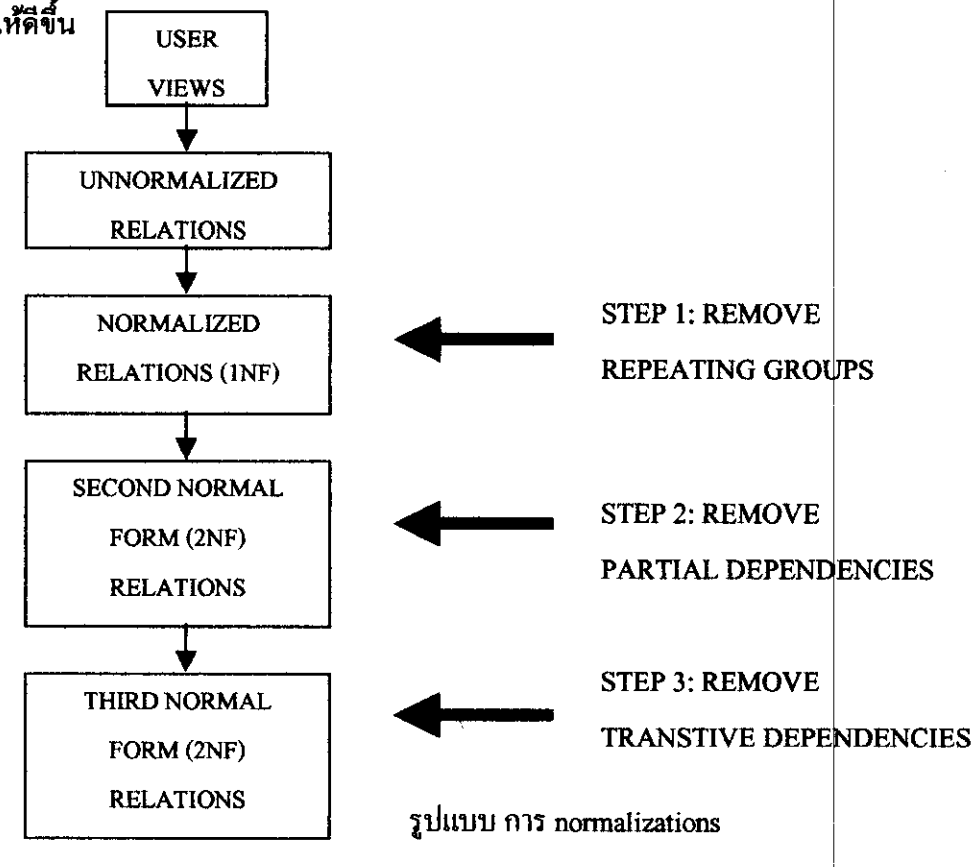
บทที่ 8
การทำให้เป็นบรรทัดฐานตารางฐานข้อมูล
(Normalization of Database tables)

ฐานข้อมูลที่ดีจะต้องมาจากตารางที่มีโครงสร้างที่ดี ซึ่งในเรื่องของการทำ Normalization นี้จะทำให้ได้เรียนรู้ถึงการพัฒนาและการออกแบบโครงสร้างตารางที่ดีในการควบคุมการเกิดความซ้ำซ้อนของข้อมูล (Data Redundancy) เพื่อหลีกเลี่ยงการเกิดเหตุการณ์ที่ข้อมูลเดียวกันแต่อยู่ต่างแฟ้มกันกลับมีค่าของข้อมูลไม่เหมือนกัน (Data Anomalies) โดยจะต้องใช้การทำ Normalization ในการทำงานเพื่อให้ได้โครงสร้างของตารางที่ดีตามความต้องการ

เพื่อให้รู้จักคุณลักษณะของโครงสร้างของตารางที่ดี จะเริ่มจากการพิจารณา โครงสร้างของตารางที่ไม่ดีและปัญหาที่จะเกิดขึ้นหลักจากสร้างตารางที่ไม่ดีแล้ว และจะแสดงให้เห็นว่าจะมีวิธีการแก้ไขโครงสร้างของตารางที่ไม่ดีได้อย่างไร โดยจะแบ่งการทำงานออกเป็น 2 ส่วนคือ การออกแบบโครงสร้างตารางที่ดี และการปรับปรุงโครงสร้างของตารางที่มีอยู่แล้วแต่เป็นตารางที่มีโครงสร้างที่ไม่ดีให้ดีขึ้น



รูปแบบ การ normalizations

FORM (2NF) RELATIONS
TRANSTIVE DEPENDENCIES
รูปแบบ การ normalizations

8.1 ตารางฐานข้อมูล และการทำให้เป็นบรรทัดฐาน (Database Tables and Normalization)

ปัจจุบันนี้ได้มีโปรแกรมที่จะจัดการกับงานด้าน Relation Database เกิดขึ้นมากมายแต่ก็ยังไม่ดีพอที่จะทำการลดปัญหาความซ้ำซ้อนของข้อมูล (Data Redundancy) ได้

การทำ Normalization จะสามารถจัดการเปลี่ยนแปลงโครงสร้างของฐานข้อมูลที่ไม่ดีให้เป็นฐานข้อมูลที่ดีได้ และสามารถสร้างฐานข้อมูลที่ดีได้ด้วย โดยใช้การทำ Normalization กำหนด Attributes ต่างๆ ใน Entities การทำ Normalization จะช่วยลดปัญหาการซ้ำซ้อนของข้อมูล (Data Redundancy) และขจัดปัญหา Anomalies ได้แต่ยังไม่ขจัดปัญหาความซ้ำซ้อนของข้อมูลให้หมดไปได้ เพราะจะต้องใช้ข้อมูลที่ซ้ำซ้อนเหล่านี้ในการเชื่อมตารางฐานข้อมูลต่างๆ ให้สามารถใช้ข้อมูลร่วมกันได้

การทำงานของ Normalization จะแบ่งการทำงานออกเป็นระดับโดย 3 ระดับแรกได้แก่ First Normal Form :1NF (รูปแบบบรรทัดฐานที่หนึ่ง), Second Normal Form :2NF (รูปแบบบรรทัดฐานที่สอง) และ Third Normal 3NF (รูปแบบบรรทัดฐานที่สาม) ถ้าเรามองที่โครงสร้างของตารางแล้วจะพบว่าตารางที่มีคุณสมบัติ 2NF จะลดปัญหาความซ้ำซ้อนของข้อมูล (Data Redundancy) ได้ดีกว่าตารางที่มีคุณสมบัติ 1NF และตารางที่มี คุณสมบัติ 3NF ก็จะลดปัญหาความซ้ำซ้อนของข้อมูลได้ดีกว่าตารางที่มีคุณสมบัติ 2NF โดยทั่วไปการออกแบบฐานข้อมูลในระบบธุรกิจใช้ตารางที่มีคุณสมบัติ Normal Form ที่ 3 (3NF) ก็เพียงพอแล้ว แต่ในความเป็นจริงแล้วผู้ที่เขียน Application Program อาจจะมีความต้องการ การ Normalization ในระดับที่สูงขึ้นไปอีกเรียกว่า Higher-Level Normalization (บรรทัดฐานระดับสูง)

การทำ Normalization มีความสำคัญมากในการออกแบบฐานข้อมูล แต่อย่าเข้าใจว่าการใช้ Normalization ในระดับสูงๆ จะดีที่สุด เพราะยิ่งใช้คุณสมบัติ Normal Form ในระดับสูงขึ้นไปก็ยิ่งมีการแยกข้อมูลออกไปเป็นตารางย่อยๆ มากขึ้น ทำให้การค้นหาข้อมูลที่ต้องการช้ามาก เพราะต้องทำการเชื่อมแต่ละตารางเข้าด้วยกันเพื่อให้ได้ข้อมูลที่ต้องการ ในทางปฏิบัติผู้ใช้ส่วนใหญ่ต้องการการค้นหาข้อมูลที่รวดเร็วและมีประสิทธิภาพ ดังนั้นจึงต้องมีการใช้ Denormalize (ไม่เป็นบรรทัดฐาน) ซึ่งคือการใช้ตารางที่มีคุณสมบัติ Normal Form ที่อยู่ในระดับต่ำลงมา เช่น ถ้าตารางฐานข้อมูลมีคุณสมบัติ 3NF แต่มีการแตกเป็นตารางย่อยๆ มากมาย ซึ่งทำให้การค้นหาข้อมูลช้ามาก ก็ทำการลดระดับคุณสมบัติ Normal Form ของตารางลงให้มีคุณสมบัติเป็นแค่ 2NF ก็พอเพื่อให้มี

การทำงานที่เร็วขึ้น แต่การทำ Denormalize นั้นมีข้อเสียตรงที่ทำให้สิ้นเปลืองเนื้อที่จัดเก็บข้อมูลมาก เพราะจะทำให้เกิดปัญหาความซ้ำซ้อนของข้อมูล (Data Redundancy)

8.1.1 (The Need For Normalization)

เพื่อให้เห็นการทำ Normalization อย่างชัดเจนจะยกตัวอย่างการทำงานของระบบธุรกิจ โดยจะสร้างฐานข้อมูลของบริษัทคอมพิวเตอร์แห่งหนึ่ง ได้มีการรับงานต่างๆ พร้อมกันหลายๆ งาน ซึ่งในแต่ละงานจะเก็บรหัสและชื่อของงานและข้อมูลของพนักงานที่ทำงานนี้โดยข้อมูลของพนักงานจะจัดเก็บ รหัส, ชื่อ และประเภทของงานที่ทำ โดยบริษัทนี้จ่ายเงินให้พนักงานตามชั่วโมงการทำงาน โดยในแต่ละประเภทของงานก็จะได้รับเงินในอัตราที่ต่างกัน ดังข้อมูลในตารางที่ 8.1

