บทที่ 6
การนอร์มัลไลเซชั่น

ในบทนี้จะอธิบายเกี่ยวกับการออกแบบฐานข้อมูลที่ตั้งอยู่บนพื้นฐานของแบบจำลองเชิงสัมพันธ์เป็นสำคัญ ถึงแม้ว่าปัจจุบันจะมีแบบจำลองข้อมูลอื่นๆ ให้เลือกใช้งาน แต่มีเหตุผลสำหรับการเลือกออกแบบจำลองเชิงสัมพันธ์ ด้วยเหตุผลที่ว่า แบบจำลองข้อมูลเชิงสัมพันธ์ เป็นแบบจำลองที่นิยมใช้งานบนแอปพลิเคชันมากที่สุด และการออกแบบฐานข้อมูลเชิงตรรกะที่ใช้งานบนแบบจำลองเชิงสัมพันธ์ จะนำเสนอในรูปแบบของรีเลชั่นหรือตาราง ซึ่งสามารถนำเสนอความสัมพันธ์ของข้อมูลในภาพรวมได้ดี และเข้าใจง่ายกว่าแบบจำลองชนิดอื่นๆ

วัตถุประสงค์หลักของการออกแบบฐานข้อมูลเพื่อใช้งานในองค์กรคือ การสร้างตัวแทนของข้อมูลที่มีความถูกต้องทั้งในส่วนของความสัมพันธ์ระหว่างข้อมูลและข้อบังคับบนข้อมูล โดยแนวทางในการช่วยให้บรรลุถึงวัตถุประสงค์ดังกล่าว จำเป็นต้องใช้เทคนิคการออกแบบฐานข้อมูลเข้าช่วย ซึ่งเทคนิคหนึ่งที่ ได้กล่าวมาแล้วคือแบบจำลอง E-R ต่อไปนี้จะนำเสนอเทคนิคที่ใช้สำหรับการออกแบบฐานข้อมูลนั่นก็คือ การนอร์มัลไลเซชั่น (normalization)

กฎความคงสภาพในข้อมูล

แบบจำลองข้อมูลเชิงสัมพันธ์ประกอบด้วยข้อบังคับต่างๆ หลายชนิด รวมถึงกฎเกณฑ์ทางธุรกิจที่สามารถนำมาบังคับใช้เพื่อให้ข้อมูลที่บันทึกเป็นไปตามนโยบายขององค์กร กฎความคงสภาพในข้อมูล (data integrity rule) คือกฎเกณฑ์และข้อจำกัดที่กำหนดเพื่อควบคุมความถูกต้องของข้อมูลเพื่อให้ข้อมูลในฐานข้อมูลนั้นมีความถูกต้องตรงกันตามความเป็นจริง และป้องกันมิให้เกิดข้อผิดพลาดในข้อมูล ซึ่งอาจเกิดขึ้นจากข้อผิดพลาดจากการเพิ่มข้อมูล ข้อผิดพลาดจากการลบข้อมูล และข้อผิดพลาดจากการเปลี่ยนแปลงข้อมูลที่บันทึกอยู่ในฐานข้อมูล

กฎความคงสภาพในข้อมูลประกอบด้วย 3 กฎเกณฑ์หลักๆ คือ กฎความคงสภาพของเอนทิตี (entity integrity) กฎความคงสภาพของโดเมน (domain integrity) และกฎความคง

สภาพของการอ้างอิง (referential integrity) ซึ่งจะอธิบายดังต่อไปนี้ (Mata-Toledo & Cushman, 2000, p. 252)

1. กฎความคงสภาพของเอนทิตี

กฎความคงสภาพของเอนทิตี (entity integrity) เป็นกฎที่ว่าด้วยการออกแบบเพื่อความแน่นอนของ รีเลชั่นที่มีคีย์หลัก โดยรับประกันว่าทุกแอตทริบิวต์ที่เป็นคีย์หลักจะต้องไม่เป็นค่าว่าง หากพบว่าทูเพิลใดมีคีย์หลักเป็นค่าว่าง (null) การอ้างอิงความเป็นเอกลักษณ์ของทูเพิลนั้นก็จะผิดกฎความคงสภาพของเอนทิตีไปโดยปริยาย เช่น รีเลชั่นลูกค้ามีการกำหนดให้รหัสลูกค้าเป็นคีย์หลัก และค่าข้อมูลของรหัสลูกค้าจะใช้ตัวเลขเป็นตัวกำหนดรหัส แต่ถ้ามีการเพิ่มข้อมูลลูกค้ารายใหม่เข้าไปแล้วปล่อยให้รหัสลูกค้าเป็นค่าว่าง นั่นหมายถึงมีคีย์เป็นค่าว่าง จึงถือเป็นการผิดหลักในกฎความคงสภาพของเอนทิตี ดังนั้น หากมีการตั้งกฎความคงสภาพของเอนทิตีลงไป ระบบก็จะไม่อนุญาตให้คีย์หลักเป็นค่าว่าง ทำให้แต่ละทูเพิลในรีเลชั่นนั้นสามารถอ้างอิงความเป็นเอกลักษณ์ของตัวเองได้สำหรับกฎความคงสภาพของเอนทิตีนั้น บางครั้งอาจเรียกชื่อหนึ่งว่า key integrity

2. กฎความคงสภาพของโดเมน

กฎความคงสภาพของโดเมน (domain integrity) เป็นกฎเกณฑ์ที่ใช้เพื่อควบคุมค่าข้อมูลให้อยู่ในช่วงที่เหมาะสมและถูกต้อง ตัวอย่างเช่น เกรดเฉลี่ยสะสม (GPA) ซึ่งจะต้องมีค่าระหว่าง 0.00 ถึง 4.00 ดังนั้นการพัฒนาฐานข้อมูลก็จะต้องสามารถควบคุมค่าดังกล่าวได้ โดยเกรดเฉลี่ยสะสมต้องไม่เป็นค่าติดลบและจะมีค่าเกินกว่า 4.00 ไม่ได้ เนื่องด้วยกฎเกณฑ์ในลักษณะนี้มักจะถูกควบคุมหรือตรวจสอบด้วยแอปพลิเคชันโปรแกรมมากกว่า แต่อย่างไรก็ตามระบบจัดการฐานข้อมูลหรือ DBMS ในปัจจุบันมักจะมีการผนวกความสามารถกฎการบังคับใช้โดเมนนี้ได้ ซึ่งผู้ใช้งานอาจจะเลือกใช้งานอย่างใดอย่างหนึ่งหรืออาจนำมาใช้ร่วมกัน

3. กฎความคงสภาพของการอ้างอิง

กฎความคงสภาพของการอ้างอิง (referential integrity) เป็นกฎความคงสภาพของการอ้างอิงที่ว่า ถ้าหากมีรีเลชั่นหนึ่งซึ่งเป็นคีย์นอก (foreign key) มีการอ้างอิงกับคีย์หลักในอีกรีเลชั่น โดยพิจารณาจากภาพที่ 6.2 ซึ่งเป็นรีเลชั่นที่ได้จากชุดคำสั่ง SQL ดังภาพที่ 6.1

ที่ผ่านมา จะพบว่ารีเลชั่น ORDER จะมีคีย์หลักคือ orderID และจะมีคีย์นอกคือ customerID ซึ่งอ้างอิงกับแอตทริบิวต์ customerID ในรีเลชั่น CUSTOMER

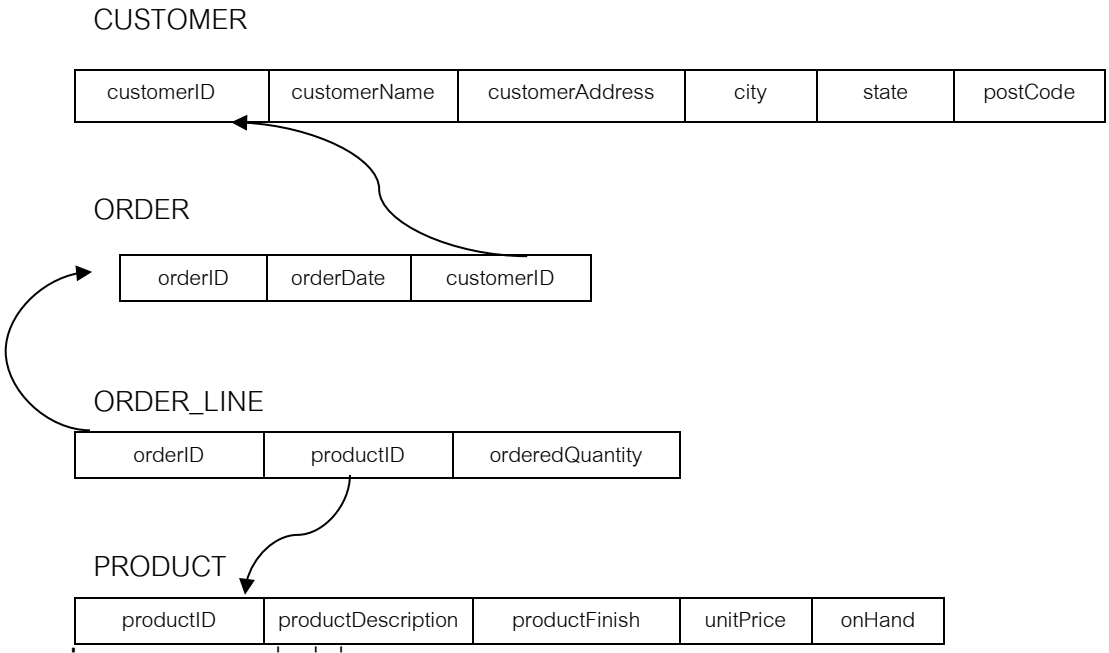
```
CREATE TABLE CUSTOMER
      (customerID                    VARCHAR  (5)        NOT NULL,
       customerName                  VARCHAR (25)        NOT NULL,
       customerAddress               VARCHAR (30)        NOT NULL,
       city                          VARCHAR (20)        NOT NULL,
       state                         CHAR (2)            NOT NULL,
       postcode                      CHAR (10)           NOT NULL,
PRIMARY  KEY   (customerID) ) ;

CREATE TABLE ORDER
      (orderID                       CHAR (5)            NOT NULL,
       orderDate                     DATE                NOT NULL,
       customerID                    VARCHAR  (5)        NOT NULL,
PRIMAR   KEY  (orderID) ;
FOREIGN  KEY  (customerID)  REFERNCE  CUSTOMER (customerID)) ;

CREATE TABLE ORDER_LINE
       (orderID                      CHAR (5)            NOT NULL,
        product                      CHAR (5)            NOT NULL,
        orderedQuantity              INT                 NOT NULL,
PRIMARY KEY (orderID); productID),
FOREIGN KEY (orderID)  REFERENCES  ORDER (orderID),
FOREIGN KEY (productID) REFERENCES PRODUCT (productID) ) ;

CREATE TABLE  PRODUCT
(product ID                          CHAR (5)            NOT NULL,
productDescription                   VARCHAR (25) ,
 productFinish                       VARCHAR (12) ,
 unitprice                           DECIMAL(8, 2)       NOT NULL,
onhand                               INT                 NOT NULL,
 PRIMARY  KEY (productID) ) ;
```

ภาพที่ 6.1 ตัวอย่างชุดคำสั่ง SQL ในการสร้างตาราง และการกำหนดกฎความคงสภาพของแอตทริบิวต์ที่ใช้เป็นคีย์หลัก ซึ่งจะเป็นค่า null ไม่ได้ รวมถึงการกำหนดคีย์หลักและคีย์นอก
ที่มา : Mata-Toledo & Cushman (2000, p. 254)



ภาพที่ 6.2 ตัวอย่างรีเลชั่นที่เชื่อมโยงกันผ่านคีย์

ที่มา : Mata-Toledo & Cushman (2000, p. 255)

รีเลชั่น ORDER มีคีย์นอกคือ customerID ที่สามารถเชื่อมโยงไปยังรีเลชั่น CUSTOMER ดังนั้นจึงทำให้ทราบข้อมูลลูกค้าที่สั่งซื้อสินค้านั้นๆ ได้

ความเป็นไปได้กับการเปลี่ยนแปลงในข้อมูลตามกฎความคงสภาพของการอ้างอิงนี้มีความเป็นไปได้ 3 กรณี ดังนี้ (Pratt & Adamski, 2005, p. 277)

3.1 Restricted เป็นกฎที่ไม่อนุญาตให้มีการแก้ไขข้อมูลที่เป็นคีย์ ถ้าคีย์ที่ต้องการแก้ไขนั้นมีการอ้างอิงหรือใช้งานจากรีเลชั่นอื่น

3.2 Cascades เป็นกฎที่อนุญาติให้สามารถแก้ไขข้อมูลที่เป็นคีย์หลักได้ แต่จะทำการแก้ไขคีย์นอกในรีเลชั่นอื่นๆ ที่มีการอ้างถึงคีย์หลักนี้โดยอัตโนมัติ ตัวอย่างเช่น ได้มีการเปลี่ยนรหัสสินค้าในรีเลชั่น PRODUCT จากเดิม P0014 มาเป็น P0089 ดังนั้นรีเลชั่นอื่นๆ ที่อ้างอิงกับคีย์ productID ที่มีข้อมูลเดิมเป็น P0014 จะต้องถูกเปลี่ยนแปลงเป็น P0089 ทั้งหมด ซึ่งในที่นี้ก็คือ รีเลชั่น ORDER_LINE นั่นเอง กล่าวคือ แอตทริบิวต์ productID ในรีเลชั่น ORDER_LINE ที่มีค่าข้อมูลเดิมเป็น P0014 จะถูกเปลี่ยนเป็น P0089 ทั้งหมดโดยอัตโนมัติ

3.3 Nullify เป็นกฎที่อนุญาตให้บันทึกค่า null ให้กับคีย์นอกได้ เช่น ในรีเลชั่น *Order_Line* เป็นรีเลชั่นการสั่งซื้อสินค้าของลูกค้า สมมติว่ามีสินค้าชิ้นหนึ่งยังไม่ได้มีการกำหนด

รหัสสินค้า เนื่องจากเป็นสินค้าชนิดใหม่ แต่กฎนี้จะอนุญาตให้ใส่ค่า null แก่รหัสสินค้า (productID) ที่เป็นคีย์นอกได้

และจากกฎเกณฑ์ทั้ง 3 สามารถนำไปประยุกต์ใช้เพื่อจัดการกับข้อมูลด้วยการเขียนชุดคำสั่งดังภาพที่ 6.3

```
CREATE TABLE  ORDER_LINE
    ( orderID                    CHAR (5)                NOT NULL,
      productID                  CHAR (5)                NOT NULL,
      quantity                   INT
PRIMARY  KEY ( orderID, productID ),
FOREIGN KEY  (orderID)       REFERENCES  ORDER ( orderID ) ,
FOREIGN KEY  (productID)     REFERENCES  PRODUCT ( productID )
SET   NULL   on  update  CASCADE );
```

ภาพที่ 6.3 ตัวอย่างการใช้ชุดคำสั่งเพื่อกำหนด Referential Integrity
ที่มา : Pratt & Adamski (2005, p. 278)

ขั้นตอนการแปลงแบบจำลอง E-R มาเป็นรีเลชั่น

กิจกรรมของการออกแบบฐานข้อมูลเชิงตรรกะ ก็คือการดำเนินการแปลงแบบจำลอง E-R ที่ได้พัฒนาจากการออกแบบฐานข้อมูลเชิงแนวคิดให้มาอยู่ในรูปแบบของรีเลชั่นสคีมา

การแปลงภาพ E-R มาเป็นรีเลชั่น เครื่องมืออย่างเคสทูลมักผนวกฟังก์ชันดังกล่าวนี้มาให้ด้วย แต่อย่างไรก็ตาม ก็ถือว่าเป็นสิ่งที่สำคัญที่ควรทำความเข้าใจในกระบวนการแปลงแบบจำลอง E-R มาเป็นรีเลชั่นด้วยตนเอง รวมถึงบางกรณีอาจใช้เครื่องมืออย่างเคสทูลเข้ามาใช้งานร่วมด้วย ทั้งนี้ด้วยเหตุผล 3 ประการคือ (Teorey, 2006, p. 192)

1. เคสทูลบางตัวไม่สามารถแปลงแบบจำลองข้อมูลที่มีความสัมพันธ์ซับซ้อนได้ เช่น ความสัมพันธ์ในรูปแบบของ Ternary และความสัมพันธ์แบบ Supertype/Subtype ซึ่งหากเกิดกรณีดังกล่าวก็สามารถทำการแปลงได้ด้วยตนเองด้วยกระบวนการทำด้วยมือ

2. การมีหนทางการแก้ไขปัญหามากกว่าหนึ่งทางย่อมเป็นสิ่งที่ดี กล่าวคือ ในบางครั้งอาจนำเคสทูลมาใช้ ในขณะที่บางครั้งอาจทำด้วยมือ ทั้งนี้ขึ้นอยู่กับความเหมาะสมในแต่สถานการณ์

3. ในกรณีที่ต้องการมาตรฐานและคุณภาพของงาน ก็สามารถใช้เคสทูลเพื่อตรวจสอบแบบจำลองที่สร้างขึ้น ทั้งนี้ความถูกต้องและความต้องการคุณภาพงานมากขึ้น

เนื้อหาต่อไปนี้จะเป็นตัวอย่างขั้นตอนในการแปลงแบบจำลอง E-R เป็นรีเลชั่น แต่ก่อนที่จะเข้าสู่ขั้นตอนดังกล่าว ขอทบทวนความเข้าใจเกี่ยวกับเอนทิตีต่อไปนี้เสียก่อน

1. เอนทิตีปกติ (strong entity) คือเอนทิตีที่เกิดขึ้นได้ด้วยตัวเอง เป็นอิสระโดยไม่ขึ้นกับเอนทิตีใด สัญลักษณ์ที่ใช้คือรูปสี่เหลี่ยมผืนผ้า โดยสามารถเรียก strong entity ได้อีกชื่อหนึ่งว่า regular entity ดังภาพที่ 6.4 (a)

2. เอนทิตีอ่อนแอ (weak entity) ซึ่งเอนทิตีชนิดนี้จะไม่สามารถเกิดขึ้นได้ตามลำพังโดยอิสระเช่นเดี่ยวกับ strong entity กล่าวคือ จะเกิดขึ้นได้จากเอนทิตีหลัก ดังนั้น weak entity จะถูกลบออกไปทันทีเมื่อเอนทิตีหลักถูกลบ โดยสัญลักษณ์ที่ใช้คือรูปสี่เหลี่ยมผืนผ้าโดยมีเส้นเป็นเส้นคู่ ดังภาพที่ 6.4 (b)

3. Associative Entity เป็นรูปแบบที่เกิดขึ้นจากความสัมพันธ์ระหว่างเอนทิตีในลักษณะกลุ่มต่อกลุ่ม (many-to-many) โดยหากเกิดความสัมพันธ์ระหว่างเอนทิตีในลักษณะกลุ่มต่อกลุ่มแล้ว จะต้องเพิ่มเอนทิตีอีกตัวหนึ่งคั่นกลาง เพื่อใช้จัดเก็บเหตุการณ์ของข้อมูลที่เกิดขึ้น โดยสัญลักษณ์ของ associative entity จะเป็นรูปสี่เหลี่ยม และภายในเป็นสัญลักษณ์รูปสี่เหลี่ยมขนมเปียกปูน ดังภาพที่ 6.4 (c)

