST_NAME,CITY, ZONE_SALE,ORDER_QTY

เมื่อพิจารณาค่าของ Attribute "CUST_NO", "CUST_NAME", "CITY", ZONE_SALE" จะสังเกตเห็นว่า Tuple ที่ประกอบขึ้นจาก Attribute เหล่านี้ จะมีข้อมูลที่ซ้ำกันเป็นชุด ๆ และมีเพียง Attribute "ORDER_QTY" เท่านั้น ที่มีค่าเปลี่ยนตามค่าของ Relation Key ดังนั้น จึง

d1: CUST_NO,PORODUCT_ID → ORDER_QTY

d2: CUST_NO→ CUST_NAME,CITY, ZONE_SALE

สามารถเขียนด้วย Fuctional Dependency ได้ดังนี้

ใน d2 จะสังเกตเห็นว่า Attribute "CUST_NAME", "CITY" และ "ZONE_SALE" เป็น Nonprime Attribute ของ Relation ที่ไม่ได้ขึ้นอยู่กับเฉพาะ Relation Key แต่กลับขึ้นอยู่กับค่าของ Attribute "CUST_NO" ด้วย แสดงว่า d2 นี้ไม่เป็นไปตามคุณสมบัติข้อที่ 2 ดังนั้น Relation"ORDER1" จึงไม่มีคุณสมบัติเป็นไปตามคุณสมบัติของ 2NF จึงต้องแตก Relation "ORDER1" ออกเป็น 2 Relation ตาม d1 และ d2 ดังนี้

ตารางที่ 2.29 ตัวอย่าง Second Normal Form

CustOrder

CUST_NO	PRODUCT_ID	ORDER_QTY
C001	P001	24
C001	P003	30
C001	P004	50
C002	P001	29
C002	P001	40
C002	P001	30
C003	P005	60
C004	P002	50
C004	P003	38
C009	P003	40

Cust

CUST_NO	CUST_NAME	CITY	ZONE_SALE
C001	นารี เกิดสว่าง	อยุธยา	001
C002	สลักจิต สว่างภพ	ศรีสะเกษ	002
C003	สุทิศา เจ็กสกุล	เชียงใหม่	004
C004	ฟ้า เพิ่มพร	ศรีสะเกษ	002
C009	ต้นสาย ตันเจริญ	เชียงใหม่	004

สำหรับโครงสร้างของ Relation "CustOrder" และ Cust" นี้ จะสังเกตเห็นว่าสามารถแก้ปัญหา Anomaly ที่เกิดขึ้นใน Relation "ORDER1" ได้ดังนี้

- Relation "Cust" สามารถเพิ่มข้อมูลลูกค้าได้โดยไม่ต้องการสั่งซื้อสินค้าที่เกิดขึ้นเนื่องจากข้อมูลการสั่งซื้อสินค้าจะถูกแยกจัดเก็บใน Relation "CustOrder" จึงสามารถแก้ปัญหา Insert Anomaly ที่เกิดขึ้นได้
- สามารถลบข้อมูลรายการสั่งซื้อสินค้ารหัส "P005" ของลูกค้ารหัส "C003" ใน Relation "CustOrder" ได้โดยไม่ส่งผลต่อข้อมูลของลูกค้ารหัส "C003" เนื่องจากข้อมูลลูกค้าถูกแยกจัดเก็บใน Relation "Cust" จึงสามารถแก้ปัญหา Delete Anomally ที่เกิดขึ้นได้
- เนื่องจากข้อมูลของลูกค้าที่ซ้ำซ้อนจะถูกแยกมาจัดเก็บใน Relation "Cust" ดังนั้นการเปลี่ยนแปลงรายละเอียดข้อมูลลูกค้าจึงกระทำกับ Relation "Cust" เพียง Relation เดียวจึงไม่ก่อให้เกิดปัญหา Update Anomaly

