ส่วนที่ 5
การออกแบบฐานข้อมูลในระดับตรรกะ
(Logical Database Design)

วัตถุประสงค์ของส่วนนี้ เพื่อนำเค้าร่างจากการออกแบบในระดับแนวคิด (Conceptual Database Design) มาปรับเข้ากับโมเดลระบบจัดฐานข้อมูลเชิงสัมพันธ์ที่ใช้ (RDBMS) ในส่วนนี้จะกล่าวถึงขั้นตอนในการออกแบบฐานข้อมูลในระดับตรรกะ โดยจะเสริมแนวคิดเรื่องการทำให้เป็นบรรทัดฐาน (Normalization) และการดีนอร์มอลไลซ์ (Denormalization) เพื่อสามารถนำไปใช้เป็นเครื่องมือในการปรับการออกแบบเพื่อให้ได้รีเลชั่นที่เหมาะสม

บทที่ 7
การทำให้เป็นบรรทัดฐาน

7.1 บทนำ

การทำให้เป็นบรรทัดฐาน (Normalization) เป็นกระบวนการที่ใช้ในการทดสอบการออกแบบรีเลชั่นตามเกณฑ์ของขั้นตอนต่าง ๆ ในการทำให้เป็นบรรทัดฐาน เป็นการวิเคราะห์การออกแบบในลักษณะ Bottom-up การทำให้เป็นบรรทัดฐานเป็นการพิจารณาว่าคีย์หลักหรือคีย์คู่แข่งสามารถระบุค่าของแอททริบิวต์อื่นๆของทูเพิลหนึ่งในรีเลชั่นได้ (Functional Dependency) เพื่อให้เค้าร่างของรีเลชั่นที่เหมาะสมและไม่มีปัญหา ซึ่งช่วยลดความซ้ำซ้อนในฐานข้อมูลอันจะเป็นผลให้ลดเนื้อที่ในการจัดเก็บฐานข้อมูล และทำให้ข้อมูลมีความตรงกัน (Consistency) รวมถึงไม่มีปัญหาในการจัดดำเนินการข้อมูล (Anomaly) เช่น การเพิ่ม การลบ หรือปรับปรุงข้อมูล

ในบทนี้จะกล่าวถึงการที่แอททริบิวต์หนึ่งสามารถระบุค่าของแอททริบิวต์อื่นในทูเพิลหนึ่งของรีเลชั่น (Dependency) กระบวนการในการทำให้เป็นบรรทัดฐานและการแตกรีเลชั่นมากเกินไป (Overnormalization)

7.2 Functional Dependency

เป็นความสัมพันธ์เชิงฟังก์ชั่นที่แอทริบิวต์หนึ่ง หรือกลุ่มของแอททริบิวต์ (X) สามารถระบุค่าของแอททริบิวต์ต่างๆ ของทูเพิลหนึ่ง (Y) ในรีเลชั่นหนึ่ง นั่นคือ

$$X \rightarrow Y$$

Empno → Empname, Hiredate, Position, Salary

Prodno → Prodname, Cost , Balance

การที่แอททริบิวต์ต่างๆ ของทูเพิลหนึ่งสามารถระบุค่าโดยแอททริบิวต์หนึ่งหรือกลุ่มของแอททริบิวต์ในทูเพิลของรีเลชั่นได้อย่างชัดเจน เรียกว่า Fully Functional Dependency ซึ่งในกรณีเช่นนี้แอททริบิวต์ที่มีคุณสมบัติเป็นคีย์หลักจะสามารถระบุค่าของแอททริบิวต์ของทูเพิลหนึ่งได้อย่างชัดเจนเช่น Empno และ Prodno สามารถระบุค่าแอททริบิวต์อื่นของทูเพิล คุณสมบัติของแอททริบิวต์ Empno หรือ Prodno เรียกว่าดีเทอร์มินันท์ (Determinant)

อย่างไรก็ตาม อาจมีความซ้ำซ้อนของการระบุค่าของแอททริบิวต์ต่างๆ เกิดขึ้นหากไม่มีการออกแบบที่เหมาะสม อาทิเช่น เกิด Partial Dependency เป็นกรณีที่บางส่วนแอททริบิวต์ที่เป็นคีย์ผสม(Composite Key)สามารถระบุค่าของแอททริบิวต์อื่นในทูเพิลได้

อีกตัวอย่างหนึ่งที่เป็นปัญหาคือ Transitive Dependency เป็นกรณีที่แอททริบิวต์ที่ไม่ใช่คีย์หลัก (Non-Key Attribute) สามารถระบุค่าของแอททริบิวต์ที่ไม่ใช่คีย์หลักด้วยกัน (Non-Key Attribute)

ดังนั้นการทำให้เป็นบรรทัดฐาน ก็คือกระบวนการที่ขจัดปัญหาต่างๆ ที่ไม่ใช่ Fully Functional Dependency หรือความซ้ำซ้อนอื่น ๆ ให้หมดไป

7.3 กฎอ้างอิงที่เกี่ยวข้อง (Inference Rules for Functional Dependency)

ในการวิเคราะห์ Functional Dependency (FD) สามารถนำกฎของการอ้างอิงมาช่วยในวิเคราะห์ กฎสำคัญที่สามารถนำมาใช้พิจารณาว่าแอททริบิวต์หนึ่งสามารถกำหนดค่าของแอททริบิวต์อื่นๆ ได้ กฎที่ใช้ในการอ้างอิงเรียกว่าทฤษฎีของอัมสตรองค์ (Amstrong's Axioms) ประกอบด้วยกฎต่อไปนี้

1) กฎเกี่ยวกับ Reflexivity (Reflexivity Rule)

ถ้า Y เป็นเซทย่อยของ X แล้ว X → Y

ตัวอย่างเช่น ถ้า Empname เป็นเซทย่อยของ Empno แล้ว

Empno →Empname

กฎการอ้างอิงนี้เรียกว่าเป็นลักษณะที่เรียกว่า Trivial Dependency กล่าวคือ X สามารถระบุค่า Y และ Y เป็นเซทย่อยของ X

2) **กฎเกี่ยวกับ** Augmentation (Augmentation Rule)

ถ้า X → Y แล้ว X Z → Y Z

กล่าวคือ ถ้าแอททริบิวต์ Y มี Functional Dependency กับ X แล้ว Y ก็มี Functional Dependency กับ แอททริบิวต์ X ร่วมกับแอททริบิวต์อื่น

ตัวอย่างเช่น ถ้า Cust_no → Cust name แล้ว Cust_no และ Ordno → Custname

นอกจากนี้ก็มีกฎอื่นๆ เพิ่มเติมที่นิยมใช้อื่นๆ เช่น

3) **กฎเกี่ยวกับ** Transitivity (Transitivity Rule)

ถ้า X → Y และ Y → Z แล้ว X → Z

ตัวอย่างเช่น ถ้า Custno → Custname และ Cust_name → Address แล้ว

Custno → Address

4) **กฎเกี่ยวกับ** Union (Union Rule)

ถ้า X → Y และ X → Z แล้ว X → YZ

ตัวอย่างเช่น ถ้า Custno → Custname และ Custno → Address แล้ว

Custno → Custname, Address

5) **กฎเกี่ยวกับ Pseudotransitivity (Pseudotransitivity Rule)**

ถ้า X → Y และ YZ → W แล้ว XZ → W

ตัวอย่างเช่น ถ้า Custno → Custname และ Custname, Address → Telno

แล้ว Custno, Address → Telno

7.4 การแตกรีเลชั่น (Decomposition)

หากการออกแบบข้อมูลมีปัญหาในการจัดดำเนินการข้อมูล การทำให้อยู่ในรูปแบบบรรทัดฐาน (Normal Form) จะทำการแตกรีเลชั่นเดิมเป็นรีเลชั่นย่อย (Decomposition) โดยการแตกรีเลชั่นจะต้องคงไว้ซึ่งคุณสมบัติสองประการคือ

