บทที่ 6 นอร์มัลไลเซชัน

วัตถุประสงค์ของบทเรียน

- ศึกษาเกี่ยวกับแนวคิดเกี่ยวกับนอร์มัลไลเซชันและความสำคัญของกระบวนการดังกล่าว
- ศึกษาเกี่ยวกับนอร์มัลฟอร์มในรูปแบบ 1NF, 2NF, 3NF, BCNF และ 4NF
- ศึกษาเกี่ยวกับวิธีการปรับเปลี่ยนตารางข้อมูลให้มีความสมบูรณ์ตามระดับต่างๆของนอร์มัลฟอร์ม
- ศึกษาเกี่ยวกับการประยุกต์ใช้แนวคิดนอร์มัลไลเซชันควบคู่กับการสร้างแบบจำลองข้อมูลเชิงสัมพันธ์
- ศึกษาเกี่ยวกับกรณีที่การดำเนินการดีนอร์มัลไลเซชันจะสามารถเพิ่มประสิทธิภาพในการจัดการข้อมูล

เนื้อหาของบทเรียน

ความต้องการในการประยุกต์ใช้นอร์มัลไลเซชัน ขั้นตอนการดำเนินการนอร์มัลไลเซชัน การปรับเปลี่ยนตารางข้อมูลให้อยู่รูปแบบ 1NF การปรับเปลี่ยนตารางข้อมูลให้อยู่รูปแบบ 2NF การปรับเปลี่ยนตารางข้อมูลให้อยู่รูปแบบ 3NF การพัฒนาการออกแบบโครงสร้างตารางข้อมูล การพิจารณาการใช้ surrogate key นอร์มัลฟอร์มระดับสูง การลดทอนระดับของนอร์มัลฟอร์ม ปัจจัยที่ต้องพิจารณาในการสร้างแบบจำลองข้อมูล

กิจกรรมการเรียน-การสอน

- อธิบายพร้อมยกตัวอย่างประกอบ
- ศึกษาจากเอกสารคำสอน
- ฝึกปฏิบัติการตามที่มอบหมาย
- ทำแบบฝึกหัดท้ายบท

อุปกรณ์ที่ใช้ในการเรียน-การสอน

- เอกสารคำสอน
- เครื่องคอมพิวเตอร์
- เครื่องฉายภาพสไลด์

การวัดและประเมินผล

- การตอบคำถามระหว่างการเรียน-การสอน
- การทำแบบทดสอบย่อยท้ายบท
- การตรวจงานตามที่มอบหมาย

การออกแบบฐานข้อมูลที่ดีจะส่งผลให้เราสามารถสร้างโครงสร้างการจัดเก็บข้อมูล (โครงสร้างตารางข้อมูล) ที่ดีได้ ดังนั้นในบทนี้เราจะทำการศึกษาเกี่ยวกับการออกแบบและการประเมินคุณภาพของโครงสร้างตารางข้อมูลเพื่อควบคุมความซ้ำซ้อนและลดทอนความผิดปกติของข้อมูล ศึกษาเกี่ยวกับขั้นตอนการออกแบบฐานข้อมูลให้มีคุณภาพที่เรียกว่า นอร์มัลไลเซชัน (Normalization) นอกเหนือจากเนื้อหาข้างต้น เราจะทำการศึกษาเกี่ยวกับคุณลักษณะของโครงสร้างตารางข้อมูลที่ดี และทำการศึกษาการปรับเปลี่ยน/แก้ไขโครงสร้างตารางข้อมูลที่ไม่ดีให้กลายเป็นโครงสร้างตารางข้อมูลที่ดี

จากเนื้อหาในบทก่อนๆหน้าจะทำให้เราทราบว่าตารางข้อมูล (เอ็นทิตี้) จะเป็นส่วนประกอบหนึ่งที่สำคัญของการออกแบบฐานข้อมูล ด้วยเหตุนี้จึงเป็นเหตุให้การออกแบบโครงสร้างของตารางข้อมูลสำหรับระบบฐานข้อมูลหนึ่งๆจะต้องมีการพิจารณาอย่างถี่ถ้วน การออกแบบครั้งหนึ่งๆอาจมีการกำหนดโครงสร้างตารางข้อมูลที่ดีหรือไม่ดีก็เป็นได้ ด้วยเหตุนี้จึงมีคำถามที่ว่าเราจะรู้ได้อย่างไรว่าตารางข้อมูลหนึ่งๆมีโครงสร้างตารางข้อมูลที่ไม่ดี? และเราจะสามารถทำการออกแบบหรือกำหนดโครงสร้างตารางข้อมูลที่ดีได้อย่างไร จากคำถามทั้งสองข้อ เราสามารถตอบได้ด้วยการประยุกต์ใช้ขั้นตอนนอร์มัลไลเซชันที่เป็นขั้นตอนสำหรับประเมินและปรับปรุงแก้ไขโครงสร้างตารางข้อมูลเพื่อที่จะลดทอนความซ้ำซ้อนและลดทอนความผิดปกติของข้อมูล

นอร์มัลไลเซชันจะเป็นการทำงานปรับเปลี่ยนรูปแบบโครงสร้างตารางหลายขั้นตอนเรียงต่อกัน โดยรูปแบบโครงสร้างที่ถูกปรับเปลี่ยนจะเรียกว่า นอร์มัลฟอร์ม (normal form) ที่ซึ่งจะประกอบไปด้วย first normal form (1NF), second normal form (2NF), third normal form (3NF) และ fourth normal form (4NF) ตามลำดับ โดยปกติของการออกแบบฐานข้อมูลของธุรกิจทั่วๆไปมักจะประยุกต์ใช้เพียงขั้นตอนการปรับเปลี่ยนตารางข้อมูลให้อยู่ในรูปแบบ 1NF, 2NF และ 3NF เพื่อที่จะได้โครงสร้างตารางข้อมูลที่ดี แต่อย่างไรก็ตาม ในการประยุกต์ใช้ขั้นตอนนอร์มัลไลเซชัน เราควรที่จะต้องพยายามดำเนินการตามนอร์มัลฟอร์มต่างๆไปจนกระทั่งนอร์มัลฟอร์มในระดับสูงสุดให้ได้ ซึ่งโดยปกติของนอร์มัลฟอร์มระดับสูงจะทำให้เกิดการเชื่อมโยงความสัมพันธ์ระหว่างตารางข้อมูลเพิ่มมากขึ้นที่ซึ่งจะทำให้การที่จะเข้าถึงข้อมูลเราจะต้องใช้การ JOIN ที่เพิ่มขึ้นด้วย (จะทำให้สิ้นเปลืองเวลาในการเข้าถึงข้อมูลเพิ่มมากขึ้น) ดังนั้น ในบางสถานการณ์เราอาจทำเป็นต้องลดทอนระดับของนอร์มัลฟอร์ม (เรียกว่า denormalization) เพื่อลดทอนเวลาในการเข้าถึงข้อมูล ตัวอย่างเช่น การลดทอนระดับของนอร์มัลฟอร์มจาก 3NF ไปเป็น 2NF เป็นต้น แต่อย่างไรก็ตาม การลดทอนระดับของนอร์มัลฟอร์มจะก่อให้เกิดความซ้ำซ้อนของข้อมูลเพิ่มขึ้น ดังนั้น เราควรที่จะพิจารณาถึงจุดสมดุลระหว่างความซ้ำซ้อนของข้อมูลและเวลาที่ใช้ในการเข้าถึงข้อมูลเพื่อที่จะได้ฐานข้อมูลที่มีคุณภาพ

6.1 ความต้องการในการประยุกต์ใช้นอร์มัลไลเซชัน

โดยปกติของการออกแบบฐานข้อมูลมักจะทำการประยุกต์ใช้นอร์มัลไลเซชันร่วมกับการออกแบบแบบจำลองข้อมูลเชิงสัมพันธ์เอ็นทิตี้ ที่ซึ่งจะประยุกต์ใช้นอร์มัลไลเซชันใน 2 สถานการณ์ คือ

1) เมื่อเราทำการออกแบบโครงสร้างการจัดเก็บข้อมูลจากความต้องการทางธุรกิจรวมถึงกฎเกณฑ์ทางธุรกิจ เราจะทำการออกแบบแบบจำลองข้อมูลด้วยการสร้าง ERD หลังจากสร้าง ERD เสร็จสิ้น เรา

สามารถประยุกต์ใช้นอร์มัลไลเซชันเพื่อ (i) วิเคราะห์ความสัมพันธ์ระหว่างแอทริบิวในแต่ละเอ็นทิตี้ และ (ii) พัฒนา/ปรับปรุงโครงสร้างการจัดเก็บข้อมูล และ

2) เมื่อเราถูกร้องขอให้ทำการปรับปรุง/แก้ไขโครงสร้างของข้อมูลที่ถูกจัดเก็บอยู่ในรูปแบบของแฟ้มข้อมูล เอกสารตารางข้อมูล และฐานข้อมูลเก่า เป็นต้น

ด้วยเหตุนี้เราสามารถประยุกต์ใช้นอร์มัลไลเซชันเพื่อทำการปรับปรุงโครงสร้างการจัดเก็บข้อมูลเดิมให้มีคุณภาพที่ดีขึ้นได้ จากสองสถานการณ์ข้างต้นจะเป็นการประยุกต์ใช้นอร์มัลไลเซชันสำหรับการกำหนดโครงสร้างข้อมูลใหม่หรือการประยุกต์ใช้นอร์มัลไลเซชันสำหรับปรับปรุง/พัฒนาโครงสร้างข้อมูลเดิมที่ซึ่งจะเป็นการดำเนินการนอร์มัลไลเซชันด้วยกระบวนการเดียวกัน

ในการที่จะมีความเข้าใจเกี่ยวกับกระบวนการนอร์มัลไลเซชันมากขึ้น ลองพิจารณาโครงสร้างการจัดเก็บข้อมูลของบริษัท KOM ที่ซึ่ง 1) บริษัทสามารถรับงานได้หลายงาน/โปรเจค 2) แต่ละโปรเจคจะมีรหัสโปรเจค ชื่อโปรเจค พนักงานที่ถูกกำหนดให้ทำงานในโปรเจค และ อื่นๆ 3) พนักงานแต่ละคนจะมีรหัสพนักงาน ชื่อ และตำแหน่งหน้าที่ เป็นต้น ในการที่จะคิดค่าใช้จ่ายของโปรเจคหนึ่งๆ บริษัท KOM จะคิดจากจำนวนชั่วโมงที่พนักงานทำงานที่ซึ่งจะมีอัตราค่าจ้างตามตำแหน่งต่อชั่วโมงการทำงานหนึ่งๆ ตัวอย่างเช่น ค่าจ้างของพนักงานคอมพิวเตอร์จะมีค่าจ้างแตกต่างจากค่าจ้างของวิศวกร ดังนั้นบริษัท KOM จะสามารถสร้างรายงานสรุปค่าใช้จ่ายได้ดังรูป 6.1

Table name: RPT_FORMAT **Database name: Ch06_ConstructCo**

PROJ_NUM	PROJ_NAME	EMP_NUM	EMP_NAME	JOB_CLASS	CHG_HOUR	HOURS
15	Evergreen	103	June E. Arbough	Elect. Engineer	84.50	23.8
		101	John G. News	Database Designer	105.00	19.4
		105	Alice K. Johnson *	Database Designer	105.00	35.7
		106	William Smithfield	Programmer	35.75	12.6
		102	David H. Senior	Systems Analyst	96.75	23.8
18	Amber Wave	114	Annelise Jones	Applications Designer	48.10	24.6
		118	James J. Frommer	General Support	18.36	45.3
		104	Anne K. Ramoras *	Systems Analyst	96.75	32.4
		112	Darlene M. Smithson	DSS Analyst	45.95	44.0
22	Rolling Tide	105	Alice K. Johnson	Database Designer	105.00	64.7
		104	Anne K. Ramoras	Systems Analyst	96.75	48.4
		113	Delbert K. Joenbrood *	Applications Designer	48.10	23.6
		111	Geoff B. Wabash	Clerical Support	26.87	22.0
		106	William Smithfield	Programmer	35.75	12.8
25	Starflight	107	Maria D. Alonzo	Programmer	35.75	24.6
		115	Travis B. Bawangi	Systems Analyst	96.75	45.8
		101	John G. News *	Database Designer	105.00	56.3
		114	Annelise Jones	Applications Designer	48.10	33.1
		108	Ralph B. Washington	Systems Analyst	96.75	23.6
		118	James J. Frommer	General Support	18.36	30.5
		112	Darlene M. Smithson	DSS Analyst	45.95	41.4

รูปที่ 6.1 ตัวอย่างตารางค่าแรงของพนักงานในโปรเจคต่างๆ

จากรูป 6.1 จะแสดงถึงโปรเจคต่างๆของบริษัท พนักงานที่ทำงานในแต่ละโปรเจค ตำแหน่งของพนักงาน ค่าแรงของพนักงานต่อชั่วโมง และจำนวนชั่วโมงที่พนักงานคนหนึ่งๆทำงานในโปรเจคหนึ่งๆ ตามลำดับ เมื่อเราทำการสังเกตุที่ข้อมูลในรูป 6.1 เราจะพบว่าพนักงานคนหนึ่งๆสามารถทำงานได้หลายโปรเจค ตัวอย่างเช่น Darlene Smithson (รหัสพนักงาน 112) จะทำงานในสองโปรเจคคือ Amber Wave และ Starflight ตามลำดับ และในรายงานจะไม่ทำการจัดเก็บค่าแรงที่ต้องจ่ายในแต่ละโปรเจค รวมถึงค่าแรงทั้งหมดที่ต้องจ่ายสำหรับทุกโปรเจค ดังนั้น เพื่อให้ได้มาซึ่งข้อมูลดังกล่าว เราจะต้องทำการคำนวณจาก ค่าแรงของพนักงานต่อชั่วโมง (CHG_HOUR) คูณกับ จำนวนชั่วโมงที่พนักงานคนหนึ่งๆทำงานในโปรเจคหนึ่งๆ (HOURS) ที่ซึ่งราคาค่าแรงทั้งหมดจะมีลักษณะเป็น derived attribute ที่ไม่ถูกจัดเก็บอยู่ในฐานข้อมูล

นอกเหนือจากกข้อสังเกตุข้างต้น เรายังสังเกตุได้อีกว่าโครงสร้างของข้อมูลที่แสดงในรูป 6.1 จะไม่สอดคล้องกับการขั้นตอนการออกแบบ ERD ในบทที่ 3 และ 4 ที่ซึ่งจะมีข้อบกพร่องต่างๆดังนี้

