ความรู้เบื้องต้นเกี่ยวกับฐานข้อมูล

ความหมายของฐานข้อมูล

การนำโปรแกรมสำเร็จรูปต่างๆ มาใช้สำหรับการจัดเก็บข้อมูลนั้น จะจัดเก็บในรูปของไฟล์ข้อมูล ซึ่งในบางระบบงานอาจต้องมีการนำข้อมูลต่างๆ จากหลายๆ ไฟล์ข้อมูลมาใช้ร่วมกันตัวอย่างเช่น ในระบบงานซื้อ-ขายสินค้า มีไฟล์ Customer ใช้สำหรับจัดเก็บข้อมูลของลูกค้าโดยภายในจะประกอบไปด้วยข้อมูลที่เกี่ยวกับ รหัสลูกค้า ชื่อลูกค้า ที่อยู่ หมายเลขโทรศัพท์ Email เป็นต้น เมื่อต้องการออกใบเสนอราคาให้กับลูกค้า อาจต้องมีการสร้างไฟล์ขึ้นมาใหม่ที่จะต้องมีข้อมูลต่างๆ ที่เกี่ยวข้องกับลูกค้าอีก เช่น รหัสลูกค้า ชื่อลูกค้า ที่อยู่ เป็นต้น ซึ่งจะทำให้เกิดภาวะที่เรียกว่า **“การซ้ำซ้อนของข้อมูล (Data Redundancy)”** ซึ่งเป็นปัญหาของการจัดเก็บข้อมูลเดียวกันไว้หลายๆ ที่

การที่มีข้อมูลเดียวกันแต่ถูกนำไปจัดเก็บไว้คนละไฟล์ข้อมูลกันนั้น อาจทำให้เกิดปัญหาในเรื่องของความถูกต้องของข้อมูลได้ ตัวอย่างเช่น เมื่อลูกค้าย้ายที่อยู่ใหม่ ผู้ดูแลข้อมูลจะต้องเปลี่ยนแปลงข้อมูลเกี่ยวกับที่อยู่ใหม่ของลูกค้าคนนี้ในทุกๆ ไฟล์ข้อมูลที่เกี่ยวข้อง ซึ่งอาจทำให้เกิดข้อผิดพลาดในความถูกต้องของข้อมูลได้ง่าย ดังนั้นเพื่อการแก้ไขปัญหาเหล่านั้นจึงมีการนำข้อมูลต่างๆ ที่มีความสัมพันธ์กันและต้องใช้งานร่วมกันมาทำการจัดเก็บไว้ที่เดียวกัน เมื่อใดที่มีความต้องการจะเรียกใช้ข้อมูลเหล่านี้ก็สามารถเรียกใช้งานได้จากแหล่งเดียวกัน การนำข้อมูลที่มีความสัมพันธ์กัน มีวัตถุประสงค์ของการใช้งานข้อมูลเป็นไปในทิศทางเดียวกันมาทำการจัดเก็บไว้รวมกันนี้ เราเรียกว่า **“ฐานข้อมูล”**

ฐานข้อมูลเชิงสัมพันธ์ (Relation Model)

ในโครงสร้างสำหรับการจัดเก็บข้อมูลในฐานข้อมูลนั้นจะมีอยู่หลายๆ แบบ สำหรับโครงสร้างที่นิยมใช่กันในปัจจุบันจะเป็นแบบ ฐานข้อมูลเชิงสัมพันธ์ หรือ Relation Model ที่เป็นโครงสร้างที่จัดเก็บข้อมูลต่างๆ ลงในฐานข้อมูลในรูปแบบของตาราง โดยที่ในแต่ละตารางจะมีความสัมพันธ์กันทั้งหมด ในส่วนของการค้นหาข้อมูลสามารถที่จะทำได้ง่ายคือ ทำการค้นหาตามเส้นที่ใช้เชื่อมความสัมพันธ์ระหว่างตาราง ซึ่งโครงสร้างแบบนี้สามารถที่จะทำงานกับฐานข้อมูลได้หลายกลุ่มในเวลาเดียวกัน ช่วยลดความซ้ำซ้อนของการจัดเก็บข้อมูล และมีความยืดหยุ่นสูงในการเปลี่ยนแปลงแก้ไข