2.8.9 Third Normal Form (3NF)

สำหรับ Relation ที่จะมีโครงสร้างในแบบที่ 3NF จะต้องมีคุณสมบัติดังนี้

1. ต้องมีคุณสมบัติของ 2NF
2. ต้องไม่มี Fuctional Dependency เกิดขึ้นระหว่าง Nonprime Attribute ด้วยกันเองที่เรียกว่า "Transitive Depency"

จาก Relation "Cust" ในหัวข้อที่ผ่านมา ถึงแม้ว่า จะมีโครงสร้างเป็นไปตามคุณสมบัติของ 2NF แต่จะสังเกตเห็นว่า ค่าของ Attribute "CITY" และ "ZONE_SALE" ถ้าปรากฏข้อมูลที่มีค่าซ้ำซ้อนกันอยู่เป็นคู่ ๆ หรือกล่าวอีกนัยหนึ่ง ทั้ง 2 Attribute สามารถที่จะระบุค่าระหว่างกันได้ กล่าวคือ เมื่อระบุค่าให้กับ Attribute "ZONE_SALE" จะสามารถทราบถึงชื่อเมืองใน Attribute "CITY" ได้ ซึ่งความสัมพันธ์ในลักษณะนี้ จะเรียกว่า Transitive Dependency ดังนั้น Relation นี้ จึงขาดคุณสมบัติของ 3 NF และยังก่อให้เกิดปัญหาความผิดพลาดทางด้าน Anomaly ดังนี้

1. Insert Anomaly

ใน Relation "Cust" เมื่อต้องการเพิ่มข้อมูลใหม่ให้กับ Attribute "CITY" และ"ZONE_SALE" ซึ่งมีความสัมพันธ์ในแบบ Transitive Dependency จะไม่สามารถกระทำได้ เนื่องจากข้อมูลใน 2 Attribute นี้ไม่ใช่ Relation Key ดังนั้น จึงต้องเพิ่มข้อมูลนี้ของลูกค้ารายใหม่ให้กับ Attribute "CUST_NO" และ "CUST_NAME" ตามไปด้วย

2 . Update Anomaly

ใน Relation " Cust" จะสังเกตเห็นวา เมื่อมีการแก้ไขข้อมูลใน Attribute "ZONE_SALE" จาก "004" ไปเป็น "005" จะต้องทำการแก้ไขข้อมูลในทุก ๆ Tupe ที่มีค่าของ Attribute "ZONE_SALE" เท่ากับ "004" ให้ครบถ้วน เนื่องจากเมื่อทำการแก้ไขข้อมูลไม่ครบถ้วนจะก่อให้เกิดข้อมูลใน Relation ที่มีความขัดแย้งกันได้

เท่ากับ "004" ให้ครบถ้วน เนื่องจากเมื่อทำการแก้ไขข้อมูลไม่ครบถ้วนจะก่อให้เกิดข้อมูลใน Relation ที่มีความขัดแย้งกันได้

3. Delete Anomaly

ใน Relation "Cust" จะสังเกตเห็นว่า ถ้ามีการลบข้อมูลของ Tuple ที่จัดเก็บข้อมูลในกลุ่ม Transitive Dependency ที่ปรากฏอยู่เพียงชุดเดียวใน Relation จะส่งผลให้ข้อมูลในกลุ่ม Transitive Dependency ที่ปรากฏอยู่เพียงชุดเดียวใน Relation จะส่งผลให้ข้อมูลในกลุ่ม Transitive Dependency นั้นสูญหายไปจาก Relation ได้เช่น เมื่อทำการลบข้อมูลใน Tuple ของลูกค้ารหัส "C001" นอกจากจะทำให้ข้อมูลของลูกค้ารหัส "C001" หายไปแล้ว ยังส่งผลให้ข้อมูลของเขตการขายที่อยุธยาสูญหายไปด้วย

จากปัญหา anomaly ที่เกิดขึ้นจาก Transitive Dependency เหล่านี้ จึงต้องทำการแยก Nonprime Attribute ที่ก่อให้เกิด Transitive Dependency ของ Relation "Cust" ออกมาเป็น Relation ใหม่ ซึ่งจากตัวอย่าง ได้แก่ Attribute "CITY" และ ZONE_SALE" ดังนี้