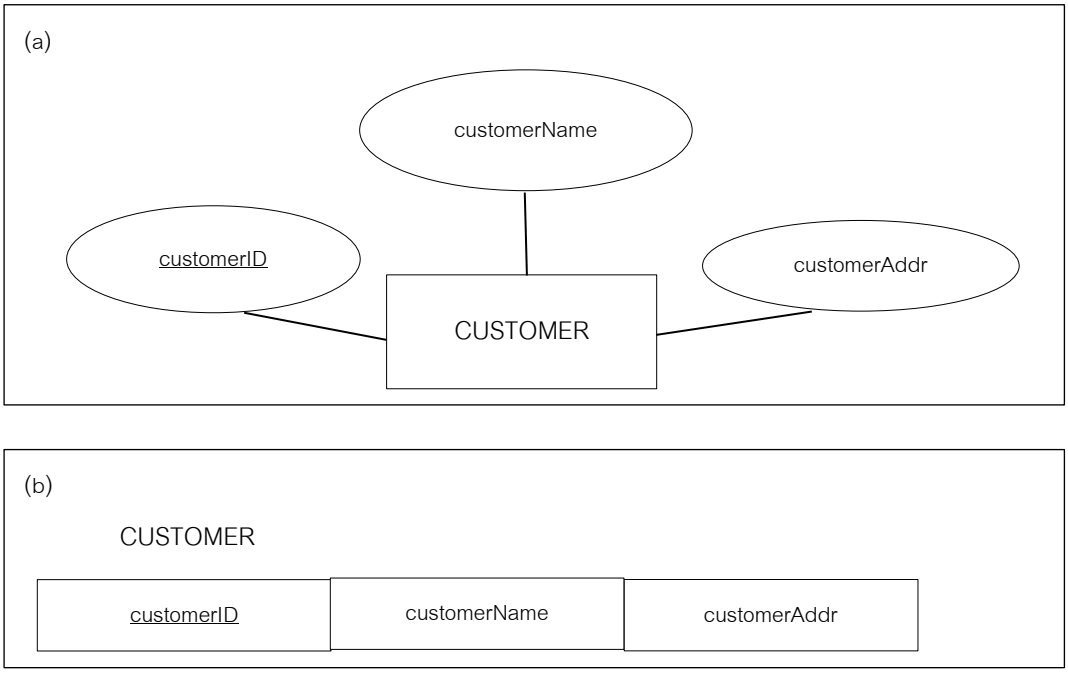
Strong Entity (a) | Weak Entity (b) | Associative Entity (c)

ภาพที่ 6.4 สัญลักษณ์ของเอนทิตีชนิดต่างๆ
ที่มา : Teorey (2006, p. 193)

ขั้นตอนที่ 1 : การแปลง Strong Entity

เอนทิตีแบบ Strong หรือแบบ Regular หรือต่อไปนี้จะเรียกว่า E เมื่อแปลงมาเป็นรีเลชั่น R ซึ่งประกอบด้วย Simple Attribute ที่มาจาก Entity E พร้อมทั้งมีการกำหนดคีย์หลักให้กับ

รีเลชั่น R ซึ่งตรงกับแอตทริบิวต์ที่กำหนดเป็นคีย์หลักใน E โดยภาพที่ 6.5 เป็นการแปลง Strong Entity มาเป็นรีเลชั่น



ภาพที่ 6.5 การแปลง Strong Entity ชื่อ CUSTOMER มาเป็นรีเลชั่น

(a) เอนทิตี CUSTOMER

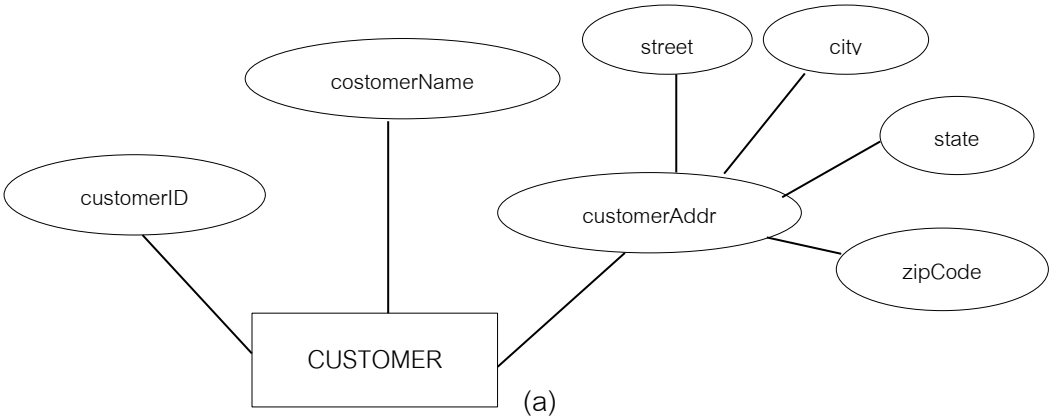
(b) รีเลชั่น CUSTOMER

ที่มา : Teorey (2006, p. 195)

Composite Attributes กรณีที่เอนทิตี E ประกอบด้วยแอตทริบิวต์ชนิด Composite ซึ่งเป็นแอตทริบิวต์ที่มีหลายองค์ประกอบ เมื่อแปลงเป็นรีเลชั่น R ให้นำแอตทริบิวต์แบบ Simple ที่บรรจุอยู่ในแอตทริบิวต์แบบ Composite มาเท่านั้น พิจารณาจากภาพที่ 6.6 (a) จะพบว่าแอตทริบิวต์ชื่อ customerAddr เป็นแอตทริบิวต์แบบ Composite ดังนั้น เมื่อนำมาแปลงเป็นรีเลชั่น R ก็นำเฉพาะแอตทริบิวต์ street, city, state และ zipCode มาเท่านั้น ดังภาพที่ 6.6 (b)

Multivalued Attributes กรณีที่เอนทิตี E บรรจุแอตทริบิวต์แบบ Multivalued ซึ่งเป็นแอตทริบิวต์ที่สามารถมีได้หลายค่า เมื่อแปลงเป็นรีเลชั่น R จะต้องเพิ่มรีเลชั่นใหม่ขึ้นมาอีกรีเลชั่นหนึ่ง โดยรีเลชั่นแรกให้บรรจุค่าแอตทริบิวต์ที่มีอยู่ใน E ทั้งหมด ยกเว้นแอตทริบิวต์ที่เป็นแบบ multivalued ส่วนอีกรีเลชั่นให้บรรจุ 2 แอตทริบิวต์ลงไปด้วยกัน โดยแอตทริบิวต์แรกคือ

แอตทริบิวต์ที่ใช้เป็นคีย์หลักที่อยู่ในรีเลชั่นแรก ส่วนแอตทริบิวต์ตัวที่สองก็คือแอตทริบิวต์ที่เป็น multivalued นั่นเอง



(a)

CUSTOMER

customerID	costomerName	street	city	state	zipCode

(b)

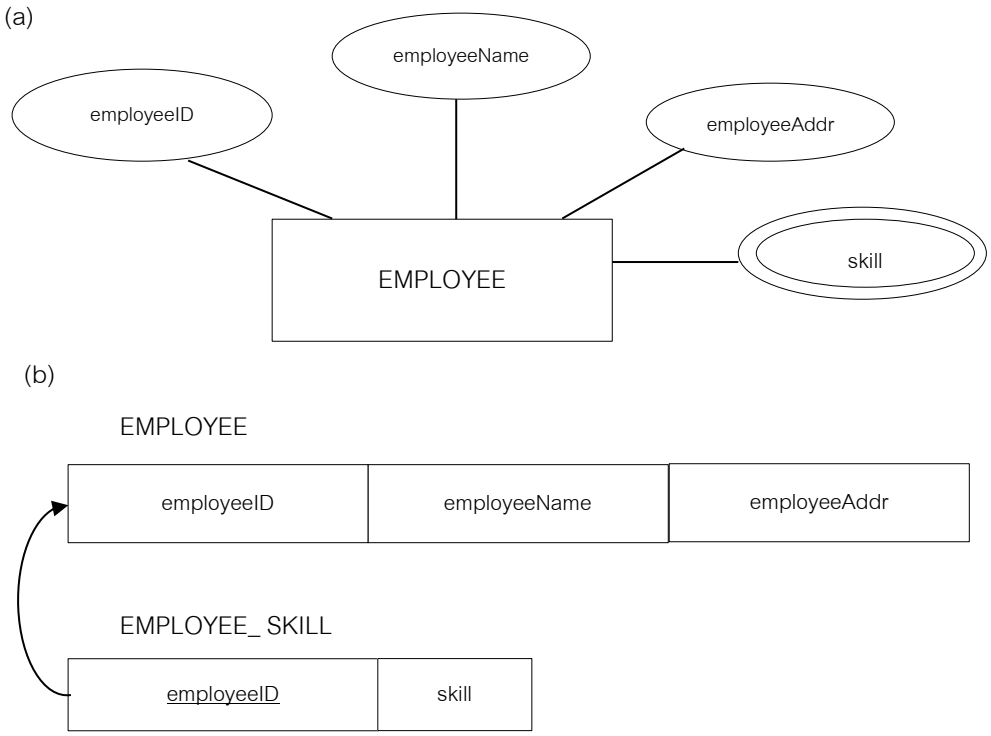
ภาพที่ 6.6 การแปลง Composite Attributes

(a) เอนทิตี CUSTOMER ที่มีบางแอตทริบิวต์เป็นแบบ Composite

(b) รีเลชั่น CUSTOMER ที่ได้

ที่มา : Teorey (2006, p. 196)

พิจารณาจากภาพที่ 6.7 (a) เอนทิตี EMPLOYEE ประกอบด้วยแอตทริบิวต์ชื่อ skill เป็นแอตทริบิวต์แบบ multivalued ครั้นเมื่อแปลงเป็นรีเลชั่นดังภาพที่ 6.7 (b) จะมีการเพิ่มรีเลชั่นใหม่เข้าไปคือ EMPLOYEE_ SKILL โดยรีเลชั่น EMPLOYEE จะจัดเก็บข้อมูลทั่วไปของพนักงาน (ยกเว้นข้อมูล skill) ในขณะที่รีเลชั่น EMPLOYEE_ SKILL จะจัดเก็บข้อมูลทักษะความรู้ของพนักงานแต่ละคน ที่หนึ่งคนอาจมีหลาย skill ก็ได้ โดยจะมี employeeID เป็นคีย์เชื่อมโยงไปยังเอนทิตี EMPLOYEE



ภาพที่ 6.7 การแปลงเอนทิตีที่มี Multivalued Attributes มาเป็นรีเลชั่น

(a) เอนทิตี EMPLOYEE ที่มีแอตทริบิวต์แบบ Multivalued บรรจุอยู่

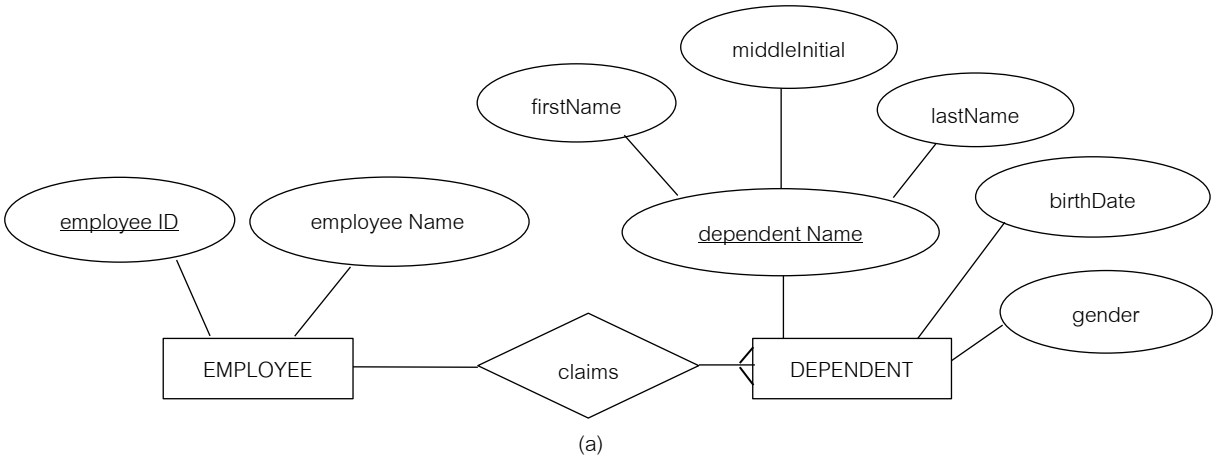
(b) การแปลงแอตทริบิวต์ที่เป็น Multivalued มาเป็นรีเลชั่น

ที่มา : Teorey (2006, p. 197)

ขั้นตอนที่ 2 : **การแปลง** Weak Entity

กรณีที่ต้องการแปลงเอนทิตีแบบ weak มาเป็นรีเลชั่น (ต่อไปนี้จะเรียกเอนทิตีชนิดนี้ว่า W) ซึ่งเป็นที่เข้าใจว่าเอนทิตีแบบ W นั้นจะเกิดได้จากเอนทิตีหลัก ซึ่งก็คือเอนทิตี E นั่นเอง

เมื่อนำมาแปลงเป็นรีเลชั่น ให้สร้างรีเลชั่น R ขึ้นมา ด้วยการบรรจุแอตทริบิวต์แบบ simple ทั้งหมดลงไป รวมถึงกรณีที่มีแอตทริบิวต์แบบ composite ก็ให้นำเพียงแอตทริบิวต์แบบ simple ที่บรรจุอยู่ในแอตทริบิวต์แบบ composite มาเท่านั้น นอกจากนี้ในรีเลชั่น R ที่สร้างขึ้น ก็ให้บรรจุแอตทริบิวต์ที่เป็นคีย์นอกลงไป เพื่อเชื่อมโยงกับเอนทิตี E ซึ่งเป็นเอนทิตีหลัก โดยคีย์หลักของรีเลชั่น R จะเป็นการนำคีย์หลักของเอนทิตี E มารวมกับคีย์บางตัวใน weak entity โดยพิจารณาจาก ภาพที่ 6.8 ซึ่งเป็นการแปลง weak entity มาเป็นรีเลชั่น



ภาพที่ 6.8 การแปลง Weak Entity มาเป็นรีเลชั่น

(a) เอนทิตี DEPENDENT ซึ่งเป็นเอนทิตีแบบ Weak

(b) ผลลัพธ์จากการแปลงเอนทิตีแบบ Weak มาเป็นรีเลชั่น

ที่มา : Teorey (2006, p. 199)

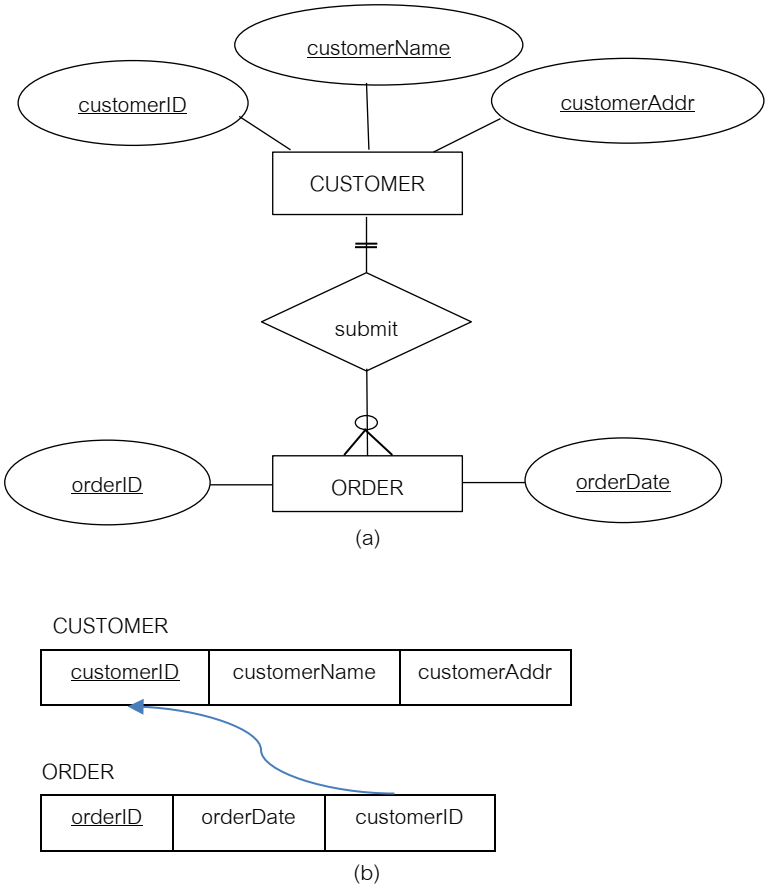
อย่างไรก็ตาม ก็มีอีกแนวทางหนึ่งในการกำหนดคีย์หลักให้กับเอนทิตี DEPENDENT ทั้งนี้เพื่อให้ง่ายต่อการอ้างอิงด้วยการบรรจุแอตทริบิวต์ใหม่เข้าไปหนึ่งตัว ซึ่งในที่นี้คือ dependentNo ก็สามารถเขียนอยู่ในรูปแบบของรีเลชั่นสคีมาได้ดังนี้

DEPENDENT (employeeID, DependentNo, firstName, middleInitial, lastName, birthdate, gender)

ขั้นตอนที่ 3 : การแปลงความสัมพันธ์แบบ Binary

การนำเสนอเกี่ยวกับความสัมพันธ์ของเอนทิตีขึ้นอยู่กับดีกรีของความสัมพันธ์ (Unary, Binary, Ternary) และคาร์ดินัลลิตี้ของความสัมพันธ์ โดยในที่นี้จะขอกล่าวถึงความสัมพันธ์แบบ Binary ก่อนเนื่องจากเป็นความสัมพันธ์ที่มีเอนทิตีเกี่ยวข้องกัน 2 เอนทิตี และจัดเป็นความสัมพันธ์ชนิดหนึ่งที่สามารถพบได้บ่อยที่สุด

การแปลงความสัมพันธ์แบบไบนารีชนิดหนึ่งต่อกลุ่ม (1:M) ขั้นตอนแรกของการแปลงความสัมพันธ์แบบไบนารีชนิดหนึ่งต่อกลุ่มคือ ให้สร้างรีเลชั่นขึ้นมาตามแต่ละเอนทิตี ในที่นี้กำหนดให้รีเลชั่น S คือเอนทิตีที่อยู่ฝั่ง many รีเลชั่นดังกล่าวจะต้องมีแอตทริบิวต์ที่เป็นคีย์นอกเพื่อเชื่อมโยงเข้ากับรีเลชั่น T ซึ่งก็คือเอนทิตีที่อยู่ฝั่ง one โดยคีย์นอกของรีเลชั่น S ก็คือคีย์หลักของรีเลชั่น T นั่นเอง



ภาพที่ 6.9 การแปลงความสัมพันธ์แบบ 1:M มาเป็นรีเลชั่น

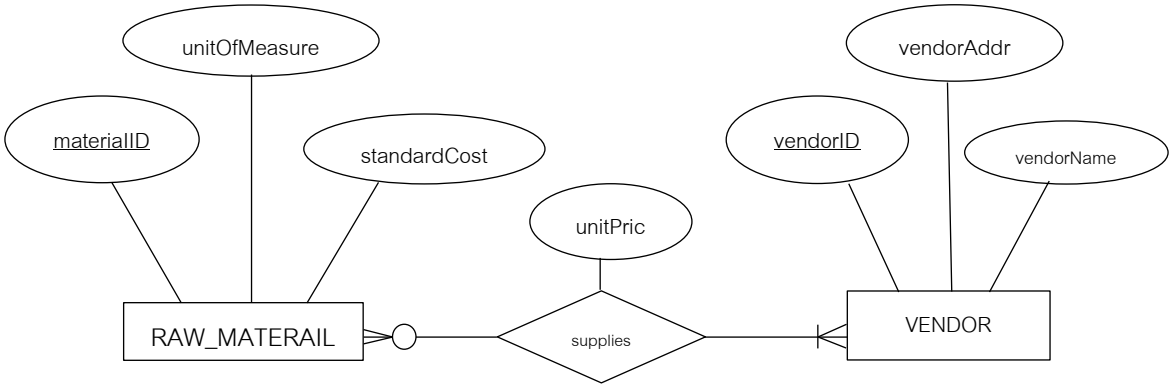
(a) ความสัมพันธ์ระหว่าง CUSTOMER กับ ORDER

(b) การแปลงเป็นรีเลชั่น

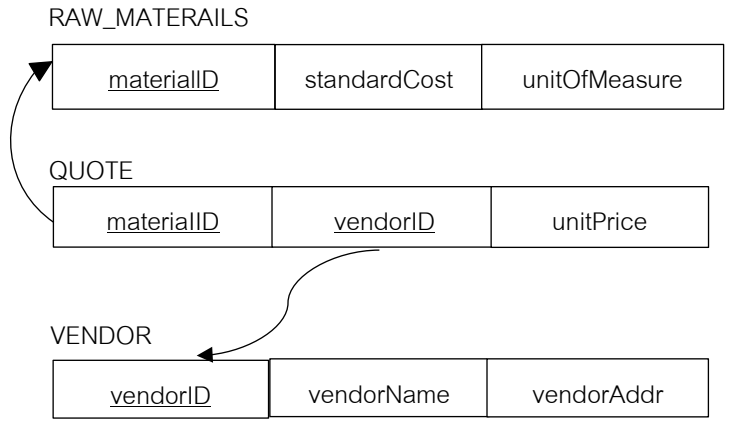
ที่มา : Teorey (2006, p. 200)