ข้อดีของการใช้ฐานข้อมูลเชิงสัมพันธ์

- ในฐานข้อมูลที่มีโครงสร้างแบบสัมพันธ์นี้จะช่วยลดความซ้ำซ้อนของข้อมูลที่เกิดขึ้น ด้วยข้อกำหนดที่ว่า ข้อมูลหนึ่งๆ จะต้องมีอยู่ภายในตารางใดตารางหนึ่งเท่านั้น เว้นแต่ว่าข้อมูลนั้นจะทำหน้าที่เชื่อมความสัมพันธ์ไปยังตารางอื่น ซึ่งการลดความซ้ำซ้อนนี้จะช่วยให้สามารถควบคุมความถูกต้องของข้อมูลได้ง่ายขึ้น นอกจากนี้ยังจะช่วยประหยัดพื้นที่ที่ใช้สำหรับจัดเก็บข้อมูลด้วย
- ฐานข้อมูลเชิงสัมพันธ์สามารถจะทำงานแบบหลายผู้ใช้ได้ (Concurrent Users) ทำให้ผู้ใช้หลายๆ คน หรือโปรแกรมหลายๆ โปรแกรมสามารถเข้าถึงข้อมูลในฐานข้อมูลได้ในเวลาเดียวกัน ซึ่งจะช่วยให้งานนั้นเกิดความสะดวกมากขึ้น
- ฐานข้อมูลแบบนี้สามารถออกแบบให้ง่ายต่อการเปลี่ยนแปลง เมื่อต้องมีการแก้ไขโครงสร้างของข้อมูลใหม่ ไม่ว่าจะเป็นการเพิ่มข้อมูล หรือการลบข้อมูลบางอย่างออกไป

- สามารถใช้กับภาษา SQL (Select Query Language) ที่เป็นภาษามาตรฐานที่ใช้สำหรับการเข้าถึงฐานข้อมูลได้ ทำให้โปรแกรมต่างๆ ที่มีการใช้ภาษา SQL สามารถเข้าถึงฐานข้อมูลได้โดยไม่ยึดติดว่าจะต้องเป็นโปรแกรมใด (Application Independence)

สำหรับซอฟต์แวร์ที่ใช้จัดการกับฐานข้อมูลที่จัดเก็บในโครงสร้างเชิงสัมพันธ์นี้จะเรียกว่า **"ระบบจัดการฐานข้อมูลเชิงสัมพันธ์ (RDBMS: Relational Database Management System)"** ตัวอย่างโปรแกรมพวกนี้ได้แก่ Microsoft Access, Microsoft SQL Server, Oracle, DB2, Informix เป็นต้น

คำศัพท์ที่ควรรู้เกี่ยวกับฐานข้อมูล

Entity หมายถึง กลุ่มของข้อมูลที่อยู่ภายในฐานข้อมูล ซึ่งมีความหมายของกลุ่มข้อมูลเป็นไปในลักษณะเดียวกัน หรืออยู่ภายใต้วัตถุประสงค์เดียวกัน Entity ในฐานข้อมูลมักอยู่ในรูปของบุคคล สถานที่ สิ่งของ หรือการกระทำที่เกิดขึ้น เช่น ในฐานข้อมูลเกี่ยวกับระบบการเช่าวีซีดี จะมีกลุ่มของข้อมูลหรือ Entity คือ กลุ่มข้อมูลสมาชิก (Member) กลุ่มข้อมูลเกี่ยวกับภาพยนตร์ (Movie) กลุ่มข้อมูลเกี่ยวกับการเช่า (Rental) เป็นต้น ในโปรแกรมจัดการฐานข้อมูลทั่วไปอาจเรียก Entity นี้ว่า ***"ตารางข้อมูล (Table)"***

Entity Instance หมายถึง ข้อมูลในแต่ละรายการของกลุ่มข้อมูลเช่น ในกลุ่มข้อมูลพนักงานจะประกอบไปด้วยพนักงานหลายๆ คน โดยจะเรียกข้อมูลในแต่ละรายการหรือของพนักงานแต่ละคนว่า *Entity Instance* ซึ่งในโปรแกรมจัดการฐานข้อมูลทั่วๆ ไป จะเรียกว่า ***"เรคคอร์ด (Record)"***

Entity (Table)