ตารางที่ 2.30 ตัวอย่าง Third Normal Form

Cust 2

CUST_NO	CUST_NAME	CITY
C001	นารี เกิดสว่าง	อยุธยา
C002	สลักจิต สว่างภพ	ศรีสะเกษ
CUST_NO	CUST_NAME	CITY
C003	สุทิศา เจ็กสกุล	เชียงใหม่
C004	ฟ้า เพิ่มพร	ศรีสะเกษ
C009	ต้นสาย ตันเจริญ	เชียงใหม่

CityZone

CITY	ZONE_SALE
เชียงใหม่	001
ศรีสะเกษ	002
เชียงใหม่	**004**

ซึ่งสามารถเขียนด้วย Fuctional Dependency ได้ดังนี้

d1: CUST_NO → CUST_NAME,CITY
d2: CITY → ZONE_SALE

<u>ข้อสังเกต</u> ในการแยก Nonprime Attribute ที่ก่อให้เกิด Transitive Dependency ออกมาเป็น Relation ใหม่ มีหลักการอยู่ 2 ข้อ ดังนี้

1. แยก Nonprime Attribute ที่ก่อให้เกิด Transitive Dependency ออกเป็น Relation ใหม่
2. กำหนดให้ Nonprime Attribute ที่เป็นตัวระบุค่า ให้เป็น Relation Key ของ Relation ใหม่

2.8.10 Boyce-Codd Normal Form (BCNF)

สำหรับ Relation ที่จะมีโครงสร้างในแบบ BCNF จะต้องมีคุณสมบัติดังนี้

1. ต้องมีคุณสมบัติ 3NF
2. Attribute ที่เป็น Determinant จะต้องเป็น Relation Key

ตัวอย่างเช่น Relation "Customer4" ที่มีตัวอย่างข้อมูลต่อไปนี้

ตารางที่ 2.31 ตัวอย่าง Boyce-Codd Normal Form

Customer 4

CUST_ID	TAX_ID	CUST_NAME	ADDRESS
C001	8906543134	โสภา เตีย	123 ยานนาวา กทม.
C002	5465465465	นันทิดา หยุ่น	4/55 พระโขนง กทม.
CUST_ID	TAX_ID	CUST_NAME	ADDRESS
C003	6034546608	ฮีแมน เมก	3 สุขุมวิทย กทม.
C005	0665409879	วันดา อิ่มใจ	3/7 คลองตัน กทม.

จาก Relation ที่กำหนด สามารถเขียน FD ได้ดังนี้

d1:CUST_ID → TAX_ID,CUST_NAME,ADDRESS
d2: TAX_ID → CUST_ID,CUST_NAME,ADDRESS
d3: CUST_ID → CUST_ID →

เมื่อพิจารณาใน d1 และ d2 จะสังเกตเห็นว่า Attribute "CUST_ ID และ "TAX_ID" ต่างสามารถทำหน้าที่เป็น Determinant เพื่อใช้อ้างถึง Attribute "CUST_NAME" และ "ADDRESS" ได้เช่นเดียวกัน และในขณะเดียวกัน เมื่อพิจารณาใน d3 จะสังเกตเห็นว่า ความสัมพันธ์ของ 2 Attribute นี้เป็นความสัมพันธ์แบบ 2 ทาง กล่าวคือ ต่างฝ่ายสามารถทำหน้าที่แทนหน้าที่ของอีกฝ่ายหนึ่งได้ ดังนั้นในกรณีเช่นนี้ จึงสามารถกำหนดให้ Attribute ใด Attribute หนึ่งใน ใน 2 Attribute นี้ ทำหน้าที่เป็น Relation Key ได้ ซึ่งในตัวอย่าง Attribute "CUST_ID" ถูกกำหนดให้ทำหน้าที่เป็นไปตามคุณสมบัติของ BCNF