จากภาพที่ 6.9 (a) จะพบว่าเอนทิตี ORDER อยู่ฝั่ง many มีแอตทริบิวต์ orderID (คีย์หลัก) และ orderDate เมื่อแปลงเป็นรีเลชั่นดังภาพที่ 6.9 (b) รีเลชั่น ORDER จะต้องเพิ่มคีย์นอกเข้าไป ซึ่งในที่นี้คือ customerID เพื่อเชื่อมโยงกับ customerID ที่เป็นคีย์หลักของรีเลชั่น CUSTOMER

การแปลงความสัมพันธ์แบบไบนารีชนิดกลุ่มต่อกลุ่ม (M:N) เมื่อมีความสัมพันธ์แบบกลุ่มต่อกลุ่มเกิดขึ้นให้ดำเนินการสร้างรีเลชั่นใหม่ขึ้นมา โดยรีเลชั่นใหม่ที่สร้างขึ้นมานี้จะนำคีย์หลักของรีเลชั่น R และ รีเลชั่น S มาใช้เป็นคีย์นอก จากนั้นก็รวมคีย์นอกทั้งสองมาเป็นคีย์หลักของตัวเอง



(a)



(b)

ภาพที่ 6.10 การแปลงความสัมพันธ์แบบ M:N มาเป็นรีเลชั่น

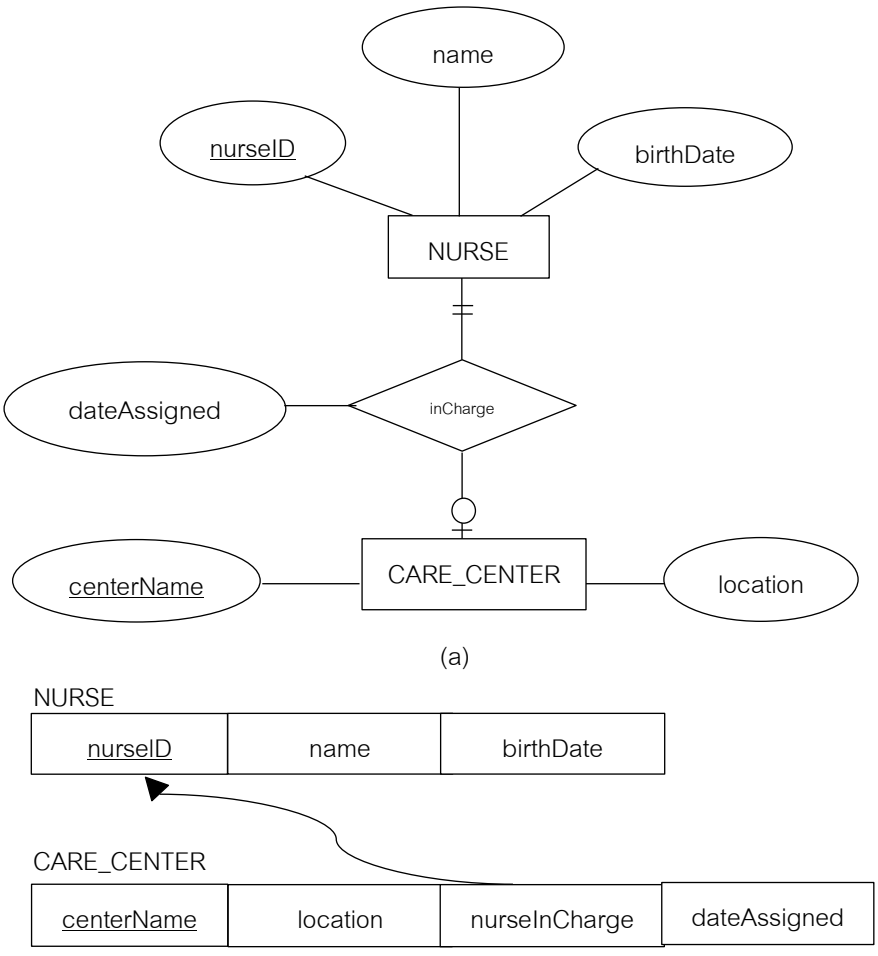
(a) เอนทิตีที่มีความสัมพันธ์แบบหนึ่งต่อกลุ่ม

(b) รีเลชั่นที่ได้จากการแปลง

ที่มา : Teorey (2006, p. 201)

พิจารณาจากภาพที่ 6.10 (a) จะพบว่าเอนทิตี RAW_MATERIALS กับ VENDOR มีความสัมพันธ์แบบกลุ่ม เมื่อแปลงเป็นรีเลชั่นดังรูปที่ 6.10 (b) ก็จะเพิ่มรีเลชั่นใหม่คั่นระหว่างกลางคือรีเลชั่น QUOTE โดยรีเลชั่น QUOTE จะมีคีย์นอกซึ่งเป็นคีย์หลักของ RAW_MATERIALS และ VENDOR คีย์หลักของรีเลชั่น QUOTE ก็คือคีย์นอกทั้งสองมารวมกัน

การแปลงความสัมพันธ์แบบไบนารีชนิดหนึ่งต่อหนึ่ง (1:1) สำหรับการแปลงความสัมพันธ์แบบไบนารีชนิดหนึ่งต่อหนึ่ง สามารถใช้หลักเดียวกันกับการแปลงความสัมพันธ์แบบหนึ่งต่อกลุ่มที่ผ่านมา ซึ่งเป็นไปดังภาพที่ 6.11 ต่อไปนี้

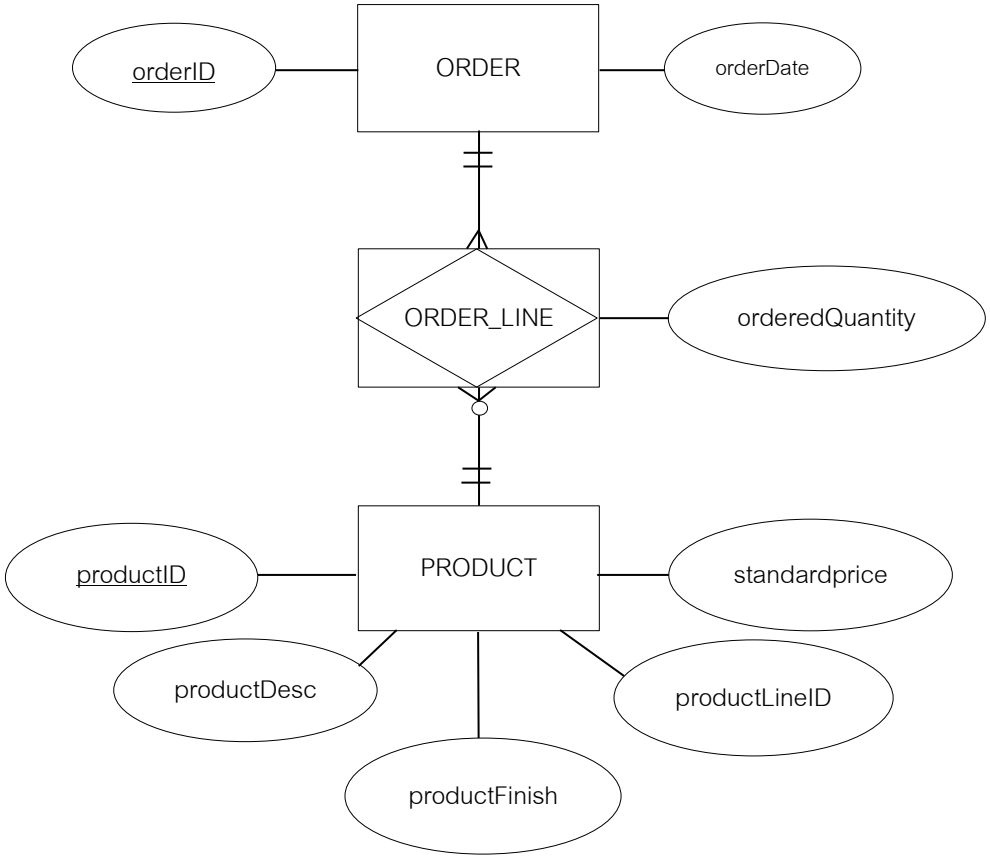


ภาพที่ 6.11 การแปลงความสัมพันธ์แบบ 1:1 มาเป็นรีเลชั่น

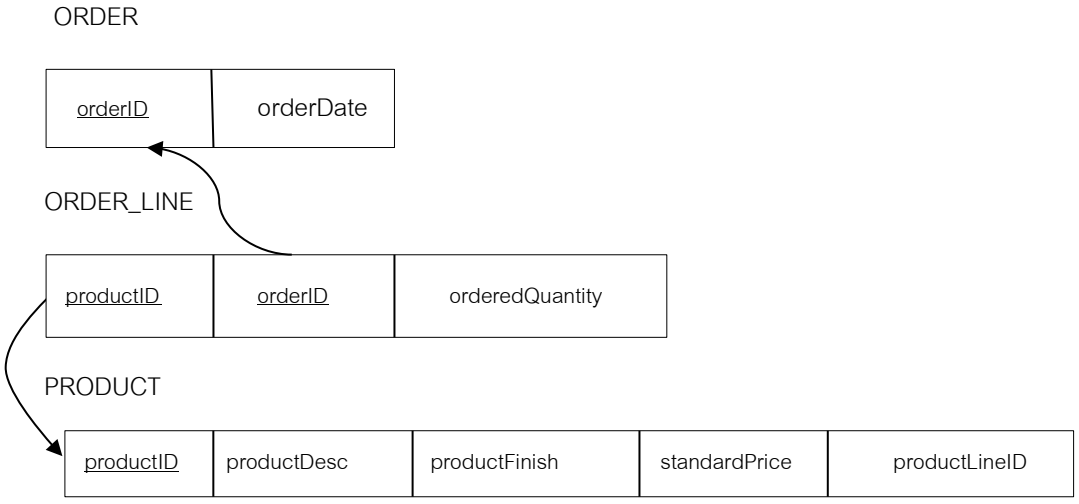
(a) เอนทิตีที่มีความสัมพันธ์แบบหนึ่งต่อหนึ่ง

(b) รีเลชั่นที่ได้จากการเปลี่ยนแปลง

ที่มา : Teorey (2006, p. 203)



(a)



(b)

ภาพที่ 6.12 การแปลงเอนทิตีชนิด Associative มาเป็นรีเลชั่น

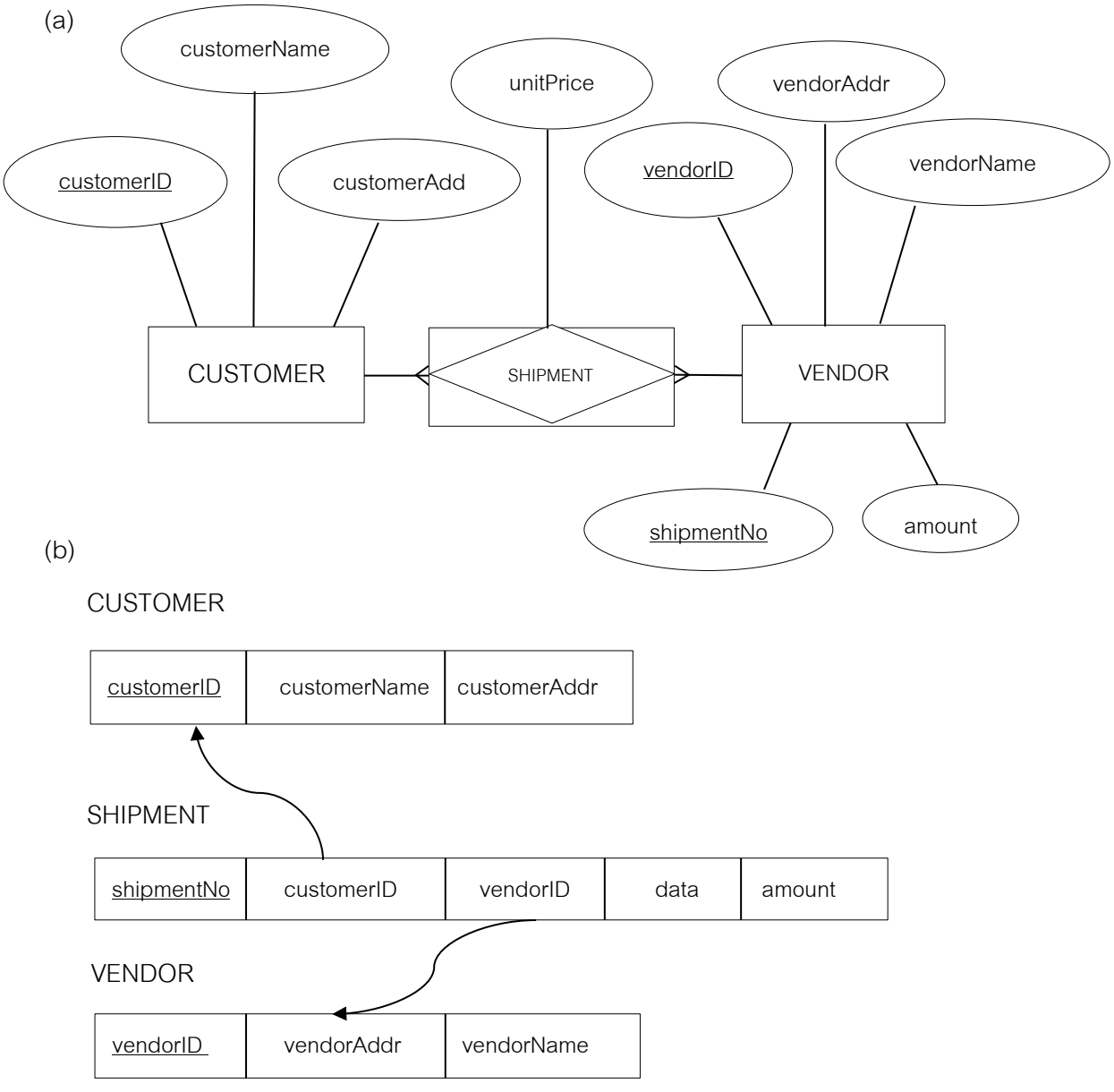
(a) เอนทิตี ORDER_LINE เป็นเอนทิตีแบบ Associative ที่ไม่ได้ระบุคีย์หลักมาให้

(b) รีเลชั่นทั้งสามที่ได้มากจากการแปลง

ที่มา : Teorey (2006, p. 204)

ขั้นตอนที่ 4 : การแปลงเอนทิตีชนิด Associative

เอนทิตี Associative จะเกิดขึ้นจากความสัมพันธ์แบบกลุ่มต่อกลุ่ม ซึ่งจะต้องเพิ่มรีเลชั่นใหม่คั่นกลางระหว่างเอนทิตีทั้งสอง โดยจะเห็นได้ว่าวิธีดังกล่าวจะเหมือนกับการแปลงความสัมพันธ์แบบไบนารีชนิดกลุ่มต่อกลุ่มที่ผ่านมานั่นเอง แบบจำลอง E-R ที่นำเสนอเอนทิตีชนิด Associative นั้น สามารถมีได้ 2 กรณีด้วยกันคือ



ภาพที่ 6.13 การแปลงเอนทิตีชนิด Associative ที่มีการระบุคีย์หลัก

(a) เอนทิตี SHIPMENT เป็นเอนทิตีแบบ Associative ที่มีการระบุคีย์หลักมาให้

(b) รีเลชั่นทั้งสามที่ได้จากเปลี่ยนแปลง

ที่มา : Teorey (2006, p. 205)

กรณีไม่ได้ระบุคีย์หลัก หากผู้ออกแบบไม่ได้ระบุคีย์หลักมาให้ ซึ่งเป็นไปดังภาพที่ 6.12 (a) จะพบว่าเอนทิตี Associative ในที่นี้คือ ORDER_LINE ซึ่งไม่ได้ระบุคีย์หลักมาให้ ดังนั้นให้คิดว่าคีย์หลักปกติที่สมควรคือ คีย์หลักของเอนทิตีทั้งสองมารวมกันนั่นเอง (ORDER และ PRODUCT) และยังใช้เป็นคีย์นอกเพื่อเชื่อมโยงกับรีเลชั่นทั้งสอง ซึ่งเป็นไปดังภาพที่ 6.12 (b)

กรณีมีการระบุคีย์หลัก ในบางครั้งผู้ออกแบบได้ระบุคีย์หลักบนเอนทิตีแบบ Associative มาให้ด้วย ดังภาพที่ 6.13 (a) เอนทิตี SHIPMENT ซึ่งเป็นเอนทิตีแบบ Associative มีแอตทริบิวต์ shipmentNo เป็นคีย์หลัก ทั้งนี้สืบมาจากเหตุผลดังกล่าวต่อไปนี้คือ

เอนทิตี Associative ดังกล่าว โดยธรรมชาติแล้วสมควรมีคีย์หลักเพื่อการอ้างอิง

คีย์หลักปกติ (ที่เกิดจากนำคีย์หลักของ 2 รีเลชั่นมารวมกัน) อาจไม่มีความหมายเป็นเอกลักษณ์จึงทำให้เกิดการซ้ำกันของทูเพิลได้

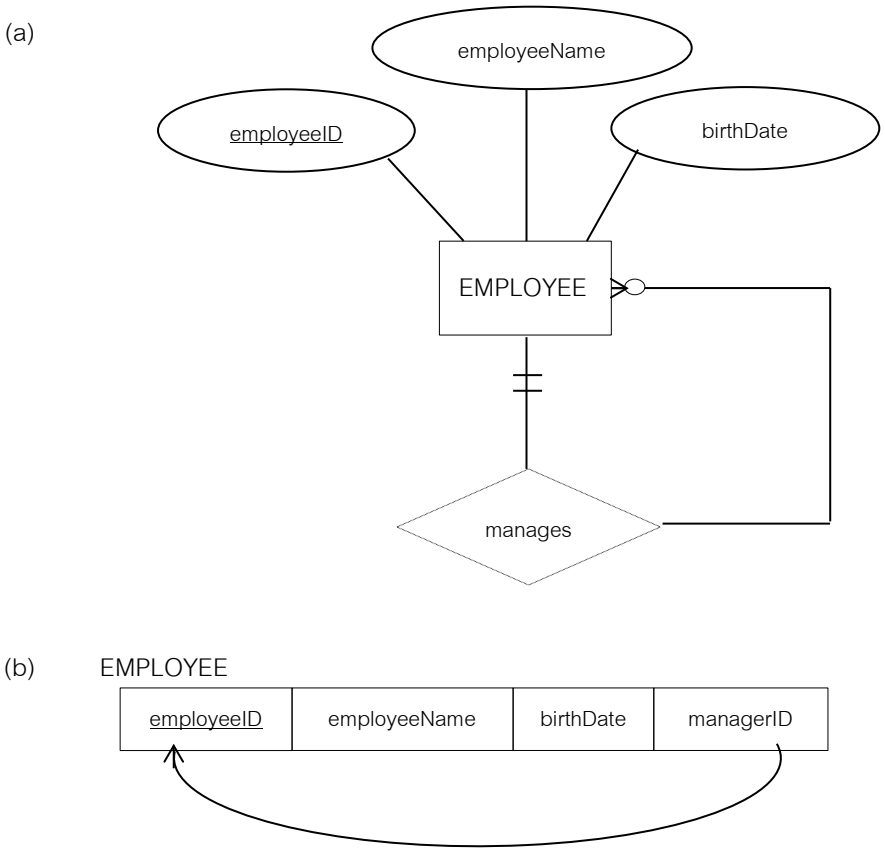
ดังนั้น กระบวนการเปลี่ยนแปลงเป็นรีเลชั่นสำหรับกรณีดังกล่าวก็สามารถปรับเพียงเล็กน้อยด้วยการกำหนดคีย์หลักให้กับรีเลชั่น Associative ขึ้นมา ส่วนคีย์หลักปกติเกิดจากรีเลชั่นทั้งสอง ก็จะนำไปใช้เป็นคีย์นอกเท่านั้น เพื่อใช้ในการเชื่อมโยงกับรีเลชั่นดังกล่าว ซึ่งเป็นดังรูปที่ 6.13 (b)