Proj-Num	Project-Name	Employee-Number	Employee-Name	Job-Class	Chg/Hour	Hour Billed	Total Charge
15	Evergreen	103	June E. Arbough	Elect. Engineer	$84.50	23.8	$2,0211.10
		101	John G. News	Database Designer	$105.00	19.4	$2,037.00
		105	Alice K. Johnson*	Database Designer	$105.00	35.7	$3,748.50
		106	William Smith Field	Programmer	$35.75	12.6	$450.45
		102	David H. Senior	System Analyst	$96.75	23.8	$2,302.65
				Subtotals			$10,549.70
18	Amber wave	114	Annelise Jones	Application Designer	$48.10	24.6	$1,183.26
		118	Jones J. Frommer	General Support	$18.36	45.3	$831.71
		104	Anne K. Ranoras*	System Support	$96.75	32.4	$3,134.70
		112	Dedene M. Smithson	DSS Analyst	$45.95	44.0	$2,021.80
				Subtotals			$7,171.47
22	Rolling Tide	105	Alice K. Johnson	Database Designer	$105.00	64.7	$6,793.50
		104	Anne K. Ramoras	System Analyst	$96.75	48.4	$4,682.70
		113	Delbert K. Joenbrood*	Application Designer	$48.10	23.6	$1,135.16
		111	Geoff B. Wabash	Clerical Support	$26.87	22.0	$591.14
		106	William Smithfield	Programmer	$35.75	12.8	$457.60
				Subtotals			$13,660.10
25	Startfight	107	Maria D. Alonzo	Programmer	$35.75	24.6	$879.45
		115	Travis B. Bawahgi	System Analyst	$96.75	45.8	$4,431.15

Proj-Num	Project-Name	Employee-Number	Employee-Name	Job-Class	Chg/Hour	Hour Billed	Total Charge
		101	John G. News*	Database Designer	$105.00	56.3	$5,911.50
		114	Anelise Jones	Application Designer	$48.10	33.1	$1,592.11
		108	Ralph B. Washington	Systems Analyst	$96.75	23.6	$2,283.30
		118	Jomes J. Frammer	General Support	$18.36	30.5	$559.98
		112	Derlene M. Smithson	DSS Analyst	$45.95	41.4	$1,902.33
				Subtotals			$17,559.82
				Total			$48,941.09

หมายเหตุ * หมายถึงหัวหน้างาน

ตาราง A Sample Report Layout

วิธีการที่จะสร้างตารางฐานข้อมูลที่ตรงตามความต้องการและง่ายที่สุดทำได้ดังตารางฐานข้อมูล 8.1 ดังต่อไปนี้

PROJ_NUM	PROJ_NAME	EMP_NUM	EMP_NAME	JOB_CLASS	CHG_HOUR	HOUR
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 Smith Field	Programmer	$35.75	12.6
		102	David H. Senior	System Analyst	$96.75	23.8
18	Amber wave	114	Annelise Jones	Application Designer	$48.10	24.6
		118	Jones J. Frommer	General Support	$18.36	45.3
		104	Anne K. Ranoras*	System Support	$96.75	32.4
		112	Dedene 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	System Analyst	$96.75	48.4
		113	Delbert K. Joenbrood*	Application Designer	$48.10	23.6
		111	Geoff B. Wabash	Clerical Support	$26.87	22.0
		106	William Smithfield	Programmer	$35.75	12.8
25	Startfight	107	Maria D. Alonzo	Programmer	$35.75	24.6
		115	Travis B. Bawahgi	System Analyst	$96.75	45.8

PROJ_NUM	PROJ_NAME	EMP_NUM	EMP_NAME	JOB_CLASS	CHG_HOUR	HOUR
		101	John G. News*	Database Designer	$105.00	56.3
		114	Anelise Jones	Application Designer	$48.10	33.1
		108	Ralph B. Washington	Systems Analyst	$96.75	23.6
		118	Jornes J. Frammer	General Support	$18.36	30.5
		112	Derlene M. Smithson	DSS Analyst	$45.95	41.4

ตารางฐานข้อมูล 8.1 A Table Whose Structure Matches The Report Format

จากตารางฐานข้อมูล 8.1 จะเห็นว่าไม่มี Total Charge เมื่อเทียบกับรายงานที่ต้องการแต่ว่า Total Charge สามารถคำนวณได้จาก Hours Billed คูณ Charge per Hour แต่จากตารางฐานข้อมูลที่ 8.1 ที่สร้างขึ้นมานี้ เราจะพบข้อผิดพลาดต่างๆ ดังต่อไปนี้

1. PROJ_NUM ทำหน้าที่เป็น Primary Key หรืออย่างน้อยที่สุดก็ต้องเป็นส่วนประกอบของ Primary Key แต่ว่าสามารถมีค่าเป็น Null ได้

2. อาจจะทำให้เกิดความไม่สอดคล้องกันของข้อมูล (Data Inconsistencies) คือในช่อง JOB_CLASS ในงานประเภท "Elec. Engineer" ผู้ใช้อาจจะใส่เป็น "Elect. Eng" หรือ "EE" ก็ได้ซึ่งจะเป็นผลให้เกิดความผิดพลาดของข้อมูลได้ ดังจะแยกความผิดพลาดออกได้ดังนี้

a. ความผิดพลาดในการปรับปรุงข้อมูล (Update Anomalies) คือถ้าจะเปลี่ยนประเภทของงาน (JOB_CLASS) ของพนักงานหมายเลข 105 ซึ่งมีข้อมูลปรากฏอยู่ในหลาย Entities ถ้าแก้ข้อมูลไม่ครบทุก Entities ก็จะเกิดความสับสนว่า ที่จริงแล้วพนักงานหมายเลข 105 ทำงานประเภทไหนกันแน่

b. ความผิดพลาดในการเพิ่มข้อมูล (Addition Anomalies) คือถ้ามีพนักงานใหม่เข้ามาแต่ยังไม่ได้ระบุว่าให้ทำงานอะไรและจะทำงานประเภทไหน ข้อมูลของพนักงานคนนี้ก็จะยังไม่สามารถเก็บไว้ในฐานข้อมูลได้

c. ความผิดพลาดในการลบข้อมูล (Deletion Anomalies) คือถ้าพนักงานรหัส 103 ลาออกเราต้องลบข้อมูลทุก Entities ที่มี EMP_NUM เท่ากับ 103 ถ้าลบออกไม่หมดก็ จะทำให้เปลืองเนื้อที่จัดเก็บแต่ถ้าลบออก จากตารางฐานข้อมูลที่ 8.1 จะเห็นว่าจะเป็นการลบ Entities ที่มี

PROJ_NUM = 15 และ PROJ_NAME = Evergreen ออกไป ทำให้ Entities ของพนักงานอื่นๆ ที่ทำงานในงานๆ นี้ไม่สามารถระบุถึงชื่องานที่รับผิดชอบได้

8.1.2 Conversion To First Normal Form

ในการสร้างตารางข้อมูลกำหนดว่า Key ทุกตัวต้องไม่เป็นค่า Null แต่จากตารางฐานข้อมูลที่ 8.1 ไม่เป็นไปตามข้อกำหนด และจะพบว่าเกิด Repeating Groups คือข้อมูลใน PROJ_NUM 1 แถวอ้างอิงถึงข้อมูลหลายแถว หรือก็คือใช้ข้อมูลเพียงแถวเดียวซ้ำๆ กันเพื่ออ้างอิงถึงข้อมูลในแถวอื่นๆ ได้หลายแถว ดังตัวอย่างตารางฐานข้อมูลที่ 8.2

PROJ_NUM	PROJ_NAME	EMP_NUM	EMP_NAME	JOB_CLASS	CHG_HOUR	HOUR
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 Smith Field	Programmer	$35.75	12.6
		102	David H. Senior	System Analyst	$96.75	23.8

ตารางฐานข้อมูล 8.2 The Evergreen Data

เนื่องจากตารางจะต้องไม่มี Repeating Groups ถ้าหากว่าตารางมี Repeating Groups ปรากฏอยู่จะแก้ไขโดยการกำหนดแถวให้เป็นแบบ Single Entity การทำที่ง่ายที่สุดคือการเพิ่มข้อมูลใน Column ของ Primary Key โดยในตัวอย่างตารางฐานข้อมูลที่ 8.3 ที่ได้จากตารางฐานข้อมูลที่ 8.1 แต่มีการเพิ่มข้อมูลเข้าไปเพื่อให้ตารางมีลักษณะเป็น Single Entity ทำให้ตารางฐานข้อมูลที่ 8.3 มีคุณสมบัติ First Normal Form (1NF) โดยการลด Repeating Groups

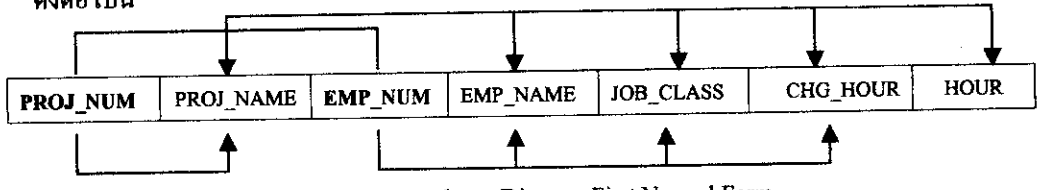
PROJ_NUM	PROJ_NAME	EMP_NUM	EMP_NAME	JOB_CLASS	CHG_HOUR	HOUR
15	Evergreen	103	June E. Arbough	Elect. Engineer	$84.50	23.8
15	Evergreen	101	John G. News	Database Designer	$105.00	19.4
15	Evergreen	105	Alice K. Johnson*	Database Designer	$105.00	35.7
15	Evergreen	106	William Smith Field	Programmer	$35.75	12.6
15	Evergreen	102	David H. Senior	System Analyst	$96.75	23.8
18	Amber wave	114	Annelise Jones	Application Designer	$48.10	24.6
18	Amber wave	118	Jones J. Frommer	General Support	$18.36	45.3
18	Amber wave	104	Anne K. Ranoras*	System Support	$96.75	32.4
18	Amber wave	112	Dedene M. Smithson	DSS Analyst	$45.95	44.0