รหัสพนักงาน	ชื่อพนักงาน	นามสกุล	ที่อยู่
4102	พงศ์ศักดิ์	แซ่แต้	แจ้งวัฒนะ
4104	ระพีพร	มงคลชัย	บางลำพู
4125	ชูวิทย์	กมลรัตน์	แจ้งวัฒนะ

ตารางพนักงาน

4104	ระพีพร	มงคลชัย	บางลำพู

Entity Instance (Record)

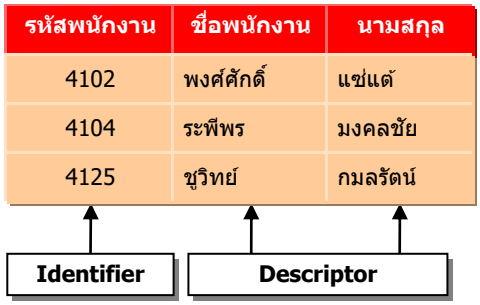
Attribute หมายถึง ข้อมูลที่แสดงรายละเอียดของสิ่งที่อยู่ภายในกลุ่มข้อมูลหรือ Entity เช่น ในกลุ่มข้อมูลพนักงานจะมี รหัสพนักงาน ชื่อ-นามสกุลพนักงาน ที่อยู่ เป็นต้น ข้อมูลหนึ่งๆ ที่อยู่ในกลุ่มข้อมูลอาจเป็นข้อมูลที่มีเพียง Attribute เดียวหรืออาจมีหลายๆ Attribute ก็ได้ เช่น ถ้าพิจารณาข้อมูล "รหัสพนักงาน" จะเห็นว่าไม่สามารถแยกเป็นหลายๆ Attribute ได้เนื่องจากว่าจะทำให้ความหมายของข้อมูลผิดไป แต่ในข้อมูลชื่อพนักงานสามารถที่จะแยกออกเป็น 2 Attribute ได้ เช่น ข้อมูลชื่อพนักงานอาจแยกเป็น "ชื่อพนักงาน" และ "นามสกุล" เป็นต้น ในโปรแกรมจัดการฐานข้อมูลทั่วไปจะเรียก Attribute ว่า **"ฟิลด์ (Field)"** โดยทั่วไปแล้ว สามารถแบ่งฟิลด์ได้เป็น 2 ชนิด คือ

- **Identifier**

เป็นฟิลด์ที่จะทำหน้าที่ในการบอกความแตกต่างของแต่ละรายการในตารางข้อมูล ดังนั้นฟิลด์ข้อมูลที่จะใช้เป็น Identifier จึงควรเป็นฟิลด์ที่ข้อมูลในฟิลด์ไม่มีโอกาสซ้ำกัน (Unique) เช่น ในกลุ่มข้อมูลพนักงานสามารถที่จะใช้ฟิลด์ “รหัสพนักงาน” เป็นตัวบอกความแตกต่างของพนักงานแต่ละคนได้ และจะไม่ใช้ฟิลด์ที่ข้อมูลในฟิลด์มีโอกาสซ้ำกันมาเป็น Identifier เช่น ชื่อพนักงาน

- **Descriptor**

เป็นฟิลด์ที่แสดงรายละเอียดของข้อมูล จากตัวอย่างฟิลด์ที่เป็น Descriptor คือ ชื่อและนามสกุลของพนักงาน



ความสัมพันธ์ (Relationships)

ในระบบฐานข้อมูลเชิงสัมพันธ์นั้น กลุ่มข้อมูลจะถูกสร้างให้อยู่ในรูปของตาราง (Table) ซึ่งข้อมูลที่อยู่ในแต่ละตารางของฐานข้อมูลจะต้องมีส่วนเกี่ยวข้องหรือมีความสัมพันธ์กัน จะเรียกความสัมพันธ์ระหว่างตารางนี้ว่า **"Relationships"** ซึ่งเป็นหัวใจหลักของฐานข้อมูลในการเชื่อมโยงข้อมูลในแต่ละตารางเข้าไว้ด้วยกัน โดยอาศัยฟิลด์ที่มีค่าตรงกันในระหว่างตารางเป็นตัวเชื่อมหรือที่เราเรียกว่า**"Key"** ความสัมพันธ์ระหว่างเรคคอร์ดในแต่ละตารางสามารถแบ่งได้เป็น 3 ประเภทคือ