1) คุณสมบัติด้าน Lossless-Join (Nonadditive Join Property) เป็นการคงไว้ซึ่งข้อมูลของทูเพิล หรืออินสแตนซ์ของรีเลชั่นเดิมไว้ (Instance of Original Relation) กล่าวอีกนัยหนึ่งคือ ต้องไม่มีข้อมูลที่ไม่เหมือนเดิมเกิดขึ้นหรือมีข้อมูลใหม่ที่เกิดขึ้นจากการเชื่อมโยงข้อมูล

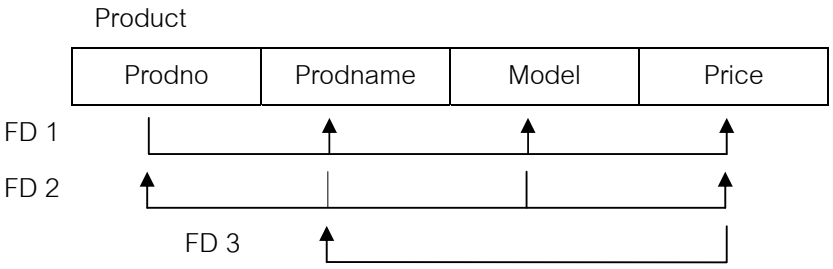
2) คุณสมบัติด้าน Dependency Preservation เป็นการคงไว้ซึ่งข้อกำหนดของรีเลชั่นเดิมไว้ (Constraint on the Original Relation) หากมีการแตกรีเลชั่นย่อย ยังคงรักษาไว้ ซึ่งข้อกำหนดเดิมไว้ให้ได้มากที่สุดและเป็นประโยชน์ต่อการใช้งาน

ตัวอย่างเช่น รีเลชั่น Product ประกอบด้วยรหัสสินค้า (Prodno) ชื่อสินค้า (Prodname) รุ่น (Model) ราคา (Price) โดยมีความสัมพันธ์ในการระบุค่าของแอททริบิวต์ดังนี้

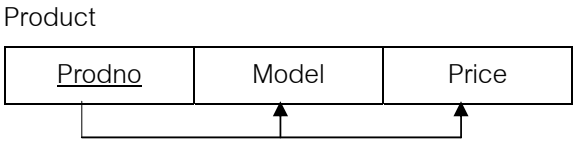
FD 1 : Prodno → (Prodname, Model, Price)

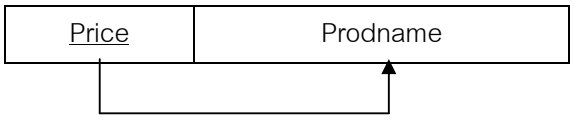
FD 2 : (Prodname, Model) → (Prodono, Price)

FD 3 : Price → (Prodname)



หากมีการแตกรีเลชั่นเป็น Product และ ProductPrice เพื่อความเหมาะสม ซึ่งจะทำให้ FD2 หายไป การแตกนี้จะไม่รักษาความสัมพันธ์ของข้อมูลที่มีอยู่เดิมไว้





7.5 การทำให้เป็นบรรทัดฐาน (Normalization)

การทำให้เป็นบรรทัดฐานมีรูปแบบบรรทัดฐาน(Normal Form) ที่ใช้ในการพิจารณาความเหมาะสมของแอททริบิวต์ในรีเลชั่นหนึ่ง ๆ ดังนี้

7.5.1 รูปแบบบรรทัดฐานขั้นที่ 1 (First Normal Form: 1NF)

รีเลชั่นหนึ่ง ๆ จะอยู่ในรูปแบบบรรทัดฐานขั้นที่ 1 ก็ต่อเมื่อรีเลชั่นนั้น ๆ ไม่มีแอททริบิวต์ของทูเพิลหนึ่ง ๆ ที่มีค่าของข้อมูลหลายค่า (No Repeating Group)

กล่าวคือรีเลชั่นที่อยู่ในรูปแบบบรรทัดฐานขั้นที่ 1 จะต้องเป็นรีเลชั่นที่ค่าของแอททริบิวต์ต่าง ๆ ในแต่ละทูเพิลจะมีค่าของข้อมูลเพียงค่าเดียว (Atomic value) ตัวอย่างเช่น รายงานการสั่งซื้อของลูกค้ามีรายละเอียด ดังนี้

Orderno	Prodno	Orderqty	Orderprice	Orderdate	Prodname	Custno
100	P1	10	100	31/05/01	PEN	1001
	P2	5	50	31/05/01	PENCIL	1001
101	P1	10	100	01/06/01	PEN	1002

รูป 7.1

จากรูป 7.1 เป็นรีเลชั่นที่แสดงข้อมูลการสั่งซื้อของลูกค้า ประกอบด้วยแอททริบิวต์ของรหัสของลูกค้า (Custno) ที่มีข้อมูลการสั่งซื้อสินค้าโดยมีแอททริบิวต์รหัสสินค้า (Prodno) หลายค่า วิธีแก้ไขให้รีเลชั่นนี้อยู่ในรูปแบบบรรทัดฐานขั้นที่ 1 คือการใส่ข้อมูลรหัสคำสั่งซื้อเพื่อให้ข้อมูลครบดังรูป 7.2 โดยคีย์หลักจะประกอบด้วย รหัสคำสั่งซื้อ (Orderno) และรหัสสินค้า (Prodno)

Orderno	Prodno	Orderqty	Orderprice	Orderdate	Prodname	Custno
100	P1	10	100	31/05/01	PEN	1001
100	P2	5	50	31/05/01	PENCIL	1001
101	P1	10	100	01/06/01	PEN	1002

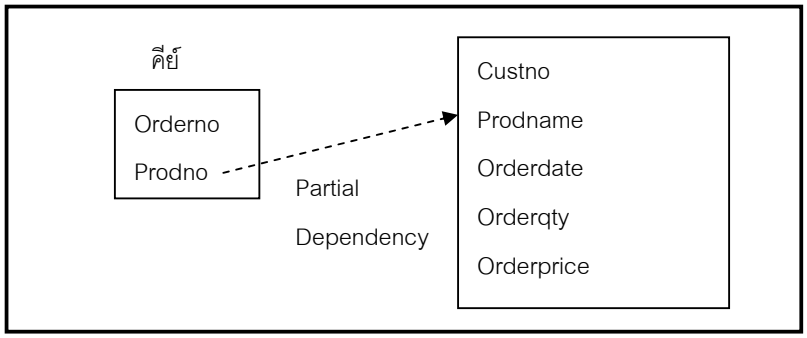
รูป 7.2

7.5.2 **รูปแบบบรรทัดฐานขั้นที่** 2 (Second Normal Form: 2NF)

รีเลชั่นหนึ่ง ๆ จะอยู่ในรูปแบบบรรทัดฐานขั้นที่ 2 ก็ต่อเมื่อรีเลชั่นนั้นอยู่ในรูปแบบบรรทัดฐานขั้นที่ 1 และไม่มี Partial Dependency

Partial Dependency เกิดขึ้นเมื่อแอททริบิวต์ที่ไม่ใช่คีย์หลักสามารถระบุค่าโดยบางส่วนของแอททริบิวต์ที่เป็นคีย์หลัก ในกรณีที่คีย์หลักเป็นคีย์ผสม รีเลชั่นจะอยู่ในรูปแบบบรรทัดฐานขั้นที่ 2 ต่อเมื่อไม่มีกรณีที่แอททริบิวต์ที่ไม่ใช่คีย์ (Non-Key Attribute) สามารถระบุค่าโดยแอททริบิวต์บางส่วนของคีย์หลักที่เป็นคีย์ผสม (Fully Functional Dependency)