ขั้นตอนที่ 5 : การแปลงความสัมพันธ์แบบ Unary

ความสัมพันธ์แบบ Unary เป็นความสัมพันธ์ระหว่างเอนทิตีเดียวหรือเรียกว่าความสัมพันธ์แบบรีเคอร์ซีพ ซึ่งจะมีทั้งความสัมพันธ์แบบหนึ่งต่อกลุ่มและกลุ่มต่อกลุ่ม

ความสัมพันธ์แบบ Unary ชนิดหนึ่งต่อกลุ่ม ขั้นตอนแรกของการแปลงรีเลชั่นของความสัมพันธ์ Unary ชนิดหนึ่งต่อกลุ่ม ให้ทำการเพิ่มแอตทริบิวต์ที่ใช้เป็นคีย์นอกเข้าไป โดยค่าข้อมูลคีย์นอกคือค่าข้อมูลของคีย์หลักนั่นเอง ที่ใช้สำหรับเชื่อมโยงบนรีเลชั่นเดียวกันในลักษณะรีเคอร์ซีพ (คีย์นอกจะต้องมีโดเมนเดียวกันคีย์หลัก)

พิจารณาจากภาพที่ 6.14 (a) ที่แสดงถึงความสัมพันธ์แบบ Unary ชนิดหนึ่งต่อกลุ่ม โดยความสัมพันธ์ชื่อ <*manages*> ได้เชื่อมโยงความสัมพันธ์ให้กับพนักงานแต่ละคนในองค์กรว่าพนักงานเหล่านั้นถูกดูแลโดยใคร โดยผู้ดูแลก็คือผู้จัดการซึ่งก็คือพนักงานคนหนึ่งนั่นเองและเมื่อนำมาแปลงเป็นรีเลชั่นก็จะเป็นดังภาพที่ 6.14 (b)



ภาพที่ 6.14 การแปลงความสัมพันธ์แบบ Unary ชนิดหนึ่งต่อกลุ่มมาเป็นรีเลชั่น

(a) เอนทิตี EMPLOYEE กับความสัมพันธ์แบบ Unary

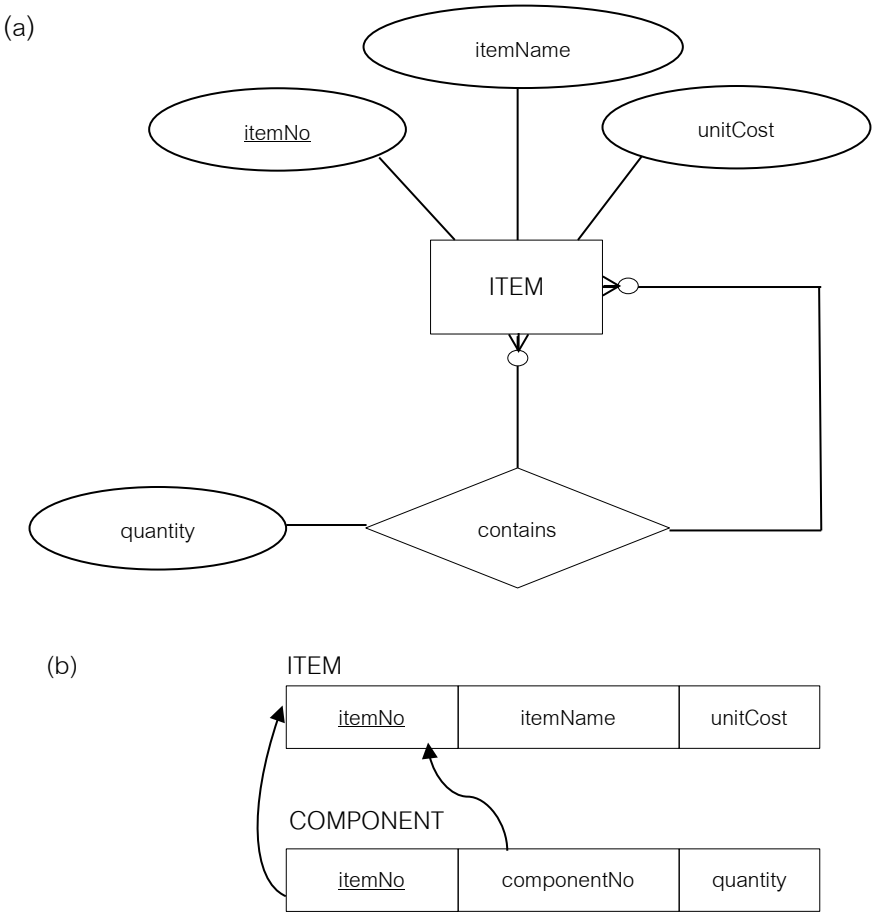
(b) รีเลชั่น EMPLOYEE กับคีย์นอกที่เชื่อมโยงแบบรีเคอร์ซีพ

ที่มา : Teorey (2006, p. 206)

ความสัมพันธ์แบบ Unary ชนิดกลุ่มต่อกลุ่ม สำหรับความสัมพันธ์ในรูปแบบดังกล่าวจะต้องสร้างรีเลชั่น 2 รีเลชั่นด้วยกัน โดยรีเลชั่นแรกจะมีคีย์หลักพร้อมแอตตริบิวต์ที่เกี่ยวข้อง ส่วนอีกรีเลชั่นหนึ่งก็คือรีเลชั่นแบบ Associative ที่ประกอบด้วย 2 แอตทริบิวต์ที่ใช้เป็นคีย์หลัก ซึ่งแอตทริบิวต์ทั้งสองก็คือค่าข้อมูลเดียวกันกับคีย์หลักในเอนทิตีแรก แต่จะต้องตั้งชื่อให้แตกต่างกัน ซึ่งเป็นไปดังภาพที่ 6.15

ดังนั้นในการเรียกดูข้อมูล ในกรณีที่ต้องการทราบว่า component อะไรบ้างในหมายเลข item นั้นๆ เช่น ต้องการทราบว่ามีรายการ component ใดบ้างที่อยู่ใน item หมายเลข 100 ก็สามารถเขียนคำสั่ง SQL ได้ดังนี้

```
SELECT  componentNo, quantity
        FROM COMPONENT
        WHERE itemNo = 100;
```



ภาพที่ 6.15 การแปลงความสัมพันธ์แบบ Unary ชนิดกลุ่มต่อกลุ่มมาเป็นรีเลชั่น

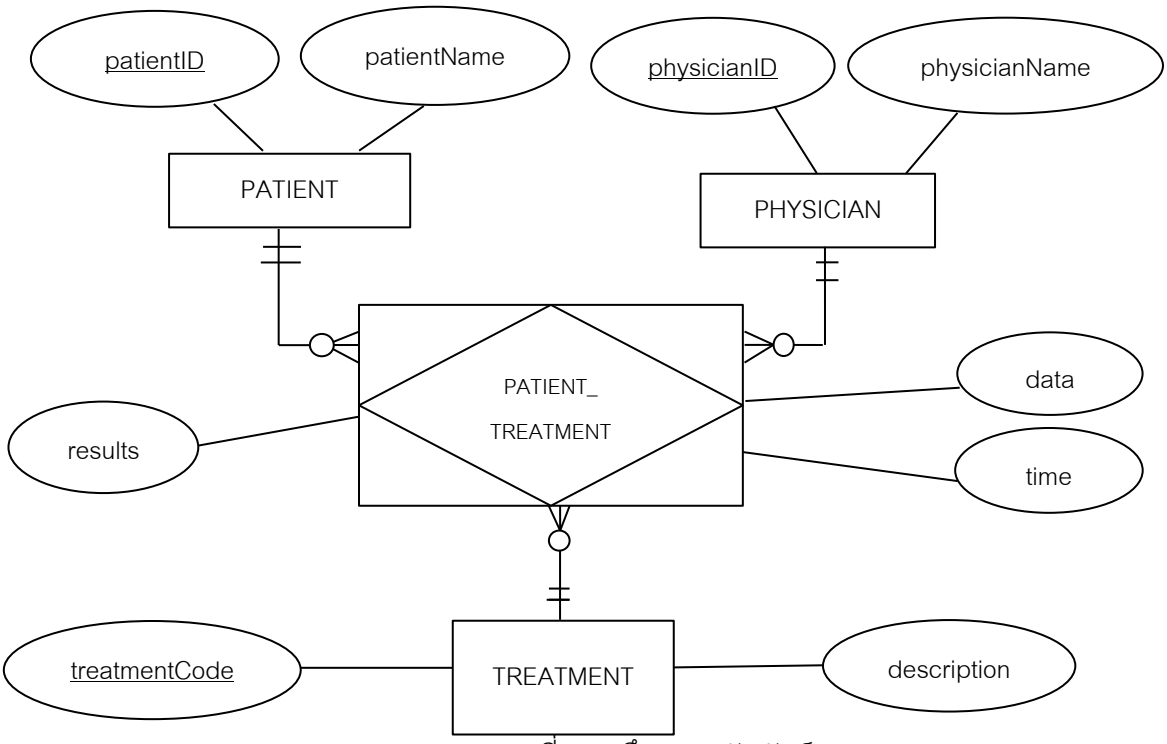
(a) เอนทิตี ITEM กับความสัมพันธ์แบบ Unary

(b) รีเลชั่น COMPONENT กับคีย์นอกที่เชื่อมโยงแบบรีเคอร์ซีพ

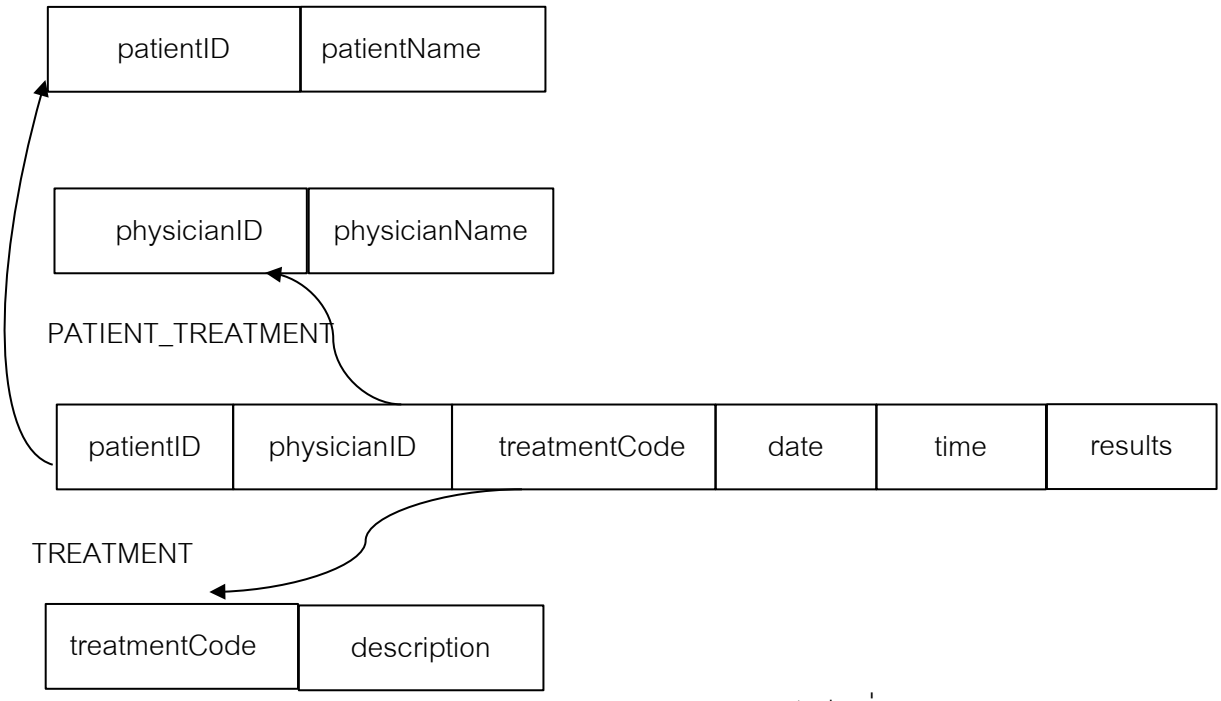
ที่มา : Teorey (2006, p. 207)

ขั้นตอนที่ 6 : การแปลงความสัมพันธ์แบบ Ternary

ความสัมพันธ์แบบ Ternary ซึ่งประกอบด้วยความสัมพันธ์ระหว่างเอนทิตี 3 ตัวด้วยกัน และในบทนี้เราจะทำการแปลงความสัมพันธ์แบบ Ternary มาเป็นรีเลชั่น



(a) แผนภาพ E-R ที่แสดงถึงความสัมพันธ์แบบ Ternary



(b) การแปลงความสัมพันธ์แบบ Ternary มาเป็นรีเลชั่น

ภาพที่ 6.16 การแปลงความสัมพันธ์แบบ Ternary มาเป็นรีเลชั่น

ที่มา : Teorey (2006, p. 208)

ขั้นตอนแรก จะต้องแปลงเอนทิตีแบบ Associative ให้มีคีย์ที่สามารถเชื่อมโยงไปยังเอนทิตีทั้ง 3 ให้ได้ก่อน โดยในที่นี้จะสร้างรีเลชั่นใหม่ขึ้นมาคือรีเลชั่นแบบ Associative ที่จะมีคีย์หลักแบบปกติที่มาจากรีเลชั่นทั้งสาม (ในบางกรณีอาจเพิ่มแอตทริบิวต์ใหม่เข้าไปเพื่อใช้เป็นคีย์หลัก)

จากภาพที่ 6.16 (a) จะเห็นได้ว่าแผนภาพ E-R ดังกล่าวจะประกอบด้วยเอนทิตี PATIENT, PHYSICAN และ TREATMENT โดยจะมีเอนทิตีชื่อ PATIENT_TREATMENT เป็นเอนทิตีแบบ Associative ซึ่งมีเพียงแอตทริบิวต์ results, date และ time ดังนั้นเมื่อแปลงเป็นรีเลชั่นดังรูปที่ 6.16 (b) จะพบว่ารีเลชั่นชื่อ PATIENT_TREATMENT ภายในจะบรรจุไปด้วยแอตทริบิวต์ที่เป็นคีย์หลักของรีเลชั่นทั้งสาม ทั้งนี้ก็เพื่อให้สามารถเชื่อมโยงไปยังรีเลชั่นดังกล่าวได้นั่นเอง

ขั้นตอนที่ 7 : การแปลงความสัมพันธ์แบบ Supertype และ Subtype

ปกติแล้วแบบจำลองข้อมูลเชิงสัมพันธ์จะไม่สนับสนุนความสัมพันธ์แบบ Supertype/Subtype แต่อย่างไรก็ตาม ก็ยังมีแนวทางในการออกแบบเพื่อรับรองความสัมพันธ์ดังกล่าวได้ต่อไปนี้

1. ให้สร้างรีเลชั่นแยกออกต่างหาก สำหรับ Supertype และแต่ละ Subtype

2. แอตทริบิวต์ของ Subtype จะได้รับการถ่ายทอดจากรีเลชั่นที่เป็น Supertype รวมถึงแอตทริบิวต์ที่เป็นคีย์หลักด้วย

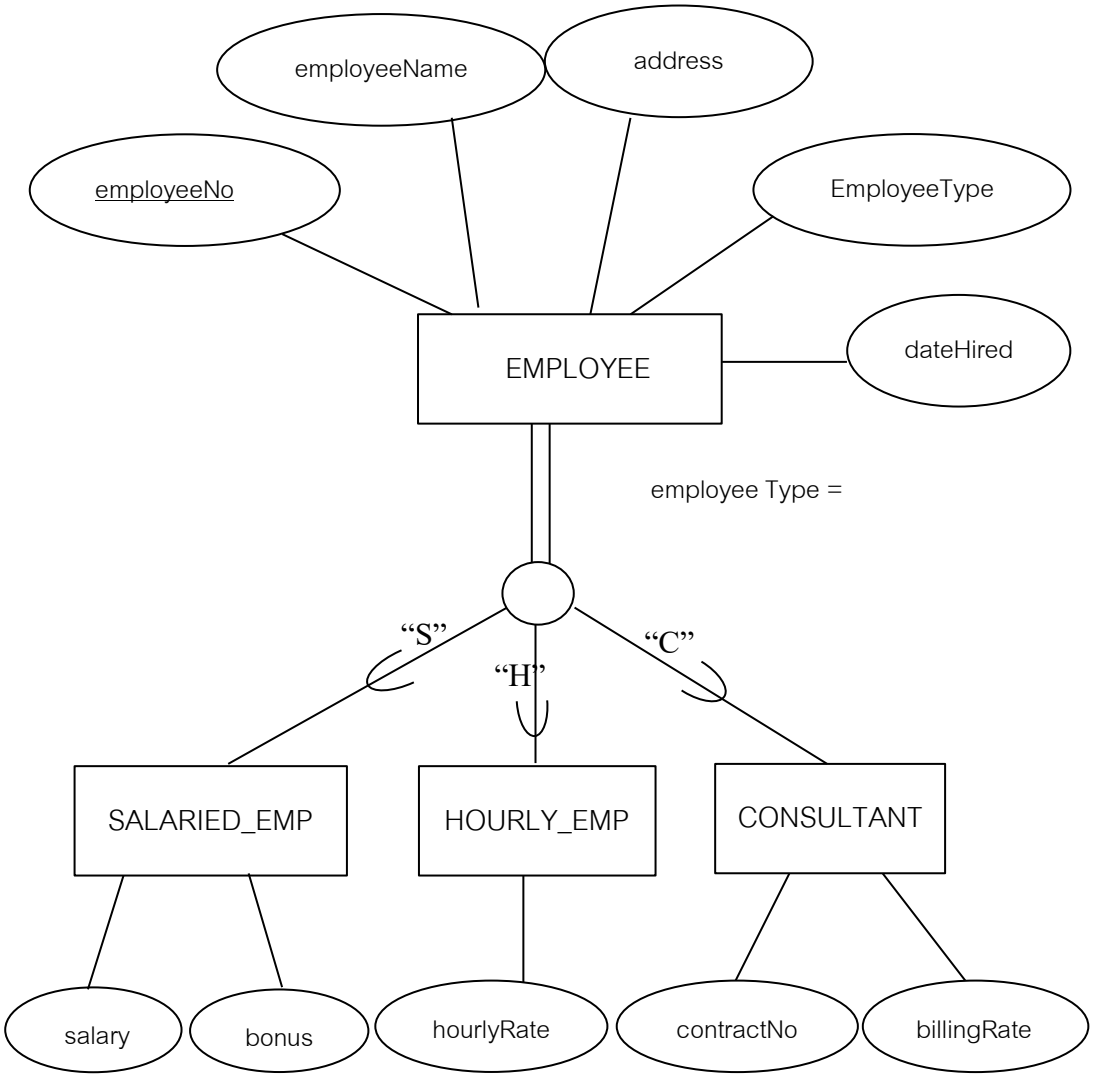
3. กำหนดคีย์หลักให้กับแต่ละรีเลชั่นที่เป็น Subtype โดยจะต้องมีแอตทริบิวต์ที่แตกต่างกัน เพื่อใช้ระบุความแตกต่างของแต่ละ Subtype

4. กำหนด Subtype Discriminator ให้กับ Supertype

ภาพที่ 6.17 (a) เป็นแบบจำลอง EER ที่ประกอบด้วยเอนทิตีที่เป็น Supertype/Subtype และภาพที่ 6.17 (b) เป็นผลลัพธ์ของรีเลชั่นที่ผ่านการแปลงเรียบร้อยแล้ว

และกรณีที่ต้องการแสดงข้อมูลเฉพาะพนักงานประจำ ก็สามารถเป็นคำสั่ง SQL ได้ดังนี้

```
SELECT *
FROM EMPLOYEE AS E , SALARIED_EMP AS S
WHERE E.employeeNo = S.employeeNo;
```