- ความสัมพันธ์แบบหนึ่งต่อหนึ่ง (one-to-one)
- ความสัมพันธ์แบบหนึ่งต่อกลุ่ม (one-to-many)
- ความสัมพันธ์แบบกลุ่มต่อกลุ่ม (many-to-many)

ความสัมพันธ์แบบหนึ่งต่อหนึ่ง

เป็นความสัมพันธ์ของข้อมูลในลักษณะที่รายการหนึ่งในตารางสามารถเชื่อมความสัมพันธ์ไปยังข้อมูลที่อยู่ในอีกตารางหนึ่งได้เพียงรายการเดียวเท่านั้น โดยปกติ ตารางที่จะมีความสัมพันธ์เป็นแบบหนึ่งต่อหนึ่ง มักจะเป็นตารางที่เป็นข้อมูลชุดเดียวกัน แต่ต้องการจัดเก็บแยกเป็นหลายๆ ตารางเพื่อประโยชน์บางอย่างในการประมวลผลข้อมูล ตัวอย่างเช่น ในการบันทึกรายละเอียดของพนักงานแต่ละคน ผู้ออกแบบฐานข้อมูลอาจออกแบบให้แยกเก็บรายละเอียดของพนักงานเป็น 2 ตาราง ได้แก่ ตารางที่ใช้เก็บข้อมูลทั่วไปของพนักงาน (Employee) กับตารางที่ต้องการเก็บข้อมูลที่เป็นความลับของพนักงาน (Salary) เช่น อัตราเงินเดือน เป็นต้น

รหัสพนักงาน	ชื่อพนักงาน	นามสกุล
4102	พงศ์ศักดิ์	แซ่แต้
4104	ระพีพร	มงคลชัย
4125	ชูวิทย์	กมลรัตน์

ตารางพนักงาน

รหัสพนักงาน	เงินเดือน
4102	7,500
4104	8,500
4125	9,000

ตารางอัตราเงินเดือน

ความสัมพันธ์แบบหนึ่งต่อกลุ่ม

เป็นความสัมพันธ์ของข้อมูลในลักษณะที่รายการหนึ่งในตารางสามารถเชื่อมความสัมพันธ์ไปยังข้อมูลที่อยู่ในอีกตารางหนึ่งได้หลายรายการ ความสัมพันธ์ประเภทนี้จะเป็นความสัมพันธ์ที่พบมากที่สุด จากตัวอย่าง ความสัมพันธ์ระหว่างตาราง 'สมาชิก' กับตาราง 'การเช่า' จัดเป็นแบบ one-to-many กล่าวคือ รายการหนึ่งในตาราง 'สมาชิก' สามารถเชื่อมโยงความสัมพันธ์ไปยังตาราง 'การเช่า' ได้มากกว่าหนึ่งรายการ หรือสมาชิกแต่ละคนสามารถมีการเช่าได้มากกว่าหนึ่งครั้ง

เลขสมาชิก	ชื่อสมาชิก	นามสกุล
C012	วรรณภา	สาครชัย
C062	บรรเจิด	มณีกุล
C073	วิเชียร	สาครชัย
C105	ระพี	ศรีสวัสดิ์

ตารางสมาชิก

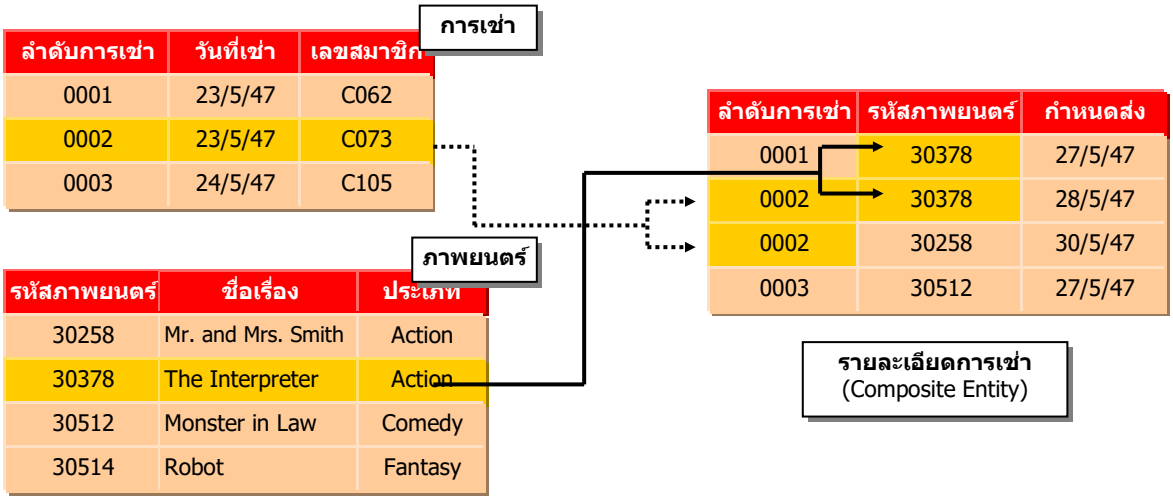
ลำดับการเช่า	วันที่เช่า	เลขสมาชิก
0001	23/5/47	C012
0002	23/5/47	C062
0003	24/5/47	C062
0004	28/5/47	C073