PROJ_NUM	PROJ_NAME	EMP_NUM	EMP_NAME	JOB_CLASS	CHG_HOUR	HOUR
22	Rolling Tide	105	Alice K. Johnson	Database Designer	$105.00	64.7
22	Rolling Tide	104	Anne K. Ramoras	System Analyst	$96.75	48.4
22	Rolling Tide	113	Delbert K. Joenbrood*	Application 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	Startfight	107	Maria D. Alonzo	Programmer	$35.75	24.6
25	Startfight	115	Travis B. Bawahgi	System Analyst	$96.75	45.8
25	Startfight	101	John G. News*	Database Designer	$105.00	56.3
25	Startfight	114	Anelise Jones	Application Designer	$48.10	33.1
25	Startfight	108	Ralph B. Washington	Systems Analyst	$96.75	23.6
25	Startfight	118	Jornes J. Frammer	General Support	$18.36	30.5
25	Startfight	112	Derlene M. Smithson	DSS Analyst	$45.95	41.4

ตารางฐานข้อมูล 8.3 Data Organization : First Normal Form

จากตารางฐานข้อมูลที่ 8.3 จะสังเกตเห็นว่า PROJ_NUM ไม่ใช่ Primary Key อีกแล้วเพราะ Project Number ไม่ได้ระบุ Entity ใด Entity หนึ่งโดยเฉพาะ เช่น PROJ_NUM = 15 สามารถระบุถึงข้อมูลพนักงานได้ถึง 5 คนแต่กฎของ Primary Key คือค่า Primary Key จะต้องระบุถึง Entity เพียง Entity เดียวเท่านั้น ต้องมีการกำหนดค่า Key ใหม่ โดย Primary Key ใหม่จะได้จากการรวม PROJ_NUM และ EMP_NUM

โดยสามารถแสดงการ Dependencies กันโดยใช้ Dependency Diagram ดังแผนภาพที่ 8.1 ดังต่อไปนี้



แผนภาพ 8.1 Dependency Diagram First Normal Form

จากแผนภาพ 8.1 จะสรุปได้ว่า

1.จะเห็นว่าลูกศรจะลากจาก PROJ_NUM และ EMP_NUM ไปยัง Attribute อื่นๆ ซึ่งแสดงว่า Attribute เหล่านั้น Dependent บน PROJ_NUM และ EMP_NUM ซึ่งเป็น Primary Key ซึ่งแสดงว่าถ้าเรารู้ค่าของ PROJ_NUM และ EMP_NUM ที่แน่นอนเราก็สามารถหาค่า Attribute ที่ต้องการได้จากตารางฐานข้อมูล 5.3 เช่น ถ้าเรารู้ PROJ_NUM = 15 และ EMP_NUM = 103 เพื่อค้นหาข้อมูลของ Attribute PROJ_NAME, EMP_NAME, JOB_CLASS, CHG_HOUR และ HOURS จะได้ผลลัพธ์เท่ากับ Evergreen, June E. Arbough, Elect Engineer, $84.50 และ 23.8

2.เส้นของลูกศรจะเริ่มจาก Primary Key หรือส่วนของ Primary Key เท่านั้น ซึ่งจะเห็นว่า PROJ_NUM ยังระบุไปถึง PROJ_NAME ได้ด้วย ซึ่ง PROJ_NUM นี้เป็นส่วนประกอบของ Primary Key และถ้าเรารู้ค่า EMP_NUM ก็จะสามารถค้นหา EMP_NAME, JOB_CLASS และ CHG_HOUR ได้ และการที่มี Attribute บาง Attribute สามารถถูกระบุได้โดยบางส่วนของ Primary Key เราเรียกว่าเกิด **Partial Dependency** จากแผนภาพที่ 5.1 สามารถเขียนในลักษณะของ Function ได้ดังนี้

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

PROJ_NUM ⟶ PROJ_NAME

EMP_NUM ⟶ EMP_NAME, JOB_CLASS, CHG_HOUR

ถ้ากำหนดให้ตารางฐานข้อมูลนี้ชื่อ Charge สามารถเขียนโครงสร้าง Schema ได้ดังนี้

Charge(PROJ_NAME, PROJ_NUM, PROJ_NAME,EMP_NAME, JOB_CLASS, CHG_HOUR, HOURS)

จะเห็นว่าตาราง Charge นั้นมี Primary Key ที่เกิดจากการร่วม Attribute PROJ_NUM และ EMP_NUM เราเรียกว่า Attribute ที่เป็น Primary Key หรือ Attribute ที่ประกอบรวมกันเป็น Primary Key ว่า **Prime Attribute** หรือ **Key Attribute** ซึ่ง Attribute อื่นๆ นอกเหนือจาก PROJ_NUM และ EMP_NUM ซึ่งเป็น Prime(Key) Attribute นั้นเราเรียกว่า Nonprime Attribute หรือ Nonkey Attribute คือ Attribute ที่ไม่ใช่ส่วนประกอบของ Key

1NF DEFINITION

The term first normal form (1NF) describes the tabular format in which:

1. All the key attributes are defined.
2. There are no repeating groups in the table. In other words, each row/column intersection can contain one and only one value, rather than a set of values.
3. All attributes are dependent on the primary key.

นิยาม 1NF

เงื่อนไขในการมีคุณสมบัติเป็น First Normal Form (1NF) คือ

1. ต้องมีการกำหนด Attribute ที่ทำหน้าที่เป็น Key ทั้งหมด
2. ต้องไม่มี Repeating Groups ในตาราง คือในตำแหน่งที่ Row ตัดกับ Column ค่าที่ได้จะต้องเป็นค่าๆ เดียวไม่ใช่กลุ่มของค่าข้อมูล
3. ทุกๆ Attribute ต้องถูกระบุได้ด้วย Primary Key

ปัญหาที่เกิดขึ้นของตารางที่มีคุณสมบัติ 1NF คือมีบางส่วนของ Key สามารถระบุถึง Attribute ต่างๆ ในตารางได้ (Partial Dependencies) โดยที่ความจริงแล้วในตารางที่ดีไม่ควรมี Partial Dependencies เพราะตารางที่มี Dependencies เช่นนี้ จะทำให้เกิดการซ้ำซ้อนของข้อมูล (Data Redundancies) และความผิดปกติของข้อมูล (Data Anomalies) โดย Data Redundancies จะเกิดในทุกแถวที่มีการซ้ำกันของข้อมูล เช่น ทุกๆ ครั้งที่ใส่ EMP_NUM = 105 ก็จะต้องใส่ EMP_NAME, JOB_CLASS และ CHG_HOUR เหมือนกันทุกๆ ครั้งถ้าใส่ข้อมูลผิดเพียงเล็กน้อยก็จะทำให้เกิด Data Anomalies ทันที ไม่ว่าจะใส่ EMP_NAME ผิด หรือใส่ JOB_CLASS ผิด หรือใส่ CHG_HOUR ผิด อีกทั้ง ชื่อพนักงานที่มีรหัส EMP_NUM = 102 อาจจะใส่ข้อมูลเป็น Dave Senior หรือ D. Senior ก็ได้ และอาจจะทำงานในงานชื่อ Evergreen หรือถ้าพิมพ์ผิดอาจจะกลายเป็นงานชื่อ Evergreen ก็ได้ ซึ่งทำให้เกิด Data Anomalies ซึ่งผิดกฎความมั่นคงและความสอดคล้องกันของข้อมูล (Integrity and Consistency) ใน Relational Database

8.1.3 Conversion to Second Normal Form

ในการออกแบบตารางแบบ Relational Database ทำให้สามารถปรับปรุงโครงสร้างได้ง่าย การเปลี่ยนฐานข้อมูลให้อยู่ในรูปแบบ Second Normal Form (2NF) โดยเปลี่ยนจาก 1NF ไปเป็น 2NF จะเป็นเรื่องที่ทำได้ง่ายมาก โดยเริ่มจาก 1NF ในแผนภาพที่ 8.1 จะต้องทำตามขั้นตอนต่อไปนี้

1. เขียน Attribute ที่เป็นส่วนประกอบของ Key ออกมาบรรทัดละ Attribute และเขียน Key ที่เป็นตัวหลักไว้บรรทัดสุดท้ายจะได้ ดังต่อไปนี้

PROJ_NUM

EMP_NUM

PROJ_NUM EMP_NUM

Attribute ในแต่ละบรรทัดนี้จะเป็น Key ของตารางใหม่ที่จะแยกออกมาเพื่อให้ตารางมีคุณสมบัติเป็น 2NF คือจากตารางเริ่มต้นจะถูกแบ่งออกมาเป็น 3 ตาราง จะกำหนดชื่อของตารางใหม่เหล่านี้ว่า ตาราง Project, ตาราง Employee และตาราง Assign ตามลำดับ

2. เขียน Attribute ที่ถูกระบุได้โดย Key ใหม่ไว้ข้างหลัง Key ของตารางใหม่ในแต่ละ Key ได้ดังนี้

PROJECT(PROJ_NUM,PROJ_NAME)

EMPLOYEE(EMP_NUM,EMP_NAME,JOB_CLASS,CHG_HOUR)

ASSIGN(PROJ_NUM,EMP_NUM,HOURS)

เพราะว่า Attribute HOURS ถูกระบุได้โดย PROJ_NUM และ EMP_NUM เท่านั้นจึงต้องใส่ไว้ที่ตาราง Assign เท่านั้น

ผลลัพธ์ของการทำงานนี้จะได้ Dependency Diagram ใหม่ดัง แผนภาพ 8.2 จะเห็นว่า CHG_HOUR จะถูกระบุถึงได้โดยใช้ JOB_CLASS เพราะไม่ CHG_HOUR ก็ JOB_CLASS เป็น Prime(Key) Attribute ซึ่งไม่ใช่ส่วนใดส่วนหนึ่งของ Key เราเรียกเหตุการณ์ว่า Transitive Dependency และ Transitive Dependency ยังทำให้เกิด Data Anomalies อีกด้วย