รีเลชั่น Order



รูป 7.3

จากรูป 7.3 รีเลชั่น Order มีแอททริบิวต์ Orderno และ Prodno ประกอบกันเป็นคีย์หลัก ขณะเดียวกัน ส่วนหนึ่งของคีย์หลักคือ Prodno เพียงแอททริบิวต์เดียวสามารถระบุชื่อสินค้า (Prodname) ซึ่งกรณีนี้จะเกิดปัญหาคือ

- ปัญหาการปรับปรุงข้อมูล (Update Anomaly) หากชื่อสินค้า (Prodname) มีการเปลี่ยนแปลง จะต้องมีปรับปรุงข้อมูลในหลายทูเพิลซึ่งซ้ำซ้อนกัน
- ปัญหาการลบข้อมูล (Delete Anomaly) หากมีการลบข้อมูลบางทูเพิล ข้อมูลของสินค้าบางรายการหายไปจากฐานข้อมูล
- ปัญหาการเพิ่มข้อมูล (Insert Anomaly) หากต้องการเพิ่มข้อมูลสินค้า จะทำไม่ได้เลยหากสินค้านั้นยังไม่มีคำสั่งซื้อจากลูกค้า เพราะแต่ละทูเพิลจะต้องมีข้อมูลของทั้งรหัสคำสั่งซื้อและรหัสสินค้า

เพราะฉะนั้น รีเลชั่น Order ไม่ได้อยู่ในรูปแบบบรรทัดฐานขั้นที่ 2 และต้องทำการแตกรีเลชั่น เพื่อลดปัญหาดังกล่าวข้างต้น ดังนี้

- รีเลชั่น Product มีแอททริบิวต์ Prodno เป็นคีย์หลัก

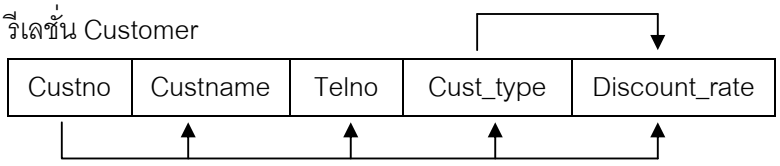
  Product (<u>Prodno</u>, Prodname)
- รีเลชั่น Order มี Prodno และ Orderno เป็นคีย์หลัก

  Order (<u>Orderno</u>, <u>Prodno</u>, Custno, Orderdate, Orderqty, Orderprice)

7.5.3 **รูปแบบบรรทัดฐานขั้นที่** 3 (Third Normal Form: 3NF)

รีเลชั่นหนึ่ง ๆ จะอยู่ในรูปแบบบรรทัดฐานขั้นที่ 3 ก็ต่อเมื่อ รีเลชั่นนั้น ๆ อยู่ในรูปแบบบรรทัดฐานขั้นที่ 2 และไม่มีกรณีที่แอททริบิวต์ที่ไม่ได้เป็นคีย์หลักสามารถระบุค่าของแอททริบิวต์อื่นที่ไม่ใช่คีย์หลัก (ไม่มี Transitive Dependency)

จากรูป 7.4 รีเลชั่น Customer อยู่ในรูปแบบบรรทัดฐานขั้นที่ 2 ยังมีปัญหาในการปรับปรุงข้อมูล เช่น เมื่อมีการปรับปรุงอัตราส่วนลดของลูกค้า (Discount_rate) ก็จะต้องปรับปรุงหลาย ๆ ทูเพิล รูปแบบบรรทัดฐานขั้นที่ 3 เป็นเรื่องของการพิจารณาถึง Transitive Dependency (A → B และ B → C)



รูป 7.4

รีเลชั่น Customer เกิดกรณีที่แอททริบิวต์ Cust_type สามารถระบุ Discount_rate ซึ่งแอททริบิวต์ Discount_rate มีคุณสมบัติ Fully Functional Dependency กับทั้ง Custno และ Discount_rate เพราะฉะนั้นจะทำการแตกรีเลชั่นเป็นดังนี้

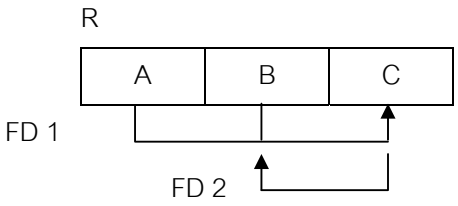
รีเลชั่น Customer ประกอบด้วยแอททริบิวต์ Custno, Custname, Cust_type โดยมี Custno เป็นคีย์หลัก

รีเลชั่น Customer_Type ประกอบด้วยแอททริบิวต์ Cust_type, Discount_rate โดยมี Cust_type เป็นคีย์หลัก

7.5.4 **รูปแบบบรรทัดฐาน** BCNF (Boyce–Codd Normal Form)

รีเลชั่นหนึ่งจะอยู่ในรูปแบบบรรทัดฐาน BCNF ก็ต่อเมื่อรีเลชั่นนั้นอยู่ในรูปแบบบรรทัดฐานขั้นที่ 3 และแอททริบิวต์ที่ระบุค่าของแอททริบิวต์อื่นในทูเพิลหนึ่ง (Determinant) ต้องเป็นคีย์คู่แข่ง (Candidate key)

รูปแบบบรรทัดฐานนี้ เป็นรูปแบบที่ขยายขอบเขตของรูปแบบบรรทัดฐานขั้นที่ 3 รูปแบบรีเลชั่นที่จะต้องผ่านการทำให้เป็นบรรทัดฐาน BCNF จะมีคุณสมบัติดังนี้คือเป็นรีเลชั่นที่มีคีย์คู่แข่งหลายคีย์ (Multiple Candidate Key) และมีแอททริบิวต์ที่ไม่ใช่คีย์สามารถระบุค่าของคีย์ได้ดังนี้



FD 1 : (A, B) $\rightarrow$ C

FD 2 : C $\rightarrow$ B

ตัวอย่างเช่น การสั่งซื้อสินค้าจากผู้ผลิตที่ประกอบด้วยข้อมูล ดังนี้

รีเลชั่น Supplier

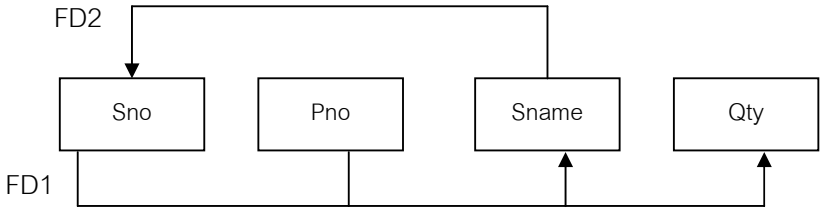
Sno	Sname	Pno	Qty
S1	Mechai	P1	10
S1	Mechai	P2	20
S2	Sirada	P3	30

รูป 7.5

จากรีเลชั่น Supplier ในรูป 7.5 สมมุติให้ชื่อผู้ผลิต (Sname) ไม่ซ้ำ คีย์คู่แข่งของรีเลชั่นนี้คือ แอททริบิวต์ Sno กับแอททริบิวต์ Pno หรือแอททริบิวต์ Sname กับแอททริบิวต์ Pno ซึ่งคีย์คู่แข่งทั้งสองเป็นคีย์ผสม และมีความซ้ำซ้อนกันโดยต่างก็มีแอททริบิวต์ Pno รีเลชั่นนี้มีปัญหาในการออกแบบข้อมูล กล่าวคือ หากเลือกแอททริบิวต์ Sno และ Pno เป็นคีย์หลัก จะเกิดกรณีที่แอททริบิวต์ Sname สามารถระบุค่าของแอททริบิวต์ Sno ดังรูป 7.6