ตารางการเช่า

ความสัมพันธ์แบบกลุ่มต่อกลุ่ม

เป็นความสัมพันธ์ของข้อมูลในลักษณะที่หลายๆ รายการที่อยู่ในตารางหนึ่งสามารถเชื่อมความสัมพันธ์ไปยังข้อมูลที่อยู่ในอีกตารางหนึ่งได้หลาย ๆ รายการ เช่น ความสัมพันธ์ระหว่าง ตาราง 'การเช่า' กับตาราง 'ภาพยนตร์' เป็นแบบ many-to-many กล่าวคือ การเช่าภาพยนตร์แต่ละครั้งสามารถเช่าได้หลายๆ เรื่อง และภาพยนตร์แต่ละเรื่องก็สามารถถูกเช่าได้มากกว่าหนึ่งครั้ง

ในการสร้างความสัมพันธ์แบบ many-to-many นี้เป็นความสัมพันธ์ที่ไม่สามารถเชื่อมกันได้โดยตรง แต่จะต้องมีการสร้างตารางตัวกลางที่ใช้เชื่อมความสัมพันธ์ระหว่างตารางทั้งสอง โดยตารางตัวกลางนี้จะเชื่อมความสัมพันธ์ไปยังตารางทั้งสองในแบบ one-to-many และจะเรียกตารางกลางที่เกิดขึ้นนี้ว่า **Composite Entity**



จากตัวอย่าง ความสัมพันธ์ระหว่างตาราง 'การเช่า' กับตาราง 'วีซีดี' เป็นแบบ **many-to-many** ซึ่งจะต้องสร้างตารางตัวกลางขึ้นมาใหม่ ได้แก่ ตาราง 'รายละเอียดการเช่า' โดยความสัมพันธ์ระหว่างตาราง 'รายละเอียดการเช่า' กับตาราง 'การเช่า' และตาราง 'ภาพยนตร์' จะเป็นแบบ one-to-many จึงอาจกล่าวได้ว่า *ตาราง 'การเช่า' สัมพันธ์กับ ตาราง 'ภาพยนตร์' โดยผ่านทางตาราง 'รายละเอียดการเช่า'*

สำหรับข้อมูลที่จะอยู่ในแต่ละตารางสำหรับกรณีข้างต้น อาจมีข้อสังเกตได้ดังนี้

- ข้อมูลที่จะอยู่ภายในตาราง 'การเช่า' จะเป็นข้อมูลที่เกี่ยวข้องกับการเช่าวีซีดีในแต่ละครั้ง โดยที่ข้อมูลนี้จะเป็นข้อมูลเดียวกันแม้ว่าจะเช่าไปกี่เรื่องก็ตาม ข้อมูลนี้ได้แก่ ลำดับการเช่า วันที่เช่า และสมาชิกที่เช่า
- ข้อมูลที่จะอยู่ภายในตาราง 'รายละเอียดการเช่า' จะเป็นข้อมูลของภาพยนตร์ที่เช่าแต่ละเรื่อง ซึ่งเก็บข้อมูลที่จะเปลี่ยนแปลงไปตามเรื่องที่เช่า ข้อมูลนี้ได้แก่ รหัสภาพยนตร์ที่เช่า และกำหนดส่งคืน
- เนื่องจากตาราง 'รายละเอียดการเช่า' จะต้องมีความสัมพันธ์กับตาราง 'การเช่า' จึงต้องมีข้อมูลตัวหนึ่งที่ใช้ทำหน้าที่เชื่อมต่อไปยังรายการที่อยู่ในตาราง 'การเช่า' ดังนั้น จึงมีข้อมูล *ลำดับการเช่า* ถูกสร้างเพิ่มในตาราง 'รายละเอียดการเช่า' แม้ว่าจะทำให้ซ้ำซ้อนกับในตาราง 'การเช่า' ก็ตาม (รายละเอียดเพิ่มเติมจะกล่าวถึงในเรื่อง Primary Key และ Foreign Key)