แต่ใน Relation "CusOrder" ที่มีตัวอย่างข้อมูลต่อไปนี้

ตารางที่ 2.31 (ต่อ) ตัวอย่าง Boyce-Codd Normal Form

CustOrder

CUST_ID	TAX_ID	PRODUCT_ID	ORDER_QTY
C001	8906543134	I001	90
C002	5465465465	I003	80
C003	6034546608	I004	220
C005	0665409879	I003	100

จาก Relation ที่กำหนด สามารถเขียน FD ได้ดังนี้

d1:CUST_ID ,PRODUCT_ID → ORDER_QTY
d2: TAX_ID ,PRODUCT_ID → ORDER_QTY
d3: CUST_ID ←→ ORDER_QTY

เมื่อพิจารณาตามคุณสมบัติข้อ 2 ของ BCNF จะพบว่า Relation นี้ มีโครงสร้างที่ไม่เป็น BCNF เนื่องจาก Attribute "PROCUCT_ID" ไม่ได้ทำหน้าที่เป็น Relation Key ของ Relation จึงต้องแยก Relation นี้ออกเป็น 2 Relation โดยกระทำได้ดังนี้

<u>วิธีที่ 1</u>

แยก Relation ออกตามความสัมพันธ์ที่กำหนดใน d1 และ d3 ซึ่งจะทำให้ได้ Relation ดังนี้

ตารางที่ 2.31 (ต่อ) ตัวอย่าง Boyce-Codd Normal Form

Cust1

CUST_ID	TAX_ID
C001	8906543134
C002	5465465465
C003	6034546608
C005	0665409879

Cust_Order1

CUST_ID	PRODUCT_ID	ORDER_QTY
C001	I001	90
C002	I003	80
C003	I004	220
C005	I003	100

วิธีที่ 2

แยก Relation ออกตามความสัมพันธ์ที่กำหนดใน d2 และ d3 ซึ่งจะทำให้ได้ Relation ดังนี้

ตารางที่ 2.31 (ต่อ) ตัวอย่าง Boyce-Codd Normal Form

Cust1

CUST_ID	TAX_ID
C001	8906543134
C002	5465465465
C003	6034546608
C005	0665409879

Cust_Order2

TAX_ID	PRODUCT_ID	ORDER_QTY
8906543134	I001	90
5465465465	I003	80
6034546608	I004	220
0665409879	I003	100

ซึ่งจะสังเกตเห็นว่า โครงสร้างของ Relation ใหม่ตามวิธีที่ 1 และ 2 ต่างมีคุณสมบัติของ BCNF

2.8.11 Fourth Normal Form (4NF)

สำหรับ Relation ที่จะมีโครงสร้างแบบ 4 NF จะต้องมีคุณสมบัติดังนี้

1. ต้องมีคุณสมบัติของ BCNF

2. ต้องไม่ปรากฏความสัมพันธ์ระหว่าง Attribute ในแบบ Multi-value Dependency เช่น ตัวอย่างข้อมูลของ Relation "EMPLOYEE_SKILL" ซึ่งใช้จัดเก็บข้อมูลเกี่ยวกับควมสามารถของพนักงาน ทางด้านการใช้คอมพิวเตอร์ (Attribute "COMPUTER_SKILL") และทางด้านการใช้ภาษาต่างประเทศ (Attribute "LANGUAGE_SKILL") ดังนี้

ตารางที่ 2.32 ตัวอย่าง Fourth Normal Form

EMPLOYEE_SKILL

EMPLOYEE#	COMPUTER_SKILL	LANGUAGE_SKILL
1267	Word Processing	ฝรั่งเศส
1267	Word Processing	เยอรมัน
1345	Spreadsheets	สเปน
1267	Spreadsheets	ฝรั่งเศส
1267	Spreadsheets	เยอรมัน
1345	COBOL	สเปน
1193	Word Processing	ฝรั่งเศส