1. ข้อมูลรหัสโปรเจค (PROJ_NUM) ควรจะทำหน้าที่เป็น primary key หรือเป็นส่วนหนึ่งของ primary key แต่ข้อมูลรหัสโปรเจคในรูป 6.1 จะมีค่า NULL ในบางแถวข้อมูลที่ซึ่งจะทำให้ตารางข้อมูลขาดคุณสมบัติ entity integrity
2. ข้อมูลมีความซ้ำซ้อนที่ซึ่งจะก่อให้เกิดความผิดปกติของข้อมูลดังนี้
    a. Update anomalies—เช่น การปรับเปลี่ยนหน้าที่ในการทำงานของพนักงานรหัส 105 จะต้องทำการอัพเดทหลายแถวข้อมูลด้วยกัน ที่ซึ่งจะขัดกับหลักการอัพเดทข้อมูลที่ควรที่จะต้องทำการอัพเดทข้อมูลเพียงครั้งเดียว
    b. Insertion anomalies—เช่น การเพิ่มพนักงานใหม่ให้ทำงานในโปรเจคหนึ่งๆ ถ้าพนักงานคนหนึ่งๆยังไม่ถูกกำหนดให้ทำงานโปรเจคใดๆเลย เราจะต้องทำการสร้างโปรเจคสมมติขึ้นมาเพื่อจัดเก็บข้อมูลพนักงาน
    c. Deletion anomalies—เช่น พนักงานคนหนึ่งๆจะมีความสอดคล้องกับโปรเจคต่างๆ ถ้าพนักงานลาออกจากบริษัทจะทำให้ข้อมูลพนักงานถูกลบ และข้อมูลการทำงานโปรเจคต่างๆของพนักงานก็จะต้องถูกลบด้วยเช่นกัน แต่ถ้าเราไม่ต้องการให้ข้อมูลโปรเจคสูญหายไป เราจะต้องจำลองข้อมูลพนักงานสมมติขึ้นมาเพื่อคงไว้ซึ่งข้อมูลโปรเจคต่างๆ

จากข้อบกพร่องข้างต้นจึงเป็นเหตุให้มีความต้องการที่จะปรับเปลี่ยนแก้ไขโครงสร้างการจัดข้อมูลที่ซึ่งเราสามารถประยุกต์ใช้นอร์มัลไลเซชันเพื่อทำให้โครงสร้างข้อมูลมีคุณภาพที่ดีได้

6.2 ขั้นตอนการดำเนินการนอร์มัลไลเซชัน

ในส่วนนี้เราจะทำการศึกษาเกี่ยวกับการประยุกต์ใช้นอร์มัลไลเซชันเพื่อทำการสร้าง (ปรับเปลี่ยน) ตารางข้อมูลให้อยู่ในรูปแบบนอร์มัลฟอร์มเพื่อใช้ในการจัดเก็บข้อมูล วัตถุประสงค์ของการประยุกต์ใช้นอร์มัลไลเซชันคือการทำให้แน่ใจได้ว่าแต่ละตารางข้อมูลจะมีความสอดคล้องกับแนวความคิดของตารางข้อมูลที่ดีที่จะมีคุณลักษณะดังนี้

- แต่ละตารางข้อมูลจะถูกใช้ในการจัดเก็บข้อมูลประเภทหนึ่งๆเท่านั้น
- จะต้องไม่มีข้อมูลที่ไม่จำเป็นถูกจัดเก็บอยู่ในหลายๆตาราง
- แอทริบิวต่างๆในตารางข้อมูลที่ไม่ได้เป็น primary key หรือเป็นส่วนหนึ่งของ primary key จะต้องขึ้นกับ primary key ที่ซึ่งการขึ้นกับ primary key จะช่วยทำให้แน่ใจได้ว่าข้อมูลจะมีความเป็นเอกลักษณ์ และเราสามารถประยุกต์ใช้ primary key หนึ่งๆเพื่ออ้างถึงแถวข้อมูลหนึ่งๆได้
- แต่ละตารางข้อมูลจะต้องไม่มีปัญหาเกี่ยวกับ insertion anomaly, update anomaly และ deletion anomaly ที่ซึ่งจะทำให้แน่ใจได้ว่าข้อมูลจะมีความสมบูรณ์และความสอดคล้องซึ่งกันและกัน

ในการที่จะทำให้ตารางข้อมูลหนึ่งๆมีคุณลักษณะข้างต้น การประยุกต์ใช้นอร์มัลไลเซชันจะดำเนินการปรับเปลี่ยน/แก้ไขโครงสร้างของตารางข้อมูลที่ซึ่งจะปรับเปลี่ยนให้อยู่ในรูปแบบนอร์มัลฟอร์มในระดับที่สูงขึ้นที่ซึ่งแต่ละระดับของนอร์มัลฟอร์มจะมีคุณลักษณะเบื้องต้นดังรูป 6.2

NORMAL FORM	CHARACTERISTIC
First normal form (1NF)	Table format, no repeating groups, and PK identified
Second normal form (2NF)	1NF and no partial dependencies
Third normal form (3NF)	2NF and no transitive dependencies
Boyce-Codd normal form (BCNF)	Every determinant is a candidate key (special case of 3NF)
Fourth normal form (4NF)	3NF and no independent multivalued dependencies

รูปที่ 6.2 คุณลักษณะเบื้องต้นของนอร์มัลฟอร์ม

จากรูป 6.2 เราจะสังเกตุได้ว่านอร์มัลฟอร์มในระดับต่ำสุด (1NF) จะต้องการให้มีการกำหนด primary key ที่อาจเกิดจากการเลือกแอทริบิวหนึ่งๆหรือกลุ่มของแอทริบิวที่มีลักษณะเป็น candidate key ที่จะใช้ในการอ้างถึงข้อมูลแถวหนึ่งๆในตารางข้อมูล แต่อย่างไรก็ตาม ตารางข้อมูลหนึ่งๆอาจมีได้หลาย candidate key ที่ซึ่งอาจจะก่อให้เกิดความสับสนได้ ดังนั้น ในระหว่างขั้นตอนนอร์มัลไลเซชัน *เราจะทำการสมมติว่าในตารางข้อมูลหนึ่งๆจะมีเพียงแค่ 1 candidate key ที่ทำหน้าที่เป็น primary key เท่านั้น*

จากรูป 6.2 เราจะสังเกตุได้ว่านอร์มัลฟอร์มนั้นมีหลายระดับ แต่*โดยส่วนใหญ่ของการประยุกต์ใช้นอร์มัลไลเซชันมักจะมีวัตถุที่จะทำให้ตารางข้อมูลมีโครงสร้างในรูปแบบของ 3NF* ที่ซึ่งจะสามารถลดทอนความซ้ำซ้อนของข้อมูลได้ แต่อย่างไรก็ตาม อาจมีบางงานที่ต้องการที่จะทำการปรับเปลี่ยนโครงสร้างตารางข้อมูลให้อยู่ในรูปแบบนอร์มัลฟอร์มในระดับที่สูงขึ้นที่จะทำให้ต้องใช้เวลาในการเข้าถึง/จัดการกับข้อมูลที่มากขึ้น ด้วยเหตุนี้ ในการออกแบบฐานข้อมูลขององค์กรภาคธุรกิจมักจะทำการปรับเปลี่ยน/แก้ไขโครงสร้างข้อมูลให้อยู่ในรูปแบบของ 3NF เพื่อรักษาสมดุลระหว่างปริมาณความซ้ำซ้อนของข้อมูลและเวลาที่ใช้ในการเข้าถึงข้อมูล ดังนั้น เนื้อหาในบทนี้จะครอบคลุมถึงการปรับเปลี่ยนโครงสร้างตารางข้อมูลให้อยู่ในรูปแบบของ 3NF เท่านั้น

ก่อนที่จะทำการศึกษาเกี่ยวกับกระบวนการนอร์มัลไลเซชั่น เราควรที่จะทบทวนความเข้าใจในกรอบความคิดเกี่ยวกับ functional dependence ที่ซึ่งจะสามารถสรุปได้ดังรูปที่ 6.3

CONCEPT	DEFINITION
Functional dependence	The attribute *B* is fully functionally dependent on the attribute *A* if each value of *A* determines one and only one value of *B*. Example: PROJ_NUM → PROJ_NAME (read as "PROJ_NUM functionally determines PROJ_NAME") In this case, the attribute PROJ_NUM is known as the "determinant" attribute, and the attribute PROJ_NAME is known as the "dependent" attribute.
Functional dependence (generalized definition)	Attribute *A* determines attribute *B* (that is, *B* is functionally dependent on *A*) if all of the rows in the table that agree in value for attribute *A* also agree in value for attribute *B*.
Fully functional dependence (composite key)	If attribute *B* is functionally dependent on a composite key *A* but not on any subset of that composite key, the attribute *B* is fully functionally dependent on *A*.

รูปที่ 6.3 แนวความคิดเกี่ยวกับ functional dependence

ความเข้าใจเกี่ยวกับ functional dependence จะเป็นสิ่งสำคัญที่ซึ่งกรอบความคิดดังกล่าวจะใช้ในการกำหนด functional dependency (การขึ้นแก่กันของแอทริบิวหรือการพึ่งพาอาศัยกันของแอทริบิว) ของข้อมูลในตารางข้อมูลหนึ่งๆ ซึ่งโดยปกติของขั้นตอนนอร์มัลไลเซชันจะทำการพิจารณาตารางข้อมูลทีละตาราง จากนั้นจะทำการระบุถึง dependency ของข้อมูลในตารางข้อมูลที่พิจารณาแล้วทำการนอร์มัลไลซ์ dependency ของตารางข้อมูลนั้นๆ

functional dependency จะสามารถแบ่งออกได้เป็น 2 ประเภทคือ 1) partial dependency และ 2) transitive dependency โดย partial dependency จะปรากฏขึ้นเมื่อมี functional dependency ที่ซึ่งจะมีการพึ่งพาอาศัย (determines) แอทริบิวที่ทำหน้าที่เป็น candidate primary key เพียงบางส่วน ตัวอย่างเช่น ถ้ากำหนดให้ (A,B) เป็น composite primary key ที่เป็น candidate key และมี functional dependency เป็น (A, B) ⟶ (C,D) และ B ⟶ C ตามลำดับ ซึ่งจาก functional dependency ทั้งสอง เราจะสามารถสรุปได้ว่า functional dependency B ⟶ C จะมีลักษณะเป็น partial dependency เนื่องจาก C ที่เป็นแอทริบิวหนึ่งในตารางข้อมูลจะมีการขึ้นกับแอทริบิว B ที่ซึ่งเป็นส่วนหนึ่งของ primary key (เราสามารถสรุปได้ว่า แอทริบิว C จะขึ้นกับบางส่วนของ candidate primary key) แต่ในส่วนของ transitive dependency จะปรากฏขึ้นเมื่อมี functional dependency ระหว่างแอทริบิวที่ไม่ได้ทำหน้าที่เป็น primary key (หรือเป็นส่วนหนึ่งของ candidate primary key) ตัวอย่างเช่น ถ้ากำหนดให้ X เป็น candidate primary key และมี functional dependency เป็น X ⟶ Y, Y ⟶ Z ซึ่งจาก functional dependency ทั้งสอง เราจะสามารถสรุปได้ว่า Y ⟶ Z จะเป็น functional dependency ระหว่างแอทริบิวที่ไม่ได้ทำหน้าที่เป็น primary key ด้วยเหตุนี้ ในระหว่างขั้นตอนนอร์มัลไลเซชัน เราจะต้องทำการค้นหา functional dependency ทั้งสองประเภทข้างต้นในตารางข้อมูลหนึ่งๆแล้วทำการกำจัด functional dependency เหล่านั้นเพื่อที่จะทำการปรับเปลี่ยนโครงสร้างของตารางข้อมูลให้มีความสมบูรณ์มากยิ่งขึ้น

6.3.1 การปรับเปลี่ยนตารางข้อมูลให้อยู่รูปแบบ 1NF

การสร้างแบบจำลองข้อมูลหนึ่งๆจะเป็นการกำหนดโครงสร้างของเอ็นทิตี้หรือตารางข้อมูลต่างๆที่ซึ่งจะมีคีย์ต่างๆภายในตารางข้อมูลเหล่านั้น แต่เมื่อเราทำการพิจารณาข้อมูลในรูป 6.1 เราจะสังเกตุได้ว่าข้อมูล

จะมี “repeating group” ที่ซึ่งจะเป็นกลุ่มของแถวข้อมูลที่ขึ้นกับการปรากฏขึ้นครั้งหนึ่งๆของแอทริบิวที่ทำหน้าที่เป็น primary key ตัวอย่างเช่น การปรากฏขึ้นของรหัสโปรเจค (PROJ_NUM) แต่ละครั้งสามารถอ้างถึงข้อมูลหลายแถว เช่น การปรากฏขึ้นครั้งหนึ่งๆของรหัสโปรเจค 15 (PROJ_NUM = 15) ที่เป็นรหัสของโปรเจค Evergreen จะมีการอ้างถึงข้อมูลทั้งสิ้น 5 แถว โดยข้อมูลทั้ง 5 แถวนี้จะเป็นข้อมูลที่มีความเกี่ยวเนื่องกัน เนื่องจากข้อมูลเหล่านี้มีรหัสโปรเจคเดียวกัน จากการปรากฏขึ้นของ repeating group จะก่อให้เกิดความซ้ำซ้อนของข้อมูลที่ซึ่งเป็นสิ่งที่เราควรที่จะต้องหลีกเลี่ยงในการสร้างแบบจำลองข้อมูล ด้วยเหตุนี้ เราจึงควรที่จะต้องทำการประยุกต์ใช้วิธีการนอร์มัลไลเซชันเพื่อกำจัด repeating group ออกจากตารางข้อมูลที่ซึ่งจะทำให้แน่ใจได้ว่า primary หนึ่งๆจะต้องอ้างถึงข้อมูลเพียงแถวข้อมูลเดียวเท่านั้น การกำจัด repeating group ออกจากตารางข้อมูลจะช่วยทำให้เราทราบได้ว่าเราได้มีการพยายามที่จะทำให้ตารางข้อมูลอยู่ในรูปแบบของนอร์มัลฟอร์ม โดยขั้นตอนวิธีการนอร์มัลไลเซชันเพื่อทำให้ตารางข้อมูลอยู่ในรูปแบบของ 1NF จะประกอบไปด้วย 3 ขั้นตอนดังนี้