Primary Key

Primary Key หมายถึง ฟิลด์ในตารางที่จะทำหน้าที่บอกความแตกต่างของแต่ละรายการในตาราง โดยทั่วไปแล้วจะใช้ฟิลด์ที่มีคุณสมบัติเป็นแบบ Identifier ทำหน้าที่เป็น Primary Key โดยในทุกๆ ตารางจะต้องมีฟิลด์ที่จะทำหน้าที่เป็น Primary Key เสมอ

ตารางสมาชิก

เลขสมาชิก	ชื่อสมาชิก	นามสกุล
C012	วรรณภา	สาครชัย
C062	บรรเจิด	มณีกุล
C073	วิเชียร	สาครชัย
C105	ระพี	ศรีสวัสดิ์

Primary Key

ในการค้นหาฟิลด์ที่จะทำหน้าที่เป็น Primary Key ในตารางใดๆ นั้น บางครั้งอาจไม่มีฟิลด์ใดเลยที่สามารถใช้บอกความแตกต่างของแต่ละรายการได้ ในกรณีนี้อาจมีวิธีแก้ปัญหาได้ 2 วิธีคือ

- **Composite Key** หมายถึง การกำหนดให้ฟิลด์มากกว่าหนึ่งฟิลด์ทำหน้าที่เป็น Primary Key ร่วมกัน โดยมีข้อกำหนดว่า ถ้าข้อมูลในฟิลด์ใดฟิลด์หนึ่งซ้ำกัน ข้อมูลที่อยู่ในอีกฟิลด์จะต้องไม่ซ้ำกันด้วย โดยทั่วไป มักจะพบ Composite Key ในตารางที่เป็นตารางกลาง (Composite Entity) ในความสัมพันธ์แบบ many-to-many
- **Surrogate Key** หมายถึง การสร้างฟิลด์ใหม่ขึ้นมาโดยฟิลด์นั้นจะไม่ได้ใช้จัดเก็บข้อมูลของตารางแต่อย่างใด เพียงแต่เป็นฟิลด์ที่จะทำหน้าที่ในการบอกความแตกต่างเท่านั้น โดยปกติฟิลด์ประเภทนี้มักจะกำหนดให้เป็นลักษณะของตัวเลขเรียงต่อกันโดยอัตโนมัติ (Auto Number)

ลำดับการเช่า	รหัสภาพยนตร์	กำหนดส่ง
0001	30378	27/5/47
0002	30378	28/5/47
0002	30258	30/5/47
0003	30512	27/5/47

Composite Key

รหัสรายการ	ลำดับการเช่า	รหัสภาพยนตร์	กำหนดส่ง
1	0001	30378	27/5/47
2	0002	30378	28/5/47
3	0002	30258	30/5/47
4	0003	30512	27/5/47