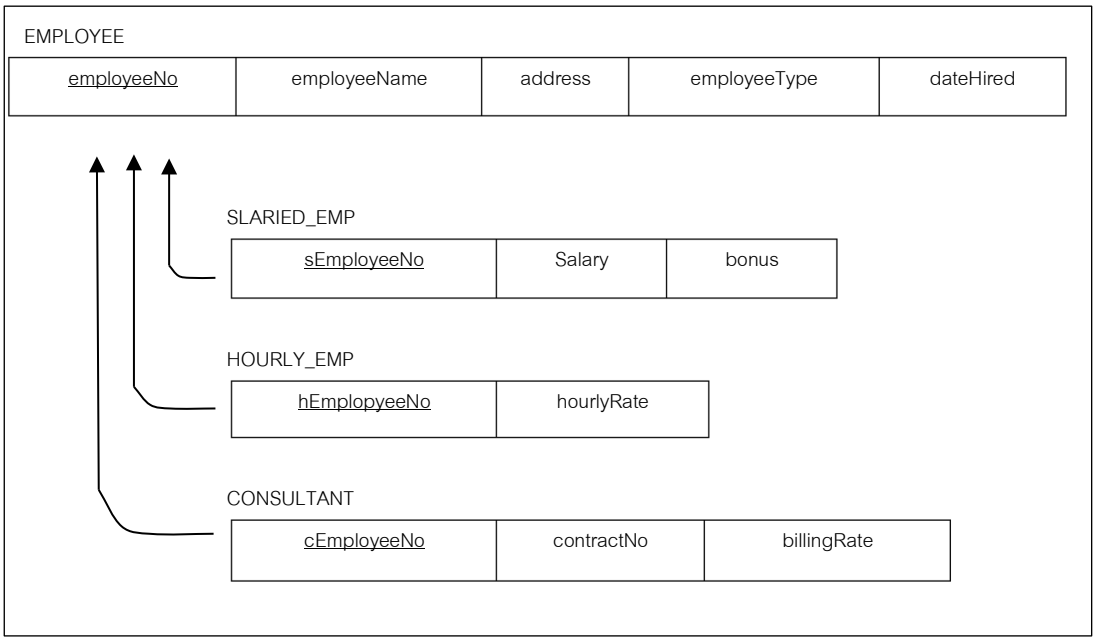
(a)

ภาพที่ 6.17 แผนภาพ EER และการแปลงเป็นรีเลชั่น

(a) ความสัมพันธ์แบบ Supertype/Subtype

(b) การแปลงความสัมพันธ์แบบ Supertype/Subtype เป็นรีเลชั่น

ที่มา : Teorey (2006, p. 210)



(b)

ภาพที่ 6.17 **(ต่อ)** แผนภาพ EER และการแปลงเป็นรีเลชั่น

(a) ความสัมพันธ์แบบ Supertype/Subtype

(b) การแปลงความสัมพันธ์แบบ Supertype/Subtype เป็นรีเลชั่น

ที่มา : Teorey (2006, p. 211)

การทำนอร์มัลไลเซชั่น

เมื่อมีการออกแบบฐานข้อมูลเพื่อใช้งานในองค์กร วัตถุประสงค์หลักก็คือการสร้างตัวแทนของข้อมูลที่มีความถูกต้อง ทั้งในส่วนของความสัมพันธ์ระหว่างข้อมูลและข้อบังคับบนข้อมูล โดยแนวทางในการช่วยให้บรรลุถึงวัตถุประสงค์ดังกล่าว จำเป็นต้องใช้เทคนิคการออกแบบฐานข้อมูลเข้าช่วย ซึ่งเทคนิคหนึ่งที่เราได้กล่าวมาแล้วคือแบบจำลองข้อมูลหรือแผนภาพ E-R และเนื้อหาต่อไปนี้จะนำเสนออีกเทคนิคหนึ่งที่นำมาใช้ในการออกแบบฐานข้อมูล คือ **การนอร์มัลไลเซชั่น** (Normalization)

การนอร์มัลไลเซชั่นเป็นกระบวนการที่มีแบบแผนซึ่งใช้สำหรับจัดการกลุ่มแอตทริบิวต์ที่รวมเข้าด้วยกันภายในรีเลชั่น ตัวอย่างเช่น ได้ดำเนินการนอร์มัลไลเซชั่นเพื่อแปลงตาราง

EMPLOYEE_BRANCH ซึ่งภายในตารางดังกล่าวจะมีข้อมูลซ้ำซ้อนอยู่ ดังนั้นจึงได้แตกตารางใหม่ออกมา ซึ่งประกอบด้วยตาราง EMPLOYEE กับตาราง BRANCH (ภาพที่ 6.18 - 6.19) โดยภาพรวมของกระบวนการนอร์มัลไลเซชั่นจะดำเนินการในช่วงต่อไปนี้

1. ในช่วงของการออกแบบจำลองเชิงแนวคิด ได้มีการนอร์มัลไลเซชั่นแบบจำลอง E-R ระหว่างการสร้างแบบจำลองดังกล่าว

2. ในช่วงของการออกแบบฐานข้อมูลเชิงตรรกะ ซึ่งจะมีกระบวนการแปลงแบบจำลอง E-R มาเป็นรีเลชั่น ทั้งนี้ให้รีเลชั่นที่ออกแบบมามีโครงสร้างที่ดีและมีคุณภาพยิ่งขึ้น ดังนั้นจึงสามารถนำเทคนิคการนอร์มัลไลเซชั่นมาใช้เพื่อตรวจสอบโครงสร้างรีเลชั่นดังกล่าวได้

3. ในการปรับปรุงเปลี่ยนแปลงระบบเดิม ซึ่งอาจมีจำนวนตารางหรือรีเลชั่นอยู่มากมาย ทั้งนี้อาจมีความซ้ำซ้อนของข้อมูลปนอยู่ตามรีเลชั่นต่างๆ ซึ่งเป็นสาเหตุของข้อผิดพลาดในข้อมูล ดังนั้นจึงต้องขจัดข้อผิดพลาดเหล่านั้นออกจากการนอร์มัลไลเซชั่น (Date, 2013, p. 217)

จึงกล่าวโดยสรุปว่า การนอร์มัลไลเซชั่นเป็นกระบวนการนำโครงร่างของรีเลชั่นมาแตกเป็นรีเลชั่นต่างๆ ให้อยู่ในรูปที่เรียกว่ารูปแบบบรรทัดฐานหรือที่เรียกว่า Normal Form โดยมีเป้าหมายเพื่อให้รีเลชั่นที่ได้รับการออกแบบนั้นอยู่ในรูปแบบบรรทัดฐานระดับที่เหมาะสม

กระบวนการนอร์มัลไลเซชั่นได้ถูกพัฒนาขึ้นโดย E.F.Codd (1972) ซึ่งเป็นเทคนิคที่ใช้ในการวิเคราะห์รีเลชั่นให้อยู่ในรูปแบบของนอร์มัลฟอร์ม ซึ่งมีอยู่ 3 ระดับด้วยกันคือ

1. นอร์มัลฟอร์มระดับที่ 1 หรือเรียกว่า 1NF
2. นอร์มัลฟอร์มระดับที่ 2 หรือเรียกว่า 2NF
3. นอร์มัลฟอร์มระดับที่ 3 หรือเรียกว่า 3NF

นอกจากนี้ยังมีระดับที่ทำให้นอร์มัลฟอร์มระดับที่ 3 มีความแข็งแกร่งขึ้นกว่าเดิม เรียกว่า BCNF ซึ่งพัฒนาขึ้นโดย R.Boyce และ E.F.Codd โดยนอร์มัลฟอร์มทุกระดับจะตั้งอยู่บนพื้นฐานของฟังก์ชั่นการขึ้นต่อกันระหว่างแอตทริบิวต์ของรีเลชั่น

สำหรับนอร์มัลฟอร์มในระดับที่สูงขึ้นไปอีกที่อยู่ถัดจาก BCNF ก็ถูกพัฒนาขึ้นมาเพิ่มเติมคือ นอร์มัลฟอร์มระดับที่ 4 (4NF) และนอร์มัลฟอร์มระดับที่ 5 (5NF) ซึ่งพัฒนาโดย Fagin (1977,1979) อย่างไรก็ตามรูปแบบนอร์มัลฟอร์มระดับที่ 4 และระดับที่ 5 ในเชิงปฏิบัติเกิดขึ้นค่อนข้างน้อย โดยนอร์มัลฟอร์มระดับที่ 3 นั้น สามารถพบเห็นได้ทั่วไปกว่า 90 เปอร์เซ็นต์

วัตถุประสงค์ของการนอร์มัลไลเซชั่น (Chittayasothorn, 2007, p. 283)

1. **ลดเนื้อที่ในการจัดเก็บข้อมูล** กระบวนการนอร์มัลไลเซชั่น เป็นการออกแบบเพื่อลดความซ้ำซ้อนในข้อมูล ดังนั้นการลดความซ้ำซ้อนในข้อมูลย่อมสามารถลดเนื้อที่ในการจัดเก็บข้อมูลตามมาด้วย

2. **ลดปัญหาข้อมูลที่ไม่ถูกต้อง** เมื่อข้อมูลไม่มีความซ้ำซ้อน ในการปรับปรุงข้อมูลก็สามารถปรับปรุงข้อมูลได้จากแหล่งข้อมูลเพียงแหล่งเดียว จึงช่วยลดปัญหาการปรับปรุงข้อมูลไม่ถูกต้องได้ซึ่งหมายรวมถึงการลดปัญหาจากการเพิ่มข้อมูล ลบข้อมูล และการปรับปรุงข้อมูล

ต่อไปจะกล่าวถึงปัญหาความซ้ำซ้อนและข้อผิดพลาดจากการปรับปรุงข้อมูล แนวคิดฟังก์ชันการขึ้นต่อกัน และกระบวนการนอร์มัลไลเซชั่น ดังต่อไปนี้

1. ความซ้ำซ้อนและข้อผิดพลาดจากการปรับปรุงข้อมูล

สำหรับปัญหาด้านความซ้ำซ้อนและข้อผิดพลาดจากการปรับปรุงข้อมูล (data redundancy and update anomalies) แนวคิดหลักอันสำคัญของการออกแบบฐานข้อมูลเชิงสัมพันธ์คือ ในการรวมกลุ่มของแอตทริบิวต์เพื่อไปเป็นรีเลชั่นนั้น ขอให้เกิดความซ้ำซ้อนในข้อมูลน้อยที่สุด ทั้งนี้จะช่วยลดปัญหาการใช้พื้นที่จัดเก็บข้อมูลลดลงได้

ตัวอย่างปัญหาของความซ้ำซ้อนในข้อมูลได้แสดงไว้ในรีเลชั่น EMPLOY_BRANCH ดังภาพที่ 6.18 ต่อไปนี้

EMPLOYEE_BRANCH

empNo	empName	position	salary	branchNo	Address
SL21	ชูชัย สุขศรี	ผู้จัดการ	30000	B005	21 ถ.ห้วยแก้ว จ.เชียงใหม่
SG37	ศิริทัตร มณีจนทร์	ผู้ช่วย	12000	B003	143 ถ.วิภาวดีรังสิต จ.กรุงเทพฯ
SG14	สมศักดิ์ แซ่ตั้ง	ซูเปอร์ไวเซอร์	18000	B003	143 ถ.วิภาวดีรังสิต จ.กรุงเทพฯ
SA09	ปิยะฉัตร เอี่ยมสุข	ผู้ช่วย	9000	B007	56 ถ.พหลโยธิน จ.พิษณุโลก
SG05	พรรัตน์ ธนะศิลป์	ผู้จัดการ	24000	B003	143 ถ.วิภาวดีรังสิต จ.กรุงเทพฯ
SL41	ลัดดา วงค์ดี	ผู้จัดการ	9000	B005	21 ถ.ห้วยแก้ว จ.เชียงใหม่

ภาพที่ 6.18 รีเลชั่น EMPLOYEE_BRANCH

จะพบว่าข้อมูลสาขา ซึ่งก็คือที่อยู่ของแต่ละสาขามีข้อมูลซ้ำซ้อนกัน ดังนั้นจึงต้องดำเนินการแตกรีเลชั่นออกเป็น 2 รีเลชั่นด้วยกันคือ EMPLOYEE และ BRANCH ดังภาพที่ 6.19 และสามารถเขียนอยู่ในรูปแบบของรีเลชั่นสคีมาได้ดังนี้ โดยแอตทริบิวต์ที่ถูกขีดเส้นใต้หมายถึงแอตทริบิวต์ที่ใช้เป็นคีย์หลัก

EMPLOYEE (<u>empNo</u>, empName, position, salary, branchNo)

BRANCH (<u>branchNo</u>, address)

EMPLOYEE

empNo	empName	position	salary	branchNo
SL21	ชูชัย สุขศรี	ผู้จัดการ	30000	B005
SG37	ศิริทัตร มณีจันทร์	ผู้ช่วย	12000	B003
SG14	สมศักดิ์ แซ่ตั้ง	ซูเปอร์ไวเซอร์	18000	B003
SA09	ปิยะฉัตร เอี่ยมสุข	ผู้ช่วย	9000	B007
SG05	พรรัตน์ ธนะศิลป์	ผู้จัดการ	24000	B003
SL41	ลัดดา วงค์ดี	ผู้จัดการ	9000	B005

BRANCH

branchNo	Address
B0005	21 ถ.ห้วยแก้ว จ.เชียงใหม่
B003	143 ถ.วิภาวดีรังสิต จ.กรุงเทพฯ
B007	56 ถ.พหลโยธิน จ.พิษณุโลก

ภาพที่ 6.19 รีเลชั่น EMPLOYEE และ BRANCH

รีเลชั่น EMPLOYEE_BRANCH ดังภาพที่ 6.18 ได้ถูกออกแบบภายใต้ความซ้ำซ้อนของข้อมูล โดยรายละเอียดของสาขานั้นจะมีข้อมูลซ้ำๆ กันในพนักงานทุกคน ซึ่งหากมีการจัดเก็บข้อมูลตามรูปแบบดังกล่าวด้วยการจัดเก็บข้อมูลพนักงานและข้อมูลสาขารวมเข้าด้วยกัน ก็จะส่งผลในด้านของปัญหาความซ้ำซ้อนในข้อมูลซึ่งจะก่อให้เกิดข้อผิดพลาดในข้อมูลดังต่อไปนี้

1. **ข้อผิดพลาดจาดการเพิ่มข้อมูล** (Insertion Anomalies) แนวคิดหลักของข้อผิดพลาดจากการเพิ่มข้อมูลจะมีอยู่ 2 ชนิดหลักๆ ด้วยกัน โดยให้พิจารณารีเลชั่น EMPLOYEE_BRANCH ดังภาพที่ 6.18

- เมื่อมีการเพิ่มข้อมูลพนักงานใหม่เข้าไปในรีเลชั่น EMPLOYEE_BRANCH จะต้องกรอกรายละเอียดของสาขาเข้าไปด้วยทุกครั้ง เช่น ต้องเพิ่มพนักงานใหม่ ซึ่งพนักงานดังกล่าวสังกัดสาขา B007 ก็จะต้องดำเนินการกรอกที่อยู่ของสาขา B007 เข้าไปอีกทำให้เกิดที่อยู่ของสาขา B007 ซ้ำกันสองทูเพิล

- กรณีต้องการเพิ่มสาขาใหม่เข้าไป ก็จะเห็นได้ว่าเราพบปัญหาใหม่เข้าแล้ว เนื่องจากรีเลชั่น EMPLOYEE_BRANCH มีทั้งข้อมูลพนักงานและข้อมูลสาขารวมกันอยู่ในรีเลชั่นเดียวกัน ดังนั้นหากต้องการเพิ่มข้อมูลสาขาใหม่เข้าไป ก็จะดำเนินการได้ด้วยการบันทึกค่าว่างให้กับแอตทริบิวต์ empNo, empName, position, และ salary โดยทำการกรอกเฉพาะข้อมูล branchNo และ address ลงไป แต่ทั้งนี้ก็ไม่สามารถทำได้อยู่ดี เนื่องจากรีเลชั่นดังกล่าวมี employeeNo เป็นคีย์หลัก ซึ่งไม่สามารถบันทึกค่าว่างได้ ดังนั้นข้อมูลของสาขาใหม่จะบันทึกได้ก็ต่อเมื่อมีพนักงานคนหนึ่งคนใดเข้าไปสังกัดอยู่เสียก่อน จึงสามารถดำเนินการบันทึกสาขาใหม่เข้าไปได้

2. **ข้อผิดพลาดจากการลบข้อมูล** (Deletion Anomalies) ข้อผิดพลาดจากการลบข้อมูลเป็นข้อผิดพลาดที่เกิดจากการลบข้อมูลหนึ่ง แล้วส่งผลกระทบต่อข้อมูลอื่นๆ ที่ต้องถูกลบออกตามไปด้วย ตัวอย่างเช่น สมมติว่าพนักงานหมายเลข SA09 ได้ลาออกจากการเป็นพนักงานของบริษัทดังนั้นจึงต้องดำเนินการลบเรคอร์ดพนักงานคนนี้ออกไปจากฐานข้อมูล ซึ่งผลจากการลบเรคอร์ดของพนักงานรายนี้จะทำให้ต้องลบข้อมูลสาขา B007 ออกไปด้วย และหากมีเพียงพนักงานคนนี้คนเดียวที่สังกัดสาขา B007 ดังนั้นจึงทำให้ฐานข้อมูลนี้ไม่มีรายละเอียดของสาขา B007 ไปโดยปริยาย

3. **ข้อผิดพลาดจากการเปลี่ยนแปลงข้อมูล** (Modification Anomalies) ในกรณีที่ต้องการเปลี่ยนแปลงข้อมูลบางตัวของสาขา เช่น ต้องการเปลี่ยนแปลงที่อยู่ของสาขา B003 ก็จะต้องทำการปรับปรุงข้อมูลเรคอร์ดพนักงานที่สังกัดอยู่ของสาขา B003 ทั้งหมด ซึ่งหากมีพนักงานอยู่ 50 คนที่สังกัดสาขา B003 ก็จะต้องตามแก้ทั้งหมด ซึ่งถือว่าเป็นการจัดการที่ผิดพลาดจากสาเหตุของความซ้ำซ้อนในข้อมูลนั่นเอง

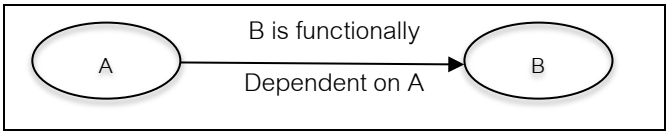
จากความผิดพลาดในข้อมูลที่เกิดจากปัญหาความซ้ำซ้อนของข้อมูล จึงสามารถแก้ไขได้ด้วยกระบวนการนอร์มัลไลเซชั่น เพื่อจัดรูปแบบรีเลชั่นให้อยู่ในรูปแบบที่เหมาะสม อย่างไรก็ตามจำเป็นต้องศึกษาแนวคิดพื้นฐานในเรื่องของ **ฟังก์ชันการขึ้นต่อกัน** (Functional Dependencies) ดังกล่าวถือเป็นพื้นฐานเพื่อปูทางไปสู่กระบวนการนอร์มัลไลเซชั่น

2. ฟังก์ชันการขึ้นต่อกัน

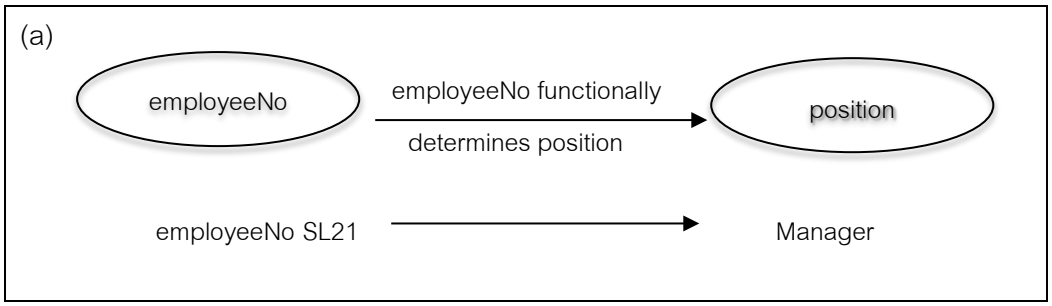
แนวคิดสำคัญที่เกี่ยวข้องกับการนอร์มัลไลเซชั่นคือ ฟังก์ชันการขึ้นต่อกัน (functional dependency) ที่ใช้อธิบายถึงความสัมพันธ์ระหว่างแอตทริบิวต์ภายในรีเลชั่น โดยค่าของแอตทริบิวต์หนึ่งหรือกลุ่มของแอตทริบิวต์ที่ถูกนำมาใช้เป็นคีย์ของรีเลชั่นนั้น สามารถไป