ขั้นตอนที่ 1: การกำจัด repeating groups

ขั้นตอนนี้จะเริ่มจากการแสดงข้อมูลในรูปแบบของตาราง (มีลักษณะเหมือนกันตารางข้อมูลสเปรดชีท) ที่ซึ่งแต่ละเซลล์จะมีค่าของข้อมูลที่ปรากฏเพียงค่าเดียวและแต่ละเซลล์ของข้อมูลจะไม่มี repeating group จากนั้นเราจะดำเนินการกำจัด repeating group ด้วยการลบค่า NULL ออกทั้งหมดเพื่อทำให้แน่ใจได้ว่าแต่ละแอทริบิวที่เป็น repeating group จะมีค่าของข้อมูลที่เหมาะสม ดังนั้นเมื่อเราพิจารณาข้อมูลในรูป 6.1 เราจะสามารถทำการปรับเปลี่ยนตารางข้อมูลให้เป็น 1NF ได้ดังแสดงในรูป 6.4

PROJ_NUM	PROJ_NAME	EMP_NUM	EMP_NAME	JOB_CLASS	CHG_HOUR	HOURS
15	Evergreen	103	June E. Arbough	Elect. Engineer	84.50	23.8
15	Evergreen	101	John G. News	Database Designer	105.00	19.4
15	Evergreen	105	Alice K. Johnson *	Database Designer	105.00	35.7
15	Evergreen	106	William Smithfield	Programmer	35.75	12.6
15	Evergreen	102	David H. Senior	Systems Analyst	96.75	23.8
18	Amber Wave	114	Annelise Jones	Applications Designer	48.10	24.6
18	Amber Wave	118	James J. Frommer	General Support	18.36	45.3
18	Amber Wave	104	Anne K. Ramoras *	Systems Analyst	96.75	32.4
18	Amber Wave	112	Darlene M. Smithson	DSS Analyst	45.95	44.0
22	Rolling Tide	105	Alice K. Johnson	Database Designer	105.00	64.7
22	Rolling Tide	104	Anne K. Ramoras	Systems Analyst	96.75	48.4
22	Rolling Tide	113	Delbert K. Joenbrood *	Applications Designer	48.10	23.6
22	Rolling Tide	111	Geoff B. Wabash	Clerical Support	26.87	22.0
22	Rolling Tide	106	William Smithfield	Programmer	35.75	12.8
25	Starflight	107	Maria D. Alonzo	Programmer	35.75	24.6
25	Starflight	115	Travis B. Bawangi	Systems Analyst	96.75	45.8
25	Starflight	101	John G. News *	Database Designer	105.00	56.3
25	Starflight	114	Annelise Jones	Applications Designer	48.10	33.1
25	Starflight	108	Ralph B. Washington	Systems Analyst	96.75	23.6
25	Starflight	118	James J. Frommer	General Support	18.36	30.5
25	Starflight	112	Darlene M. Smithson	DSS Analyst	45.95	41.4

รูปที่ 6.4 ตัวอย่างตารางข้อมูลที่ไม่มี NULL ในแอทริบิวที่เป็น repeating group

ขั้นตอนที่ 2: การกำหนด primary key

จากรูป 6.2 เราจะสังเกตุได้ว่าแอทริบิวรหัสโปรเจค (PROJ_NUM) ไม่สามารถทำหน้าที่เป็น primary key ได้ เนื่องจาก PROJ_NUM ไม่สามารถระบุได้ถึงแถวข้อมูลหนึ่ง (ไม่สามารถแยกความแตกต่างระหว่างแถวข้อมูลได้) ด้วยเหตุนี้เราจึงต้องทำการกำหนด primary key ให้กับตารางข้อมูลดังกล่าวที่ซึ่งจะเป็น composite primary key ที่เกิดจากการรวมกันระหว่างแอทริบิวรหัสโปรเจค (PROJ_NUM) และรหัสพนักงาน (EMP_NUM) ที่ซึ่งจะสามารถอ้างถึงแถวข้อมูลหนึ่งๆเท่านั้น

ขั้นตอนที่ 3: การระบุถึงการขึ้นแก่กัน/การพึ่งพาอาศัยระหว่างแอทริบิวต่างๆในตารางข้อมูล

หลังจากขั้นตอนการกำหนด primary key ให้กับตารางข้อมูล เราจะสามารถระบุได้ถึงการขึ้นแก่กัน/การพึ่งพาอาศัยกันระหว่างแอทริบิว (dependency) ได้เป็น

PROJ_NUM, EMP_NUM ⟶ PROJ_NAME, EMP_NAME, JOB_CLASS, CHG_HOUR, HOURS

ที่ซึ่งข้อมูลในแอทริบิว PROJ_NAME, EMP_NAME, JOB_CLASS, CHG_HOUR, HOURS จะขึ้นกับแอทริบิว PROJ_NUM และ EMP_NUM นอกเหนือจากการขึ้นแก่กันข้างต้นแล้วยังมีขึ้นแก่กันของแอทริบิวอื่นๆในตารางข้อมูลอีก เช่น รหัสโปรเจคหนึ่งๆจะสามารถระบุได้ถึงชื่อโปรเจคหนึ่งๆได้ ดังนั้นเราสามารถระบุได้ว่าชื่อโปรเจค (PROJ_NAME) จะขึ้นกับรหัสโปรเจค (PROJ_NUM) ที่ซึ่งเราจะสามารถแสดงการขึ้นแก่กันของแอทริบิวได้เป็น

PROJ_NUM ⟶ PROJ_NAME

นอกจากนั้น เรายังทราบได้อีกว่าถ้าเราทราบถึงรหัสพนักงานหนึ่งๆจะทำให้เราทราบถึงชื่อของพนักงาน ตำแหน่งของพนักงาน และอัตราค่าจ้างต่อชั่วโมงของพนักงานคนนั้นๆ ดังนั้น เราสามารถระบุถึงการขึ้นแก่กันของแอทริบิวได้เป็น

EMP_NUM ⟶ EMP_NAME, JOB_CLASS, CHG_HOUR

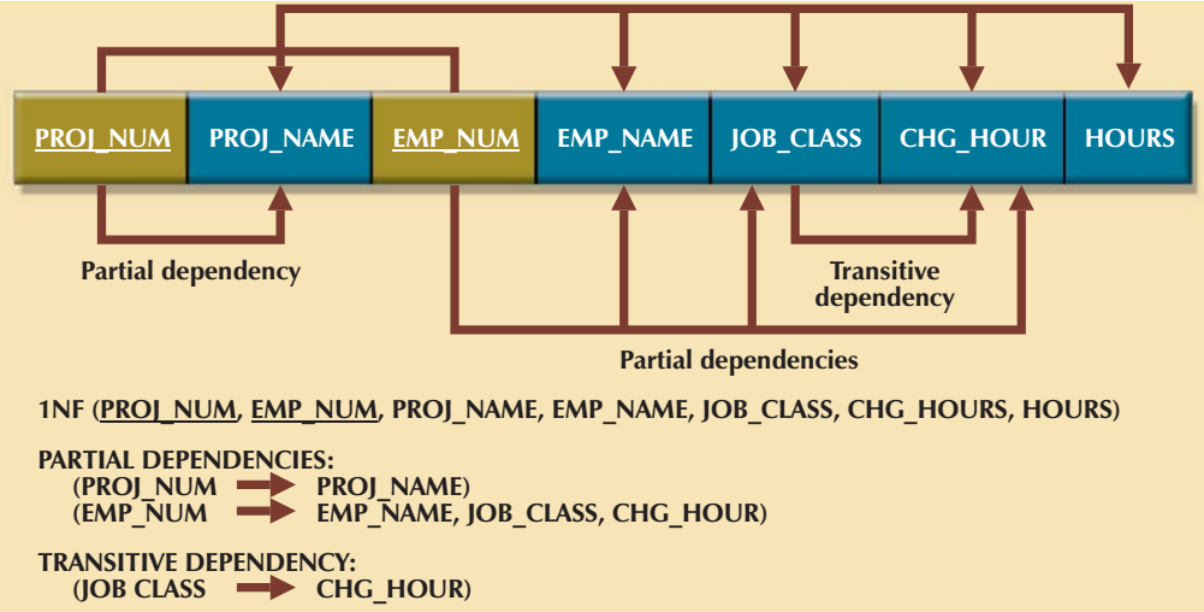
และท้ายสุด เราจะสามารถระบุได้ว่าถ้าเราทราบถึงตำแหน่งงานหนึ่งๆเราจะสามารถทราบถึงอัตราค่าจ้างต่อชั่วโมงของตำแหน่งๆงานนั้นๆได้ ดังนั้นเราจะสามารถระบุถึงการขึ้นแก่กันของแอทริบิวได้เป็น

JOB_CLASS ⟶ CHG_HOUR

การขึ้นแก่กันของแอทริบิวท้ายสุดจะเป็นการขึ้นแก่กันของแอทริบิวที่ไม่ได้เป็นหรือไม่ได้เป็นส่วนหนึ่งของ primary key ที่ซึ่งจะส่งสัญญาณบ่งบอกถึงการมี transitive dependency ในตารางข้อมูลที่ซึ่งเราจะต้องทำการพิจารณาต่อไป แต่สำหรับการขึ้นแก่กันของแอทริบิวทั้งหมดที่เราทำการพิจารณาข้างต้น เราจะสามารถแสดงให้อยู่ในรูปของแผนภาพการขึ้นแก่กันของแอทริบิว (dependency diagram) ได้ดังรูป 6.5 ที่ซึ่งจะช่วย

ให้เราเห็นภาพรวมของความสัมพันธ์ระหว่างแอทริบิวในเอ็นทิตี้หนึ่งๆได้ และเมื่อเราทำการพิจารณาแผนภาพการขึ้นแก่กันของแอทริบิวจะทำให้เราสังเกตุได้ว่า

- แอทริบิวที่ทำหน้าที่เป็น primary key จะถูกขีดเส้นใต้และจะถูกแรเงาด้วยสีที่แตกต่างจากแอทริบิวอื่นๆ
- ลูกศรบนแอทริบิวจะแสดงถึงการขึ้นแก่กันของแอทริบิวทั้งหมดที่ซึ่งการขึ้นแก่กันของแอทริบิวจะขึ้นกับ primary key ซึ่งจากตัวอย่างเราจะสังเกตุได้ว่าแอทริบิวทั้งหมด (แอทริบิวที่ไม่ได้เป็นส่วนหนึ่งของ primary key) จะขึ้นกับแอทริบิว PROJ_NUM และ EMP_NUM
- ลูกศรใต้แอทริบิวจะแสดงถึงการขึ้นแก่กันใน 2 ลักษณะ คือ
    - Partial dependencies—จะมี 2 การขึ้นแก่กันคือ 1) PROJ_NUM → PROJ_NAME ที่ซึ่ง PROJ_NAME จะขึ้นกับส่วนหนึ่งของ primary key (แอทริบิว PROJ_NUM) และ 2) EMP_NUM → EMP_NAME, JOB_CLASS, CHG_HOUR ที่ซึ่ง EMP_NAME, JOB_CLASS, CHG_HOUR จะขึ้นกับส่วนหนึ่งของ primary key (แอทริบิว EMP_NUM) ด้วยเช่นกัน
    - Transitive dependencies—จะมี 1 การขึ้นแก่กันคือ JOB_CLASS → CHG_HOUR ที่ซึ่งทั้งสองแอทริบิวนี้ไม่ได้ทำหน้าที่เป็น/เป็นส่วนประกอบของ primary key การมี transitive dependency



รูปที่ 6.5 ตัวอย่างแผนภาพการขึ้นแก่กันของแอทริบิว

นอกเหนือจากสิ่งที่เราสังเกตุได้จากข้างต้นแล้ว รูป 6.5 ยังแสดงถึงโครงสร้างของตารางข้อมูลที่อยู่ในรูปแบบของ 1NF ที่ยังคงมีแอทริบิวที่มีการขึ้นแก่กับแบบ partial dependency และ transitive dependency ที่ซึ่งจะก่อให้เกิดความซ้ำซ้อนของข้อมูลและความผิดปกติของข้อมูล

ความซ้ำซ้อนของข้อมูลภายใต้ partial dependency จะเกิดขึ้นจากการที่ทุกแถวข้อมูลจะต้องการข้อมูลที่ซ้ำกัน ตัวอย่างเช่น เมื่อ Alice K. Johnson ทำการกรอกข้อมูลชั่วโมงในการทำงาน เราจะต้องทำการกรอกข้อมูล EMP_NAME, JOB_CLASS และ CHG_CLASS ซ้ำ แม้ว่าข้อมูลเหล่านี้จะเหมือนกับที่ถูกกรอกไว้ก่อนหน้าแล้วก็ตาม จากการซ้ำที่มีข้อมูลซ้ำกันจะทำให้การทำงานไม่มีประสิทธิภาพและยังก่อให้เกิดความผิดปกติของข้อมูลด้วยเช่นกัน ตัวอย่างเช่น การกรอกชื่อพนักงาน เราไม่มีกระบวนการที่จะควบคุมการกรอกชื่อพนักงานคนหนึ่งในรูปแบบที่แตกต่างกัน เช่น การกรอกชื่อของพนักงานรหัส 102 ที่ซึ่งอาจทำการกรอกข้อมูลเป็น Dave Senior หรือ D. Senior เป็นต้น จากความผิดปกติของข้อมูลดังกล่าวจะทำให้ฐานข้อมูลขาดความสมบูรณ์และจะทำให้เกิดความไม่สอดคล้องของข้อมูลขึ้น

6.3.2 การปรับเปลี่ยนตารางข้อมูลให้อยู่รูปแบบ 2NF

การปรับเปลี่ยนตารางข้อมูลให้อยู่ในรูปแบบ 2NF จะถูกดำเนินการก็ต่อเมื่อตารางข้อมูลที่จะทำการปรับเปลี่ยนอยู่ในรูปแบบ 1NF และมี primary key ที่มีลักษณะเป็น composite primary key แต่ถ้าตารางข้อมูลมี primary key ทีเป็นแอทริบิวหนึ่งๆ เราจะถือว่าตารางข้อมูลนั้นๆอยู่ในรูปแบบของ 2NF โดยอัตโนมัติ ด้วยเหตุนี้ การปรับเปลี่ยนตารางข้อมูลจาก 1NF ให้เป็น 2NF จะสามารถดำเนินการได้โดยง่ายดังนี้