จากตัวอย่างข้อมูล จะสังเกตเห็นว่า Relation นี้ มีคุณสมบัติเป็น BCNF แต่ยังไม่เป็น 4NF เนื่องจาก ปรากฏโครงสร้างข้อมูลในแบบ Multi-value Dependency กล่าวคือ เมื่อระบุค่าของ Attribute "EMPLOYEE#" ซึ่งทำหน้าที่เป็น Determinant จะสามารถแสดงค่าของ Attribute "COMPUTER_SKELL" และ "LANGUAGE_SKILL" ที่ทำหน้าที่เป็น Dependency ได้มากกว่า 1 ค่า

ดังนั้น จึงต้องแบ่ง Relation นี้ออกเป็น Relation ใหม่ตามโครงสร้างข้อมูลแบบ Multi-value Dependency ที่ปรากฏอยู่ ซึ่งได้แก่ Relation "COMPUTER_SKILLS" และ "LANGUAGE_SKILLS" ดังนี้

ตารางที่ 2.32 (ต่อ) ตัวอย่าง Fourth Normal Form

COMPUTER_SKILLS

EMPLOYEE#	COMPUTER_SKILL
1267	Word Processing
1345	Spreadsheets
1267	Spreadsheets
1345	COBOL
1193	Word Processing

LANGUAGE_SKILL

EMPLOYEE#	LANGUAGE_SKILL
1267	ฝรั่งเศส
1267	เยอรมัน
1345	สเปน
1193	ฝรั่งเศส

2.8.13 สรุป

โครงสร้างของฐานข้อมูลที่ออกแบบขึ้น ควรที่จะนำมาปรับปรุงโดยใช้วิธีของการทำ Normalization เพื่อปรับเปลี่ยนโครงสร้างของ Relation ต่างๆ ที่ได้ออกแบบไว้ ให้มีโครงสร้างที่เป็นไปตามคุณสมบัติของ Relation ที่กำหนดไว้ใน Relation Model รวมทั้งมีโครงสร้างที่เหมาะสมต่อการนำไปใช้งาน ในการทำ Normalization จะมีอยู่ด้วยกัน 5 ขั้นตอน แต่ส่วนใหญ่ในทางปฏิบัติแล้ว การทำ Normalization จะกระทำถึงขั้นตอน Boyce-Codd Normal Form (BCNF) เท่านั้น ก็เพียงพอแล้ว

2.9 การประยุกต์นำบาร์โค้ดมาใช้กับโครงงาน

ในโลกปัจจุบันผลิตภัณฑ์ทางด้านไฮ-เทคโนโลยีมีการเปลี่ยนแปลงอย่างรวดเร็วมาก และการเปลี่ยนแปลงที่เกิดขึ้นก็ล้วนแต่นำมาซึ่งความเจริญก้าวหน้าต่อการดำรงชีวิตทั้งสิ้นไม่ว่าจะเป็นการประกอบธุรกิจ หรือแม้กระทั่งการเรียนการสอนในสถาบันต่างๆซึ่งแทบจะกล่าวได้ว่าทุกๆสิ่งที่อยู่รอบตัวเรานั้นมีเทคโนโลยีเข้ามาเกี่ยวข้องด้วยเสมอ ด้วยเหตุนี้การทำความเข้าใจ กับเทคโนโลยีใหม่ๆดังเช่น รหัสแท่ง หรือที่เรียกว่าบาร์โค้ด จึงเป็นสิ่งที่น่าสนใจเพื่อชีวิตที่ดีขึ้นกว่าเดิม

2.9.1 ความหมายของรหัสแท่ง

รหัสแท่ง หมายถึง ระบบสัญลักษณ์หรือเครื่องหมายประจำตัวของสินค้า หรือนำมาประยุกต์ทำบัตรประจำตัวสมาชิกอย่างเช่นในโครงงานนี้ เพื่อแทนเลขรหัสโดยทั่วไปจะเป็นภาษาสากลสำหรับคอมพิวเตอร์ที่ใช้สื่อ หรือบ่งบอกถึงข้อมูลที่ต้องการ ซึ่งจะสามารถประหยัดเวลาและค่าใช้จ่ายลงได้เป็นอย่างมาก