Table name PROJECT

Table name EMPLOYEE

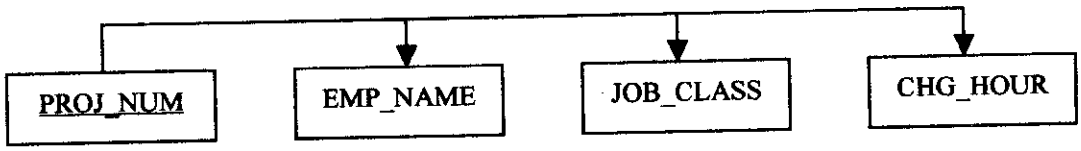
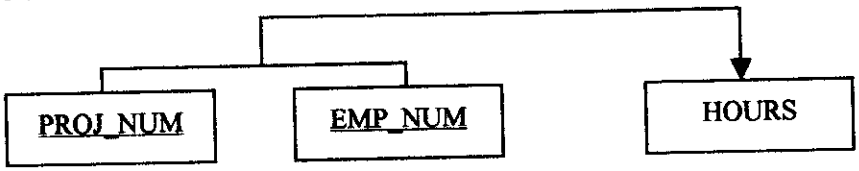


Table name ASSIGN



แผนภาพ 8.2 Second Normal Form (2NF) Conversion Results

2NF DEFINITION

A table is in 2NF if :

1. It is in 1NF

and

2. It includes no partial dependencies, that is no attribute is dependent on only a portion of the primary key.

(But it is still possible for a table 2NF to exhibit transitive dependency, that is one or more attributes may be functionally dependent on nonkey attribute)

นิยาม 2NF

ตารางจะมีคุณสมบัติ 2NF ถ้า

1. **เป็น 1NF และ**
2. **ไม่มี Partial Dependencies คือไม่มี Attribute ใดถูกระบุได้โดยใช้ส่วนหนึ่งของ Primary Key**

(แต่ตารางยังคงมีคุณสมบัติเป็น 2NF ได้ถ้ามีลักษณะ Transitive Dependencies ถ้าเป็นการเกิด Transitive Dependencies กับ Nonkey Attribute)

เพราะการเกิด Partial Dependency สามารถเกิดได้เฉพาะตารางที่มี Primary Key เกิดจากการรวมหลาย Attribute เข้าด้วยกันเป็น Key ถ้าตารางใดใช้ Attribute เดียวเป็น Primary Key และเป็น 1NF ก็จะมีคุณสมบัติ 2NF ไปด้วย

8.1.4 Conversion to Third Normal Form

จากฐานข้อมูลตั้งต้นในแผนภาพ 8.2 เป็นการง่ายมากที่จะแบ่งตารางที่มี Transitive Dependency ออกเป็นตารางใหม่คือแยกส่วนที่เป็น Transitive Dependency คือ JOB_CLASS และ CHG_HOUR ออกไปสร้างตารางใหม่ แต่ JOB_CLASS ยังต้องมีอยู่ในตารางเดิม กับตารางใหม่ที่จะสร้างขึ้น หลังจากแบ่งฐานข้อมูลแล้วจะได้ฐานข้อมูลที่มี 4 ตารางดังนี้

PROJECT(PROJ_NUM,PROJ_NAME)

ASSIGN(PROJ_NUM,EMP_NUM,HOURS)

EMPLOYEE(EMP_NUM,EMP_NAME,JOB_CLASS,CHG_HOUR)

JOB(JOB_CLASS,CHG_HOUR)

3NF DEFINITION

A table is in 3NF if

1. It is in 2NF.

and

2. If contains no transitive dependencies.

นิยาม 3NF

ตารางจะมีคุณสมบัติ 3NF ถ้า

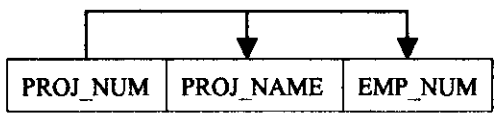
1. เป็น 2NF และ
2. ไม่มี Transitive Dependencies

เรายังสามารถปรับปรุงฐานข้อมูลนี้ให้มีความสามารถในการเป็น Information ได้มากขึ้นไปอีกโดยการ

- เพิ่มรหัสให้กับ JOB_CLASS โดยเพิ่ม Attribute JOB_CODE เพื่อให้การค้นหาข้อมูลที่ง่ายขึ้นและเปลี่ยน Attribute JOB_CLASS ในตาราง EMPLOYEE ให้เป็น Attribute JOB_CODE เพื่อให้ Attribute JOB_CODE เป็น Foreign Key แทน JOB_CLASS
- เปลี่ยนชื่อ CHG_HOUR เป็น JOB_CHG_HOUR และจาก JOB_CLASS เป็น JOB_DESCRIPTION และในตาราง ASSIGN เปลี่ยนชื่อเป็น Attribute HOURS เป็น ASSIGN_HOURS เพื่อให้ชื่อ Attribute สามารถบอกได้ว่ามาจากตารางชื่ออะไร
- ในตาราง EMPLOYEE เพิ่ม Attribute EMP_LNAME ใส่นามสกุลพนักงาน และเพิ่ม Attribute EMP_FNAME ใส่ชื่อพนักงาน และ เพิ่ม Attribute EMP_INTIAL ใส่ชื่อกลางของพนักงาน
- และในตาราง EMPLOYEE เพิ่ม Attribute EMP_HIREDATE เพื่อใส่วันเริ่มทำงานเพื่อจะได้ดูว่าทำงานมานานเท่าไหร่แล้ว
- เพิ่ม EMP_NUM เป็น Foreign Key ในตาราง PROJECT เพื่อเก็บรายละเอียดของหัวหน้างานในแต่ละงาน
- เพิ่ม EMP_NUM และ PROJ_CODE ในตาราง ASSIGN เพื่อเพิ่มความสะดวกในการค้นหาข้อมูลต่างๆ

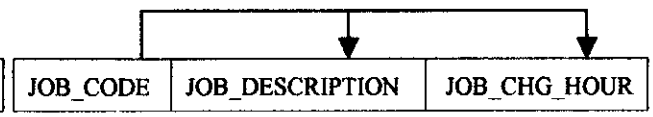
โดยปรับปรุงใหม่ได้ดังแผนภาพ 8.3

Table name : PROJECT



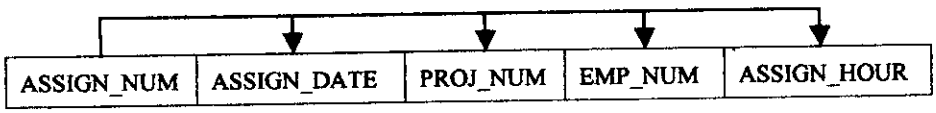
PROJ_NUM	PROJ_NAME	EMP_NUM
15	Evergreen	105
18	Amber Wave	104
22	Rolling Tide	112
25	Startfight	101

Table name : JOB



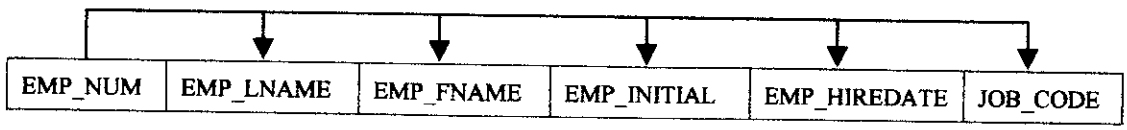
JOB_CODE	JOB_DESCRIPTION	JOB_CHG_HOUR
500	Programmer	$35.75
501	System Analyst	$96.75
502	Database Designer	$105.00
503	Elect. Engineer	$84.50
504	Mechanical Engineer	$67.90
505	Civil Engineer	$55.78
506	Clerical Support	$26.87
507	DSS Analyst	$45.95
508	Application Designer	$48.10
509	Bio Technician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

Table name : ASSIGN



ASSIGN_NUM	ASSIGN_DATE	PROJ_NUM	EMP_NUM	ASSIGN_HOUR
1001	9/8/96	15	103	2.6
1002	9/8/96	18	118	1.4
1003	9/8/96	15	101	3.6
1004	9/8/96	22	113	2.5
1005	9/8/96	15	103	1.8
1006	9/8/96	25	115	4.1
1007	9/8/96	22	105	5.2
1008	9/8/96	25	101	1.7
1009	9/8/96	15	105	2.0
1010	9/10/96	15	102	3.3
1011	9/10/96	22	104	2.6
1012	9/10/96	15	101	2.3
1013	9/10/96	25	114	1.8
1014	9/10/96	22	111	3.9
1015	9/10/96	25	114	3.4
1016	9/10/96	18	112	1.2
1017	9/10/96	18	118	2.1
1018	9/10/96	18	104	2.6
1019	9/10/96	15	103	3.0
1020	9/11/96	22	105	2.7
1021	9/11/96	25	108	4.1
1022	9/11/96	25	114	5.8
1023	9/11/96	22	106	2.4

Table name : EMPLOYEE



EMP_NUM	EMP_LNAME	EMP_FNAME	EMP_INITIAL	EMP_HIREDATE	JOB_CODE
101	News	John	G	11/8/92	502
102	Senor	David	H	7/12/87	501
103	Arbough	June	E	12/1/94	503
104	Remoras	Anne	K	11/15/85	501
105	Smithfield	Alice	K	2/1/91	502
106	Alonzo	Wiliam		6/23/90	500
107	Washington	Mana	D	10/10/91	500
108	Smith	Ralph	B	8/22/89	501
109	Olenko	Larry	W	7/18/95	501
110	Wabash	Gerald	A	12/11/93	505
111	Smithson	Geoff	B	4/4/89	506
112	Joen brood	Darlene	M	10/23/92	507
113	Jones	Delbert	K	11/15/94	508
114	Bawangi	Annelise		8/20/91	508
115	Pratt	Travis	B	1/25/90	501
116	Wiliamson	Gerald	L	3/5/95	510
117	Frammer	Ange	H	6/19/94	509
118		James	J	1/1/95	510