ขั้นตอนที่ 1: การสร้างตารางใหม่เพื่อกำจัด partial dependency

สำหรับ partial dependency หนึ่งๆที่เป็นการขึ้นแก่กันระหว่างแอทริบิวที่ไม่ได้เป็นส่วนหนึ่งของ primary key กับแอทริบิวที่เป็นส่วนหนึ่งของ primary key ในการที่จะกำจัด partial dependency หนึ่งๆ เราจะเริ่มจากการสร้างตารางข้อมูลใหม่สำหรับจัดเก็บข้อมูลแอทริบิวที่เป็นส่วนหนึ่งของ primary key (ทำการคัดลอกแอทริบิวจากตารางข้อมูลเดิม) และทำการกำหนดให้แอทริบิวที่ถูกคัดลอกทำหน้าที่เป็น primary key ของตารางใหม่ที่สร้างขึ้น โดยหลังจากสร้างตารางข้อมูลใหม่แล้วแอทริบิวที่ถูกคัดลอกจะยังคงอยู่ในตารางข้อมูลเดิมด้วยเนื่องจากแอทริบิวนี้จะทำหน้าที่เป็น foreign key ที่จะเชื่อมโยงความสัมพันธ์ระหว่างตารางข้อมูลเดิมและตารางข้อมูลใหม่ที่สร้างขึ้น ตัวอย่างเช่น จากรูป 6.5 ตารางข้อมูลจะมี primary key ที่ประกอบไปด้วยแอทริบิว PROJ_NUM และ EMP_NUM ที่ซึ่งแต่ละแอทริบิวจะมีการขึ้นแก่กันแบบ partial dependency ดังนั้น เราสามารถสร้างตารางใหม่ 2 คือ PROJECT และ EMPLOYEE ที่ซึ่งจะประกอบไปด้วยข้อมูล 1 แอทริบิว ดังนี้

PROJECT (PROJ_NUM)

EMPLOYEE (EMP_NUM)

นอกจากนั้นเรายังสามารถกำหนดโครงสร้างของตารางข้อมูลเดิมได้เป็น

ASSIGNMENT (PROJ_NUM, EMP_NUM, PROJ_NAME, EMP_NAME, JOB_CLASS, CHG_HOUR, HOURS)

ขั้นตอนที่ 2: การกำหนดแอทริบิวให้กับตารางข้อมูลที่สร้างขึ้นใหม่

การกำหนดแอทริบิวให้กับตารางข้อมูลใหม่ที่สร้างจากขั้นตอนที่ 1 จะสามารถพิจารณาได้จากลูกศรใต้แอทริบิวใน dependency diagram ของตารางข้อมูลเดิมที่ซึ่งจะแสดงถึงการขึ้นแก่กันของแอทริบิวต่างๆที่เป็นและไม่เป็นส่วนประกอบของ primary key (partial dependency และ transitive dependency) จากนั้นทำการพิจารณา partial dependency หนึ่งๆ แล้วทำการเคลื่อนย้ายแอทริบิวที่ไม่เป็นส่วนประกอบของ primary key ไปจัดเก็บในตารางข้อมูลใหม่ที่สร้างขึ้นจากขั้นตอนที่ 1 ที่ซึ่งจะมีแอทริบิวที่มีความเกี่ยวเนื่องกันในรูปแบบ partial dependency ถูกจัดเก็บอยู่ก่อนหน้า เมื่อทำการพิจารณาทุก partial dependency เสร็จสิ้น จะทำให้ตารางข้อมูลเดิมจะมีข้อมูลเพียงแอทริบิวที่เป็นส่วนประกอบของ primary key และ แอทริบิวที่ไม่ได้มีส่วนเกี่ยวข้องกับ partial dependency เท่านั้น ตัวอย่างเช่น

1) พิจารณา partial dependency ระหว่าง PROJ_NUM → PROJ_NAME จากนั้นนำแอทริบิว PROJ_NAME ที่มีการขึ้นกับแอทริบิว PROJ_NUM ไปจัดเก็บในตาราง PROJECT ที่ซึ่งมีแอทริบิว PROJ_NUM ถูกจัดเก็บไว้แล้ว ดังนั้น ตาราง PROJECT จะมีลักษณะเป็น

   PROJECT (PROJ_NUM, PROJ_NAME)

2) พิจารณา partial dependency ระหว่าง EMP_NUM → EMP_NAME, JOB_CLASS, CHG_HOUR จากนั้นนำแอทริบิว EMP_NAME, JOB_CLASS, CHG_HOUR ที่มีการขึ้นกับแอทริบิว EMP_NUM ไปจัดเก็บในตาราง EMPLOYEE ที่ซึ่งมีแอทริบิว EMP_NUM ถูกจัดเก็บไว้แล้ว ดังนั้น ตาราง EMPLOYEE จะมีลักษณะเป็น

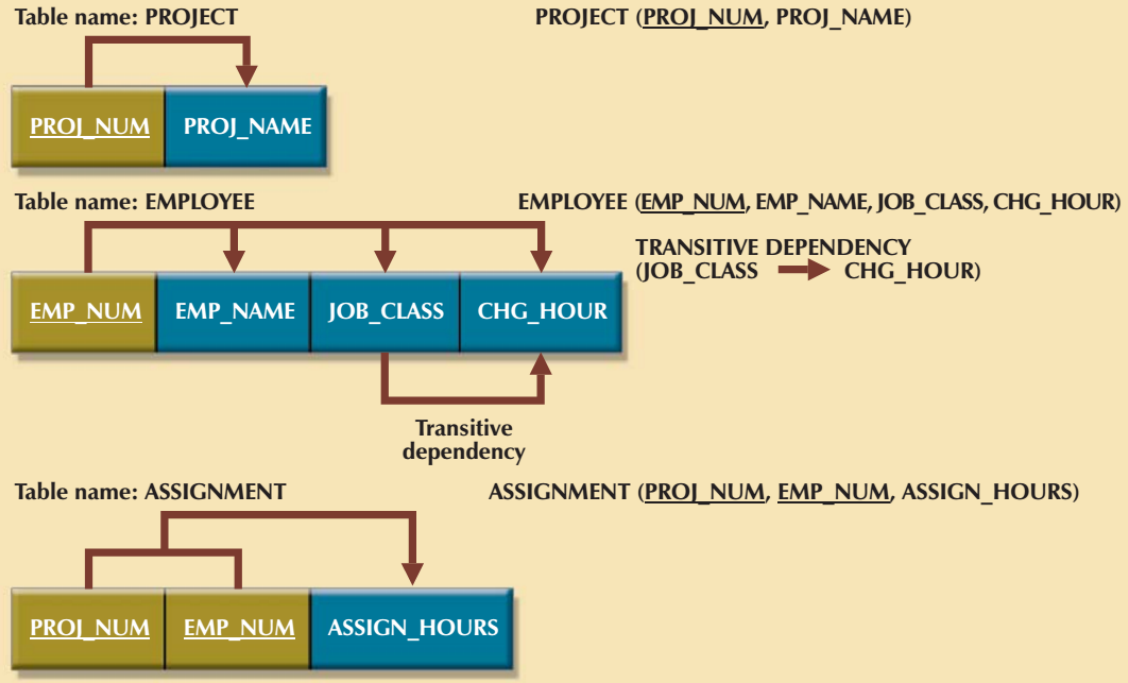
   EMPLOYEE (EMP_NUM, EMP_NAME, JOB_CLASS, CHG_HOUR)

3) ตารางข้อมูลเดิมหลังจากการพิจารณาทุก partial dependency จะมีลักษณะดังนี้

   ASSIGNMENT (PROJ_NUM, EMP_NUM, ASSIGN_HOURS)

จากข้างต้นเราจะสังเกตุได้ว่าในตารางข้อมูลหลัก (ASSIGNMENT) จะมีการเปลี่ยนแปลงชื่อแอทริบิวข้อมูลจากเดิมเป็น HOUR ให้กลายเป็น ASSIGN_HOURS ที่ซึ่งจะเป็นแอทริบิวของจำนวนชั่วโมงที่พนักงานแต่ละคนถูกกำหนดให้ทำงานในโปรเจคหนึ่งๆ นอกจากนั้น เรายังสังเกตุได้ว่าจากการดำเนินการในขั้นตอนที่ 1 และ 2 จะทำให้เราได้ 3 ตารางข้อมูลที่มีความสัมพันธ์กันดังแสดงในรูป 6.6 ที่ซึ่งตารางข้อมูลทั้งหมดจะมีความสัมพันธ์กันผ่านการกำหนดให้แอทริบิว PROJ_NUM และ EMP_NUM ทำหน้าที่เป็น foreign key ในตาราง ASSIGNMENT ที่ซึ่งจะสามารถกำจัดความผิดพลาดของข้อมูลออกไปได้ ตัวอย่างเช่น ถ้าเราต้องการที่จะทำการเพิ่ม เปลี่ยนแปลง หรือลบข้อมูลเกี่ยวกับโปรเจคหนึ่งๆ เราจะสามารถดำเนินการได้โดยการเข้าถึงข้อมูลแถวหนึ่งๆในตาราง PROJECT เท่านั้น แต่อย่างไรก็ตาม ขั้นตอนที่ 1 และขั้นตอนที่ 2 ของการปรับเปลี่ยนตารางข้อมูลให้มีรูปแบบเป็น 2NF จะทำการพิจารณาเฉพาะ partial dependency ดังนั้นตารางข้อมูลเดิมและตารางข้อมูลใหม่จะยังคงมี transitive dependency ที่ซึ่งจะยังคงก่อให้เกิดความผิดปกติ

ของข้อมูลได้ ตัวอย่างเช่น ถ้าอัตราค่าจ้างต่อชั่วโมงของตำแหน่งงานหนึ่งๆมีการเปลี่ยนแปลง เราจะต้องทำการแก้ไขข้อมูลในแถวข้อมูลพนักงานที่ทำงานในตำแหน่งนั้นๆที่ซึ่งอาจมีพนักงานหลายคน (หลายแถวข้อมูล) และถ้าเราหลงลืมที่จะทำการอัพเดทข้อมูลในบางแถวข้อมูลจะทำให้อัตราค่าจ้างของพนักงานที่ทำงานในตำแหน่งเดียวกันมีอัตราค่าจ้างไม่เท่ากันที่ซึ่งจะทำให้เกิดข้อผิดพลาดในการคำนวณต้นทุนหรือค่าใช้จ่ายในการทำโปรเจคหนึ่งๆได้



รูปที่ 6.6 ผลลัพธ์จากการปรับเปลี่ยนตารางข้อมูลให้อยู่ในรูปแบบ 2NF

6.3.3 การปรับเปลี่ยนตารางข้อมูลให้อยู่รูปแบบ 3NF

หลังจากทำการปรับเปลี่ยนตารางข้อมูลให้อยู่ในรูปแบบ 2NF แล้ว เราควรที่จะต้องทำการปรับเปลี่ยนตารางข้อมูลจากรูปแบบ 2NF ให้อยู่ในรูปแบบ 3NF เพื่อกำจัดความผิดปกติของข้อมูลในตารางข้อมูลเหล่านั้นด้วยการดำเนินการ 2 ขั้นตอนดังต่อไปนี้

ขั้นตอนที่ 1: การสร้างตารางข้อมูลใหม่เพื่อกำจัด transitive dependency

ขั้นตอนการทำงานจะเริ่มจากการพิจารณา transitive dependency หนึ่งๆของตารางข้อมูลหนึ่งๆ จากนั้นทำการสร้างตารางข้อมูลใหม่แล้วคัดลอกแอทริบิวในฝั่งซ้ายของการขึ้นแก่กัน (แอทริบิวที่เป็น “**determinant**” ที่ซึ่งจะเป็นแอทริบิวที่ถูกแอทริบิวอื่นๆพึ่งพาอาศัย; เพื่อที่จะเข้าใจมากขึ้นควรย้อนกลับไปศึกษาเกี่ยวกับ functional dependency) ไปจัดเก็บในตารางข้อมูลใหม่และกำหนดให้ทำหน้าที่เป็น primary key ของตารางข้อมูลใหม่ (หมายเหตุ—การคัดลอกแอทริบิวจะทำให้ตารางข้อมูลเดิมและตารางข้อมูลใหม่มีการเชื่อมโยงความสัมพันธ์กันผ่านแอทริบิวที่ถูกคัดลอกที่ซึ่งจะทำหน้าที่เป็น foreign key ในตารางข้อมูลเดิม) ตัวอย่างเช่น จากรูป 6.6 เราจะสังเกตุได้ว่าตารางข้อมูล EMPLOYEE จะมี transitive dependency ระหว่างแอทริบิว JOB_CLASS → CHG_HOUR ที่ซึ่งเราจะทำการสร้างตารางใหม่ที่ชื่อว่า

JOB ทำการคัดลอกแอทริบิว JOB_CLASS (แอทริบิวที่เป็น determinant) ไปจัดเก็บไว้ในตาราง JOB และกำหนดให้ JOB_CLASS ทำหน้าที่เป็น primary key ในตาราง JOB ดังนั้น เราจะได้ตารางข้อมูลใหม่ ดังนี้

JOB (JOB_CLASS)

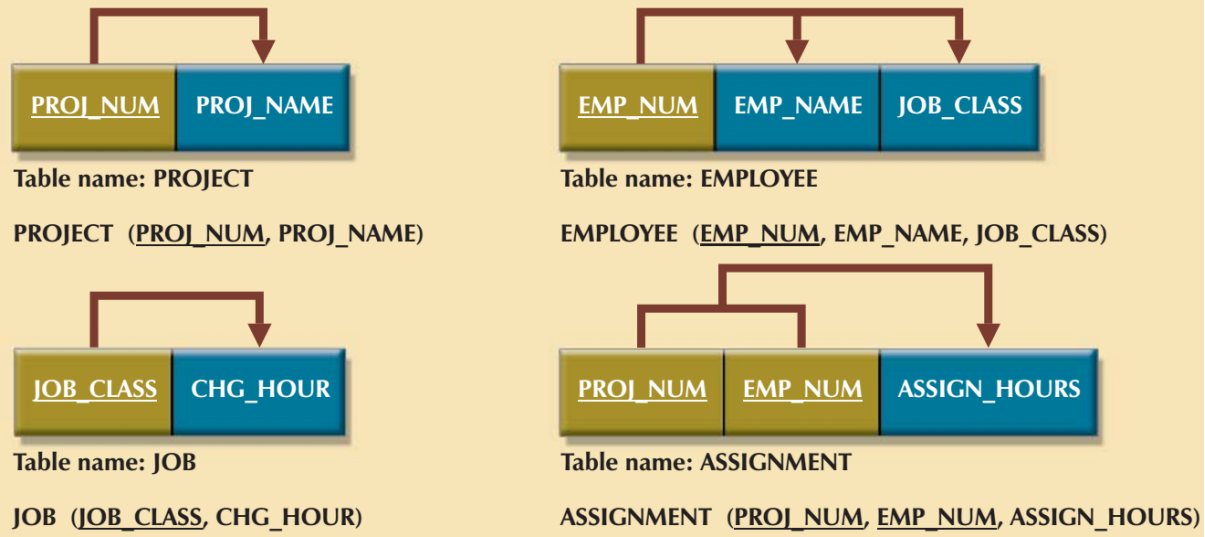
ขั้นตอนที่ 2: การกำหนดแอทริบิวให้กับตารางข้อมูลที่สร้างขึ้นใหม่