FD1 : (Sno, Pno) → Sname, Qty

FD2 : Sname → Sno



รูป 7.6

ดังนั้น รีเลชั่นมีปัญหาความซ้ำซ้อนของข้อมูล และปัญหาในการเพิ่มข้อมูล จึงแตกรีเลชั่น Supplier ออกเป็นดังนี้

กรณีที่ 1 : รีเลชั่น Supplier ประกอบด้วยแอททริบิวต์ Sno และ Sname โดยมี Sno และ Sname เป็นคีย์หลัก

รีเลชั่น Order ประกอบด้วยแอททริบิวต์ Sno และ Pno และ Qty โดยมี Sno และ Pno เป็นคีย์หลัก

กรณีที่ 2 : รีเลชั่น Supplier ประกอบด้วยแอททริบิวต์ Sno, Sname โดยมี Sno และ Sname เป็นคีย์หลัก

รีเลชั่น Order ประกอบด้วยแอททริบิวต์ Sname, Pno, Qty โดยมี Sname และ Pno เป็นคีย์หลัก

7.5.5 รูปแบบบรรทัดฐานขั้นที่ 4 (Fourth Normal Form: 4NF)

รีเลชั่นหนึ่ง ๆ อยู่ในรูปแบบบรรทัดฐานขั้นที่ 4 ก็ต่อเมื่อ รีเลชั่นนั้นอยู่ในรูปแบบ BCNF และไม่มี Nontrivial Multivalued Dependency

โดยทั่วไป อาจจะกล่าวได้ว่า เมื่อรีเลชั่นหนึ่งอยู่ในรูปแบบบรรทัดฐาน BCNF จะไม่มีปัญหา Fully Functional Dependency อย่างไรก็ตามในกรณีที่เป็น Multivalued Dependency (MVD) ซึ่งจะเกิดขึ้นกับรีเลชั่น (R) ที่มีแอททริบิวต์อย่างน้อยสามแอททริบิวต์ โดยแอททริบิวต์หนึ่ง (A) สามารถระบุค่าของอีกแอททริบิวต์หนึ่ง (B) ได้หลายค่า และยังสามารถระบุค่าของแอททริบิวต์ที่สาม (C) ได้หลายค่าเช่นกัน นั่นคือ

$X \rightarrow\rightarrow Y$

$X \rightarrow\rightarrow Z$

ตัวอย่างเช่น สมมุติให้รีเลชั่น Emp_skill_custno ข้อมูลพนักงานของบริษัทมีแอททริบิวต์ที่ชื่อว่า Skill เป็นข้อมูลเกี่ยวกับความสามารถพิเศษ โดยพนักงานหนึ่งคนจะมีความสามารถพิเศษได้หลายอย่าง และพนักงานคนหนึ่งจะดูแลลูกค้าหลายคน (Custno) ดังนี้

รีเลชั่น Emp_skill_custno

Empno	Skill	Custno
3001	Computer	C001
	Law	C003
3002	English	C001
	Law	C002

รูป 7.7

จากรูป 7.7 จะเห็นว่ารีเลชั่น Emp_skill_custno ดังกล่าวข้างต้นเป็นลักษณะของการออกแบบที่พนักงานหนึ่งจะมีความเชี่ยวชาญ (Skill) หลายอย่าง และพนักงานหนึ่งคนรับผิดชอบดูแลลูกค้าหลายคน นั่นคือ Empno →→ Skill และ Empno →→ Custno หากปรับรีเลชั่นนี้ให้อยู่ในรูปแบบบรรทัดฐานขั้น BCNF โดยให้ทั้งสามแอททริบิวต์ประกอบกันเป็นคีย์หลักของรีเลชั่น ดังรูป 7.8

Emp_skill_custno

Empno	Skill	Custno
3001	Computer	C001
3001	Computer	C003
3001	Law	C001
3001	Law	C003
3002	English	C001
3002	English	C002
3002	Law	C001
3002	Law	C002

รูป 7.8

อย่างไรก็ตาม รีเลชั่นนี้ยังมีปัญหา Multivalued Dependency (MVD) ซึ่งก่อให้เกิดปัญหาเมื่อเป็นกรณีของ Nontrivial Multivalued Dependency (Independently Multivalued Dependency) กล่าวคือ Y ไม่เป็นเซทย่อยของ X

กล่าวคือ รีเลชั่น Emp_skill_custno ในรูป 7.8 เป็น Nontrivial MVD เนื่องจากแอททริบิวต์ Skill และแอททริบิวต์ Custno ไม่มีความสัมพันธ์กัน หรือไม่เป็นเซทย่อยกัน แต่ทั้งสองแอททริบิวต์สัมพันธ์โดยตรงกับแอททริบิวต์ Empno ทำให้ต้องใส่ค่าซ้ำลงในรีเลชั่น และเกิดความซ้ำซ้อนขึ้น เช่น ปัญหาความซ้ำซ้อนในการปรับปรุงข้อมูล ดังนั้น จึงต้องแตกรีเลชั่นออกเพื่อเป็น Trivial Multivalued Dependency ดังรูป 7.9

รีเลชั่น Emp_skill ประกอบด้วย Empno, Skill โดยมี Empno และ Skill เป็นคีย์หลัก

รีเลชั่น Emp_cust ประกอบด้วย Empno, Custno โดยมี Empno และ Custno เป็นคีย์หลัก

Emp_Skill

Empno	Skill
3001	Computer
3001	Law
3002	English
3002	Law

Emp_custno

Empno	Custno
3001	C001
3001	C003
3002	C001
3002	C002

รูป 7.9 : **Trivial Multivalued Dependency**

7.5.6 **รูปแบบบรรทัดฐานขั้นที่** 5 **(Fifth Normal Form: 5NF)**

รีเลชั่นหนึ่ง อยู่ในรูปแบบบรรทัดฐานชั้นที่ 5 ก็ต่อเมื่อ รีเลชั่นนั้นอยู่ในรูปแบบบรรทัดฐานขั้นที่ 4 และไม่มี Join Dependency

รูปแบบบรรทัดฐานขั้นที่ 5 บางครั้งเรียกว่า Project-Join Normal Form (PJ/NF) เกิดขึ้นในกรณีที่รีเลชั่นหนึ่งมีแอททริบิวต์อย่างน้อยสามแอททริบิวต์ขึ้นไป และแอททริบิวต์เหล่านี้ ไม่มีปัญหา Functional Dependency หรือ Nontrivial Multivalued Dependency แต่ยังคงมีปัญหาเกิดขึ้น

ตัวอย่างเช่น รีเลชั่น Emp_skill_project ประกอบด้วย แอททริบิวต์รหัสพนักงาน (Empno) ความเชี่ยวชาญ (Skill) และรหัสโครงการที่ทำ (Projno) รีเลชั่นนี้มีความสัมพันธ์ของแอททริบิวต์ทั้งสามในลักษณะที่เป็นวงจร (Cyclic Nature) กล่าวคือ

- พนักงานมีความเชี่ยวชาญ (Empno กับ Skill)
- โครงการต้องการความเชี่ยวชาญ (Projno กับ Skill)
- พนักงานที่มีความเชี่ยวชาญถูกมอบหมายให้ทำงานในโครงการ (Empno กับ Projno)

รีเลชั่น Emp_skill_project

Empno	Skill	Projno
1001	Computer	Proj01
1001	Computer	Proj02
1001	English	Proj02
1001	English	Proj03
1002	English	Proj02
1003	Computer	Proj02