แผนภาพ 8.3 The Complete Database

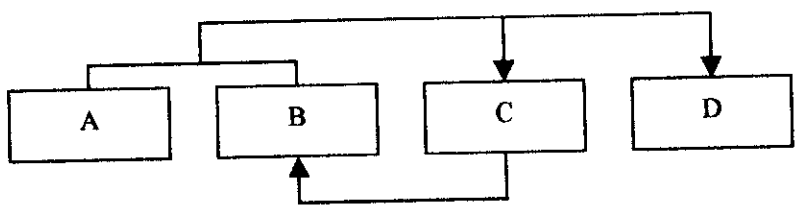
หลังการปรับตามแผนภาพ 8.3 แล้วจะได้ตารางที่มีคุณสมบัติ 3NF 3 ตารางคือตาราง PROJECT, ตาราง ASSIGN, และตาราง EMPLOYEE และตารางที่มีคุณสมบัติ 2NF อีก 1 ตาราง คือ ตาราง JOB

8.1.5 The BOYCE-CODD normal Form (BCNF)

ตารางที่มีคุณสมบัติเป็น BOYCE-CODD Normal Form (BCNF) ก็ต่อเมื่อทุกๆ Determinant ในตารางเป็น Candidate Key

Determinant คือ Attribute ใดๆ ที่มีค่าสามารถระบุไปถึงข้อมูลต่างๆ ในแถวข้อมูล คือถ้าตารางมีคุณสมบัติ 3NF และมี Candidate Key เพียงตัวเดียว ตารางก็จะมีคุณสมบัติ BCNF ไปด้วย

ผู้ออกแบบส่วนมากจะแบ่ง BOYCE-CODD Normal Form (BCNF) เป็นลักษณะพิเศษออกจาก 3NF เพราะบางทีตารางที่มีคุณสมบัติ 2NF แต่ไม่มี Transitive Dependencies แต่มี Nonkey Attribute บางตัวสามารถระบุไปถึง Attribute ที่เป็น Key ได้ดังนั้นตารางที่ไม่มีคุณสมบัติ 3NF แต่มีคุณสมบัติ BCNF ได้ ดังนั้นตารางที่ไม่มีคุณสมบัติ 3NF ก็สามารถมีคุณสมบัติ BCNF ได้ เพราะคุณสมบัติ BCNF ต้องการแค่มี Determinant ทุกตัวในตารางเป็น Candidate Key ตัวอย่าง แผนภาพที่ 8.4 แสดงตารางที่มีคุณสมบัติ 3NF แต่ไม่มีคุณสมบัติ BCNF



แผนภาพ 8.4 Table That is in 3NF but not BCNF

จากแผนภาพ 8.4 เราจะเขียนเป็น Function ได้ดังนี้

A + B ⟶ C, D

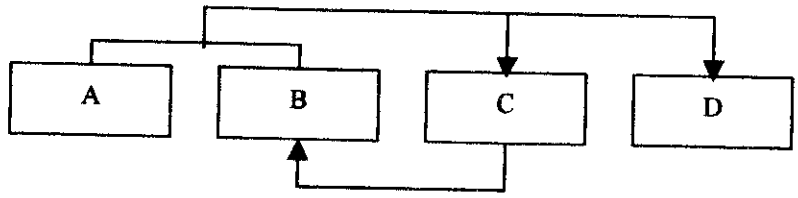
C ⟶ B

จะเห็นว่าไม่มีการเกิด Partial Dependencies เพราะว่า C → B ไม่ใช่ Nonkey Attribute ที่ระบุได้โดยใช้บางส่วนของ Primary Key และไม่ Transitive ด้วย จึงทำให้ตารางนี้มีคุณสมบัติ 3NF แต่ไม่มีคุณสมบัติ BCNF

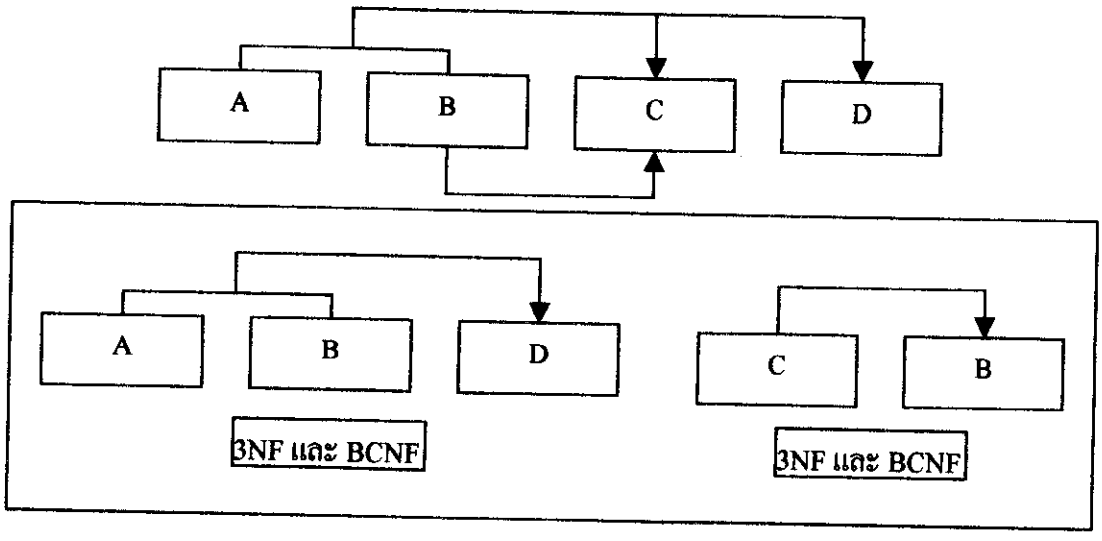
ในการทำตารางนี้ให้มีคุณสมบัติเป็น BCNF ต้องเริ่มจากเปลี่ยนโครงสร้างตารางในแผนภาพที่ 8.4 ให้มีคุณสมบัติ 3NF และ BCNF โดยขั้นแรกเปลี่ยน Primary Key ให้เป็น A + C ก่อน ที่เปลี่ยนได้เนื่องจากมี Function C → B หมายความว่า C เป็น Superset ของ B คือ B ทุกตัวสามารถถูกแทนที่ได้ด้วย C แต่ถ้าเปลี่ยน Primary Key ไปเป็น A + C แล้วจะทำให้ตารางนี้

คุณสมบัติแค่ 1NF เพราะ C → B เป็น Partial Dependency บน Key และเราสามารถแบ่ง (Decomposition) ออกไปให้ได้คุณสมบัติเป็น 3NF และ BCNF ดังแผนภาพ 8.5

3NF แต่ไม่เป็น BCNF



1NF



แผนภาพ 8.5 The decomposition of a table structure to meet BCNF requirements

เราสามารถนำวิธีการ Decomposition นี้มาประยุกต์ใช้งานกับข้อมูลตัวอย่างได้กับ ตัวอย่างตารางฐานข้อมูล 8.2

STU_ID	STAFF_ID	CLASS_CODE	ENROLL_GRADE
125	25	21334	A
125	20	32456	C
135	20	28458	B
144	25	37563	C
144	20	32456	B

ตารางฐานข้อมูล 8.2 Sample data for a BCNF conversion

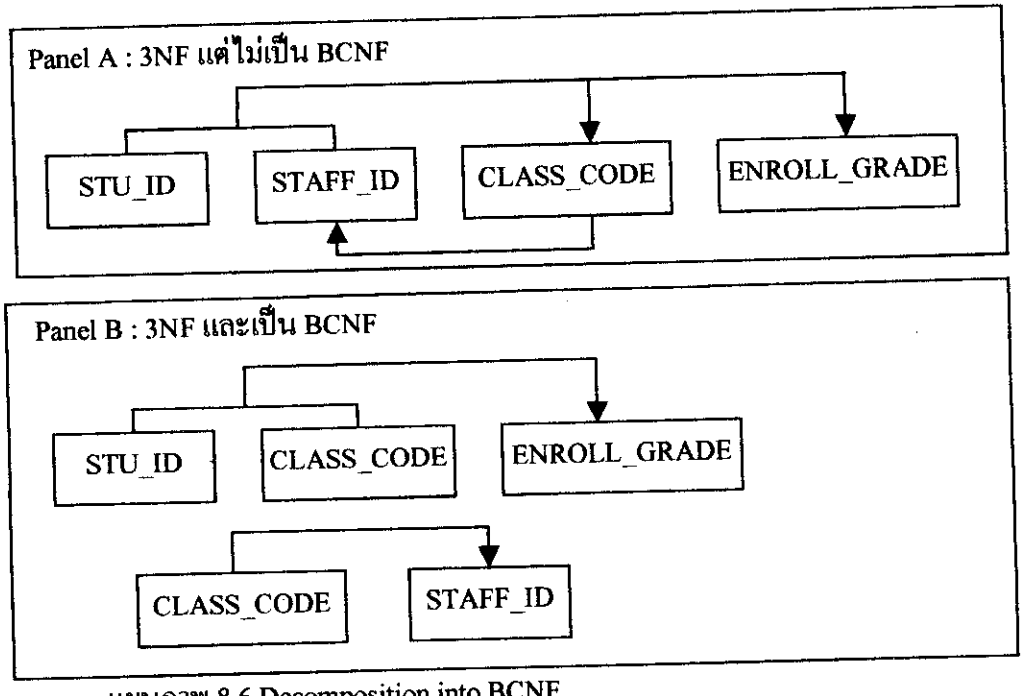
จากตารางฐานข้อมูล 8.2 กำหนดเงื่อนไขได้ดังนี้

- CLASS_CODE เป็นรหัสห้องเรียนที่ระบุถึงห้องๆ เดียว
- นักเรียนคนหนึ่งสามารถเรียนได้หลายวิชา
- ผู้สอนสามารถสอนในหลายห้องเรียนแต่ละห้องเรียนจะมีผู้สอนได้ 1 คนเท่านั้น