การกำหนดแอทริบิวให้กับตารางข้อมูลที่สร้างขึ้นใหม่จะเป็นการระบุถึงแอทริบิวที่มีส่วนเกี่ยวข้องกับ transitive dependency ที่ซึ่งแอทริบิวเหล่านี้ไม่ได้มีลักษณะเป็น determinant โดยเราจะทำการเคลื่อนย้ายแอทริบิวเหล่านั้นออกจากตารางข้อมูลเดิมไปจัดเก็บไว้ในตารางข้อมูลใหม่ที่สร้างขึ้น จากการดำเนินการดังกล่าวจะทำให้ตารางข้อมูลเดิมจะไม่มีแอทริบิวที่ไม่ได้มีลักษณะเป็น determinant ที่มีส่วนเกี่ยวข้องกับ transitive dependency หลงเหลืออยู่เลย ตัวอย่างเช่น จากตาราง EMPLOYEE ในรูป 6.6 ที่ซึ่งจะมีหนึ่ง transitive dependency คือ JOB_CLASS → CHG_HOUR ดังนั้น เราจะทำการเคลื่อนย้ายแอทริบิว CHG_HOUR (แอทริบิวที่ไม่ได้มีลักษณะเป็น determinant) ออกจากตาราง EMPLOYEE เพื่อไปจัดเก็บในตาราง JOB ที่สร้างขึ้นจากขั้นตอนที่ 1 ซึ่งจากการดำเนินการดังกล่าวจะทำให้ตาราง EMPLOYEE และ JOB มีลักษณะดังนี้

JOB (JOB_CLASS, CHG_HOUR)

EMPLOYEE (EMP_NUM, EMP_NAME, JOB_CLASS)

หลังจากดำเนินการปรับเปลี่ยนตารางข้อมูลให้อยู่ในรูปแบบ 3NF ด้วยขั้นตอนที่ 1 และ 2 ข้างต้น จะทำให้ฐานข้อมูลประกอบไปด้วย 4 ตารางข้อมูลดังแสดงในรูปที่ 6.7 จากนั้นเราควรที่จะต้องทำการตรวจสอบเพื่อให้แน่ใจได้ว่าแต่ละตารางข้อมูลไม่มี partial และ/หรือ transitive dependency หลงเหลืออยู่ โดยหลังจากการตรวจสอบ เราจะสามารถสรุปได้ว่าทั้ง 4 ตารางข้อมูลจะเป็นตารางข้อมูลในรูปแบบ 3NF



รูปที่ 6.7 ผลลัพธ์จากการปรับเปลี่ยนตารางข้อมูลให้เป็น 3NF

จากขั้นการปรับเปลี่ยนตารางข้อมูลให้อยู่ในรูปแบบ 1NF, 2NF และ 3NF เราจะสังเกตุได้ว่าการปรับเปลี่ยนตารางข้อมูลให้อยู่ในรูปแบบ 2NF และ 3NF จะมีความคล้ายคลึงกัน กล่าวคือ การสร้างตารางข้อมูลใหม่ การคัดลอกแอทริบิวที่ถูกพึ่งพาอาศัย (แอทริบิวทางฝั่งซ้ายของกฎ dependency หนึ่งๆ) ไปจัดเก็บไว้ในตารางใหม่และทำการกำหนดให้ทำหน้าที่เป็น primary key ของตารางใหม่ และขั้นตอนท้ายสุดคือ การเคลื่อนย้ายแอทริบิวที่มีลักษณะเป็น determinant ไปจัดเก็บไว้ในตารางข้อมูลใหม่ ตามลำดับ การปรับเปลี่ยนตารางข้อมูลให้อยู่ในรูปแบบ 2NF และ 3NF จะทำการพิจารณา partial dependency และ transitive dependency ที่เกิดขึ้นกับแอทริบิวในตารางต่างๆ

6.3 การพัฒนาการออกแบบโครงสร้างตารางข้อมูล

อย่างที่เราทราบดีว่า การออกแบบฐานข้อมูลจะเริ่มจากการสร้างแบบจำลองข้อมูล จากนั้นทำการประยุกต์ใช้นอร์มัลไลเซชันเพื่อกำจัด partial dependency และ transitive dependency ที่ซึ่งจะช่วยให้เราสามารถลดความซ้ำซ้อนและความผิดปกติของข้อมูลในตารางข้อมูลลงได้ แต่อย่างไรก็ตาม ในการที่จะสร้างตารางข้อมูลให้อยู่ในรูปแบบนอร์มัลไลเซชัน เราควรที่จะต้องพิจารณาปัจจัยอื่นๆที่จะสามารถช่วยให้โครงสร้างของตารางข้อมูลมีความสมบูรณ์มากขึ้นที่ซึ่งจะสามารถอธิบายด้วยการยกตัวอย่างดังนี้

การประเมินการกำหนด primary key (Evaluate PK assignments)

ในแต่ละครั้งที่ข้อมูลพนักงานใหม่คนหนึ่งๆถูกเพิ่มเข้าไปยังตาราง EMPLOYEE เราจะต้องทำการเพิ่มตำแหน่งงาน JOB_CLASS ของพนักงานคนนั้นๆด้วย แต่อย่างไรก็ตาม การดำเนินการดังกล่าวอาจก่อให้เกิดข้อผิดพลาดในการเพิ่มข้อมูลที่ซึ่งจะเป็นการฝ่าฝืนกฎ referential integrity ตัวอย่างเช่น เมื่อเราเพิ่มข้อมูลตำแหน่งงาน DB Designer แทนที่จะเป็น Database Designer เข้าไปในแอทริบิว JOB_CLASS ในตาราง EMPLOYEE จะทำให้เกิด การฝ่าฝืนกฎ referential integrity ดังนั้น เราอาจทำการเพิ่มแอทริบิวรหัสตำแหน่ง JOB_CODE ที่มีลักษณะเป็น surrogate key เข้าไปในตาราง JOB และทำการกำหนดให้ JOB_CODE ทำหน้าที่เป็น primary key ที่ซึ่งจะทำให้เกิดการขึ้นแก่กันของแอทริบิวและจะทำให้ตาราง JOB มีแอทริบิวดังนี้

JOB_CODE → JOB_CLASS, CHG_HOUR

JOB (<u>JOB_CODE</u>, JOB_CLASS, CHG_HOUR)

การประเมินการตั้งชื่อแอทริบิวต่างๆ (Evaluate Naming Conventions)

การตั้งชื่อแอทริบิวหนึ่งๆให้สามารถเข้าใจได้ง่ายจะสามารถช่วยให้ผู้ออกแบบฐานข้อมูล ผู้เขียนโปรแกรมและผู้ใช้สามารถเข้าใจเกี่ยวกับข้อมูลที่ถูกจัดเก็บในแอทริบิวนั้นๆได้โดยง่าย ด้วยเหตุนี้ เราควรที่จะทำการเปลี่ยนชื่อแอทริบิว CHG_HOUR ไปเป็น JOB_CHG_HOUR เพื่อแสดงถึงอัตราค่าจ้างในตำแหน่งงานหนึ่งๆ และทำการปรับเปลี่ยนชื่อแอทริบิว JOB_CLASS ไปเป็น JOB_DESCRIPTION เพื่อที่จะสื่อความหมายของข้อมูลที่ถูกจัดเก็บในแอทริบิวได้ดีขึ้น

การพิจารณาแอทริบิวที่เป็นไม่สามารถแบ่งย่อยได้ (Refine attribute atomicity)

แอทริบิวที่ไม่สามารถแบ่งย่อยได้ (atomic attribute) จะเป็นแอทริบิวที่ไม่สามารถแบ่งออกเป็นแอทริบิวย่อยๆหลายแอทริบิวได้ ดังนั้น เราควรที่จะพยายามทำให้แอทริบิวในตารางเป็นแอทริบิวที่ไม่สามารถแบ่งย่อยได้ ตัวอย่างเช่น แอทริบิว EMP_NAME จะเป็นแอทริบิวที่สามารถแบ่งย่อยได้ โดยสามารถแบ่งย่อยได้เป็นชื่อ ชื่อกลาง และ นามสกุล โดยเราสามารถปรับเปลี่ยนการจัดเก็บข้อมูลจาก EMP_NAME ให้กลายเป็น EMP_LNAME, EMP_FNAME และ EMP_INITIAL ตามลำดับ ซึ่งจากการแบ่งแอทริบิวให้เป็นส่วนย่อยๆจะทำให้เราสามารถสร้างลิสต์เบอร์โทรศัพท์โดยเรียงลำดับตามนามสกุลของพนักงานได้ แต่ถ้าเราไม่ทำการแบ่งแอทริบิวออกเป็นแอทริบิวย่อยๆจะทำให้เราดำเนินการข้างต้นได้ยาก

การระบุแอทริบิวใหม่ (Identify new attributes)

ถ้าตาราง EMPLOYEE ถูกประยุกต์ใช้จริง เราจะต้องเพิ่มแอทริบิวต่างๆเพื่อจัดเก็บข้อมูล ตัวอย่างเช่น รหัสประจำตัวประชาชน วันที่เริ่มทำงาน และอื่นๆ ที่บ่งบอกถึงข้อมูลอื่นๆของพนักงานที่จำเป็น ดังนั้นเราควรที่จะทำการพิจารณาว่าแต่ละตารางข้อมูลมีการจัดเก็บข้อมูลแอทริบิวที่จำเป็นต่อการประมวลผลครบถ้วนหรือไม่

การระบุถึงความสัมพันธ์ใหม่ (Identify new relationships)

จากรายงานตั้งต้นในรูป 6.1 ผู้บริหารองค์กรอาจมีความต้องการที่จะตรวจสอบว่าพนักงานรายใดทำหน้าที่เป็นหัวหน้าโปรเจคหนึ่งๆ จากความต้องการดังกล่าวจะทำให้เราต้องเพิ่มเติมการพิจารณาความสัมพันธ์ระหว่างตาราง EMPLOYEE และ PROJECT โดยการกำหนดให้แอทริบิว EMP_NUM (ข้อมูลรหัสพนักงานที่ทำหน้าที่เป็น primary key ในตาราง EMPLOYEE) ทำหน้าที่เป็น foreign key ในตาราง PROJECT จากตัวอย่างข้างต้น เราควรที่จะต้องทำการทบทวนความต้องการต่างๆของผู้ใช้และทำการระบุถึงความสัมพันธ์ระหว่างตารางข้อมูลเพิ่มเติมเพื่อตอบสนองต่อความต้องการของผู้ใช้

การพิจารณา primary key เพื่อปรับเปลี่ยนระดับความละเอียดของข้อมูล (Refine PK as required for data granularity)

ความละเอียดของข้อมูล (data granularity) คือระดับของรายละเอียดของค่าของข้อมูลที่ปรากฏในแถวข้อมูลหนึ่งๆในตารางหนึ่งๆ โดยข้อมูลที่ถูกจัดเก็บในระดับความละเอียดสูงสุดจะเรียกว่า “atomic data” ตัวอย่างเช่น ข้อมูลในแอทริบิว ASSIGN_HOURS ในตาราง ASSIGNMENT จะแสดงถึงจำนวนชั่วโมงที่พนักงานคนหนึ่งๆทำงานในโปรเจคหนึ่งๆ แต่อย่างไรก็ตาม เราอาจตั้งคำถามที่ว่า ค่าของข้อมูลที่ปรากฏในแอทริบิว ASSIGN_HOURS จะเป็นค่าของข้อมูลที่มีระดับความละเอียดสูงสุดหรือไม่? หรือ แอทริบิว ASSIGN_HOURS สามารถแสดงถึงจำนวนชั่วโมง จำนวนวัน จำนวนสัปดาห์ จำนวนเดือน และจำนวนปีที่พนักงานคนหนึ่งๆทำงานได้หรือไม่? จากคำถามทั้งสอง เราควรที่จะต้องทำการพิจารณาถึงระดับความละเอียดของข้อมูลที่จะถูกจัดเก็บในแอทริบิว ที่ซึ่งจะต้องสอดคล้องกับการดำเนินการต่างๆกับข้อมูล

การจัดการกับข้อมูลที่มีการเปลี่ยนแปลงได้อย่างถูกต้อง (Maintain historical accuracy)

ข้อมูลเกี่ยวกับอัตราค่าจ้างของพนักงานตำแหน่งต่างๆต่อหนึ่งชั่วโมง (JOB_CHG_HOUR) อาจเป็นข้อมูลที่มีการเปลี่ยนแปลงที่ซึ่งจะส่งผลต่อความถูกต้องของการคำนวณค่าใช้จ่ายของโปรเจคหนึ่งๆ โดยค่าของข้อมูลอาจมีการปรับเพิ่มขึ้นเมื่อเวลาผ่านไป ดังนั้นเราควรที่จะที่จะต้องเตรียมการจัดการกับการจัดเก็บข้อมูลที่มีการเปลี่ยนแปลงทั้งหมด

การประเมินการประยุกต์ใช้ derived attribute (Evaluate using derived attributes)

อย่างที่เราทราบดีว่าข้อมูลใน derived attribute จะเป็นแอทริบิวที่ได้มาจากการประมวลผลข้อมูลในแอทริบิวอื่นๆ ตัวอย่างเช่น ถ้าเราต้องการคำนวนค่าจ้างทั้งหมดในโปรเจคหนึ่งๆ เราจะต้องทำการกำหนด derived attribute ใหม่ที่ชื่อว่า ASSIGN_CHARGE ที่ได้มาจากการคูณกันระหว่าง ASSIGN_HOURS และ JOB_CHG_HOUR ที่ซึ่งเราสามารถเลือกได้ว่าจะทำการจัดเก็บ derived attribute ไว้ในฐานข้อมูลหรือไม่ โดยในการตัดสินใจเราควรที่จะต้องคำนึงถึงความสมดุลระหว่างพื้นที่ในการจัดเก็บข้อมูลที่ต้องเพิ่มขึ้นและเวลาที่ใช้ในการคำนวณที่ต้องเพิ่มขึ้นด้วยเช่นกัน