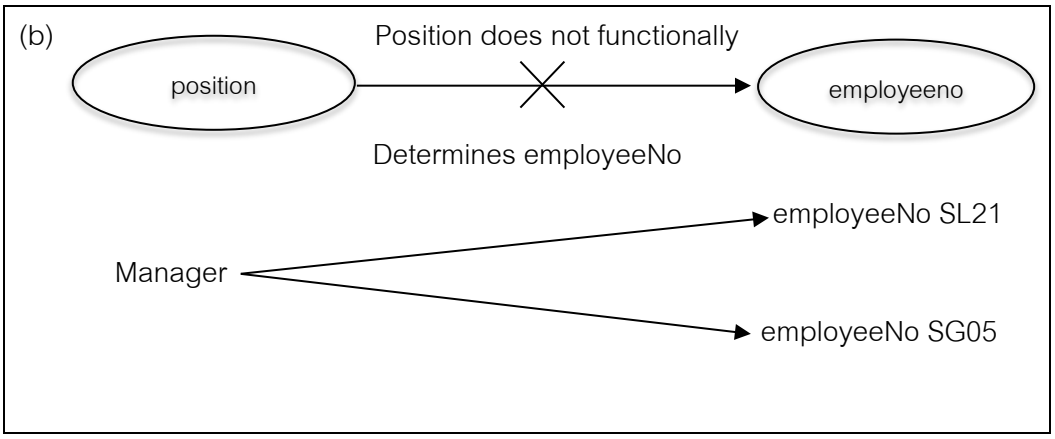
ระบุค่าแอตทริบิวต์อื่นๆ ในทูเพิลเดียวกันของรีเลชั่นนั้นได้ โดยกำหนดให้แอตทริบิวต์ที่เป็นตัวระบุค่าจะเรียกว่า Determinant ในขณะที่แอตทริบิวต์ที่ถูกระบุค่าจะเรียกว่า Dependent

สมมติว่า ให้ A และ B คือแอตทริบิวต์ของรีเลชั่น R แอตทริบิวต์ B เป็นฟังก์ชันที่ขึ้นอยู่กับแอตทริบิวต์ A ซึ่งสามารถเขียนให้อยู่ในรูปแบบสัญลักษณ์ได้ว่า A → B



ภาพที่ 6.20 แผนภาพแสดงฟังก์ชันการขึ้นต่อกัน (A → B)





ภาพที่ 6.21 ตัวอย่างฟังก์ชันการขึ้นต่อกัน

(a) employeeNo → Position

(b) position (not) → EmployeeNo

ที่มา : Chittayasothorn (2007, p. 295)

ภาพที่ 6.20 แสดงความสัมพันธ์ระหว่างแอตทริบิวต์ในลักษณะฟังก์ชันการขึ้นต่อกัน โดยแอตทริบิวต์ A สามารถระบุค่าของแอตทริบิวต์ B ได้ หมายความว่า A เป็น determinant ของ B หรือ B ขึ้นอยู่กับ A

และจากตัวอย่างฟังก์ชันการขึ้นต่อกันดังภาพที่ 6.22 โดยกำหนดให้

employeeNo คือ รหัสพนักงาน

position คือ ตำแหน่ง

ดังนั้นภาพที่ 6.21 (a) สามารถอธิบายได้ว่า

รหัสพนักงานเป็นดีเทอร์มิแนนต์ของตำแหน่ง เพราะรหัสพนักงานเป็นแอตทริบิวต์ที่เมื่อมีการระบุค่าใดค่าหนึ่งแล้วจะสามารถแสดงค่าของ Dependent ออกมาได้ (1 : 1)

ในภาพที่ 6.21 (b) ถือว่าผิด เนื่องจากตำแหน่ง (position) ไม่เป็นฟังก์ชันการขึ้นต่อกันกับรหัสพนักงาน เช่น พนักงานตำแหน่งผู้จัดการ (Manager) สามารถระบุรหัสพนักงานที่เป็นผู้จัดการได้มากกว่าหนึ่งคน (1 : M)

2.1 Fully Functional Dependency เป็นฟังก์ชันการขึ้นต่อกัน โดยที่มีตัว determinant ขนาดเล็กที่สุด ที่สามารถระบุตัว Dependent ได้อย่างชัดเจน โดยพิจารณาตัวอย่างฟังก์ชันการขึ้นต่อกันต่อไปนี้

fd1: employeeNo → branchNo

fd2: employeeNo, employeeName → branchNo

fd1 มีดีเทอร์มิแนนต์เดียวในการระบุค่า branchNo ได้

fd2 มีถึง 2 ดีเทอร์มิแนนต์ในการระบุค่า branchNo หรือกล่าวอีกนัยหนึ่งได้ว่า

branchNo เป็น Dependent อยู่บนซับเซตของ (employeeNo, employName)

จึงสรุปว่า fd1 เป็น Fully Functional Dependency เนื่องจากมีดีเทอร์มิแนนต์ขนาดเล็กที่สุด

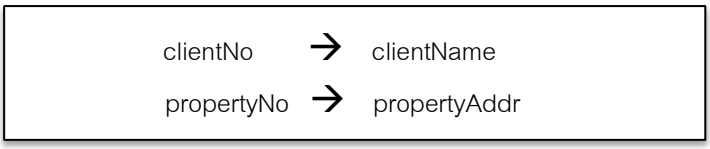
2.2 Partially Dependency ความสัมพันธ์แบบบางส่วนสามารถเกิดขึ้นได้กรณีมีคีย์หลักที่ประกอบไปด้วยหลายๆ คีย์รวมกัน หรือที่เรียกว่าคีย์รวม (Composite Key) โดยความสัมพันธ์แบบบางส่วนหรือแบบ Partial นี้เกิดขึ้นได้ก็ต่อเมื่อมีแอตตริบิวต์บางส่วนของคีย์หลัก สามารถไประบุค่าแบตตริบิวต์อื่นๆ ที่ไม่ใช่คีย์หลักของรีเลชั่นได้โดยสามารถศึกษาจากรีเลชั่น CLIENT_RENTAL ต่อไปนี้

CLIENT_RENTAL

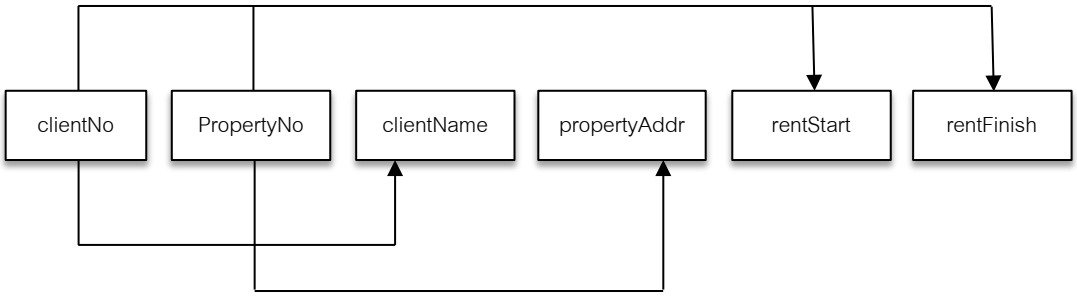
clientNo	propertyNo	clientName	propertyAddr	rentStart	rentFinish	rent	ownerNo	ownerName
CR76	PG04	ยงยุทธ ธนเลิศ	6 ถ.วิภาวดีรังสิต กรุงเทพฯ	01/07/2550	30/08/2551	350	CO40	กานดา ใจมา
CR76	PG16	ยงยุทธ ธนเลิศ	5 ถ.พญาไท กรุงเทพฯ	01/09/2551	01/09/2552	450	CO93	สุขใจ แซ่ลี้
CR56	PG04	สิราณี พรมจรรย์	6 ถ.วิภาวดีรังสิต กรุงเทพฯ	15/02/2549	15/02/2550	350	CO40	กานดา ใจมา
CR56	PG36	สิราณี พรมจรรย์	2 ถ.ประชาอุทิศ กรุงเทพฯ	10/01/2550	01/12/2551	375	CO93	สุขใจ แซ่ลี้
CR56	PG16	สิราณี พรมจรรย์	5 ถ.พญาไท กรุงเทพฯ	01/11/2552	10/08/2553	450	CO93	สุขใจ แซ่ลี้

ภาพที่ 6.22 รีเลชั่น CLIENT_RENTAL ซึ่งจัดเก็บข้อมูลลูกค้าที่เช่าบ้าน

และสามารถแสดงฟังก์ชันการขึ้นต่อกันได้ดังนี้

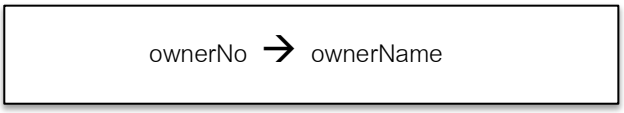


จะพบว่า clientNo และ propertyNo ซึ่งจัดเป็นส่วนหนึ่งของคีย์หลัก (clientNo, propertyNo) สามารถระบุค่าได้คือ clientNo → clientName → propertyNo → propertyAddr ซึ่งกรณีดังกล่าวจะก่อให้เกิดปัญหาในด้านการปรับปรุงข้อมูล เนื่องจากมีข้อมูลซ้ำซ้อน



ภาพที่ 6.23 แสดงการวิเคราะห์ฟังก์ชันการขึ้นต่อกันของรีเลชั่น CLIENT_RENTAL

2.3 Transitive Dependency เป็นกรณีของแอตทริบิวต์ที่ไม่ใช่คีย์ (Nonkey) สามารถระบุค่าของแอตทริบิวต์อื่นๆ ในรีเลชั่นนั้นได้ โดยพิจารณาจากรีเลชั่น CLIENT_RENTAL จากภาพที่ 6.22 จะพบว่าแอตทริบิวต์ ownerNo ซึ่งไม่ได้เป็นส่วนหนึ่งของคีย์ใดๆ แต่แอตทริบิวต์ดังกล่าวกลับสามารถระบุค่า ownerNo ได้

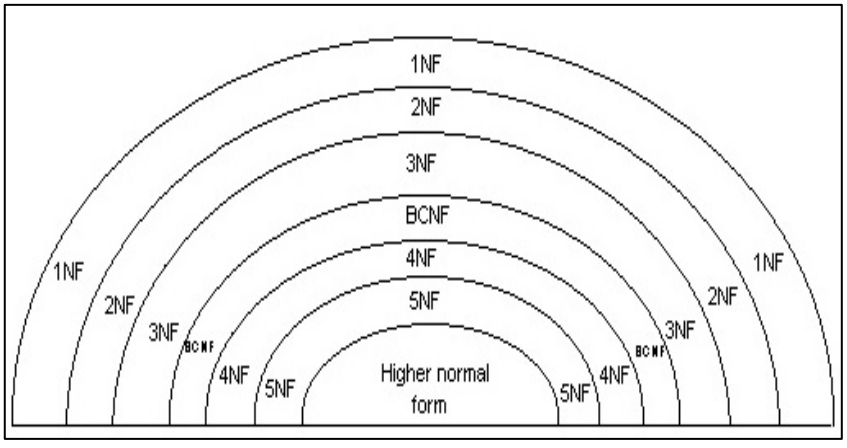


3. กระบวนการนอร์มัลไลเซชั่น

นอร์มัลไลเซชั่นคือเทคนิคที่มีแบบแผนที่นำมาใช้สำหรับวิเคราะห์รีเลชั่นที่อยู่บนพื้นฐานของคีย์หลัก (รวมถึงคีย์คู่แข่ง) และฟังก์ชั่นการขึ้นต่อกัน ซึ่งเทคนิคดังกล่าวจะเป็นกฎเกณฑ์ที่สามารถนำมาใช้ทดสอบรีเลชั่นต่างๆ ที่มีอยู่ในฐานข้อมูล

นอร์มัลฟอร์ม 3 ระดับ ประกอบด้วย 1NF, 2NF และ 3NF ต่อมา R. Boyce และ E.F. Codd ได้บัญญัติให้นอร์มัลฟอร์มระดับที่ 3 มีความแข็งแกร่งยิ่งขึ้นที่เรียกว่า Boyce-Codd Normal Form (BCNF) โดยหากรีเลชั่นต่างๆ ได้เข้าสู่กระบวนการนอร์มัลไลเซชั่น (process of normalization) แล้ว นอร์มัลฟอร์มในระดับที่สูงจะมีความซ้ำซ้อนในข้อมูลน้อยหรือไม่มีเลย ในขณะที่นอร์มัลฟอร์มระดับต่ำลงมาอาจยังมีความซ้ำซ้อนในข้อมูลอยู่ (Sikha & Richard, 2012, p. 275)

นอร์มัลฟอร์มระดับต่างๆ แสดงในภาพที่ 6.24 ซึ่งนอร์มัลฟอร์มทุกระดับ (ยกเว้นนอร์มัลฟอร์มระดับที่ 1) จะต้องอยู่บนพื้นฐานฟังก์ชันการขึ้นต่อกันระหว่างแอตทริบิวต์ของรีเลชั่น ส่วนนอร์มัลฟอร์มที่อยู่ถัดจาก BCNF คือ 4NF และ 5NF นั้น ถือเป็นนอร์มัลฟอร์มระดับสูง ซึ่งในทางปฏิบัติจะเกิดค่อนข้างยาก ดังนั้นเนื้อหาต่อไปนี้จะมุ่งประเด็นถึงนอร์มัลฟอร์มระดับมาตรฐานตั้งแต่ 1NF ถึง 3NF เป็นสำคัญ



ภาพที่ 6.24 นอร์มัลฟอร์มในระดับต่างๆ
ที่มา : Sikha & Richard (2012, p. 275)

การนอร์มัลไลเซชั่นระดับที่ 1 (First Normal Form : 1NF)

ก่อนที่จะเข้าสู่เนื้อหาของการนอร์มัลไลเซชั่นระดับที่ 1 (First Normal Form : 1NF) ควรเข้าใจถึงนิยามความหมายของนอร์มัลฟอร์มที่อยู่ลำดับก่อนหน้า 1NF เสียก่อน นั่นก็คือ Unnormalized Form หรือ UNF ซึ่งเป็นตารางที่บรรจุกลุ่มข้อมูลซ้ำๆ ที่เรียกว่า repeating group

นอร์มัลฟอร์มระดับที่ 1 ถือเป็นขั้นตอนแรกของกระบวนการนอร์มัลไลเซชั่น โดยรีเลชั่นที่มีคุณสมบัติอยู่ในนอร์มัลฟอร์มระดับที่ 1 ก็คือ การจัดการกับกลุ่มของแอตทริบิวต์ที่รวมกลุ่มกัน (repeating group) ให้เป็นค่าแอตทริบิวต์ที่มีเพียงค่าเดียว

DreamHome Lease

Client Number CR76	Property Number PG04
	Property Address
Full Name ยงยุทธ ธนเลิศ	6 ถ.วิภาวดีรังสิต
Daily Rent 350 บาท	Owner Number CO40
Rent Start 01/07/2550	
Rent Finish 31/08/2551	Full Name กานดา ใจมา

ภาพที่ 6.25 แผ่นการ์ดเอกสารการเช่าบ้านพักของลูกค้าที่รวบรวมไว้

จากแผ่นการ์ดเอกสารการเช่าบ้านพักของลูกค้า ดังภาพที่ 6.25 เมื่อนำเสนอในรูปแบบของตาราง CLIENT_RENTAL ที่ยังไม่ผ่านการนอร์มัลไลเซชั่น (UNF) ก็จะเป็นไปยังภาพที่ 6.26

ในตารางข้อมูลการเช่าที่พักของลูกค้าในรูปแบบ UNF ดังภาพที่ 6.26 นั้น เมื่อพิจารณาโดยคร่าวๆ แล้ว ดูเหมือนว่าแอตทริบิวต์ clientNo น่าจะกำหนดเป็นคีย์หลัก นอกจากนี้ยังพบว่า

มีกลุ่มข้อมูลเกี่ยวกับการเช่าที่พักอาศัยที่มีรายการข้อมูลซ้ำๆ กัน โดยมีโครงสร้างของกลุ่มข้อมูลที่ซ้ำกันดังนี้คือ

Repeating Group = (propertyNo, propertyAddr, rentStart, rentFinish, rent,
ownerNo, ownerName)

เมื่อทราบกลุ่มข้อมูลที่ซ้ำกันแล้ว ต่อไปจะดำเนินการแปลงตาราง UNF ให้อยู่ใน NF1 ด้วยการขจัดกลุ่มข้อมูลที่ซ้ำกันหรือ Repeating Group ออกไป โดยให้บันทึกข้อมูลลงไปในแต่ละแถวให้ครบ ก็จะได้ผลลัพท์ของรีเลชั่น CLIENT_RENTAL ที่อยู่ในนอร์มัลฟอร์มระดับที่ 1 ดังภาพที่ 6.27

CLIENT_RENTAL

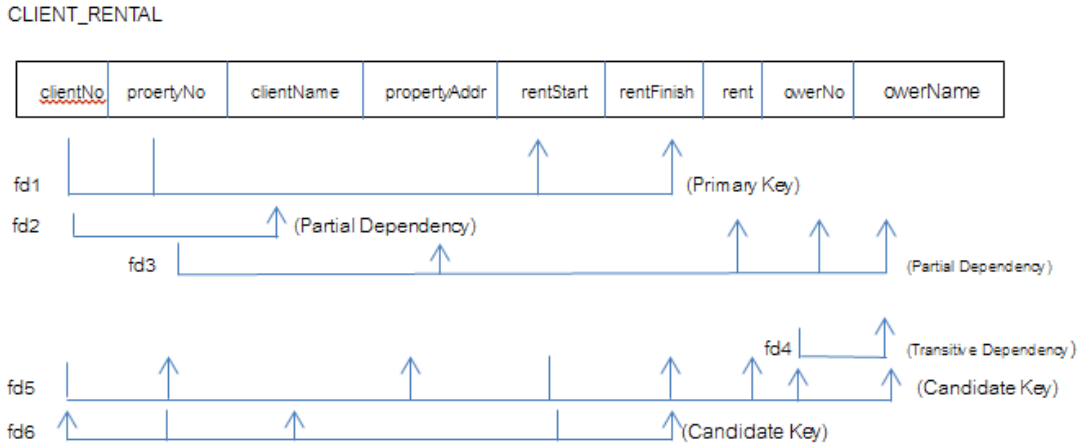
clientNo	clientName	propertyNo	propertyAddr	rentStart	rentFinish	rent	owerNo	owerName
CR76	ยงยุทธ ธนเลิศ	PG04	6 ถ.วิภาวดีรังสิต กรุงเทพฯ	01/07/2550	30/06/2551	350	CO40	กานดา ใจมา
		PG16	5 ถ.พญาไท กรุงเทพฯ	01/09/2551	01/09/2552	450	CO93	สุขใจ แซ่ลี้
CR56	สิราณี พรมจรรย์	PG04	6 ถ.วิภาวดีรังสิต กรุงเทพฯ	15/02/2546	15/02/2550	350	CO40	กานดา ใจมา
		PG3616	2 ถ.ประชาอุทิศ กรุงเทพฯ	10/10/2550	01/12/2551	375	CO939	สุขใจ แซ่ลี้
		PG	5 ถ.พญาไท กรุงเทพฯ	07/11/2552	10/08/2553	450	CO93	สุขใจ แซ่ลี้

ภาพที่ 6.26 ตาราง CLIENT_RENTAL ในรูปแบบ UNF แสดงกลุ่มข้อมูลที่เป็น Repeating Group