จะสามารถแสดงรายการ การเกิด Dependencies ได้ดังนี้

STUD_ID + STARR_ID ⟶ CLASS_CODE,ENROLL_GRADE

CLASS_CODE ⟶ STAFF_ID



แผนภาพ 8.6 Decomposition into BCNF

จากแผนภาพ 8.6 แสดงโครงสร้างที่คุณสมบัติ 3NF อย่างชัดเจน แต่ตารางนี้ยังมีปัญหาใหญ่อยู่ เพราะมันจะทำให้สิ่งที่ซ้ำกันเช่น ครูที่สอนห้อง 32456 2 แถวซึ่งอาจจะทำให้เกิดปัญหา Anomaly ในการ Update ข้อมูล ถ้านักเรียน Drop วิชาที่เรียนที่ห้อง 28458 จะทำให้ข้อมูลของผู้สอนในห้องนี้หายไปด้วย ทำให้เกิด Anomaly ในการลบข้อมูลจากแผนภาพ 8.6 จะเห็นว่าโครงสร้างตารางทั้ง 2 ตารางเป็น 3NF ที่เหมือนกับ BCNF

BOYCE-CODD NORMAL FORM DEFINITION

A table is in BCNF if every determinant in that table is a candidate key. If a table contains only one candidate key, 3NF and BCNF are equivalent

ตารางที่มีคุณสมบัติเป็น BCNF ก็ต่อเมื่อทุกๆ Determinant ในตารางนั้นเป็น Cadidate Key ทั้งหมด ถ้าตารางนั้นมี Cadidate Key ตัวเดียว คุณสมบัติ 3NF และ BCNF ก็เหมือนกัน

8.2 การทำให้เป็นบรรทัดฐานและออกแบบฐานข้อมูล (Normalization and Database Design)

การทำ Normalization ควรจะเป็นส่วนหนึ่งในขั้นตอนการออกแบบ เพื่อให้เป็นการทดสอบดูก่อนว่าได้ Entities ที่ต้องการจริงก่อนการสร้างโครงสร้างของตารางทั้งหมด

ส่วนการทำ E-R Diagram เป็นการแสดงภาพรวมของการรวบรวมความต้องการและการทำงานของข้อมูล โดยเริ่มจากการกำหนด Entities ท่ามีความสัมพันธ์กันคือ Attributes ใน Entities และความสัมพันธ์ระหว่าง Attribute ใน Entities

ในการทำ Normalization จะต้องระบุถึงคุณสมบัติของแต่ละ Entity ซึ่ง Normalization จะแสดงรายละเอียดของ Entities ได้มากกว่าใน E-R Diagram และ Normalization จะแยก Entities และ Attribute ที่มีอยู่ใน E-R Diagram ออกไปอีกเนื่องจากเป็นการยากที่จะทำ Normalization จากการทำ E-R Diagram แต่ว่าควรจะใช้ทั้ง 2 วิธีควบคู่กันไป

เพื่อแสดงให้เห็นบทบาทของการทำ Normalization ในขั้นตอนการออกแบบจะขอยกตัวอย่างการทำงานของบริษัท Contracting Company โดยมีการทำงานสรุปได้ดังนี้

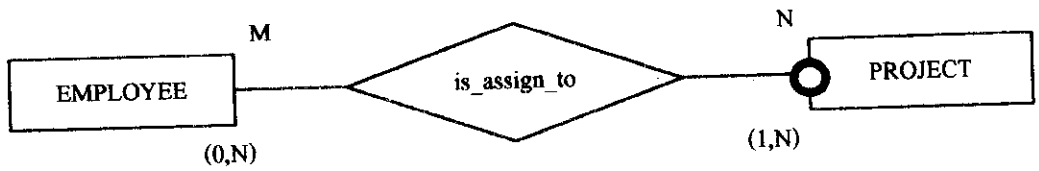
1. บริษัทรับงานหลายๆ งานพร้อมกันในเวลาเดียวกัน
2. ในแต่ละงานจะมีพนักงานทำได้หลายคน
3. พนักงานคนหนึ่งอาจถูกระบุให้ทำงานที่ต่างกันในหลายงาน
4. พนักงานแต่ละคนจะถูกกำหนด ลักษณะงานพื้นฐานไว้โดยลักษณะของงานนี้จะเป็นตัวกำหนดอัตราค่าแรงรายชั่วโมง
5. พนักงานหลายคนสามารถมีลักษณะงานเดียวกันได้

กำหนด Attribute ตั้งต้นไว้ดังนี้

1. PROJECT(PROJ_NUM,PROJ_NAME)
2. EMPLOYEE(EMP_NUM,EMP_LNAME,EMP_FNAME,EMP_INITIAL,JOB_DESCRI PTION,JOB_CHG_HOUR)

2 Entities นี้จะเขียนเป็น E-R Diagram ได้ดังแผนภาพ 5.7 หลังจากการเขียน E-R Diagram แล้วจะกำหนด Normal Form ได้ดังนี้

1. PROJECT มีคุณสมบัติ 3NF และไม่ต้องปรับปรุงใดๆ อีกแล้ว

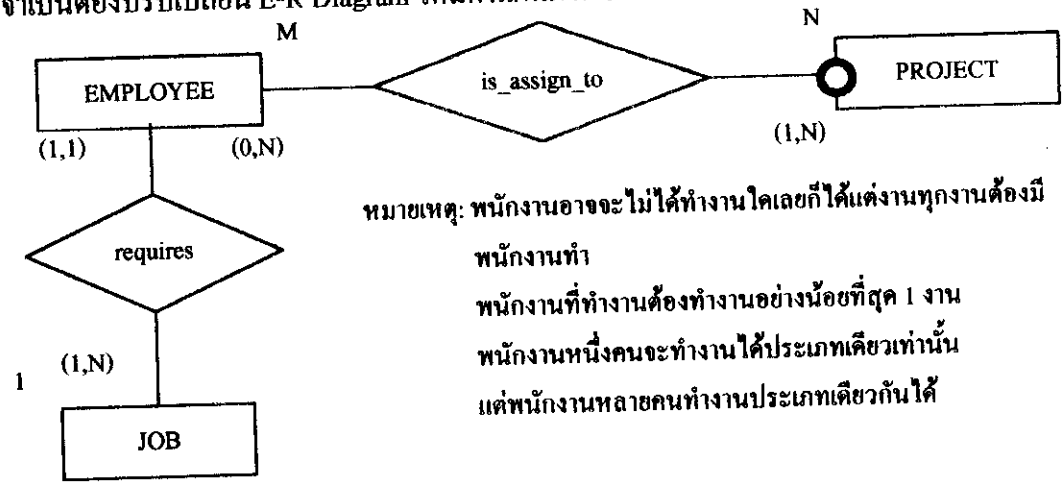


แผนภาพ 5.7 The initial E-R Diagram for a Contracting Company

2. EMPLOYEE ต้องการวิเคราะห์เพิ่ม จะเห็นว่า JOB_DESCRIPTION จะสามารถระบุไปถึง JOB_CHG_HOUR ได้แสดงว่า EMPLOYEE มีลักษณะเป็น Transitive Dependency

ทำการแบ่ง Transitive Dependency ของตาราง EMPLOYEE ออกจะได้ Entities ทั้งหมด 3 Entities ได้แก่

เพราะว่าจากการทำ Normalization นี้ทำให้เกิด Entity ใหม่อีกหนึ่ง Entity คือ Entity JOB เราจึงจำเป็นต้องปรับเปลี่ยน E-R Diagram ใหม่ดังแผนภาพ 5.8

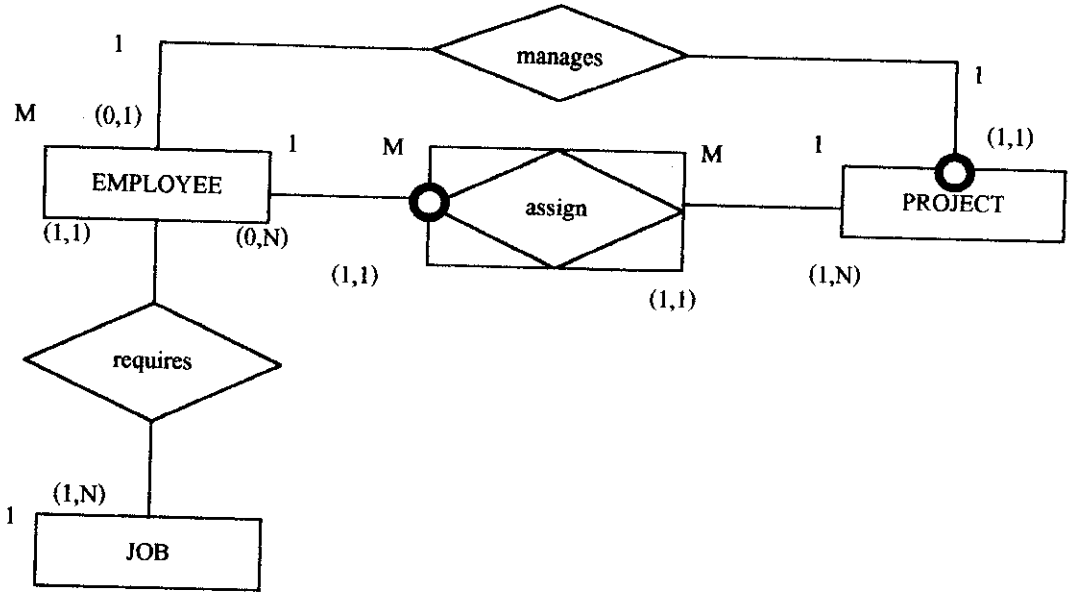


หมายเหตุ: พนักงานอาจจะไม่ได้ทำงานใดเลยก็ได้แต่งานทุกงานต้องมีพนักงานทำ
พนักงานที่ทำงานต้องทำงานอย่างน้อยที่สุด 1 งาน
พนักงานหนึ่งคนจะทำงานได้ประเภทเดียวเท่านั้น
แต่พนักงานหลายคนทำงานประเภทเดียวกันได้

แผนภาพ 5.8 The modified E-R Diagram for a Contracting Company

CT 316 (S)