รูป 7.10

จากรูป 7.10 ข้อมูลในรีเลชั่น Emp_skill_project พนักงานรหัส 1001 ที่มีความเชี่ยวชาญ (Skill) คือ Computer และ English โดยโครงการ Proj02 มีความต้องการผู้เชี่ยวชาญทั้ง Computer และ English รีเลชั่นนี้อยู่ในรูปแบบบรรทัดฐานขั้นที่ 4 โดยมีทั้งสามแอททริบิวต์ประกอบกันเป็นคีย์หลัก แต่รีเลชั่นนี้ก็ยังมีปัญหาของข้อมูลอยู่

กล่าวคือ หากทูเพิล (1001, Computer, Proj02) ถูกลบออกจากรีเลชั่นก็ต้องลบทูเพิล (1001, English Proj02) เนื่องจากพนักงาน 1001 มีความเชี่ยวชาญทั้ง Computer และ English ที่ใช้กับโครงการ Proj02 จึงทำการแตกรีเลชั่นเป็นสามรีเลชั่นในลักษณะตามความสัมพันธ์ที่กล่าวมาข้างต้น คือ

รีเลชั่น Empno_skill มีแอททริบิวต์ Empno และ Skill โดยมีทั้งสองแอททริบิวต์เป็นคีย์หลัก

รีเลชั่น Skill_project มีแอททริบิวต์ Skill และ Projno โดยมีทั้งสองแอททริบิวต์เป็นคีย์หลัก

รีเลชั่น Emp_project มีแอททริบิวต์ Empno และ Projno โดยมีทั้งสองแอททริบิวต์เป็นคีย์หลัก

เมื่อมีการแตกรีเลชั่นจากรีเลชั่นหนึ่งเป็นหลายแอททริบิวต์ในลักษณะนี้คือ R(a1, a2, a3) เป็น R1 (a1, a2) และ R2 (a2, a3) และ R3 (a3, a1) โดยรีเลชั่นที่แตกจะมีแอททริบิวต์บางส่วนที่เหมือนกัน (Consequence) กล่าวคือ R1 มีแอททริบิวต์ a1 และ a2 R2 มีแอททริบิวต์ a2 และ a3 ซึ่งมีแอททริบิวต์เหมือนกับ R1 คือ a2 R3 มีแอททริบิวต์ a3 และ a1 และมีแอททริบิวต์ที่เหมือนกับ R2 คือ a3 นั่นคือ แอททริบิวต์a1, a2, a3 ของรีเลชั่น R แตกเป็น a1, a2 และ a2, a3 และ a3, a1 ของรีเลชั่น R1 R2 และ R3 ตามลำดับ

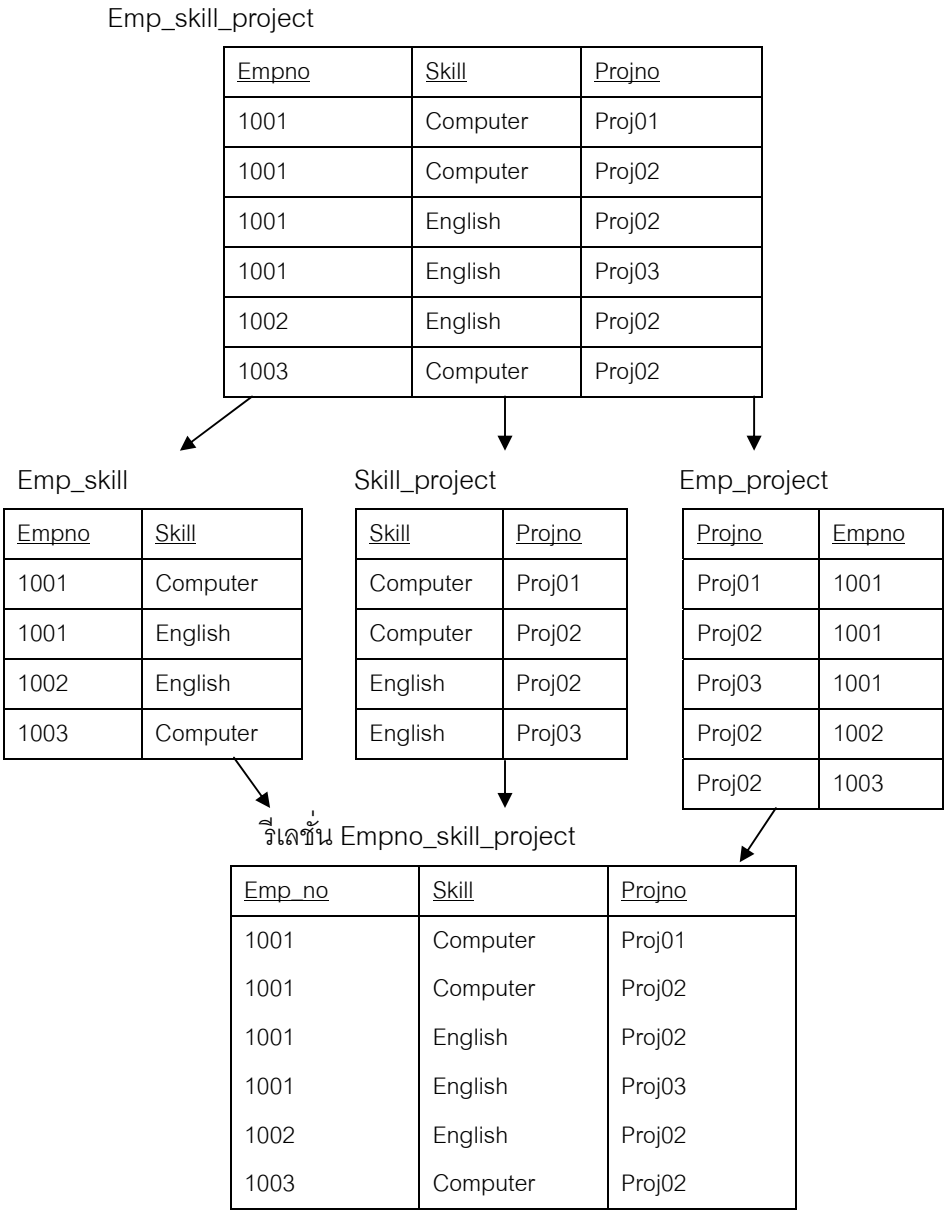
Join dependency ของรีเลชั่นจะเกิดขึ้นเมื่อมีการเชื่อมโยง (Join) รีเลชั่นที่แตกในลักษณะดังกล่าวข้างต้น โดยการเชื่อมโยงจะใช้แอททริบิวต์ที่เหมือนกันระหว่างรีเลชั่นย่อย(Consequence) ที่แตกมาเชื่อมโยงกัน ผลจากการเชื่อมโยงรีเลชั่นที่ถูกแตกออกมาทั้งหมด จะต้องได้ข้อมูลเหมือนรีเลชั่นเดิม (Lossless Decomposition) หรือ Lossless Join Dependency)

การแตกรีเลชั่น Emp_skill_project เป็นสามรีเลชั่น คือ รีเลชั่น Emp_skill รีเลชั่น Skill_project และรีเลชั่น Emp_project โดยมีแอททริบิวต์ Skill หรือ Projno หรือ Empno ซึ่งใช้เป็น Consequence ในการเชื่อมโยงและเมื่อมีการเชื่อมโยงรีเลชั่นย่อยทั้งสามจะได้ข้อมูลเหมือนรีเลชั่นเดิมดังรูป 7.11 นั่นคือรีเลชั่น Emp_skill_project ไม่ได้อยู่ในบรรทัดฐานขั้นที่ 5 เพราะมี Join Dependency และแอททริบิวต์ Skill หรือ Projno หรือ Empno ที่ใช้ในการเชื่อมโยงเป็นเพียงส่วนหนึ่งของคีย์หลัก จึงต้องทำการแตกเป็นรีเลชั่นย่อยสามรีเลชั่น