CLIENT_RENTAL

clientNo	propertyNo	clientName	propertyAddr	rentStart	rentFinish	rent	owerNo	owerName
CR76	PG04	ยงยุทธ ธนเลิศ	6 ถ.วิภาวดีรังสิต กรุงเทพฯ	01/07/2550	30/08/2551	350	CO40	กานดา ใจมา
CR76	PG16	ยงยุทธ ธนเลิศ	5 ถ.พญาไท กรุงเทพฯ	01/09/2550	01/09/2552	450	CO93	สุขใจ แซ่ลี้
CR56	PG04	สิราณี พรมจรรย์	6 ถ.วิภาวดีรังสิต กรุงเทพฯ	15/02/2549	15/02/2550	350	CO40	กานดา ใจมา
CR56	PG36	สิราณี พรมจรรย์	2 ถ.ประชาอุทิศ กรุงเทพฯ	01/12/2550	01/12/2551	375	CO93	สุขใจ แซ่ลี้
CR56	PG16	สิราณี พรมจรรย์	5 ถ.พญาไท กรุงเทพฯ	01/11/2552	10/08/2553	450	CO93	สุขใจ แซ่ลี้

ภาพที่ 6.27 นอร์มัลฟอร์มระดับที่ 1 ของรีเลชั่น CLIENT_RENTAL ที่มี clientNo และ propertyNo เป็นคีย์หลัก



ภาพที่ 6.28 ฟังก์ชันการขึ้นต่อกันของรีเลชั่น CLIENT_RENTAL

ภาพที่ 6.28 แสดงฟังก์ชันการขึ้นต่อกัน (fd1 ถึง fd6) ของรีเลชั่น CLIENT_RENTAL โดยในที่นี้จะใช้ฟังก์ชันการขึ้นต่อกันในการกำหนดคีย์คู่แข่งขึ้นมาให้กับรีเลชั่น ซึ่งประกอบด้วย Composite Key ต่างๆ ดังนี้คือ (clientNo, propertyNo) เป็นคีย์หลัก ดังนั้นจึงสามารถเขียนรีเลชั่น CLIENT_RENTAL ในรูปแบบของสคีมาได้ดังนี้

CLIENT_RENTAL (clientNo, propertyNo, clientName, propertyAddr, rentStart, rentFinish, rent, owerNo, owerName)

การนอร์มัลไลเซชั่นระดับที่ 2

สำหรับการนอร์มัลไลเซชั่นระดับที่ 2 (Second Normal Form : 2NF) รีเลชั่นมีคุณสมบัติเป็น 2NF ก็ต่อเมื่อ

1. รีเลชั่นต้องอยู่ในรูปแบบ 1NF มาก่อน

2. รีเลชั่นนั้นต้องตั้งอยู่บนพื้นฐานของ Full Functional Dependency กล่าวคือรีเลชั่นจะต้องไม่มีความสัมพันธ์ระหว่างแอตทริบิวต์แบบ Partially Dependency

การแปลงรีเลชั่น 1NF มาเป็น 2NF จะต้องขจัด Partially Dependency ออกไป โดยหากพบความสัมพันธ์ดังกล่าวในรีเลชั่นนั้นให้ขจัดออกไปด้วยการนำไปสร้างเป็นรีเลชั่นใหม่

จากภาพที่ 6.28 ที่ผ่านมาได้มีการวิเคราะห์ฟังก์ชันการขึ้นต่อกันของรีเลชั่น CLIENT_RENTER ซึ่งสามารถสรุปได้ดังภาพที่ 6.29

fd1 : clientNo, propertyNo → rentstart, rentfinish (Primary Key)

fd2 : clientNo → clientName (Partial Dependency)

fd3 : propertyNo → propertyAddr, rent, ownerNo, ownerName (Partial Dependency)

fd4 : ownerNo → ownerName (Transitive Dependency)

fd5 : clientNo rentStart → propertyNo, propertyAddr, rentFinish, rent, ownerNo, ownerName (Candidate Key)

fd6 : propertyNo, rentStart → clientNo, clientName, rentFinish (Candidate Key)

ภาพที่ 6.29 ผลสรุปฟังก์ชันการขึ้นต่อกันของรีเลชั่น CLIENT_RENTAL

ต่อไปจะนำฟังก์ชันการขึ้นต่อกันที่สรุปไว้ดังกล่าวไปใช้กับกระบวนการนอร์มัลไลเซชั่นในระดับสูงขึ้นต่อกันไป โดยในขั้นนี้จะทำการแปลงเป็น 2NF ซึ่งมีกฎอยู่ว่า ทุกๆ Partial Dependency ที่อยู่บนคีย์หลัก ให้ย้ายออกไปสร้างเป็นรีเลชั่นใหม่ ซึ่งประกอบด้วย

fd2: clientNo → clientName (Partial Dependency)

fd3: propertyNo → propertyAddr, rent, ownerName (Partial Dependency)

ดังนั้น รีเลชั่นต่างๆ ที่อยู่ใน 2NF ก็จะเป็นไปดังภาพ 6.30 หรือสามารถเขียนอยู่ในรูปแบบของสคีมาได้ดังต่อไปนี้

CLIENT (clientNo, clientName)

RENTAL (clientNo, propertyNo, rentStart, rentFinish)

PROPERTY_OWNER (propertyNo, propertyAddr, rent, ownerNo, ownerName)

CLIENT

clinetNo	clientName
CR76	ยงยุทธ ธนเลิศ
CR56	สิราณี พรม

RENTAL

clientNo	propertyNo	rentStart	rentFinish
CR76	PG04	01/07/2550	30/08/2551
CR76	PG16	01/09/2551	01/09/2552
CR56	PG04	15/02/2549	15/02/2550
CR56	PG36	10/10/2550	01/12/2551
CR56	PG16	01/11/2550	10/08/2553

ภาพที่ 6.30 นอร์มัลฟอร์มระดับที่ 2 ซึ่งประกอบไปด้วยรีเลชั่น CLIENT, RENTAL และ PROPERTY_OWNER

PROPERTY_OWNER

propertyNo	propertyAddr	Rent	ownerNo	ownerName
PG04	6 ถ.วิภาวดีรังสิต กรุงเทพ	350	CO40	การดา ใจมา
PG16	5 ถ.พญาไท กรุงเทพ	450	CO93	สุขใจ แซ่ลี้
PG04	6 ถ.วิภาวดีรังสิต กรุงเทพ	350	CO40	กานดา ใจมา
PG36	2 ถ.ประชาอุทิศ กรุงเทพ	375	CO93	สุขใจ แซ่ลี้
PG16	5 ถ.พญาไท กรุงเทพ	450	CO93	สุขใจ แซ่ลี้

ภาพที่ 6.30 (ต่อ) นอร์มัลฟอร์มระดับที่ 2 ซึ่งประกอบไปด้วยรีเลชั่น CLIENT, RENTAL และ PROPERTY_OWNER

การนอร์มัลไลเซชั่นระดับที่ 3

ถึงแม้ว่านอร์มัลฟอร์มระดับที่ 2NF จะสามารถช่วยลดความซ้ำซ้อนในข้อมูลได้แล้ว แต่ก็ยังคงพบความซ้ำซ้อนของข้อมูลในรีเลชั่น PROPERTY_OWNER อยู่ สำหรับการนอร์มัลไลเซชั่นระดับที่ 3 (Third Normal Form : 3NF) การเปลี่ยนแปลงรีเลชั่นให้อยู่ใน NF3 จะต้องมีคุณสมบัติดังนี้

1. รีเลชั่นนั้นต้องอยู่ในรูปแบบ 2NF มาก่อน

2. รีเลชั่นดังกล่าวจะต้องไม่มีความสัมพันธ์ระหว่างแอตทริบิวต์แบบ Transitive Dependency

และจากผลสรุปการวิเคราะห์ฟังก์ชันการขึ้นต่อกันดังภาพ 6.29 ที่ผ่านมาก็จะพบว่า

fd4: ownerNo → ownerName (Transitive Dependency)

ดังนั้นจะทำการขจัด Transitive Dependency ออกไป ด้วยการนำไปสร้างเป็นรีเลชั่นใหม่ และท้ายสุดก็จะได้รีเลชั่นที่อยู่ใน 3NF ดังรูปที่ 6.31 ซึ่งสามารถเขียนอยู่ในรูปแบบของรีเลชั่นสคีมาได้ดังนี้

CLIENT (clientNo, clientName)

RENTAL (clientNo, propertyNo, rentStart, rentFinish)

PROPERTY_FOR_RENT (propertyNo, propertyAddr, rent, ownerNo)

OWNER (ownerNo, ownerName)

CLIENT

clientNo	clientName
CR76	ยงยุทธ ธนเลิศ
CR56	สิราณี พรมจรรย์

RENTAL

clientNo	propertyNo	rentStart	rentFinish
CR76	PG04	01/07/2550	30/08/2551
CR76	PG16	01/09/2551	01/09/2552
CR56	PG04	15/02/2549	15/02/0550
CR56	PG36	10/10/2550	01/12/2551
CR56	PG16	01/11/2552	10/08/2553

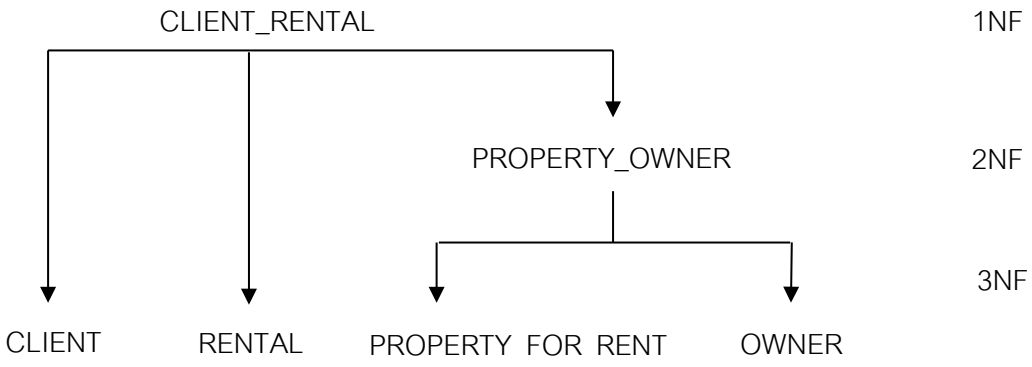
PROPERTY_FOR_RENT

propertyNo	propertyAddr	rent	ownerNo
PG04	6 ถ.วิภาวดีรังสิต กรุงเทพฯ	350	CO40
PG16	5 ถ.พญาไท กรุงเทพฯ	450	CO93
PG04	6 ถ.วิภาวกีรังสิต กรุงเทพฯ	350	CO40
PG36	2 ถ.ประชาอุทิศ กรุงเทพฯ	375	CO93
PG16	5 ถ.พญาไท กรุงเทพฯ	450	CO93

OWNER

ownerNo	ownerName
CO40	กานดา ใจมา
CO93	สุขใจ แซ่ลี้
CO40	กานดา ใจมา
CO93	สุขใจ แซ่ลี้
CO93	สุขใจ แซ่ลี้

ภาพที่ 6.31 นอร์มัลฟอร์มระดับที่ 3 ซึ่งประกอบไปด้วยรีเลชั่น CLIENT, RENTAL, PROPERTY_FOR_RENT และ OWNER



ภาพที่ 6.32 แผนภาพแสดงการแตกตาราง CLIENT_RENTAL มาเป็นรีเลชั่นระดับ 1NF – 3NF
ที่มา : Sikha & Richard (2012, p. 275)

Boyce-Codd **นอร์มัลฟอร์ม** (BCNF)

นอร์มัลฟอร์มในรูปแบบ BCNF ตั้งอยู่บนพื้นฐานของฟังก์ชันการขึ้นต่อกันที่เกี่ยวข้องกับคีย์คู่แข่งในรีเลชั่น โดย BCNF จัดเป็นเวอร์ชั่นหนึ่งของนอร์มัลฟอร์มที่ทำให้ 3NF มีความแข็งแกร่งยิ่งขึ้น ซึ่งเข้าไปจัดการกับแอตทริบิวต์ที่ไม่ใช่เป็นคีย์ที่สามารถไประบุค่าดีเทอร์มิแนนต์ของแอตทริบิวต์ที่เป็นส่วนหนึ่งของคีย์หลัก กล่าวคือ แอตทริบิวต์ที่เป็นส่วนหนึ่งของคีย์หลักไปขึ้นอยู่กับแอตทริบิวต์ที่ไม่ใช่คีย์นั่นเอง

ก่อนที่จะเข่าสู่ขั้นตอนการแปลงนอร์มัลฟอร์ม BCNF ขอย้อนไปถึงตัวอย่างก่อนหน้า ที่ได้มีการแตกรีเลชั่นออกเป็น CLIENT , RENTAL , PROPERTY_FOR_RENT และ OWNER ดังรูปที่ 6.31 ที่ผ่านมารีเลชั่นทั้งหมดเหล่านี้อยู่ในรูปแบบนอร์มัลฟอร์ม BCNF โดยปริยาย เพราะว่าแต่ละรีเลชั่นจะมีแค่เพียงหนึ่งดีเทอร์มิแนนต์เท่านั้น โดยสามารถพิจารณาจากฟังก์ชันการขึ้นต่อของรีเลชั่น RENTAL ต่อไปนี้

RENTAL (clientNo, propertyNo, rentStart, rentFinish)

fd1 : clientNo, propertyNo → rentStart, rentFinish

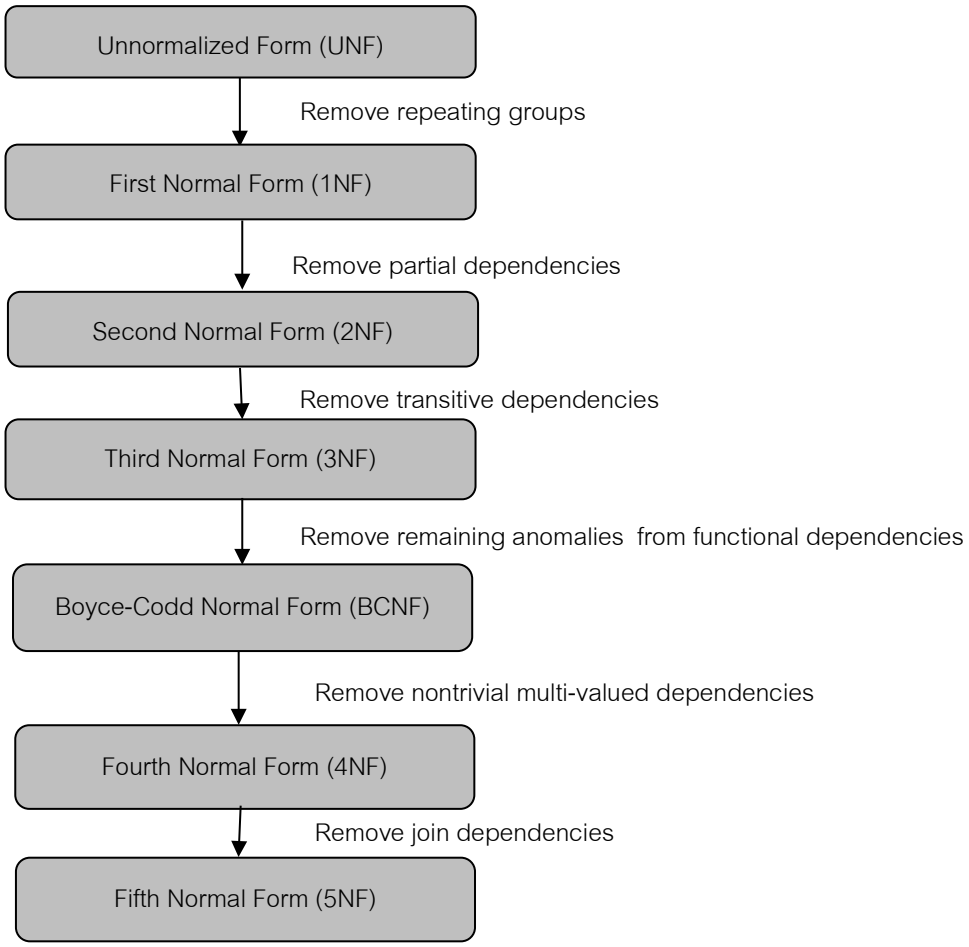
fd5 : clientNo , rentStart → propertyNo, rentFinish

fd6 : propertyNo, rentStart → clientNo, rentFinish

ทั้ง 3 ดีเทอร์มิแนนต์ของรีเลชั่น RENTAL ล้วนแต่เป็นคีย์คู่แข่ง และถือว่ารีเลชั่น RENTAL เป็น RCNF พร้อมสรรพ อย่างไรก็ตาม การฝ่าฝืนกฎ BCNF นั้นเกิดขึ้นยาก แต่ก็อาจปรากฏขึ้นได้เมื่อ

1. รีเลชั่นนั้นบรรจุคีย์คู่แข่งที่เป็น Composite Key ตั้งแต่สองตัวขึ้นไป

2. แอตตริบิวต์ที่ไม่ใช่คีย์ กลับระบุค่าดีเทอร์มิแนนต์ของแอตทริบิวต์ที่เป็นส่วนหนึ่งของคีย์หลักได้



ภาพที่ 6.33 กระบวนการนอร์มัลไลเซชั่น

ที่มา : Sikha and Richard (2012, p. 275)

ภาพที่ 6.33 แสดงภาพรวมของกระบวนการนอร์มัลไลเซชั่น และจากตัวอย่างต่อไปนี้เป็นตัวอย่างเพิ่มเติมเพื่อไว้เป็นกรณีศึกษา โดยมีรายละเอียดเพิ่มเติมเกี่ยวกับข้อมูลการสัมภาษณ์ของลูกค้าโดยพนักงาน ซึ่งแสดงไว้ในรีเลชั่น CLIENT_INTERVIEW ดังภาพที่ 6.34

CLIENT_INTERVIEW

clientNo	interviewDate	interviewTime	empNo	roomNo
CR76	13/05/2550	10:30	SG05	G101
CR56	13/05/2550	12:00	SG05	G101
CR74	13/05/2550	12:00	SG37	G102
CR56	01/07/2550	10:30	SG05	G102

ภาพที่ 6.34 รีเลชั่น CLIENT_INTERVIEW ที่แสดงข้อมูลการสัมภาษณ์ลูกค้า

ที่มา : Sikha & Richard (2012, p. 277)

รีเลชั่น CLIENT_INTERVIEW จะมีคีย์คู่แข่งอยู่ 3 ตัวด้วยกันคือ (clientNo, interviewDate), (empNo, interviewDate, interviewTime) และ (roomNo, interviewDate, interviewTime) หรือกล่าวอีกนัยหนึ่งว่ารีเลชั่น CLIENT_INTERVIEW มีคีย์คู่แข่งที่เป็น Composite Key อยู่ 3 คีย์นั่นเอง โดยมีการกำหนดคีย์หลักคือ clientNo+inteviewDate ซึ่งสามารถเขียนอยู่ในรูปแบบของรีเลชั่นสคีมาได้ดังนี้

CLIENT_INTERVIEW (clientNo, interviewDate, interviewTime, empNo, roomNo)

โดยมีรายละเอียดของฟังก์ชันการขึ้นต่อกันดังนี้คือ

fd1 : clientNo, interviewDate → interviewTime, empNo, roomNo (Primary Key)

fd2 : empNo, interviewDate, interviewTime → clientNo (Candidate Key)

fd3 : roomNo, interviewDate, interviewTime → empNo, clientNo (Candidate Key)

fd4 : empNo, interviewDate → roomNo

พิจารณาจากฟังก์ชันการขึ้นต่อกันของ fd1, fd2, fd3 ซึ่งเป็นคีย์คู่แข่งในรีเลชั่น จะไม่พบข้อผิดพลาดใดๆ จะมีเพียงแต่ fd4 ซึ่งไม่ใช่คีย์ แต่สามารถระบุค่าดีเทอร์มิแนนต์ของแอตทริบิวต์ที่เป็นส่วนหนึ่งของคีย์หลักได้ ดังนั้นรีเลชั่นนี้จึงไม่ใช่ BCNF จึงก่อให้เกิดข้อผิดพลาดในข้อมูลได้

ตัวอย่างเช่น หากมีการเปลี่ยนแปลงข้อมูลห้องที่ใช้สัมภาษณ์ (roomNo) ของพนักงานรหัส SG05 ในวันที่ 13-05-2550 ก็ต้องมีการอัปเดตถึงสองทูเพิล โดยหากมีเพียงทูเพิลเดียวก็อัปเดตด้วยห้องใหม่ ผลลัพธ์ที่ได้ก็คือ จะเกิดความไม่สอดคล้องในข้อมูลบนฐานข้อมูลทันที