จากแผนภาพ 8.8 มีความสัมพันธ์ M : N ระหว่าง EMPLOYEE กับ PROJECT จึงต้องแบ่ง E-R Diagram ใหม่ดังแผนภาพ 8.9 โดยการเพิ่ม Entity Assign ขึ้นมา

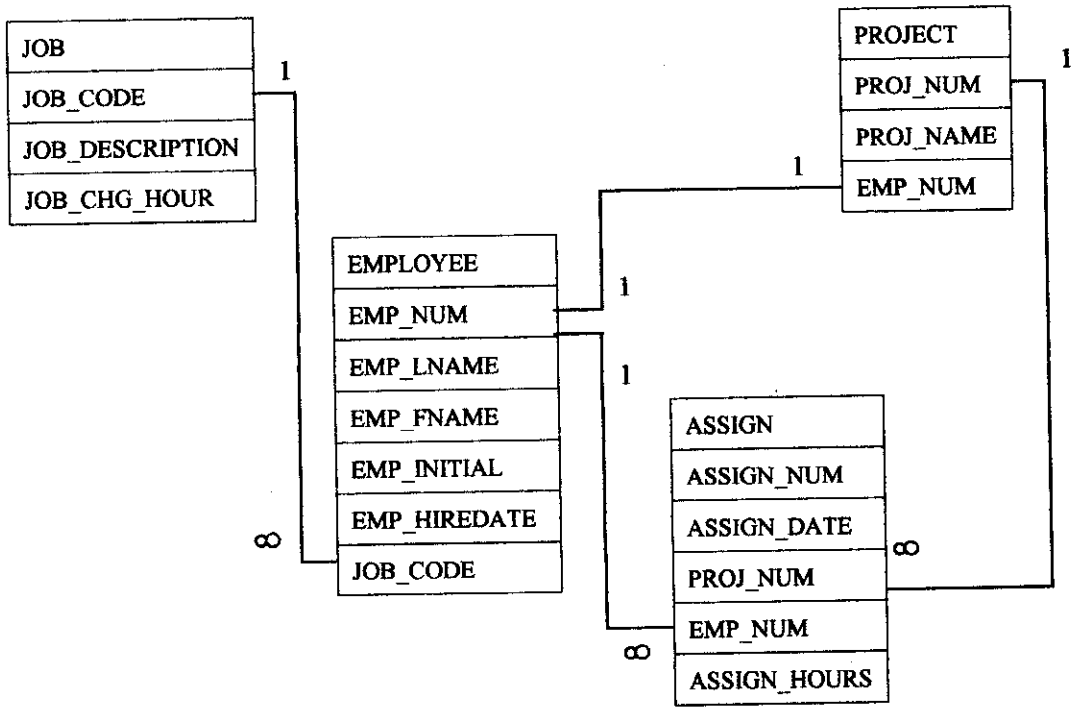


แผนภาพ 8.9 The final (Implementable) E-R Diagram for The Contracting Company

แต่ในแผนภาพ 8.9 ยังไม่แสดงถึงจำนวนชั่วโมงการทำงานของพนักงานแต่ละคนในแต่ละงาน จึ้งต้องเพิ่ม Attribute ASSIGN_HOURS ใน Entity Assign และเพิ่มรหัสพนักงานที่เป็นหัวหน้างานไว้ใน Entity PROJECT ด้วย เพื่อให้รู้ว่าแต่ละงานมีพนักงานคนไหนเป็นหัวหน้างาน และเพิ่ม Attribute EMP_HIREDATE ลงไปใน Entity EMPLOYEE เพื่อให้รู้ว่าพนักงานเริ่มทำงานกับบริษัทมานานเท่าไหร่แล้วจากแผนภาพ 8.9 และหลังจากเพิ่ม Attribute ต่างๆ แล้วจะได้ฐานข้อมูลใหม่ที่มี 4 Entities ได้แก่

1. PROJECT(PROJ_NUM,PROJ_NAME)
2. EMPLOYEE(EMP_NUM,EMP_LNAME,EMP_FNAME,EMP_INITIAL,JOB_CODE)
3. JOB(JOB_CODE,JOB_DESCRIPTION,JOB_CHG_HOUR)
4. ASSIGN(ASSIGN_NUM,ASSIGN_DATE,PROJ_NUM,EMP_NUM, ASSIGN_HOURS)

ถึงตรงนี้ทุก Entities มีคุณสมบัติ 3NF และเราจะเขียนความสัมพันธ์ของ Scheme ได้ดังแผนภาพ 8.10



แผนภาพ 8.10 The relational schema for the Contracting Company

8.3 Higher-Level Normal Forms

ถึงแม้ว่าคุณสมบัติ 3NF จะมีความเพียงพอต่อระบบธุรกิจแล้วก็ตาม ก็ยังมีบางเหตุกรณีที่อาจต้องการใช้คุณสมบัติในระดับ Higher-Level Normal Forms โดยที่ Attribute ในแต่ละแถวจะต้องมีสิ่งต่อไปนี้

1. Attribute ทั้งหมดจะต้องถูกระบุได้โดย Primary Key แต่ Attribute เหล่านั้นอาจถูกระบุได้โดย Attribute อื่นๆ ก็ได้
2. ไม่มีแถวข้อมูลใดที่อ้างอิงไปถึงกลุ่มของค่าที่มากกว่า 1 ค่าใน Entity

ตัวอย่างฐานข้อมูลเก็บสถานที่ทำงานที่พนักงานทำงานว่าที่ Red Cross หรือ United Way ก็ได้โดยในพนักงานคนเดียวกันก็อาจจะถูกกำหนดให้ทำงาน 1,5 และ12 ก็ได้ ในตัวอย่างตารางฐานข้อมูล 5.4 จะเห็นว่าระบบนี้สามารถสร้างตารางเก็บข้อมูลได้หลายวิธี

Version 1

EMP_NUM	EMP_SERVICE	EMP_ASSIGN
10123	Red Cross	1
10123	United Way	5
10123		12

Version 2

EMP_NUM	EMP_SERVICE	EMP_ASSIGN
10123	Red Cross	
10123	United Way	
10123		1
10123		5
10123		12

Version3

EMP_NUM	EMP_SERVICE	EMP_ASSIGN
10123	Red Cross	1
10123	Red Cross	5
10123	United Way	12

ตาราง 8.4 Table with multi valued dependencies

อย่างไรก็ตามก็ตาม แต่ละตารางที่แสดงในตารางฐานข้อมูล 8.4 นั้นมีคุณสมบัติ 3NF เราจะพบปัญหาว่า Attribute EMP_SERVICE และEMP_ASSIGN อาจจะมีค่าที่ต่างกันได้มากมายเพราะว่าในตารางเก็บค่าข้อมูล 2 อย่างที่ขึ้นอยู่กับค่าที่แตกต่างกันอีกมากมาย

ปัญหานี้คือลักษณะ Multiple set of Multi valued Dependencies ถ้า Version 1 และ 2 ถูกเพิ่มข้อมูลอีกจะพบว่ามีค่า Null values เกิดขึ้นมากถ้า Version 3 ถูกเพิ่มก็จะเกิดความซ้ำซ้อนของข้อมูลมากมาย

วิธีแก้ปัญหาคือต้องขจัดตัวปัญหาที่เป็น Multi valued Dependencies ออก คือสร้างตารางขึ้นใหม่ 2 ตาราง ดังตัวอย่างข้อมูล 8.5 ไม่มี Multi valued Dependencies จึงทำให้มีคุณสมบัติ Fourth Normal Form (4NF) โดยเราสามารถนิยาม 4NF ได้ดังนี้

4NF DEFINITION

A table is in fourth normal form if it is in 3NF and has no multiple sets of multi valued dependencies.

นิยาม 4NF

ตารางจะมีคุณสมบัติ 4NF ถ้าตารางนี้มีคุณสมบัติ 3NF และไม่มี Multiple set ของ Multivalued Dependencies

EMP_NUM	EMP_SERVICE
1023	Red Cross
1023	United Way

EMP_NUM	EMP_ASSIGN
1023	1
1023	5
1023	12

ตารางฐานข้อมูล 8.5 A set of tables in 4NF

ในความเป็นจริงยังมี Higher-Level Normal Form อื่นๆ อีกคือ 5NF และ Domain Key Normal Form (DKNF) แต่ในระดับเริ่มต้นนี้แค่ 4NF ก็เพียงพอกับการใช้งานในระบบธุรกิจแล้ว

8.4 การทำให้ไม่เป็นบรรทัดฐาน (Denormalization)

ถึงแม้ว่าการทำ Normalization มีความสำคัญในการออกแบบฐานข้อมูลแต่การทำ Normalization เป็นเพียงวิธีเดียวในอีกหลายๆ วิธีที่จะทำให้เกิดฐานข้อมูลที่ดีการออกแบบฐานข้อมูลที่ดีจะต้องประกอบด้วยการพิจารณาถึงความต้องการในการทำงาน แต่การแบ่งตารางออกไปเพื่อให้มีคุณสมบัติ Normal Form ที่ต้องการอาจทำให้ลดความเร็วของระบบคอมพิวเตอร์ได้เช่นการแบ่งตารางออกไปมากการเชื่อมตารางด้วยคำมีมีขนาดใหญ่ซึ่งจะทำให้เปลืองเนื้อที่ใน

อุปกรณ์จัดเก็บซึ่งในบางทีอาจจะใช้การทำ Denormalization คือใช้ Normal Form ที่มีระพับต่ำลงไป เพื่อช่วยเพิ่มความเร็วในการทำงานได้

แต่ถ้าต้องการการทำงานที่มีความเร็วสูงกว่าเดิมอาจทำให้เกิดผลเสียคือเกิด Data Anomalies เช่น ต้องการฐานข้อมูลของคนที่อาศัยอยู่ทั่วโลกโดยต้องการ 2 Attribute คือ ZIP_CODE และ CITY โดยให้นำข้อมูลมาจากตาราง CUSTOMER โดยใช้ ZIP_CODE มีค่าระบุไปถึง CITY ได้โดยจะแบ่งออกมาเป็นตารางใหม่ได้ดังนี้

ZIP(ZIP_CODE,CITY)