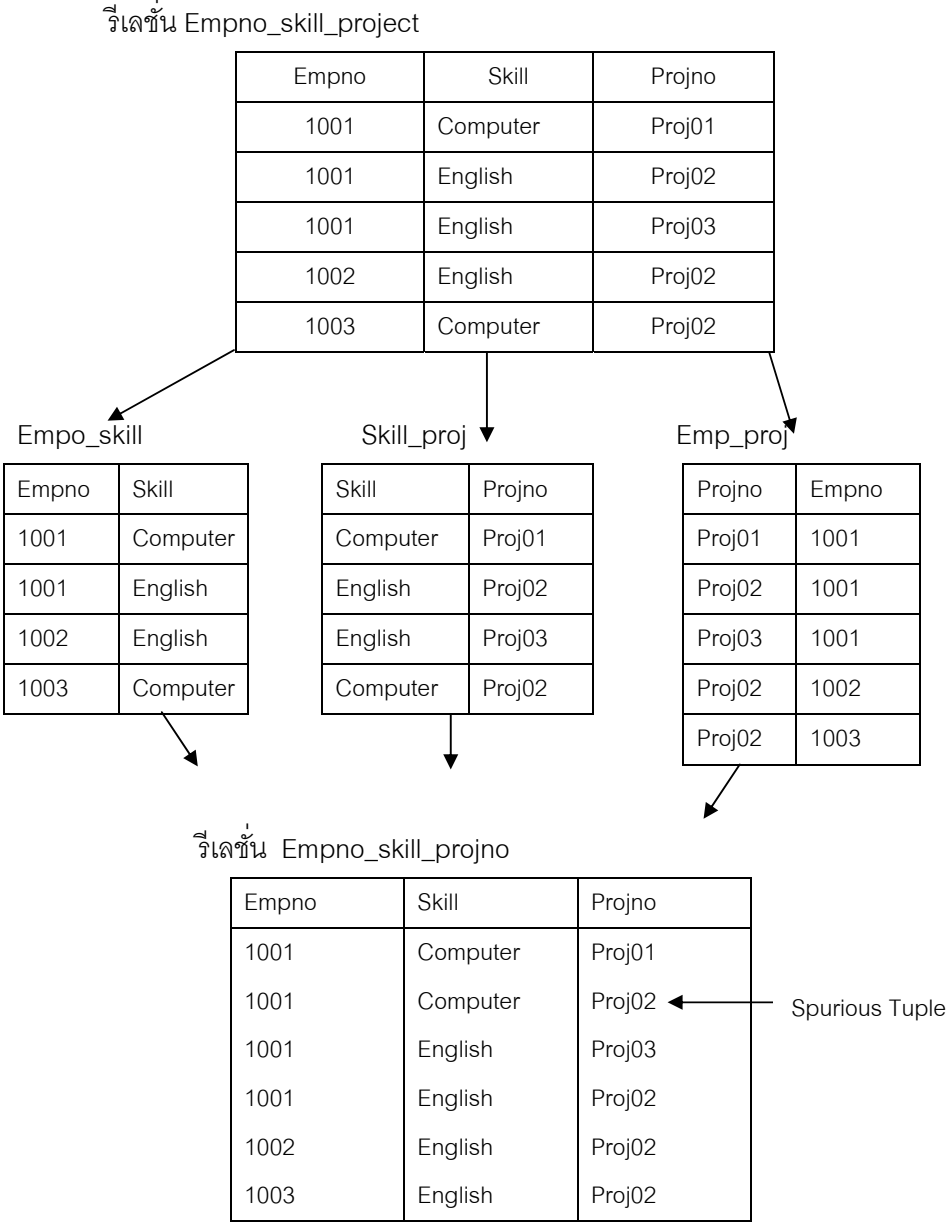
กล่าวอีกนัยหนึ่งคือ รีเลชั่นหนึ่งจะอยู่ในรูปแบบบรรทัดฐานขั้นที่ 5 ก็ต่อเมื่อ Join Dependency ซึ่งเป็นผลจากการเชื่อมโยงแอททริบิวต์ที่เหมือนกันระหว่างรีเลชั่นที่แตก (Consequence) โดยแอททริบิวต์ที่ใช้ในการเชื่อมโยงเหล่านี้ต้องเป็นการเชื่อมโยงด้วยแอททริบิวต์ที่เป็นคีย์ของรีเลชั่นนั้น (Every join dependency is a consequence of its relation keys) ในทางตรงกันข้ามถ้าไม่เกิด Join Dependency ก็ให้คือว่ารีเลชั่นอยู่ในรูปแบบบรรทัดฐานขั้นที่ 5 ตัวอย่างเช่น หากมีข้อกำหนดความสัมพันธ์ของแอททริบิวต์เปลี่ยนไป คือ

- พนักงานมีความเชี่ยวชาญต่าง ๆ (Empno กับ Skill)
- โครงการต้องการความเชี่ยวชาญต่าง ๆ (Skill, Projno)
- แต่พนักงานที่มีความเชี่ยวชาญอาจไม่ได้ใช้ความเชี่ยวชาญทำงานให้โครงการต่าง ๆ (Empno ไม่สัมพันธ์กับ Projno)

สมมุติจากรูป 7.11 ให้ลบข้อมูลแถวที่ 2 ของพนักงานรหัส1001 ซึ่งมีความเชี่ยวชาญด้าน Computer ที่ต้องใช้กับ Proj02 ออกไป เมื่อทำการแตกรีเลชั่นเป็นรีเลชั่นแล้วเชื่อมโยงข้อมูลของรีเลชั่นย่อยจะไม่เป็น Join Dependency เพราะเกิด Spurious Tuple ขึ้นมา ดังรูป 7.12 กรณีนี้จึงไม่สามารถแตกรีเลชั่นและให้ถือว่ารีเลชั่น Empno_skill_project จากรูป 7.11 อยู่ในรูปแบบบรรทัดฐานขั้นที่ 5


Emp_skill_project
Empno Skill Projno
1001 Computer Proj01
1001 Computer Proj02
1001 English Proj02
1001 English Proj03
1002 English Proj02
1003 Computer Proj02
Emp_skill
Empno Skill
1001 Computer
1001 English
1002 English
1003 Computer
Skill_project
Skill Projno
Computer Proj01
Computer Proj02
English Proj02
English Proj03
Emp_project
Projno Empno
Proj01 1001
Proj02 1001
Proj03 1001
Proj02 1002
Proj02 1003
รีเลชั่น Empno_skill_project
Emp_no Skill Projno
1001 Computer Proj01
1001 Computer Proj02
1001 English Proj02
1001 English Proj03
1002 English Proj02
1003 Computer Proj02

รูป 7.11


รีเลชั่น Empno_skill_project
Empno Skill Projno
1001 Computer Proj01
1001 English Proj02
1001 English Proj03
1002 English Proj02
1003 Computer Proj02
Empo_skill
Empno Skill
1001 Computer
1001 English
1002 English
1003 Computer
Skill_proj
Skill Projno
Computer Proj01
English Proj02
English Proj03
Computer Proj02
Emp_proj
Projno Empno
Proj01 1001
Proj02 1001
Proj03 1001
Proj02 1002
Proj02 1003
รีเลชั่น Empno_skill_projno
Empno Skill Projno
1001 Computer Proj01
1001 Computer Proj02
1001 English Proj03
1001 English Proj02
1002 English Proj02
1003 English Proj02
Spurious Tuple

รูป 7.12

7.6 รูปแบบบรรทัดฐานอื่น ๆ : Domain Key Normal Form (DKNF)