จากปัจจัยที่ต้องพิจารณาทั้งหมดข้างต้น เมื่อเราทำการพิจารณาทุกปัจจัยกับตารางข้อมูลในรูป 6.7 จะทำให้เราสามารถเพิ่มเติมรายละเอียดของตารางข้อมูลต่างๆที่จะทำให้เกิดความสมบูรณ์มากขึ้นดังแสดงในรูปแบบ dependency diagram ในรูป 6.8, 6.9 และ 6.10

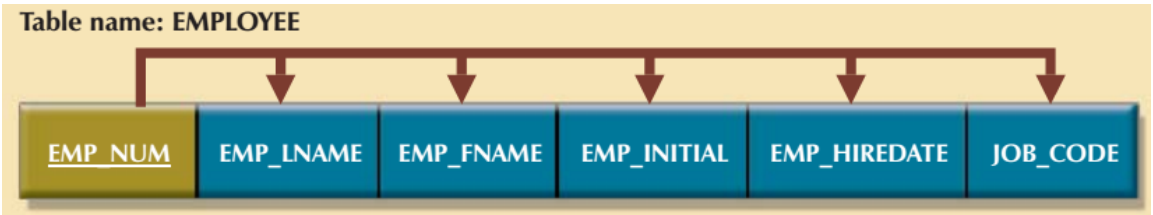


Table name: EMPLOYEE

EMP_NUM	EMP_LNAME	EMP_FNAME	EMP_INITIAL	EMP_HIREDATE	JOB_CODE
101	News	John	G	08-Nov-00	502
102	Senior	David	H	12-Jul-89	501
103	Arbough	June	E	01-Dec-97	503
104	Ramoras	Anne	K	15-Nov-88	501
105	Johnson	Alice	K	01-Feb-94	502
106	Smithfield	William		22-Jun-05	500
107	Alonzo	Maria	D	10-Oct-94	500
108	Washington	Ralph	B	22-Aug-89	501
109	Smith	Larry	W	18-Jul-99	501
110	Olenko	Gerald	A	11-Dec-96	505
111	Wabash	Geoff	B	04-Apr-89	506
112	Smithson	Darlene	M	23-Oct-95	507
113	Joenbrood	Delbert	K	15-Nov-94	508
114	Jones	Annelise		20-Aug-91	508
115	Bawangi	Travis	B	25-Jan-90	501
116	Pratt	Gerald	L	05-Mar-95	510
117	Williamson	Angie	H	19-Jun-94	509
118	Frommer	James	J	04-Jan-06	510

รูปที่ 6.8 ตัวอย่างตาราง EMPLOYEE ที่สมบูรณ์และสามารถสนับสนุนการดำเนินการต่างๆ

Database name: Ch06_ConstructCo

Table name: PROJECT

PROJ_NUM	PROJ_NAME	EMP_NUM
15	Evergreen	105
18	Amber Wave	104
22	Rolling Tide	113
25	Starflight	101

Table name: JOB

JOB_CODE	JOB_DESCRIPTION	JOB_CHG_HOUR
500	Programmer	35.75
501	Systems Analyst	96.75
502	Database Designer	105.00
503	Electrical Engineer	84.50
504	Mechanical Engineer	67.90
505	Civil Engineer	55.78
506	Clerical Support	26.87
507	DSS Analyst	45.95
508	Applications Designer	48.10
509	Bio Technician	34.55
510	General Support	18.36

รูปที่ 6.9 ตัวอย่างตาราง PROJECT และ JOB ที่สมบูรณ์และสามารถสนับสนุนการดำเนินการต่างๆ

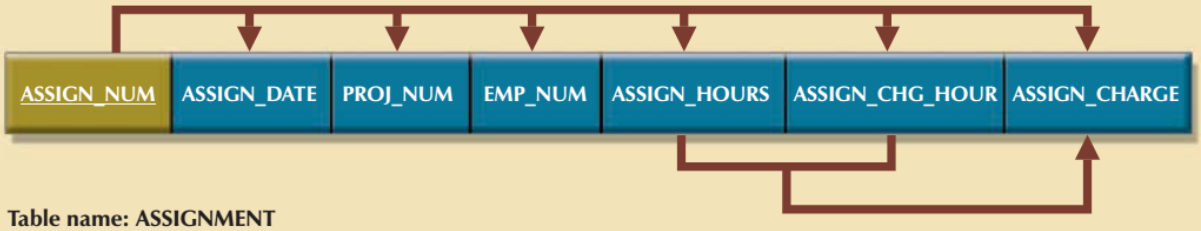


Table name: ASSIGNMENT

ASSIGN_NUM	ASSIGN_DATE	PROJ_NUM	EMP_NUM	ASSIGN_HOURS	ASSIGN_CHG_HOUR	ASSIGN_CHARGE
1001	04-Mar-10	15	103	2.6	84.50	219.70
1002	04-Mar-10	18	118	1.4	18.36	25.70
1003	05-Mar-10	15	101	3.6	105.00	378.00
1004	05-Mar-10	22	113	2.5	48.10	120.25
1005	05-Mar-10	15	103	1.9	84.50	160.55
1006	05-Mar-10	25	115	4.2	96.75	406.35
1007	05-Mar-10	22	105	5.2	105.00	546.00
1008	05-Mar-10	25	101	1.7	105.00	178.50
1009	05-Mar-10	15	105	2.0	105.00	210.00
1010	06-Mar-10	15	102	3.8	96.75	367.65
1011	06-Mar-10	22	104	2.6	96.75	251.55
1012	06-Mar-10	15	101	2.3	105.00	241.50
1013	06-Mar-10	25	114	1.8	48.10	86.58
1014	06-Mar-10	22	111	4.0	26.87	107.48
1015	06-Mar-10	25	114	3.4	48.10	163.54
1016	06-Mar-10	18	112	1.2	45.95	55.14
1017	06-Mar-10	18	118	2.0	18.36	36.72
1018	06-Mar-10	18	104	2.6	96.75	251.55
1019	06-Mar-10	15	103	3.0	84.50	253.50
1020	07-Mar-10	22	105	2.7	105.00	283.50
1021	08-Mar-10	25	108	4.2	96.75	406.35
1022	07-Mar-10	25	114	5.8	48.10	278.98
1023	07-Mar-10	22	106	2.4	35.75	85.80

รูปที่ 6.10 ตัวอย่างตาราง ASSIGNMENT ที่สมบูรณ์และสามารถสนับสนุนการดำเนินการต่างๆ

6.4 การพิจารณาการใช้ surrogate key

จากเนื้อหาในบทที่ 5 เราจะทราบถึงนิยามและการประยุกต์ใช้ surrogate key ให้ทำหน้าที่เป็น primary key ในตารางข้อมูลหนึ่งๆ ที่ซึ่งจะช่วยแก้ไขปัญหาจากการประยุกต์ใช้ composite primary key ได้ เช่น การกำหนดให้ primary key ของตารางข้อมูลหนึ่งๆทำหน้าที่เป็น foreign key ในอีกตารางข้อมูลหนึ่ง แต่ถ้า primary key มีลักษณะเป็น composite primary key ที่ซึ่งประกอบไปด้วยหลายแอทริบิว จะทำให้เราต้องจัดเก็บ foreign key เป็นจำนวนหลายแอทริบิวด้วยเช่นกัน ด้วยเหตุนี้ เราจึงควรที่จะพิจารณาเกี่ยวกับการประยุกต์ใช้ surrogate key แทนที่การใช้ composite primary key

surrogate key มักจะอยู่ในรูปแบบของตัวเลขที่ซึ่งค่าของข้อมูลหนึ่งๆจะถูกสร้างจากการเพิ่มขึ้นของค่าของข้อมูลล่าสุดที่ถูกเพิ่มในตารางข้อมูล แต่อย่างไรก็ตาม การประยุกต์ใช้ surrogate key อาจทำให้เกิดปัญหาการซ้ำกันของข้อมูล ตัวอย่างเช่น จากรูป 6.9 เราได้กำหนดให้แอทริบิวรหัสตำแหน่ง (JOB_CODE) เป็น surrogate primary key ที่ซึ่งค่าของข้อมูลรหัสตำแหน่งจะไม่ซ้ำกัน แต่สำหรับค่าในแอทริบิวอื่นๆอาจมีค่าเหมือนกันทั้งหมดดังแสดงในรูป 6.11

JOB_CODE	JOB_DESCRIPTION	JOB_CHG_HOUR
511	Programmer	$35.75
512	Programmer	$35.75

รูปที่ 6.11 ตัวอย่างการซ้ำกันของข้อมูลเมื่อประยุกต์ใช้ surrogate key

จากรูป 6.11 เราจะสังเกตุได้ว่ามีข้อมูลตำแหน่งงานซ้ำกัน (แต่มีรหัสตำแหน่งแตกต่างกัน) ที่ซึ่งเป็นปัญหาที่เกิดจากการประยุกต์ใช้ surrogate primary key แต่อย่างไรก็ตามเราสามารถหลีกเลี่ยงได้ด้วยการกำหนดให้ค่าของข้อมูลที่ปรากฏในแอทริบิว JOB_DESCRIPTION มีความเป็นเอกลักษณ์ (ไม่สามารถซ้ำกันได้) ที่ซึ่งจะทำให้ไม่สามารถมีข้อมูลตำแหน่งงานซ้ำกันถูกจัดเก็บในตารางข้อมูลได้

จากการดำเนินข้างต้น เราจะเห็นว่าเมื่อเราประยุกต์ใช้ surrogate key เราอาจจำเป็นต้องกำหนดเงื่อนไขเพิ่มเติมให้กับแอทริบิวอื่นๆเพื่อหลีกเลี่ยงความซ้ำซ้อนของข้อมูล แต่อย่างไรก็ตาม ในการออกแบบโครงสร้างตารางข้อมูล เราควรที่จะต้องคำนึงจุดสมดุลระหว่างความสมบูรณ์ของข้อมูลและความยืดหยุ่น โดยอาจไม่จำเป็นต้องกำหนดเงื่อนไขเพิ่มเติมในทุกๆตารางข้อมูล (การกำหนดเงื่อนไขทำให้ความยืดหยุ่นของการปรากฏขึ้นของข้อมูลลดลง) หรืออาจทำการกำหนดเงื่อนไขเมื่อจะเป็นเท่านั้น

6.5 นอร์มัลฟอร์มระดับสูง

อย่างที่เราทราบก่อนหน้าว่าตารางข้อมูลที่อยู่ในรูปแบบของ 3NF จะสามารถตอบสนองต่อการจัดเก็บข้อมูลในการดำเนินธุรกิจต่างๆได้ แต่อย่างไรก็ตาม อาจมีบางสถานการณ์ที่ต้องการที่จะปรับเปลี่ยนตารางข้อมูลให้อยู่ในรูปแบบของนอร์มัลฟอร์มที่มีระดับสูงกว่า 3NF ที่ซึ่งจะสามารถแสดงรายละเอียดได้ดังนี้

6.5.1 Boyce-Codd Normal Form (BCNF)

ตารางข้อมูลหนึ่งๆจะอยู่ในรูปแบบของ Boyce-Codd Normal Form (BCNF) ก็ต่อเมื่อทุกๆแอทริบิวที่ถูกพึ่งพาอาศัย (แอทริบิวที่เป็น determinant) ในตารางข้อมูลทำหน้าที่เป็น candidate key (หมายเหตุ—candidate key จะเป็นแอทริบิวหรือกลุ่มของแอทริบิวที่สามารถระบุได้ถึงแถวข้อมูลหนึ่งๆได้ โดยที่จะเป็นกลุ่มของแอทริบิวที่ไม่มีแอทริบิวที่ไม่จำเป็น แต่อย่างไรก็ตาม candidate key หนึ่งๆอาจไม่ถูกเลือกเป็น primary key ด้วยเหตุผลบางประการก็เป็นได้)

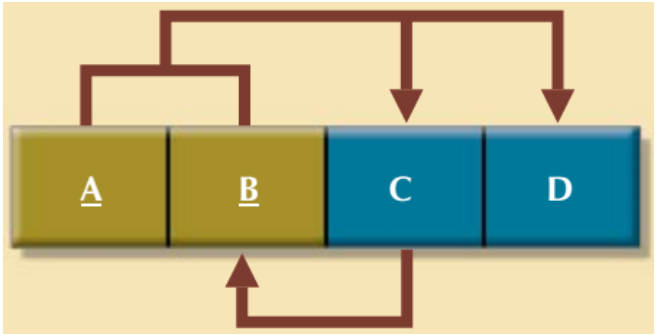
ถ้าตารางข้อมูลหนึ่งๆมีเพียง candidate key เดียว เราสามารถสรุปได้ว่าตารางข้อมูลที่อยู่ในรูปแบบ 3NF จะมีความเท่าเทียมกับตารางข้อมูลที่อยู่ในรูปแบบ BCNF แต่ถ้าตารางข้อมูลหนึ่งๆมีหลาย candidate key จะทำให้ตารางข้อมูลที่อยู่ในรูปแบบ 3NF จะมีความแตกต่างจากตารางข้อมูลที่อยู่ในรูปแบบ BCNF ที่ซึ่งผู้ออกแบบฐานข้อมูลส่วนใหญ่มักจะมองว่าตารางข้อมูลที่อยู่ในรูปแบบ BCNF จะเป็นกรณีพิเศษของตารางข้อมูลที่อยู่ในรูปแบบ 3NF ที่ซึ่งตารางข้อมูลใดก็ตามที่อยู่ในรูปแบบ BCNF จะมีคุณสมบัติเป็นตารางข้อมูลที่อยู่ในรูปแบบ 3NF ด้วยเช่นกัน แต่ตารางข้อมูลที่อยู่ในรูปแบบ 3NF จะไม่เป็นตารางข้อมูลตามที่อยู่ในรูปแบบ BCNF ถ้าตารางข้อมูลที่อยู่ในรูปแบบ 3NF มีแอทริบิวที่ไม่ได้เป็นส่วนประกอบของ primary key ถูกพึ่งพาอาศัยโดยแอทริบิวที่เป็นส่วนประกอบของ primary key ตัวอย่างเช่น ตารางข้อมูลในรูป 6.12 ที่มี 2 candidate key คือ (A+B) และ (A+C) และมี functional dependency ดังต่อไปนี้