เพื่อหลีกเลี่ยงการเกิด Transitive Dependency ในตาราง Customer ข้อมูลที่ได้จะใช้ในงานธุรกิจประเภท Mailing Lists

แต่การทำ Normalization เพียงอย่างเดียวยังไม่เพียงพอแก่การทำงานในระบบฐานข้อมูลยุคใหม่ซึ่งมีความแตกต่างกันระหว่าง การออกแบบอย่างมีประสิทธิภาพ ความต้องการข้อมูลที่จะนำไปใช้ประกอบการตัดสินใจได้ง่ายและความเร็วในกรทำงานบ่อยครั้งต้องมีการใช้การทำ Denormalization เข้ามาด้วยบางทีอาจใช้ Normal Form ในระดับที่ต่ำลงไปเพื่อช่วยในการลดปัญหาลง ตัวอย่างเช่นใน แผนภาพ 5.3 ในตาราง JOB มีคุณสมบัติแต่ 2NF ก็เพียงพอเพื่อลดปัญหาด้านความคงสภาพของข้อมูล (Referential Integrity) ซึ่ง Normal Form ในระดับที่ต่ำกว่าจะถูกใช้ในฐานข้อมูลพิเศษที่เรียกว่า Data Warehouses ผู้ออกแบบงานฐานข้อมูล แต่ละคนจะคิดถึงขอบเขตของการเพิ่มขึ้นของข้อมูลซึ่งใช้ในการตัดสินใจเพื่อเพิ่มความเชื่อมั่นในระบบธุรกิจ คือถ้ายิ่งมีข้อมูลมากก็จะมีช่วงควรเชื่อมั่นที่สูงขึ้นโดยส่วนมากการทำงานในระบบ Data Warehouses จะใช้ Normal Form ในระดับ 2NF ก็เพียงพอแล้ว

แต่ไม่ได้หมายความว่า การทำให้ตารางข้อมูลมี Partial Dependencies และปัญหา Transitive Dependencies ลดลงแล้วจะเป็นการหลีกเลี่ยงการเกิดปัญหา Data Anomalies ได้และการใช้ฐานข้อมูลที่มีคุณสมบัติเป็น Normal Form ต่ำๆ และตารารงที่ไม่ได้ถูกทำ Normalization จะเป็นตารางที่ดีเพราะยังมีข้อบกพร่องดังต่อไปนี้

1. การปรับปรุงข้อมูลจะมีประสิทธิภาพต่ำเพราะโปรแกรมจะต้องอ่านข้อมูลและปรับปรุงข้อมูลในตารางที่มีขนาดใหญ่

2. ทำ Index ได้ยากมากเพราะต้องใช้หลาย Attribute รวมกันในการค้นหาข้อมูลที่ต้องการ

3. จะไม่สามารถปรับเปลี่ยนมุมมอง (Logical View) ได้มากใช้ได้เพียงแต่มุมมองข้อมูลอย่างง่ายๆ เท่านั้น

Summary

Normalization คือเทคนิคที่ช่วยในการออกแบบตารางฐานข้อมูลเพื่อลดการเกิดปัญหา Data Redundancies โดย 1NF,2NF และ 3NF เป็นการทำ Normalization อย่างง่าย ตารางฐานข้อมูลที่มีคุณสมบัติ Normal Form ในระดับสูงก็จะลดปัญหาการเกิด Data Redundancies ได้ดีกว่าตารางฐานข้อมูลที่มีคุณสมบัติ Normal Form ในระดับที่ต่ำกว่า ในระบบธุรกิจทั่วไปใช้ตารางฐานข้อมูลที่มีคุณสมบัติ 3NF ก็เพียงพอแล้ว

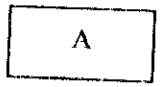
ตารางจะมีคุณสมบัติ 1NF เมื่อมีการกำหนด Attribute ที่เป็น Key ทุกตัว และทุกๆ Attribute จะต้องถูกระบุได้โดย Primary Key และ Attribute ในแต่ละ Row และ Column ที่ตัดกันต้องได้ข้อมูลเป็นค่าๆ เดียวเท่านั้น ไม่ใช่กลุ่มของค่าข้อมูลแต่ในตาราง 1NF อาจจะพบว่ามีการเกิดปัญหาทั้ง Partial Dependencies และ Transitive Dependencies แต่ถ้ามี Primary Key เกิดจาก Attribute เดียวก็จะไม่เกิดปัญหา Partial Dependencies

ตารางจะมีคุณสมบัติ 2NF ก็ต่อเมื่อ เป็นตารางที่มีคุณสมบัติ 1NF และไม่มี Partial Dependencies โดยตารางที่มีคุณสมบัติ 1NF จะมีคุณสมบัติเป็นตาราง 2NF ทันทีถ้าตารางนั้นมีคุณสมบัติ 1NF และใช้ Attribute เพียง Attribute เดียวเป็น Primary Key แต่ว่าในบางที อาจจะพบปัญหา Transitive Dependencies ในตารางที่มีคุณสมบัติ 2NF

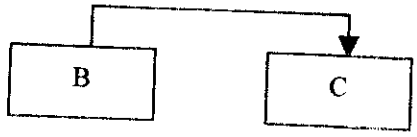
ตารางจะมีคุณสมบัติ 3NF ก็ต่อเมื่อตารางนั้นมีคุณสมบัติเป็น 2NF และไม่มี Transitive Dependencies และตารางที่มีคุณสมบัติ BCNF จะเป็นตารางที่มีลักษณะพิเศษกว่าตารางที่มีคุณสมบัติ 3NF นิดหน่อยคือ ทุกๆ Determinant ต้องเป็น Candidate Key แต่ถ้าตาราง 3NF มี Candidate Key ตัวเดียวก็จะทำให้ตารางที่มีคุณสมบัติเป็น BCNF ไปด้วย

ตารางที่ไม่มีคุณสมบัติ 3NF อาจจะถูกแบ่งออกเป็นตารางใหม่หลายๆ ตารางจนกระทั่งมีคุณสมบัติเป็น 3NF ถ้ามีความต้องการตารางที่มีคุณสมบัติ 3NF ทำได้โดยการทำ ดังต่อไปนี้

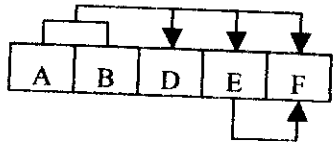
สร้างตารางที่มีคุณสมบัติ 2NF โดยแยก Partial Dependencies ออกไปเป็นตารางใหม่เขียนส่วนประกอบของ Primary Key ไว้ทีละบรรทัดและเขียน Primary Key ไว้บรรทัดสุดท้าย แล้วสร้างตารางขึ้นมาใหม่แล้วเขียน Attribute ที่ถูกระบุได้โดย Key ตัวใหม่ต่อท้าย Key แต่ละตัวและทำการหา Primary Key ใหม่ ถ้า Key ตัวใดไม่สามารถระบุไปถึง Attribute ใดๆ ได้เลยก็ไม่ต้องพิจารณา ดังต่อไปนี้



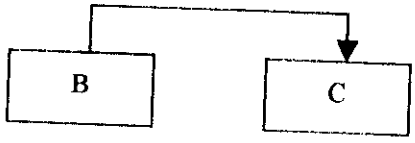
ไม่มี Attribute ถูกระบุได้โดย A แสดงว่า A ไม่สามารถเป็น Primary Key ของตารางได้



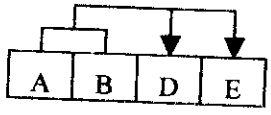
ตารางนี้มีคุณสมบัติเป็น 3NF เพราะตารางนี้มีคุณสมบัติเป็น 2NF และตารางนี้ไม่มี Transitive Dependencies



ตารางที่มีคุณสมบัติ 2NF เพราะมี Transitive Dependency : E → F



แยก E,F ออกมาเป็นตารางใหม่และให้ E ยังอยู่ในตารางเดิมเพื่อให้เป็น Foreign Key เชื่อมไปยังตารางใหม่



ทั้งสองตารางมีคุณสมบัติเป็น 3NF เพราะไม่มี Partial Dependencies และ Transitive Dependency

การทำ Normalization เป็นส่วนหนึ่งในการออกแบบการทำงาน โดย Entities และ Attributes จะถูกกำหนดในการทำ E-R Model แต่ละกลุ่มของ Entity จะถูกตรวจสอบโดยการทำ Normalization และจากกลุ่ม Entity ใหม่ จะต้องรวมเข้าไปใจ E-R Diagram และทำเช่นนี้ไปเรื่อยๆ จนกว่าจะได้ตารางในคุณสมบัติ 3NF หรือคุณสมบัติที่ต้องการ

ตารางหนึ่งๆ ที่เป็น 3NF อาจจะมีลักษณะ Multi valued Dependencies ซึ่งผลลัพธ์ที่ได้อาจทำให้เกิด Null Values ในฐานข้อมูลหรือเกิดความซ้ำซ้อนของข้อมูล ในบางทีอาจต้องการเปลี่ยนตาราง 3NF ไปเป็น Fourth Normal Form(4NF) โดยการแยกตารางโดยแยกส่วนที่จะเป็นลักษณะ Multi valued Dependencies ออก ซึ่งจะทำให้ตารางมีคุณสมบัติ 4NF ถ้าตารางนั้นเป็น 3NF และไม่มี Multi valued Dependencies

ยิ่งมีการแบ่งตารางออกไปมากๆ จะทำให้ใช้เนื้อที่จัดเก็บสูงและจะทำให้ทำงานได้ช้าลง บางทีอาจต้องใช้ Normal Form ที่ต่ำลง (Denormalization) เพื่อเพิ่มความเร็วให้แก่ระบบ แต่ถ้าทำ Denormalization ก็จะทำให้เกิด Data Redundancy และ Data Anomalies ซึ่งทำให้ใช้เนื้อที่มากขึ้น แต่ก็จะเพิ่มความเร็วให้ระบบเพราะการค้นหาไม่ต้องไปค้นหาข้อมูลในหลายตารางมาก