รูปแบบบรรทัดฐานที่กล่าวมาแล้วทั้งหมดเป็นการพิจารณาถึง Dependency ต่างๆ เช่น Transitive หรือ Partial หรือ Multivalued Dependency นอกจากนี้ การออกแบบฐานข้อมูลยังต้องพิจารณารูปแบบบรรทัดฐานที่เป็นข้อกำหนดทั่วไป เช่น ข้อกำหนดที่ว่า เงินเดือนของผู้จัดการจะมีค่าตั้งแต่ 50,000 บาท ข้อกำหนดทั่วไปข้างต้นสามารถระบุเป็น 2 ลักษณะ คือ

ก. Domain Dependency (Domain Constraint) เป็นการกำหนดให้ค่าของแอททริบิวต์เป็นไปตามที่กำหนด เช่นใช้โอเปอเรเตอร์ IN (ชื่อแอททริบิวต์ Constaint) ในการกำหนดคุณสมบัติ

ข. Key Dependency (Key Constraint) เป็นการระบุถึงแอททริบิวต์ที่เป็นคีย์หลักของรีเลชั่นนั้น ๆ เช่นระบุเป็น Key (ชื่อแอททริบิวต์)

การพิจารณารูปแบบบรรทัดฐานนี้ให้พิจารณาเพิ่มเติมในเรื่องการเพิ่มหรือลบข้อมูลของรีเลชั่นหนึ่ง ๆ ว่าจะเกิดปัญหา Anomaly ตามข้อกำหนดหรือไม่ หากไม่มีปัญหารีเลชั่นนั้นอาจอยู่ในรูปแบบ DKNF ตัวอย่างเช่น

รีเลชั่นนี้มี Empno เป็นคีย์หลัก และมีการกำหนดโดเมนให้แอททริบิวต์ต่าง ๆ ดังนี้

in (Empno, ‘Char’)

in (Empname, ‘Varchar2’)

in (Hiredate, ‘Date’)

in (Position, (‘Salesman’, ‘Manager’))

in (Salary, ‘Number’)

Key (Empno)

If Position = ‘Manager’, then salary > 50000

ข้อกำหนดของแอททริบิวต์ต่างๆ ของรีเลชั่นนี้มีปัญหา นั่นคือ เงื่อนไขของผู้ที่มีตำแหน่งงานเป็น Manager จะต้องมีเงินเดือนมากกว่า 50000 หากมีการเพิ่มข้อมูลว่าพนักงานที่มีตำแหน่ง MANAGER และมีเงินเดือน 10000 ก็อาจทำได้ ทั้งนี้เพราะหากระบบจัดการฐานข้อมูลนั้น ๆ ถือว่าตัวอักษรตัวพิมพ์เล็กและตัวพิมพ์ใหญ่ เป็นค่าที่แตกต่างกัน การเพิ่มข้อมูลดังกล่าวก็ทำได้ ทั้งที่ผิดข้อกำหนดโดเมนของแอททริบิวต์ตำแหน่งงาน

ดังนั้นจึงจำเป็นต้องแตกรีเลชั่นให้อยู่ใน DKNF ข้างล่าง โดยรีเลชั่น Manager จะมีข้อกำหนดของเงินเดือนมากกว่า 50000 ดังนี้ in (salary, number > 50000) โดยเงื่อนไขข้อกำหนดสามารถตรวจสอบด้วยชุดคำสั่งงานและอีกรีเลชั่นหนึ่งเป็น Non-manager สำหรับพนักงานที่เป็น Manager และมีเงินเดือนมากกว่า 50000 ดังรูป 7.13

Manager

Empno	Empname	Hiredate	Salary
1002	_ _ _ _ _ _ _	_ _ _ _ _ _	60000

Non_manager

Empno	Empname	Hiredate	Position	Salary
1001	_ _ _ _ _ _	_ _ _ _ _ _	Salesman	35000
1003	_ _ _ _ _ _ _	_ _ _ _ _ _	Clerk	15000

รูป 7.13

อย่างไรก็ตามในทางปฏิบัติ ข้อกำหนดต่าง ๆ จะพิจารณารายละเอียดในเรื่องของความบูรณภาพของข้อมูล (Integrity Constraint) โดยการกำหนดโดเมนของแอททริบิวต์ด้วยภาษาสำหรับนิยามข้อมูล (Data Definition Language) ของ SQL มากกว่าที่จะมาพิจารณาเป็นลักษณะของรูปแบบบรรทัดฐาน

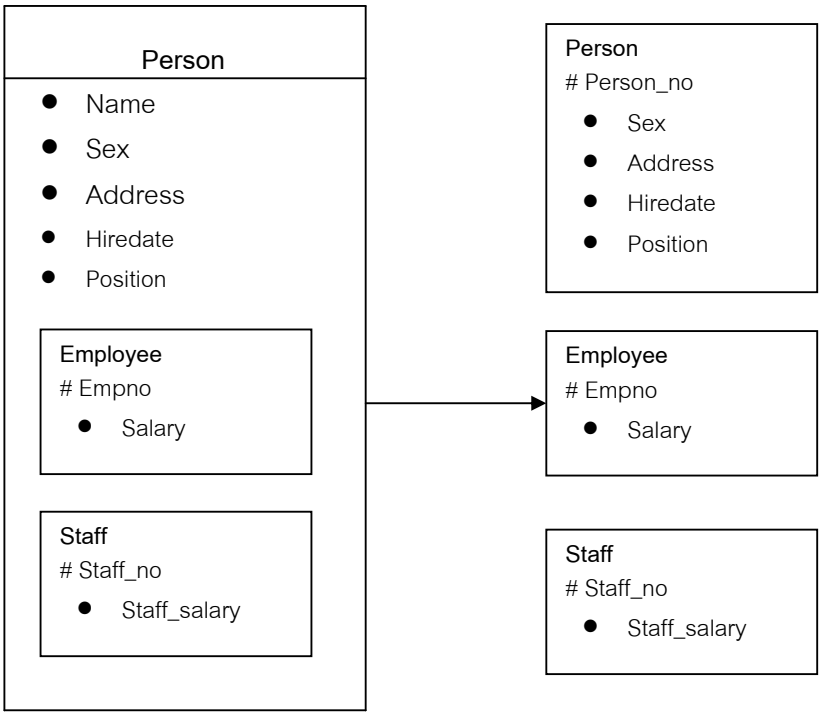
7.7 การแตกรีเลชั่นมากเกินไป (Overnormalization)

การทำให้เป็นบรรทัดฐาน มีวัตถุประสงค์เพื่อลดปัญหาในเรื่องความซ้ำซ้อนของข้อมูล และลดปัญหาในการจัดดำเนินการข้อมูลไม่ว่าจะเป็นการเพิ่ม ลบ หรือปรับปรุงข้อมูล บางครั้งอาจจะเกิดปัญหาที่ผู้ออกแบบฐานข้อมูลมีการแตกรีเลชั่น เป็นรีเลชั่นย่อยมากเกินความจำเป็น (Overnormalization)

ข้อเสียของการแตกรีเลชั่นมากเกินความจำเป็น จะส่งผลทำให้ระบบทำงานมีประสิทธิภาพไม่ดีนัก หากมีการเรียกใช้ข้อมูลที่ต้องเชื่อมโยงข้อมูลระหว่างรีเลชั่นอาจจะทำให้สิ้นเปลืองทรัพยากรและอาจทำให้ไม่มีประสิทธิภาพเท่าที่ควร นอกจากนี้ยังทำให้สูญเสียความสัมพันธ์ของข้อมูลที่จะใช้ประโยชน์จากรีเลชั่นนั้นโดยตรง