A, B $\longrightarrow$ C, D

A, C $\longrightarrow$ B, D

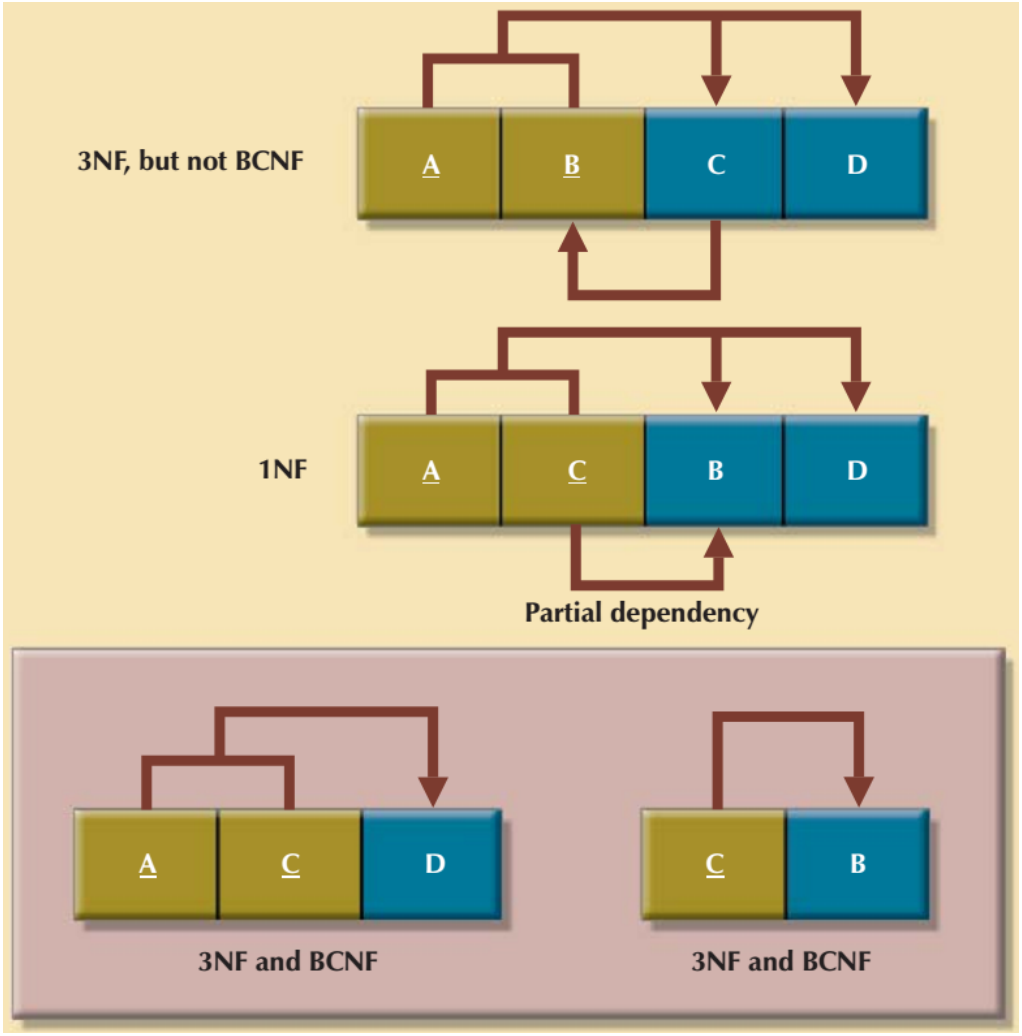
C $\longrightarrow$ B

จากการสังเกตุ functional dependency ทั้ง 3 ข้างต้น เราทราบได้ว่าตารางข้อมูลจะไม่มี partial dependency และ transitive dependency เลย แต่ยังคงมี C $\longrightarrow$ B ที่ซึ่งแอทริบิว C ที่ซึ่งไม่ได้เป็นส่วนประกอบของ primary key ถูกพึ่งพาอาศัยโดยแอทริบิว B ที่เป็นส่วนประกอบของ primary key



รูปที่ 6.12 ตัวอย่างตารางข้อมูลที่อยู่ในรูปแบบ 3NF แต่ไม่อยู่ในรูปแบบ BCNF

ในการที่จะปรับเปลี่ยนตารางข้อมูลจากรูปแบบ 3NF ในรูป 6.12 ให้กลายเป็นรูปแบบ BCNF จะเริ่มจากการปรับเปลี่ยน primary key จาก (A+B) ให้กลายเป็น (A+C) ที่ซึ่งจะทำให้ตารางข้อมูลกลายเป็นรูปแบบ 1NF เนื่องจากตารางข้อมูลมี partial dependency ที่เกิดจากแอทริบิว C ที่เป็นส่วนประกอบของ primary ถูกพึ่งพาอาศัยโดยแอทริบิว B ที่ไม่เป็นส่วนประกอบของ primary key จากนั้นเราจึงต้องทำการปรับเปลี่ยนตารางข้อมูลให้อยู่ในรูปแบบ 2NF และ 3NF ตามลำดับ (แสดงได้ดังรูปที่ 6.13)



รูปที่ 6.13 ตัวอย่างการปรับเปลี่ยตารางข้อมูลจากรูปแบบ 3NF ให้กลายเป็นรูปแบบ BCNF

ในการที่จะเข้าใจเกี่ยวกับการปรับเปลี่ยนตารางข้อมูลให้อยู่ในรูปแบบ BCNF ลองพิจารณาตารางข้อมูลในรูป 6.14 ที่ซึ่งจะเป็นตารางข้อมูลการลงทะเบียนเรียนของนิสิตที่มีรายละเอียดดังนี้

- แอทริบิว CLASS_CODE จะระบุถึงชั้นเรียนหนึ่งๆที่ไม่ซ้ำกัน
- นิสิตคนหนึ่งๆสามารถลงทะเบียนเรียนได้หลายชั้นเรียน
- อาจารย์คนหนึ่งๆ (STAFF) สามารถสอนได้หลายชั้นเรียน แต่สำหรับชั้นเรียนหนึ่งๆจะถูกสอนโดยอาจารย์คนหนึ่งๆเท่านั้น

STU_ID	STAFF_ID	CLASS_CODE	ENROLL_GRADE
125	25	21334	A
125	20	32456	C
135	20	28458	B
144	25	27563	C
144	20	32456	B

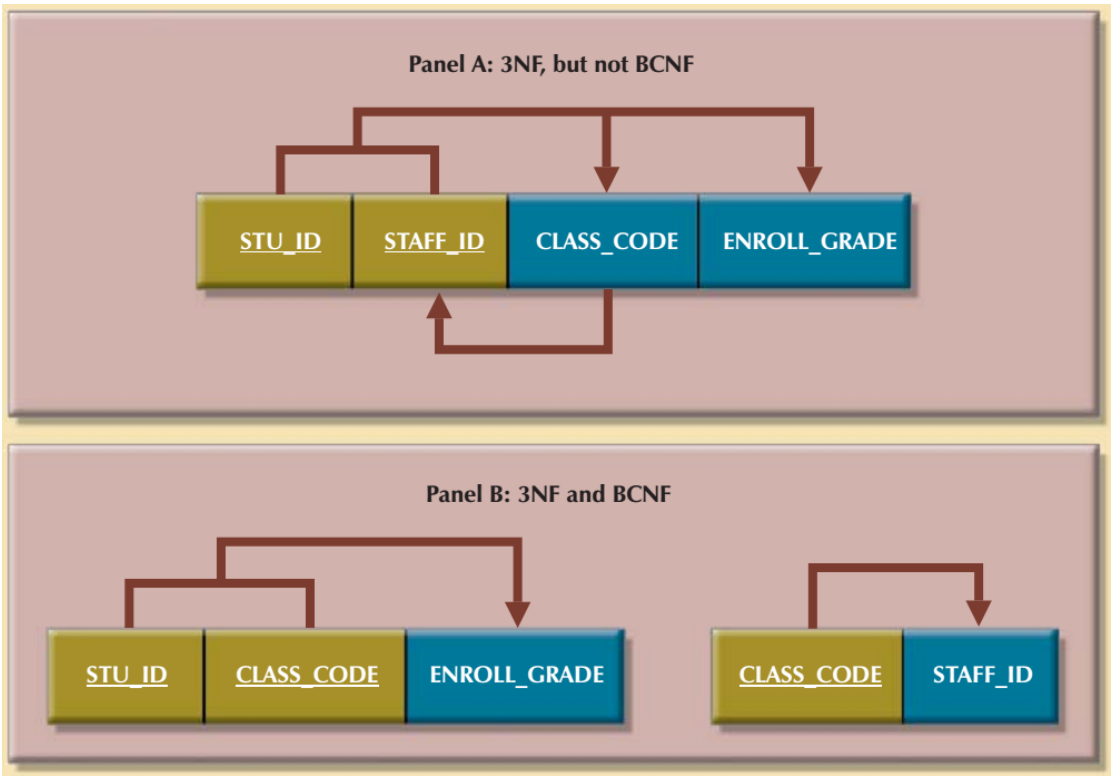
รูปที่ 6.14 ตัวอย่างตารางข้อมูลการลงทะเบียนที่ต้องการการปรับเปลี่ยนให้อยู่ในรูปแบบ BCNF

จากรายละเอียดข้างต้นเราจะสามารถแสดงแผนผังการขึ้นแก่กันได้ดังรูป 6.15 ที่ซึ่งจะประกอบไปด้วย 2 การขึ้นแก่กันของแอทริบิว คือ

STU_ID, STAFF ⟶ CLASS_CODE, ENROLL_GRADE

CLASS_CODE ⟶ STAFF_ID

จากการขึ้นแก่กันของแอทริบิวทั้งสอง เราจะสังเกตุได้ว่าตารางข้อมูลจะอยู่ในรูปแบบ 3NF แต่ไม่อยู่ในรูปแบบ BCNF ด้วยเหตุนี้ เราจึงควรที่จะทำการปรับเปลี่ยนตารางข้อมูลให้อยู่ในรูปแบบ BCNF ด้วยการปรับเปลี่ยน primary key แล้วทำการปรับเปลี่ยนตารางข้อมูลให้อยู่ในรูป 3NF อีกครั้งหนึ่ง โดยหลังจากทำการปรับเปลี่ยนตารางข้อมูลแล้วจะทำให้โครงสร้างของตารางข้อมูลเป็นดังรูป 6.15



รูปที่ 6.15 ตัวอย่างการปรับเปลี่ยนตารางข้อมูลให้อยู่ในรูปแบบ BCNF

6.5.2 นอร์มัลฟอร์มระดับที่ 4 (4NF)

ในการออกแบบฐานข้อมูลครั้งหนึ่งๆ เราอาจถูกร้องขอให้ทำการปรับเปลี่ยนการจัดเก็บข้อมูลจากการประยุกต์ใช้สเปรดชีทให้กลายเป็นรูปแบบของฐานข้อมูลที่ซึ่งจะมีข้อมูลที่มีลักษณะเป็น multivalued attribute ปรากฏอยู่ ตัวอย่างเช่น จากตัวอย่างบริษัทข้างต้น อาจมีพนักงานคนหนึ่งๆถูกกำหนดให้ทำงานในหลายๆโปรเจคและอาจจะเกี่ยวข้องกับทำงานให้กับบริษัทอื่นๆในเครือเดียวกัน สมมติว่า พนักงานรหัส 10123 อาสาสมัครที่จะช่วยเหลืองานในบริษัท Red Cross (RC) และ United Way (UW) และถูกกำหนดทำงานใน 3 โปรเจคของบริษัทที่ตนเองสังกัดอยู่ คือ โปรเจค 1, 3, และ 4 ตามลำดับ เราจะสามารถจัดเก็บข้อมูลสเปรดชีทได้หลายวิธีดังแสดงในรูป 6.16

Table name: VOLUNTEER_V1

EMP_NUM	ORG_CODE	ASSIGN_NUM
10123	RC	1
10123	UW	3
10123		4

Table name: VOLUNTEER_V2

EMP_NUM	ORG_CODE	ASSIGN_NUM
10123	RC	
10123	UW	
10123		1
10123		3
10123		4

Table name: VOLUNTEER_V3

EMP_NUM	ORG_CODE	ASSIGN_NUM
10123	RC	1
10123	RC	3
10123	UW	4

รูปที่ 6.16 ตัวอย่างการจัดเก็บข้อมูลที่มีลักษณะเป็น multivalued

จากรูป 6.16 เราจะเห็นว่าแอทริบิว ORG_CODE และ ASSIGN_NUM จะมีค่าที่แตกต่างกันที่จะทำให้เกิด “**multivalued dependency**” ที่ซึ่งจะเป็นเหตุการณ์ที่คีย์หนึ่งๆถูกพึ่งพาอาศัยโดย 2 แอทริบิวที่มีลักษณะเป็น multivalued attribute และแอทริบิวทั้งสองจะไม่ขึ้นกัน ดังนั้น เราจำเป็นที่จะต้องกำจัด multivalued dependency โดยการสร้างตารางข้อมูลใหม่สำหรับแอทริบิวที่มีลักษณะเป็น multivalued ที่ซึ่งจะสามารถดำเนินการได้โดยการสร้างตาราง ASSIGNMENT และ SERVICE_V1 ดังแสดงในรูป 6.17 ที่จะทำให้ตารางข้อมูลเหล่านั้นอยู่ในรูปแบบ 4NF

จากรูป 6.17 เราจะสามารถสรุปได้ว่าตารางข้อมูลที่อยู่ในรูปแบบ 4NF จะเป็นตารางข้อมูลที่อยู่ในรูปแบบ 3NF ที่ไม่มี multivalued dependency ปรากฏอยู่เลย

Table name: PROJECT

PROJ_CODE	PROJ_NAME	PROJ_BUDGET
1	BeThere	1023245.00
2	BlueMoon	20198608.00
3	GreenThumb	3234456.00
4	GoFast	5674000.00
5	GoSlow	1002500.00