ขั้นตอนของการใช้รหัสแท่ง

เริ่มจาก กำหนดเลขหมายประจำตัวให้กับสมาชิก แล้วจึงนำเลขรหัสมาแปลงเป็นรหัสแท่งโดยกำหนดเป็นสัญลักษณ์สีเข้มสลับสีสว่างที่มีขนาดแตกต่างกัน นำไปติดบนบัตรสมาชิก การจะอ่านรหัสนี้สามารถกระทำได้โดยนำแถบนี้ไปผ่านเครื่องมือซึ่งเรียกว่าเครื่องสแกนเนอร์ (Scanner) ซึ่งต่อเชื่อมโยงกับคอมพิวเตอร์ที่มีข้อมูลของสมาชิกอยู่ เมื่อเครื่องสแกนเนอร์รับรู้รหัสจากความแตกต่างของแถบสีเข้มสลับสีสว่าง ที่หนาบางต่างกันก็จะส่งผ่านไปยังเครื่องคอมพิวเตอร์ เพื่อทำประมวลผลข้อมูลที่ได้อ่านจากรหัสแท่ง เพื่อแสดงข้อมูลของสมาชิก

ประโยชน์ของเลขหมายประจำตัว และรหัสแท่ง

- เพิ่มประสิทธิภาพในการวางแผนการบริหาร

ระบบเลขหมายประจำตัวสมาชิก และรหัสแท่งจะช่วยให้ผู้ผลิต และผู้ประกอบการสามารถตัดสินใจทางด้านการผลิตและการตลาดได้อย่างรวดเร็วมีประสิทธิภาพ เนื่องจากข้อมูลที่ได้จากรหัสแท่งจะถูกนำแปรเป็นข้อมูลที่สำคัญทั้งยอดขาย ประเภทของสิ้นค้าที่ขาย ตลอดจนยอดสินค้าคงเหลือในสต็อกโดยผ่านเครื่องคอมพิวเตอร์ ดังนั้นข้อมูลจากระบบเลขหมายประจำตัวสินค้า และรหัสแท่งทำให้ผู้ผลิต ตลอดจนผู้ประกอบการสามารถทราบถึงรูปแบบ รสนิยม และความต้องการของผู้บริโภค ได้อย่างรวดเร็วด้วยค่าใช้จ่ายที่ต่ำ อันเป็นประโยชน์ต่อการวางแผนการผลิตและการกำหนดกลยุทธด้านการตลาดที่ทันต่อเหตุการณ์

2.9.2 ระบบที่เป็นที่รู้จักและใช้ในปัจจุบัน

ประเทศต่างๆในโลกใช้สัญลักษณ์รหัสแท่งมี 2 ระบบด้วยกันได้แก่

1. ระบบ UPC (Universal Product Code)

ระบบ UPC เป็นระบบรหัสแท่งที่ได้รับการพัฒนาในประเทศสหรัฐอเมริกา ประมาณปี พ.ศ. 2515 โดยมีองค์กร UCC (Uniform Code Council) เป็นผู้ดูแลระบบ UPC ได้พัฒนามาใช้กับระบบค้าปลีกยุคของสหรัฐอเมริกา และแคนาดาเท่านั้น มิได้เป็นที่นิยมแพร่หลาย

2. ระบบ EAN (International Article Numbering Code)

เป็นระบบที่กลุ่มประเทศในแถบยุโรป พัฒนาขึ้นมาโดยมีองค์กร EAN International ณ กรุงบลัสเซล ประเทศเบลเยี่ยม เป็นผู้ดูแลตั้งแต่ปี พ.ศ. 2519 ในปัจจุบันมีผู้นิยมใช้ระบบนี้กว่า 91 ประเทศทั่วโลกโดยใช้กันแพร่หลายในภาคพื้นยุโรป เอเชีย และ แปซิฟิก รวมทั้งประเทศไทยด้วย