กล่าวคือ หากข้อมูลในรีเลชั่นถูกจัดดำเนินการในลักษณะการเพิ่ม ลบ หรือปรับปรุงข้อมูลบ่อยมาก ก็อาจจะพิจารณาในเรื่องการออกแบบรีเลชั่นโดยใช้แนวคิดการทำให้เป็นบรรทัดฐานที่เคร่งครัด เพื่อลดปัญหาความซ้ำซ้อนของข้อมูล ในทางตรงกันข้าม ถ้าข้อมูลในรีเลชั่นถูกเรียกใช้มาก (Select) ก็อาจจะไม่ใช้เกณฑ์การทำให้อยู่ในรูปแบบบรรทัดฐานที่เคร่งครัดมากนัก นั่นคือ อาจจะทำการดีนอร์มอลไลซ์รีเลชั่น (Denormalization) เพื่อเพิ่มประสิทธิภาพในการเรียกใช้ข้อมูล

ตัวอย่างเช่น จากรูป 7.14 เป็นการออกแบบฐานข้อมูลที่มีการแบ่งข้อมูลออกเป็น Superclass และ Subclass โดยการแตกเป็น 3 รีเลชั่น ทั้งๆ ที่รายละเอียดของแอททริบิวต์ของ Subclass ไม่มีความแตกต่างกัน

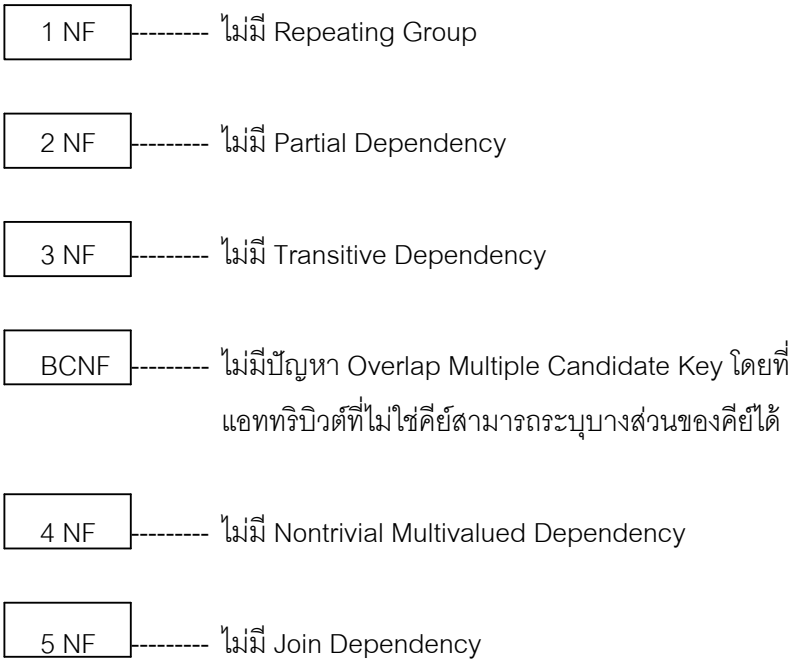


รูป 7.14

การแตกในลักษณะนี้ อาจจะมีข้อเสียคือ ทำให้ต้องมีการเชื่อมโยงกันมากและเปลืองเนื้อที่ในการจัดเก็บมาก ถึงแม้ว่าแต่ละรีเลชั่นจะสามารถเพิ่มเติมข้อมูลของ Subclass ได้ง่ายและการปรับปรุงข้อมูลจะทำได้ดีกว่าก็ตาม ดังนั้น ผู้ออกแบบฐานข้อมูลจะต้องพิจารณาในเรื่องของการปรับรีเลชั่นให้อยู่ให้รูปแบบบรรทัดฐานที่เหมาะสม โดยคำนึงถึงประเด็นปัญหาที่อาจจัดดำเนินการกับข้อมูล (การเพิ่มลบหรือปรับปรุงข้อมูล) และประเด็นประสิทธิภาพของระบบด้วยการพิจารณาถึงลักษณะการใช้งานข้อมูลของระบบนั้น ๆ ประกอบ

7.8 บทสรุป

การทำให้บรรทัดฐานเพื่อลดปัญหาความซ้ำซ้อนของข้อมูลและการจัดดำเนินการข้อมูลได้มีประสิทธิภาพ การทำให้เป็นบรรทัดฐานมีการตั้งรูปแบบบรรทัดฐานขั้นที่ 1 ถึงขั้นที่ 5 รวมถึงรูปแบบบรรทัดฐานแบบ Domain Key ขั้นตอนการทำให้เป็นบรรทัดฐานทั้ง 5 ขั้น สามารถสรุปได้ดังรูป



แบบฝึกหัด

7.1 ร้านเช่าวีดีโอแห่งหนึ่งได้ออกแบบฐานข้อมูลที่มีรีเลชั่นการเช่าวีดีโอ (Video Rental) ซึ่งประกอบด้วยรายละเอียดดังนี้

Video Rental	Video_ ID	Title	Qty	Rate	Rent date	Cust_ ID	Cust name	Cust_ tel	Return_ date

Video_ID	หมายถึง รหัสหมายเลขวีดีโอ
Title	หมายถึง ชื่อภาพยนตร์
Qty	หมายถึง จำนวนม้วนวีดีโอที่เช่า
Rate	หมายถึง อัตราค่าเช่าต่อม้วน
Rentdate	หมายถึง วันที่เริ่มเช่า
Cust_ID	หมายถึง รหัสลูกค้า
Custname	หมายถึง ชื่อลูกค้า
Cust_tel	หมายถึง หมายเลขโทรศัพท์ลูกค้า
Return_date	หมายถึง วันที่ต้องคืนวีดีโอ

ข้อมูลเพิ่มเติม

1. ภาพยนตร์ 1 เรื่อง จะมีหลาย Copy
2. ลูกค้าหนึ่งคน จะเช่าวีดีโอครั้งละได้หลายเรื่อง และในแต่ละเรื่องจะเช่าเพียงหนึ่งม้วนต่อครั้งหรือถ้ากรณีที่ 1 เรื่องมีหลายม้วนจะเช่าเพียง 1 ชุด ของเรื่องนั้น
3. ลูกค้าจะเช่าเพียงวันละหนึ่งครั้งเท่านั้น

คำสั่ง : ให้ทำรีเลชั่น Video Rental ให้อยู่ในรูปแบบบรรทัดฐานที่เหมาะสม

7.2 จากรีเลชั่น Project ให้ระบุปัญหาของการออกแบบและให้ปรับให้อยู่ในรูปแบบบรรทัดฐานที่เหมาะสม

Projno	Projname	Budget	Depname	Dep_Location	Depno
P1	New Product	100,000	Marketing	Silom	10
P2	Investment	550,000	Finance	Silom	20
P3	ISO 14000	4,000,000	Production	Bangna	30
P4	ISO 9000	2,000,000	Production	Bangna	30

Projno หมายถึง รหัสโครงการ
Projname หมายถึง ชื่อโครงการ
Budget หมายถึง งบประมาณของโครงการ
Depname หมายถึง แผนกที่รับผิดชอบโครงการ
Dep_Location หมายถึง ที่ตั้งของแผนกที่รับผิดชอบโครงการ
Depno หมายถึง รหัสแผนกที่รับผิดชอบโครงการ

7.3 จากแบบฝึกหัดข้อ 6.2 เอนทิตี้ Apartment_for_Rent ประกอบด้วยรายละเอียดดังนี้

Apt_no	Room_no	Type	Rent_rate	Status

Apt_no หมายถึง รหัสหอพัก
Room_no หมายถึง เลขที่ห้องพัก
Type หมายถึง ประเภทหอพัก
Rent_rate หมายถึง อัตราค่าเช่ารายเดือน
Status หมายถึง สถานะของห้องว่าว่างหรือมีผู้เช่าแล้ว

คำสั่ง : ให้ปรับเอนทิตี้ดังกล่าวให้อยู่ในรูปแบบบรรทัดฐานที่เหมาะสม