Table name: EMPLOYEE

EMP_NUM	EMP_LNAME
10121	Rogers
10122	O'Leery
10123	Panera
10124	Johnson

Table name: ASSIGNMENT

ASSIGN_NUM	EMP_NUM	PROJ_CODE
1	10123	1
2	10121	2
3	10123	3
4	10123	4
5	10121	1
6	10124	2
7	10124	3
8	10124	5

Table name: ORGANIZATION

ORG_CODE	ORG_NAME
RC	Red Cross
UW	United Way
WF	Wildlife Fund

Table name: SERVICE_V1

EMP_NUM	ORG_CODE
10123	RC
10123	UW
10123	WF

รูปที่ 6.17 ตัวอย่างตารางข้อมูลที่อยู่ในรูปแบบ 4NF

6.6 การลดทอนระดับของนอร์มัลฟอร์ม

ในการสร้างฐานข้อมูลจริงจะต้องการให้โครงสร้างการจัดเก็บข้อมูลของทุกตารางข้อมูลอยู่ในรูปแบบ 3NF เป็นอย่างน้อยที่ซึ่งจะสามารถลดทอนความซ้ำซ้อนและความผิดปกติของข้อมูลได้แต่ก็นำมาซึ่งเวลาในการเข้าถึงข้อมูลที่เพิ่มขึ้น แต่อย่างไรก็ตาม ในบางสถานการณ์เราอาจจำเป็นต้องลดทอนระดับของนอร์มัลฟอร์มลง (เรียกวิธีการนี้ว่า “denormalization”) เพื่อลดเวลาในการเข้าถึงข้อมูล แต่การดำเนินการดังกล่าวก็แลกมาด้วยความซ้ำซ้อนและความผิดปกติของข้อมูลที่เพิ่มขึ้น ดังนั้น ในการออกแบบฐานข้อมูลเราควรที่ต้องคำนึงถึงจุดสมดุลระหว่างความซ้ำซ้อนและความผิดปกติของข้อมูลกับเวลาที่ใช้ในการเข้าถึงข้อมูล ตัวอย่างเช่น การจัดเก็บข้อมูลรหัสไปรษณีย์ (ZIP_CODE) และชื่อเมือง (CITY) ซึ่งเป็นข้อมูลที่อยู่ของลูกค้าในตารางลูกค้า (CUSTOMER) โดยจากการพิจารณาเราจะเห็นว่าแอทริบิว CITY จะขึ้นกับแอทริบิว ZIP_CODE และจากขั้นตอนวิธีในการปรับเปลี่ยนตารางข้อมูลให้อยู่ในรูปแบบ 3NF เราจะต้องทำการสร้างตารางใหม่เพื่อลดทอน transitive dependency ที่ซึ่งจะทำให้เราได้ตารางใหม่เป็น

ZIP (ZIP_CODE, CITY)

จากการดำเนินการข้างต้น เราจะสังเกตุได้ว่าเมื่อทำการแยกข้อมูล ZIP_CODE และ CITY มาจัดเก็บในตารางข้อมูลใหม่ จะทำให้เวลาในการเข้าถึงข้อมูลที่อยู่ของลูกค้าเพิ่มขึ้น เนื่องจากเราต้องทำการค้นหาข้อมูลลูกค้าคนหนึ่งๆในตาราง CUSTOMER เพื่อที่จะได้รหัสไปรษณีย์ จากนั้นจึงนำรหัสไปรษณีย์ที่ได้มาทำการค้นหาชื่อเมืองอีกครั้งหนึ่ง จากดำเนินการดังกล่าวถ้าเรายอมให้มีความซ้ำซ้อนของชื่อเมืองในตารางข้อมูลลูกค้าจะทำให้เราสามารถลดเวลาในการค้นหาที่อยู่ของลูกค้าได้ ด้วยเหตุนี้ เราจึงควรที่จะพิจารณาถึงการ

ลดทอนระดับของนอร์มัลฟอร์มของตาราง CUSTOMER เพื่อหาจุดสมดุลระหว่างปริมาณความซ้ำซ้อนของข้อมูลที่จะปรากฏกับเวลาที่ใช้ในการเข้าถึงข้อมูล

ในการที่จะเข้าใจถึงข้อดี-เสียของการลดระดับนอร์มัลฟอร์มของตารางข้อมูล ลองพิจาณาตัวอย่างในรูป 6.18 ที่จะแสดงถึงการลดทอนระดับนอร์ฟอร์มในหลายๆกรณี

CASE	EXAMPLE	RATIONALE AND CONTROLS
Redundant data	Storing ZIP and CITY attributes in the CUSTOMER table when ZIP determines CITY. (See Table 1.4.)	• Avoid extra joint operations. • Program can validate city (drop-down box) based on the zip code.
Derived data	Storing STU_HRS and STU_CLASS (student classification) when STU_HRS determines STU_CLASS. (See Figure 3.29.)	• Avoid extra joint operations. • Program can validate classification (lookup) based on the student hours.
Preaggregated data (also derived data)	Storing the student grade point average (STU_GPA) aggregate value in the STUDENT table when this can be calculated from the ENROLL and COURSE tables. (See Figure 3.29.)	• Avoid extra joint operations. • Program computes the GPA every time a grade is entered or updated. • STU_GPA can be updated only via administrative routine.
Information requirements	Using a temporary denormalized table to hold report data. This is required when creating a tabular report in which the columns represent data that are stored in the table as rows. (See Figure 6.17 and Figure 6.18.)	• Impossible to generate the data required by the report using plain SQL. • No need to maintain table. Temporary table is deleted once report is done. • Processing speed is not an issue.

รูปที่ 6.18 ตัวอย่างการดำเนินการ denormalization

ตารางข้อมูลที่ไม่มีการดำเนินการนอร์มัลไลเซชันจะก่อให้เกิดข้อบกพร่องต่างๆ เช่น การปรับปรุงข้อมูลจะมีประสิทธิภาพน้อยลงเพราะโปรแกรมต้องทำการอ่านข้อมูลและอัพเดทข้อมูลปริมาณมาก และ การสร้างดัชนีจะทำได้ยุ่งยากมากขึ้นเนื่องจากตารางข้อมูลมีแอทริบิวเป็นจำนวนมาก เป็นต้น ดังนั้น ในการที่จะตัดสินใจดำเนินการ denormalization เราควรที่จะต้องทราบอย่างแน่ชัดว่าข้อมูลที่ถูกจัดเก็บจะมีความซ้ำซ้อนมากขึ้น และเราควรที่จะตอบคำถามให้ได้ว่าเพราะเหตุใดเราจึงต้องทำการการ denormalization ที่ซึ่งมีข้อดีกว่าการดำเนินการนอร์มัลไลเซชันอย่างไร

6.7 ปัจจัยที่ต้องพิจารณาในการสร้างแบบจำลองข้อมูล

จากเนื้อหาในบทที่ 4 - 6 เราได้ศึกษาเกี่ยวกับการสร้างแบบจำลองที่ซึ่งประกอบไปด้วยรายละเอียดต่างๆมากมาย แต่อย่างไรก็ตาม ถ้าเรามีต้นแบบหรือลิสต์ในการตรวจสอบแบบจำลองข้อมูลที่ออกแบบไว้ จะทำให้เราแน่ใจได้ว่าแบบจำลองข้อมูลที่เราออกแบบจะเป็นแบบจำลองที่ดี โดยลิสต์ของปัจจัยต่างๆที่ต้องพิจารณาจะสามารถแสดงได้ดังรูป 6.19

DATA-MODELING CHECKLIST

BUSINESS RULES

- Properly document and verify all business rules with the end users.
- Ensure that all business rules are written precisely, clearly, and simply. The business rules must help identify entities, attributes, relationships, and constraints.
- Identify the source of all business rules, and ensure that each business rule is justified, dated, and signed off by an approving authority.

DATA MODELING

Naming Conventions: All names should be limited in length (database-dependent size).

- Entity Names:
  - Should be nouns that are familiar to business and should be short and meaningful
  - Should document abbreviations, synonyms, and aliases for each entity
  - Should be unique within the model
  - For composite entities, may include a combination of abbreviated names of the entities linked through the composite entity
- Attribute Names:
  - Should be unique within the entity
  - Should use the entity abbreviation as a prefix
  - Should be descriptive of the characteristic
  - Should use suffixes such as _ID, _NUM, or _CODE for the PK attribute
  - Should not be a reserved word
  - Should not contain spaces or special characters such as @, !, or &
- Relationship Names:
  - Should be active or passive verbs that clearly indicate the nature of the relationship

Entities:

- Each entity should represent a single subject.
- Each entity should represent a set of distinguishable entity instances.
- All entities should be in 3NF or higher. Any entities below 3NF should be justified.
- The granularity of the entity instance should be clearly defined.
- The PK should be clearly defined and support the selected data granularity.

Attributes:

- Should be simple and single-valued (atomic data)
- Should document default values, constraints, synonyms, and aliases
- Derived attributes should be clearly identified and include source(s)
- Should not be redundant unless this is required for transaction accuracy, performance, or maintaining a history
- Nonkey attributes must be fully dependent on the PK attribute

Relationships:

- Should clearly identify relationship participants
- Should clearly define participation, connectivity, and document cardinality

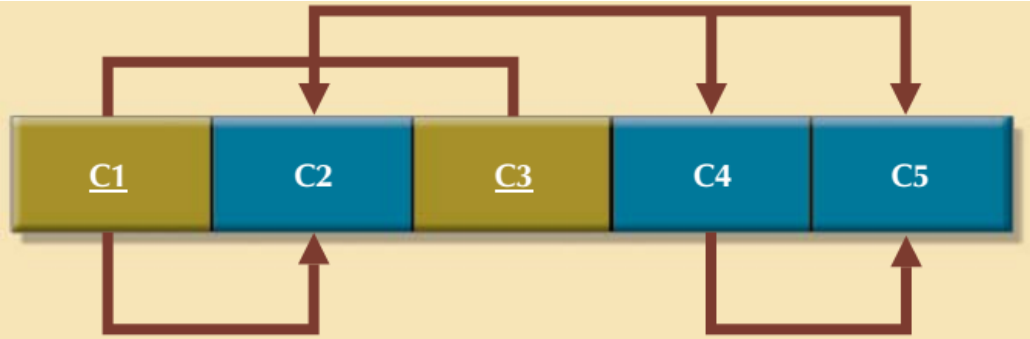
ER Model:

- Should be validated against expected processes: inserts, updates, and deletes
- Should evaluate where, when, and how to maintain a history
- Should not contain redundant relationships except as required (see attributes)
- Should minimize data redundancy to ensure single-place updates
- Should conform to the minimal data rule: "All that is needed is there, and all that is there is needed."

รูปที่ 6.19 ลิสต์ของปัจจัยที่ต้องพิจารณาและตรวจสอบก่อนการออกแบบแบบจำลองข้อมูลเสร็จสิ้น

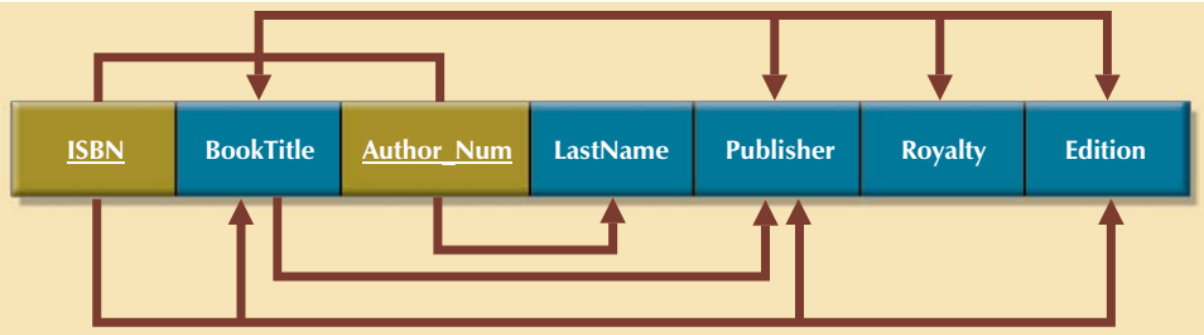
คำถามท้ายบท

1. นอร์มัลไลเซชันคืออะไร
2. ตารางข้อมูลที่อยู่ในรูปแบบ 1NF จะมีลักษณะอย่างไร? จงยกตัวอย่างประกอบ
3. ตารางข้อมูลที่อยู่ในรูปแบบ 2NF จะมีลักษณะอย่างไร? จงยกตัวอย่างประกอบ
4. ตารางข้อมูลที่อยู่ในรูปแบบ 3NF จะมีลักษณะอย่างไร? จงยกตัวอย่างประกอบ
5. ตารางข้อมูลที่อยู่ในรูปแบบ 4NF จะมีลักษณะอย่างไร? จงยกตัวอย่างประกอบ
6. จากรูปด้านล่าง จงตอบคำถามต่อไปนี้



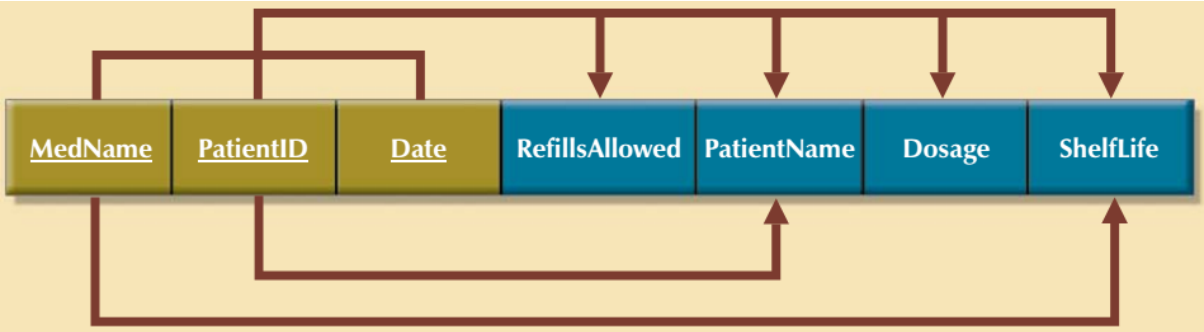
   a. จงระบุถึง functional dependency ที่ปรากฏใน dependency diagram ข้างต้น
   b. จงแสดงวิธีการปรับเปลี่ยนตารางข้อมูลข้างต้นให้อยู่ในรูปของ 2NF
   c. จงแสดงวิธีการปรับเปลี่ยนตารางข้อมูลข้างต้นให้อยู่ในรูปของ 3NF

7. จากรูปด้านล่างจะแสดงถึงโครงสร้างการจัดเก็บข้อมูลหนังสือ



   a. จงแสดงขั้นตอนวิธีในการปรับเปลี่ยนตารางข้อมูลข้างต้นให้อยู่ในรูป 2NF และแสดง dependency diagram ของทุกตารางข้อมูล
   b. จงแสดงขั้นตอนวิธีในการปรับเปลี่ยนตารางข้อมูลข้างต้นให้อยู่ในรูป 3NF และแสดง dependency diagram ของทุกตารางข้อมูล

8. จากรูปด้านล่างจะแสดงถึงข้อมูลใบสั่งยาของผู้ป่วยคนหนึ่งๆที่ซึ่งจะสามารถได้รับใบสั่งยาได้หลายใบ และแต่ละใบสั่งยาจะมียาได้หลายชนิด จงทำการปรับเปลี่ยนตารางให้อยู่ในรูปแบบ 3NF แล้วทำการสร้าง dependency diagram เพื่อแสดงถึง dependency ต่างๆ



9. Partial dependency และ transitive dependency คืออะไร? จงยกตัวอย่างประกอบ
10. Surrogate key คืออะไร? เราควรใช้ surrogate key เมื่อใด?