Surrogate Key

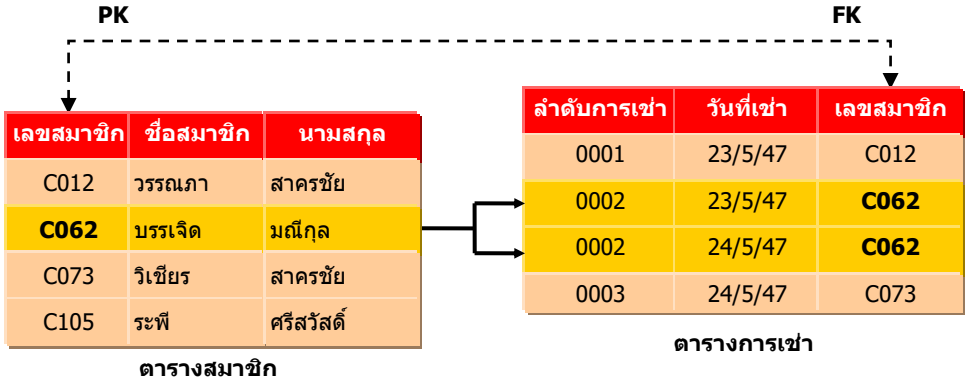
Primary Key นอกจากจะทำหน้าที่บอกความแตกต่างของแต่ละรายการในตารางแล้ว ข้อมูลที่อยู่ในฟิลด์ที่เป็น Primary Key ยังทำหน้าที่ในการเชื่อมความสัมพันธ์ไปยังตารางอื่นอีกด้วย โดยเป็นไปในลักษณะของตัวชี้ หรือ *Indexed* ที่ชี้ไปยังรายการที่มีความสัมพันธ์ในตารางอื่นๆ

คุณสมบัติของ Primary Key

- ข้อมูลที่อยู่ในฟิลด์ที่จะทำหน้าที่เป็น Primary Key จะต้องไม่มีข้อมูลที่ซ้ำกันในแต่ละรายการ ถึงแม้ว่าจะลบข้อมูลเดิมออกจากฐานข้อมูลแล้วก็ตาม
- ข้อมูลที่อยู่ใน Primary Key ควรมีขนาดของข้อมูลเล็กที่สุดเท่าที่เป็นไปได้ เนื่องจากข้อมูลนี้จะต้องไปอยู่ในหลายๆ ตาราง (เพื่อใช้เชื่อมความสัมพันธ์) โดยปกติแล้วมักกำหนดให้เป็นข้อมูลประเภทตัวเลขหรือเป็นตัวอักษรที่มีไม่กี่ตัวอักษร
- ข้อมูลที่อยู่ในฟิลด์ Primary Key ไม่สามารถกำหนดเป็นค่าว่าง (Null Value) ได้

Foreign Key

เมื่อมีการเชื่อมความสัมพันธ์ระหว่างตารางความสัมพันธ์จะถูกเชื่อมโยงจาก **Primary Key (PK)** ที่อยู่ในตารางหนึ่งไปยังฟิลด์สัมพันธ์ที่อยู่ในอีกตารางหนึ่ง โดยจะเรียกฟิลด์สัมพันธ์นี้ว่า **Foreign Key (FK)**



จากตัวอย่าง ความสัมพันธ์ระหว่าง ตาราง ‘สมาชิก’ กับ ตาราง ‘การเช่า’ เป็นแบบ one-to-many โดยผ่านทางฟิลด์ ‘เลขสมาชิก’

***สำหรับตารางที่เชื่อมความสัมพันธ์แบบ one-to-one จะไม่มี Foreign Key เพราะเป็นการเชื่อมความสัมพันธ์ระหว่าง Primary Key ของตารางหนึ่งไปยัง Primary Key ของอีกตารางหนึ่ง

คุณสมบัติของ Foreign Key

- ข้อมูลที่อยู่ใน Foreign Key สามารถที่จะมีข้อมูลที่ซ้ำกันได้ในแต่ละรายการ
- ข้อมูลที่อยู่ใน Foreign Key จะต้องเป็นข้อมูลที่มีประเภทของข้อมูลและขนาดของข้อมูลเหมือนกับ Primary Key ที่อยู่ในตารางที่มีความสัมพันธ์กัน จากตัวอย่างความสัมพันธ์ระหว่าง ตาราง ‘สมาชิก’ กับ ตาราง ‘การเช่า’ ถ้า *เลขสมาชิก* ในตาราง ‘สมาชิก’ (Primary Key) มีการกำหนดประเภทของข้อมูลเป็น ตัวอักษรขนาด 4 ตัว *เลขสมาชิก* ในตาราง ‘การเช่า’ (Foreign Key) ก็ต้องมีประเภทของข้อมูลเป็น ตัวอักษรขนาด 4 ตัวด้วยเช่นกัน
- ในกรณีที่ตารางหนึ่งมีความสัมพันธ์กับตารางอื่นๆ มากกว่า 1 ตารางจะส่งผลให้ตารางนั้นมี ฟิลด์ที่ทำหน้าที่เป็น Foreign Key มากกว่า 1 ฟิลด์ก็ได้