สำหรับการแปลงรีเลชั่น CLIENT_INTERVIEW มาอยู่ในรูปแบบ BCNF จะต้องขจัดด้วยการนำความสัมพันธ์ที่บกพร่องเหล่านี้สร้างเป็นอีกหนึ่งรีเลชั่น ซึ่งในที่นี้ได้สร้างป็นรีเลชั่น INTERVIEW และรีเลชั่น STAFF_ROOM โดยสามารถเขียนให้อยู่ในรูปแบบของรีเลชั่นสคีมาได้ดังนี้

INTERVIEW (clientNo, interviewDate, interviewTime, empNo)

STAFF_ROOM (empNo, interviewDate, roomNo)

โดยแตกรีเลชั่นออกเป็น INTERVIEW และ STAFF_ROOM ดังภาพที่ 6.35

INTERVIEW

clientNo	interviewDate	interviewTime	empNo
CR76	13/05/2550	10:30	SG05
CR56	13/05/2550	12:00	SG05
CR74	13/05/2550	12:00	SG37
CR56	01/07/2550	10:30	SG05

STAFF_ROOM

empNo	interviewDate	roomNo
SG05	13/05/2550	10:30
SG37	13/05/2550	12:00
SG05	13/05/2550	12:00

ภาพที่ 6.35 การแตกตารางเป็นรีเลชั่น INTERVIEW และ STAFF_ROOM เพื่อเข้าสู่นอร์มัลฟอร์ม BCNF

การหลีกเลี่ยงปัญหาที่เกิดจาก Multivalued Dependencies ใน 4NF

Pratt และ Adamski (2005, p. 143) ได้แนะแนวทางเพื่อหลีกเลี่ยงปัญหาที่เกิดจาก Multivalued ใน 4NF ไว้ว่าสำหรับกรณีนอร์มัลฟอร์มระดับสูงขึ้นไปอีก เช่น 4NF ที่จะต้องขจัดฟังก์ชันการขึ้นต่อกันในลักษณะแบบ Multivalued ออกไป แต่อย่างไรก็ตาม หากผู้ออกแบบได้ดำเนินการตามแบบแผนกระบวนการนอร์มัลไลเซชั่นอย่างเคร่งครัด และรวมไปถึงประสบการณ์กับทักษะของผู้ออกแบบเอง ก็จะสามารถป้องกันเหตุการณ์ดังกล่าว ย้อนความถึงแบบแผนในการแปลงตาราง UNF มาเป็น 1NF ซึ่งจะต้องขจัด Repeating Groups ออกไป ดังตัวอย่างเช่น

ORDERS (orderNO , orderDate, (partNo, numOrdered))

เมื่อแปลงมาเป็น 1NF ก็จะได้รีเลชั่นต่อไปนี้

ORDERS (orderNo, orderDate)
ORDER LINE (orderNo, partNo, numOrdered)

และต่อไปนี้เป็นตัวอย่างของตารางแบบ UNF ที่ประกอบไปด้วย 2 Repeating Groups

FACULTY (facultyCode, facultyName, (stdCode, stdName) , (committeeCode, committeeName))

เมื่อมี Repeating Groups มากกว่าหนึ่งกลุ่ม วิธีการก็คือให้แทนที่ Repeating Groups เหล่านั้นไปเป็นตารางใหม่ขึ้นมา โดยแต่ละตารางให้บรรจุแอตทริบิวต์ใน Repeating Groups ทั้งหมดพร้อมทั้งบรรจุคีย์หลักเข้าไป ซึ่งคีย์หลักของตารางที่เป็น Repeating Groups ก็คือการนำคีย์หลักที่อยู่ในตาราง UNF เดิม บวกกับคีย์หลักของตาราง Repeating Groups มารวมกัน ดังนั้นตารางใหม่ที่เกิดจาก Repeating Groups ก็จะเป็นคีย์หลักแบบ Composite Key หรือ Concatenate Key

จากสคีมา FACULTY ข้างต้น สามารถอธิบายได้ว่า facultyName คือชื่อของคณะวิชา และ stdName คือชื่อของนักศึกษา ส่วน committeeCode และ committeeName ใช้เพื่อการ

อ้างอิงรหัสคณะทำงานและชื่อคณะทำงาน ซึ่งคณะวิชาหนึ่งๆ สามารถสังกัดเข้าเป็นสมาชิกในคณะทำงานต่างๆ ได้มากกว่าหนึ่งคณะทำงาน ครั้นเมื่อดำเนินการเปลี่ยนเป็น 1NF ก็จะได้

FACULTY (facultyCode, facultyName)

FAC_STUDENT (facultyCode, stdCode, stdName)

FAC_COMMITTEE (facultyCode, committeeCode, CommitteeName)

เมื่อพิจารณาจากรีเลชั่นข้างต้น ก็จะพบว่ารีเลชั่นดังกล่าว ได้หลีกเลี่ยงปัญหาจาก Multivalued Dependencies ด้วยการขจัดแอตทริบิวต์แบบ Multivalued ออกไปตั้งแต่กระบวนการออกแบบในเบื้องต้น จึงกล่าวโดยสรุปได้ว่า หากผู้ออกแบบได้พยายามทำกระบวนการ 1NF อย่างระมัดระมัง ก็จะสามารถหลีกเลี่ยงปัญหาของ Multivalued ได้ และ ณ จุดนี้เอง เราก็จะได้กลุ่มของรีเลชั่นที่อยู่ใน 1NF ตามข้านต้น แล้วจึงค่อยดำเนินการแปลงรีเลชั่นเหล่านี้ไปเป็น 2NF, 3NF ในที่สุด อย่างไรก็ตาม ผลลัพธ์ที่ได้จะรับประกันว่ารีเลชั่นจะอยู่ในขั้นระดับ 4NF ไปโดยปริยาย

การดีนอร์มัลไลเซชั่น

จากกระบวนการนอร์มัลไลเซชั่นที่ได้กล่าวไว้แล้วในข้างต้น จะพบว่าจุดประสงค์หลักของการนอร์มัลไลเซชั่นก็คือการลดความซ้ำซ้อนในข้อมูลด้วยการแตกรีเลชั่น โดยรีเลชั่นที่แตกออกไปนั้นจะไม่ส่งผลให้ข้อมูลที่มีอยู่เดิมสูญหายไปหรือเพิ่มเติมจากรีเลชั่นเดิมเมื่อมีการนำมา join กัน

ถึงแม้ว่าการแตกรีเลชั่นออกเป็นหลายๆ ตารางจะช่วยลดความซ้ำซ้อนในข้อมูลก็จริงอยู่ แต่หากมีการแตกรีเลชั่นมากเกินความจำเป็น จะส่งผลกระทบต่อประสิทธิภาพในการเรียกดูข้อมูลเนื่องจากจำเป็นต้องใช้เวลามากขึ้นในการประมวลผล จึงเป็นที่มาของการดีนอร์มัลไลเซชั่น (denormalization)

การดีนอร์มัลไลเซชั่นคือการลดรูปแบบนอร์มัลฟอร์มลงจากเดิม ซึ่งผลที่ตามมาจะก่อให้เกิดความซ้ำซ้อนในข้อมูลตามมา รวมถึงปัญหาเกี่ยวกับการปรับปรุงข้อมูล แต่ต้องพึงเข้าใจว่าความซ้ำซ้อนที่เกินขึ้นนั้นเป็นปัญหาที่สามารถควบคุมได้หรือไม่ และยอมรับได้หรือไม่ กล่าวคือ หากข้อมูลที่ใช้งานอยู่นั้น ส่วนใหญ่นำไปใช้เพื่อการเรียกดูข้อมูลหรือค้นหาข้อมูลเป็นหลัก กระบวนการดีนอร์มัลไลเซชั่นก็จะส่งผลดี เพราะจะทำให้การเรียกดูเพื่อค้นหาข้อมูลต่างๆ มีความรวดเร็วขึ้น (Batini, 1992, p. 254)

ปัจจัยที่เกิดขึ้นหลังจากได้มีการดีนอร์มัลไลเซชั่นคือ

1. การดีนอร์มัลไลเซชั่นทำให้การพัฒนาฐานข้อมูลมีความยุ่งยากขึ้น

2. การดีนอร์มัลไลเซชั่นทำลายความยืดหยุ่น

3. การดีนอร์มัลไลเซชั่นทำให้การเรียกดูข้อมูลมีความเร็วสูงขึ้น

4. การดีนอร์มัลไลเซชั่นทำให้การปรังปรุงข้อมูลช้าลง เนื่องจากมีข้อมูลซ้ำซ้อนจึงต้องใช้เวลาปรังปรุงข้อมูลมากขึ้นกว่าเดิม

PROPERTY_TYPE

type	description
01	แฟลต
02	บ้าน

PROPERTY_FOR_RENT

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	empNo	branchNo
PA14	19 ถ.พลหลโยธิน	พิษณุโลก	65000	02	6	650	CO46	SA09	B007
PL94	14 ถ.ลำห้วย	เชียงใหม่	50310	01	4	400	CO87	SL41	B005
PG04	6 ถ.วิภาวดีรังสิต	กรุงเทพฯ	10200	01	3	350	CO40		B003
PG36	2 ถ.ประชาอุทิศ	กรุงเทพฯ	10160	01	3	375	CO93	SG37	B003
PG21	18 ถ. พณาไทย	กรุงเทพฯ	10400	02	5	600	CO87	SG37	B003
PG16	5 ถ.พญาไท	กรุงเทพฯ	10400	01	4	450	CO93	SG14	B003

ภาพที่ 6.36 รีเลชั่น PROPERTY_TYPE และ PROPERTY_FOR_RENT

จากภาพที่ 6.36 ให้พิจารณาแอตทริบิวต์ type ในรีเลชั่น PROPERTY_FOR_RENT ซึ่งเป็นแอตทริบิวต์ที่ใช้เก็บรหัสประเภทบ้านพัก โดยในที่นี้ค่าที่เป็นไปได้จะมีอยู่ 2 ประเภทด้วยกัน คือ รหัส 01 คือแฟลตและรหัส 02 คือบ้าน

PROPERTY_FOR_RENT

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	empNo	branchNo
PA14	19 ถ.พลหลโยธิน	พิษณุโลก	65000	บ้าน	6	650	CO46	SA09	B007
PL94	14 ถ.ลำห้วย	เชียงใหม่	50310	แฟลต	4	400	CO87	SL41	B005
PG04	6 ถ.วิภาวดีรังสิต	กรุงเทพฯ	10200	แฟลต	3	350	CO40		B003
PG36	2 ถ.ประชาอุทิศ	กรุงเทพฯ	10160	แฟลต	3	375	CO93	SG37	B003
PG21	18 ถ. พณาไทย	กรุงเทพฯ	10400	บ้าน	5	600	CO87	SG37	B003
PG16	5 ถ.พญาไท	กรุงเทพฯ	10400	แฟลต	4	450	CO93	SG14	B003

ภาพที่ 6.37 รีเลชั่น PROPERTY_FOR_RENT ที่ยอมให้เกิดความซ้ำซ้อนของแอตทริบิวต์รหัสประเภทบ้านพัก (type)

แต่ให้ลองพิจารณาภาพที่ 6.37 ซึ่งเป็นกรณีที่เรายอมให้เกิดความซ้ำซ้อนในรีเลชั่นด้วยการดีนอร์มัลไลเซชั่นโดยไม่แตกตาราง ด้วยเหตุผลว่าบ้านพักมีเพียง 2 ประเภท โดยยอมให้เกิดความซ้ำซ้อนในข้อมูลและมั่นใจว่าจะสามารถควบคุมได้ ดังนั้นในกรณีที่ต้องการแสดงผลข้อมูลว่าบ้านเช่าแต่ละหลังเป็นประเภทบ้านพักแบบใด ก็สามารถเรียกใช้งานจากรีเลชั่น PROPERTY_FOR_RENT ได้ทันที โดยไม่จำเป็นต้องไปเชื่อมโยงค้นหาชนิดบ้านเช่าจากรีเลชั่น PROPERTY_TYPE อีก

เครื่องมือที่ช่วยในการออกแบบฐานข้อมูล

ในการออกแบบฐานข้อมูลนิยมใช้ซอฟต์แวร์ต่างๆ มาเป็นเครื่องมือ เนื่องจากเครื่องมือดังกล่าวช่วยให้การทำงานสะดวกและรวดเร็วขึ้น และสามารถช่วยนักออกแบบฐานข้อมูลตรวจสอบความถูกต้องของโครงร่างได้อย่างอัตโนมัติอีกด้วย

เคสทูล

เคสทูล (CASE Tools : Computer-Aided Software Engineering) เป็นซอฟต์แวร์ที่ใช้สำหรับเป็นเครื่องมือช่วยออกแบบ และสามารถนำมาสนับสนุนกรอบการทำงานตามแบบแผนของ SDLC ได้ ตัวอย่างโปรแกรมเคสทูล มีดังต่อไปนี้

1. **เคสทูลแบบฟร้อนต์เอ็นด์** (Front-End CASE Tools) เป็นเครื่องมือที่ใช้สนับสนุนงานด้านระยะการวางแผน (planning) ระยะการวิเคราะห์ (analysis) และระยะการออกแบบ (design)

2. **เคสทูลแบบแบ็คเอ็นด์** (Back-End CASE Tools) เป็นเครื่องมือที่ใช้สนับสนุนงานเกี่ยวกับการสร้างชุดคำสั่ง (coding) และระยะการนำไปใช้ (implementation)

ประโยชน์ที่ได้รับจากการใช้โปรแกรมเคสทูลช่วยในการออกแบบคือ

1. ช่วยลดต้นทุนในด้านการพัฒนา
2. สนับสนุนการทำงานโดยอัตโนมัติตามแบบแผนของ SDLC
3. ทำให้แบบแผนการพัฒนาระบบมีมาตรฐาน
4. ช่วยให้งานด้านพัฒนาโปรแกรมและงานบำรุงรักษามีความสะดวกและง่ายขึ้น

ข้อเสียของการใช้โปรแกรมเคสทูลในการออกแบบฐานข้อมูล ได้แก่

1. บางผลิตภัณฑ์มีราคาแพง
2. ทีมงานอาจต้องได้รับการฝึกอบรมการใช้งานโปรแกรมเคสทูลก่อนใช้งานจริง

ตัวอย่างโปรแกรมเคสทูล พร้อมเจ้าของผลิตภัณฑ์และเว็บไซต์แสดงในตารางที่ 6.1

ตารางที่ 6.1 โปรแกรมเคสทูลพร้อมเว็บไซต์
ที่มา : Pratt & Adamski (2005, p. 280)

โปรแกรมเคสทูล	เจ้าของผลิตภัณฑ์	เว็บไซต์
Corporate Modeler Suit	Casewise	www.casewide.com
Designer	Oracle	www.oracle.com/technology/products/designer
ERwin	Computer Associates	www3.ca.com/solutions/product.asp?ID=260
Power Designer	Sybase	www.sybase.com/products/developmentintegration/powerdesigner
System Architect	Telelogic	www.telelogic.com
Visible Analyst	Visible	www.visble.com
Visio	Microsoft	office.microsoft.com/en-us/fx010857981033.aspx

สรุป

การนอร์มัลไลเซชั่นเป็นกระบวนการนำโครงร่างของรีเลชั่นมาแตกเป็นรีเลชั่นต่างๆ ให้อยู่ในรูปแบบที่เรียกว่ารูปแบบบรรทัดฐานหรือที่เรียกว่า Normal Form โดยมีเป้าหมายเพื่อให้รีเลชั่นที่ได้รับการออกแบบอยู่ในรูปแบบบรรทัดฐานระดับที่เหมาะสม การนอร์มัลไลเซชั่นมีวัตถุประสงค์คือ ช่วยลดเนื้อที่ในการจัดเก็บข้อมูลและลดปัญหาข้อมูลที่ไม่ถูกต้อง รีเลชั่นที่มีคุณสมบัติอยู่ในฟอร์มระดับที่ 1 คือ การจัดการกับกลุ่มของแอตทริบิวต์ที่รวมกลุ่มกัน (repeating groups) ให้เป็นค่าแอตทริบิวต์ที่มีเพียงอย่างเดียว รีเลชั่นที่มีคุณสมบัติเป็น 2NF ก็ต่อเมื่อรีเลชั่นนั้นต้องอยู่ในรูปแบบ 1NF มาก่อน และรีเลชั่นนั้นต้องอยู่บนพื้นฐานของ fully functional dependency การแปลงรีเลชั่นให้อยู่ในรูปแบบ 3NF จะต้องมีคุณสมบัติดังนี้คือ รีเลชั่นนั้นต้องอยู่ในรูปแบบ 2NF มาก่อน และรีเลชั่นดังกล่าวจะต้องไม่มีความสัมพันธ์ระหว่างแอตทริบิวต์แบบ transitive dependency การดีนอร์มัลไลเซชั่น คือการลดรูปแบบนอร์มัลฟอร์มลงจากเดิม ซึ่งผลที่ตามมาทำให้ข้อมูลมีความซ้ำซ้อน แต่จะส่งผลดีด้านความรวดเร็วในการเรียกดูข้อมูล เคสทูลเป็นซอฟต์แวร์ที่ใช้สำหรับเป็นเครื่องมือช่วยการออกแบบและสามารถนำมาสนันสนุนการทำงานตามแบบแผนของ SDLC ได้

แบบฝึกหัดทบทวน

1. ความคงสภาพในข้อมูลหมายถึงอะไร จงอธิบาย
2. จงอธิบายกฎความคงสภาพของเอนทิตีพร้อมยกตัวอย่างประกอบ
3. จงอธิบายกฎความคงสภาพของโดเมนพร้อมยกตัวอย่างประกอบ
4. จงอธิบายกฎความคงสภาพของ การอ้างอิงพร้อมยกตัวอย่างประกอบ
5. จงสรุปขั้นตอนในการแปลงแบบจำลอง E-R มาเป็นรีเลชั่นให้เข้าใจพอสังเขป
6. การนอร์มัลไลเซชั่นหมายถึงอะไร
7. จงบอกวัตถุประสงค์ของการนอร์มัลไลเซชั่น
8. รีเลชั่นในระดับ 1NF จะต้องมีคุณสมบัติอย่างไร
9. รีเลชั่นในระดับ 2NF จะต้องมีคุณสมบัติอย่างไร
10. รีเลชั่นในระดับ 3NF จะต้องมีคุณสมบัติอย่างไร
11. จงแปลงแบบจำลอง E-R ระบบงานห้องสมุดที่สร้างขึ้นจากแบบฝึกหัดทบทวนในบทที่ 5 มาเป็นรีเลชั่น
12. จากข้อมูลอินวอยซ์ของโรงงานผลิตเฟอร์นิเจอร์แห่งหนึ่งที่บันทึกอยู่ในรูปแบบ UNF ต่อไปนี้

INVOICE

OrderID	orderDate	cusID	cusName	cusAddr	productID	Product Description	Unit Price	Ordered Quantity
1006	24/10/2551	2	มงคลเฟอร์นิเจอร์	ขอนแก่น	7	โต๊ะอาหาร	1,500	2
					5	โต๊ะคอมพิวเตอร์	2,450	2
					4	โต๊ะรับแขก	3,500	1
1007	25/10/2551	6	เจริญเฟอร์นิเจอร์	กรุงเทพฯ	11	โต๊ะประชุม	25,850	1
					4	โต๊ะรับแขก	3,500	3

จงแสดงกระบวนการนอร์มัลไลเซชั่นจาก 1NF – 3NF

เอกสารอ้างอิง

Batini, C. (1992). *Conceptual Database Design: An Entity-Relationship Approach.* USA: The Benjamin/Cummings Publishing.

Date, C. J. (2013). *Database Design and Relational Theory: Normals Forms and All That Jazz*. USA: O'Reilly Media.

Chittayasothorn, S. (2007). *Relational Database Design.* Thailand: Institute of Thai Information Technology.

Mata-Toledo, R. A., & Cushman, P. K. (2000). *Fundamentals of Relational Databases.* USA: McGraw-Hill.

Pratt, P. J., & Adamski, J. J. (2005). *Concepts of Database Management* (5th ed.). USA: Course Technology.

Sikha, B., & Richard, E. (2012). *Database Design Using Entity-Relationship Diagrams* (2nd ed.). UK: Taylor & Francis Ltd.

Teorey, T. J. (2006). *Database Modeling and Design: Logical Design* (4th ed.). USA: Morgan Kaufmann.